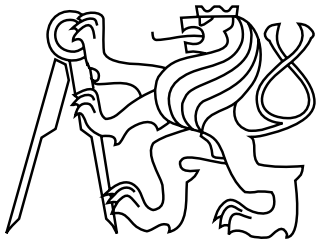
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA STAVEBNÍ

DIPLOMOVÁ PRÁCE

PRAHA 2013 Eva LINHARTOVÁ

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA STAVEBNÍ
OBOR GEODÉZIE A KARTOGRAFIE



DIPLOMOVÁ PRÁCE

TOPOLOGICKÉ OPERACE VE VYBRANÝCH SOFTWARE A GEODATABÁZÍCH

Vedoucí práce: Ing. Jiří CAJTHAML, Ph.D.
Katedra mapování a kartografie

leden 2013

Eva LINHARTOVÁ

ZDE VLOŽIT LIST ZADÁNÍ

Z důvodu správného číslování stránek

ABSTRAKT

Práce se věnuje popisu prostorových databází Oracle Spatial a PostGIS. Popisuje práci v těchto databázích a porovnává je. Pomocí několika prostorových dotazů srovnává rychlost a výsledky dotazů. Výsledky jsou na závěr vizualizovány a porovnány v prostředí ArcGIS. V práci je zahrnut popis možnosti uložení prostorových dat do topologických struktur v databázích. Prostorová data jsou názorně převedena na topologicky strukturovaná data.

KLÍČOVÁ SLOVA

prostorová data, prostorová databáze, ArcGIS, PostGIS, PostGIS Topology, Oracle Spatial, okřídlená hrana, NAA stuktura

ABSTRACT

The diploma thesis is focused on description of spatial databases Oracle Spatial and PostGIS. It describes work in these spatial databases and then compares them. The comparison focuses on their speed and results by using several spatial queries. At the end the results of these queries are visualised and checked in ArcGIS software. The thesis also includes decription of topology structures of data in both spatial databases. Spatial data are imported into these structures.

KEYWORDS

spatial data, spatial database, ArcGIS, PostGIS, PostGIS Topology, Oracle, Spatial, Winged Edge, NAA structure

PROHLÁŠENÍ

Prohlašuji, že diplomovou práci na téma „Topologické operace ve vybraných software a geodatabázích“ jsem vypracovala samostatně. Použitou literaturu a podkladové materiály uvádím v seznamu zdrojů.

V Praze dne ....

(podpis autora)

PODĚKOVÁNÍ

Chtěla bych poděkovat vedoucímu práce Jiřímu Cajthamlovi, Ph.D. za připomínky a pomoc při zpracování této práce. Dále bych chtěla poděkovat Ing. Petru Šebestovi a společnosti T–Mapy spol. s r.o. za odborné konzultace a poskytnutí dat, na kterých byly dotazy zkoušeny. V neposlední řadě bych ráda poděkovala Ing. Martinu Landovi za konzultace ohledně práce v databázi PostGIS.

Obsah

Úvod **9**

1 Prostorová data **11**
1.1 Prostorové databáze . . . 11
1.1.1 Historie . . . 12
1.2 Uložení prostorových dat . . . 13
1.2.1 Souborový způsob uložení dat . . . 13
1.2.2 Databázový způsob uložení dat . . . 14

2 ArcGIS **15**
2.1 Historie . . . 15
2.2 Struktura dat . . . 17
2.2.1 Shapefile . . . 17
2.2.2 Personal geodatabase . . . 17
2.2.3 File geodatabase . . . 17
2.2.4 ArcSDE geodatabase . . . 17
2.3 Topologie v ArcGIS . . . 18

3 PostGIS **19**
3.1 Historie . . . 19
3.2 Simple Features for SQL Specification . . . 20
3.2.1 Validace . . . 21
3.2.2 Prostorové indexy . . . 24
3.2.3 Prostorové vztahy . . . 25
3.2.4 Prostorové operátory . . . 28
3.2.5 Datový typ Geometry . . . 30
3.3 PostGIS Topology . . . 32
3.3.1 Datový typ TopoGeometry . . . 34
3.3.2 Topologické tabulky . . . 34
3.4 Práce s daty . . . 38
3.4.1 Založení schéma a nastavení cesty vyhledávání . . . 38
3.4.2 Import dat . . . 38
3.4.3 Validace dat . . . 39
3.4.4 Vytvoření prostorových indexů . . . 40
3.4.5 Vytvoření topologicky strukturovaných dat . . . 40

4 Oracle Spatial **42**
4.1 Historie . . . 42
4.2 Objektově–relační model . . . 43
4.2.1 Datový typ SDO_GEOMETRY . . . 43
4.2.2 Dotazovací model . . . 44
4.2.3 Validace dat . . . 44
4.2.4 Prostorové indexy . . . 46
4.2.5 Prostorové vztahy . . . 47

4.2.6 Prostorové procedury a funkce . . . 50
4.3 Koncept topologického datového modelu . . . 52
4.3.1 Datový typ SDO_TOPO_GEOMETRY . . . 53
4.3.2 Tabulky topologického modelu . . . 54
4.3.3 Editace topologie . . . 58
4.4 Postup práce s daty . . . 59
4.4.1 Načtení dat . . . 59
4.4.2 Aktualizace metadat . . . 61
4.4.3 Validace . . . 61
4.4.4 Vytvoření prostorových indexů . . . 62
4.4.5 Převedení prostorových dat do topologické struktury . . . 62

5 Použitá data **65**
5.1 Historie . . . 65
5.2 Digitální mapa Prahy . . . 66
5.2.1 Mapa technického vyžití území . . . 67

6 Vybrané Topologické operace **70**
6.1 ArcGIS . . . 70
6.2 PostGIS . . . 71
6.2.1 Hledání děr v polygonu . . . 71
6.2.2 Hledání duplicitních linií . . . 72
6.2.3 Překrývající se polygony . . . 73
6.2.4 Volné konce linií . . . 74
6.3 Spatial . . . 75
6.3.1 Hledání děr v polygonu . . . 75
6.3.2 Hledání duplicitních linií . . . 77
6.3.3 Hledání překrývajících se polygonů . . . 77
6.3.4 Volné konce linií . . . 78
6.4 Srovnání . . . 78
6.4.1 PostGIS . . . 78
6.4.2 Spatial . . . 79
6.5 Porovnání validních a nevalidních dat . . . 81
6.6 Možnosti prezentace . . . 82
6.6.1 PostGIS . . . 82
6.6.2 Spatial . . . 83
6.7 Možnosti topologické editace . . . 85
6.7.1 PostGIS . . . 85
6.7.2 Spatial . . . 85

Závěr **86**

Použité zdroje **87**

Seznam příloh **93**

A postgis.sql **94**

B postgis_topology.sql **97**

C spatial.sql **98**

D get_geom_set.sql **104**

E dangles.sql **105**

F spatial_topology.sql **107**

Úvod

Při hledání možného téma diplomové práce jsem oslovila společnost T–MAPY s r.o., která mi rovněž poskytla reálná data, na kterých jsou prostorové dotazy aplikovány. Při výběru téma jsem přihlédla i na možnou návaznost na bakalářskou práci na téma Topologie v GIS z roku 2011.

Práce přímo navazuje na bakalářskou práci na téma Topologie v GIS z roku 2011. Ta se zabývala vymezením pojmu topologie. Rovněž jsou v ní popsány vektorové datové modely, zejména topologický model, ale zmíněny jsou i špagetový a hierarchický model. Práce dále obsahuje stručný popis vektorových datových struktur. Praktická část práce byla zaměřena na práci s topologií v prostředí ArcGIS. Z tohoto důvodu v diplomové práci podrobně nepopisuji práci a možnosti softwaru ArcGIS.

V diplomové práci jsem se zaměřila na práci s prostorovými daty v prostorových databázích. Pro porovnání byla volena komerční prostorová databáze Oracle Spatial a open source prostorová databáze PostGIS. Obě tyto databáze umožňují uložení prostorových dat v objektově–relačním modelu a topologickém modelu.

Práce je rozdělena do šesti kapitol. V první kapitole jsou popsána prostorová data a možnosti jejich uložení. Je zde definována prostorová databáze a popsán stručný vývoj databází. Kapitola shrnuje výhody a nevýhody souborového a databázového uložení dat.

V druhé kapitole je popsán ArcGIS produkt společnosti ESRI, jeho historie a způsoby uložení dat. Jak už bylo napsáno výše, práce neobsahuje podrobný popis topologie v ArcGIS. Je uveden jen stručný souhrn postupu při práci s topologií.

V třetí kapitole jsem se věnovala popisu prostorové databáze PostGIS a jejímu rozšíření PostGIS Topology. Je zde popsaná stručná historie. Dále obsahuje popis implementace specifikace Simple Features definované OGC (OpenGIS Consortium). V kapitole je rovněž definován datový typ Geometry, validace, indexování. Uvedeny jsou topologické vztahy podle OGC a možnost využití DE-9IM (rozšířeného 9-ti průsečíkového modelu). Kapitola zahrnuje základní topologické operace AND, MINUS, OR, XOR. Topologický model uložení PostGIS Topology je popsán spolu s topologickými tabulkami a datovým typem TopoGeometry. Závěr kapitoly se soustředí na praktickou část práce s prostorovými daty v PostGIS a vytvoření topologické struktury dat v PostGIS Topology.

Ve čtvrté kapitole je popsaná prostorová databáze Oracle Spatial. Stručně je zde uvedena historie. Kapitola je věnována popisu datového typu SDO_GEOMETRY, způsobu indexování, validaci a dotazování. Obsahuje popis pojmenovaných topologických vztahů a jsou zde uvedeny základní topologické operace AND, MINUS, OR, XOR. V neposlední řadě popisuje koncept topologického modelu uložení dat ve Spatial a datový typ SDO_TOPO_GEOMETRY. Závěr kapitoly je věnován práci ve Spatial a je doplněn o praktické ukázky.

Pátá kapitola se vztahuje k datům poskytnutým společností T–MAPY s r.o.. Jedná se o data Mapy technického využití území (MTVU). Poslední šestá kapitola obsahuje konkrétní topologické operace provedené v jednotlivých databázích. Tyto

operace byly porovnány mezi sebou na základě rychlosti provedení dotazu a výsledku.

Cílem práce je shrnout možnosti ukládání prostorových dat a práce s nimi pro topologické operace. Během zpracování jsem se poprvé seznámila s databázovým systémem Oracle Spatial. Z tohoto důvodu může práce být prospěšná pro začínající spíše než pro pokročilé uživatele těchto databází.

1 Prostorová data

„Data o poloze, tvaru a vztazích mezi jevy reálného světa, vyjádřená zpravidla ve formě souřadnic a topologie. “ [12]

„Prostorová data (Spatial Data) jsou data, která obsahují formální referenci na konkrétní místa v prostoru a zároveň musí být pro danou úroveň rozlišení těchto dat uvedené polohy míst v prostoru známé.“ [13]

GIS je často používán k ukládání, získávání a vykreslení prostorových dat. Typy prostorových dat mohou být uloženy pomocí prostorového vkládání dat z CAD a CAM systémů. Namísto pracování na objektech v zeměpisném měřítku CAD/CAM systémy pracují s většími měřítky například v automobilovém průmyslu nebo ve výrobě desek plošných strojů. [8]

Pod pojem prostorová data řadíme všechny údaje, které se vztahují k prvku, jehož výskyt má dvojrozměrný, trojrozměrný nebo n-rozměrný charakter. Zvláštním případem prostorových dat jsou geodata, která se vztahují k zemskému povrchu. Prostorová data v digitální podobě jsou reprezentována dvěma způsoby: rastrově, nebo vektorově. [4]

Objekty jsou charakterizovány dvěma skupinami parametrů prostorových dat:

- Geometrické – popisují druh použitého symbolu spolu s informací o poloze a souřadnicovém systému.
- Negeometrické – obsahují informaci o typu vztahu k okolí a atributy, které popisují vlastnosti prvků a často i časové údaje o době pořízení dat.

Rastrová reprezentace – zaměřuje se na lokalitu jako na celek. Používá se pro spojitě se měnící jevy, jako je například DMR (Digitální model reliéfu). Základní prvkem je buňka (cell), buňky (nazývané také jako pixely) jsou organizovány do mozaiky. Topologie v rastrovém modelu je dána implicitně, není nutné ji ukládat jako pro vektorový model.

Vektorová reprezentace – opírá se o vyjádření geometrie objektů přes jejich lineární charakteristiky. Typy prostorových dat jsou často označovány jako objekty a jsou jimi body, linie a polygony.

1.1 Prostorové databáze

„DBMS (Database Management System) nebo-li Systém řízení báze dat (zkracováno SŘBD) je softwarový systém, který uživatelům umožňuje definovat, vytvářet a udržovat databázi a také poskytuje řízený přístup k databázi.“ [1]

Databáze jsou dnes tak nedílnou součástí života, že si jejich používání ani neuvědomujeme. V případě prostorových dat můžeme použít rozšíření, která umožňují

s těmito daty dále pracovat. Tato rozšíření můžeme nazývat prostorovými databázemi.

1.1.1 Historie

DBMS má své kořeny v 60. letech v projektu přistání Apolla na měsíci. V té době nebyl systém schopný zpracovat a spravovat tak obrovské množství dat, jaké projekt obnášel. V počátcích 80. let byly prostorové databáze vyvinuty, aby spojovaly počítačové mapování s tradičními databázemi. Vývoj jednotlivých prostorových databází je popsán v kapitolách vztahujících se ke konkrétním databázovým systémům. Vývoj DBMS je možné rozdělit do tří fází – generací. [1]

První generace

V polovině 60. let se objevily dva DBMS. Jedním byl IMS (Information Management System) firmy IBM a druhým byl IDS (Integrated Data Store) firmy General Electric.

IMS uchovával záznam pomocí stromové invertované struktury. Ta umožňovala použití magnetické pásky. Tento DBMS je také znám jako ***hierarchický***. Jedná se o jeden z prvních komerčních DBMS, je stále užívaný vetšinou velkých střediskových instalací.

IDS byl vyvinut, aby vyřešil reprezentaci komplexnějších datových struktur, což nebylo možné pomocí hierarchických struktur, dalším důvodem bylo vytvoření databázového standardu. Tento DBMS se stal znám jako ***síťový***. V síťových systémech jsou data reprezentována jako kolekce záznamů a relace jako množiny (jeden vlastník, více členů). Nejoblíbenějším síťovým DBMS je IDMS/R firmy Computer Associate. Nevýhody těchto přístupů:

- minimální datová nezávislost, aplikace nejsou chráněny před změnami formátu dat,
- potřeba složitých programů i pro jednoduché dotazy založené na navigačním přístupu pouze k jednomu záznamu současně,
- neexistuje široce přijímaný teoretický základ.

Druhá generace

Jako reakci na nevýhody hierarchického a síťového přístupu vydal roce 1970 E.F.Codd z IBM Research Laboratory pojednání o ***relačním*** datovém modelu. Tento model reprezentuje data ve formě tabulek. První komerční produkty se objevily koncem 70. a začátkem 80. let. Klíčovým krokem ve vývoji byl SQL, standardní dotazovací jazyk v relačních databázích. Nevýhodou však je omezená modelovací schopnost. Mnoho výzkumů se zabývalo řešením tohoto problému. V roce 1976 Chen představil ***entitně–relační*** model, který je přijímanou technikou pro návrh databází.

Třetí generace

Během 90. let se jako reakce na vzrůstající složitost databází objevily dva nové systémy. ***Objektově orientovaný*** (OODBMS) a ***objektově–relační*** model (ORDBMS). Vznik internetu a třívrstvé architektury v 90. letech podnítil požadavek na možnost integrace podnikové databáze s webovým prostředím. Vývoj XML (eXtensible Markup Language) koncem 90. let měl velký vliv na integrace databází, grafických rozhraní, databázových systémů, atd.

1.2 Uložení prostorových dat

1.2.1 Souborový způsob uložení dat

Jedná se o starší způsob ukládání, který byl uplatňován v počátcích zpracování dat okolo 60. let 20. století. Souborově orientované systémy předcházely vývoji databází, jednalo se o snahu počítačově zpracovat ruční kartotékový systém. Tento systém byl vytvořen z důvodu potřeby průmyslu efektivněji zpracovávat data. [1]

Existuje dvojí způsob souborového uložení. Buď jsou to formáty, které uchovávají vše v jediném souboru, mezi ně patří soubory programu Microstation společnosti Bentley Systems $*.dgn$, $*.dxf$, nebo formáty skládající se z více souborů. Jedním z příkladů takového formátu je vektorový formát od společnosti ESRI *shapefile*, který je více popsán v kapitole 2.2.1.

Výhody:

- *Přenosnost.* Soubory jsou lehce přenosné a podporované většinou software.
- *Jednoduchost.* Pro většinu uživatelů je jednodušší.
- *Dostupnost.* Úřady poskytují prostorová data v souborovém formátu.

Nevýhody:

- *Závislost dat.* Známa také jako *programově – datová závislost.* Vlastní aplikační programy mají vlastní zpracovatelské rutiny a strukturu, na níž jsou založeny a která je v nich vnořena. Změny existující struktury jsou obtížné, obvykle zahrnují napsání software pro převod souboru ze staré struktury do nové, což může být časově náročné a zdrojem chyb.
- *Duplikace dat.* Vede k plýtvání a vyžaduje úložný prostor. Duplikaci se lze vyhnout sdílením souborů. Může vést ke ztrátě integrity dat, data jsou nekonzistentní, protože formáty souborů nemusí být slučitelné nebo si hodnoty dat neodpovídají (nebo obojí).
- *Oddělení a izolace dat.* Izolovaná data v oddělených souborech je obtížnější zpřístupnit v okamžiku, kdy mají být k dispozici. Někdy je nutné synchronizovat zpracování dvou nebo více souborů pro určitou zprávu, aby byla zajištěna správná extrakce dat. To může být obtížné, zvláště pokud jsou formáty souborů nekonzistentní.

- *Omezené sdílení dat.* Každé oddělení má vlastní množinu souborů. Může vést ke ztrátě potenciální informace, což se ukáže až při přístupu ke všem datům.
- *Přístup jediného uživatele.* Uživatelé přistupují přímo k obsahu souboru, což znemožňuje současný přístup více než jednoho uživatele.

1.2.2 Databázový způsob uložení dat

„*Databáze je sdílená kolekce logicky souvisejících dat (a popisu těchto dat), navržená pro plnění informačních potřeb organizace.*“ [1]

Výhody:

- *Kontrola redundance dat.* Databázový přístup sice redundanci neodstraňuje zcela, ale eliminuje ji, kdekoliv je to možné. Duplikovat datové položky je nutné pro modelování relací nebo pro zvýšení výkonu.
- *Konzistence dat.* Výskyt nekonzistence se redukuje eliminací a kontrolou redundance. Pokud je zaznamenána datová položka v databázi více jak jednou, může systém zajistit, aby kopie dat byly udržovány konzistentní. Mnoho současných databází automaticky tento typ konzistence nezajišťuje.
- *Sdílení dat.* Soubory obvykle patří lidem nebo oddělením, ale databáze patří celé organizaci a lze ji sdílet se všemi uživateli.
- *Zlepšení integrace dat.* Integrita databáze je vyjádřena jejím omezením a jejími pravidly konzistence. Omezení se mohou vztahovat k jednomu záznamu, nebo k vztahům mezi záznamy.
- *Zlepšení výkonu pomocí nezávislosti dat.* Oddělením popisu dat od aplikací činí databáze aplikace odolnými proti změnám v popisu dat. Poskytování nezávislosti dat usnadňuje údržbu databázových aplikací.

Nevýhody:

- *Složitost.* Všichni uživatelé musí rozumět funkčnosti DBMS, aby ho mohli plně využít.
- *Náklady na DBMS.* DBMS pro střediskové počítače obsluhující stovky uživatelů, mohou být velmi nákladné.
- *Náklady přechodu.* Součástí těchto nákladů je i školení zaměstnanců a zaměstnání specialistů.
- *Výkon.* DBMS je napsán pro obecnější účely, než je souborový systém.
- *Vnější vliv selhání.* Uživatelé spoléhají na dostupnost DBMS, v případě selhání kterékoli komponenty může nastat zastavení operací až do opravy.

2 ArcGIS

„Systém ArcGIS firmy Esri tvoří řada škálovatelných produktů určených pro kompletní nasazení GIS na jakékoli úrovni. Součástí ArcGIS jsou desktopové, serverové i vývojářské produkty, nechybí ani řešení pro mobilní zařízení a specializované nadstavby.“ [16]

ArcGIS zahrnuje následující Windows desktop software:

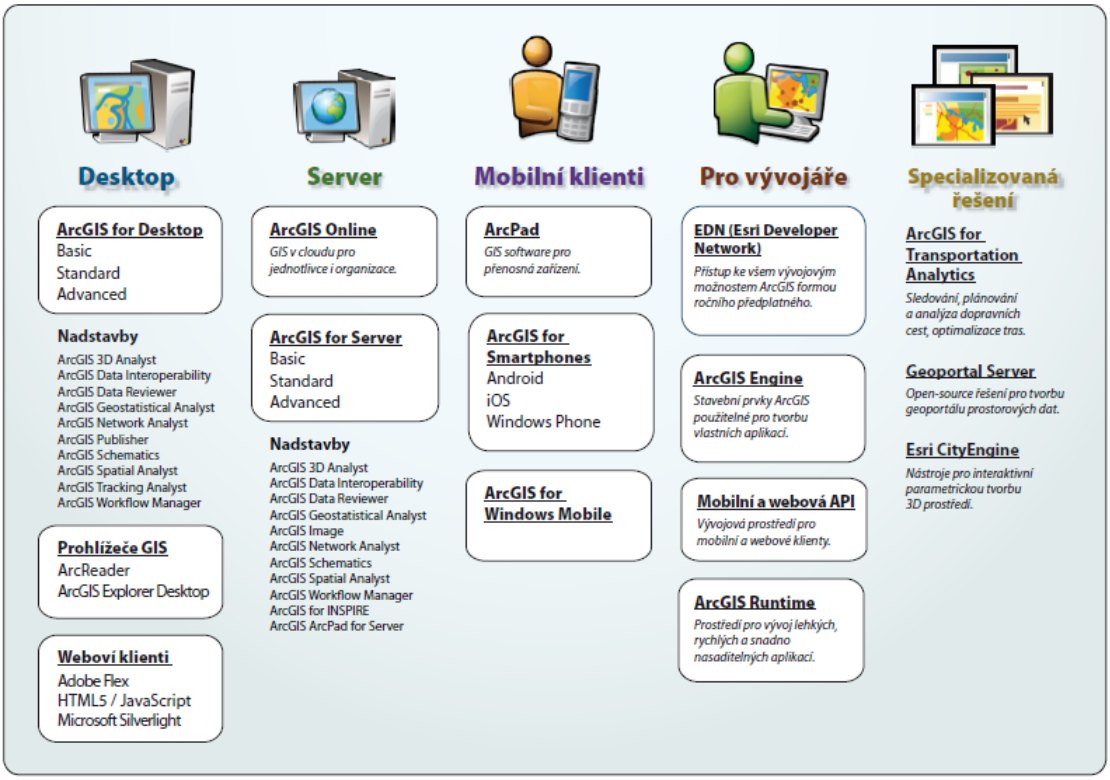
- ArcReader je jednoduchý prohlížeč map vytvořených pomocí ArcGIS produktů. Je volně stažitelný.
- ArcGIS for Desktop je poskytován ve třech licencích:
  - ***Basic*** původně známa jako *ArcView* – slouží především k zobrazování, analýze a tvorbě mapových výstupů. Obsahuje základní nástroje pro tvorbu, správu a editaci dat. Je tvořen sadou ArcMap, ArcCatalog, ArcScene, ArcGlobe, ModelBuilder a okno ArcToolbox.
  - ***Standard*** původně známa jako *ArcEditor* – obsahuje všechny funkce ArcView, hlavním přínosem je pokročilými způsoby vytvářet nová data, kontrolovat kvalitu a přesnost, pomocí nástrojů editovat a spravovat shapefile a geodatabáze.
  - ***Advanced*** původně známa jako *ArcInfo* – obsahuje funkcionalitu předchozích licencí, navíc přináší škálu analytických a kartografických funkcí pro maximální využití potenciálu GIS.

Produkty ArcGIS for Desktop jsou tvořeny aplikacemi ArcMap, ArcCatalog. Pro správu a analýzu slouží soubor nástrojů ArcToolbox umístěný v uživatelském rozhraní. V případě potřeby může být ArcGIS přizpůsoben potřebám pomocí programovacího prostředí ModelBuilder. Pro náročnější úpravy lze prostřednictvím programovacího jazyka Python přistupovat k funkcím ArcGIS. Funkcionalita produktů může být rozšířena přidáním vyvinutých nadstaveb, nebo si pomocí komponent ArcObjects lze vytvořit nadstavby vlastní. Architektura ArcGIS je znázorněna na obr. 2.1. [16]

2.1 Historie

V roce 1969 současný prezident ESRI Jack Dangermond a jeho žena Jane založili Enviromental Systems Research Institute (ESRI) v Redlands v Kalifornii jako konzultační firmu. Prvotním úkolem ESRI byla organizace a analýza geografických informací jako pomoc při územním plánování a resource managerům činit informovaná ekologická rozhodnutí.

V 70. letech San Diego v Kalifornii vybralo firmu ESRI pro vývoj POIS (Polygon Information Overlay System), tím udělala první krok k vytváření Geografického Informačnímu Systému (GIS). K zefektivnění projektů potřebovala firma ESRI způsob, jak automatizovat manuální mapovací postupy. ARC/INFO je první komerční



Obr. 2.1: Architektura ArcGIS, převzato z [16]

GIS, byl vydán v roce 1982. Je kombinací počítačového zobrazení geoprvků, jako jsou body, linie a polygony, s databází pro přidělení atributů k těmto geoprvkům.

Vývoj rychlejších a levnějších počítačů, síťová zpracování, elektronické publikování dat a nové techniky pro zpracování dat, jako je GPS, podnítil v roce 1990 rychlý rozvoj ESRI. Prvním ESRI řešením pro stolní počítače je ArcView, toto řešení otevřelo možnosti GIS pro zcela novou skupinu uživatelů. Ke konci roku 1990 začalo přepracování ARC/INFO pro rozvoj modulární GIS platformy, která by fungovala jak na ploše stolního počítače, tak v rámci celého podniku. Výsledkem byl ArcGIS. Koncem roku 1999 byla vydána verze ArcGIS 8.0, která kombinovala vizuální uživatelské rozhraní ArcView s ARC/INFO.

V roce 1999, povzbuzována myšlenkou zástupce spotřebitelů Ralphem Naderem, zahájila společnost ESRI ve spolupráci s Národní Geografickou Společností GIS Day. Následovaly verze 8.0.1, 8.1, 8.2. V roce 2002 byla přestavena verze ArcGIS 8.3, která se vyznačovala přidáním topologie do geodatabáze. V roce 2004 byl vydán ArcGIS 9, vylepšená desktop platforma navýšená o vývojové prostředí a serverové platformy. Následovaly verze 9.1, 9.2, 9.3, 9.3.1. [14]

V roce 2010 byla vydaná nová řada ArcGIS 10. Aktuální verzí je ArcGIS 10.1, která vyšla v červnu roku 2012 a nabízí inovace ve všech oblastech využití GIS. Vývoj byl zaměřen především na zlepšení komfortu a rychlosti ovládání.

2.2 Struktura dat

Data v prostředí ArcGIS mohou být uložena několika způsoby. Jedním z nich je formát Shapefile, který byl vyvinut firmou ESRI pro datovou interoperabilitu s ostatními software produkty. Dalším způsobem ukládání dat může být geodatabáze. Geodatabází rozumíme prostorovou databázi navrženou společností ESRI pro správu geografických dat, jak vektorových, tak i rastrových.

2.2.1 Shapefile

Shapefile ukládá netopologickou geometrii a atributy pro každý prvek. Právě proto, že neukládá topologii, umožňuje rychlejší vykreslení a editaci oproti ostatním datovým zdrojům. Každý atributový záznam je asociován pouze s jediným prvkem. Shapefile se skládá z povinných a doplňkových souborů.

Povinné soubory:
$*.shp$ hlavní soubor obsahuje geometrii (prostorová data),
$*.shx$ indexový soubor propojuje hlavní soubor s atributy,
$*.dbf$ databázová tabulka dBase obsahuje atributy prvků.

Nepovinné soubory: $*.prj$, $*.sbn$, $*.sbx$, $*.xml$, apod.

2.2.2 Personal geodatabase

Jedná se o originální datový formát pro geodatabáze ArcGIS. Poprvé se tato geodatabáze objevila v ArcGIS 8.0. Data můžou být prohlížena více uživateli najednou, ale editaci může provádět pouze jeden uživatel. Používá formát uložení a správy dat .mdb od Microsoft Access. Geodatabáze je velikostně omezena 2 GB, efektivní limit je stanoven 250 – 500 MB na jeden soubor .mdb. Použití je vhodné pro správu malých, nebo středních datových souborů. Je podporována pouze v operačních systémech Microsoft Windows. [17]

2.2.3 File geodatabase

Je novějším typem geodatabáze, která je používaná od ArcGIS 9.2. Je uložena jako adresář na disku v souborovém systému. Každý soubor může dosahovat velikosti 1 TB. Podporuje více uživatelů, kteří mohou do geodatabáze nahlížet v jednom okamžiku než personal geodatabáze, ale editovat ji může opět jenom jediný uživatel. Tento formát .gdb může být používán pouze pro práci v ArcGIS, ale oproti předchozí geodatabázi jej lze používat na různých operačních systémech. [17]

2.2.4 ArcSDE geodatabase

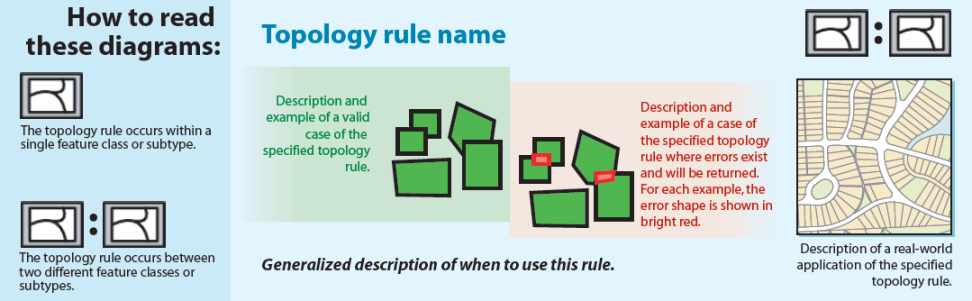
Je klient/server technologie (tzv. middleware) používaná pro ukládání a správu prostorových dat v prostředí klasických RDBMS. Umožňuje čtení a editaci více

uživatelů najednou. Tato geodatabáze není volně dostupná a její limit záleží na limitu použitého DBMS. Umožňuje spravovat objemnou geografickou databázi. [15]

2.3 Topologie v ArcGIS

„Topologie je obor matematiky, který se zabývá popisem a analýzou prostorových vztahů mezi goemetrickými objekty. Zkoumá geometrické vlastnosti, jež jsou pro určité druhy transfromací invariantní, jako je třeba roztažení nebo ohýbání.“[4]

Kontrola topologie probíhá prostřednictvím topologických pravidel definovaných ESRI. Seznam topologických pravidel je dostupný na plakátu [18].



Obr. 2.2: Způsob prezentace topologických pravidel na plakátu ESRI, převzato z [18]

Topologii lze vytvořit pouze v geodatabázi, kde jsou uložena data, nad nimiž má být vytvořena. Během procesu validace jsou kontrolována zvolená topologická pravidla včetně míst, kde byla porušena.

Shrnutí postupu práce s topologií:

- Vytvoření sad tříd prvků (features classes) v rámci datové sady (features dataset) geodatabáze.
- Import dat do třídy prvků.
- Vytvoření topologie nad třídou prvků pomocí ArcCatalogu, nebo nástrojů v ArcMapu. V tomto kroku jsou volena topologická pravidla.
- Validace topologie.
- Přidání topologie do ArcMapu a nastavení vlastností pro její vizualizaci.
- Použití editačních nástrojů pro identifikaci a opravu chyb.

Podrobnější popis vytváření topologie a možnosti editace jsou popsány v [7].

3 PostGIS

PostGIS je geodatabáze, která je nadstavbou objektově–relační databázi PostgreSQL. Přidává podporu pro geografické objekt a umožňuje provádět prostorové dotazování v jazyku SQL. PostGIS implementuje *Simple Features for SQL Specification* uvedené v [30], [31] organizace Open Geospatial Consortium (OGC).

PostGIS je převážně vyvíjen firmou Refractions Research Inc. jako open source software pod licencí GNU General Public License.

3.1 Historie

PostgreSQL, původně nazýván Postgres, byl vytvořen na univerzitě Berkeley v Kalifornii týmem vedeným profesorem Michaelem Stonebrakerem. Jak napovídá název, navazuje na předchůdce Ingres (INteractive Graphics and REtrieval System), který byl vyvíjen v letech 1977–1994 a nyní je majetkem společnosti Computer Associates. Postgres byl vyvíjen v letech 1986–1994, rozšiřoval obzory v databázích zkoumáním objektově–relační technologie.

Společnost Illustra převzala zdrojový kód a vytvořila z něho komerční produkt, později byla odkoupena společností Informix Software Inc. a začleněna do databázového serveru INFORMIX – Universal Server. V roce 2001 byla Informix Software Inc. odkoupena společností IBM.

V roce 1995 dva studenti doktorského studia z Stonebrakerovy laboratoře Andrew Yu a Jolly Chen nahradili dotazovací jazyk POSTQUEL jazykem SQL. Výsledný projekt byl označen jako Postgres95. Po ukončení studia pokračoval Chen v údržbě systému Postgres95.

V roce 1996 byl na základě potřeby Open Source SQL databázového serveru sestaven tým pro pokračování ve vývoji. Zpočátku byli zapojeni Marc Fournier, Thomas Lockhart, Vadim Mikheev a Bruce Momjian. Marc Fournier nabídl server k obsluze elektronické diskuze a hostování zdrojového kódu.

Na konci roku 1996 byl Postgres přejmenován na PostgreSQL, což symbolizuje původ vzniku na Berkeley a zároveň podporu jazyka SQL. První verze PostgreSQL 6.0 vyšla v lednu 1997. Následující verze vycházely každé tři až pět měsíců. Nejnovější verzí je PostgreSQL 9.2.4. [21]

Přidání prostorových funkcí se mohlo zdát být zbytečné, jelikož PostgreSQL už geometrické typy obsahoval. Tyto geometrické typy jsou ale příliš omezené pro GIS data a jejich analýzy. Byly vybudovány pro výzkumné akademické účely a jsou vhodnější pro počítačovou grafiku spíše než pro GIS.

V roce 2001 začalo budování prostorové databáze PostGIS pomocí vlastního geometrického typu. Výkon první implementace geometrického typu PostGISu byl 100-krát rychlejší než načtení do relačního modelu a 10-krát rychlejší než použití obecného BLOB (Binary Large OBject) sub-systému. První verze PostGIS 0.1 byla zveřejněna 31. května 2001.

V roce 2002 byla vydaná verze 0.7, která podporuje prostorové indexy GiST a PostgreSQL 7.2, který byl vydán ve stejném roce. Verze 0.7 rovněž zahrnuje podporu pro transformace mezi souřadnicovými systémy[1].

V roce 2007 přišla ve verzi 1.2 podpora normy ISO SQL/MM uvedené v [29], ta obsahuje více prostorových objektů. Tato verze byla první, která přinesla podporu pro křivky. Verze 1.3 pokračovala v následování normy SQL/MM. Přesunula všechny popisy funkcí PostGIS tak, aby odpovídaly normě SQL/MM, přidala prefix ST_ před názvy funkcí a několik nových funkcí.

Od verze 2.0 jsou součástí PostGISu rozšíření pro topologickou správu vektorových dat PostGIS Topology a podpora rastrových dat PostGIS Raster. Aktuální verzí je PostGIS 2.0.3. [19]

3.2 Simple Features for SQL Specification

GIS objekty podporované v PostGISu jsou podmnožinou Simple Features definované OpenGIS Consortium (OGC). OGC definuje dva standardní způsoby vyjádření prostorových objektů WKT (Well–Known Text) a WKB (Well–Known Binary). Oba tyto způsoby obsahují informace o typu a souřadnicích, kterými je geometrický typ určen. OGC vyžaduje, aby formát uložení zahrnoval informaci o souřadnicovém systému ve formě identifikátoru SRID.

Příklady WKT reprezentace:

- POINT(0 0)
- LINESTRING(0 0,1 1,1 2)
- POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1, 2 1, 2 2, 1 2,1 1))
- MULTIPOINT(0 0,1 2)
- MULTILINESTRING((0 0,1 1,1 2),(2 3,3 2,5 4))
- MULTIPOLYGON(((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1)), ((-1 -1,-1 -2,-2 -2,-2 -1,-1 -1)))
- GEOMETRYCOLLECTION(POINT(2 3),LINESTRING(2 3,3 4))

SQL/MM rozšíření:

- CIRCULARSTRING(0 0, 1 1, 1 0)
- COMPOUNDCURVE(CIRCULARSTRING(0 0, 1 1, 1 0),(1 0, 0 1))
- CURVEPOLYGON(CIRCULARSTRING(0 0, 4 0, 4 4, 0 4, 0 0),(1 1, 3 3, 3 1, 1 1))

[1] transformace jsou řešeny pomocí knihovny PROJ.4

- MULTICURVE((0 0, 5 5),CIRCULARSTRING(4 0, 4 4, 8 4))
- MULTISURFACE(CURVEPOLYGON(CIRCULARSTRING(0 0, 4 0, 4 4, 0 4, 0 0),(1 1, 3 3, 3 1, 1 1)),((10 10, 14 12, 11 10,10 10),(11 11, 11.5 11, 11 11.5, 11 11)))

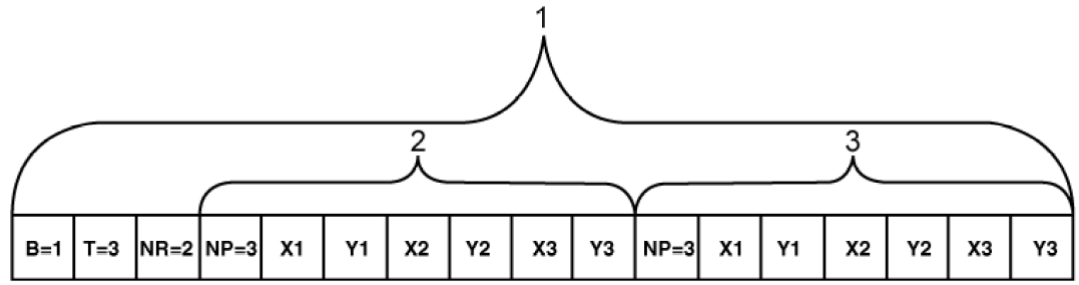
K vstupu a výstupu těchto formátů je třeba použití těchto funkcí:

```
bytea WKB = ST_AsBinary(geometry);
text WKT = ST_AsText(geometry);
geometry = ST_GeomFromWKB(bytea WKB, SRID);
geometry = ST_GeometryFromText(text WKT, SRID);
```

Příklad vstupu a výstupu polygonu ve formátu WKT:

```
INSERT INTO table_name (geometry) VALUES (ST_GeomFromText('Polygon(
       (0 0, 100 0, 100 100, 0 100, 0 0))'));

SELECT ST_AsText(geometry) FROM table_name;
```

Princip WKB reprezentace je znázorněn na obr. 3.1, jedná se o polygon s jedním vnitřním a jedním vnějším řetězcem (ringem):



Key

1 WKB Polygon
2 ring 1
3 ring 2

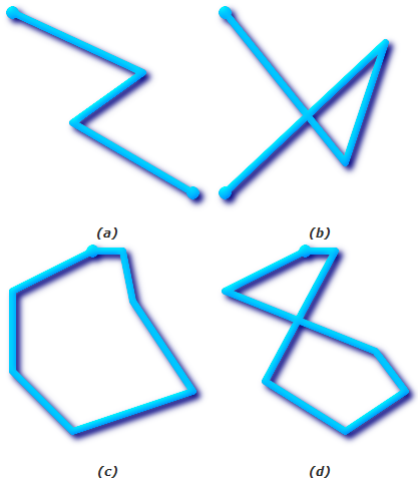
Figure 25: Well-known Binary Representation for a geometric object in NDR format (B = 1) of type Polygon (T = 3) with 2 LinearRings (NR = 2) each LinearRing having 3 points (NP = 3)

Obr. 3.1: Znázornění WKB fomátu, převzato z [30]

3.2.1 Validace

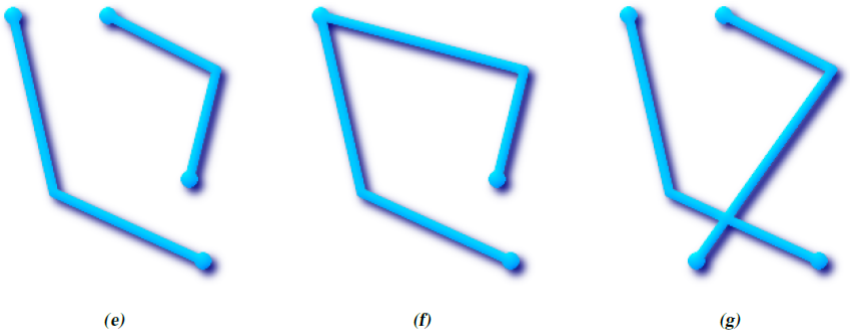
PostGIS, stejně jako mnoho dalších, vyžaduje pro svoje metody, aby geometrie, se kterou pracuje, byla validní a jednoduchá (simple). Podle OGC specifikace simple geometrie je ta, která neobsahuje žádné neobvyklé geometrické body, jako je křížení sama sebe (self intersection), dotýkání se sama sebe (self tangency) a primárně se odkazuje na geometrii s dimenzí 0 nebo 1. Naopak validace se odkazuje na geometrické objekty s dimenzí 2 a charakterizuje validní (platný) polygon. [20]

- POINT je simple, pokud je geometrickým typem s dimenzí 0.
- MULTIPOINT je simple, pokud žádné dvě souřadnice (body) nejsou totožné.
- LINESTRING je simple, pokud neprochází dvakrát stejným bodem, výjimkou je uzavřená linie, kdy se počáteční a koncový bod rovnají (c). Na obr. 3.2 jsou znázorněny linie (a), (c) jako simple a (b), (d) simple nejsou.



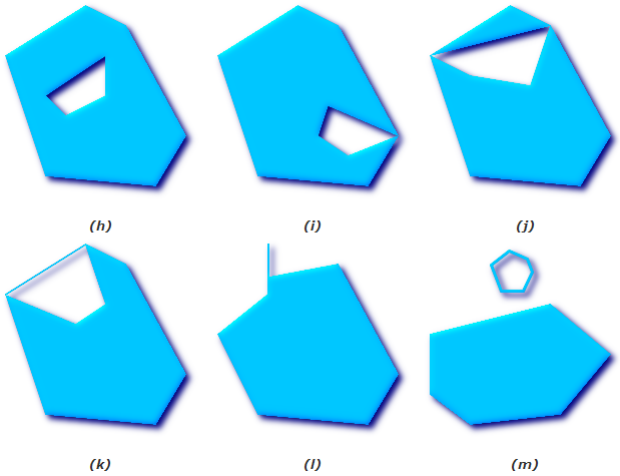
Obr. 3.2: Simple LINESTRING, převzato z [20]

- MULTILINESTRING je simple, pokud všechny její elementy jsou simple. Na obr. 3.3 můžeme (e) a (f) považovat za simple, ne však (g).



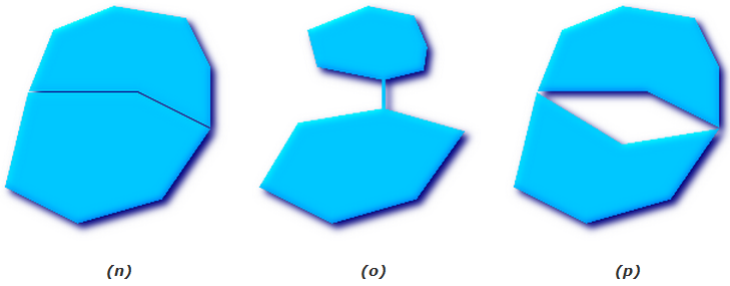
Obr. 3.3: Simple MULTILINESTRING, převzato z [20]

- Při testování POLYGONU funkcí `ST_IsSimple` bude výsledek vždy `TRUE`. POLYGON je validní, pokud se žádné dva řetězce nekříží na hranici (tvořené vnějším a vnitřním řetězcem) viz. polygon (h) na obr. 3.4.
- Hranice POLYGONU se mohou protínat pouze v jednom bodě (jako tangenta) (j).



Obr. 3.4: Validní POLYGON, převzato z [20]

- POLYGON nemůže mít dělící čáry (k) nebo hroty (l). Vnitřní řetězec musí být plně obsažen ve vnějším polygonu (m). Na obr. 3.4 (j – m) není reprezentací validního polygonu, (j) a (m) mohou být validní jako MULTIPOLYGON.
- MULTIPOLYGON je validní, pokud všechny jeho elementy jsou validní a vnitřní řetězce se neprotínají.
- Hranice dvou elementů MULTIPOLYGONU se mohou dotýkat pouze v konečném počtu bodů. Na obr. 3.5 (n) a (o) nejsou validní multipolygony, validní je pouze (p).



Obr. 3.5: Validní MULTIPOLYGON, převzato z [20]

K zjišťování validity a jednoduchosti dat slouží následující funkce:

```
SELECT ST_IsSimple(ST_GeomFromText('LINESTRING(0 0, 1 1)'));
 -- true

SELECT ST_IsSimple(geom_column)
FROM table_name;

SELECT ST_IsValid(ST_GeomFromText('POLYGON((1 1, 1 2, 2 2, 1 1))'));
 -- true
```

```
SELECT ST_IsValid(geom_column)
FROM table_name;

SELECT ST_IsValidReason('LINESTRING(220227 150406,2220227
  150407,222020 150410)');

SELECT gid, ST_IsValidReason(geom_column) as validity_info
FROM table_name;
```

Funkce ST_IsValidDetail vrací sloupce valid, reason, location. V případě, že se jedná o validní geometrii sloupce reason a location zůstanou prázdné.

```
SELECT * FROM ST_IsValidDetail('LINESTRING(220227 150406,2220227
 150407,222020 150410)');

SELECT gid, reason(ST_IsValidDetail(the_geom)),
 ST_AsText(location(ST_IsValidDetail(the_geom))) as location
FROM table_name;
```

3.2.2 Prostorové indexy

Indexy umožňují databázím zrychlit práci s větším objemem dat. Bez indexování by probíhalo vyhledávání jako sekvenční scanování každého záznamu v relaci. Indexování zvyšuje rychlost vyhledávání tím, že organizuje data do vyhledávacího stromu, ve kterém lze rychle najít konkrétní záznam. PostgreSQL podporuje následující typy indexů [20]:

- B–tree: Používá se pro data, která mohou být uložena podél jedné osy, jako jsou např. čísla, písmena, binární data. GIS data logicky nemohou být ukládána v jedné ose, takže B–tree indexování nelze použít pro práci s prostorovými daty.
- R–tree: Rozbíjí data do hierarchicky co nejmenších obdélníků. Tento způsob indexování je používán pro práci s prostorovými daty v některých prostorových databázích, jako je např. Spatial více v kapitole 4.2.4. Nicméně PostrgreSQL implementace R–tree není tak robustní jako implementace GiST.
- GiST: Generalized Search Trees rozbíjí data na „things to one side“, „things which overlap“, „things which are inside“. Může být použit na široké spektrum typů dat včetně GIS dat. PostGIS používá pro GIS data R–tree index implementovaný jako GiST.

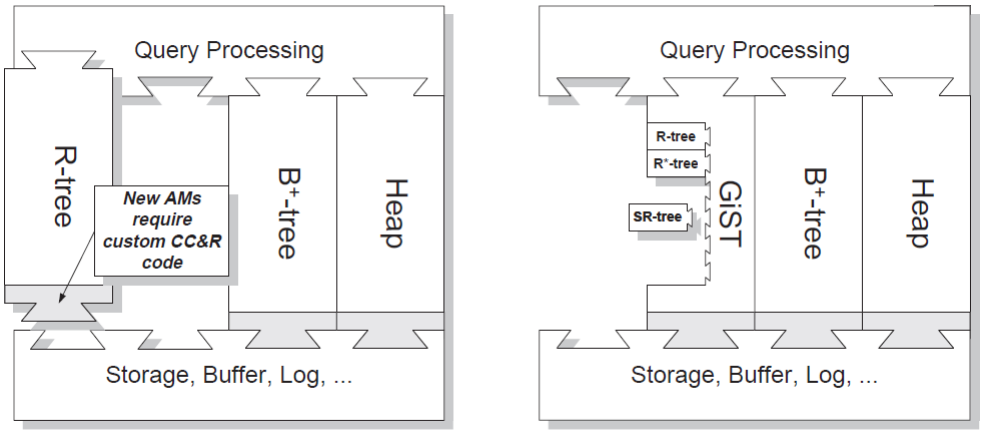
GiST

Poskytuje funkcionalitu všech „tree“ indexů (např. R–tree, B+–tree, hB–tree, TV–tree, CH–tree, atd.) v jednom. Kromě sjednocení všech těchto struktur, má GiST klíčovou vlastnost, kterou ostatní stromy postrádají, umožňuje rozšířitelnost

dat a dotazů. GiST je rozšiřitelná datová struktura, která umožňuje uživateli vytvořit indexy nad všemi typy dat a vyhledávání nad nimi. [23], [24]

Prostorový index je vytvořen pomocí následujícího dotazu:

```
CREATE INDEX index_name
ON table_name USING GIST (geom_column);
```



Obr. 3.6: Rozhraní metody GiST, převzato z [24]

3.2.3 Prostorové vztahy

V PostGISu mohou být testovány vztahy mezi dvěma geometriemi pomocí následujících pojmenovaných prostorových vztahů dle OGC [30]:

`ST_Equals` – vrací hodnotu 'TRUE', jestliže je geometrie a prostorově shodná s geometrií b.

$$a.Equals(b) \Leftrightarrow a \subseteq b \wedge b \subseteq a$$

`ST_Disjoint` – vrací hodnotu 'TRUE', jestliže je geometrie a prostorově různá s geometrií b.

$$a.Distjoint(b) \Leftrightarrow a \cap b = \emptyset$$

`ST_Intersects` – vrací hodnotu 'TRUE', jestliže geometrie a prostorově protíná geometrii b.

$$a.Intersects(b) \Leftrightarrow !a.Disjoint(b)$$

`ST_Touches` – vrací hodnotu 'TRUE', jestliže se geometrie a prostorově dotýká s geometrií b.

$$a.Touch(b) \Leftrightarrow (I(a) \cap I(b) = \emptyset) \wedge (a \cap b \neq \emptyset)$$

`ST_Crosses` – vrací hodnotu 'TRUE', jestliže se geometrie a prostorově kříží s geometrií b.

$$a.Cross(b) \Leftrightarrow (I(a) \cap I(b) \neq \emptyset) \wedge (a \cap b \neq a) \wedge (a \cap b \neq b)$$

`ST_Within` – vrací hodnotu 'TRUE', jestliže je geometrie a prostorově uvnitř geometrie b.

$$a.Within(b) \Leftrightarrow (a \cap b = a) \wedge (I(a) \cap E(b) \neq \emptyset)$$

`ST_Contains` – vrací hodnotu 'TRUE', jestliže geometrie a prostorově obsahuje geometrii b.

$$a.Contains(b) \Leftrightarrow b.Within(a)$$

`ST_Overlaps` – vrací hodnotu 'TRUE', jestliže geometrie a prostorově překrývá geometrii b.

$$a.Overlaps(b) \Leftrightarrow (dim(I(a)) = dim(I(b)) = dim(I(a) \cap I(b))) \wedge (a \cap b \neq a) \wedge (a \cap b \neq b)$$

PostGIS obsahuje i pojmenované prostorové vztahy, které nejsou součástí specifikace *Simple Features(SF)*, ale jsou obdobou vztahů v Oracle Spatial např.:

- `ST_Covers`,
- `ST_CoveredBy`,
- `ST_ContainsProperly` a mnoho dalších.

DE-9IM

V některých případech mohou být pojmenované předdefinované prostorové predikáty (ST_Contains, ST_Crosses, ...) nedostatečné. Příkladem je analýza silniční sítě, kdy požadavek na identifikování segmentů křížící samy sebe je kladen na linie, ne na křížení v bodě. V tomto případě není adekvátní použití ST_Crosses k provedení prostorového filtru, protože vrací hodnotu 'TRUE', pouze pokud se geometrie kříží v bodě. Řešením může být provedení ST_Intersection na dvojici silničních úseků ve vztahu ST_Intersects a následné porovnání geometrického typů křížení ST_GeometryType s 'LINESTRING'. Elegantnější a rychlejší řešení může přinést použití rozměrově rozšířenýho 9-ti průsečíkového modelu, nebo-li DE-9IM (Dimensionally Extended Nine – Intersection Model). [19]

Podle specifikace SF se jedná o základní přístup k porovnávání dvou geometrií pomocí testování průniků vnitřních částí (Interiors), hranic (Boundaries) a vnějších částí (Exteriors) obou geometrií. Vztahy těchto geometrií jsou specifikovány ve výsledné matici (intersection matrix).

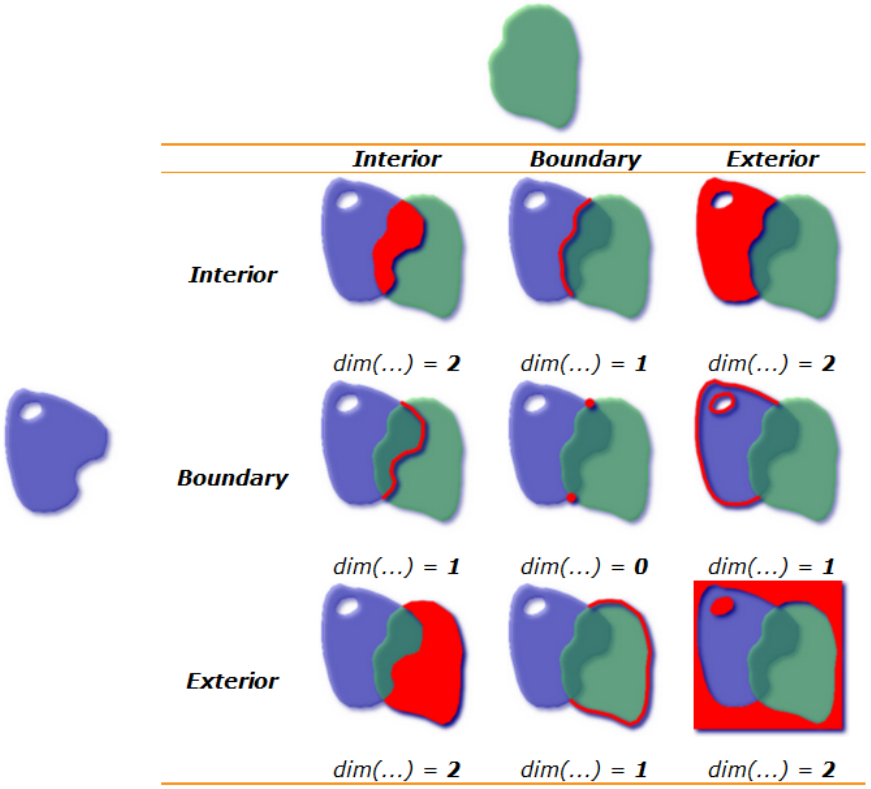
Boundary – Hranice geometrie je sada geometrie o dimenzi menší než objekt. V případě bodů, jejichž dimenze je 0, je hranice prázdná množina. Hranicí linie jsou dva koncové body. Pro polygony je hranicí liniová kresba, kterou tvoří vnitřní a vnější řetězce.

Interior – Vnitřní část geometrie jsou ty body geometrie, které zůstanou po odstranění hranice. Pro bod je vnitřní částí bod sám. V případě linií se jedná o sadu reálných bodů mezi koncovými body. Vnitřní částí polygonu je plošný povrch uvnitř polygonu.

Exterior – Vnější částí geometrie je plošný povrch mimo hranici a vnitřní část objektů.

	Interior	***Boundary***	***Exterior***
Interior	*dim(I(a) ∩ I(b))*	*dim(I(a) ∩ B(b))*	*dim(I(a) ∩ E(b))*
Boundary	*dim(B(a) ∩ I(b))*	*dim(B(a) ∩ B(b))*	*dim(B(a) ∩ E(b))*
Exterior	*dim(E(a) ∩ I(b))*	*dim(E(a) ∩ B(b))*	*dim(E(a) ∩ E(b))*

Obr. 3.7: Matematická reprezentace matice, převzato z [20]



Obr. 3.8: Znázornění dvou překrývajících se polygonů, převzato z [20]

Vztahy testovaných geometrických objektů jsou definovány následovně:

$$T \Leftrightarrow dim(x) \in \{0, 1, 2\},\ tj.\ x \neq \emptyset$$

$$F \Leftrightarrow dim(x) = -1,\ tj.\ x = \emptyset$$

$$* \Leftrightarrow dim(x) \in \{-1, 0, 1, 2\}$$

$$0 \Leftrightarrow dim(x) = 0$$

$$1 \Leftrightarrow dim(x) = 1$$

$$2 \Leftrightarrow dim(x) = 2$$

Vztahy definované v matici je možné aplikovat pomocí funkce `ST_Relate`, kterou rovněž můžeme použít pro zjištění matice dvou geometrií.

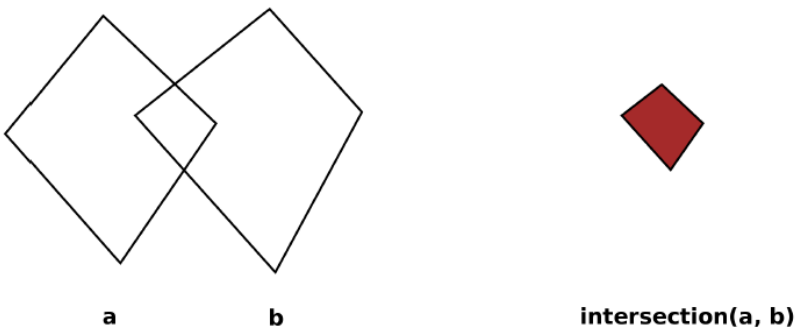
```
SELECT ST_Relate(ST_GeometryFromText('POINT(1 2)'),
        ST_Buffer(ST_GeometryFromText('POINT(1 2)'),2), '*FF*FF212');

SELECT ST_Relate(ST_GeometryFromText('POINT(1 2)'),
                ST_Buffer(ST_GeometryFromText('POINT(1 2)'),2));
```

3.2.4 Prostorové operátory

PostGIS obsahuje velké množství operátorů pro práci s prostorovými daty. V této kapitole jsou popsány základní operace AND, OR, MINUS a XOR.

ST_Intersection

Vrací novou geometrii, která je průnikem (operace AND) vstupujících geometrií. Ve spojení s `ST_Intersects` může být velice užitečná k ořezu geometrií.

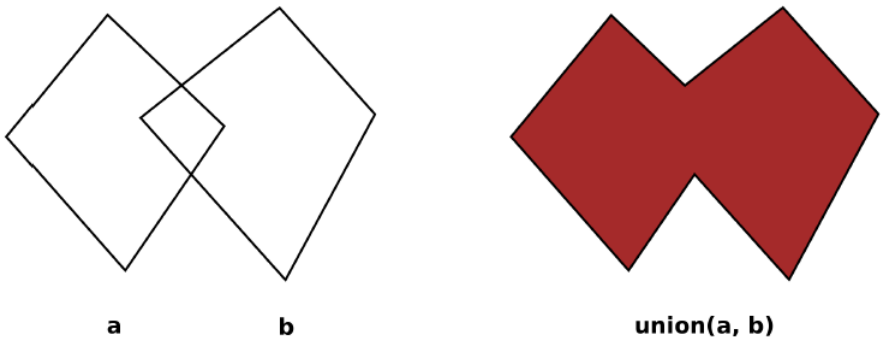


Obr. 3.9: Vstup a výstup geometrie při použití ST_Intersection, převzato z [32]

ST_Union

Výsledkem může být geometrie typu multi, single nebo collection[2], která je sjednocením (operace OR) vstupujících geometrií. Tato funkce může být použita k vytvoření nové geometrie sjednocení, nebo na principu agregační funkce. Zaměnit ji lze s funkcí `ST_Collect`, která je uváděna jako řádově rychlejší. Union je pomalejší, protože provádí dissolve na hranicích a snaží se měnit pořadí geometrií, aby nevznikaly multi geometrie s překrývajícími se regiony.

[2]v geometrii typu *collection* mohou být uloženy všechny typy geometrie, v geometrii typu *multi* může být uloženo více geometrií stejného typu(multipoint, multilinestring, multipolygon), v geometrii typu *single* je pro každý záznam uložena jedna geometrie (point, linestring, polygon)



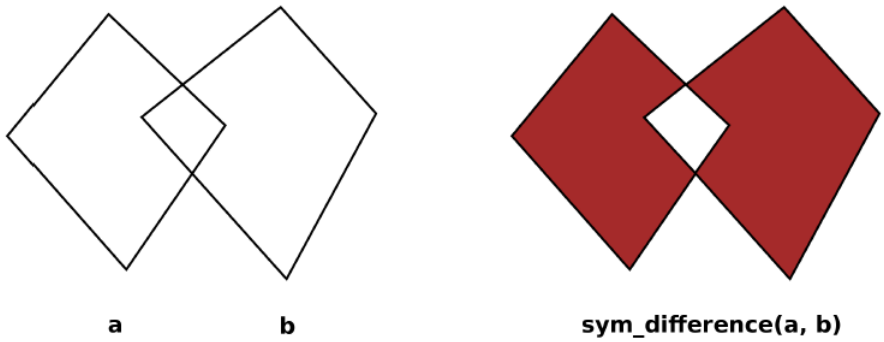
Obr. 3.10: Vstup a výstup geometrie při použití ST_Union, převzato z [32]

ST_SymDifference

Vrací geometrii, která reprezentuje části, které nemají vstupující geometrie společné (operace XOR). Nazývá se symetrickým rozdílem, protože platí:

```
ST_SymDifference(a,b) = ST_SymDifference(b,a)
```

Jedná se o plochu, která rovněž může vzniknout takto:
`ST_Union(a,b) - ST_Intersection(a,b)`

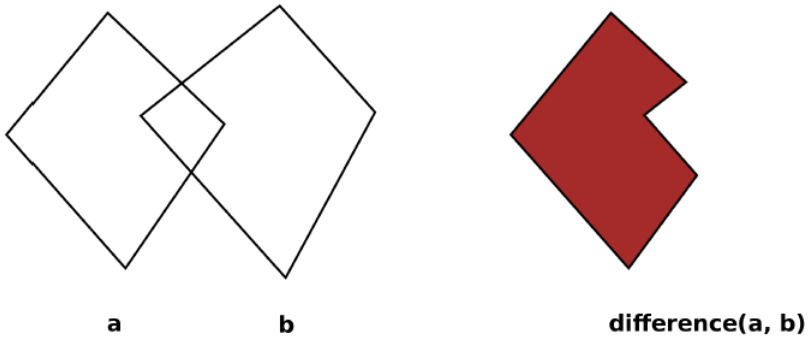


Obr. 3.11: Vstup a výstup geometrie při použití ST_SymDifference, převzato z [32]

ST_Difference

Vrací geometrii, která je rozdílem (operace MINUS) geometrií. Stejného výsledku můžeme rovněž dosáhnout použitím `GeometryA - ST_Intersection(a,b)`.

Další užitečné funkce pro práci v PostGIS lze nalézt v [20].



Obr. 3.12: Vstup a výstup geometrie při použití ST_Difference převzato z [32]

3.2.5 Datový typ Geometry

Geometrie dat je uložena v jediném řádku a sloupci typu Geometry, který odpovídá konkrétnímu záznamu. Datový typ Geometry je definován následujícím dotazem:

```
CREATE TYPE geometry AS OBJECT(
   catalog_name VARCHAR,
   schema_name VARCHAR,
   table_name VARCHAR,
   column_name VARCHAR,
   srid INTEGER,
   type VARCHAR,
   dimension INTEGER);
```

Informace o geometrii jsou ukládány do metadatových tabulek SPATIAL_REF_SYS a GEOMETRY_COLUMNS.

SPATIAL_REF_SYS

Jedná se o metadatovou tabulku, která je součástí PostGIS a je kompatibilní se specifikací SF, která uvádí přes 3000 souřadnicových systémů a detailů potřebných k transformaci mezi nimi. Tabulka je v databázi definována následovně:

```
CREATE TABLE spatial_ref_sys (
 srid INTEGER NOT NULL PRIMARY KEY,
 auth_name VARCHAR(256),
 auth_srid INTEGER,
 srtext VARCHAR(2048),
 proj4text VARCHAR(2048));
```

Atribut	Popis
SRID	Identifikátor souřadnicového systému v databázi.
AUTH_NAME	Název normy nebo orgánu, který je uvedený pro ref. systém.
AUTH_SRID	Identifikátor ref. systému podle definice normy AUTH_NAME.
SRTEXT	WKT popis referenčního systému.
PROJ4TEXT	PostGIS používá knihovny PROJ4 k zajištění transformace. Tento sloupec obsahuje PROJ4 definice řetězce souřadnic pro konkrétní SRID.

Tab. 3.1: Popis tabulky SPATIAL_REF_SYS

GEOMETRY_COLUMNS

V dřívějších verzích PostGISu představoval GEOMETRY_COLUMNS tabulku, od verze 2.0.0 se stal pohledem (view) se stejnou strukturou, jaká je v předchozích verzích.

Atribut	Popis
F_CATALOG_NAME	Převzato ze Spatial, PostgreSQL nemá analogii pro katalog, vkládá proto název databáze.
F_SCHEMA_NAME	Obsahuje název použitého schéma, pokud nemáme žádné vlastní, vkládá schéma PUBLIC.
F_TABLE_NAME	Obsahuje název tabulky, která obsahuje sloupec s geometrií.
F_COLUMN_NAME	Název sloupce, ve kterém je uložena geometrie objektu.
SRID	Identifikátor souřadnicového systému, který je cizím klíčem propojující pohled s metadatovou tabulkou SPATIAL_REF_SYS.
TYPE	Typ prostorového objektu jako je bod, linie, polygon ('POINT', 'LINESTRING','POLYGON').
COOR_DIMENSION	Dimenze sloupce s geometrií (= 2, 3, 4).

Tab. 3.2: Popis pohledu GEOMETRY_COLUMNS

3.3 PostGIS Topology

PostGIS Topology je rozšíření pro PostGIS umožňující topologickou správu vektorových dat v prostředí PostGIS/PostgreSQL.

Vychází z datového modelu Topo – Geo z technické normy SQL/MM (ISO 13249–3:2006) uvedené v [28]. Ten je založen na principu NAA struktury, více o topologických modelech a strukturách v [7]. Topologické prvky[3], které je možné ukládat v PostGIS Topology, jsou stejné jako ve Spatial. Více o těchto prvcích v kapitole 4.3. Topologie dat je uložena v pojmenovaných schématech, kde název topologie je stejný jako název schéma, ve kterém jsou uložena topologická data.

Sandro Santilli, jeden z hlavních vývojářů PostGIS, uvádí v [25], že hlavními důvody používání topologického modelu uložení dat jsou:

- Topologická integrita dat.
  - V každém křížení hran je vytvořen uzel.
  - Hrany jsou sdílené.
- Redukce velikosti uložení.
  - Každá hrana je uložena pouze jednou.
  - Prvky uložené v hierarchických vrstvách mohou být definovány skladbou (státy jsou kolekcí regionů).
- Explicitní prostorové vztahy.
  - Dotýkající se polygony i hrany mají společnou hranu.
  - Pro každou hranu známe pravou a levou stěnu.
  - Pro každý izolovaný uzel známe stěnu, ve které leží.
  - V místech křížení jsou uzly.
  - Uzly jsou sdílené.

Rozšíření PostGIS Topology vytvoří v databázi nové schéma topology. Schéma obsahuje dvě metadatové tabulky *Layer* a *Topology*.

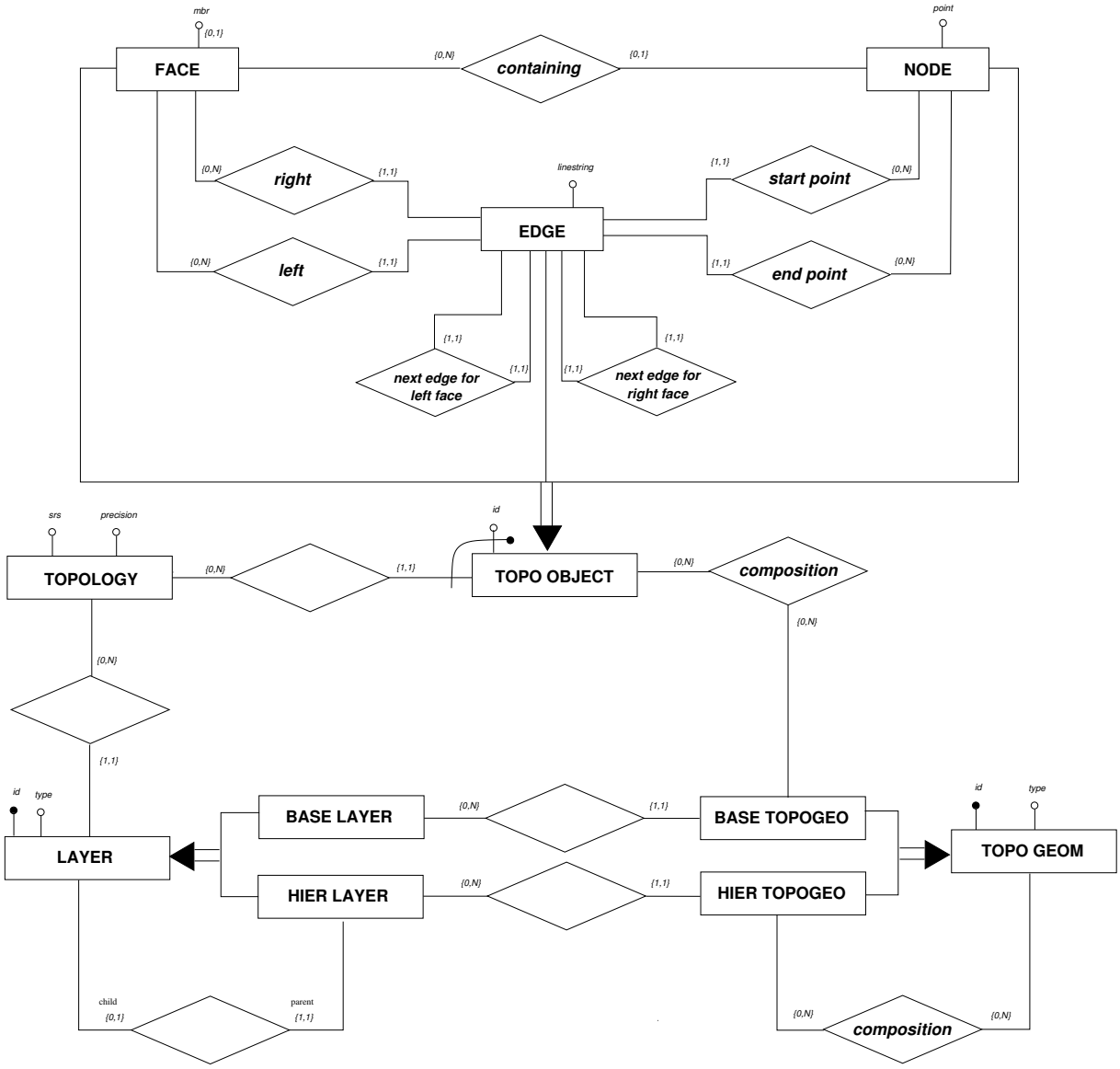
Atribut	Datový typ	Popis
ID	INTEGER	Identifikátor topologie.
NAME	VARCHAR	Název topologie.
SRID	INTEGER	Identifikátor odkazující na souřadnicový systém topologie.
PRECISION	DOUBLE	Tolerance definující přesnost dat.

Tab. 3.3: Struktura tabulky Topology ve schématu topology

[3]Topologickými prvky jsou topologická primitiva (uzly, hrany, stěny), izolované uzly, izolované hrany, smyčky a univerzální stěna F0.

Atribut	Datový typ	Popis
TOPOLOGY_ID	INTEGER	Identifikátor topologie.
LAYER_ID	INTEGER	Identifikátor vrstvy.
SCHEMA_NAME	VARCHAR	Název schéma topologie.
TABLE_NAME	VARCHAR	Název tabulky.
FEATURE_COLUMN	VARCHAR	Název sloupce typu TopoGeometry.
FEATURE_TYPE	VARCHAR	Typ prvku uloženého v topologii.
LEVEL	INTEGER	Úroveň v hierarchickém uložení.
CHILD_ID	INTEGER	Identifikátor odkazující se na child layer v hierarchii.

Tab. 3.4: Struktura tabulky Layer ve schématu topology



Obr. 3.13: Koncepční model PostGIS Topology, převzato z [22]

3.3.1 Datový typ TopoGeometry

TopoGeometry objekty jsou definovány specifikováním jejich komponent topologických elementů. PostGIS podporuje jak jednoduchou Topo geometrii, tak i hierarchické uložení vrstev v topologii.

```
CREATE TYPE geometry AS OBJECT(
   topology_id INTEGER,
   layer_id INTEGER,
   id INTEGER,
   type INTEGER,);
```

Atribut	Popis
TOPOLOGY_ID	Identifikátor, který odkazuje na topologii uloženou v tabulce topology.topology, kde je definováno topology schéma a SRID.
LAYER_ID	Identifikátor vrstvy ve které je sloupec typu TopoGeometry. Kombinace TOPOLOGY_ID a LAYER_ID jednoznačně odkazuje na vrstvu v tabulce topology.layer.
ID	Identifikátor definující TopoGeometry v příslušné topologické vrstvě.
TYPE	Definuje geometrický typ 1:[multi]point, 2:[multi]line, 3:[multi]polygon, 4:collection.

Tab. 3.5: Popis datového typu TopoGeometry

3.3.2 Topologické tabulky

Během procedury `CreateTopology()`jsou ve schématu s názvem topologie vytvořeny topologické tabulky. Jedná se o tabulky, které obsahují topologické elementy (primitiva)[4]. Obrázek 3.14 ukazuje princip propojení tabulek od sloupce typu TopoGeometry po topologické tabulky primitiv.

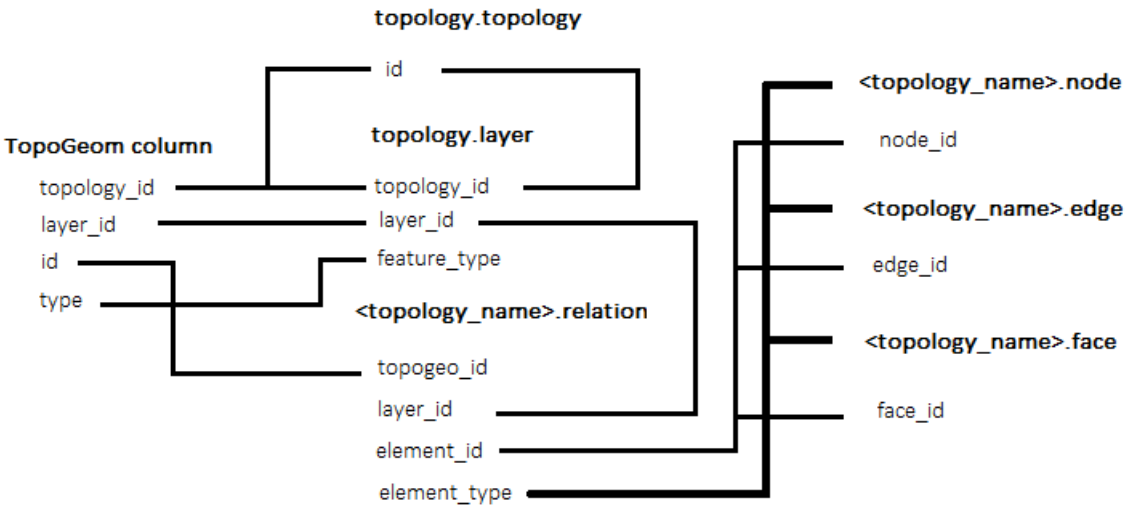
Tabulka uzlů

Tabulka `<topology_name>_node` je vytvořena automaticky během procedury `CreateTopology(topology_name)`. Obsahuje uzly topologických primitiv obsažených v topologii.

[4]jedná se o uzly, hrany a stěny

Atribut	Datový typ	Popis
NODE_ID	INTEGER	Identifikátor uzlu.
CONTAINING_FACE	INTEGER	Identifikátor stěny ve které leží izolovaný uzel.
GEOM	GEOMETRY	Geometrický popis objektu reprezentujícího uzel.

Tab. 3.6: Struktura tabulky uzlů



Obr. 3.14: Propojení tabulek v PostGIS

Tabulka hran

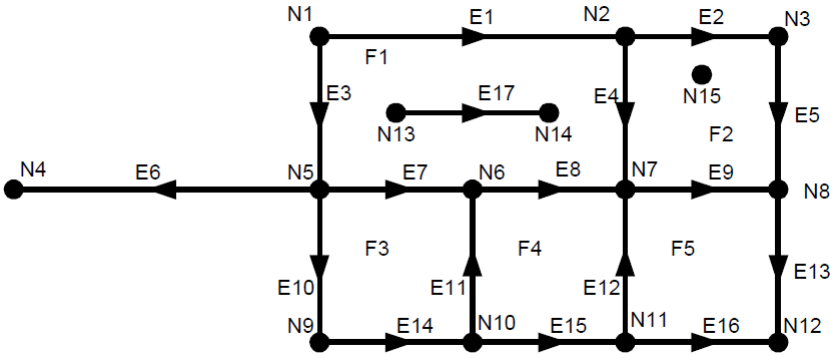
Obsahuje hrany topologických primitiv obsažených v topologii. Během procedury vytváření topologie je vytvořen pohled `<topology_name>.edge`[5] a tabulka `<topology_name>.edge_data`.

Na rozdíl od okřídlené hrany, kterou využívá Spatial, neobsahuje Topo – Geo informace o hranách typu PREV_LEFT_EDGE a PREV_RIGHT_EDGE. Znaménka u identifikátorů NEXT_LEFT_EDGE a NEXT_RIGHT_EDGE indikují směr. Princip je stejný, jaký je popsán v kapitole 4.3.2.

[5]Pohled obsahuje stejná data a atributy jako topologická tabulka hran. Tabulka je navíc doplněna o absolutní hodnoty následujících hran.

Atribut	Datový typ	Popis
EDGE_ID	INTEGER	Identifikátor hrany.
START_NODE	INTEGER	Identifikátor počátečního uzlu hrany.
END_NODE	INTEGER	Identifikátor koncového uzlu hrany.
NEXT_LEFT_EDGE	INTEGER	Identifikátor (včetně znaménka) NEXT_LEFT hrany.
NEXT_RIGHT_EDGE	INTEGER	Identifikátor (včetně znaménka) NEXT_RIGHT hrany.
LEFT_FACE	INTEGER	Identifikátor stěny ležící nalevo od hrany při pohledu ve směru hrany.
RIGHT_FACE	INTEGER	Identifikátor stěny ležící napravo od hrany při pohledu ve směru hrany.
GEOM	GEOMETRY	Geometrický popis objektu reprezentující hranu.

Tab. 3.7: Struktura pohledu hran



Obr. 3.15: Topologické prvky, převzato z [28]

Edge ID	Start Node ID	End Node ID	Next Left Edge	Next Right Edge	Left Face ID	Right Face ID	Geometry
1	1	2	2	3	0	1	(8, 8), (16, 8)
2	2	3	5	4	0	2	(16, 8), (20, 8)
3	1	5	7	1	1	0	(8, 8), (8, 4)
4	2	7	9	-1	1	2	(16, 8), (16, 4)
5	3	8	13	-2	0	2	(20, 8), (20, 4)
6	5	4	-6	-3	0	0	(8, 4), (0, 4)
7	5	6	8	10	1	3	(8, 4), (12, 4)
8	6	7	-4	-11	1	4	(12, 4), (16, 4)

Obr. 3.16: Ukázka tabulky hran popisující topologické prvky z obr. 3.15, převzato z [28]

Tabulka stěn

Tabulka `<topology_name>_face` je vytvořena automaticky během procedury `CreateTopology(topology_name)`. Obsahuje stěny topologických primitiv obsažených v topologii. Oproti Spatial je stěna *universal face F0*[6] vytvořena automaticky.

[6]stěna obsahující vše v topologii, vyznačuje se negativním identifikátorem

Atribut	Datový typ	Popis
FACE_ID	INTEGER	Identifikátor stěny.
MBR_GEOMETRY	GEOMETRY	Geometrie minimálního ohraničujícího obdélníku stěny.

Tab. 3.8: Struktura tabulky stěn

Tabulka relací

Tabulka vztahů (relací) `<topology_name>.relation` je rovněž vytvořena automaticky. Propojuje prostorové tabulky obsahující atribut typu TopoGeometry s tabulkami topologie (topology, Layer) a s tabulkami primitiv, jak je naznačeno v obr. 3.14.

Atribut	Datový typ	Popis
TOPOGEO_ID	INTEGER	Identifikátor topologie uložené v tabulce topology.
LAYER_ID	INTEGER	Identifikátor vrstvy uložené v tabulce layer.
ELEMENT_ID	INTEGER	Identifikátor primitiva v topologii.
ELEMENT_TYPE	INTEGER	Obsahuje typ topologického primitiva, v případě nehierarchického uložení 1:node, 2:edge, 3:face, v případě hierarchického uložení identifikátor vrstvy.

Tab. 3.9: Struktura tabulky relací

3.4 Práce s daty

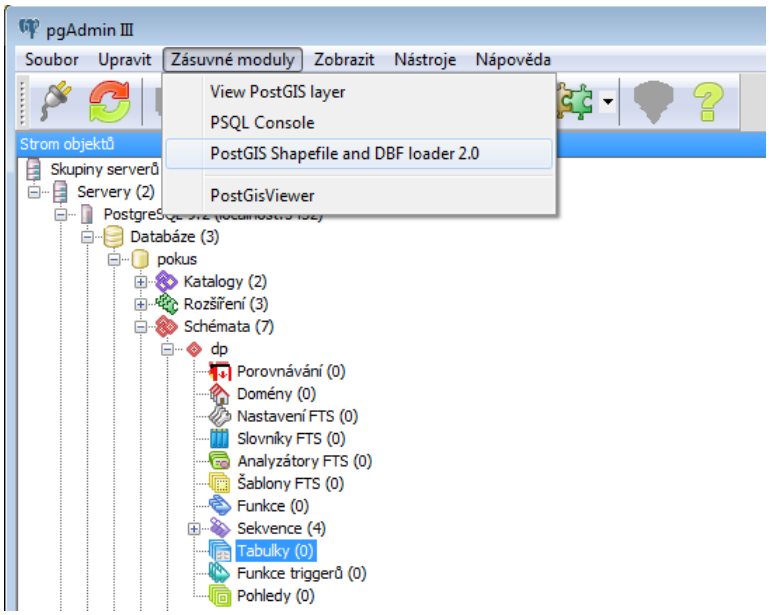
1. Založení schématu a nastavení cesty vyhledávání na *name_schema*, *topology* a *public*.
2. Import prostorových dat do *name_schema* v databázi.
3. Validace dat.
4. Vytvoření prostorových indexů.
5. Vytvoření topologie pomocí `CreateTopology()`, automaticky je vytvořeno schéma s názvem shodným se jménem topologie.
6. Přidání sloupce typu TopoGeometry pomocí `AddTopoGeometryColumn()` do tabulek, které budou součástí topologie.
7. Naplnění topologických tabulek pomocí `toTopoGeom()`.

3.4.1 Založení schéma a nastavení cesty vyhledávání

```
CREATE SCHEMA DP;

SET search_path TO dp, topology, public;
```

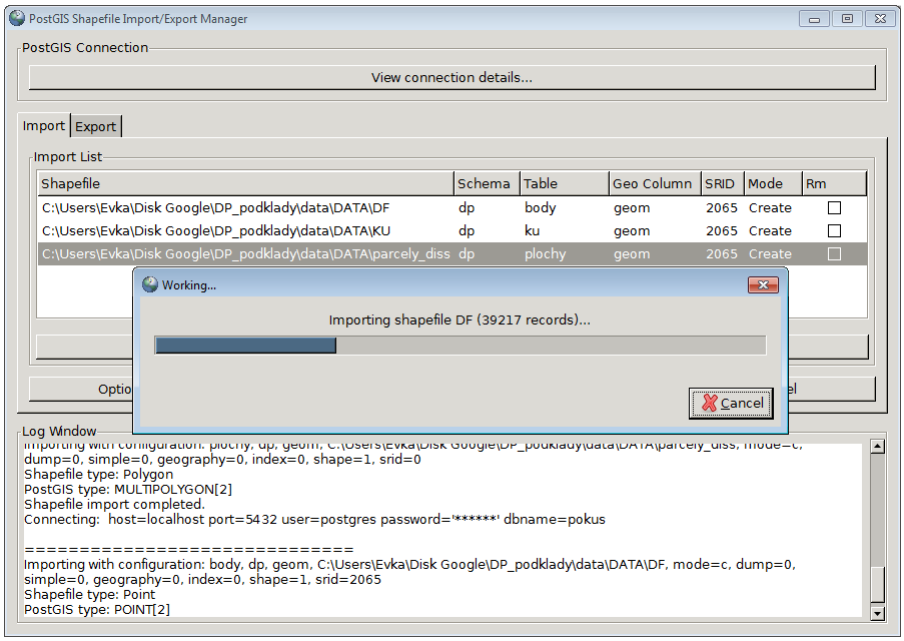
3.4.2 Import dat

Import dat z formátu ∗.shp byl proveden pomocí pluginu *PostGIS Shapefile Import/Export Manager*. Na obr. 3.17 je znázorněno spuštění pluginu.



Obr. 3.17: Ukázka spuštění pluginu pro import

Na obr. 3.18 je ukázka importu dat. Tímto způsobem byly nahrány tabulky *ku*, *body*, *linie* a *plochy* do schématu DP více o datech, která byla importována v kapitole 5.



Obr. 3.18: Ukázka importu dat pomocí PostGIS Shapefile Import/Export Manager

3.4.3 Validace dat

Kontrolovaná tabulka	Validita/Simplicita	Popis
BODY	TRUE	
LINIE	9 FALSE	Funkce `ST_IsSimple` nevypisuje důvod.
KU	TRUE	
PLOCHY	78 FALSE	Ring Self-intersection.

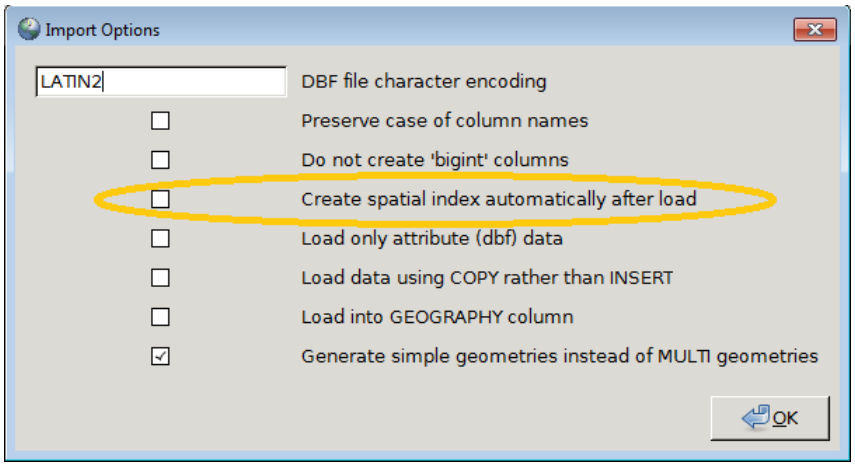
Tab. 3.10: Validace a jednoduchost dat

Po prohlédnutí linií, které nejsou simple byly linie odstraněny:

```
DELETE FROM linie
  WHERE NOT ST_IsSimple(geom);
```

Křížení nevalidních polygonů bylo odstraněno pomocí:

```
UPDATE plochy
SET geom=ST_Buffer(geom, 0.0);
```



Obr. 3.19: Ukázka možností nastavení importu dat

3.4.4 Vytvoření prostorových indexů

Prostorové indexy mohou být vytvořeny dvojím způsobem. Buď pomocí dotazu, nebo, jak je patrné z obr. 3.19, pomocí nastavení pluginu při importu dat.

Prostorové indexy byly všechny vytvořeny pomocí dotazu:

```
CREATE INDEX ku_idx
ON ku USING GIST (geom);
```

3.4.5 Vytvoření topologicky strukturovaných dat

Založení topologie

```
SELECT CreateTopology('dp_topo',2065,0.005);
```

Během procedury `CreateTopology` byly vytvořeny čtyři tabulky a jeden pohled, jak je znázorněno na obr. 3.20.

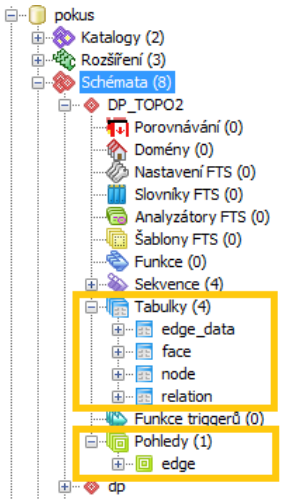
Přidání datového typu TopoGeometry do tabulek

Sloupec typu TopoGeometry je přidán do existujících prostorových tabulek následujícím dotazem:

```
SELECT AddTopoGeometryColumn('dp_topo', 'dp', 'ku', 'topo', 'POLYGON');
SELECT AddTopoGeometryColumn('dp_topo', 'dp', 'body', 'topo', 'POINT');
SELECT AddTopoGeometryColumn('dp_topo', 'dp', 'plochy', 'topo', 'POLYGON');
SELECT AddTopoGeometryColumn('dp_topo', 'dp', 'linie', 'topo', 'LINESTRING');
```

Naplnění topologických tabulek

Pomocí procedury `toTopoGeom()` byly topologické tabulky naplněny.



Obr. 3.20: Ukázka vytvořených topologických tabulek

```
UPDATE ku SET topo = topology.toTopoGeom(geom, 'dp_topo',1,0.005);
UPDATE body SET topo = topology.toTopoGeom(geom, 'dp_topo',2,0.005);
UPDATE plochy  SET topo = topology.toTopoGeom(geom, 'dp_topo',3,0.005);
UPDATE linie SET topo = topology.toTopoGeom(geom, 'dp_topo',4,0.005);
```

4 Oracle Spatial

Oracle Spatial, často označován jen jako Spatial, je integrovaná sada funkcí a procedur, která umožňuje efektivně a rychle spravovat a analyzovat prostorová data v Oracle databázi. Spatial je dostupný pouze v licenci Oracle Enterprise Edition.

Spatial se skládá z následujících částí:

- Schéma MDSYS (Multi Dimensional), které předepisuje ukládání, syntaxe a sémantiku podporovaných geometrických datových typů.
- Mechanismus prostorového indexování.
- Sada operátorů a funkcí pro provádění dotazů jako *spatial join*, *area–of–interest* nebo jiných prostorových analýz.
- Funkce a postupy pro utility.
- Topologický datový model pro práci s daty jako jsou uzly, hrany a stěny v topologii.
- Síťový datový model pro reprezentaci vlastností nebo objektů, které jsou modelovány jako uzly nebo odkazy v síti.

4.1 Historie

V roce 1977, na základě článku v *IBM Journal of Research and Development* z roku 1970 popisujícího prototyp relačního databázového systému, založil Larry Ellison se svými kolegy Bobem Minerem a Edem Oatesem System Development Laboratories (SDL). Jejich cílem bylo vytvořit softwarový databázový projekt a poskytovat ho na komerční úrovni. První verze vyšla v roce 1978, ale nebyla nikdy komerčně dostupná. O rok později společnost změnila jméno na Relation Software Inc. (RSI) a vydala první komerční SQL relační databázový systém jako verzi 2.

V roce 1982 se společnost opět přejmenovala, novým názvem se stalo původně kódové označení projektu pro CIA Oracle Corporation. Verze 3 z roku 1983 byla napsána v jazyce C, poprvé se objevuje podpora transakčního zpracování (commit, rollback). Tato verze byla prvním RDBMS určeným pro sálové a stolní počítače. Potenciál pro prostorová data byl naznačen ve verzi 4 vědci pracujícími s CHS (Canadian Hydrographic Service). Podporu Klient – server modelu přinesla verze 5. PL/SQL jazyk byl podporován od Oracle verze 6 v roce 1988. [10]

Spatial byl původně označován jako SDO a ještě dříve jako MultiDimension. Od verze 7 je možné pomocí nadstavby Spatial vkládat do databáze body. SDO (Spatial Data Option) byl dodán ve verzi 7.3.3, ta zajišťuje možnost práce nejen s body, ale i s liniemi a polygony. Ve verzi 8 je Spatial doplněn navíc o práci s oblouky a úhly, rovněž byly zabudovány datové typy a Spatial operátory. Podpora topologie přišla až ve verzi 10g (2004) spolu s GeoRaster a Spatial Analysis. Podpora pro SQL/MM typy a funkce je obsažena v 11g release 1. Poslední verzí je Oracle database 11g

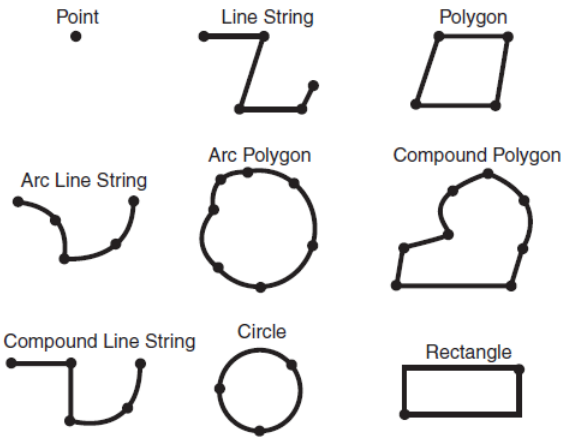
release 2. [11]

4.2 Objektově–relační model

Spatial podporuje objektově–relační model pro reprezentaci geometrie. Tento model používá tabulky s jedním sloupcem (SDO_GEOMETRY) a jedním řádkem pro každý geometrický záznam. Koresponduje s „SQL with Geometry Types“ implementací prostorových tabulek podle OGC specifikace.

Výhody objektově–relačního modelu:

- Podporuje mnoho geometrických typů včetně oblouků, kružnic, atd.
- Snadné použití během vytváření a udržování indexů a provádění prostorových dotazů.
- Indexy spravované Oracle databází.
- Geometrie uložené v jediném řádku a jediném sloupci.
- Optimální výkon.



Obr. 4.1: Podporované geometrické typy ve Spatial, převzato z [8]

4.2.1 Datový typ SDO_GEOMETRY

Tabulka, která obsahuje sloupec SDO_GEOMETRY, musí zároveň obsahovat jiný sloupec, který definuje unikátní primární klíč.

```
CREATE TYPE sdo_geometry AS OBJECT (
   sdo_gtype NUMBER,
   sdo_srid NUMBER,
```

```
sdo_point SDO_POINT_TYPE,
sdo_elem_info SDO_ELEM_INFO_ARRAY,
sdo_ordinates SDO_ORDINATE_ARRAY);
```

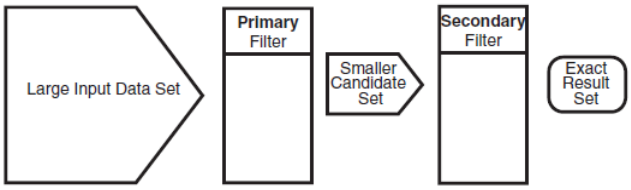
Atribut	Popis
SDO_GTYPE	Označuje geometrický typ objektu.
SDO_SRID	Slouží k identifikaci souř. systému, v rámci jednoho sloupce musí být hodnota stejná.
SDO_POINT	Pokud jsou ve vrstvě body, jsou zde uloženy souřadnice, tzn. SDO_ELEM_INFO a SDO_ORDINATES jsou NULL, jinak je tento atribut ignorován.
SDO_ELEM_INFO	Popisuje způsob interpretace souřadnic geom. prvku uloženého v SDO_ORDINATES.
SDO_ORDINATES	Zde jsou uloženy souřadnice bodů, které tvoří geom. prvek. Musí být vždy užit s atributem SDO_ELEM_INFO.

Tab. 4.1: Popis datového typu SDO_GEOMETRY

4.2.2 Dotazovací model

Spatial používá k řešení prostorových dotazů a prostorových spojení dvouvrstvý dotazovací model (two – tier query model). Tento termín se užívá při použití dvou různých operací pro řešení dotazů. Hledaný výsledek je kombinací těchto dvou operací, které jsou označovány jako primární a sekundární operace filtrování. [8]

- Primární filtr umožňuje rychlý výběr kandidátních záznamů, které předává do sekundárního filtru. Porovnává přibližnou geometrii, tím snižuje výpočetní náročnost dotazu a vrací nadhodnocenou množinu přesného výsledku.
- Sekundární filtr aplikuje přesné výpočty na data vystupující z primárního filtru. Získává přesné výsledky z prostorového dotazu. Operace je výpočetně náročná, ale je prováděna pouze na části dat.



Obr. 4.2: Dotazovací model, převzato z [8]

4.2.3 Validace dat

Během validace je kontrolována konzistence dat z hlediska typu a geometrie. Níže je popsáno, co všechno je ve Spatial nastaveno ke kontrole validity dat.

- SDO_GTYPE je validní.
- Hodnota SDO_ETYPE je konzistentní s hodnotou SDO_GTYPE.
- SDO_ELEM_INFO_ARRAY má validní triplet hodnotu.
- Polygon má minimálně 4 body, což zahrnuje bod, který uzavírá polygon (poslední bod polygon je totožný s prvním).
- Polygony samy sebe nepřekrývají.
- Žádné dva vrcholy linie nebo polygonu nejsou totožné.
- Polygony jsou správně orientovány. (Vnější hranice musí být orientována proti směru hodinových ručiček a vnitřní hranice musí být orientována ve směru hodinových ručiček).
- Vnitřní polygon se dotýká vnějšího polygonu maximálně jediným bodem.
- Pokud dva nebo více vnitřních polygonových řetězců leží uvnitř vnějšího řetězce, nesmí se tyto vnitřní řetězce vzájemně dotýkat více jak jedním bodem.
- Linie má nejméně dva vrcholy.
- SDO_ETYPE číslice 1 a 4 nejsou kombinovány (tzn. nejsou obě použity) v definici prvku polygonu.
- Body na oblouku nejsou kolineární (tzn. nejsou na přímé linii) a nejsou totožné.
- Geometrie jsou v daných mezích platných hodnot ve sloupci DIMINFO v pohledu USER_SDO_GEOM_METADATA.

Validita je kontrolována pomocí následujících funkcí.

SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT

Tato funkce vrací:

- pro validní geometrii hodnotu **TRUE**,
- jestliže geometrie není validní, vrací funkce chybovou zprávu s kódem chyby odkazující na důvod invalidity, nebo vrátí hodnotu **FALSE**, pokud geometrie nevyhovuje z nějakého jiného důvodu,
- kontext chyby – pokud je geometrie invalidní, výsledek může zahrnovat informaci o následující kombinaci: souřadnice, prvek, řetězec a hrana.

SDO_GEOM.VALIDATE_LAYER_WITH_CONTEXT

Tato funkce načítá výsledky validace do tabulky výsledků. Prázdná tabulka výsledků (`val_results`) musí být vytvořena před voláním této procedury. Formát této tabulky je popsán níže.

```
CREATE TABLE val_results (sdo_rowid ROWID, result varchar2(1000));
```

Tabulka výsledků obsahuje jeden řádek pro každou invalidní geometrii. Pokud je geometrie validní, nebude pro ni vytvořen záznam v této tabulce. Sloupec SDO_ROWID obsahuje hodnotu ROWID, která identifikuje záznam obsahující nevalidní geometrii. Sloupec RESULT obsahuje chybovou zprávu s kódem Oracle chyby a popisem. Tato procedura prochází všechny geometrie ve vrstvě.

- Všechny konzistence typů a geometrií jsou kontrolovány funkcí `SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT`.
- Geometrická hodnota SRID je stejná jako hodnota specifikovaná ve sloupci DIMINFO.

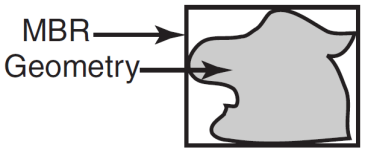
4.2.4 Prostorové indexy

Možnost prostorového indexování v databázi Oracle je klíčovým prvkem v produktu Spatial. Prostorový index, stejně jako jakýkoli jiný index, podporuje mechanismus pro limitování vyhledávání, ale v tomto případě mechanismus vychází z prostorových kritérií. Spatial podporuje tyto typy indexů:

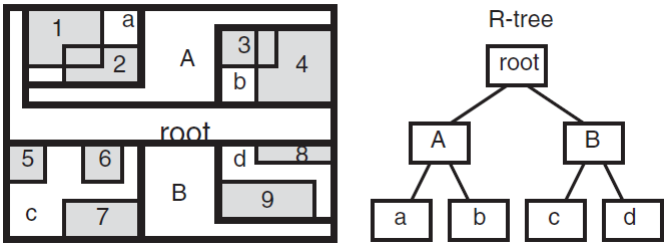
- Quadtree: Nebo-li čtyřstrom je typ dlaždicového indexu. Quadtree indexování je kritizovaným rysem Spatial a jeho použití se nedoporučuje.
- R–tree: Aproximuje každé geometrii jediný nejmenší ohraničující obdélník MBR (tj. minimum bounding rectangle).

R-Tree

Pro vrstvu geometrických popisů se R–tree index skládá z hierarchického indexu konvexní obálky prvků ve vrstvě, obr. 4.4. Objekty 1 – 9 jsou geometrie uložené ve vrstvě. Kořen obsahuje MBR A,B, které dále obsahují listy a,b,c,d. Ty obsahují MBR spolu s odkazy na geometrii.



Obr. 4.3: Nejmenší ohraničující obdélník, převzato z [8]



Obr. 4.4: R–tree index, převzato z [8]

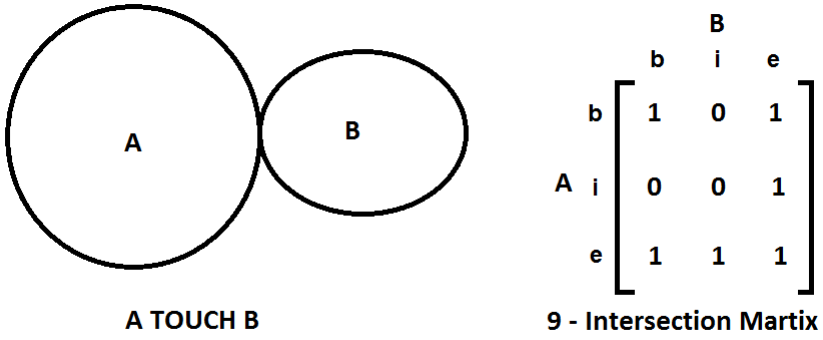
4.2.5 Prostorové vztahy

Spatial používá k určení prostorových vztahů mezi entitami sekundárních filtrů. Většina prostorových vztahů je založena na topologii a vzdálenosti. K určení prostorových vztahů slouží tyto základní prostorové operátory:

- `SDO_RELATE` – operátor na principu sekundárního filtru, který posuzuje topologická kritéria.
- `SDO_WITHIN_DISTANCE` – operátor na principu sekundárního filtru, který určuje, zda jsou dva prostorové objekty od sebe vzdáleny ve specifikované vzdálenosti.
- `SDO_NN` – operátor na principu sekundárního filtru, který identifikuje nejbližší geometrický prvek prostorového objektu.
- `SDO_NN_DISTANCE` – doplňkový operátor k `SDO_NN`, který vrací vzdálenost nejbližšího nalezeného geometrického prvku od prostorového objektu.
- `SDO_FILTER` – operátor na principu primárního filtru, jenž specifikuje geometrické objekty, které mohou mít prostorový vztah s daným prostorovým objektem.
- `SDO_JOIN` – funkce, která je uváděna mezi operátory, protože její používání je stejné jako použití operátoru. Funkce není součástí žádného balíčku funkcí. Provádí prostorové spojení založené na jednom nebo více prostorových vztazích.

The Nine-Intersection Model

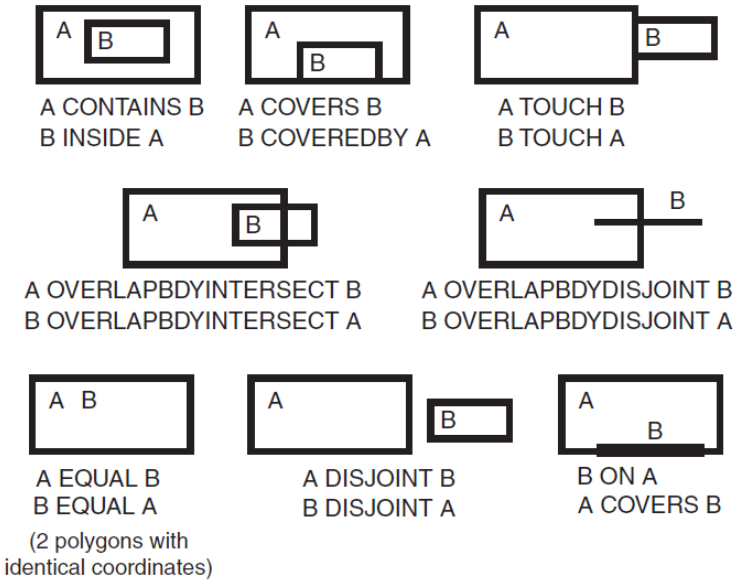
Nebo-li 9-ti průsečíkový model, je model pro kategorizaci binárních topologických vztahů mezi body, liniemi a polygony. Každý prostorový objekt má vnitřní část (interior), hranici (boundary) a vnější část (exterior). Oproti DE – 9IM nepopisuje matice dimenzi průniku. Model je implementován operátorem `SDO_RELATE`.



Obr. 4.5: Topologický vztah TOUCH definovaný 9IM, převzato z [8]

Spatial používá následující názvy topologických vztahů zobrazených na obr. 4.6:

- DISJOINT – hranice a vnitřní části se nekříží.
- TOUCH – hranice se kříží, ale vnitřní části nikoli.
- OVERLAPBDYDISJOINT – vnitřní část jednoho objektu kříží hranici a vnitřní část druhého objektu, ale hranice obou objektů se nekříží.
- OVERLAPBDYINTERSECT – hranice a vnitřní části dvou objektů se kříží.
- EQUAL – dva objekty mají totožné hranice a vnitřní části.
- CONTAINS – hranice a vnitřní část jednoho objektu je zcela obsažena ve vnitřní části druhého objektu.
- COVERS – vnitřní část jednoho objektu je zcela obsažena ve vnitřní části nebo hranici druhého objektu a jejich hranice se kříží.
- INSIDE – opak CONTAINS, z A INSIDE B vyplývá, že B CONTAINS A.
- COVEREDBY – opak COVERS, z A COVEREDBY B vyplývá, že B COVERS A.
- ON – hranice a vnitřní část jednoho objektu je na hranici druhého objektu, pro druhý objekt zase platí vztah COVERS.
- ANYINTERACT – dva objekty nejsou disjunktní.



Obr. 4.6: Topologické vztahy v 9IM, převzato z [8]

Pro větší pohodlí nabízí Spatial operátory, které usnadňují kontrolu topologických vztahů. Tyto operátory jsou ekvivalentním zadáním `SDO_RELATE` s použitím určité masky.

Operátor	**Popis**
SDO_ANYINTERACT	Kontroluje, zda mají vstupující geometrie vztah ANYINTERACT.
SDO_CONTAINS	Kontroluje, zda mají vstupující geometrie vztah CONTAINS.
SDO_COVEREDBY	Kontroluje, zda mají vstupující geometrie vztah COVEREDBY.
SDO_COVERS	Kontroluje, zda mají vstupující geometrie vztah COVERS.
SDO_EQUAL	Kontroluje, zda mají vstupující geometrie vztah EQUAL.
SDO_INSIDE	Kontroluje, zda mají vstupující geometrie vztah INSIDE.
SDO_ON	Kontroluje, zda mají vstupující geometrie vztah ON.
SDO_OVERLAPBDYDISJOINT	Kontroluje, zda mají vstupující geometrie vztah OVERLAPBDYDISJOINT.
SDO_OVERLAPBDYINTERSECT	Kontroluje, zda mají vstupující geometrie vztah OVERLAPBDYINTERSECT.
SDO_OVERLAPS	Kontroluje, zda mají vstupující geometrie vztah OVERLAPS.
SDO_TOUCH	Kontroluje, zda mají vstupující geometrie vztah TOUCH.

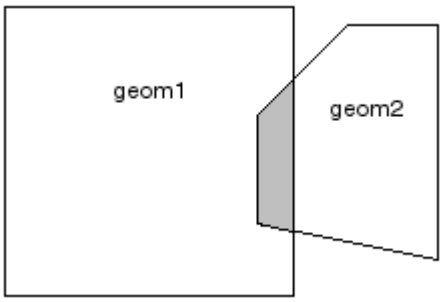
Tab. 4.2: Operátory pro kontrolu topologických vztahů

4.2.6 Prostorové procedury a funkce

V následující podkapitole jsou uvedeny základní prostorové funkce analogické k funkcím uvedených v kapitole 3.2.4.

SDO_GEOM.SDO_INTERSECTION

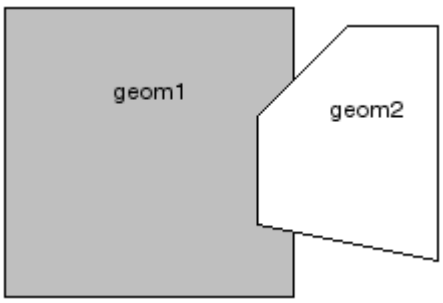
Tato funkce vrací geometrický popis objektu, který vznikne jako prostorový průnik (operace AND) dvou vstupujících geometrií.



Obr. 4.7: Výsledek funkce SDO_GEOM.SDO_INTERSECTION, převzato z [8]

SDO_GEOM.SDO_DIFFERENCE

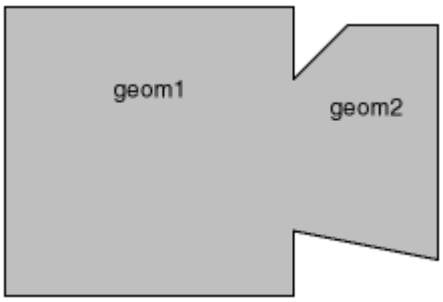
Tato funkce vrací geometrický popis objektu, který vznikne jako prostorový rozdíl (operace MINUS) dvou vstupujících geometrií.



Obr. 4.8: Výsledek funkce SDO_GEOM.SDO_DIFFERENCE, převzato z [8]

SDO_GEOM.SDO_UNION

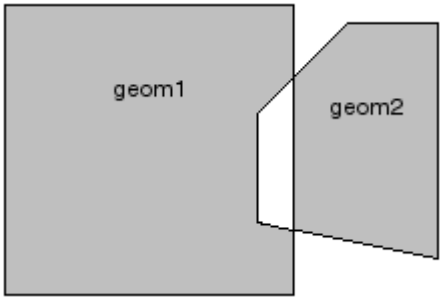
Tato funkce vrací geometrický popis objektu, který vznikne jako prostorové sjednocení (operace OR) dvou vstupujících geometrií.



Obr. 4.9: Výsledek funkce SDO_GEOM.SDO_UNION, převzato z [8]

SDO_GEOM.SDO_XOR

Tato funkce vrací geometrický popis objektu, který vznikne jako symetrický rozdíl (operace XOR) dvou vstupujících geometrií.

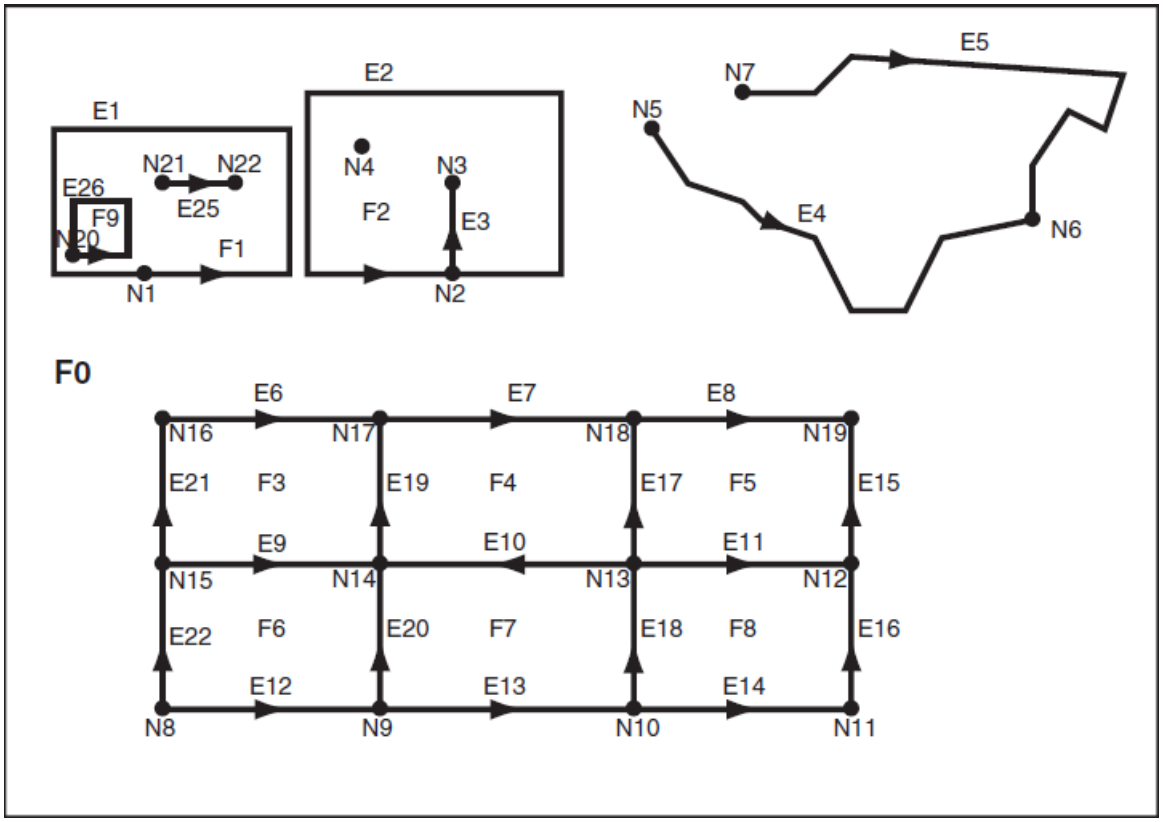


Obr. 4.10: Výsledek funkce SDO_GEOM.SDO_XOR, převzato z [8]

4.3 Koncept topologického datového modelu

Topologie je obor matematiky zabývající se objekty v prostoru. Základními elementy v topologii jsou uzly, hrany a stěny.

Spatial umožňuje ukládání dat do datové struktury Winged Edge, tzv. okřídlené hrany. Základem této struktury jsou hrany. Každé hraně náleží dva uzly (počáteční a koncový), dvě přilehlé stěny (po pravé a po levé straně ve směru hrany) a čtyři přilehlé hrany. [9]



Obr. 4.11: Topologické prvky, převzato z [9]

Obrázek zobrazuje zjednodušenou topologii uzlů, hran a stěn.

- E elementy (E1, E2, ...) jsou orientované hrany, F elementy (F1, F2, ...) jsou stěny a N elementy (N1, N2, ...) jsou uzly.
- ***F0*** je vytvořena pro každou topologii. Je to univerzální stěna obsahující vše v topologii. Stěna F0 nemá žádnou geometrii a má negativní identifikátor ID = -1.
- Uzel je vytvořen pro každý bod geometrie, stejně tak jako pro konečný a počáteční uzel hrany. Např. F1 má pouze uzavřenou hranu E1, ta má stejný počáteční a koncový uzel N1. F1 má také hranu E25 s počátečním uzlem N21 a koncovým N22.

- ***Izolovaný uzel*** (nazývaný *island node*) je uzel izolovaný uvnitř stěny. Příkladem je N4 v F2. V praxi izolovaným uzlem mohou být definiční body parcel.
- ***Izolovaná hrana*** (nazývaná *island edge*) je hrana izolovaná uvnitř stěny. Příkladem je E25 v F1.
- ***Smyčka*** je hrana, která má totožný počáteční a koncový uzel. Příkladem je E1.
- Hrana nemůže obsahovat *island node*. Hrana může být rozdělena na dvě hrany přidáním uzlu na hranu.
- Informace o topologických vztazích je uložena ve speciálních tabulkách.

4.3.1 Datový typ SDO_TOPO_GEOMETRY

SDO_TOPO_GEOMETRY je hlavním datovým typem v topologickém datovém modelu. Popisuje geometrii topologie v jediném řádku a jediném sloupci typu SDO_TOPO_GEOMETRY v uživatelem definované tabulce.

```
CREATE TYPE sdo_topo_geometry AS OBJECT(
   tg_type NUMBER,
   tg_id NUMBER,
   tg_layer_id NUMBER,
   topology_id NUMBER);
```

Atribut	Popis
TG_GTYPE	Obsahuje typ topologického prvku 1 – bod, 2 – linie, 3 – polygon a 4 – různorodá kolekce.
TG_ID	Unikátní identifikátor (generuje Spatial) pro geometrii topologického prvku.
TG_LAYER_ID	Identifikátor topologické vrstvy, které topologický prvek náleží (ID generuje Spatial a je unikátní v rámci topologické vrstvy).
TOPOLOGY_ID	Identifikátor (generuje Spatial) pro topologii.

Tab. 4.3: Popis datového typu SDO_TOPO_GEOMETRY

Každý topologický prvek je v topologii jednoznačně určen kombinací hodnot TG_ID a TG_LAYER_ID.

Tolerance je použita pro přiřazení stupně přesnosti k prostorovým datům. Určuje vzdálenost, ve které mohou být dva body od sebe a stále budou považovány za jeden, například chyby ze zaokrouhlení. Hodnota tolerance musí být kladné číslo větší než nula. V modelu topologie může mít tolerance dva významy v závislosti na operaci, která probíhá. První význam je tradiční Oracle Spatial definice o toleranci, ta bývá zadávána během procedury `CREATE_TOPO.CREATE_TOPOLOGY`. Druhý

význam tolerance je pevná hodnota 10e-15 použitá pro interní výpočty během procedury `CREATE_TOPO-MAP.VALIDATE_TOPOLOGY`. [9]

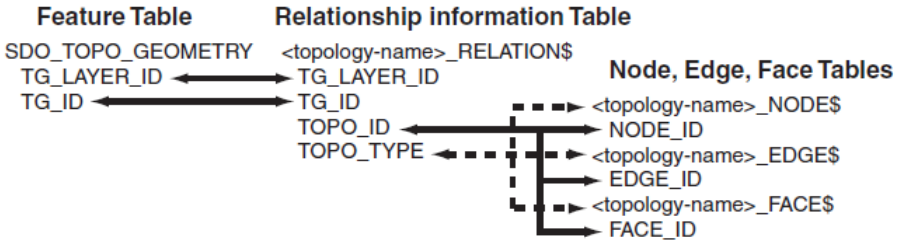
Topologický prvek, tzv. *topology geometry features*, je prostorové zobrazení reálného objektu. Geometrie je uložena jako soubor topologických prvků (uzel, hrana,stěna), tzv. primitiv. Každý topologický prvek má unikátní ID, který přiřazuje Spatial při importu dat.

Vrstva topologických prvků, tzv. *topology geometry layer*, se skládá z množiny topologických prvků. Data pro každou topologickou vrstvu jsou uložena v tabulce prvků.

Každý topologický prvek je definován jako objekt typu SDO_TOPO_GEOMETRY. Metadata o topologii automaticky spravuje Spatial v pohledech USER_SDO_TOPO_METADATA a ALL_SDO_TOPO_METADATA.

4.3.2 Tabulky topologického modelu

K využití schopností topologie ve Spatial musí být nejprve vložena data do speciálních tabulek uzlů, hran a stěn, které jsou vytvořeny při založení topologie. Princip ukládání dat a propojení mezi tabulkami prvků a tabulkami topologických primitiv je zobrazeno na obrázku 4.12.



Obr. 4.12: Znázornění propojení tabulek, převzato z [9]

Jak je z obrázku patrné, k propojení prvkových tabulek a tabulek primitiv dochází pomocí `<topology_name>_RELATION$`.
Platí:

- Každá tabulka prvků obsahuje sloupec typu SDO_TOPO_GEOMETRY, tento typ obsahuje atributy TG_LAYER_ID a TG_ID. Hodnoty v těchto sloupcích jsou identické s hodnotami ve sloupcích TG_LAYER_ID, TG_ID v tabulce `<topology_name>_RELATION$`.
- Každý prvek má jeden nebo více záznamů v tabulce `<topology_name>_RELATION$`.
- Vzhledem k hodnotám atributů TG_LAYER_ID a TG_ID pro prvek je množina uzlů, hran a stěn přiřazených k danému prvku určena pomocí atributu TOPO_ID.

Tabulka uzlů

Informace o uzlech jsou uloženy v tabulce `<topology_name>_NODE$` vytvořené během procedury `SDO_TOPO.CREATE_TOPOLOGY`.

Atribut	Datový typ	Popis
NODE_ID	NUMBER	Identifikátor uzlu.
EDGE_ID	NUMBER	Identifikátor (se znaménkem) hrany přidružené s uzlem.
FACE_ID	NUMBER	Identifikátor stěny, pokud je spojená s uzlem.
GEOMETRY	SDO_GEOMETRY	Geometrický popis objektu reprezentujícího uzel.

Tab. 4.4: Struktura tabulky uzlů

EDGE_ID nebo FACE_ID musí vždy nabývat hodnoty NULL pro každý uzel:

- EDGE_ID nabývá hodnoty NULL, pokud je uzel izolovaný.
- FACE_ID nabývá hodnoty NULL, pokud je uzel počátečním nebo koncovým uzlem hrany (uzel není izolovaný).

Tabulka stěn

Informace o stěnách jsou uloženy v tabulce `<topology_name>_FACE$` vytvořené během procedury `SDO_TOPO.CREATE_TOPOLOGY`.

Atribut	Datový typ	Popis
FACE_ID	NUMBER	Identifikátor stěny.
BOUNDARY_EDGE_ID	NUMBER	Identifikátor hraniční hrany pro stěnu. Znaménko indikuje orientaci, která je použita pro hraniční komponenty. (Kladný identifikátor indikuje směr nalevo od hrany a záporný napravo od hrany.)
ISLAND_EDGE_ID_LIST	SDO_LIST_TYPE	Izolovaná hrana stěny.
ISLAND_NODE_ID_LIST	SDO_LIST_TYPE	Izolovaný bod stěny.
MBR_GEOMETRY	SDO_GEOMETRY	Minimální ohraničující obdélník, který uzavírá stěnu. MBR musí být uložen jako optimální obdélník (jsou uvedeny pouze souřadnice pro levý dolní roh a pravý horní roh). Nad tímto sloupcem je vytvořen prostorový index.

Tab. 4.5: Struktura tabulky stěn

SDO_LIST_TYPE je definován takto:

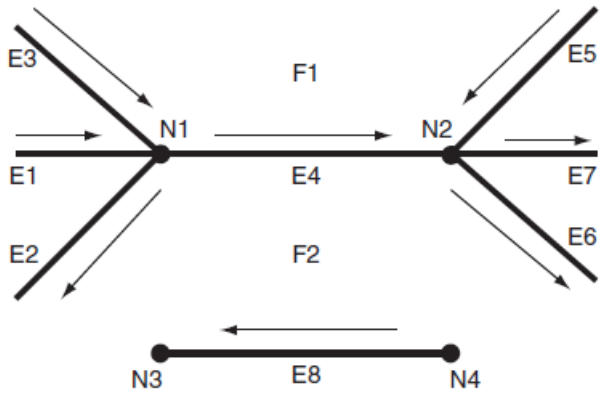
```
CREATE TYPE sdo_list_type as VARRAY(2147483647) OF NUMBER;
```

Tabulka hran

Informace o hranách jsou uloženy v tabulce `<topology_name>_EDGE$` vytvořené během procedury `SDO_TOPO.CREATE_TOPOLOGY`. Hodnoty atributů NEXT_LEFT_EDGE_ID a NEXT_RIGHT_EDGE_ID odkazují na následující hrany, na které narazíme v proti směru hodinových ručiček po obvodu levé a pravé stěny. Hodnoty atributů PREV_LEFT_EDGE_ID a PREV_RIGHT_EDGE_ID odkazují na předcházející hrany, na které narazíme v proti směru hodinových ručiček po obvodu levé a pravé stěny. Hodnota atributu LEFT_FACE_ID odkazuje na stěnu nalevo v kladném směru orientace hrany. Hodnota atributu RIGHT_FACE_ID odkazuje na stěnu napravo v kladném směru orientace hrany. Pro všechny numerické hodnoty identifikátoru znaménko indikuje orientaci odkazovaných prvků.

Atribut	Datový typ	Popis
EDGE_ID	NUMBER	Identifikátor hrany.
START_NODE_ID	NUMBER	Identifikátor počátečního uzlu hrany.
END_NODE_ID	NUMBER	Identifikátor koncového uzlu hrany.
NEXT_LEFT_EDGE_ID	NUMBER	Identifikátor (včetně znaménka) následující levé hrany.
PREV_LEFT_EDGE_ID	NUMBER	Identifikátor (včetně znaménka) předcházející levé hrany.
NEXT_RIGHT_EDGE_ID	NUMBER	Identifikátor (včetně znaménka) následující pravé hrany.
PREV_RIGHT_EDGE_ID	NUMBER	Identifikátor (včetně znaménka) předcházející pravé hrany.
LEFT_FACE_ID	NUMEBER	Identifikátor stěny ležící nalevo od hrany.
RIGHT_FACE_ID	NUMBER	Identifikátor stěny ležící napravo od hrany.
GEOMETRY	SDO_GEOMETRY	Geometrický popis objektu reprezentující hranu. Seznam souřadnic jde v přirozeném řádu pro kladný směr hrany.

Tab. 4.6: Struktura tabulky hran



Obr. 4.13: Vztahy mezi primitivy v tabulce hran, převzato z [9]

EDGE_ID	START_NODE_ID	END_NODE_ID	NEXT_LEFT_EDGE_ID	PREV_LEFT_EDGE_ID	NEXT_RIGHT_EDGE_ID	PREV_RIGHT_EDGE_ID	LEFT_FACE_ID	RIGHT_FACE_ID
E4	N1	N2	-E5	E3	E2	-E6	F1	F2
E8	N4	N3	-E8	-E8	E8	E8	F2	F2

Obr. 4.14: Tabulka vztahů primitiv, převzato z [9]

- Pro hranu E4 je počátečním uzlem N1 a koncovým N2. Následující levá hrana je E5, ale její orientace je opačná než orientace hrany E4, a proto je NEXT_LEFT_EDGE uložena jako -E5.
- Předchozí levá hrana pro hranu E4 je E3, protože je stejně orientovaná jako hrana E4, bude PREV_LEFT_EDGE uložena jako E3.
- Pravá stěna je určena opačnou orientací hrany E4. To může být chápáno jako obrácení směru hrany E4, tudíž postup je stejný jako u levé stěny. V tomto případě NEXT_RIGHT_EDGE je hrana E2 a PREV_RIGHT_EDGE je hrana -E6 (opačná orientace než obrácený směr E4). Pro hranu E4 je levá stěna F1 a pravá F2.
- Hrany E1 a E7 nejsou hrany nejvíce vlevo, ani nejvíce vpravo vzhledem k hraně E4, a proto nebudou uloženy v tabulce hran pro záznam hrany E4.

Tabulka vztahů

Informace o vztazích (relacích) jsou uloženy v tabulce `<topology_name>_RELATION$` vytvořené během procedury `SDO_TOPO.CREATE_TOPOLOGY`. Jak je vidět na obr. 4.12, tabulka relací propojuje topologické tabulky primitiv s tabulkami obsahující atribut typu SDO_TOPO_GEOMETRY.

Atribut	Datový typ	Popis
TG_LAYER_ID	NUMBER	Identifikátor topologické vrstvy, které topologický prvek náleží.
TG_ID	NUMBER	Identifikátor topologického objektu.
TOPO_ID	NUMBER	Identifikátor topologického elementu v topologickém objektu. Pro topologii, která má hierarchické uspořádání, vloží hodnotu Spatial.
TOPO_TYPE	NUMBER	Pro topologii bez hierarchického uspořádání vrstev: 1 = uzel, 2 = hrana, 3 = stěna. Pro topologii, která má hierarchické uspořádání, vloží hodnotu Spatial.
TOPO_ATRIBUTE	VARCHAR2	Hodnotu vloží Spatial.

Tab. 4.7: Struktura tabulky vztahů

4.3.3 Editace topologie

Editace topologických dat se ve Spatial provádí výhradně za pomoci vyrovnávací paměti, nebo-li TopoMap objektu. Při editaci je nutné vždy použít buď PL/SQL API, nebo Java API, nikdy neupravovat přímo topologické tabulky uzlů, hran, stěn, nebo vztahů. Existují dva přístupy jak vytvořit mezipaměť potřebnou k editaci.[9]

- Explicitní vytvoření mezipaměti (lze používat PL/SQL API i Java API).
- Automatické vytvoření mezipaměti pomocí Spatial(lze používat PL/SQL API).

Topologický model dat API zahrnuje:

- PL/SQL funkce a procedury v balíčcích SDO_TOPO a SDO_TOPO_MAP,
- PL/SQL topologické operátory,
- Java API.

Pro topologický model lze použít stejné operátory jako pro Spatial kromě:

- `SDO_RELATE` – v tomto případě lze nahradit vylepšením (viz. tab.4.2) nebo lze použít `SDO_TOPO.RELATE`.
- `SDO_NN`
- `SDO_NN_DISTANCE`
- `SDO_WITHIN_DISTANCE`

TopoMap objekt

Jedná se o objekt, jenž je asociován s vyrovnávací pamětí, která je asociována s topologií. Explicitní vytvoření mezipaměti zahrnuje více kroků než jednodušší automatické provedení pomocí Spatial, přesto je mnohem rychlejší a efektivnější pro většinu editovaných topologických relací. Ty mohou zahrnovat stovky nebo tisíce topologických elementů. Při explicitním vytváření a používání mezipaměti pro editaci jsou nutné následující kroky:

- Vytvořit TopoMap objekt asociovaný s topologií.
- Načíst celou nebo část topologie do paměti.
- Editovat objekty.
- Pravidelně aktualizovat topologii k zápisu změn v databázi.
- Umístit změny vytvořené v mezipaměti.
- Vyčistit mezipaměť.

Tento přístup je uveden ve většině návodů a ilustrací. TopoMap objekt může být aktualizovatelný, nebo pouze ke čtení. To záleží na parametru *allow_ updates*, který je volán při proceduře `SDO_TOPO_MAP.LOAD_TOPO_MAP`.

Dva uživatelé mohou editovat stejnou topologii ve stejný čas tak dlouho, dokud se jejich editační okna nepřekrývají.

4.4 Postup práce s daty

Vytvoření topologie z prostorových dat
Pro založení topologie z prostorových dat je nutné nejprve provést standardní operace přípravy dat pro použití ve Spatial, jak je popsáno v krocích 1 – 5.

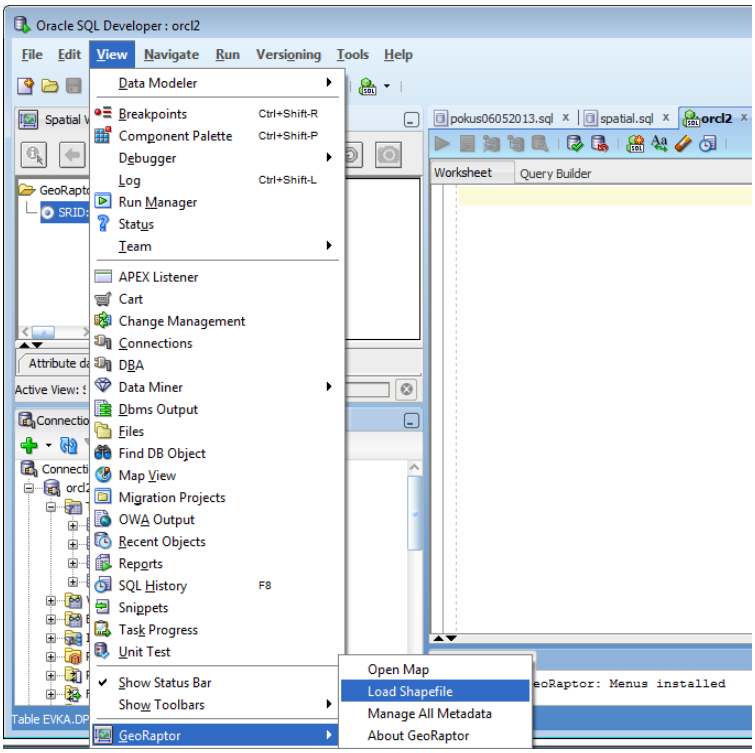
1. Založení prostorové tabulky.
2. Aktualizace prostorových metadat (pohledu USER_SDO_GEOM_METADATA).
3. Načtení dat do prostorové tabulky.
4. Validace prostorových dat.
5. Vytvoření prostorových indexů.
6. Založení topologie pomocí procedury `CREATE_TOPO.CREATE_TOPOLOGY`.
7. Vytvoření *universe face F0*.
8. Založení prvkových tabulek *feature table* pro každý typ topologické geometrie vrstev.
9. Přidružení tabulek k topologii pomocí procedury `SDO_TOPO.ADD_TOPO_GEOMETRY_LAYER`.
10. Vytvoření objektu TopoMap a načtení celé topologie.
11. Načtení *feature table*, vložení dat z prostorových tabulek a použití funkce `SDO_TOPO_MAP.CREATE_FEATURE`.
12. Inicializace topologických metadat pomocí procedury `SDO_TOPO.INITIALIZE_METADATA`.
13. Dotazy nad topologickými daty.
14. Editace topologických dat.

4.4.1 Načtení dat

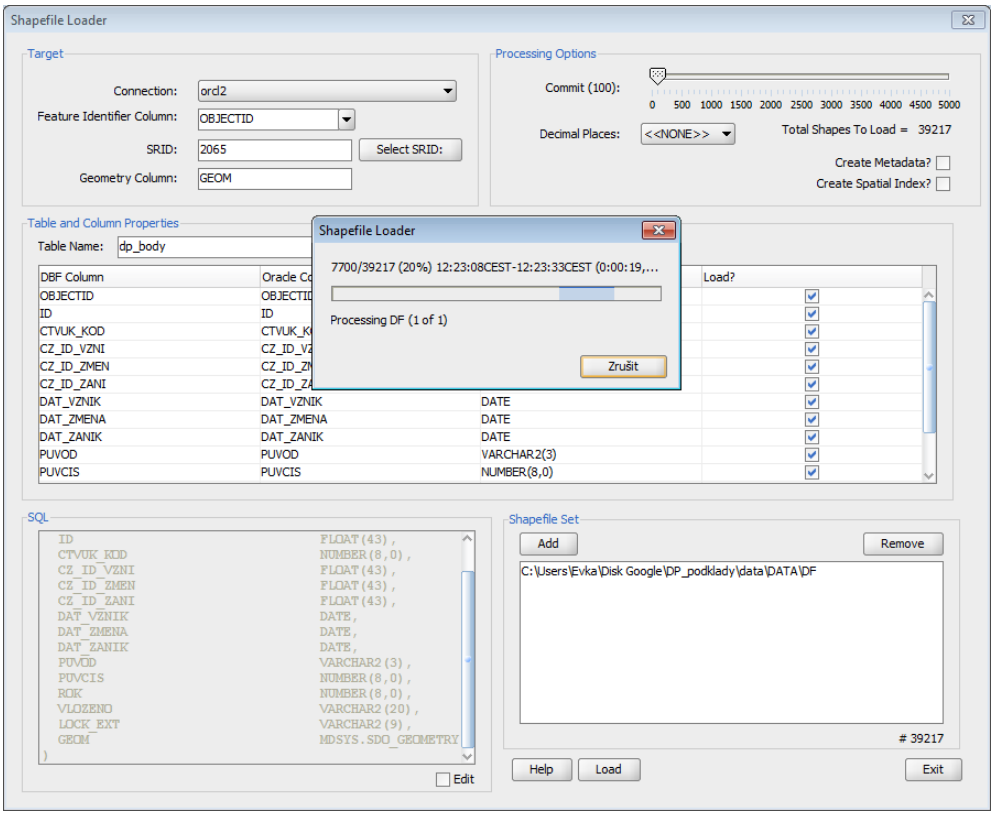
Prostorová data byla do Oracle Spatial načtena z formátu *.shp pomocí nástroje GeoRaptor, podrobnosti o použitých datech jsou uvedeny v kapitole 5. **GeoRaptor** je přídavný modul pro SQL Developer. Jedná se o open source projekt vyvíjený od roku 2006, pomocí kterého je možné zobrazovat a spravovat prostorová data. Umožňuje vizualizovat 2D data, nad kterými je vytvořený prostorový index.

Na obr. 4.15 je znázorněno spouštění funkce *Load shapefile*. Na obr. 4.16 je ukázka nastavení importu dat, jako je referenční systém EPSG: 2065, název sloupce, ve kterém bude uložena geometrie, a název sloupce obsahující identifikátor.

Tímto způsobem byla načtena data do tabulek dp_body, dp_ku, dp_linie, dp_plochy.



Obr. 4.15: Spuštění nástroje Georaptor pro načtení ∗.shp



Obr. 4.16: Import dat z formátu ∗.shp do Spatial databáze

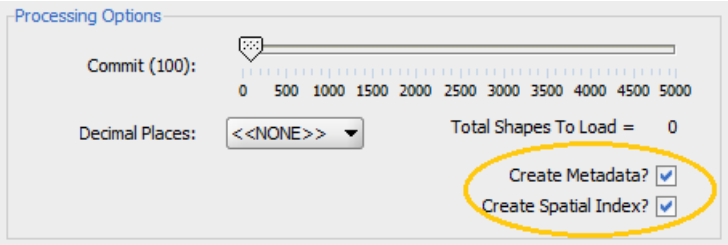
4.4.2 Aktualizace metadat

Aktualizace metadat prostorových dat je nutná k možnosti vytvoření prostorových indexů. Tuto aktualizaci je možné provést níže uvedeným dotazem, nebo pomocí nástroje GeoRaptor.

Všechna metadata byla aktualizována pomocí uvedeného dotazu.

```
INSERT INTO user_sdo_geom_metadata VALUES (
   'dp_ku',
   'geom',
   SDO_DIM_ARRAY(
   SDO_DIM_ELEMENT('X', -745000, -741000, 0.005), -- max a~min sour.
   SDO_DIM_ELEMENT('Y', -1042000, -1038000, 0.005) -- 0.005 tolerance
   ),
2065 -- SRID);
```

Na obr. 4.17 je znázorněna možnost nastavení vytvoření metadat pomocí GeoRaptor.



Obr. 4.17: Možnost aktualizace metadat a vytvoření indexů v nastavení GeoRaptor

4.4.3 Validace

Validita byla kontrolována pomocí `SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT`

```
SELECT SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT(geom , 0.005) FROM dp_ku;

SELECT count(*)
FROM dp_linie
WHERE SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT(geom , 0.005)<> 'TRUE';
```

V případě zjištění, že se jedná o nevalidní geometrii, byla vytvořena tabulka DP_VAL_RESULTS a naplněna nevalidní geometrií:

```
CREATE TABLE dp_val_results (
            sdo_rowid ROWID,
            RESULT VARCHAR2(2000)
);
```

```
CALL
SDO_GEOM.VALIDATE_LAYER_WITH_CONTEXT('dp_linie','geom', 'DP_VAL_RESULTS');
```

Validovaná data	Výsledek validace	Popis
DP_KU	TRUE	
DP_BODY	TRUE	
DP_LINIE	129 FALSE	ORA:13356
DP_PLOCHY	247 FALSE	ORA:13356 (110) a ORA:13349 (137)

Tab. 4.8: Kontrola validace dat

Oprava zjištěných nevalidních dat byla provedena následovně:

```
UPDATE dp_linie SET geom = SDO_UTIL.REMOVE_DUPLICATE_VERTICES(geom, 0.005);

UPDATE dp_plochy SET geom = SDO_UTIL.RECTIFY_GEOMETRY(geom, 0.005);
```

`SDO_UTIL.REMOVE_DUPLICATE_VERTICES` funkce odstraní redundantní vrcholy z geometrie (ORA:13356 – adjacent points in a geometry are redundant).

`SDO_UTIL.RECTIFY_GEOMETRY` funkce odstraní jak redundantní vrcholy (ORA:13356), tak vyřeší i hranice polygonů, které kříží samy sebe (ORA:13349 – polygon boundary crosses itself). Tato funkce rovněž opravuje nekorektní orientaci řetězců polygonu.

4.4.4 Vytvoření prostorových indexů

Stejně jako aktualizaci metadat, tak i prostorové indexy lze vytvořit dvěma způsoby: pomocí dotazu, nebo pomocí nástroje GeoRaptor při importu dat, jak je vidět na obr. 4.17. Prostorový index nemůže být vytvořen bez aktualizace metadat. Naopak metadata lze vytvořit bez vytvoření indexu.

Všechny prostorové indexy byly vytvořeny pomocí dotazu:

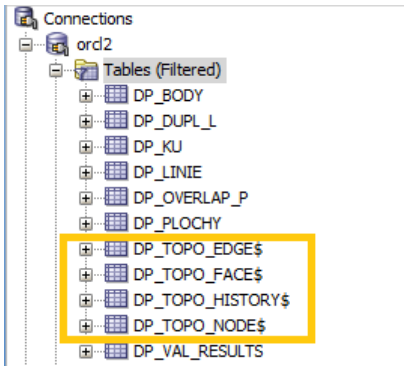
```
CREATE INDEX dp_ku_idx
 ON dp_ku(geom)
 INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

4.4.5 Převedení prostorových dat do topologické struktury

Založení topologie DP_TOPO

```
EXECUTE SDO_TOPO.CREATE_TOPOLOGY('DP_TOPO',0.005,2065);
```

Během procedury jsou vytvořeny topologické tabulky, znázorněny na obr. 4.18.



Obr. 4.18: Topologické tabulky vytvořené během procedury

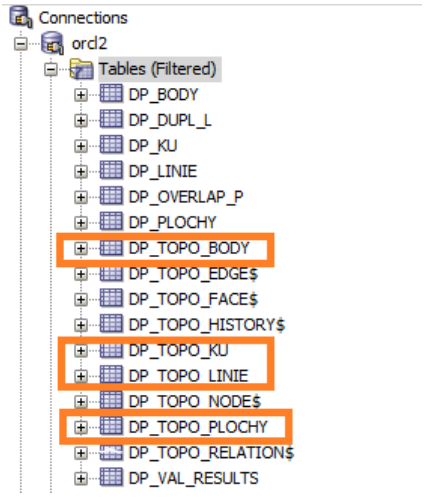
Vytvoření *universal face* F0

```
INSERT INTO dp_topo_face\$ values (
 -1,
 NULL,
 SDO_LIST_TYPE(),
 SDO_LIST_TYPE(),
 NULL);

COMMIT;
```

Vytvoření prvkových tabulek obsahující sloupec typu TopoGeometry

```
CREATE TABLE DP_TOPO_KU (
OBJECTID VARCHAR2(30) PRIMARY KEY,
feature SDO_TOPO_GEOMETRY);
```

Pomocí dotazu byly vytvořeny tabulky zobrazené na obr. 4.19.



Obr. 4.19: Tabulky prvků obsahující sloupec typu TopoGeometry

Vytvoření TopoMap objektu

```
EXECUTE SDO_TOPO_MAP.CREATE_TOPO_MAP('DP_TOPO', 'DP_TOPOMAP');
```

Načtení dat z prostorových tabulek do topologické struktury

```
BEGIN
  FOR ku_rec IN (SELECT objectid, geom FROM dp_ku) LOOP
   INSERT INTO dp_topo_ku VALUES(ku_rec.objectid,
     SDO_TOPO_MAP.CREATE_FEATURE('DP_TOPO', 'DP_TOPO_KU', 'FEATURE',
         ku_rec.geom));
  END LOOP;
END;
```

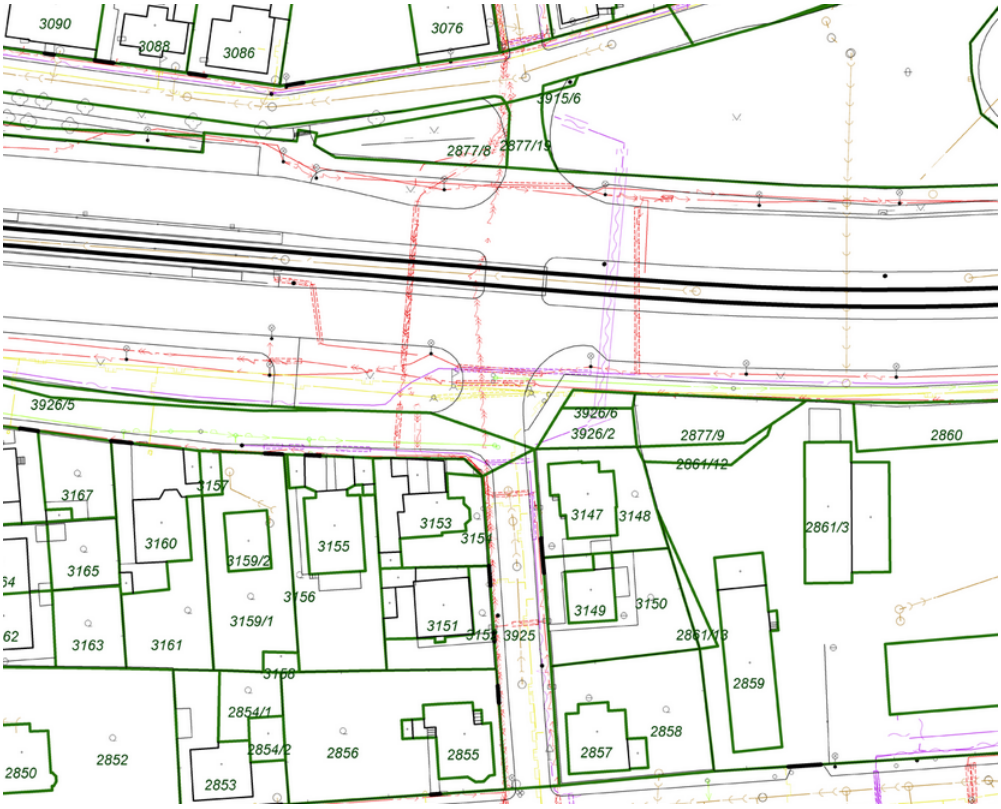
5 Použitá data

Data použitá v této práci byla poskytnuta společností T–MAPY s.r.o., která má na starosti dílo **Převod dat Jednotné digitální mapy Prahy do Digitální mapy Prahy**.

„Digitální mapa Prahy (DMP) je základní polohopisné mapové dílo popisující území hl. m. Prahy, které navazuje na zpracování Jednotné digitální mapy Prahy a Digitální referenční mapu Prahy zajišťované do roku 2007.“ [33]

Datový model DMP tvoří:

- kombinace DOKM pro katastrální území bez vyhlášené DKM a DKM na území s vyhlášenou platností,
- DTM – Digitální Technická Mapa.



Obr. 5.1: Ukázka DMP, převzato z [27]

5.1 Historie

Pro zajištění tvorby a údržby ***Jednotné digitální mapy Prahy*** (JDMP) vznikla v roce 1991 příspěvková organizace hl. m. Prahy IMIP (Institut městské informatiky

Praha). Jejím úkolem byla správa datových bází a pomoc při evidenci městského nemovitého majetku.

Jako podklady sloužily zapůjčené originály katastrální mapy na hliníkové fólii. Ty byly skenovány na stolovém skeneru s rozlišovací schopností 508DPI. Rastrová data ve formátu ∗.cit byla připojena do souřadnicového systému S – JTSK a transformována na hektometrické křížky pomocí afinní transformace. Výsledkem byl připojený a transformovaný mapový list ve formátu ∗.ras.

Dalším podkladem byly katastrální mapy na PET – fóliích, ty byly skenovány na válcových skenerech. Z důvodu nižší přesnosti nebyly transformovány, ale pouze souřadnicově připojeny na 2 rohy mapového listu.

Do roku 1995 probíhalo zpracování v technologii MAPA2 v softwaru KOKEŠ společnosti GEPRO, poté následovalo zpracování v technologii MAPA3. V roce 2002 byla JDMP dokončena na celém území města.

Digitální referenční mapy (DRM) byly podkladem pro GIS aplikace sloužící potřebám města a tvořily ucelený systém územní lokalizace. Tyto mapy vznikaly ve dvou etapách DRM93 a DRM96. V první etapě byly vytvořeny digitální mapy na celém území skenováním a vektorizací katastrálních map. Pro propojení objektů s existujícími databázemi byly objektům přiřazeny identifikátory.

V letech 1993–1996 probíhala druhá etapa, jednalo se o zpřesnění stávajícího díla pomocí dat z JDMP. Následovala aktualizace z leteckých snímků. Z důvodu dlouhých časových prodlev mezi zaměřením skutečnosti a zanesením nového stavu do katastrálních map byl zaveden dvojí stav map – právní a skutečný. Toto rozdělení bylo významnou změnou mezi etapami. Následná údržba probíhala s roční periodou.

V roce 2007 byla zrušena organizace IMIP a byla vypsána veřejná obchodní soutěž na dílo: Převod dat Jednotné digitální mapy Prahy do Digitální mapy Prahy. Útvar rozvoje hl. m. Prahy pověřil tímto úkolem sdružení firem NESS Czech, s.r.o. a T–MAPY, spol. s r.o.

V průběhu první poloviny roku 2008 byl proveden převod dat JDMP do nového datového modelu a zahájena aktualizace DMP. Dne 28. 7. 2008 byl ukončen výdej dat Jednotné digitální mapy Prahy (JDMP) a zahájen výdej dat Digitální mapy Prahy (DMP). [27]

5.2 Digitální mapa Prahy

Stávající rozsah využití JDMP a DRM a způsoby jejich poskytování nejsou dotčeny zpracováním DMP. Rozdíl mezi JDMP a DMP je oddělení katastrální mapy od technické. Důležitým požadavkem je obsahová kontinuita DMP s dosavadními mapovými díly.

Data DMP slouží jako zdroj pro informační systémy Hlavního města Prahy, především pro mapové aplikace spojené s registry MHMP a externí systémy, jako je ISKN. Aktualizace dat probíhají průběžně z několika zdrojů. **Podklady pro DMP jsou:**

- dokumentace skutečného provedení stavby,
- geometrické plány,
- data správců sítí dopravní a technické infrastruktury,
- podrobné geodetické měření.

Aktualizace mapy probíhají CAD nástroji externím dodavatelem. Následně se data převádí do prostředí Spatial, kde probíhají atributové a topologické kontroly. Ve Spatial také probíhá tvorba odvozených datových vrstev. Distribuce a vizualizace dat je provedena v prostředí ArcGIS Desktop.

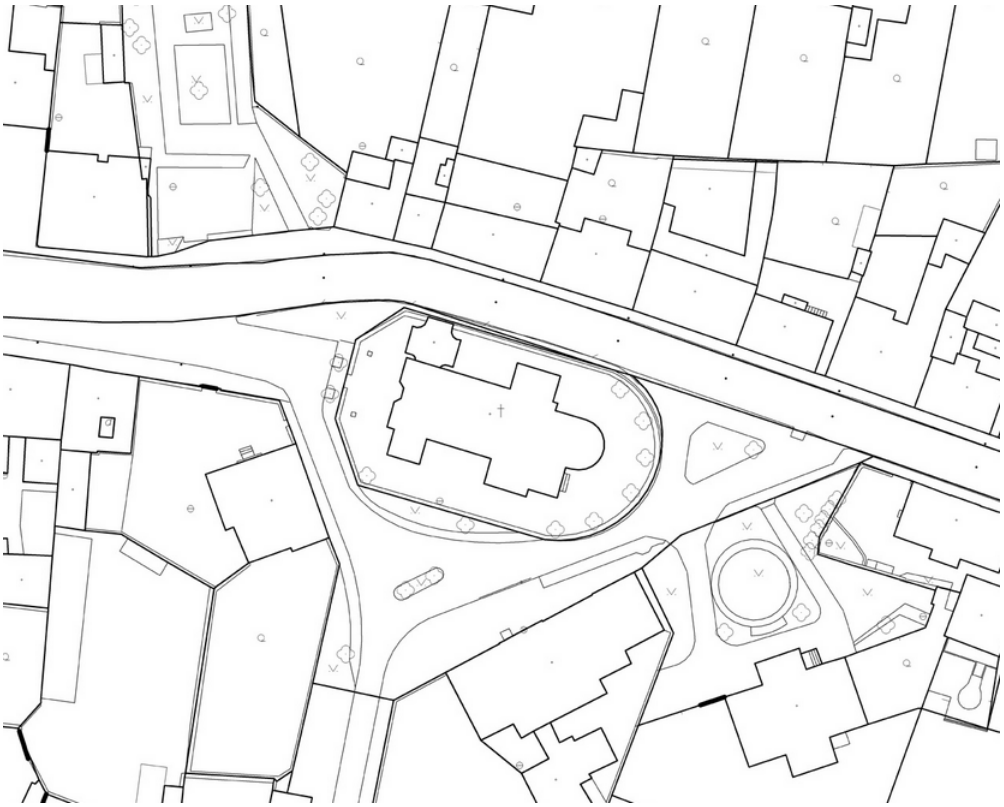
Prvky mapy jsou vybaveny informacemi o původu a datu vzniku. Integrita databáze a čistota datového modelu je zajištěna tím, že systém uchovává i odstraněné prvky, umožňuje tak sledovat historii.

Na úrovni datového modelu DMP jsou vedeny základní metainformace, které popisují samotný stav datového modelu a stav zapracování změnových podkladů. [33]

5.2.1 Mapa technického vyžití území

Poskytnutá data MTVU (Mapa technického využití města) jsou součástí DMP (Digitální mapy Prahy) a vznikla spojením DOKM (Digitální obraz katastrální mapy) a polohopisu TM.

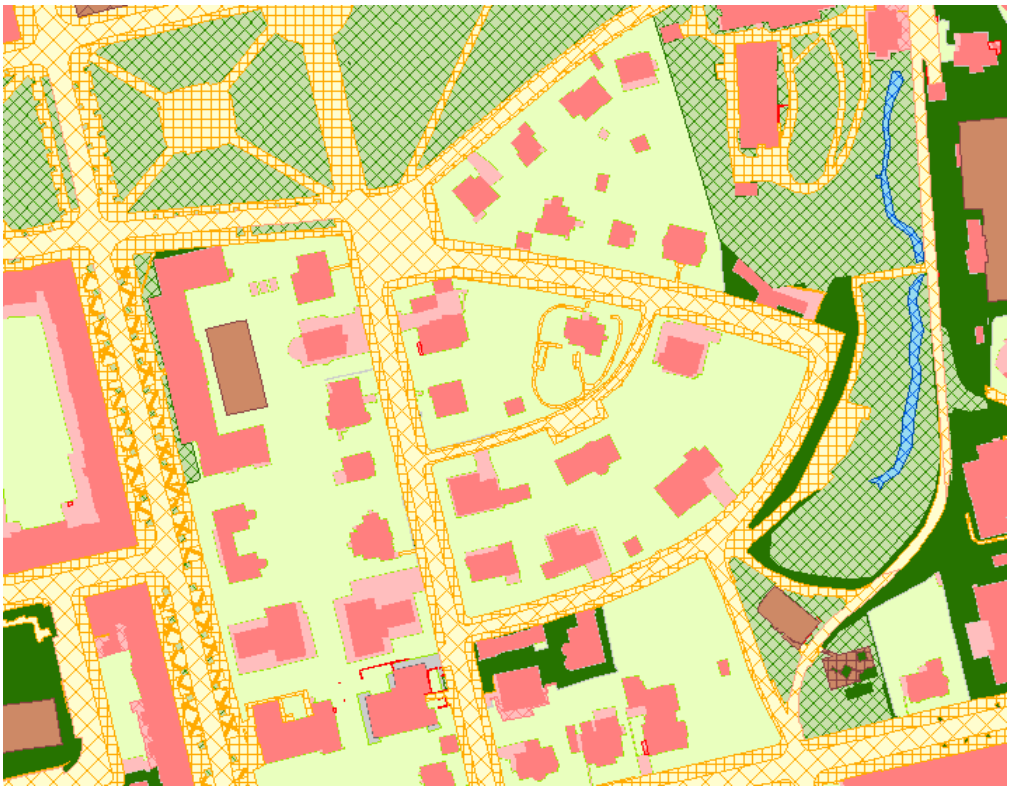
Obr. 5.2: Ukázka DOKM, převzato z [27]



Obr. 5.3: Ukázka polohopisu DTM, převzato z [27]

Jedná se o odvozenou vrstvu ploch s kódy využití území. Na těchto datech jsou během editace kontrolovány tyto chyby:

- duplicitní kategorie,
- duplicitní linie,
- volné konce,
- nevstupující linie,
- bez kategorie,
- díry v MTVU.



Obr. 5.4: Ukázka dat MTVU v prostředí ArcGIS.

Pro zpracování jsem obdržela data MTVU ve formátu geodatabáze ∗.gdb pro katastrální území Bubeneč. Vrstva linií vznikla z vrstvy ploch pomocí ArcGIS funkce *Polygon To Line*. Volné konce byly vytvořeny smazáním linie id = 17376 (linie 17511 má dva volné konce).
Z geodatabáze byla vyexportována následující data:

- ku.shp
- body.shp
- linie.shp
- plochy.shp

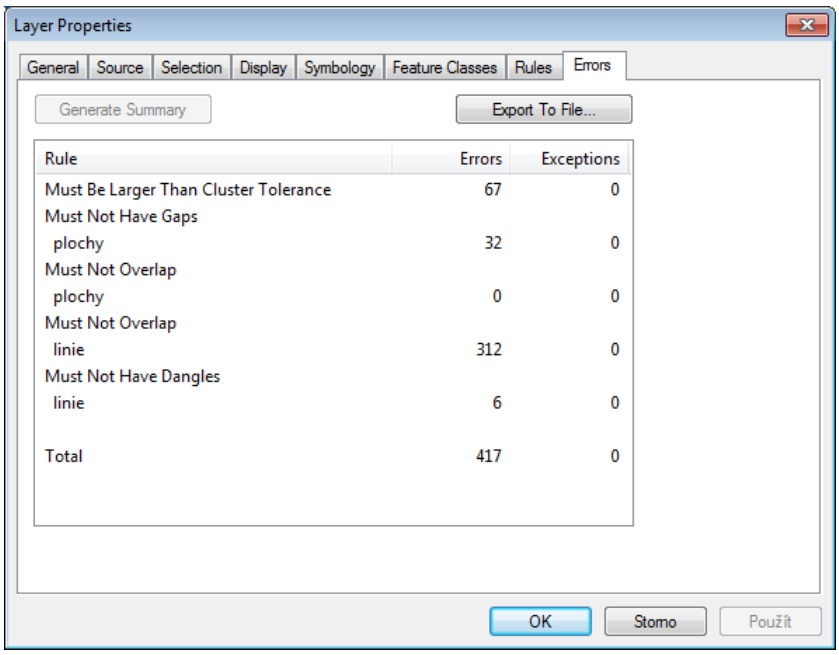
6 Vybrané Topologické operace

6.1 ArcGIS

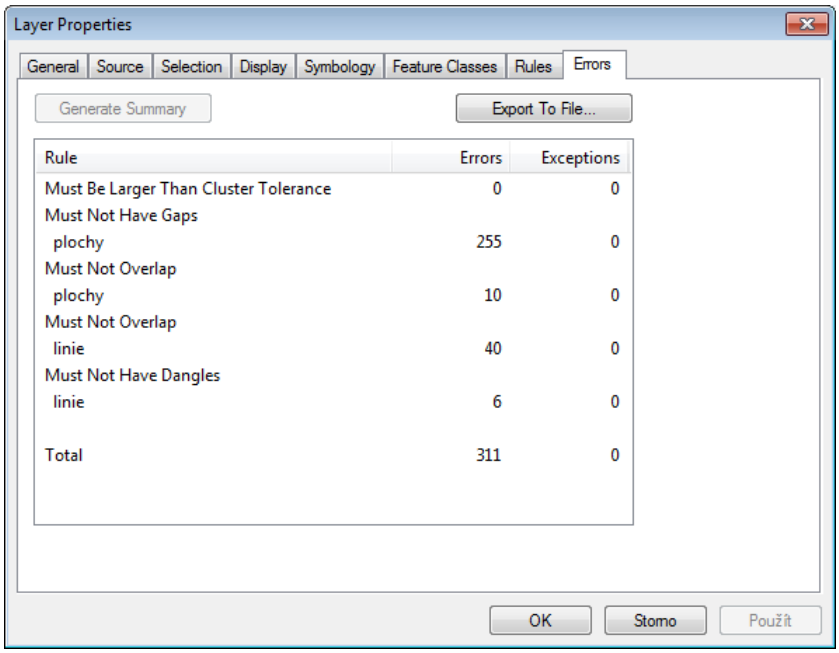
Během zpracování byla použita verze ArcGIS 10.1. V tabulce 6.1 jsou uvedena topologická pravidla aplikovaná na testovacích datech.

Pravidlo dle Esri	**Popis**
Must not have gaps	Polygony v rámci jedné vrstvy nesmí tvořit mezery.
Must not overlap	Polygony se v rámci jedné vrstvy nesmí překrývat.
Must not have dangles	Linie v rámci jedné vrstvy nesmí mít volné konce.
Must not overlap	Linie se v rámci jedné vrstvy nesmí překrývat (duplicitní linie).

Tab. 6.1: Tabulka použitých topologických pravidel dle ESRI



Obr. 6.1: Sumarizace nalezených chyb pro toleranci 0.005



Obr. 6.2: Sumarizace nalezených chyb pro toleranci 0.001

6.2 PostGIS

V geodatabázi PostGIS byly vyzkoušeny následující dotazy pro zjištění stejných chyb, jaké byly nalezeny v ArcGIS. Dotazy byly testovány v PostGIS verzi 2.0.2, PostgreSQL 9.2.

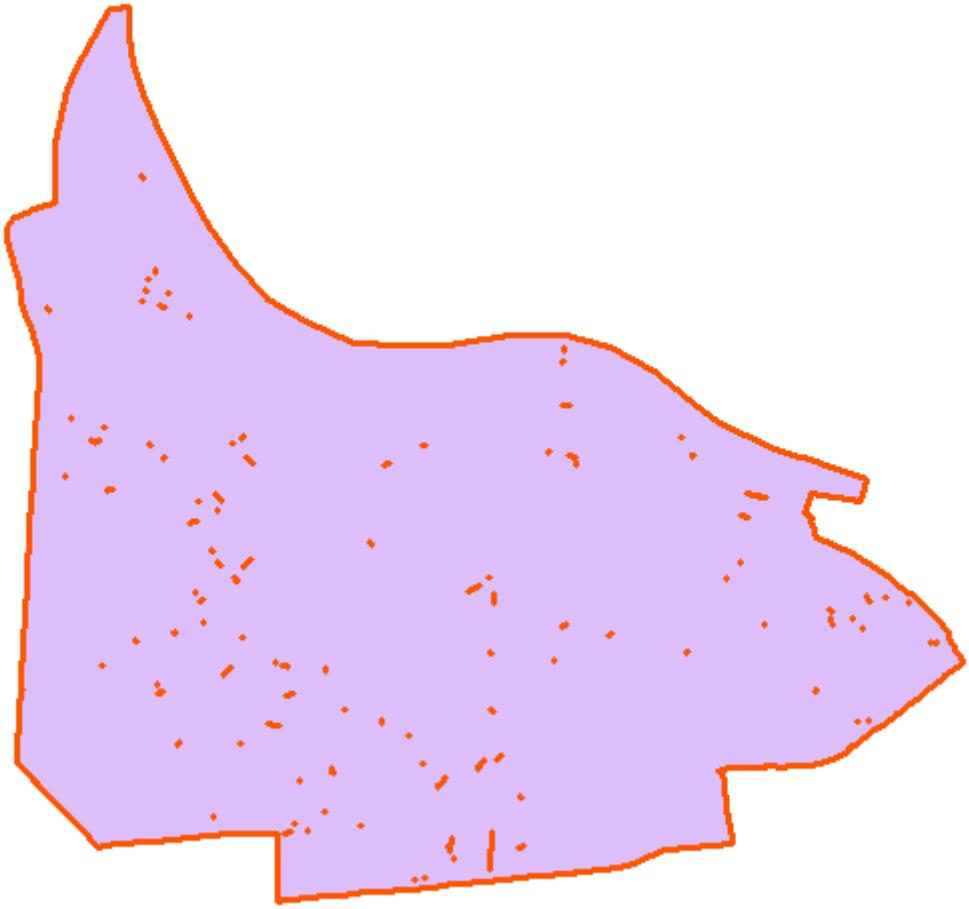
6.2.1 Hledání děr v polygonu

Pro vyhledání děr ve vrstvě polygonů byla vytvořena jedna nová geometrie `plochy_diss`:

```
CREATE TABLE plochy_diss AS
    SELECT ST_Union(geom) AS geom
    FROM plochy
    GROUP BY katuze_kod;
```

Pro zjištění počtu děr v polygonu slouží následující dotaz. Ten vrací počet ringů, které jsou uloženy v geometrii včetně vnějšího ringu.

```
SELECT ST_Nrings(geom) FROM plochy_diss;
```



Obr. 6.3: Polygon vytvořený pro zjištění počtu děr

6.2.2 Hledání duplicitních linií

Nalezení duplicitních linií jsem rozdělila do tří dotazů: totožné linie, linie částečně se překrývající a geometrie jedné linie je obsažena v geometrie druhé linie.

Totožné linie:

```
CREATE TABLE dupl_l2 AS
SELECT l1.gid AS l1_gid, l2.gid AS l2_gid,l1.geom
FROM linie AS l1, linie AS l2
WHERE l1.gid != l2.gid AND ST_Equals(l1.geom,l2.geom)='t';
```

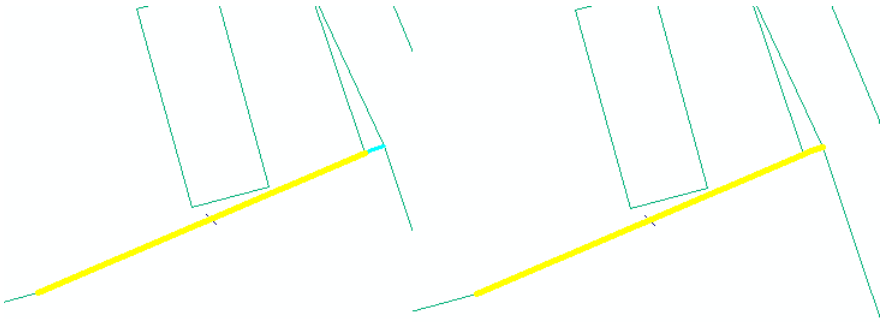
Částečně se překrývající linie:

```
CREATE TABLE dupl_l3 AS
SELECT l1.gid AS l1_gid, l2.gid AS l2_gid,l1.geom
FROM linie AS l1, linie AS l2
WHERE l1.gid != l2.gid AND ST_Overlaps(l1.geom,l2.geom)='t';
```

Obr. 6.4: Linie ve vztahu overlaps

Jedna linie je obsažena v jiné linii:

```
CREATE TABLE dupl_l4 AS
SELECT l1.gid AS l1_gid, l2.gid AS l2_gid,l1.geom
FROM linie AS l1, linie AS l2
WHERE l1.gid != l2.gid AND ST_Covers(l1.geom,l2.geom)='t';
```



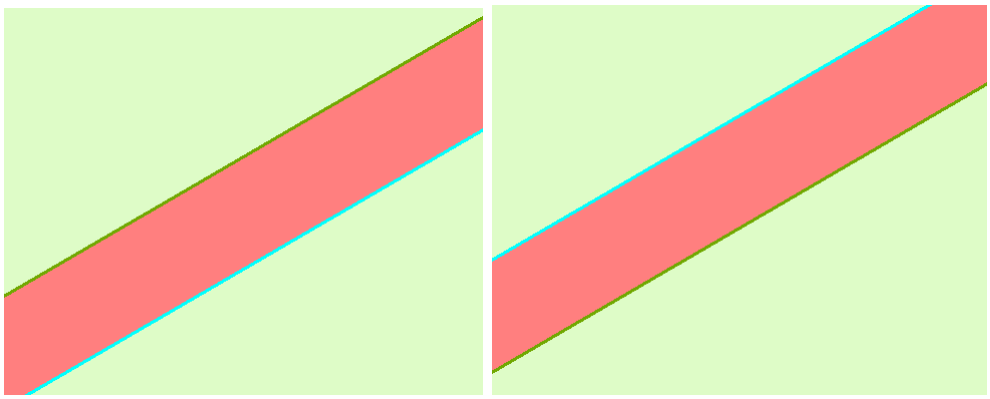
Obr. 6.5: Linie ve vztahu Covers

Výsledky `dupl_l2` a `dupl_l4` jsou až na jednu výjimku stejné. Linie, která nebyla nalezena v `dupl_l2` je zobrazena na obr. 6.5.

6.2.3 Překrývající se polygony

Všechny překrývající se polygony jsou tímto dotazem nalezeny duplicitně.

```
CREATE TABLE overlap_p AS
SELECT p.objectid AS p_id, l.objectid AS l_id,
      ST_CollectionExtract(ST_Intersection(p.geom, l.geom),3) AS geom
FROM plochy AS p, plochy AS l
WHERE p.objectid!= l.objectid AND ST_Overlaps(p.geom, l.geom) ='t';
```

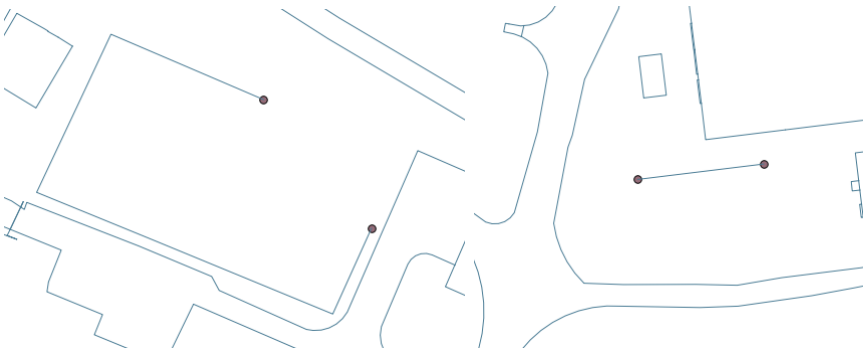


Obr. 6.6: Znázornění správnosti výsledku překrývajících se polygonů

6.2.4 Volné konce linií

Pro vyhledání volných konců byly vyzkoušeny následující dotazy:

```
CREATE VIEW unclosed_linie AS
  SELECT geom,gid
  FROM linie
  WHERE NOT ST_Equals(ST_StartPoint(geom), ST_EndPoint(geom));

CREATE VIEW linie_points AS
(SELECT ST_StartPoint(geom) AS geom,gid
 FROM unclosed_linie UNION ALL SELECT ST_EndPoint(geom),gid AS geom
 FROM unclosed_linie);

CREATE TABLE freepoints AS
 SELECT geom
 FROM linie_points
 GROUP BY geom
  HAVING geom NOT IN (SELECT p.geom FROM linie_points AS p JOIN linie AS l
  ON ST_Dwithin(p.geom,l.geom, 1) AND p.gid <> l.gid);
```



Obr. 6.7: Vyhledání volných konců

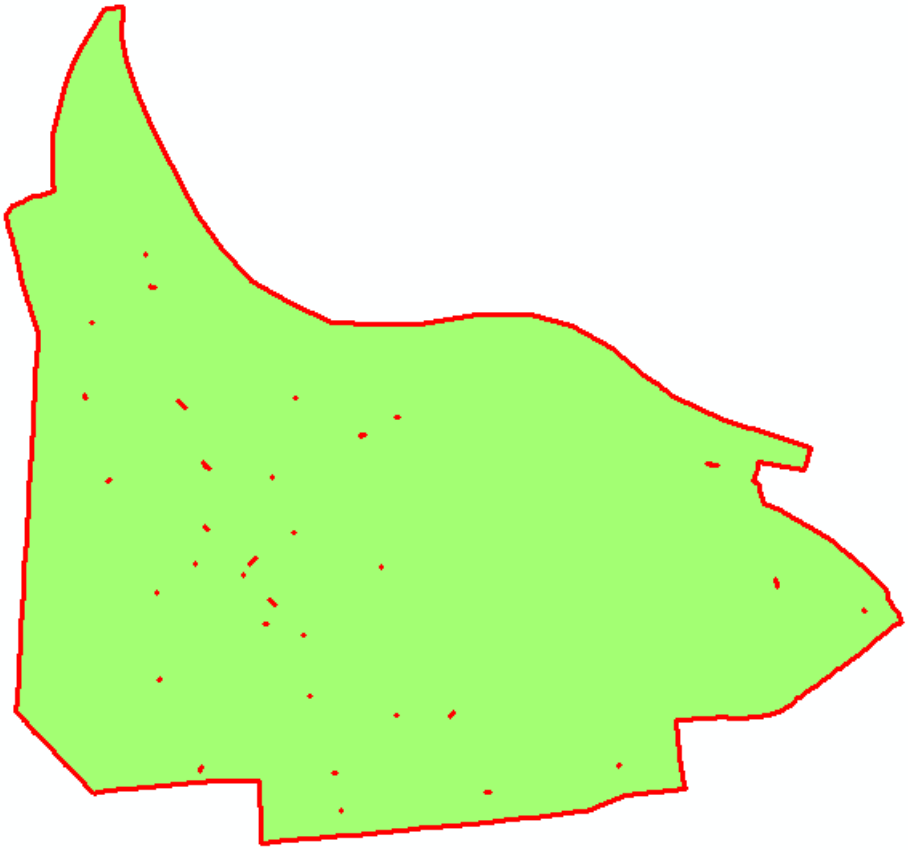
6.3 Spatial

Následující dotazy byly provedeny v Oracle Spatial verzi 11g release 2.

6.3.1 Hledání děr v polygonu

Při testování polygonu, zda obsahuje díry či ne, je nutné nejprve vytvořit jeden polygon pro celé území.

```
 CREATE TABLE diss_plochy_ku AS
  SELECT KATUZE_KOD, SDO_AGGR_UNION(
MDSYS.SDOAGGRTYPE(a.geom, 0.005))GEOM
FROM PLOCHY_DISS_VALID a
GROUP BY a.KATUZE_KOD;
```



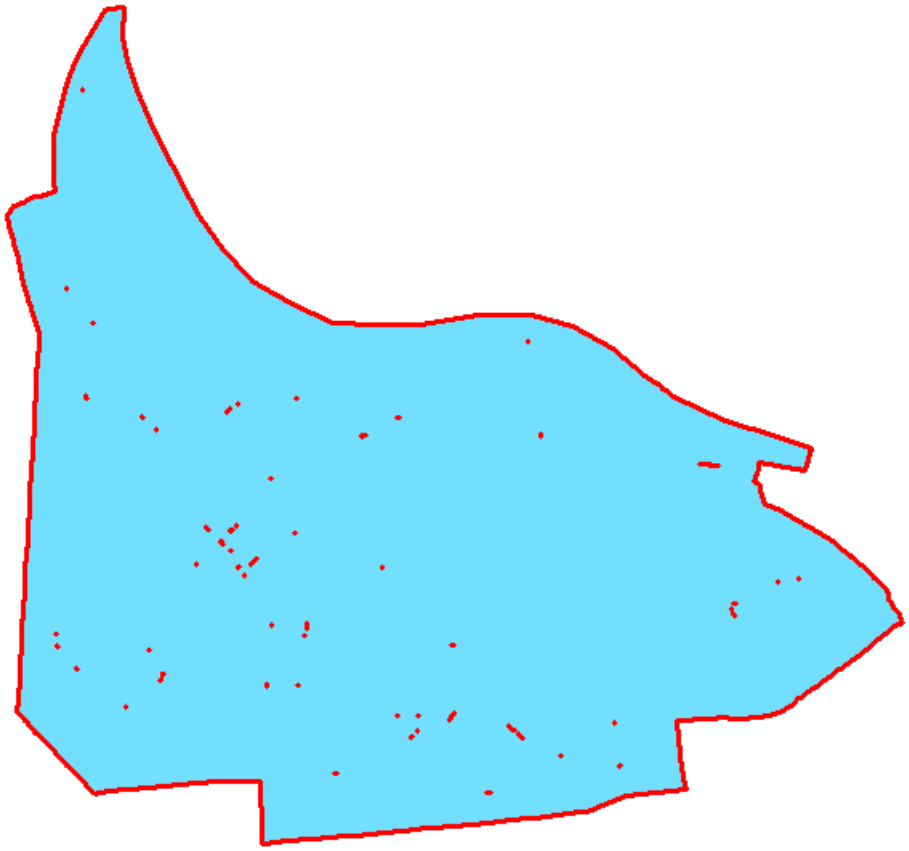
Obr. 6.8: Výsledek dotazu s použitím funkce SDO_AGGR_UNION

Z důvodu dlouhého času trvání dotazu jsem pro sjednocení ploch vyzkoušela následující dotaz, který je mnohem rychlejší, porovnání dotazů je uvedeno v tabulce 6.3:

```
CREATE TABLE dp_diss_p AS
 SELECT KATUZE_KOD, SDO_AGGR_SET_UNION
```

```
                (get_geom_set ('dp_plochy',
                    'geom','CTVUK_KOD <>''1'''), 0.005 ) geom
FROM dp_plochy a
GROUP BY a.KATUZE_KOD;
```

Funkce `get_geom_set()` je uvedena v příloze.



Obr. 6.9: Výsledek dotazu s použitím funkce SDO_AGGR_SET_UNION

Následně můžeme testovat, zda se jedná o polygon obsahující díry či ne.

```
SELECT *
FROM diss_plochy_ku
WHERE SDO_GEOM.SDO_XOR(a.geom,
            SDO_UTIL.EXTRACT(a.geom,1,1),0.005) IS NOT NULL;
```

Pro zjištění počtu děr jsem vytvořila novou geometrii obsahující pouze geometrii děr.

```
CREATE dp_gaps_p1 AS
SELECT SDO_GEOM.SDO_DIFFERENCE(a.geom, b.geom,0.005) geom
FROM dp_ku a, dp_diss_p b;

CREATE TABLE dp_gaps_p2 AS
```

```
SELECT SDO_GEOM.SDO_DIFFERENCE(a.geom, b.geom,0.005) AS geom
FROM dp_ku a, dp_diss_plochy_ku b;
```

Počet děr lze pak zjistit následujícím dotazem:

```
SELECT SDO_UTIL.GETNUMELEM(geom)
FROM dp_gaps_p1;

SELECT SDO_UTIL.GETNUMELEM(geom)
FROM dp_gaps_p2;
```

Dvojí určení děr se neliší pouze časem provádějí dotazu, ale i množstvím zjištěných chyb, jak je uvedeno v tabulce 6.3.

6.3.2 Hledání duplicitních linií

Při hledání duplicitních linií ve Spatial jsem přistupovala stejně jako v PostGIS.

Totožné linie:

```
CREATE TABLE dp_dupl_l2 AS
  SELECT l1.geom, l1.objectid
  FROM dp_linie l1,dp_linie l2
  WHERE l1.objectid <> l2.objectid AND
    SDO_RELATE(l1.geom,l2.geom,'mask=equal querytype=WINDOW')= 'TRUE';
```

Částečně se překrývající linie:

```
CREATE TABLE dp_dupl_l3 AS
  SELECT SDO_GEOM.SDO_INTERSECTION(l1.geom,l2.geom,0.005) geom,
   l1.objectid l1_id, l2.objectid l2_id
  FROM dp_linie l1, dp_linie l2
  WHERE l1.objectid <> l2.objectid AND
        SDO_OVERLAPS(l1.geom,l2.geom)= 'TRUE';
```

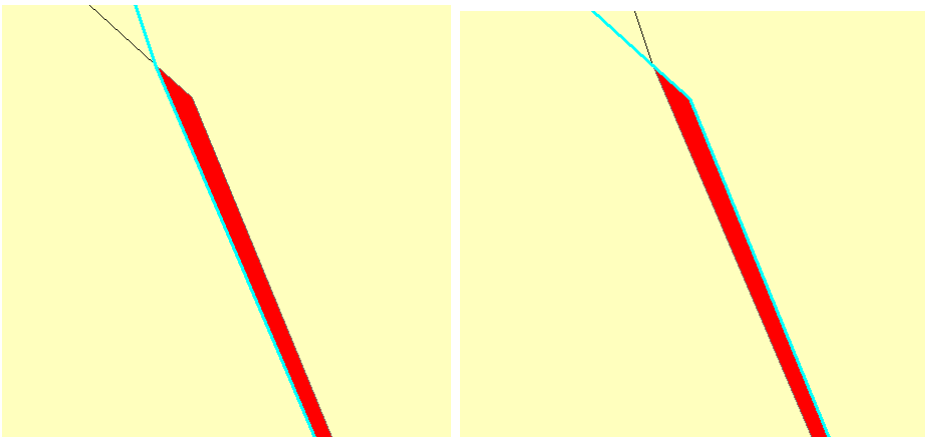
Jedna linie je obsažena v jiné linii:

```
CREATE TABLE dp_dupl_l AS
  SELECT SDO_GEOM.SDO_INTERSECTION(l1.geom,l2.geom,0.005) geom,
   l1.objectid l1_id, l2.objectid l2_id
  FROM dp_linie l1, dp_linie l2
  WHERE l1.objectid <> l2.objectid AND
    SDO_RELATE(l1.geom,l2.geom,'mask= covers querytype=WINDOW')= 'TRUE';
```

6.3.3 Hledání překrývajících se polygonů

Výsledkem dotazu je nová tabulka obsahující oblasti, kde dochází k překrytu polygonů.

```
CREATE TABLE overlap_p AS
SELECT SDO_GEOM.SDO_INTERSECTION(p.geom,l.geom,0.005) geom,
      p.objectid p_id, l.objectid l_id
FROM polygon p, polygon l
WHERE p.objectid <> l.objectid AND
SDO_RELATE(p.geom,l.geom,'mask= OVERLAPBDYINTERSECT
           querytype=WINDOW')= 'TRUE';
```



Obr. 6.10: Znázornění správnosti výsledku překrývajících se polygonů

6.3.4 Volné konce linií

Pro nalezení volných konců jsem se obrátila na diskuzní forum OTN Oracle dostupné na *https://forums.oracle.com/*. Výsledkem diskuze byl dotaz, který se snaží vyhledat linie obsahující volné konce, bohužel však nevyhledá izolované linie. Tento dotaz je uveden v příloze E. Výsledné nalezené linie dle dotazu:

```
---------------------------
   ID     SUM(ATS) SUM(ATE)
--------- -------- --------
  4615        2        0 -- asi neni volny konec
  1256        0        1 -- najde i arcgis
  7881        0        1 -- ma volny konec, ale arcgis neoznacil
  12140       1        0 -- ma volny konec, ale arcgis neoznacil
  1235        1        0 -- najde i arcgis
```

6.4 Srovnání

6.4.1 PostGIS

Všechny díry v polygonu nalezené pomocí ArcGIS jsou rovněž díry v geometrii `plochy_diss` vytvořené v PostGIS. Během převodu dat do file geodatabáze ArcGIS, s tolerancí 0.001, některé díry zanikly. Geometrie `plochy_diss` nebyla vytvořena

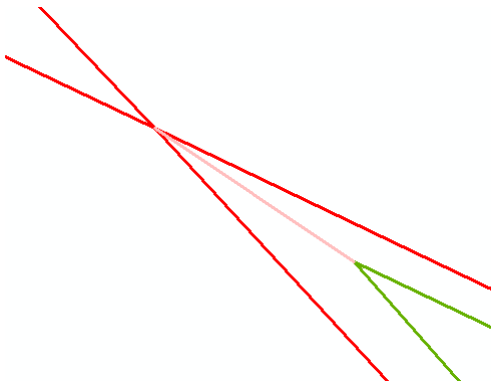
Vrstva(tabulka)	Doba trvání dotazu [s]	Počet prvků
plochy_diss	10.149	277
overlap_p	245.291	20
dupl_l2(equal)	1.188	29
dupl_l3(overlaps)	16.792	7
dupl_l4(covers)	5.054	30
freepoints	155.5	4

Tab. 6.2: Tabulka výsledků v PostGIS

s žádnou tolerancí, z tohoto důvodu dotaz v PostGIS nalezl více děr.

Všechny překrývající se polygony, které byly nalezeny v pomocí ArcGIS, jsou obsaženy ve výsledku dotazu v PostGIS. Rozdílné výsledky jsou způsobeny tolerancí v ArcGIS geodatabázi.

Při procházení jednotlivých překrývajících se linií nalezených pomocí ArcGIS bylo zjištěno, že kombinace duplicitních linií v PostGIS pokrývá tyto linie nalezené v ArcGIS až na dvě výjimky (dle počtu prvků by se měly lišit ve 3 liniích). Jeden z těchto případů je zobrazen na obr. 6.11, kde zeleně jsou linie v geodatabázi ArcGIS, červeně linie v PostGIS a růžově chyba nalezená v ArcGIS. Z obrázku je patrné, že chyba vznikla tolerancí 0.001 v geodatabázi ArcGIS. Geometrie dupl_l2 je obsažena i ve výsledné geometrii dupl_l4, ta obsahuje o jeden nalezený záznam více.



Obr. 6.11: Nenalezená linie v PostGIS

Dva volné konce nalezené v ArcGIS nebyly nalezeny pomocí dotazu v PostGIS. Nalezené volné konce v PostGIS nebylo možné vyexportovat výsledek, je zobrazen na obr. 6.7 pomocí PostGIS Viewer.

6.4.2 Spatial

Pro zjištění děr v polygonu byly použity dva dotazy. Prvním dotazem byla vytvořena geometrie `DP_DISS_PLOCHY_KU`. Z důvodu dlouhého trvání byl použit druhý dotaz pro vytvoření geometrie `DP_DISS_P`. Z počtu nalezených děr by se mohla zdát

Vrstva(tabulka)	Doba trvání dotazu [s]	Počet prvků
DP_DISS_P	39.89	44
DP_DISS_PLOCHY_KU	8506.352	65
DP_OVERLAP_P	3022.18	1
DP_DUPL_L(covers)	103.378	81
DP_DUPL_L2(equal)	49.617	139
DP_DUPL_L3(overlaps)	84.889	23

Tab. 6.3: Tabulka výsledků ve Spatial

vhodnější geometrie `DP_DISS_P`. Ta ovšem neobsahuje 16 děr polygonu, které byly nalezeny pomocí ArcGIS. Oproti tomu geometrie `DP_DISS_PLOCHY_KU` neobsahuje pouze 5 děr nalezených pomocí ArcGIS. Některé díry, které jsou nalezené pomocí Spatial, jsou na hranicích nevalidních polygonů, takže je možné, že vznikly při během opravy dat.

Nalezený překryv polygonů je správným výsledkem, jak je patrné z obr. 6.6. Nicméně tento překryv se v původních datech nevyskytuje, tudíž nemohl být nalezen pomocí ArcGIS. Překryv vznikl nejspíš během opravy nevalidních dat.

Oproti duplicitním liniím v PostGIS, obsahují geometrie DP_DUPL_L2 a DP_DUPL_L3 různé prvky. Při porovnávání výsledků bylo zjištěno, že 60 chyb nalezených pomocí ArcGIS není obsaženo ve výsledku hledání duplicitních linií ve Spatial.

Linie obsahující volné konce byly vyhledány pomocí dotazu uvedeného v příloze E. Dotaz je spíše pokusem, nalezl pouze 2 stejné volné konce jako ArcGIS a 2 volné konce, které ArcGIS neoznačil. Dotaz nevyhledává izolované linie.

Aby bylo možné všechny výsledky porovnat, byly v geodatabázi ArcGIS vytvořeny dvě topologie:

- topo_A s tolerancí 0.005 pro srovnání výsledků ze Spatial,
- topo_B s tolerancí 0.001 pro srovnání výsledků z PostGIS.

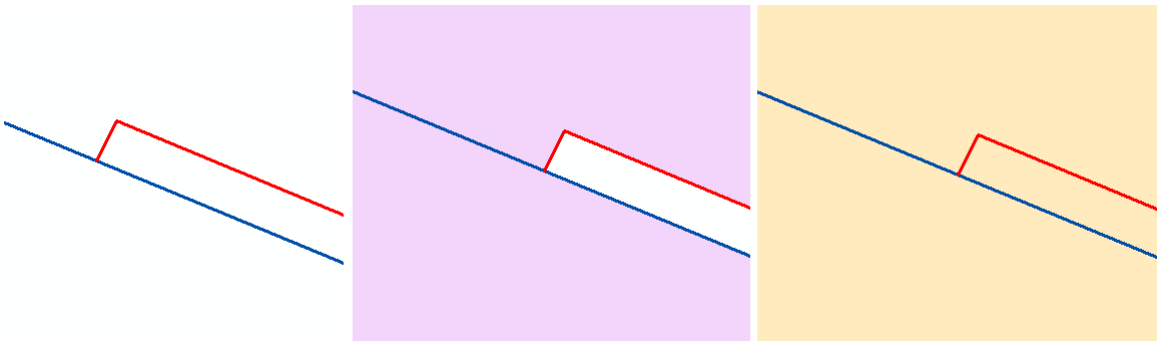
Chyba	ArcGIS(0.001/0.005)	PostGIS	Spatial
Must not have gaps	255/32	277	44/65
Must not overlap (polygony)	10/0	20	1
Must not have dangles	6/6	4	5
Must not overlap (linie)	40/312	37	243

Tab. 6.4: Porovnání výsledků

6.5 Porovnání validních a nevalidních dat

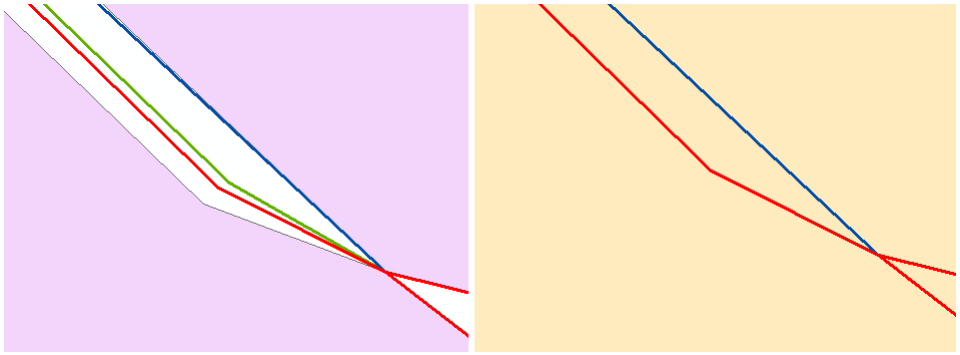
Během vytváření validních dat mohlo dojít ke změně geometrie dat. To může být příčinou rozdílnosti výsledků. V této podkapitole budou zobrazeny některé změny mezi původními a validními daty.

Obr. 6.12 znázorňuje změny validních a nevalidních dat. Červeně je zobrazena nevalidní linie, modře opravená (validní) linie, v tomto místě je nalezena duplicitní linie jak v ArcGISu, tak ve Spatial. V případě ploch fialová znázorňuje nevalidní polygon (obsahuje díru) a béžová validní.



Obr. 6.12: Porovnání validních a nevalidních dat ve Spatial

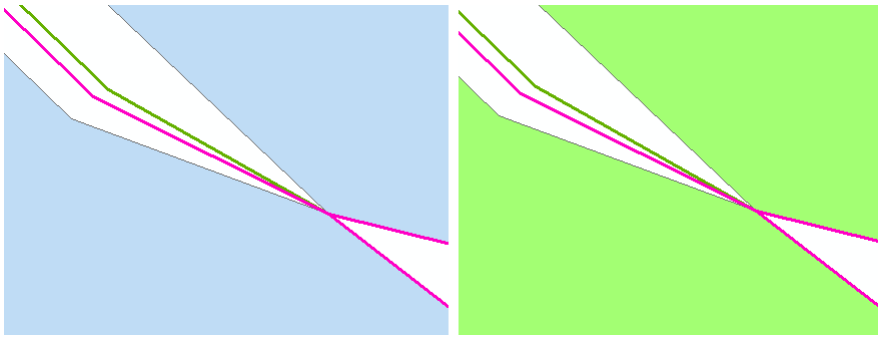
Na obr. 6.13 je červeně zobrazena nevalidní linie, zeleně původní linie a modře validní linie. Fialová plocha znázorňuje nevalidní polygon a béžová opravený validní. V tomto případě se sice v původních datech vyskytuje díra stejná, jaká je zobrazena na obr. 6.15, nicméně ve Spatial je opravou polygonu odstraněna a ani pomocí ArcGIS s tolerancí 0.005 není nalezena.



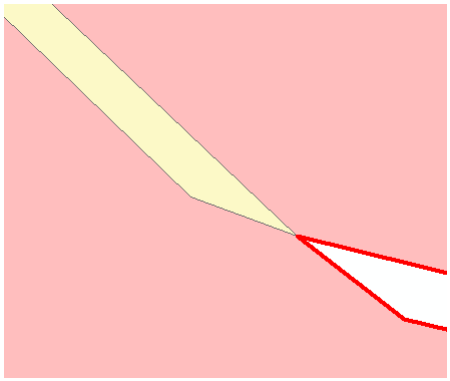
Obr. 6.13: Porovnání validních a nevalidních dat ve Spatial

Na obr. 6.14 je zeleně původní linie a růžově nejednoduchá linie (nonsimple). Modře je znárorněna nevalidní plocha a zeleně opravená. V tomto místě se polygon opravou nezměnil.

Na obr. 6.15 růžová barva reprezentuje polygon, který vznikl jako spojeni ploch, béžově původní plocha a červeně nalezená díra pomocí ArcGIS. V tomto případě v PostGISu vznikla nová díra.



Obr. 6.14: Porovnání validních a nevalidních dat v PostGIS



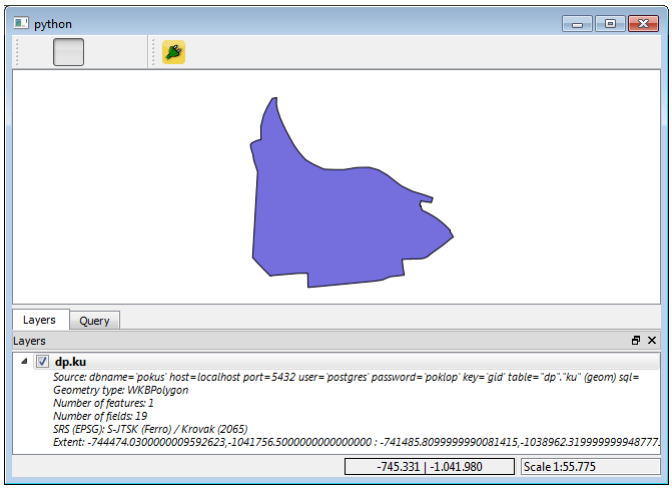
Obr. 6.15: Porovnání validních a nevalidních dat v PostGIS

6.6 Možnosti prezentace

6.6.1 PostGIS

PostGIS Viewer

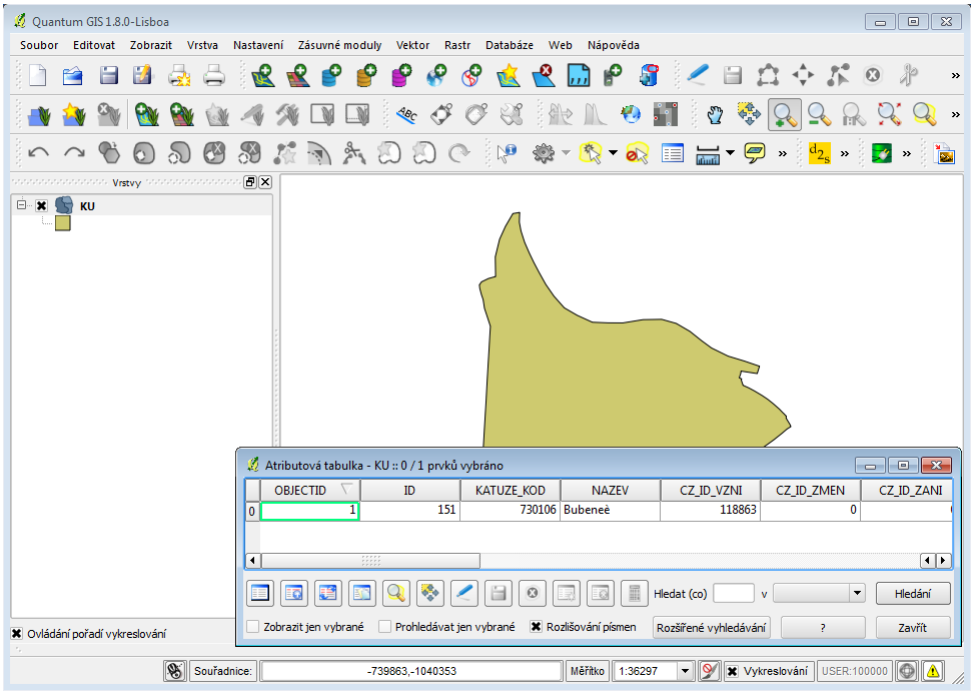
Pro rychlý náhled dat je možné přidat do PostGIS GUI PgAdmin plugin PostGIS Viewer.



Obr. 6.16: Ukázka zobrazení geometrie v PostGIS Viewer

Quantum GIS

Quantum GIS, zkráceně QGIS, je open source geografický informační systém pod licencí GNU General Public Licence (GPL). Umožňuje tvorbu, editaci a analýzy vektorových i rastrových dat. Lze jej rozšířit pomocí zásuvných modelů. [36]



Obr. 6.17: Ukázka prostředí QGIS

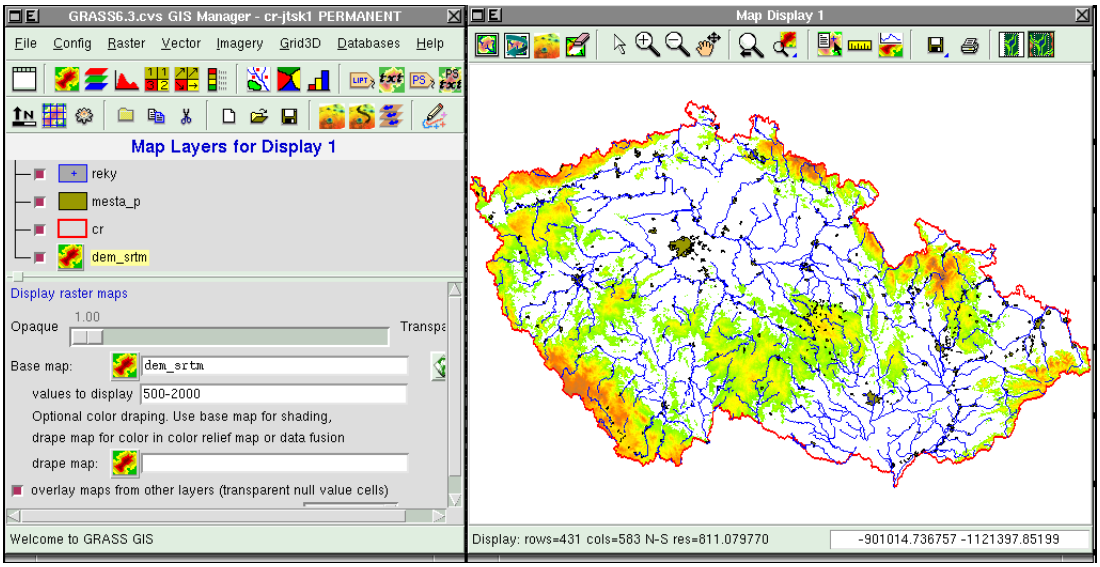
GRASS

GRASS (Geographical Resources Analysis Support System) je open source GIS využívaný pro analýzy nad prostorovým daty. Umožňuje práci s vektorovými i rastrovými daty. GRASS je oficiální projekt Open Source Geospatial Foundation pod licencí GNU GPL. [39] Ukázka grafického rozhraní GRASSu je na obr. 6.18.

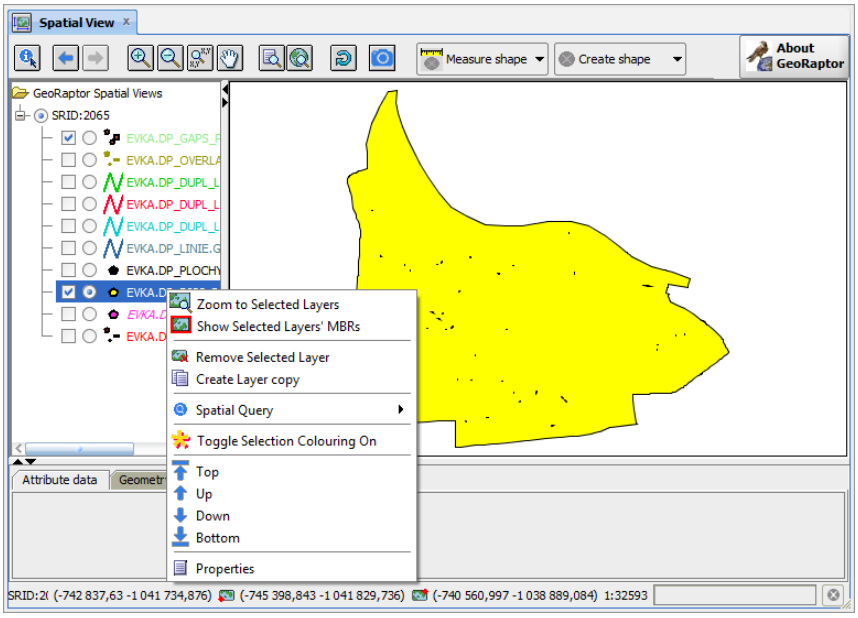
6.6.2 Spatial

GeoRaptor

GeoRaptor je rozšířením pro Oracle SQL Developer. Slouží k prohlížení a údržbě prostorových dat. Jedná se o open source projekt vyvíjený od roku 2006. [38] Ukázka prostředí GeoRaptor je na obr. 6.19.



Obr. 6.18: Ukázka GUI rozhraní GRASS, převzato z [39]



Obr. 6.19: Ukázka zobrazení geometrie v prostředí Georaptor

MapViewer

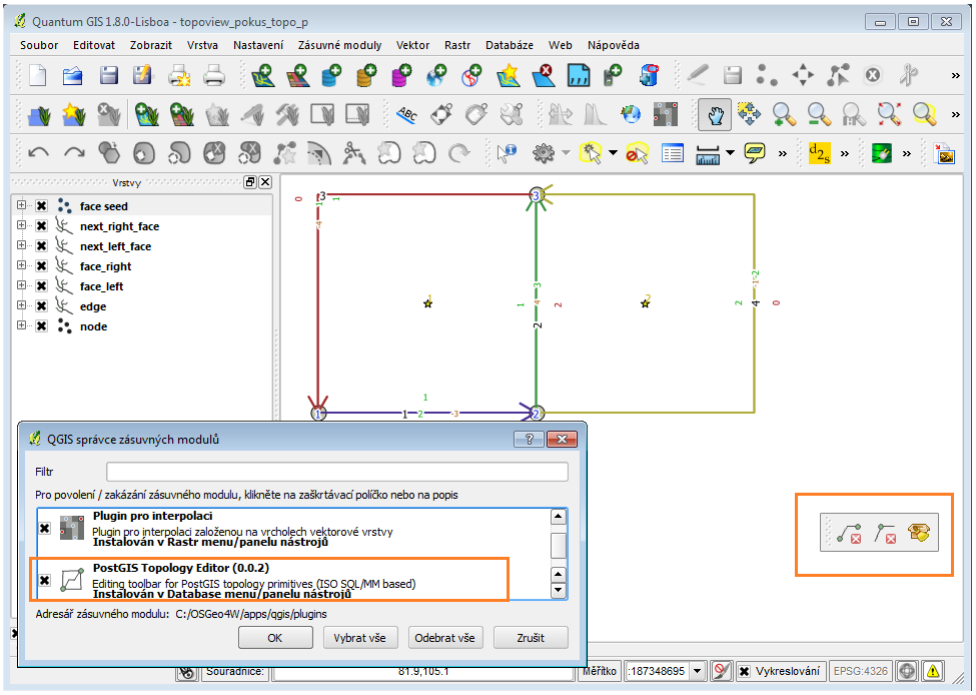
Oracle MapViewer (MapViewer) je programovatelný nástroj pro vykreslování map z prostorových dat, které jsou spravovány Oracle Spatial, nebo Oracle Locator. MapViewer poskytuje nástroje, které skrývají složitost dotazů nad prostorovými daty a kartografické vizualizace. Poskytuje i možnost přizpůsobit se potřebám pokročilejších uživatelů. MapViewer je dodáván jako součást Oracle Fusion Middleware. [37] Obsahuje následující komponenty:

- Základní nástroj pro vykreslování (Java knihovna) SDOVIS, který provádí kartografické vizualizace. Servlet je poskytován pro vykreslování funkcí ve webové aplikaci.
- Sada API, která umožňuje přístup k programovatelným funkcím MapViewer. API zahrnuje XML, Java, PL/SQL a JavaSkript API.
- Grafický nástroj Map builder, který umožňuje vytvářet mapové symboly, definovat pravidla pro vykreslování prostorových dat a vytvářet a editovat objekty v MapViewer.
- Oracle Mapa, která zahrnuje mezipaměť mapy a FOI (feature of interest)servery. Ty usnadňují vývoj interaktivních geoprostorových webových aplikací.

6.7 Možnosti topologické editace

6.7.1 PostGIS

Jednou z možností, jak spravovat topologická data, je pomocí pluginů v QGIS. V letošním roce byl vyvinut plugin *PostGIS Topology Editor* [35].



Obr. 6.20: Ukázka pluginu pro topologickou editaci v QGIS

Další možností, jak editovat topologická data, je využitím funkcí v PostGIS.

6.7.2 Spatial

Editovat topologická data je možné pomocí objektu TopoMap, více je uvedeno v kapitole 4.3.3.

Závěr

Cílem práce bylo shrnout možnosti práce s prostorovými daty a provádění topologických operací nad nimi v prostředí PostGIS a Oracle Spatial. Obsahem jsou základní informace o těchto geodatabázích a postupy práce v nich. V práci jsou rovněž zahrnuty popisy struktur a konkrétní dotazy, které by mohly budoucím uživatelům usnadnit začátky v těchto geodatabázích. Během zpracování jsem narazila na několik komplikací zejména při práci s reálnými daty.

Rozšíření PostGIS Topology je relativně nové a stále se vyvíjí, je proto možné se setkat s některými funkcemi, které nefungují na všech případech dat. V tomto ohledu jsem narazila na problém s funkcí `toTopoGeom()` při načítání polygonů do topologické struktury ve verzi 2.0. Ve verzi 2.0.2 se mi podařilo jednoduchou geometrii polygonů úspěšně převést. Bohužel v případě pokusu s objemnými reálnými daty dotaz po 30-ti hodinách skončil chybou *Spatial exception - geometry intersects*. Důležitým poznatkem k práci v PostGIS bylo zjištění, že není dobré používat velká písmena v názvech tabulek nebo topologie.

Topologicky strukturovaná data je možné vytvořit i v geodatabázi Oracle Spatial. Práce obsahuje příklad, jak takto strukturovaných dat docílit. Bohužel při snaze vyzkoušet tento postup na reálných datech nastal problém s vyrovnávací pamětí TopoMap objektu, pomocí kterého převod probíhá.

Překvapivé bylo porovnání doby trvání dotazů v PostGIS a Spatial, kdy dotazy v Oracle Spatial probíhaly mnohem déle. To by mohlo být pravděpodobně způsobeno konfigurací geodatabáze. Bohužel zjištění důvodu dlouhého času trvání dotazu vyžaduje rozsáhlejší znalosti.

Pro možnost porovnání byly výsledky dotazů vyexportovány do formátu $*$.shp a vizualizovány v ArcGIS 10.1. Výsledkem některých dotazů byly geometrie, které nebylo možné exportovat.

Komplikace, na které jsem narazila během zpracování, by mohly být odstraněny při získání větších zkušeností a dostatkem času na detailnější prostudování. Během zpracovávání této práce jsem se dozvěděla mnoho nových a užitečných informací, což bylo mým osobním cílem při hledání vhodného téma diplomové práce. Přes všechny komplikace jsem si díky této práci vyzkoušela mnoho zajímavých dotazů a porovnání použití geodatabází na reálných datech.

Použité zdroje

[1] CONOLLY T., BEGG C., HOLOWCZAK R. *Mistrovství – DATABÁZE: Profesionální průvodce tvorbou efektivních databází.*Brno: Computer Press, 2009, 584 s., ISBN 978-80-251-2328-7.

[2] MOMJIAN B. *PostgreSQL: Praktický průvodce.*Brno: Computer Press, 2003, 402 s., ISBN 80-7226-954-2.

[3] LONEY K. *ORACLE DATABASE: Kompletní průvodce.*Brno: Computer Press, 2010, 1368 s., ISBN 978-80-251-2489-5.

[4] KOLÁŘ J. *Geografické informační systémy 10*: Vydavatelství ČVUT: ČVUT v Praze Fakulta stavební, 1997, 149s., ISBN 80-01-02687-6.

[5] JANEČKA, K. *Modelování konzistentní báze geodat na úrovni datového modelu katastru nemovitostí*: disertační práce. Plzeň: ZČU Fakulta aplikovaných věd, 2009, 172s.

[6] JANEČKA, K. *Modelování geoprostorové báze dat na úrovni datového modelu KN*: diplomová práce. Plzeň: ZČU Fakulta aplikovaných věd, 2005, 83s.

[7] LINHARTOVÁ, E. *Topologie v GIS*: bakalářská práce. Praha: ČVUT v Praze Fakulta stavební, 2011, 51s.

[8] MURRAY CH. *Oracle Spatial Developer's Guide, 11g Release 2 (11.2)* [online] [cit.2013-04-22].
Dostupné z URL: <`http://docs.oracle.com/cd/E11882_01/appdev.112/e11830.pdf`>.

[9] MURRAY CH. *Oracle Spatial Topology and Network Data Models Developer's Guide, 11g Release 2 (11.2)* [online] [cit.2013-04-22].
Dostupné z URL: <`http://docs.oracle.com/cd/E18283_01/appdev.112/e11831.pdf`>.

[10] *Oracle's 30th Anniversary* [online] [cit.2013-04-22].
Dostupné z URL: <`http://www.oracle.com/us/corporate/profit/p27anniv-timeline-151918.pdf`>.

[11] *Oracle FAQ's* [online] [cit.2013-04-22].
Dostupné z URL: <`http://www.orafaq.com/wiki/Spatial`>.

[12] VUGTK *Terminologický slovník zeměměřictví a katastru nemovitostí* [online] [cit.2013-04-04].
Dostupné z URL: <`http://www.vugtk.cz/slovnik`>.

[13] VOJTEK, D. *Studijní materiály* [online] [cit.2013-04-03].
Dostupné z URL: <`http://gis.vsb.cz/vojtek/index.php?page=dict/index`>.

[14] *Oficiální stránky společnosti ESRI* [online] [cit.2013-04-10].
Dostupné z URL: <http://www.esri.com>.

[15] *ArcSDE* [online] [cit.2013-05-04].
Dostupné z URL: <http://gis.vsb.cz/gisengl/Conferences/GIS_Ova/GIS_Ova_2006/Proceedings/Firmy/arcdata/ArcSDE.PDF>.

[16] *Oficiální stránky společnosti ACRDATA PRAHA* [online] [cit.2013-04-10].
Dostupné z URL: <http://www.arcdata.cz>.

[17] *Nápověda ESRI* [online] [cit.2013-04-10].
Dostupné z URL: <http://webhelp.esri.com>.

[18] *Plakát topologických pravidel* [online] [cit.2013-05-03].
Dostupné z URL: <http://help.arcgis.com/en/arcgisdesktop/10.0/help/001t/pdf/topology_rules_poster.pdf>.

[19] *Oficiální stránky PostGIS* [online] [cit.2013-04-22].
Dostupné z URL: <http://postgis.net>.

[20] *PostGIS Manual* [online] [cit.2013-04-29].
Dostupné z URL: <http://postgis.net/docs/manual-2.0>.

[21] *Oficiální stránky PostgreSQL* [online] [cit.2013-04-22].
Dostupné z URL: <http://www.postgresql.org>.

[22] *PostGIS Tracker and Wiki* [online] [cit.2013-04-22].
Dostupné z URL: <http://trac.osgeo.org/postgis/wiki/UsersWikiPostgisTopology>.

[23] *GiST: A Generalized Search Tree for Secondary Storage* [online] [cit.2013-05-06].
Dostupné z URL: <http://gist.cs.berkeley.edu>.

[24] KORNACKER M. *High-Performance Extensible Indexing* [online] [cit.2013-05-06].
Dostupné z URL: <http://gist.cs.berkeley.edu/hiperf-gist.pdf>.

[25] SANTILLI S. *Topology with PostGIS* [online] [cit.2013-05-12].
Dostupné z URL: <http://strk.keybit.net/projects/postgis/Paris2011_TopologyWithPostGIS_2_0.pdf>.

[26] Lesnická a dřevařská fakulta, Mendelova univerzita v Brně: *Přednášky předmětu Digitální kartografie v ArcGIS.* [online] [cit.2013-04-10].
Dostupné z URL: <http://mapserver.mendelu.cz/arcgis>.

[27] *Geoportál Praha.* [online] [cit.2013-04-10].
Dostupné z URL: <http://www.geoportalpraha.cz>.

[28] *Koncept normy ISO 13249-3:2006.* [online] [cit.2013-04-16].
Dostupné z URL: <http://www.wiscorp.com/H2-2004-168r2-Topo-Geo-and-Topo-Net-1-The-Concepts.pdf>.

[29] *SQL/MM Spatial: The Standard to Manage Spatial Data in Relational Database Systems.* [online] [cit.2013-05-15].
Dostupné z URL: <http://www.fer.unizg.hr/_download/repository/SQLMM_Spatial-_The_Standard_to_Manage_Spatial_Data_in_Relational_Database_Systems.pdf>.

[30] *OpenGIS Implementace pro Simple feature část 1.* [online] [cit.2012-11-19].
Dostupné z URL: <http://www.opengeospatial.org/standards/sfs>.

[31] *OpenGIS Implementace pro Simple feature část 2.* [online] [cit.2012-11-19].
Dostupné z URL: <http://www.opengeospatial.org/standards/sfs>.

[32] *Přednášky předměru UPZD* [online] [cit.2013-04-16].
Dostupné z URL: <http://geo.fsv.cvut.cz/~gin/uzpd/uzpd.pdf>.

[33] *Útvar rozvoje HLAVNÍHO MĚSTA PRAHY.* [online] [cit.2013-04-17].
Dostupné z URL: <http://www.urm.cz/cs/digitalni_mapa_prahy>.

[34] *Testing Oracle 10g Topology using cadastral data.* [online] [cit.2013-05-06].
Dostupné z URL: <http://www.gdmc.nl/publications/reports/GISt26.pdf>.

[35] *Plugin: PostGIS Topology Editor* [online] [cit.2013-05-15].
Dostupné z URL: <http://plugins.qgis.org/plugins/pgtopoeditor/>.

[36] *Quantum GIS* [online] [cit.2013-05-15].
Dostupné z URL: <http://www.qgis.org/>.

[37] MURRAY CH. *Oracle Fusion Middleware User's Guide for Oracle MapViewer* [online] [cit.2013-05-10].
Dostupné z URL: <http://download.oracle.com/otndocs/products/mapviewer/mapviewer_11p6_ug.pdf>.

[38] *GeoRaptor* [online] [cit.2013-05-15].
Dostupné z URL: <http://sourceforge.net/apps/mediawiki/georaptor/index.php?title=Main_Page>.

[39] *GRASS wiki* [online] [cit.2013-05-15].
Dostupné z URL: <http://grass.fsv.cvut.cz/gwiki/Hlavn%C3%AD_strana>.

Seznam obrázků

2.1 Architektura ArcGIS, převzato z [16] 16
2.2 Způsob prezentace topologických pravidel na plakátu ESRI, převzato z [18] 18
3.1 Znázornění WKB fomátu, převzato z [30] 21
3.2 Simple LINESTRING, převzato z [20] 22
3.3 Simple MULTILINESTRING, převzato z [20] 22
3.4 Validní POLYGON, převzato z [20] 23
3.5 Validní MULTIPOLYGON, převzato z [20] 23
3.6 Rozhraní metody GiST, převzato z [24] 25
3.7 Matematická reprezentace matice, převzato z [20] 27
3.8 Znázornění dvou překrývajících se polygonů, převzato z [20] 27
3.9 Vstup a výstup geometrie při použití ST_Intersection, převzato z [32] 28
3.10 Vstup a výstup geometrie při použití ST_Union, převzato z [32] . . . 29
3.11 Vstup a výstup geometrie při použití ST_SymDifference, převzato z [32] 29
3.12 Vstup a výstup geometrie při použití ST_Difference převzato z [32] . 30
3.13 Koncepční model PostGIS Topology, převzato z [22] 33
3.14 Propojení tabulek v PostGIS 35
3.15 Topologické prvky, převzato z [28] 36
3.16 Ukázka tabulky hran popisující topologické prvky z obr. 3.15, převzato z [28] 36
3.17 Ukázka spuštění pluginu pro import 38
3.18 Ukázka importu dat pomocí PostGIS Shapefile Import/Export Manager 39
3.19 Ukázka možností nastavení importu dat 40
3.20 Ukázka vytvořených topologických tabulek 41
4.1 Podporované geometrické typy ve Spatial, převzato z [8] 43
4.2 Dotazovací model, převzato z [8] 44
4.3 Nejmenší ohraničující obdélník, převzato z [8] 46
4.4 R–tree index, převzato z [8] 47
4.5 Topologický vztah TOUCH definovaný 9IM, převzato z [8] 48
4.6 Topologické vztahy v 9IM, převzato z [8] 49
4.7 Výsledek funkce SDO_GEOM.SDO_INTERSECTION, převzato z [8] 50
4.8 Výsledek funkce SDO_GEOM.SDO_DIFFERENCE, převzato z [8] . 50
4.9 Výsledek funkce SDO_GEOM.SDO_UNION, převzato z [8] 51
4.10 Výsledek funkce SDO_GEOM.SDO_XOR, převzato z [8] 51
4.11 Topologické prvky, převzato z [9] 52
4.12 Znázornění propojení tabulek, převzato z [9] 54
4.13 Vztahy mezi primitivy v tabulce hran, převzato z [9] 56
4.14 Tabulka vztahů primitiv, převzato z [9] 57
4.15 Spuštění nástroje Georaptor pro načtení ∗.shp 60
4.16 Import dat z formátu ∗.shp do Spatial databáze 60
4.17 Možnost aktualizace metadat a vytvoření indexů v nastavení GeoRaptor 61
4.18 Topologické tabulky vytvořené během procedury 63
4.19 Tabulky prvků obsahující sloupec typu TopoGeometry 63

5.1 Ukázka DMP, převzato z [27] 65
5.2 Ukázka DOKM, převzato z [27] 67
5.3 Ukázka polohopisu DTM, převzato z [27] 68
5.4 Ukázka dat MTVU v prostředí ArcGIS. 69
6.1 Sumarizace nalezených chyb pro toleranci 0.005 70
6.2 Sumarizace nalezených chyb pro toleranci 0.001 71
6.3 Polygon vytvořený pro zjištění počtu děr 72
6.4 Linie ve vztahu overlaps 73
6.5 Linie ve vztahu Covers 73
6.6 Znázornění správnosti výsledku překrývajících se polygonů 74
6.7 Vyhledání volných konců 74
6.8 Výsledek dotazu s použitím funkce SDO_AGGR_UNION 75
6.9 Výsledek dotazu s použitím funkce SDO_AGGR_SET_UNION . . 76
6.10 Znázornění správnosti výsledku překrývajících se polygonů 78
6.11 Nenalezená linie v PostGIS 79
6.12 Porovnání validních a nevalidních dat ve Spatial 81
6.13 Porovnání validních a nevalidních dat ve Spatial 81
6.14 Porovnání validních a nevalidních dat v PostGIS 82
6.15 Porovnání validních a nevalidních dat v PostGIS 82
6.16 Ukázka zobrazení geometrie v PostGIS Viewer 82
6.17 Ukázka prostředí QGIS 83
6.18 Ukázka GUI rozhraní GRASS, převzato z [39] 84
6.19 Ukázka zobrazení geometrie v prostředí Georaptor 84
6.20 Ukázka pluginu pro topologickou editaci v QGIS 85

Seznam tabulek

3.1 Popis tabulky SPATIAL_REF_SYS 31
3.2 Popis pohledu GEOMETRY_COLUMNS 31
3.3 Struktura tabulky Topology ve schématu topology 32
3.4 Struktura tabulky Layer ve schématu topology 33
3.5 Popis datového typu TopoGeometry 34
3.6 Struktura tabulky uzlů 35
3.7 Struktura pohledu hran 36
3.8 Struktura tabulky stěn 37
3.9 Struktura tabulky relací 37
3.10 Validace a jednoduchost dat 39
4.1 Popis datového typu SDO_GEOMETRY 44
4.2 Operátory pro kontrolu topologických vztahů 49
4.3 Popis datového typu SDO_TOPO_GEOMETRY 53
4.4 Struktura tabulky uzlů 55
4.5 Struktura tabulky stěn 55
4.6 Struktura tabulky hran 56
4.7 Struktura tabulky vztahů 57
4.8 Kontrola validace dat 62
6.1 Tabulka použitých topologických pravidel dle ESRI 70
6.2 Tabulka výsledků v PostGIS 79
6.3 Tabulka výsledků ve Spatial 80
6.4 Porovnání výsledků 80

Seznam příloh

A postgis.sql **94**

B postgis_topology.sql **97**

C spatial.sql **98**

D get_geom_set.sql **104**

E dangles.sql **105**

F spatial_topology.sql **107**

A postgis.sql

```
CREATE SCHEMA DP;
CREATE SCHEMA DP_PUVODNI;
SET search_path TO dp, topology, public;

---------- VALIDACE DAT ----------
----------------------------------

SELECT ST_IsSimple(geom)
FROM body
WHERE NOT ST_IsSimple(geom); -- 0

CREATE TABLE linie_nonsimple AS
SELECT gid, ST_IsSimple(geom) AS simplicity, geom
FROM linie
WHERE NOT ST_IsSimple(geom); -- 9 zaznamu

SELECT ST_IsSimple(geom) AS simplicity, geom
FROM linie
WHERE NOT ST_IsSimple(geom); -- po oprave 0

SELECT ST_IsValidDetail(geom)
FROM linie
WHERE NOT ST_IsValid(geom); -- 0

SELECT count(ST_IsValid(geom))
FROM plochy
WHERE NOT ST_IsValid(geom); -- 78 zaznamu

SELECT gid, geom, ST_IsValidReason(geom) AS validity_info
FROM plochy
WHERE NOT ST_IsValid(geom); -- Ring Self-intersection

SELECT count(ST_IsValid(geom))
FROM ku
WHERE NOT ST_IsValid(geom); -- 0

---------- OPRAVA NEVALIDNICH DAT -------
-----------------------------------------

DELETE FROM linie
  WHERE NOT ST_IsSimple(geom); -- odstrani non-simple zaznamy
SELECT count(*) FROM linie;

UPDATE plochy
  SET geom=ST_Buffer(geom, 0.0); -- pomaha

---------- VYTVORENI PROST. INDEXU ------
-----------------------------------------

CREATE INDEX ku_idx
ON ku USING GIST (geom); -- 12ms
```

```
CREATE INDEX body_idx
ON body USING GIST (geom); -- 600ms

CREATE INDEX linie_idx
ON linie USING GIST (geom);  -- 1343ms

CREATE INDEX plochy_idx
ON plochy USING GIST (geom); -- 122ms

-------------------------------------------------------
-------------------------------------------------------

-------- dissolve podle ku ----------------------------
CREATE TABLE plochy_diss AS
SELECT st_union(geom) AS geom
FROM plochy
GROUP BY katuze_kod;                    -- 10149ms

SELECT st_nrings(geom) FROM plochy_diss; -- 278 = 277 der

SELECT st_astext(geom) FROM plochy_diss;
-- nema geometrii, ale jde zobrazit v~postgis viewer???

select st_geometrytype(geom),st_numgeometries(geom) from plochy_diss;

drop table gaps;
CREATE TABLE gaps AS
SELECT ST_CollectionExtract(ST_Difference(a.geom,b.geom),3) as geom
FROM ku as a, plochy_diss as b;

select st_geometrytype(geom),st_numgeometries(geom) from gaps;  --291 der

SELECT st_nrings(geom) FROM gaps;  -- 291 der

-------------------------------------------------------------------------------
-------------- duplicitni linie -------
-------------------------------------------------------------------------------
CREATE TABLE dupl_l AS
SELECT l1.gid AS l1_id, l2.gid AS l2_id,
   ST_Intersection(l1.geom,l2.geom) AS geom
FROM linie AS l1
JOIN linie AS l2 ON l1.geom && l2.geom
AND  l1.gid != l2.gid AND ST_Relate(l1.geom,l2.geom,'1*1******');  -- 16588ms

CREATE TABLE dupl_l2 AS
SELECT l1.gid AS l1_gid, l2.gid AS l2_gid,l1.geom
FROM linie AS l1, linie AS l2
WHERE l1.gid != l2.gid AND ST_EQUALS(l1.geom,l2.geom)='t';  -- 1188ms

SELECT ST_AsText(geom)
FROM dupl_l_p;

CREATE TABLE dupl_l_l AS
SELECT l1_id, l2_id, ST_CollectionExtract(geom,2) AS geom
FROM dupl_l;  -- 72ms
```

```
CREATE TABLE dupl_l3 AS
SELECT l1.gid AS l1_gid, l2.gid AS l2_gid,l1.geom
FROM linie AS l1, linie AS l2
WHERE l1.gid != l2.gid AND ST_Overlaps(l1.geom,l2.geom)='t'; -- 16792ms

CREATE TABLE dupl_l4 AS
SELECT l1.gid AS l1_gid, l2.gid AS l2_gid,l1.geom
FROM linie AS l1, linie AS l2
WHERE l1.gid != l2.gid AND ST_Covers(l1.geom,l2.geom)='t'; -- 5054ms

SELECT postgis_full_version();
drop table dupl_l_p;
```

```
---------- prekryvajici se polygony ----------
```

```
CREATE TABLE overlap_p AS -- vysledkem polygony
SELECT p.objectid AS p_id, l.objectid AS l_id,
       ST_CollectionExtract(ST_Intersection(p.geom, l.geom),3) AS geom
FROM plochy AS p, plochy AS l
WHERE p.objectid!= l.objectid AND ST_Overlaps(p.geom, l.geom) ='t';

SELECT ST_AsText(ST_CollectionExtract(geom,3))
FROM overlap_p;
```

```
CREATE TABLE overlap_p2 AS -- vysledkem je typ colection
SELECT ST_Intersection(p.geom, l.geom) AS geom,
       p.objectid AS p_id, l.objectid AS l_id
FROM plochy AS p JOIN plochy AS l ON
     ST_Overlaps(p.geom, l.geom) AND p.objectid!= l.objectid;   -- 248177ms

CREATE TABLE overlap_p2_l AS -- linie
SELECT p.objectid AS p_id, l.objectid AS l_id,
       ST_CollectionExtract(ST_Intersection(p.geom, l.geom),2) AS geom
FROM plochy AS p, plochy AS l
WHERE p.objectid!= l.objectid AND ST_Overlaps(p.geom, l.geom) ='t'; -- 248670ms
```

```
------ volne konce----------
```

```
CREATE VIEW unclosed_linie AS
  SELECT geom,gid
  FROM linie
  WHERE NOT st_equals(st_startpoint(geom), st_endpoint(geom)); -- 444ms

CREATE VIEW linie_points AS
(SELECT st_startpoint(geom) AS geom,gid
 FROM unclosed_linie UNION ALL SELECT st_endpoint(geom),gid AS geom
 FROM unclosed_linie);                                  --14ms

CREATE TABLE freepoints AS
 SELECT geom
 FROM linie_points
 GROUP BY geom
  HAVING geom NOT IN (SELECT p.geom FROM linie_points AS p JOIN linie AS l
  ON st_dwithin(p.geom,l.geom, 1) AND p.gid <> l.gid); -- 155042 ms
```

B postgis_topology.sql

```
---------- VYTVORENI TOPOLOGIE ----------
-----------------------------------------
SET search_path TO dp, topology, public;
SELECT CreateTopology('dp_topo',2065,0.005);  -- 1491 ms

---------- PRIDANI SLOUPCE TOPO DO EXISTUJICICH TABULEK ----------
------------------------------------------------------------------

SELECT AddTopoGeometryColumn('dp_topo', 'dp', 'ku', 'topo', 'POLYGON');
-- 72ms
SELECT AddTopoGeometryColumn('dp_topo', 'dp', 'body', 'topo', 'POINT');
-- 52ms
SELECT AddTopoGeometryColumn('dp_topo', 'dp', 'plochy', 'topo', 'POLYGON');
 -- 35ms
SELECT AddTopoGeometryColumn('dp_topo', 'dp', 'linie', 'topo', 'LINESTRING');
 -- 42ms

---------- NAPLNENI TOPOLOGICKYCH TABULEK ----------
----------------------------------------------------

UPDATE ku SET topo = topology.toTopoGeom(geom, 'dp_topo',1,0.005);  -- 93ms
UPDATE body SET topo = topology.toTopoGeom(geom, 'dp_topo',2,0.005); -- 163396ms
UPDATE plochy  SET topo = topology.toTopoGeom(geom, 'dp_topo',3,0.005);
 -- 106126.444s ERROR:  Spatial exception
UPDATE linie SET topo = topology.toTopoGeom(geom, 'dp_topo',4,0.005);

SELECT * from TopologySummary('dp_topo');
```

C spatial.sql

```
---------- AKTUALIZACE METADAT ----------
-----------------------------------------
INSERT INTO user_sdo_geom_metadata VALUES (
'dp_ku',
'geom',
SDO_DIM_ARRAY(
SDO_DIM_ELEMENT('X', -745000, -741000, 0.005), -- max a min sour.
SDO_DIM_ELEMENT('Y', -1042000, -1038000, 0.005) -- 0.005 tolerance
),
2065 -- SRID
);

INSERT INTO user_sdo_geom_metadata VALUES (
'dp_body',
'geom',
SDO_DIM_ARRAY(
SDO_DIM_ELEMENT('X', -745000, -741000, 0.005), -- max a min sour.
SDO_DIM_ELEMENT('Y', -1042000, -1038000, 0.005) -- 0.005 tolerance
),
2065 -- SRID
);

INSERT INTO user_sdo_geom_metadata VALUES (
'dp_linie',
'geom',
SDO_DIM_ARRAY(
SDO_DIM_ELEMENT('X', -745000, -741000, 0.005), -- max a min sour.
SDO_DIM_ELEMENT('Y', -1042000, -1038000, 0.005)-- 0.005 tolerance
),
2065 -- SRID
);

INSERT INTO user_sdo_geom_metadata VALUES (
'dp_plochy',
'geom',
SDO_DIM_ARRAY(
SDO_DIM_ELEMENT('X', -745000, -741000, 0.005), -- max a min sour.
SDO_DIM_ELEMENT('Y', -1042000, -1038000, 0.005) -- 0.005 tolerance
),
2065 -- SRID
);


---------- VALIDACE ----------
------------------------------
CREATE TABLE dp_val_results (
sdo_rowid ROWID,
RESULT VARCHAR2(2000)
);

SELECT SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT(geom , 0.005)
FROM dp_ku;
```

```
SELECT SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT(geom , 0.005)
FROM dp_body
WHERE SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT(geom , 0.005)<> 'TRUE';

SELECT count(*)
FROM dp_linie
WHERE SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT(geom , 0.005)<> 'TRUE';

SELECT count(*)
FROM dp_plochy
WHERE SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT(geom , 0.005)<> 'TRUE';

CALL
SDO_GEOM.VALIDATE_LAYER_WITH_CONTEXT('dp_ku','geom', 'DP_VAL_RESULTS');
SDO_GEOM.VALIDATE_LAYER_WITH_CONTEXT('dp_body','geom', 'DP_VAL_RESULTS');
SDO_GEOM.VALIDATE_LAYER_WITH_CONTEXT('dp_linie','geom', 'DP_VAL_RESULTS');
SDO_GEOM.VALIDATE_LAYER_WITH_CONTEXT('dp_plochy','geom', 'DP_VAL_RESULTS');

SELECT * FROM dp_val_results;
SELECT COUNT(*) FROM dp_val_results;

DROP TABLE dp_val_results;

-------------- OPRAVA NEVALIDNICH DAT --------------
----------------------------------------------------

 UPDATE dp_linie SET geom = SDO_UTIL.REMOVE_DUPLICATE_VERTICES(geom, 0.005);

 UPDATE dp_plochy SET geom = SDO_UTIL.RECTIFY_GEOMETRY(geom, 0.005);


-------------- VYTVORENI PROST. INDEXU --------------
-----------------------------------------------------
CREATE INDEX dp_ku_idx
 ON dp_ku(geom)
 INDEXTYPE IS MDSYS.SPATIAL_INDEX; -- 0.12s

CREATE INDEX dp_body_idx
 ON dp_body(geom)
 INDEXTYPE IS MDSYS.SPATIAL_INDEX;  -- 2.251s

CREATE INDEX dp_linie_idx
 ON dp_linie(geom)
 INDEXTYPE IS MDSYS.SPATIAL_INDEX; -- 1.355s

 CREATE INDEX dp_plochy_idx
 ON dp_plochy(geom)
 INDEXTYPE IS MDSYS.SPATIAL_INDEX; -- 0.66s

-------------------------------------------------------------------
-------- vytvoreni tabulky s~overlaps -----------------------------
-------------------------------------------------------------------

CREATE TABLE dp_overlap_p AS
  SELECT SDO_GEOM.SDO_INTERSECTION(p1.geom,p2.geom,0.005) geom,
                          p1.objectid p1_id, p2.objectid p2_id
```

```
  FROM dp_plochy p1, dp_plochy p2
  WHERE p1.objectid <> p2.objectid AND
        SDO_RELATE(p1.geom,p2.geom,
        'mask=_OVERLAPBDYINTERSECT_querytype=WINDOW')= 'TRUE'; -- 3022.18s

INSERT INTO user_sdo_geom_metadata VALUES (
'dp_overlap_p',
'geom',
SDO_DIM_ARRAY(
SDO_DIM_ELEMENT('X', -745000, -741000, 0.005), -- max a min sour.
SDO_DIM_ELEMENT('Y', -1042000, -1038000, 0.005)-- 0.005 tolerance
),
2065 -- SRID
);

CREATE INDEX dp_overlap_p_idx
ON dp_overlap_p(geom)
INDEXTYPE IS MDSYS.SPATIAL_INDEX;

SELECT p1_id, p2_id
FROM dp_overlap_p;
-------------------------------------------------------------
---------------- dissolve dle ku ----------------------------
-------------------------------------------------------------
CREATE TABLE dp_diss_plochy_ku AS
  SELECT KATUZE_KOD, SDO_AGGR_UNION(
    MDSYS.SDOAGGRTYPE(a.geom, 0.005))GEOM
  FROM dp_plochy a GROUP BY a.KATUZE_KOD; --8506,352s

CREATE TABLE dp_diss_p AS
 SELECT KATUZE_KOD, SDO_AGGR_SET_UNION (get_geom_set
                    ('dp_plochy', 'geom','CTVUK_KOD_<>''1'''), 0.005 ) geom
 FROM dp_plochy a GROUP BY a.KATUZE_KOD;   -- 37.891 s

CREATE TABLE dp_diss_p_bez_tolerance AS
 SELECT KATUZE_KOD, SDO_AGGR_SET_UNION (get_geom_set
                    ('dp_plochy', 'geom','CTVUK_KOD_<>''1'''), 0.0001 ) geom
 FROM dp_plochy a GROUP BY a.KATUZE_KOD; -- 21.17s

CREATE TABLE dp_diss_p01 AS
 SELECT KATUZE_KOD, SDO_AGGR_SET_UNION (get_geom_set
                    ('dp_plochy', 'geom','CTVUK_KOD_<>''1'''), 0.001 ) geom
 FROM dp_plochy a GROUP BY a.KATUZE_KOD; --39.156s

INSERT INTO user_sdo_geom_metadata VALUES (
'dp_diss_p',
'geom',
SDO_DIM_ARRAY(
SDO_DIM_ELEMENT('X', -745000, -741000, 0.005), -- max a min sour.
SDO_DIM_ELEMENT('Y', -1042000, -1038000, 0.005)-- 0.005 tolerance
),
2065 -- SRID
);

CREATE INDEX dp_diss_p_idx
ON dp_diss_p(geom)
```

```
INDEXTYPE IS MDSYS.SPATIAL_INDEX;  -- 3.526 s

INSERT INTO user_sdo_geom_metadata VALUES (
'dp_diss_p_bez_tolerance',
'geom',
SDO_DIM_ARRAY(
SDO_DIM_ELEMENT('X', -745000, -741000, 0.0001), -- max a min sour.
SDO_DIM_ELEMENT('Y', -1042000, -1038000, 0.0001)-- 0.005 tolerance
),
2065 -- SRID
);

CREATE INDEX dp_diss_p_bez_t_idx
ON dp_diss_p_bez_tolerance(geom)
INDEXTYPE IS MDSYS.SPATIAL_INDEX; -- 0.822 s

SELECT * FROM diss_plochy_ku a  -- v~pripade vice ku vybere ku obsahujici diry
WHERE SDO_GEOM.SDO_XOR(a.geom, SDO_UTIL.EXTRACT(a.geom,1,1),0.005) IS NOT NULL;

-------------------------------------------------------------
------------------------- DIRY ------------------------------
-------------------------------------------------------------

CREATE dp_gaps_p1 AS
SELECT SDO_GEOM.SDO_DIFFERENCE(a.geom, b.geom,0.005) AS geom
FROM dp_ku a, dp_diss_p b; -- 0.275 s

INSERT INTO user_sdo_geom_metadata VALUES (
'dp_gaps_p1',
'geom',
SDO_DIM_ARRAY(
SDO_DIM_ELEMENT('X', -745000, -741000, 0.0001), -- max a min sour.
SDO_DIM_ELEMENT('Y', -1042000, -1038000, 0.0001)-- 0.005 tolerance
),
2065 -- SRID
);

CREATE INDEX dp_gaps_p1_idx
ON dp_gaps_p1(geom)
INDEXTYPE IS MDSYS.SPATIAL_INDEX;  -- 0.286 s

SELECT SDO_UTIL.GETNUMELEM(geom)
FROM dp_gaps_p1;      -- 44
-------------------------------------------------------------------------
CREATE TABLE dp_gaps_p2 AS
SELECT SDO_GEOM.SDO_DIFFERENCE(a.geom, b.geom,0.005) AS geom
FROM dp_ku a, dp_diss_plochy_ku b; --  0.191s

INSERT INTO user_sdo_geom_metadata VALUES (
'dp_gaps_p2',
'geom',
SDO_DIM_ARRAY(
SDO_DIM_ELEMENT('X', -745000, -741000, 0.0001), -- max a min sour.
SDO_DIM_ELEMENT('Y', -1042000, -1038000, 0.0001)-- 0.005 tolerance
),
2065 -- SRID
```

```
);

CREATE INDEX dp_gaps_p2_idx
ON dp_gaps_p2(geom)
INDEXTYPE IS MDSYS.SPATIAL_INDEX; -- 0.609s
-------------------------------------------------------------------------
CREATE TABLE dp_gaps_p3 AS
SELECT SDO_GEOM.SDO_DIFFERENCE(a.geom, b.geom,0.005) AS geom
FROM dp_ku a, dp_diss_p_bez_tolerance b; --  0.214 s

INSERT INTO user_sdo_geom_metadata VALUES (
'dp_gaps_p3',
'geom',
SDO_DIM_ARRAY(
SDO_DIM_ELEMENT('X', -745000, -741000, 0.0001), -- max a min sour.
SDO_DIM_ELEMENT('Y', -1042000, -1038000, 0.0001)-- 0.005 tolerance
),
2065 -- SRID
);

CREATE INDEX dp_gaps_p3_idx
ON dp_gaps_p3(geom)
INDEXTYPE IS MDSYS.SPATIAL_INDEX;

CREATE TABLE dp_gaps_p01 AS
SELECT SDO_GEOM.SDO_DIFFERENCE(a.geom, b.geom,0.001) AS geom
FROM dp_ku a, dp_diss_p01 b; --  2.095 s

-------------------------------------------------------
---------------------- DUPLICITNI LINIE ---------------
-------------------------------------------------------
CREATE TABLE dp_dupl_l AS
                -- LINIE LEZI CELA NA CASTI DRUHE LINIE (NEJSOU TOTOZNE)
  SELECT SDO_GEOM.SDO_INTERSECTION(l1.geom,l2.geom,0.005) geom,
                l1.objectid l1_id, l2.objectid l2_id
  FROM dp_linie l1, dp_linie l2
  WHERE l1.objectid <> l2.objectid AND
        SDO_RELATE(l1.geom,l2.geom,'mask=covers querytype=WINDOW')= 'TRUE';
        -- 103.378s

CREATE TABLE dp_dupl_l2 AS    --- DUPLICITNI LINIE (TOTOZNE)
  SELECT l1.geom, l1.objectid
  FROM dp_linie l1,dp_linie l2
  WHERE l1.objectid <> l2.objectid AND
        SDO_RELATE(l1.geom,l2.geom,'mask=equal querytype=WINDOW')= 'TRUE';
        -- 49.617s

CREATE TABLE dp_dupl_l3 AS
  SELECT SDO_GEOM.SDO_INTERSECTION(l1.geom,l2.geom,0.005) geom,
                l1.objectid l1_id, l2.objectid l2_id
  FROM dp_linie l1, dp_linie l2
  WHERE l1.objectid <> l2.objectid AND
        SDO_OVERLAPS(l1.geom,l2.geom)= 'TRUE'; -- 84.889s

INSERT INTO user_sdo_geom_metadata VALUES (
'dp_dupl_l',
```

```
'geom',
SDO_DIM_ARRAY(
SDO_DIM_ELEMENT('X', -745000, -741000, 0.005), -- max a min sour.
SDO_DIM_ELEMENT('Y', -1042000, -1038000, 0.005)-- 0.005 tolerance
),
2065 -- SRID
);

CREATE INDEX dp_dupl_l_idx
ON dp_dupl_l(geom)
INDEXTYPE IS MDSYS.SPATIAL_INDEX;
-- nejdrive metadata, jinak nelze vytvorit index

INSERT INTO user_sdo_geom_metadata VALUES (
'dp_dupl_l2',
'geom',
SDO_DIM_ARRAY(
SDO_DIM_ELEMENT('X', -745000, -741000, 0.005), -- max a min sour.
SDO_DIM_ELEMENT('Y', -1042000, -1038000, 0.005)-- 0.005 tolerance
),
2065 -- SRID
);

CREATE INDEX dp_dupl_l2_idx
ON dp_dupl_l2(geom)
INDEXTYPE IS MDSYS.SPATIAL_INDEX;  -- 0.724 s

INSERT INTO user_sdo_geom_metadata VALUES (
'dp_dupl_l3',
'geom',
SDO_DIM_ARRAY(
SDO_DIM_ELEMENT('X', -745000, -741000, 0.005), -- max a min sour.
SDO_DIM_ELEMENT('Y', -1042000, -1038000, 0.005)-- 0.005 tolerance
),
2065 -- SRID
);

CREATE INDEX dp_dupl_l3_idx
ON dp_dupl_l3(geom)
INDEXTYPE IS MDSYS.SPATIAL_INDEX;  -- 0.304s
```

D get_geom_set.sql

```
CREATE OR REPLACE FUNCTION get_geom_set (table_name  VARCHAR2,
                                         column_name VARCHAR2,
                                         predicate   VARCHAR2 := NULL)
  RETURN SDO_GEOMETRY_ARRAY DETERMINISTIC AS

  type             cursor_type is REF CURSOR;
  query_crs        cursor_type ;
  g                SDO_GEOMETRY;
  GeometryArr      SDO_GEOMETRY_ARRAY;
  where_clause     VARCHAR2(2000);
BEGIN
  IF predicate IS NULL
  THEN
    where_clause := NULL;
  ELSE
    where_clause := ' WHERE ';
  END IF;

  GeometryArr := SDO_GEOMETRY_ARRAY();
  OPEN query_crs FOR ' SELECT ' || column_name ||
                     ' FROM ' || table_name ||
                       where_clause || predicate;
   LOOP
    FETCH query_crs into g;
     EXIT when query_crs%NOTFOUND ;
     GeometryArr.extend;
     GeometryArr(GeometryArr.count) := g;
   END LOOP;
   RETURN GeometryArr;
END;
```

E dangles.sql

```
WITH
l_vertices                              -- SUBQUERY l_vertices
  as
  (
  select v.row_id, v.id, v.vix, v.x, v.y, max(v.vix )
     OVER (PARTITION BY v.row_id ) last_v
     -- max analitical function to get last vertex index
  from
   (
     select rowid row_id, l.id, t.id vix, t.x , t.y
     from dp_linie l, TABLE(SDO_UTIL.GETVERTICES(l.geom)) t
     -- getvertices table function
   )
v~),
al_s_e_vertices       -- SUBQUERY al_s_e_vertices based on l_vertices
  as
  (
  select lvs.row_id, lvs.id, lvs.vix s_vix, lvs.x start_x, lvs.y start_y,
                        lve.vix e_vix, lve.x end_x, lve.y end_y
  from
  (
        select lv.row_id, lv.id, lv.vix, lv.x, lv.y
    from l_vertices lv
    where lv.vix =1             -- vertexindex = 1  is the startpoint
  ) lvs,
  (
   select lv.row_id, lv.id, lv.vix, lv.x, lv.y
   from l_vertices lv
   where lv.vix =lv.last_v
        -- vertexindex = last_v (last vertex index) is the endpoint
  ) lve
  where
  lvs.row_id = lve.row_id
  ),
bl_s_e_vertices               -- SUBQUERY bl_s_e_vertices based on l_vertices
  as
  (
  select lv.row_id, lv.id, lv.vix, lv.x, lv.y
    from l_vertices lv
    where lv.vix =1             -- vertexindex = 1  is the startpoint
  UNION ALL
   select lv.row_id, lv.id, lv.vix, lv.x, lv.y
   from l_vertices lv
   where lv.vix =lv.last_v
   -- vertexindex = last_v (last vertex index) is the endpoint
 )
  select a.ID, sum(atS), sum(atE)
-- Main query using SDO_JOIN (self join on LINE, LINE)
-- in combination with the derived (from LINE)
--subqueries al_s_e_vertices a, bl_s_e_vertices b

  from
  (
```

```
select l.*, CASE WHEN (start_x = x AND start_y = y) THEN 1 ELSE 0 END atS,
            CASE WHEN (end_x = x AND end_y = y) THEN 1 ELSE 0 END atE
 FROM
(
select  a.id, a.row_id, a.s_vix, a.start_x, a.start_y,
                a.e_vix, a.end_x, a.end_y,
        b.id b_id, b.row_id b_row_id, b.vix, b.x, b.y
from
table(sdo_join('dp_linie', 'geom', 'dp_linie', 'geom', 'mask=TOUCH')) c,
al_s_e_vertices a, bl_s_e_vertices b
WHERE c.rowid1 = a.row_id AND c.rowid2 = b.row_id
AND a.row_id != b.row_id
) l
) a group by ID
HAVING sum(atS)=0 or sum(atE)=0;
```

F spatial_topology.sql

```
------- VYTVORENI TOPOLOGIE DP_TOPO --------
--------------------------------------------

EXECUTE SDO_TOPO.CREATE_TOPOLOGY('DP_TOPO',0.005,2065);  -- 0.609s
 -- name_topo, tolerance, srid


------- VYTVORENI UNIVERSAL FACE 0 ---------
--------------------------------------------
INSERT INTO dp_topo_face values (
-1,
NULL,
SDO_LIST_TYPE(),
SDO_LIST_TYPE(),
NULL);                                  -- 0.024s

COMMIT;

--------- ZALOZENI PRVKOVYCH TABULEK -------
--------------------------------------------
CREATE TABLE DP_TOPO_KU (
OBJECTID VARCHAR2(30) PRIMARY KEY,
feature SDO_TOPO_GEOMETRY);                 -- 0.149s

CREATE TABLE DP_TOPO_BODY (
OBJECTID VARCHAR2(30) PRIMARY KEY,
feature SDO_TOPO_GEOMETRY);                 -- 0.03s

CREATE TABLE DP_TOPO_LINIE (
OBJECTID VARCHAR2(30) PRIMARY KEY,
feature SDO_TOPO_GEOMETRY);                 -- 0.027s

CREATE TABLE DP_TOPO_PLOCHY (
OBJECTID VARCHAR2(30) PRIMARY KEY,
feature SDO_TOPO_GEOMETRY);                 -- 0.028s

------- PRIDANI VRSTVY DO TOPOLOGIE --------
--------------------------------------------
EXECUTE SDO_TOPO.ADD_TOPO_GEOMETRY_LAYER
('DP_TOPO','DP_TOPO_KU','FEATURE','POLYGON'); -- 0.895
s~EXECUTE SDO_TOPO.ADD_TOPO_GEOMETRY_LAYER
('DP_TOPO','DP_TOPO_BODY','FEATURE','POINT'); -- 0.272
s~EXECUTE SDO_TOPO.ADD_TOPO_GEOMETRY_LAYER
('DP_TOPO','DP_TOPO_LINIE','FEATURE','LINE'); -- 0.364
s~EXECUTE SDO_TOPO.ADD_TOPO_GEOMETRY_LAYER
('DP_TOPO','DP_TOPO_PLOCHY','FEATURE','POLYGON'); --0.259 s

------- VYTVORENI TOPOMAP OBJEKTU ----------
--------------------------------------------
EXECUTE SDO_TOPO_MAP.CREATE_TOPO_MAP('DP_TOPO', 'DP_TOPOMAP');  -- 12.899
s~EXECUTE SDO_TOPO_MAP.LOAD_TOPO_MAP('DP_TOPOMAP', 'TRUE'); -- 1.114 s

COMMIT;
```

```
------ NACTENI DAT Z~PROST. TABULEK ------
-----------------------------------------

BEGIN
  FOR ku_rec IN (SELECT objectid, geom FROM dp_ku) LOOP
   INSERT INTO dp_topo_ku VALUES(ku_rec.objectid,
     SDO_TOPO_MAP.CREATE_FEATURE('DP_TOPO', 'DP_TOPO_KU', 'FEATURE',
         ku_rec.geom));
  END LOOP;
END;                                                    -- 0.937 s


BEGIN
  FOR body_rec IN (SELECT objectid, geom FROM dp_body) LOOP
   INSERT INTO dp_topo_body VALUES(body_rec.objectid,
     SDO_TOPO_MAP.CREATE_FEATURE('DP_TOPO', 'DP_TOPO_BODY', 'FEATURE',
         body_rec.geom));
  END LOOP;
END;                                                    -- 3644.61 s

BEGIN
  FOR linie_rec IN (SELECT objectid, geom FROM dp_linie) LOOP
   INSERT INTO dp_topo_linie VALUES(linie_rec.objectid,
     SDO_TOPO_MAP.CREATE_FEATURE('DP_TOPO', 'DP_TOPO_LINIE', 'FEATURE',
         linie_rec.geom));
  END LOOP;
END;          -- nepochopitelne hazi chybu, na testovacich datech bylo vse ok


BEGIN
  FOR plochy_rec IN (SELECT objectid, geom FROM dp_plochy) LOOP
   INSERT INTO dp_topo_plochy VALUES(plochy_rec.objectid,
     SDO_TOPO_MAP.CREATE_FEATURE('DP_TOPO', 'DP_TOPO_PLOCHY', 'FEATURE',
         plochy_rec.geom));
  END LOOP;
END;          -- nepochopitelne hazi chybu, na testovacich datech bylo vse ok

---------- INICALIZACE METADAT ----------
-----------------------------------------

EXECUTE SDO_TOPO.INITIALIZE_METADATA('DP_TOPO');
```