ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta stavební

Studijní program: Geodézie a kartografie

Studijní obor: Geoinformatika

Dokumentace projektu

Úvod do zpracování prostorových dat

Skupina B:

Jan Klíma

Matěj Krejčí

Jaroslav Urik

Obsah

1. Zadání

- Navrhněte a vytvořte tematické vrstvy (např. vodní toky, vodní plochy, lesy, silnice, železnice apod.) na základě dat OpenStreetMap. Pro tento účel byla na serveru 'geo102' založena databáze pgis_uzpd.
- Aplikujte testy datové integrity a odstraňte případné nekonzistence v datech
- Vytvořte tutoriál pro výuku PostGIS - tj. sadu atributových a prostorových dotazů nad databází pgis_uzpd.

2. Úvod

Tato dokumentace projektu vznikla v rámci předmětu Úvod do zpracování prostorových dat, který je vyučován ve třetím ročníku bakalářského studia na ČVUT v Praze. Tento předmět se zaměřuje na rozšíření získaných zkušeností s databázemi z předmětu Databázové systémy a to především o zpracování geoprostorových dat v prostředí PostGIS.

2.1. PostGIS

PostGIS je open source rozšíření objektově relačního databázového systémy PostgeSQL, kterému přidává možnost pracovat s geografickými objekty. První verze byla vydána v roce 2001 a od roku 2006 PostGIS implementuje specifikaci Simple Features konsorcia OGC (Open Geospatial Consortium). Umožňuje práci s geometrickými typy points, linestrings, polygons, multipoints, multilinestrings, multipolygons and geometrycollections. Dále umožňuje provádět prostorové analýzy (délka, výměra, vzdálenost, průnik, obalové zóny…).[1]

2.2. PgRouting

Open source rozšíření geoprostorové databáze PostGIS/PostgreSQL, které umožňuje provádění síťových analýz. Především vyhledávání nejkratších cest, řešení obchodního cestujícího a výpočet dojezdové vzdálenosti (izočáry). [2]

2.3. OpenStreetMap

Projekt OpenStreetMap je založen v roce 2004 na koncepci Open source a hlavním cílem projektu je kolektivní tvorba volně dostupných geografických dat a vizualizace těchto dat do podoby topografických map.

„Veškerá data vstupující do projektu musí být dostupná jako volné dílo, pod licencí kompatibilní s Creative Commons Attribution-Share Alike nebo bez copyrightu. Přispěvatelé se musí zaregistrovat a souhlasit, že poskytovaná data jsou licencována pod Creative Commons 2.0 SA. Veškeré změny provedené přispěvateli jsou zaznamenávány, to umožňuje v případě nutnosti odstranění sporných dat. Nicméně podobné kroky mají

[1] http://cs.wikipedia.org/wiki/PostGIS
[2] http://pgrouting.org/

neblahý vliv na projekt, navíc vyžadují odstranění všech souvisejících změn provedených v souvislosti s kompromitujícími daty.“ [3]

Data jsou získávány od dobrovolníků, která vytvářejí nová data nejčastěji pomocí GPS přijímačů. Data jsou dále zpracovávána a poté nahrána do databáze OpenStreetMap. V poslední době také dochází k zpřístupnění některých komerčních dat (Landsat 7, TIGER, katastrální mapa ČR ČUZK…). Jako podkladové vrstvy jsou často použity letecké a satelitní snímky od komerční společnosti Yahoo.

Obr. 1. Rozvoj OSM – roky 2007, 2008, 2009

Data jsou ukládány jako: uzly (body o známých souřadnicích v souřadnicovém systému), cesty (posloupnost uzlů, polylinie, polygony), relace, atributy.

3. Vytváření a úprava vrstev

Cílem našeho projektu je vytvoření tematických vrstev z dat OpenStreetMap (případně z dat z databáze GIS1) zaměřená na zdravotnická zařízení a nad těmito daty provést prostorové analýzy. Součástí tohoto projektu je také soubor atributových/prostorových dotazů a dotazů spojených s nástavbou PgRouting.

Data OSM je možné importovat do databáze pomocí aplikace osm2pgsql, to ale nebylo nutné, neboť data jsou již uložena v databázi pgis_student ve schématu „osm“. Pro účely projektu bylo pro naši skupinu vytvořena schéma „b13“ v databázi pgis_uzpd, ve kterém jsou uloženy všechny nově vytvořené vrstvy.

Pro zjednodušený přístup k datům byla nastavena vyhledávací cesta:

```
SET SEARCH_PATH TO b13,osm,gis1,public;
```

3.1. Vrstva silnic

Vrstva silnic byla vybrána ze schématu osm se sloupečky `osm_id` (integer – identifikátor), `name` (textový řetězec, velmi často hodnota NULL), `highway` (textový

[3] http://cs.wikipedia.org/wiki/OpenStreetMap

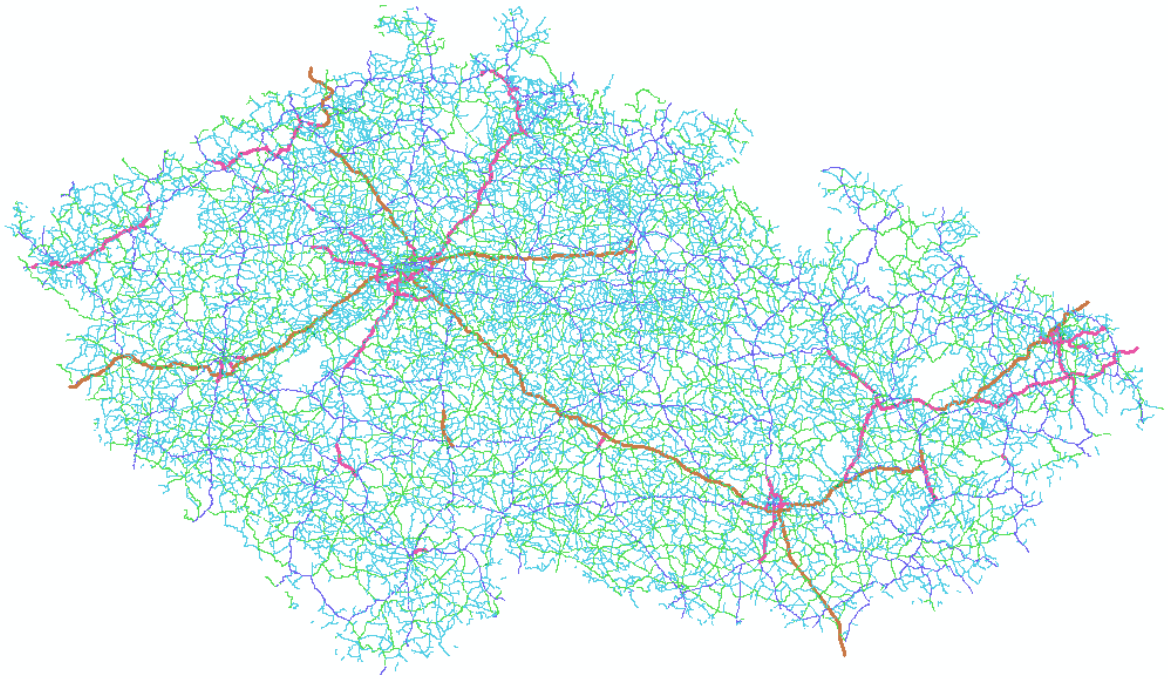
řetězec) a `geom` (geometrie). Byly vybrány ty data, které mají ve sloupečku highway (pozemní komunikace) uveden typ komunikace: `motorway` (dálnice), `trunk` (rychlostní silnice), *primary* (silnice 1. třídy), *secondary* (silnice 2. třídy), *tertiary* (silnice 3. třídy). Ve sloupci `highway` jsou dále prvky s atributem *PříslušnýTypKomunikace_link* označující nájezdy a sjezdy. [4]

```
CREATE TABLE silnice AS
SELECT osm_id, name AS nazev, highway AS druh, geom
FROM czech_line
WHERE highway IN ('motorway', 'trunk','primary', 'secondary',
'tertiary','motorway_link', 'trunk_link', 'primary_link',
'secondary_link', 'tertiary_link');
```

- Vytvoření primárního klíče do tabulky k sloupci `gid` a vytvoření prostorového indexu

```
ALTER TABLE silnice ADD COLUMN gid serial;
ALTER TABLE silnice ADD PRIMARY KEY (gid);

CREATE INDEX silnice_index ON silnice USING gist (geom);
```



Obr. 2. Vytvořená silniční síť České Republiky

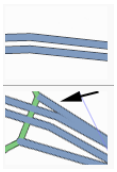
[4] http://wiki.openstreetmap.org/wiki/Cs:Map_Features#Pozemn.C3.AD_komunikace_.28Highway.29

- **Validace vrstvy**

Nejdříve byl pomocí funkce ***SELECT Populate_Geometry_Columns*** zkontrolován sloupce geom a to jestli má vhodná prostorová omezení. Funkce by měla a vrátila pouze jeden řádek s hodnotou 0.

```
SELECT Populate_Geometry_Columns('silnice'::regclass);
```

V PostGISu je velmi důležité, aby jednotlivé vrstvy měli validní geometrii, což je potřeba především pro provádění dotazů. Pro tento účel prostředí PostGISu obsahuje funkce ***ST_IsValid(), ST_MakeValid().*** První funkce zajišťuje vyhledání všech nevalidních prvků v dané vrstvě. Návratovou hodnotou je v případě, že jsou všechny prvky validní hodnota TRUE, v jiném případě výpis nevalidních prvků.



```
SELECT gid FROM silnice WHERE not st_isvalid(geom);
```

Funkce ST_MakeValid() je nově od verze 2.0 a slouží k opravě validity. V tomto případě nebylo nutné využívat neboť vrstva silnice má všechny své prvky validní a tedy v pořádku.

3.2. Vrstva zdravotnická zařízení

Tato vrstva vznikla opět na základě dat poskytovaných pod licencí OpenStreetMap v databázi student_uzpd ve schématu osm vybráním prvků, jež mají ve sloupci *amenity* uloženy hodnoty *baby_hatch, clinic, dentist, doctors, hospital, pharmacy, veterinary.* [5]

Tyto data se však nachází je ve vrstvě bodů, tak ve vrstvě polygonů, a proto vznikly dvě vrstvy, bodová a polygonová. Byly vybrány sloupečky *osm_id, name* (text), *amenity* (text) a *geom*. U bodové vrstvy není nutné kontrolovat geometrii.

```
CREATE TABLE zdravb AS
SELECT osm_id, name AS nazev, amenity AS druh, geom
FROM czech_point
WHERE amenity IN
('baby_hatch','clinic','dentist','doctors','hospital','pharmacy','veterinary');
CREATE TABLE zdravp AS
SELECT osm_id, name AS nazev, amenity AS druh, geom
FROM czech_polygon
WHERE amenity IN
('baby_hatch','clinic','dentist','doctors','hospital','pharmacy','veterinary');
```

- **Odstranění duplicit**

Jelikož se zdravotnická zařízení nachází jak v polygonové tak v bodové vrstvě OpenStreetMap dochází k duplikaci zdravotnických zařízení. To znamená, že jedno

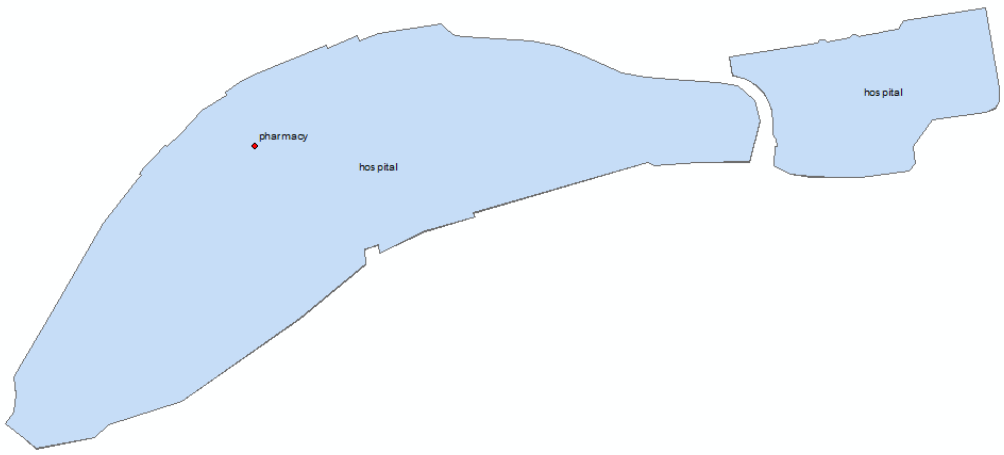
[5] http://wiki.openstreetmap.org/wiki/Cs:Map_Features#Healthcare

zdravotnické zařízení se nachází v obou vrstvách a proto je nutné tyto duplicity odstranit. Byly odstraněny ty body, které leží v polygonu a mají stejná název i druh.

```
DELETE FROM zdravb
WHERE zdravb.geom IN
(SELECT zdravb.geom
FROM zdravb JOIN zdravp
ON ST_WITHIN(zdravb.geom,zdravp.geom) AND zdravb.nazev=zdravp.nazev AND
zdravb.druh=zdravp.druh);
```

Dalším problémem byly data, u nichž chyběl název, nebo se název mírně lišil ve vrstvě polygonové a bodové (Nemocnice Náchod vs. Oblastní nemocnice Náchod). Tyto data byly po visuální kontrole také z bodové vrstvy odstraněny, ale pouze za podmínky, že byly stejného druhu, neboť je častým případem, že součástí nemocnice je například lékárna.

```
DELETE FROM zdravb
WHERE zdravb.geom IN
(SELECT zdravb.geom
FROM zdravb
JOIN zdravp
ON ST_WITHIN(zdravb.geom,zdravp.geom) AND zdravb.druh=zdravp.druh);
```



- **Sloučení vrstev**

Nejdříve bylo nutné převést polygonovou vrstvu zdravotnických zařízení na vrstvu bodovou. To se provede pomocí funkce ***ST_Centroid***, která vrací těžiště polygonových prvků. Následně pomocí funkce *INSERT_INTO* byla takto vzniklá vrstva naimportována do vrstvy zdravb a následně tabulka přejmenována na zdravotnicka_zarizeni.

```
CREATE TABLE zdravp_b AS
SELECT osm_id,nazev,druh, ST_CENTROID(geom) as geom
FROM zdravp;

INSERT INTO zdravb (osm_id,nazev,druh,geom)
```

```
SELECT osm_id,nazev,druh,geom
FROM zdravp_b;

CREATE TABLE zdravotnicka_zarizeni AS
SELECT osm_id, nazev, druh, geom
FROM zdravb;
DROP TABLE IF EXISTS zdravb;
DROP TABLE IF EXISTS zdravp;
DROP TABLE IF EXISTS zdravp_b;
```

- Přidání primárního klíče do sloupce gid

```
ALTER TABLE zdravotnicka_zarizeni ADD COLUMN gid serial;
ALTER TABLE zdravotnicka_zarizeni ADD PRIMARY KEY (gid);

Vytvoření prostorového indexu
CREATE INDEX zdravotnicka_zarizeni_index ON zdravotnicka_zarizeni USING gist
(geom);
```

- **Přejmenování atributů**

Přejmenování atributů bylo provedeno za účelem snadnějšího vyhledávání pomocí dotazů.

```
UPDATE zdravotnicka_zarizeni SET druh = 'babybox' WHERE druh='baby_hatch';
UPDATE zdravotnicka_zarizeni SET druh = 'klinika' WHERE druh='clinic';
UPDATE zdravotnicka_zarizeni SET druh = 'lekarska_ordinace' WHERE druh='doctors';
UPDATE zdravotnicka_zarizeni SET druh = 'zubni_ordinace' WHERE druh='dentist';
UPDATE zdravotnicka_zarizeni SET druh = 'nemocnice' WHERE druh='hospital';
UPDATE zdravotnicka_zarizeni SET druh = 'lekarna' WHERE druh='pharmacy';
UPDATE zdravotnicka_zarizeni SET druh = 'veterinarni_ordinace' WHERE
druh='veterinary';
```

3.3. Vrstva města

Vrstva města byly vytvořena obdobným způsobem jako vrstva zdravotnická zařízení. Byly tedy vybrány města z bodové a polygonové vrstvy OSM s atributy *osm_id, name* (obsahuje název), *place, population, geom*. U bodové vrstvy není nutné kontrolovat geometrii.

```
CREATE TABLE mesta AS
SELECT osm_id,name as nazev, place as druh, population as pocet_obyvatel,geom
FROM czech_point
WHERE place IN ('city','town','village','hamlet') AND name LIKE '%';

CREATE TABLE mestap AS
SELECT osm_id,name as nazev, place as druh, population as pocet_obyvatel,geom
FROM czech_polygon
WHERE place IN ('city','town','village','hamlet') AND name LIKE '%';
```

- **Odstranění duplicit**

U polygonových a bodových dat existují duplicity daných měst. Po podrobném prozkoumání byly z bodové vrstvy odstraněny ty města, která jsou obsaženy ve vrstvě polygonové, bez ohledu na název (Spůle vs. Spůle u Čkyně).

```
DELETE FROM mesta
```

```
WHERE mesta.geom IN
(SELECT mesta.geom from mesta
JOIN mestap
ON ST_WITHIN(mesta.geom,mestap.geom) );
```

- **Sloučení vrstev**

```
CREATE TABLE mestap_b AS
SELECT osm_id,nazev,druh, pocet_obyvatel,ST_CENTROID(geom) as geom
FROM mestap;

INSERT INTO mesta (osm_id,nazev,druh,pocet_obyvatel,geom)
SELECT osm_id,nazev,druh,pocet_obyvatel,geom
FROM mestap_b;

DROP TABLE IF EXISTS mestap;
DROP TABLE IF EXISTS mestap_b;


Přidání primárního klíče a indexu
ALTER TABLE mesta ADD COLUMN gid serial;
ALTER TABLE mesta ADD PRIMARY KEY (gid);

CREATE INDEX mesta_index ON mesta USING gist(geom);
```

3.4. **Vrstvy benzinek** (vrstvy benzinky a benzinky elektrické)

Skupina benzinky je složena z klasických benzinových stanic a ze stanic pro nabíjení elektromobilů. Obě tyto vrstvy vznikly z dat OSM. Klasické benzinky jsou uvedeny v bodových, tak i v polygonových datech OSM. Tyto data je možné vyhledat v občanské vybavenosti pod hodnotou *amenity (fuel, charging_statiton).* Benzinky elektrické jsou pouze v datech bodových. Pro data klasických benzinek je nutné tedy odstranit duplicitní data, kde je benzinka zaznamenána jak v bodové tak polygonové vrstvě. Následně tyto data sloučit do jedné bodové vrstvy, což spočívá především ve výpočtu těžiště polygonu. Postup je stejný jako u vrstev zdravotnicka_zarizeni a města.

```
CREATE TABLE benzinky as
SELECT osm_id,name as nazev, amenity as druh, geom
FROM czech_point
WHERE amenity = 'fuel';

CREATE TABLE benzinkyp as
SELECT osm_id,name as nazev, amenity as druh, geom
FROM czech_polygon
WHERE amenity = 'fuel';

CREATE TABLE benzinky_elektricke as
SELECT osm_id,name as nazev, amenity as druh, geom
FROM czech_point
WHERE amenity = 'charging_station';
```

- **Odstranění duplicit**

```
DELETE FROM benzinky
```

```
WHERE benzinky.geom IN
(SELECT benzinky.geom from benzinky
JOIN benzinkyp
ON ST_WITHIN(benzinky.geom,benzinkyp.geom) );
```

- **Sloučení vrstev**

Pro výpočet bodu v polygonové ploše lze použít funkci PointOnSurface

```
INSERT INTO benzinky
SELECT osm_id,nazev,druh,ST_PointOnSurface(geom) AS geom
FROM benzinkyp;
DROP TABLE IF EXISTS benzinkyp;
```

- Nastavení primárního klíče pro obě vrstvy

```
ALTER TABLE benzinky ADD COLUMN gid serial;
ALTER TABLE benzinky_elektricke ADD COLUMN gid serial;
ALTER TABLE benzinky ADD PRIMARY KEY (gid);
ALTER TABLE benzinky_elektricke ADD PRIMARY KEY (gid);
```

- Nastavení primárního klíče pro obě vrstvy

```
CREATE INDEX benzinky_index ON benzinky USING gist(geom);
CREATE INDEX benzinky_elektricke_index ON benzinky_elektricke USING gist(geom);
```

3.5. Vrstva kraje České Republiky

Data pro vytvoření této vrstvy byly převzaty ze schématu gis1 a z vrstvy obce. Ty byly na základě kódu obce sloučeny pomoci funkce ***ST_Union***. Jelikož data ve schémata jsou v souřadnicovém systému S-TJSK (kód srid 2065) bylo nutné je pomocí funkce ***ST_Transform*** natransformovat do souřadnicového systému OpenStreetMap, které používají souřadnicový systém Google Mercator (kód srid 900913).

```
CREATE TABLE kraje as
SELECT nk as nazev_kraje, ST_UNION(geom) AS geom from obce GROUP BY nazev_kraje;
UPDATE kraje SET geom = ST_Transform(geom,900913);

ALTER TABLE ADD COLUMN gid serial;
ALTER TABLE kraje ADD PRIMARY KEY (gid);
CREATE INDEX kraje_index ON kraje USING gist (geom);

SELECT gid FROM kraje WHERE not st_isvalid(geom);
   (0 rows)
```

3.6. Úprava dat pro PgRouting

Pro potřeby práce se síťovými analýzami je nutné vrstvu silnice upravit tak, aby mohla být vstupem do algoritmů PgRoutingu. Za tímto účelem byly do liniové vrstvy silnic přidány sloupce, které obsahují id počátečního *(source),* id koncového uzlu *(target)* a délku *(length)* každé dílčí linie. Na rozdíl od sloupce source a target, které mají datový typ INTEGER, musí mít sloupec length datový typ FLOAT. Tabulka a přiřazení uzlových bodů bylo vytvořeno pomocí funkce ***assign_vertex_id***.

```
ALTER TABLE silnice ADD COLUMN source INTEGER;
ALTER TABLE silnice ADD COLUMN target INTEGER;
ALTER TABLE silnice ADD COLUMN length FLOAT;

CREATE INDEX source_index ON silnice(source);
CREATE INDEX target_index ON silnice(target);
```

- Vytvoření topologie

```
SELECT assign_vertex_id('b13','silnice',1,'geom','gid');
```

- Výpočet délky jednotlivých úseků

```
UPDATE silnice SET length  = ST_Length(geom);
```

3.6.1. Vyhledávání nejkratší cesty

V našem projektu se budeme zaměřovat na algoritmy PgRoutingu, které vyhledávají nejkratší vzdálenost mezi dvěma uzly. Nejjednodušším nástrojem pro vyhledání nejkratší cesty je ***Dijkstrovův algoritmus,*** který byl také prvním implementovaným algoritmem pro PgRouting vůbec. Algoritmus je pojmenovaný po nizozemském informatikovy. Vstupy algoritmu:

```
CREATE OR REPLACE FUNCTION shortest_path(
                                                sql text,
                                                source_id integer,
                                                target_id integer,
                                                directed boolean,
                                                has_reverse_cost boolean)
        RETURNS SETOF path_result
```

SQL text – sql dotaz vracející následující hodnoty z tabulky (v našem případě vrstva silnice)

```
SELECT id, source, target, cost FROM edge_table
```

Dalším metodou výpočtu je ***algoritmus A-STAR***, který je výhodný pro výpočty větších data setů. Na rozdíl od předešlého je založen heuristické metodě (náhodné hledání). Vstup do algoritmu je obdobný. Liší se v názvu (shortest_path_astar) a ve vstupu sql textu, ze kterého mimo jiné vystupují i souřadnice uzlových bodů. Z tohoto důvodu je nutné upravit vrstvu silnice a přidat sloupce se souřadnicemi uzlových bodů.

```
SELECT id, source, target, cost, x1, y1, x2, y2 FROM edge_table
```

- Přidání sloupců pro souřadnice uzlových bodů

```
ALTER TABLE silnice ADD COLUMN x1 DOUBLE PRECISION;
ALTER TABLE silnice ADD COLUMN y1 DOUBLE PRECISION;
ALTER TABLE silnice ADD COLUMN x2 DOUBLE PRECISION;
ALTER TABLE silnice ADD COLUMN y2 DOUBLE PRECISION;
```

- Výpočet souřadnic uzlových bodů

```
UPDATE silnice SET x1=ST_x(ST_Startpoint(geom));
UPDATE silnice SET y1=ST_y(ST_Startpoint(geom));
UPDATE silnice SET x2=ST_x(ST_Endpoint(geom));
UPDATE silnice SET y2=ST_y(ST_Endpoint(geom));
```

Poslední v tomto dokumentu popisovaným bude ***algoritmus Shooting Star***. Jeho rozdílnost od předešlých dvou je ve způsobu výpočtu nejkratší cesty, ke kterému používá hrany a ne uzly. Opět se liší ve vstupu v hodnotě sql text, jejíž výstupem je ještě hodnota rule, to_cost. Tyto sloupce bylo tedy nutné do vrstvy silnice přidat.

```
SELECT id, source, target, cost, x1, y1, x2, y2, rule, to_cost FROM edges
```

- Přidání sloupců rule, to_cost

```
ALTER TABLE silnice ADD COLUMN rule TEXT;
ALTER TABLE silnice ADD COLUMN to_cost DOUBLE PRECISION;
```

3.6.2. Vyhledání uzlů

Našim cílem je vytvoření především nejkratší vzdálenosti mezi jednotlivými městy a zdravotnickými zařízeními, které jsou charakterizovány jednotlivými body. Metody hledání nejkratší cesty popsané víše, však hledají nejkratší cestu mezi uzlovými body vrstvy silnice. Nejlepším řešením by bylo najití nejbližšího bodu na silnici a z tohoto bodu vytvořit další uzlový bod a znovu vypočíst celou topologii znovu. Pro naše potřeby jsme však zvolily postup, kdy hledáme nejbližší existující uzlový bod z předešlé topologii a to pomocí funkce ***find_node_by-nearest_link_within_distance***. Vstupem do této funkce jsou mimo jiné také hodnoty vypočtené z funkce ***find_nearest_link_within_distance***.

Po problémech s výpočty těmito funkcemi bylo zjištěno, že pro náš případ je nutné tyto funkce upravit. Ve zdrojovém kódu byly k funkcím přidány přípony `ST_` (např. `ST_GeometryFromText`) a hodnota `the_geom` nahrazena hodnotou `geom`.

4. Atributové dotazy

4.1. Kolik zdravotnických zařízení máme v databázi

```
select count(*) from zdravotnicka_zarizeni;
```

- ✓ *87.8307*

4.2 Vypište všechny benzínky, jejichž název začíná na "A" a obsahuje písmeno p

```
select nazev from benzinky where nazev like 'A%p%';
```

- ✓ *"Agip","Autosprint Veletiny","Autogas Opava"*

4.3 Kolik km silnic máme v databázi

```
select SUM(length)/1000 AS delka FROM silnice;
```

- ✓ *96270.3*

4.4. Který kraj má největší rozlohu

```
select * from kraje order by area limit 1;
```

- ✓ *HK*

4.5 Počet obyvatel v krajích

```
select nk , SUM (obyvatel)
from obce group by nk;
```

- ✓

"JC";625097
"PA";506534
"JM";1121792
"ST";1128674
"US";819712
"KA";304220
"LB";427321
"OL";636750
"VY";517630
……

4.6 Kolik procent silnic v databázi nemá jméno

```
select ((select count(osm_id) from silnice where nazev is null )*10.0 / (select
count(osm_id) from silnice)*10.0)
as pomer from silnice limit 1;
```

- ✓ *73.559*

4.7 Kolik procent silnic vzhledem ke své délce je bezejmenných

```
select (select sum(length) from silnice where nazev is null)/
(select sum(length) from silnice  )*100;
```

- ✓ *87.8307007772788*

5. Prostorové dotazy

5.1 Který kraj má nejvíce benzinek vzhledem ke své ploše

```
SELECT * from (select nazev_kraje, count(*)/area as pomer
from benzinky join kraje on ST_Intersects(benzinky.geom, kraje.geom) group by
area, nazev_kraje)
as maxpomer group by pomer, nazev_kraje order by pomer desc limit 1;
```

- ✓ *HP*

5.2 Která města mají do 10km od centra alespoň 1 elektrickou benzinku

```
select mesta.nazev from mesta
join benzinky_elektricke
on ST_DWithin(benzinky_elektricke.geom, mesta.geom, 10000);
```

"Vestec"
"Zdiměřice"
"Rozkoš"
"Hole"
"Průhonice"
"Bobrovníky"
"Ludgeřovice"
"Vrbice"
"Ostrava"
……

5.3 Kolik benzinek má v okruhu 50m další benzinku

```
select count(b2.osm_id) from benzinky as b1
join benzinky as b2 on (ST_Dwithin(b1.geom,b2.geom,50))
and ST_Distance(b1.geom,b2.geom)>0;
```

- ✓ *110*

5.4 Jaká je prima vzdálenost z Aše do Brna

```
select St_Distance(as1.geom, brno.geom)/1000 as Vzdalenost
from obce as as1, obce as brno
where as1.nazev='Aš' and brno.nazev='Brno';
```

- ✓ *317*

5.5 Nemocnice nejblíže centroidu cr

```
select * from kraje as k
join zdravotnicka_zarizeni as zz
on st_intersects(st_buffer((select st_centroid(st_union(geom)) from kraje),30000)
,zz.geom)
where druh = 'nemocnice'
order by st_distance((select st_centroid(st_union(geom)) from kraje),zz.geom) desc
limit 1;
```

- ✓ *JC kraj, poliklinika*

5.6 Kolik nemocnic má vzdálenou lékárnu do 300 metru

```
select nem.osm_id from (select * from zdravotnicka_zarizeni where druh =
'lekarna') as lek
join (select * from zdravotnicka_zarizeni where druh = 'nemocnice') as nem
on st_DWithin(nem.geom,lek.geom, 300)group by nem.osm_id;
```

- ✓ *157*

5.7 Který kraj má nejvíce km silnic

```
select k.nazev_kraje,(sum(length)/1000) as delka_silnic from silnice as s join
kraje2 k

on st_intersects(k.geom, s.geom) group by k.nazev_kraje order by sum(length) desc
limit 1;
```

- ✓ *ST - 16981km*

6. Síťové analýzy

6.1 cesta z prahy do Liberce

```
SELECT sum(cost)/1000 FROM shortest_path(
'SELECT gid AS id,source,target,length AS cost FROM b13.silnice',
(SELECT id FROM find_node_by_nearest_link_within_distance(
(SELECT ST_AsText(geom) FROM mesta WHERE nazev = 'Praha'),1000,'silnice')),
(SELECT id FROM find_node_by_nearest_link_within_distance(
(SELECT ST_AsText(geom) FROM mesta WHERE nazev =
'Liberec'),1000,'silnice')),false,false);
```

- ✓ *výsledek 168,9*

6.2 kolik zdravotních zařízeni je v okruhu 500m od cesty z prahy do Dolních Jirčan

```
SELECT DISTINCT zz.gid, zz.nazev, zz.druh
FROM (SELECT ST_buffer(geom,500) AS buffer,silnice.gid
FROM shortest_path('SELECT gid AS id,source,target,length AS cost
FROM b13.silnice',
(select id from find_node_by_nearest_link_within_distance((SELECT ST_AsText(geom)
FROM mesta WHERE nazev = 'Praha'),5000,'silnice')),
(select id from find_node_by_nearest_link_within_distance((SELECT ST_AsText(geom)
FROM mesta WHERE nazev = 'Dolní Jirčany'),5000,'silnice'))
,false,false) AS path
JOIN silnice ON path.edge_id=silnice.gid) AS A
JOIN zdravotnicka_zarizeni as zz ON ST_intersects(A.buffer,zz.geom) ;
```

- ✓ *výsledek 37 řádků (zdravotních zařízeni)*

6.3 Které obce míjíme, jedeme-li z Prahy do Dolních Jirčan

```
SELECT DISTINCT mesta.nazev
FROM (SELECT ST_buffer(geom,100) AS buffer,silnice.gid
FROM shortest_path('SELECT gid AS id,source,target,length AS cost,x1,y1,x2,y2
FROM silnice',
(select id from find_node_by_nearest_link_within_distance((SELECT ST_AsText(geom)
FROM mesta WHERE nazev = 'Praha'),5000,'silnice')),
(select id from find_node_by_nearest_link_within_distance((SELECT ST_AsText(geom)
FROM mesta WHERE nazev = 'Dolní Jirčany'),5000,'silnice'))
, false, false) AS path
JOIN silnice ON path.edge_id=silnice.gid) AS A
JOIN mesta ON ST_intersects(A.buffer,mesta.geom);
```

- ✓ *výsledek 6 - Psary, Hlubocina, Radejovice....*

7. Závěr

Cílem této dokumentace je popsat práci s prostorovými daty v prostředí PostGIS, která probíhala v rámci předmětu Úvod do zpracování prostorových dat. Výsledkem naší práce je vytvoření šesti validních tematických vrstev z databáze OSM a GIS1, a sada atributových, prostorových a síťových dotazů. Tato dokumentace může pomoct vysvětlit a především vyzkoušet některé funkce nadstavby PostGISu, kterou je Pgadmin. Veškeré zde prováděné dotazy jsou uloženy v souboru dávka a je tedy možné si všechny příkaz vyzkoušet.