Česky vysoké učení technické v Praze
Fakulta stavební
Studijní program: Geodézie a kartografie
Obor: Geoinformatika

ÚVOD DO ZPRACOVÁNÍ PROSTOROVÝCH DAT - PROJEKT

Vyhledání přerušených řek v místě vodní plochy a následná interpolace chybějící části

Skupina B:

Renata Duchnová
Tereza Pantůčková
Petra Svobodová
Miroslav Kopecký

.

Obsah

1 Úvod

Úkolem bylo vytvoření tématických vrstev na základě dat OpenStreetMap. Pro tento účel byla na serveru `josef` založena databáze `pgis_osm`. Měly být aplikovány testy datové integrity a odstraněny případné nekonzistence v datech. Závěrem měla být vytvořena sada atributových a prostorových dotazů nad databází.

1.1 Volba tématu

Při vytváření tématických vrstev jsme zjistili, že data, se kterými máme pracovat jsou neúplná. Neúplnost se projevovala v nespojitosti liniových prvků (řeky, silnice) nebo v nevalidních plochách. Proto jsem se rozhodli, že pozměníme zadání. Pokusili jsme se doplnit aspoň malou část dat. Zaměřili jsme se na vyhledání přerušených řek v místě vodní plochy a následné interpolace chybějící části. K tomuto účelu byl sepsál skript v pythonu. Zároveň jsme vytvořili další vrstvy a sepsaly dotazy.

2 Vytvoření vrstev pomocí dat z databáze

K vytvoření vrstev jsme použili cvičnou databázi `pgis_osm`, která obsahuje OpenStreetMap data ČR. OpenStreerMap je projekt, jehož cílem je tvorba volně dostupných geografických dat a následně jejich vizualizace. My jsem vybrali data týkající se řek a vodních ploch.

- vrstva b10.reky
  ```
  CREATE TABLE b10.reky AS SELECT osm_id, name, way FROM czech_line WHERE waterway =
  'river';
  ```

- vrstva b10.vod_plochy_naz
  ```
  CREATE TABLE b10.vod_plochy_naz AS SELECT osm_id, name, way FROM czech_polygon WHERE
  landuse IN ('reservoir','basin') OR czech_polygon.natural = 'water';
  ```

- vrstva b10.sidla
  ```
  CREATE TABLE b10.sidla AS SELECT osm_id, name, way, place FROM czech_point WHERE place
  IN ('city','town','village','hamlet');
  ```

- vrstva b10.nadrazi
  ```
  CREATE TABLE b10.nadrazi AS SELECT osm_id, name, way, ref FROM czech_point WHERE railway
  IN ('station', 'halt');
  ```

- vrstva b10.zeleznice
  `CREATE TABLE b10.zeleznice AS SELECT osm_id, way, ref FROM czech_point WHERE railway IN ('rail');`

Ke všem tabulkám byl přidán primární klíč.

př. `ALTER TABLE b10.reky ADD PRIMARY KEY (osm_id);`

Dále bylo potřeba vytvořit atribut geometrie.

př. `SELECT Populate_geometry_columns('b10.reky'::regclass);`

Pro zobrazení vrstev v QGISu byly všem tabulkám přiděleny příslušná práva.

př. `GRANT SELECT ON b10.reky TO postgis;`

2.1 Vytvořené dotazy

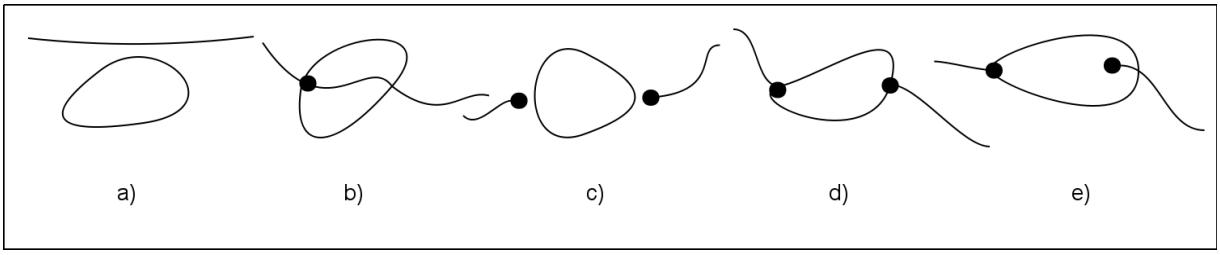
Vytvořené dotazy jsou přiloženy jako další dokument (prostý text). Vrstvy nejsou žádným způsobem ošetřeny.

3 Vyhledání přerušených řek a v místě vodní plochy

3.1 Princip

Obecný princip vyhledání příslušných vodních ploch je nalezení právě 2 koncových bodů linií řek v místě vodní plochy nebo v okolí 175 m, 350 m, 500 m, které nejsou spojeny linií. Jsou právě 2, neboť se v místě nebo okolí plochy často vyskytují 3 a více koncových bodů, které ale nejsme schopni správně spojit na základě interpolace - například případ soutoku. Nejprve byly tedy vytvořeny buffery 175 m, 350 m a 500 m kolem všech vodních ploch a následně bylo testováno, zda se v místě plochy a jejím okolí nachází právě 2 koncové body řek. Vybrané případy byly pak sjednoceny do 1 tabulky. Celkový počet ploch vstupujících do interpolace je 23.

3.2 Případy vodních ploch a řek



a) řeka plochou neprotéká
b) řeka je přerušena v jednom bodě, ale protéká celou plochou
c) řeka je přerušena, žádný její konec neleží na hranici plochy
d) řeka je přerušena, konce řeky leží na hranici plochy
e) řeka je přerušena, první konec leží na hranici plochy, druhý uvnitř plochy

c), d), e) jsou hledané případy

3.3 Výsledné tabulky

Tyto tabulky jsou vstupem do skryptu `prehrady.py`, který provádí samotnou interpolaci. V tabulce `vysl_body` je uložena geometrie koncových bodu řek (sloupec body), které se nachazí v míste vodní plochy nebo v jejím okoli. V druhé tabulce, `vysl_plochy`, je geometrie vodních ploch (way), pomocí kterých doplňujeme řeky. Obě tabulky jsou spojené na základě `id_plochy` a `id_reky`.

3.3.1 Použité dotazy při vytváření tabulek

1 Výběr vodních ploch, které mají s řekami nějaký průnik
- zde problém spočíval v tom, že se vybraly i plochy, přes které řeka teče a není rozpojená

```
SELECT id_plochy, name_plochy, r.osm_id AS id_reky, r.name AS name_reky,
ST_Dimension(ST_Intersection(r.way,geom_plochy)) FROM (SELECT v.osm_id AS id_plochy, v.name
AS name_plochy, v.way AS geom_plochy FROM b10.vodni_plochy_naz AS v JOIN b10.reky AS r
ON ST_Intersects(r.way, v.way)) AS p JOIN b10.reky AS r ON ST_Intersects(r.way, geom_plochy)
GROUP BY id_plochy, name_plochy, id_reky, name_reky, r.way, geom_plochy ORDER BY name_plochy;
```

2 Hledání vodních ploch, které mají s řekami průnik právě 2 body
- výhoda: vyberou se pouze plochy, v jejichž místě je řeka rozpojená
- nevýhoda: 2 body leží jen na hranicích vodní plochy, nevyberou se případy, kdy končí řeka uvnitř plochy (to jsou případy, kdy je průnik plochy s řekou linie, která však není přes celou plochu a končí někde uvnitř polygonu - v sql nelze ošetřit zda jsou části řeky spojené nebo ne.)
- řešení: vytvoření tabulky s koncovými body řek a vybrat ty vodní plochy, které v sobě mají právě 2

koncové body + to samé udělat pro okolí vodní plochy

```
SELECT id_plochy, name_plochy, r.osm_id AS id_reky, r.name AS name_reky FROM (SELECT v.osm_id
AS id_plochy, v.name AS name_plochy, v.way AS geom_plochy, COUNT(v.osm_id) AS pocet FROM
b10.vodni_plochy_naz AS v JOIN b10.reky AS r ON ST_Intersects(r.way, v.way) and
ST_Dimension(ST_Intersection(r.way, v.way))=0 GROUP BY v.osm_id, v.name, v.way) AS p JOIN
b10.reky AS r ON ST_Intersects(r.way, geom_plochy) and ST_Dimension(ST_Intersection(r.way,
geom_plochy))=0 WHERE pocet=2 ORDER BY name_plochy;
```

3 Vytvoření bufferu kolem všech ploch

```
CREATE TABLE b10.plochy_buffer350 AS SELECT osm_id, name, ST_Buffer(way, 350)
FROM b10.vodni_plochy_naz;
CREATE TABLE b10.plochy_buffer175 AS SELECT osm_id, name, ST_Buffer(way, 175)
FROM b10.vodni_plochy_naz;
CREATE TABLE b10.plochy_buffer500 AS SELECT osm_id, name, ST_Buffer(way, 500)
FROM b10.vodni_plochy_naz;
```

4 Tabulka s počátečními a koncovými body řek

```
CREATE TABLE b10.reky_body AS SELECT osm_id, name, ST_Startpoint(way),ST_Endpoint(way)
FROM b10.reky;
```

5 Sjednocení koncových a počátečních bodů řek

```
CREATE TABLE b10.reky_body_union AS SELECT osm_id, name, ST_Union(ST_Startpoint(way),
ST_Endpoint(way)) FROM b10.reky;
```

6 Vytvoření čtyř tabulek s geometrií plochy a koncových bodů řek, které leží v ploše
(u každé plochy dva)

```
CREATE TABLE b10.plochy2body AS SELECT p.osm_id, p.name, p.st_buffer,
ST_Numgeometries(ST_Intersection(st_buffer, st_union)),
ST_Intersection(st_buffer, st_union) FROM b10.plochy_buffer350 AS p
JOIN b10.reky_body_union AS r ON (ST_Intersects(st_buffer, st_union))
WHERE ST_Numgeometries(ST_Intersection(st_buffer, st_union)) = 2;

   CREATE TABLE b10.plochy2body2 AS SELECT p.osm_id, p.name, p.way,
ST_Numgeometries(ST_Intersection(st_buffer, st_union)),
ST_Intersection(st_buffer, st_union) FROM b10.vodni_plochy_naz AS p
JOIN b10.reky_body_union AS r ON (ST_Intersects(way, st_union))
WHERE ST_Numgeometries(ST_Intersection(way, st_union)) = 2;

   CREATE TABLE b10.plochy2body3 AS SELECT p.osm_id, p.name, p.st_buffer,
ST_Numgeometries(ST_Intersection(st_buffer, st_union)),
```

```
ST_Intersection(st_buffer, st_union) FROM b10.plochy_buffer175 AS p
JOIN b10.reky_body_union AS r ON (ST_Intersects(st_buffer, st_union))
WHERE ST_Numgeometries(ST_Intersection(st_buffer, st_union)) = 2;

   CREATE TABLE b10.plochy2body4 AS SELECT p.osm_id, p.name, p.st_buffer,
ST_Numgeometries(ST_Intersection(st_buffer, st_union)),
ST_Intersection(st_buffer, st_union) FROM b10.plochy_buffer500 AS p
JOIN b10.reky_body_union AS r ON (ST_Intersects(st_buffer, st_union))
WHERE ST_Numgeometries(ST_Intersection(st_buffer, st_union)) = 2;
```

7 Sjednocení všech čtyř tabulek

```
CREATE TABLE b10.plochy2body_u AS (SELECT osm_id, name, st_intersection
FROM b10.plochy2body) UNION (SELECT osm_id, name, st_intersection FROM b10.plochy2body2)
UNION (SELECT osm_id, name, st_intersection FROM b10.plochy2body3) UNION
(SELECT osm_id, name, st_intersection FROM b10.plochy2body4) ORDER BY name;
```

8 Vytvoření tabulky ploch, které se nebudou doplňovat (koncové body řek jsou spojené)

```
CREATE TABLE b10.nedopl_plochy AS SELECT p.osm_id AS id_plochy, p.name AS name_plochy,
r.osm_id AS id_reky, r.name AS name_reky, st_startpoint, st_endpoint
FROM b10.plochy2body_u AS p JOIN b10.reky_body AS r
ON ST_Numgeometries(ST_Intersection(st_intersection,
ST_Union(st_startpoint, st_endpoint))) = 2 ORDER BY name_plochy;
```

9 Vytvoření tabulky s názvy vodních ploch na doplnění

```
CREATE TABLE b10.vysledek AS (SELECT osm_id AS id_plochy, name AS name_plochy
FROM b10.plochy2body_u) EXCEPT (SELECT id_plochy, name_plochy FROM b10.nedopl_plochy)
ORDER BY name_plochy;
```

10 Doplnění geometrie bodů

```
CREATE TABLE b10.vysl3 AS SELECT id_plochy, name_plochy, st_intersection AS body
FROM b10.vysledek JOIN b10.plochy2body_u ON id_plochy=osm_id ORDER BY name_plochy;
```

11 Vytvoření tabulky s geometrií koncových bodů, odstranění některých vodních ploch

- řeka je mezi koncovým bodem jedné řeky a lomovým (nekoncovým) bodem druhé řeky (Šárecký potok, Vltava)
- koncové body dvou různých řek nemají být spojené

```
CREATE TABLE b10.vysl_body AS SELECT id_plochy, name_plochy, osm_id AS id_reky, name
AS name_reky, ST_Union(st_startpoint, st_endpoint) AS body
FROM b10.vysl3 JOIN b10.reky_body
```

```
ON ST_Intersects(body, ST_Union(st_startpoint, st_endpoint)) WHERE id_plochy
NOT IN (26376257, 51255318, 29489992, 51681002, 26376254) ORDER BY name_plochy;
```

12 Vytvoření tabulky s geometrií řek

```
CREATE TABLE b10.vysl_reky AS SELECT id_plochy, name_plochy, id_reky, name_reky, way
FROM b10.vysl_body JOIN b10.reky ON osm_id = id_reky ORDER BY name_plochy;
```

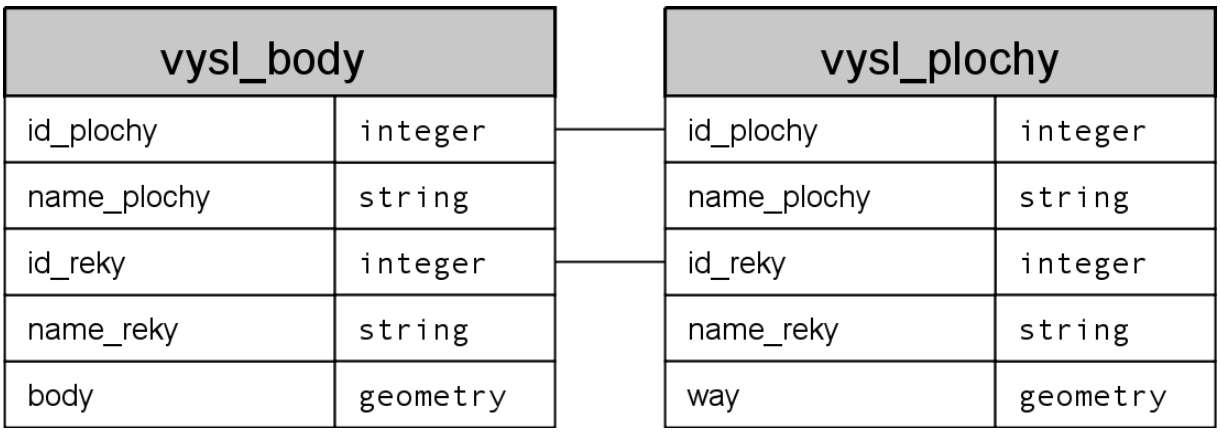
13 Vytvoření tabulky s geometrií ploch

```
CREATE TABLE b10.vysl_plochy AS SELECT id_plochy, name_plochy, id_reky, name_reky,
way FROM b10.vysl_body JOIN b10.vodni_plochy_naz
ON osm_id = id_plochy ORDER BY name_plochy;
```

14 Vytvoření tabulky s geometrií ploch, pouze s id ploch (pro zobrazení v QGisu)

```
CREATE TABLE b10.vysl_jenplochy AS SELECT id_plochy, name_plochy, way
FROM b10.vysl_plochy GROUP BY id_plochy, name_plochy, way;
```

3.3.2 Výsledné tabulky - schéma



4 Python - skript

4.1 Propojení s databází

Do databázi se připojuje skript `prehrady.py` pomocí funkce `psycopg2`, jehož parametry jsou název databáze a jméno uživatele. Tento skript vyhledá koncové body kusů řeky a příslušnou plochu. Protože není poznat ze souřadnic směr, bylo potřeba otestovat, které body jsou k ploše blíž. Pak se zavolá funkce `najdi_cestu` z modulu `uloha.py`.

- `psycopg2.connect()`
  - propojení s databází

4.2 Interpolace chybějící části řeky

Funkce `najdi_cestu` vezme geometrii plochy a provede v ní triangulaci pomocí algoritmu převzatého z webové stránky[1]. Po rozdělení se vypočítají těžište. Tato těžište pak spolu s jednotlivými konci řeky představují graf, ve kterém je potřeba najít nejkratší cestu mezi jednotlivými konci řeky. K tomuto účelu byl použit klasický Dijkstruv algoritmus implementovaný v modulu `dijkstra.py`, který jsme převzali z webové stránky[2].

- `preved()`
  - převede textovou reprezentaci na čísla
- `zjisti_blizke_body()`
  - zjistí, který bod je nejbližší k polygonu pro první řeku i pro druhou řeku
- `vypocti_drahu()`
  - vypočítá dráhu z bodu start do bodu end pres přehradu
- `triangulate()`
  - rozdělí obecný n-úhelnik na sadu trojúhelníků
- `shortestPath()`
  - klasický Dijkstruv algoritmus
- `get_nearest_center()`
  - vyhledá koncový bod a určí vzdálenost k nejbližšímu těžišti

5 Co je ještě potřeba...

Kvůli časové tísni se nepovedlo začlenit vypočtené body do databáze a spojit linií. Je potřeba napsat ještě jednu funkci, která by tento úkol provedla. Použitý algoritmus aproximace není příliš vhodný, bylo by potřeba zaměnit ho za jiný, který by minimalizoval počet trojúhelníku a maximalizoval jejich obsah. Tím by se dosáhlo opravdu nejbližší vzdálenosti. Další možností je použití proceduralního jazyka PL/pgSQL.

[1] http://www.siggraph.org/education/materials/HyperGraph/scanline/outprims/polygon1.htm
[2] http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/117228

6 Závěr

Úkolem tohoto předmětu bylo rozšíření znalostí v oblasti databáze a pochopení zpracování prostorových dat. Vyzkoušeli jsme si, jak se dají taková data (z OpenStreetMap databáze) použít, případně opravit. Schéma tabulek potřebných pro vytvoření atributových a prostorových dotazů v této zprávě neuvádíme, protože jsme jako hlavní úkol měli opravit vrstvu řek. I když jsme nedošli k úplnému konci, myslíme si, že byl tento předmět poučný a zajímavý.

Autoři projektu:

Renata Duchnová
Tereza Pantůčková
Petra Svobodová
Miroslav Kopecký

Reference

[1] Stránky předmětu Úvod do zpracování prostorových dat
http://gama.fsv.cvut.cz/wiki/index.php/153UZPD

[2] Polygon Decomposition into Triangles
http://www.siggraph.org/education/materials/HyperGraph/scanline/outprims/polygon1.htm

[3] Klasický Dijkstruv algoritmus
http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/117228

[4] Wikipedia
http://www.wikipedia.org/