

# Úvod do zpracování prostorových dat

Kvalita bydlení ve vybraných částech Prahy

Zimní semestr 2018/2019

Tereza Kulovaná  
Markéta Pecenová

# Obsah

<b>1</b>	<b>Zadání</b>	<b>2</b>
1.1	Zadání projektu . . . . .	2
1.2	Zvolené téma . . . . .	2
<b>2</b>	<b>Data</b>	<b>3</b>
2.1	Data a tématické vrstvy . . . . .	3
2.1.1	ruain_praha . . . . .	3
2.1.2	Otevřená data . . . . .	3
2.1.3	Geoportál Praha . . . . .	3
<b>3</b>	<b>Zpracování dat</b>	<b>4</b>
3.1	ruian_praha . . . . .	4
3.2	Otevřená data . . . . .	4
3.3	Geoportál Praha . . . . .	6
<b>4</b>	<b>SQL dotazy</b>	<b>8</b>
<b>5</b>	<b>Závěr</b>	<b>11</b>
<b>6</b>	<b>Přílohy</b>	<b>12</b>
<b>7</b>	<b>Zdroje</b>	<b>13</b>

# 1 Zadání

## 1.1 Zadání projektu

Navrhňte a vytvořte tématické vrstvy (např. vodní toky, vodní plochy, lesy, silnice, železnice apod.) na základě dat *OpenStreetMap* a další otevřených zdrojů. Aplikujte testy datové integrity a odstraňte případné nekonzistence v datech. Vytvořte tutoriál - tj. sadu atributových a prostorových dotazů nad databází **pgis\_uzpd**.

## 1.2 Zvolené téma

Jako téma pro semestrální projekt byla zvolena analýza kvality bydlení v Praze. Jelikož je toto téma velmi obecné a obsáhlé, bylo nutné zvolit užší zaměření. Z mnoha podkladů, které byly k dispozici, byly vybrány datové vrstvy, které nějak souvisely s občanskou vybaveností nebo zdravím obyvatel Prahy. Výsledným produktem je sada atributových a prostorových dotazů, které vyhodnocují kvalitu místa pro život na základě zvolených vstupních ukazatelů kvality.

## 2 Data

V rámci projektu bylo nutné získat potřebná vstupní data, nahrát je do databáze a zajistit jejich konzistenci. Nad těmito daty pak byly následně provedeny prostorové dotazy.

### 2.1 Data a tématické vrstvy

Data použitá v rámci projektu pochází ze tří zdrojů: ze schématu *ruian\_praha*, portálů *Otevřená data* a *Geoportál Praha*.

#### 2.1.1 ruian\_praha

Schéma *ruian\_praha* je součástí databáze **pgis\_uzpd** a je v souřadnicovém systému JTSK.

Vrstvy:

- *adresni\_mista* (adresnimista)
- *obvody* (spravniobvody)

#### 2.1.2 Otevřená data

Veškeré datové vrstvy, které byly staženy z portálu *Otevřená data*, mají uvedené jako poskytovatele *HLAVNÍ MĚSTO PRAHA* a jsou vztažena pouze na území Prahy. Geometrie všech vrstev je reprezentována bodem. Data byla stažena ve formátu geoJSON a až na poslední uvedenou vrstvu byla v souřadnicovém systému WGS84 (S-JTSK nebyl k dispozici). Pouze *Vstupy do metra* byly staženy přímo v S-JTSK.

Vrstvy:

- *detska\_hriste* (Dětská hřiště Praha, [Zdroj])
- *zdrav\_zarizeni* (Lékárny a zdravotnická zařízení v Praze, [Zdroj])
- *metro* (Vstupy do metra, [Zdroj])

#### 2.1.3 Geoportál Praha

Ze stránek *Geoportál Praha* byla stažena tématická data zachycující rozmístění košů na tříděný odpad a data zaplaveného území Prahy při povodních v roce 2013. Data byla stažena ve formátu geoJSON v systému JTSK. Geometrie rozmístění košů je *bod* a zaplaveného území *polygon*. Pro povodňová data bylo k dispozici ke stažení vrstev mnohem více, například záplavové čáry pro stoletou/padesátiletou/dvacetiletou vodu atd. Pro zjednodušení byla stažena data, který byla ucelená (zachycující jedno období) a nejaktuálnější.

Vrstvy:

- *odpad* (Mapa košů na tříděný odpad, [Zdroj])
- *zaplavy2013* (Záplavové území 2013, [Zdroj])

## 3 Zpracování dat

### 3.1 ruian\_praha

#### Adresní místa

Data byla stažena ze schématu *ruian\_praha* (vrstva *adresnimista*) do nově vytvořené tabulky *adresni\_mista*. Z původních dat byly zkopírovány pouze sloupce *kód*, *číslo popisné*, *číslo orientační* a sloupec s geometrií *geom*.

```
1 CREATE TABLE adresni_mista AS
2   SELECT kod, cislodomovni, cisloorientacni, geom
3   FROM ruian_praha.adresnimista
```

Následně byly nad sloupkem *kód* nastaven primární klíč a nad sloupkem *geom* prostorový index.

```
1 ALTER TABLE adresni_mista ADD PRIMARY KEY(kod)
2
3 CREATE INDEX adresy_index ON adresni_mista(geom)
```

Na závěr byla provedena kontrola validity geometrie, která vyšla negativní.

```
1 SELECT kod FROM adresni_mista WHERE NOT st_isvalid(geom)
```

Při zběžném prohlédnutí již zpracovaných dat bylo zjištěno, že výše uvedená funkce na kontrolu validity nefunguje zcela správně, jelikož neodstranila záznamy které měly ve sloupečku pro geometrii uvedenou hodnotu *NULL*. Bylo tedy nutné provést dodatečné pročištění dat níže uvedeným příkazem:

```
1 DELETE FROM adresni_mista WHERE geom IS NULL
```

#### Správní obvody Prahy

Data byla stažena ze schématu *ruian\_praha* (vrstva *spravniobvody*) do nově vytvořené tabulky *obvody*. Geometrie dat je *multipolygon* a představují jednotlivé pražské správní obvody. Z původních dat byly zkopírovány pouze sloupce *ogc\_fid*, *název* a sloupec s geometrií *geom*.

```
1 CREATE TABLE obvody AS
2   SELECT ogc_fid, nazev, geom
3   FROM ruian_praha.spravniobvody
```

Následně byly nad sloupkem *ogc\_fid* nastaven primární klíč.

```
1 ALTER TABLE obvody ADD PRIMARY KEY(ogc_fid)
```

Na závěr byla provedena kontrola validity geometrie, která vyšla negativní.

```
1 SELECT nazev FROM obvody WHERE NOT st_isvalid(geom)
```

### 3.2 Otevřená data

Všechna data z tohoto zdroje byla stažena ve formátu geoJSON a byla převážně v systému WGS84 (výjimka: vrstva *metro* bylo od počátku v S-JTSK). Postup při zpracování těchto dat byl obdobný. Data byla importována do příslušného schématu pomocí nástroje *ogr2ogr* a přímo v databázi byly v tabulkách promazány nepotřebné sloupce a

přejmenován sloupeček s geometrií, který po nahrání dávkou získal uživatelsky nepřívětivý název *wkb\_geometry*.

Během importu do databáze bylo u některých dat nutné provést transformaci ze systému WGS84 (EPSG: 4326) do S-JTSK (EPSG: 5514). Transformaci zajišťovala část kódu `-t_srs 'EPSG:5514'`. Nebyla-li transformace nutná, nahradila se tato část kódu za `-a_srs 'EPSG:5514'`. Při importu byl automaticky vytvořen sloupeček s primárním klíčem *ogc\_fid* a byly vytvořeny prostorové indexy. Níže je uveden obecný kód pro nahrání souboru do příslušného schématu databáze (s transformací do S-JTSK):

```
1 ogr2ogr -f "PostgreSQL" PG:"dbname=pgis_uzpd_user=uzpd18_d_host=geo102.fsv.
   cvut.cz" -t_srs 'EPSG:5514' "input.json" -nln uzpd18_d.table
```

Jelikož geometrie všech vrstev byla *bod*, pro kontrolu validity byla pro všechny vrstvy použita stejná funkce:

```
1 SELECT ogc_fid FROM nazev_tabulky WHERE NOT st_isvalid (geom)
```

## Dětská hřiště Praha

Příkaz pro smazání sloupečků, které pro další práci s daty nebyly nutné:

```
1 ALTER TABLE detska_hriste
2     DROP COLUMN url ,
3     DROP COLUMN name ,
4     DROP COLUMN perex ,
5     DROP COLUMN content ,
6     DROP COLUMN address ,
7     DROP COLUMN properties ,
8     DROP COLUMN image
```

Přejmenování sloupečku s geometrií:

```
1 ALTER TABLE detska_hriste
2     RENAME wkb_geometry TO geom
```

Výsledná tabulka má tyto sloupce:

detska_hriste			
ogc_fid	geom	id	district

## Lékárny a zdravotnická zařízení v Praze

Příkaz pro smazání sloupečků, které pro další práci s daty nebyly nutné:

```
1 ALTER TABLE zdrav_zarizeni
2     DROP COLUMN id ,
3     DROP COLUMN address ,
4     DROP COLUMN email ,
5     DROP COLUMN web ,
6     DROP COLUMN telephone ,
7     DROP COLUMN opening_hours
```

Přejmenování sloupečku s geometrií:

```
1 ALTER TABLE zdrav_zarizeni
2 RENAME wkb-geometry TO geom
```

Výsledná tabulka má tyto sloupce:

zdrav_zarizeni				
ogc_fid	geom	name	type	district

## Vstupy do metra

Jelikož stažená data byla již v S-JTSK, část kódu zajišťující transformaci při importu byla nahrazena za `-a_srs 'EPSG:5514'`. Příkaz pro smazání sloupečků, které pro další práci s daty nebyly nutné:

```
1 ALTER TABLE metro
2 DROP COLUMN objectid ,
3 DROP COLUMN vstupy_kod ,
4 DROP COLUMN vstupy_vest_kod ,
5 DROP COLUMN vstupy_vest_nazev ,
6 DROP COLUMN vstupy_vazba_bus ,
7 DROP COLUMN vstupy_vazba_csad ,
8 DROP COLUMN vstupy_vazba_kr ,
9 DROP COLUMN vstupy_vazba_pr ,
10 DROP COLUMN vstupy_vazba_privoz ,
11 DROP COLUMN vstupy_vazba_taxi ,
12 DROP COLUMN vstupy_vazba_tram ,
13 DROP COLUMN vstupy_vazba_vlak ,
14 DROP COLUMN zast_uzel_cislo ,
15 DROP COLUMN vstupy_popis ,
16 DROP COLUMN poskyt ,
17 DROP COLUMN vstupy_mimo_provoz
```

Přejmenování sloupečku s geometrií a názvu stanic:

```
1 ALTER TABLE metro
2 RENAME wkb-geometry TO geom ,
3 RENAME vstupy_uzel_nazev TO nazev ,
4 RENAME vstupy_linka TO linka
```

Výsledná tabulka má tyto sloupce:

metro			
ogc_fid	geom	nazev	linka

## 3.3 Geoportál Praha

Data získaná z tohoto portálu byla stažena ve formátu geoJSON v systému JTSK. Zpracování dat probíhalo obdobně jako u předchozího zdroje dat s výjimkou, že při importu nebylo nutné provádět transformaci. Obecný kód pro nahrání souboru do příslušného schématu databáze (bez transformace do S-JTSK):

```
1 ogr2ogr -f "PostgreSQL" PG:"dbname=pgis_uzpd_user=uzpd18_d_host=geo102.fsv.
   cvut.cz" -a_srs 'EPSG:5514' "input.json" -nln uzpd18_d.table
```

## Mapa košů na tříděný odpad

Příkaz pro smazání sloupečků, které pro další práci s daty nebyly nutné:

```
1 ALTER TABLE odpad
2     DROP COLUMN objectid ,
3     DROP COLUMN id ,
4     DROP COLUMN stationnumber ,
5     DROP COLUMN stationname ,
6     DROP COLUMN citydistrictruiancode
```

Přejmenování sloupečku s geometrií:

```
1 ALTER TABLE odpad
2     RENAME wkb_geometry TO geom
```

Na závěr byla provedena kontrola validity geometrie.

```
1 SELECT ogc_fid FROM nazev_tabulky WHERE NOT st_isvalid(geom)
```

Výsledná tabulka má tyto sloupce:

odpad			
ogc_fid	geom	citydistrict	pristup

## Záplavové území 2013

Příkaz pro smazání sloupečků, které pro další práci s daty nebyly nutné:

```
1 ALTER TABLE zaplava2013
2     DROP COLUMN objectid ,
3     DROP COLUMN nazev ,
4     DROP COLUMN typ
```

Přejmenování sloupečku s geometrií:

```
1 ALTER TABLE zaplava2013
2     RENAME wkb_geometry TO geom
```

Opět bylo nutné zkontrolovat validitu geometrie (polygonů). Níže uvedeným příkazem byly zjištěny chyby v datech:

```
1 SELECT id, geom, st_isvalidreason(geom) FROM uzpd18_d.zaplava2013 WHERE NOT
    st_isvalid(geom)
```

Ukázalo se, že nevalidní byly 4 polygony z celkových 86. Jako příčina těchto chyb se ukázala tzv. *Ring-self Intersection* polygonů. Nevalidní polygony byly opraveny vytvořením *bufferu* o velikosti 0.

```
1 UPDATE uzpd18_d.zaplava2013 SET geom = st_buffer(geom, 0.0) WHERE NOT
    st_isvalid(geom)
```

Výsledná tabulka má tyto sloupce:

zaplava2013			
ogc_fid	geom	shape_length	shape_area



## 4 SQL dotazy

1) Kolik adresních míst bylo zatopeno během povodní v roce 2013?

```
1 SELECT COUNT(a.kod)
2 FROM uzpd_d.adresni_mista AS a
3 JOIN (SELECT * FROM uzpd_d.zaplava2013) AS z
4 ON st_intersects(z.geom, a.geom);
```

1042

2) Jaká zdravotnická zařízení byla zasažena povodněmi v roce 2013? Vypište jejich název a správní obvod, ve kterém se nacházejí.

```
1 SELECT zz.name, zz.district
2 FROM uzpd_d.zdrav_zarizeni AS zz
3 JOIN uzpd_d.zaplava2013 AS za
4 ON st_intersects(zz.geom, za.geom);
```

Dr.Max LÉKÁRNA, praha-8

3) Jaké procento území Prahy zaujímá záplavové území z roku 2013?

```
1 SELECT ROUND(
2 (
3 SELECT sum(st_area(geom)) FROM uzpd18_d.zaplava2013
4 )::numeric / (
5 SELECT sum(st_area(geom)) FROM uzpd18_d.obvody
6 )::numeric, 2)*100 AS procento_zaplava;
```

5%

4) Kolik košů na tříděný odpad se nachází na území Prahy 1?

```
1 SELECT COUNT(o.ogc_fid)
2 FROM uzpd18_d.odpad AS o
3 JOIN (SELECT * FROM uzpd18_d.obvody WHERE nazev = 'Praha 1') AS p
4 ON st_intersects(p.geom, o.geom);
```

821

5) Kolik adresních míst leží ve vzdálenosti do 500 m od výlezu ze stanice metra Depo Hostivař?

```
1 SELECT count(distinct a.kod)
2 FROM uzpd18_d.adresni_mista AS a
3 JOIN uzpd18_d.metro AS m
4 ON st_dwithin(a.geom, m.geom, 500)
5 WHERE m.nazev = 'Depo Hostivař';
```

48

**6) Jaká stanice metra na Praze 4 má nejvíce vstupů?**

```
1 SELECT m.nazev FROM uzpd18_d.metro AS m
2 JOIN (SELECT * FROM uzpd18_d.obvody
3 WHERE nazev = 'Praha 4') AS p
4 ON st_intersects(p.geom, m.geom)
5 GROUP BY m.nazev
6 ORDER BY COUNT(m.ogc_fid) DESC
7 LIMIT 1;
```

Budějovická

**7) Která zdravotnická zařízení leží do 10 m od stanic metra A? V které městské části se tato zařízení nacházejí?**

```
1 SELECT z.name, z.district
2 FROM uzpd18_d.zdrav_zarizeni AS z
3 JOIN uzpd18_d.metro AS m
4 ON st_within(z.geom, m.geom, 10)
5 WHERE m.linka LIKE '%A%';
```

Dr.Max LÉKÁRNA, praha-1

BENU Lékárna OC Atrium Flora, praha-3

**8) Kolik dětských hřišť se nachází ve vzdálenosti do 2 kilometrů od nejbližší nemocnice?**

```
1 WITH
2   n AS (
3     SELECT * FROM uzpd18_d.zdrav_zarizeni AS z
4     WHERE type LIKE '%nemocnice%')
5 SELECT COUNT(DISTINCT(h.ogc_fid))
6 FROM uzpd18_d.detska_hriste AS h, n
7 WHERE n.geom && st_expand(h.geom, 2000);
```

61

**9) Kolik adresních míst má lékárnu a dětské hřiště do vzdálenosti 500 m?**

```
1 WITH
2   buff_h AS (
3     SELECT st_buffer(geom, 500) AS geom
4     FROM uzpd18_d.detska_hriste AS h),
5   buff_z AS (
6     SELECT st_buffer(geom, 500) AS geom
7     FROM uzpd18_d.zdrav_zarizeni AS z
8     WHERE type = 'Lékárna')
9 SELECT COUNT(DISTINCT(kod)), a.geom FROM uzpd18_d.adresni_mista AS a
10 INNER JOIN buff_h ON st_intersects(a.geom, buff_h.geom)
11 INNER JOIN buff_z ON st_intersects(a.geom, buff_z.geom);
```

25130

10) Který pražský správní obvod má nejlepší poměr počtu košů na tříděný odpad vzhledem ke své rozloze? Uveďte název obvodu, jeho rozlohu, počet košů a poměr těchto dvou hodnot.

```

1 WITH
2   oo AS (
3       SELECT ob.nazev, count(od.ogc_fid), ob.geom FROM obvody AS ob
4       JOIN uzpd18_d.odpad AS od
5       ON st_intersects(ob.geom, od.geom)
6       GROUP BY ob.nazev, ob.geom
7       ORDER BY COUNT(od.ogc_fid) DESC),
8   area AS (
9       SELECT obvody.nazev, st_area(obvody.geom), geom from obvody)
10 SELECT oo.nazev, ROUND(area.st_area), count, ROUND(area.st_area/count) AS
    ratio
11 FROM oo
12 JOIN area ON oo.nazev = area.nazev
13 GROUP BY oo.nazev, area.st_area, count, area.st_area/count
14 ORDER BY area.st_area/count DESC
15 LIMIT 1;

```

Praha 22, 33660132, 70, 480859

11) Na území kterého pražského správního obvodu se nachází největší množství dětských hřišť a kolik to je?

```

1 SELECT o.nazev, COUNT(distinct dh.ogc_fid)
2 FROM uzpd18_d.obvody AS o
3 JOIN uzpd18_d.detska_hriste AS dh
4 ON st_intersects(dh.geom, o.geom)
5 GROUP BY o.nazev
6 HAVING COUNT(distinct dh.ogc_fid) = (
7     SELECT COUNT(distinct dh.ogc_fid) FROM uzpd18_d.obvody AS o
8     JOIN uzpd18_d.detska_hriste AS dh
9     ON st_intersects(dh.geom, o.geom)
10    GROUP BY o.nazev
11    ORDER BY COUNT(distinct dh.ogc_fid) DESC
12    LIMIT 1);

```

Praha 3, 14

Praha 4, 14

12) Jaké je id nejbližšího dětského hřiště pro adresní místo s kódem 22560840? V jaké leží vzdálenosti?

```

1 SELECT dh.id, ROUND(st_distance(a.geom, dh.geom)) AS distance
2 FROM uzpd18_d.detska_hriste AS dh, uzpd18_d.adresni_mista AS a
3 WHERE a.kod = '22560840' AND st_distance(a.geom, dh.geom) = (
4     SELECT st_distance(a.geom, dh.geom)
5     FROM detska_hriste AS dh, uzpd18_d.adresni_mista AS a
6     WHERE a.kod = '22560840'
7     ORDER BY st_distance(a.geom, dh.geom)
8     LIMIT 1);

```

120, 506 m

## 5 Závěr

Na základě volně dostupných zdrojů dat byly navrženy a zpracovány tematické vrstvy. Nad těmito vrstvami byly následně aplikovány prostorové a atributové dotazy. Semestrální projekt nám přinesl další zkušenosti při zpracování prostorových dat a jejich uchování v databázích.

Během zpracování se vyskytlo několik problémů. Prvním problémem se ukázalo býti zvolené téma. Samo o sobě je velmi široké a při jeho volbě nás nenapadlo, že by v tom mohla být potíž. Pro naši analýzu bylo k dispozici ke stažení velké množství dat, z nichž jsme se rozhodly použít jen několik zdrojů a ty zpracovat podrobněji. Použití velkého množství dat (a že se nám jich líbilo hodně) by zapříčinilo, že výsledek naší práce by nebyl tak kvalitně zpracován, jak bychom si přály. Nakonec se ukázalo, že použité datové vrstvy neobsahovaly takové množství užitečných informací, a bylo nutné smazat většinu sloupečků. Výsledek projektu odpovídá zvolenému tématu jen částečně, jelikož se nám povedla pouze analýza bydlení v Praze na základě zvolených tematických vrstev, nikoli komplexní jak jsme původně zamýšlely.

Jako zádrhel při zpracování se ukázal import dat ve formátu geoJSON. Nikde jsme neobjevily přesný návod, jak importovat data tohoto typu do postGIS databáze přes příkazovou řádku, je-li součástí dávky i přihlašovací jméno a heslo do databáze. Správný zápis příkazu se nakonec povedlo nalézt.

## 6 Přílohy

- Příloha č. 1: Prezentace (prezentace.pdf)
- Příloha č. 2: SQL dávka (davka.sql)

## 7 Zdroje

1. *Otevřená data* [online] [cit. 28. 1. 2019].  
Dostupné z: <http://www.geoportalpraha.cz/>
2. *Datové sady – Národní katalog otevřených dat (NKOD)* [online] [cit. 28. 1. 2019].  
Dostupné z: <https://data.gov.cz/>
3. *Školení postGIS pro začátečníky* [online] [cit. 30. 1. 2019].  
Dostupné z: <http://training.gismentors.eu/>
4. *Školení postGIS pro pokročilé* [online] [cit. 30. 1. 2019].  
Dostupné z: <http://training.gismentors.eu/>
5. *155UZPD / Semestrální projekt* [online] [cit. 30. 1. 2019].  
Dostupné z: <http://geo.fsv.cvut.cz/>
6. *LaTeX/Source Code Listing* [online] [cit. 28. 1. 2019].  
Dostupné z: <https://en.wikibooks.org/>