ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta stavební
Katedra mapování a kartografie

DOKUMENTACE

Úvod do zpracování prostorových dat

skupina A:

Jan Synek
Hana Kadlecová
Lukáš Bocan
Vladimír Holubec

Geodézie a kartografie
obor: Geoinformatika

2010

Obsah

1 Úvod

Tato dokumentace slouží k seznámení s naším projektem, který vznikl v rámci předmětu Úvod do zpracování prostorových dat (UZPD). Tento předmět absolvujeme jako studenti 3. ročníku oboru Geoinformatika na Fakultě stavební ČVUT v Praze. UZPD volně navazuje na předmět Databázové systémy. Předmět je zaměřen především na zpracování geoprostorových dat. Cvičení jsou věnována práci s PostGIS.

1.1 Zadání projektu

1. Navrhnout několik tématických vrstev na základě dat OpenStreetMap (OSM)
2. Aplikovat testy datové integrity a odstranit nekonzistence v datech
3. Vytvořit tutoriál pro výuku PostGIS - sadu atributových a prostorových dotazů nad databází `pgis_osm`

1.2 OpenStreetMap

OpenStreetMap je projekt zaměřený na vytváření svobodných geografických dat. U většiny ostatních volně dostupných map je ale užívání technicky a právně omezeno. Proto vznikl tento projekt, aby umožnil lidem volně nakládat s geografickými daty, používat je neobvyklými způsoby a v neposlední řadě, aby byla data dostupná v aktualizované a platné podobě bez dalších nákladů a omezení[1].

1.2.1 Formát dat

Data se ukládají do centrální databáze jako primitiva a to:

- Uzly – body lokalizované souřadnicemi v daném referenčním systému.
- Cesty – posloupnost uzlů, reprezentující polylinii nebo v případě uzavření polylinie pak polygon.
- Relace – skupina uzlů, cest a dalších relací, které může být přiřazena daná vlastnost.
- Atributy – mohou být přiřazeny uzlům, cestám nebo relacím ve formě `<klíč>=<hodnota>`. Určují jaký objekt reálného světa reprezentují.

[1] `http://wiki.openstreetmap.org/wiki/Cz:Main_Page`

Vlastností těchto atributů je jejich rozšiřitelnost a modifikovatelnost.

1.2.2 Licence dat

Databáze OpenStreetMap Data OpenStreetMap jsou publikována pod licencí Creative Commons Attribution-Share Alike 2.0 licence.

Vstupní data 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 neblahý vliv na projekt, navíc vyžadují odstranění všech souvisejících změn provedených v souvislosti s kompromitujícími daty.

1.2.3 Zdroje dat

Od počátku projektu jsou data pořizována dobrovolníky, kteří systematicky mapují pomocí ručních GPS přijímačů. GPS data jsou poté zpracována na počítači a posléze nahrána do OpenStreetMap databanky. Mapovat lze při procházkách, na kole, v autě apod.

Nedávné zpřístupnění leteckých snímků a další dat z komerčních či veřejných zdrojů přispělo ke zrychlení a zpřesnění mapových podkladů, např. využití půdy[2].

[2] `http://cs.wikipedia.org/wiki/Openstreetmap`

2 Tvorba tématických vrstev

Po důkladné rekognoskaci databáze OpenStreetMap bylo vybráno 7 tématických vrstev. Do každé z nich byl přidán sloupec `way` zajišťující geometrii objektů a sloupec `osm_id`[3], ke kterému je přidán primární klíč. Dále pak byla přidělena práva na editaci i ostatním členům týmu. Všechny vrstvy jsou v referenčním systému Spherical Mercator. V následných podsekcích budou uvedeny příkazy pro tyto operace podle jednotlivých vrstev.

2.1 Pozemní komunikace

Pro liniovou vrstvu komunikací byly vybrány komunikace typu: dálnice, rychlostní silnice a silnice I., II. a III. třídy. Byly přidán sloupec s jménem komunikace (`name`) a typem komunikace (`highway`).

Příkaz pro tvorbu vrstvy:

```
CREATE TABLE a10.komunikace AS SELECT osm_id,way,name,highway
FROM czech_line WHERE highway IN
('motorway','trunk','primary','secondary','tertiary');
```

Příkaz pro přidání primárního klíče:

```
ALTER TABLE a10.komunikace ADD PRIMARY KEY (osm_id);
```

2.2 Občanské vyžití

Vrstva občanského vyžití je bodová vrstva všech hospod, restaurací, nevěstinců, fastfoodů, kaváren, barů a restaurací se zahrádkou na území ČR. Vedle identifikátoru a sloupce s geometrií byl přidán sloupec se jménem podniku (`name`) a jeho typem (`amenity`).

Příkaz pro tvorbu vrstvy:

```
CREATE TABLE a10.obcerstveni AS SELECT osm_id,name,amenity,way
FROM czech_point WHERE amenity IN
('pub','restaurant','brothel','fast_food','cafe','bar','biergarten')
AND name IS NOT NULL;
```

[3]jednoznačný identifikátor objektů v rámci celé databáze OSM

Příkaz pro přidání primárního klíče:

```
ALTER TABLE a10.obcerstveni ADD PRIMARY KEY (osm_id);
```

2.3 Lesy

Polygonová vrstva obsahující sloupec s identifikátorem a geometrií.

Příkaz pro tvorbu vrstvy:

```
CREATE TABLE a10.lesy AS SELECT osm_id,name
FROM czech_polygon WHERE landuse = 'forest';
```

Příkaz pro přidání primárního klíče:

```
ALTER TABLE a10.lesy ADD PRIMARY KEY (osm_id);
```

2.4 Železnice

Vrstva železnic je liniová a obsahuje sloupec s identifikátorem a geometrií.

Příkaz pro tvorbu vrstvy

```
CREATE TABLE a10.zeleznice AS SELECT osm_id,way
FROM czech_line WHERE railway = 'rail';
```

Příkaz pro přidání primárního klíče:

```
ALTER TABLE a10.zeleznice ADD PRIMARY KEY (osm_id);
```

2.5 Památky

Bodová vrstva kulturních památek zahrnuje všechny hrady(zámky), monumenty a zříceniny. Spolu se jménem (`name`) byl přidán i druh památky (`historic`).

Příkaz pro tvorbu vrstvy:

```
CREATE TABLE a10.pamatky AS SELECT osm_id,name,historic,way
FROM czech_point WHERE historic IN ('castle','monument','ruins')
AND name IS NOT NULL;
```

Příkaz pro přidání primárního klíče:

```
ALTER TABLE a10.pamatky ADD PRIMARY KEY (osm_id);
```

2.6 Obce

Tématická vrstva obce je bodovou vrstvou, která zahrnuje obce kategorie vesnice,město a velkoměsto. Obsahuje sloupce id(`osm_id`), jméno(`name`), kategorie obce(`place`) a geometrie(`name`).

Příkaz pro tvorbu vrstvy:

```
CREATE TABLE a10.obce AS SELECT osm_id,name,place,way
FROM czech_point WHERE place IN
('village','town','city') AND name IS NOT NULL;
```

Příkaz pro přidání primárního klíče:

```
ALTER TABLE a10.obce ADD PRIMARY KEY (osm_id);
```

2.7 Vodní plochy

Poslední tématickou vrstvou jsou vodní plochy na uzemí ČR s definovanou geometrií a identifikátorem objektů.

Příkaz pro tvorbu vrstvy:

```
CREATE TABLE a10.voda AS SELECT osm_id,way
FROM czech_polygon WHERE landuse = 'reservoir';
```

Příkaz pro přidání primárního klíče:

```
ALTER TABLE a10.voda ADD PRIMARY KEY (osm_id);
```

2.8 Přidělení práv

Příkaz pro přídaní přístupových práv:

```
GRANT SELECT ON a10.obcerstveni,a10.komunikace,a10.lesy,a10.pamatky,
a10.zeleznice,a10.obce,a10.voda TO holubvl3,kadleha2,bocanluk;
```

3 Modifikace dat

3.1 Kontrola a oprava dat

3.1.1 Populate_Geometry_Columns

Příkazem `Populate_Geometry_Columns` bylo zkontrolováno, jestli existuje sloupec geometrie a má vhodná prostorová omezení. Příklad je uveden pouze pro vrstvu komunikací. Pro ostatní vrstvy je příkaz obdobný. Všechny vrstvy byly otestovány se stejným výsledkem.

Příkaz pro kontrolu geometrie:

```
pgis_osm=> SELECT Populate_Geometry_Columns('a10.komunikace'::regclass);
 populate_geometry_columns
---------------------------
                         1
(1 row)
```

3.1.2 ST_IsValid

Tato funkce kontroluje validitu vrstvy a vrací TRUE, jestliže je geometrie správně vytvořená. Když objekt není validní, PostgreSQL NOTICE vypíše důvod.[4].

Příkaz pro kontrolu validity pro liniovou vrstvu – komunikace:

[4]U bodových vrstev kontrola validity neuvažována

```
pgis_osm=> SELECT osm_id FROM a10.komunikace WHERE not st_isvalid(way);
 osm_id
--------
(0 rows)
```

Z výsledku příkazu vyplývá, že všechny komunikace jsou validní.

Příkaz pro kontrolu validity a jeho výstup pro polygonovou vrstvu – voda:

```
pgis_osm=> SELECT osm_id FROM a10.voda WHERE not st_isvalid(way);
NOTICE:  Self-intersection at or near point 1.49105e+06 6.55772e+06
NOTICE:  Self-intersection at or near point 1.60927e+06 6.27764e+06
NOTICE:  Self-intersection at or near point 1.6093e+06 6.27764e+06
NOTICE:  Self-intersection at or near point 1.63337e+06 6.3653e+06
NOTICE:  Self-intersection at or near point 1.95017e+06 6.36218e+06
  osm_id
----------
 51011010
 50761876
 50761877
 50954304
 51011757
(5 rows)
```

Z výsledku příkazu vyplývá, že ve vrstvě voda je pět nevalidních polygonů. Snažili jsme se tedy zjistit, zda by se polygony daly opravit standardní metodou buffer s nulovou vzdáleností.

```
pgis_osm=> SELECT osm_id,st_isvalid(st_buffer(way,0))
pgis_osm-> FROM a10.voda WHERE not st_isvalid(way);
NOTICE:  Self-intersection at or near point 1.49105e+06 6.55772e+06
NOTICE:  Self-intersection at or near point 1.60927e+06 6.27764e+06
NOTICE:  Self-intersection at or near point 1.6093e+06 6.27764e+06
NOTICE:  Self-intersection at or near point 1.63337e+06 6.3653e+06
NOTICE:  Self-intersection at or near point 1.95017e+06 6.36218e+06
  osm_id  | st_isvalid
----------+------------
 51011010 | t
```

```
 50761876 | t
 50761877 | t
 50954304 | t
 51011757 | t
(5 rows)
```

Zdá se, že všechny nevalidní polygony lze opravit pomocí funkce ST_Buffer. Oprava tedy bude příkazem:

```
pgis_osm=> UPDATE a10.voda SET way = st_buffer(way,0)
pgis_osm-> WHERE osm_id IN (51011010,50761876, 50761877, 50954304, 51011757);
ERROR:  new row for relation "voda" violates check constraint "enforce_geotype_way"
```

Experimentálně bylo zjištěno, že je chybný polygon s `osm_id` = 51011010. Proto jsme tento polygon do opravy nezahrnuli. A použili jsme tedy příkaz:

```
pgis_osm=> UPDATE a10.voda SET way = st_buffer(way,0)
pgis_osm-> WHERE osm_id IN (50761876, 50761877, 50954304, 51011757);
UPDATE 4
```

Nevalidní polygony byly kontrolovány před a po opravě v programu QGIS, jestli se nezměnil jejich tvar, ostatní polygony byly v pořádku, a proto jsme je mohli dále použít. Dále jsme zkoušeli neopravený polygon opravit jinými funkcemi např.:

```
ST_BuildArea(ST_Boundary(geometry))
```

ST_BuildArea vytváří plošnou geometrii utvořenou základní linií této geometrie, ST_Boundary vrací uzavření kombinatorické hranice této geometrie.

Pomocí těchto funkcí se nám podařilo opravit zbylý polygon, ale při zobrazení v programu QGIS se tento polygon nezobrazil. Při dotazu na vypsání geometrie pro tento polygon se nezobrazily žadné hodnoty. Proto jsme usoudili, že touto „opravou“ byla vymazána geometrie pro tento objekt. Polygon s `osm_id` = 51011010 jsme se rozhodli vymazat, abychom s ním neměli problémy při tvorbě dotazů.

Příkaz pro vymazání záznamu:

```
pgis_osm=> DELETE FROM a10.voda where osm_id = 51011010;
DELETE 1
```

Dále u vrstvy a10.lesy jsem zjistili 289 nevalidních polygonů, příkaz je obdobný jako u vrstvy voda.

```
pgis_osm=> SELECT osm_id FROM a10.lesy WHERE not st_isvalid(way);
```

Pro opravu této vrstvy jsme opět zkoušeli standardní metody, ale protože jsme zjistili, že standardní metody zde nepomůžou a že tomuto problému se již věnuje jiná skupina, pouze jsme nevalidní polygony vymazali, abychom neměli problémy při tvorbě dotazů.

Příkaz pro vymazaní nevalidních polygonů :

```
pgis_osm=> DELETE FROM a10.voda WHERE NOT st_isvalid(way);
```

3.2 Tvorba indexu

Pro zrychlení prostorových dotazů byl použit index nad sloupcem s geometrií `way`. Byla použita metoda GiST, která je nejvhodnější pro sloupce typu geometrie. Po vytvoření se většina dotazů znatelně zrychlila zvlášť u polygonových vrstev. V příkladě je uvedena pouze tvorba indexu pro vrstvu komunikací, v ostatních případech je příkaz obdobný.

Příkaz pro vytvoření indexu:

```
CREATE INDEX komunikace_gist ON a10.komunikace USING gist (way);
```

4 Popis funkcí

V následující části jsou uvedeny popisy jednotlivých funkcí, které jsou použity v dotazech. Jako zdroj byly použity webové stránky projektu PostGIS `http://postgis.refractions.net`.

4.1 ST_Intersects

Název: `ST_Intersects` (průsečík dvou geometrický objektů)

Vstupní parametry: `(tabulka1.geometry_atribut, tabulka2.geometry_atribut)`

Popis: Jedná se o funkci typu bool, vrací TRUE nebo FALSE, na základě toho, zda se 2 vstupní tabulky protínají (tzn. překrývají se nebo se dotýkají).

Příklad: `ST_Intersects(s.the_geom, z.the_geom)` — místa kde se protínají s a z

4.2 ST_Touches

Název: `ST_Touches` (dotyk dvou geometrických objektů)

Vstupní parametry: `(tabulka1.geometry_atribut, tabulka2.geometry_atribut)`

Popis: Jedná se o funkci typu bool, která vrací TRUE, pokud se dva objekty dotýkají nejméně v jednom bodě. Platí pro vztahy: linie – linie, plocha – line, plocha – plocha, bod – linie, bod – plocha. Neplatí pro dotyk bod – bod.

Příklad: `ST_Touches('LINESTRING(0 0,1 1,0 2)'::geometry,'POINT(0 2)'::geometry)` — místa dotyku linestringu se zadanými parametry a zadaného bodu

4.3 ST_Area

Název: `ST_Area` (plocha)

Vstupní parametry: `(tabulka.geometry_atribut)`
`(tabulka1.geography_atribut, volitelný bool atribut)`

Popis: Příkaz, který vrací hodnotu typu float výměry zadaného geometrického/geografického objektu (polygon, multipolygon). Pro vstup geometry je plocha navrácena v jednotkách SRID, což jsou systémy jednoznačně definované v PostGis. V případě použití vstupního parametru typu geography, je výsledná plocha navrácena na v m^2. Parametr False u geografických dat, tudíž výpočet výměry vztažené na kouli, má výhodu, že ušetří výpočetní výkon, za cenu nižší přesnosti.

Příklad: `ST_Area(lesy.the_geom)/1e6` — počítá plochu lesů v km^2

4.4 ST_Distance

Název: `ST_Distance` (vzdálenost dvou objektů)

Vstupní parametry: `(tabulka1.geometry_atribut, tabulka2.geometry_atribut)`
`(tabulka1.geography_atribut, tabulka2.geography_atribut, volitelný bool atribut)`

Popis: Funkce vrací hodnotu typu float pro nejmenší kartézskou vzdálenost ve 2D mezi 2 geometrickými elementy v jednotkách SRID. Pro vstupní typ geography vrací v metrech nejkratší vzdálenost na WGS-84 mezi 2 geografikými objetky. Při použití parametru FALSE je tato vzdálenost počítána na kouli.

Příklad: `ST_Distance(o.the_geom, p.the_geom)` — vrací vzdálenost mezi o a p

4.5 ST_Perimeter

Název: `ST_Perimeter` (obvod)

Vstupní parametry: `(tabulka1.geometry_atribut)`

Popis: Vstupem může být uzavřený objekt (ST_Polygon/ST_Multipolygon). Návratová hodnota obvodu je v jednotkách SRID a je typu double precision.

Příklad: `SELECT SUM(ST_Perimeter(waterbody)) FROM waterbodies` — funkce perimeter vrací obvody jednotlivých vodních objektů

4.6 ST_DWithin

Název: `ST_DWithin` (vzdálenost do)

Vstupní parametry: `(tabulka1.geometry_atribut, tabulka2.geometry_atribut, vzdálenost)`

Popis: Vrací TRUE nebo FALSE podle toho, zda jsou vstupní geometrie v rámci stanovené vzdálenosti. Obě vstupní geometrii musí být ze stejného souřadnicového systému. V případě že místo typu geometry je typ geography vstupuje do hry ještě volba referenčního tělesa. Implicitně je to elipsoid (při použití FALSE bude měření vztaženo vůči referenční kouli).

Příklad: `ST_DWithin(s.the_geom, h.the_geom, 3000)` — vrací, zda odpovídá, že jsou vstupní části geometrie do 3 km od sebe

4.7 ST_Buffer

Název: `ST_Buffer` (obal)

Vstupní parametry: `(tabulka1.geometry_atribut, float poloměr)`
`(tabulka1.geography_atribut, float poloměr)`

Popis: Vstupem mohou být body (points, multipoints), polygony (polygons, multipolygons) a linestrings (včetně multilinestrings). Vrací typ geometry/geography, který představuje obal zahrnující všechny body od počátku obalu až po jeho mez danou hodnotou poloměru.

Příklad: `ST_Buffer(obce.the_geom, 1e3)` — vytvoří okolo každé obce kilometrovou obalovou zónu

4.8 ST_Disjoint

Název: `ST_Disjoint` (nespojitost)

Vstupní parametry: `(tabulka1.geometry_atribut, tabulka2.geometry_atribut)`

Popis: Vstupem jsou dva prvky typu geometry a funkce řeší, zda mají průsečík či nikoli. Návratem je hodnota bool TRUE, pokud vstupní geometrie nemají průsečík nebo FALSE, pokud mají průsečík. Pokud druhá vstupní geometrie nemá stejný souřadnicový systém jako ta první. Bude její souřadnicový systém překonvertován.

Příklad: `ST_Disjoint(a.the_geom, b.the_geom)` — řeší zda se a a b protínají

4.9 ST_Crosses

Název: `ST_Crosses` (průsečík — společný bod)

Vstupní parametry: `(tabulka1.geometry_atribut, tabulka2.geometry_atribut)`

Popis: Návratová hodnota je typu bool — TRUE, pokud vstupní geometrie mají nějaké společné body (né však všechny). V ostatních případech vrací FALSE. Průsečíky vnitřních ploch nesmí být prázdné a musí mít dimenzi maximálně na úrovni vstupních geometrií. Platí pro vztahy: bod – linie, bod – plocha, linie – plocha, linie – linie.

Příklad: `ST_Crosses(roads.the_geom, highways.the_geom)` — zkoumá zda se silnice a dálnice protínají.

4.10 ST_Overlaps

Název: `ST_Overlaps` (překrytí)

Vstupní parametry: `(tabulka1.geometry_atribut, tabulka2.geometry_atribut)`

Popis: Vrací hodnotu bool TRUE, pokud protnutí dvou geometrií vytvoří geometrii stejné dimenze, nicméně se nesmí vrátit ani jedna ze vstupních geometrií (to zjišťuje funkce ST_Union). Tzn. při průsečíku dvou ploch je výsledkem plocha, která je pro obě vstupní plochy stejná. Matematicky jde o plochu průniku množin. V ostatních případech je návratová hodnota TRUE.

Příklad: `ST_Overlaps(a.the_geom, b.the_geom)` — řeší zda se a a b překrývají

4.11 ST_Within

Název: `ST_Within` (obsah do)

Vstupní parametry: `(tabulka1.geometry_atribut, tabulka2.geometry_atribut)`

Popis: Vrací hodnotu TRUE nebo FALSE, pokud geometrie 1 je celá uvnitř geometrie 2. Obě vstupní geometrie musí mít společný systém.

Příklad: `ST_Within(a.the_geom, b.the_geom)` — řeší, zda geometrie a je uvnitř b

4.12 ST_Intersection

Název: `ST_Intersection` (protnutí — oblast průniku 2 množin)

Vstupní parametry: `(tabulka1.geometry_atribut, tabulka2.geometry_atribut)`
`(tabulka1.geography_atribut, tabulka2.geography_atribut)`

Popis: Vrací geometrický objekt, který vznikne průsečíkem 2 vstupních geometrií. V případě, že se 2 geometrie neprotínají, výsledkem je prázdná množina. V případě, že je vstupem typ geography, je vstup transformován na geometry a následně transformován do geography (elipsoid WGS-84). Vstupem může být linie, uzavřená linie (plocha) a bod.

Příklad: `ST_Intersection('POINT(0 0)'::geometry,'LINESTRING(0 0,0 2)'::geometry)` — průsečíkem zadaného bodu a linie je bod o souřadnicích POINT(0 0)

4.13 ST_Length

Název: `ST_Length` (délka)

Vstupní parametry: `(tabulka1.geometry_atribut)`
`(tabulka1.geography_atribut, volitelný bool atribut)`

Popis: Vrací 2D vzdálenost geometrie typu linestring nebo multilinestring. Jednotky výsledné délky jsou vraceny v jednotkách použitého systému SRID, případně v metrech při vstupu typu geography.

Příklad: `ST_Length(a.the_geom)` — počítá délku geometrií a

5 Dotazy

V následující části jsou uvedeny dotazy, jejich výsledky a časová náročnost v programu psql. Časová náročnost je pouze informativní. Uvádíme zde pouze jednu z mnoha variant řešení pro každý dotaz.

5.1 Atributové dotazy

Vypište jména obcí, které mají v názvu Březno.

```
pgis_osm=> SELECT name FROM a10.obce WHERE name like '%Březno%';
     name
---------------
 Březno
 Malé Březno
 Březno
 Krásné Březno
 Velké Březno
 Malé Březno
 Březno
(7 rows)
Time: 46.441 ms
```

Vypište hrady a zámky, které mají v názvu "štejn".

```
pgis_osm=> SELECT name FROM a10.pamatky WHERE historic = 'castle'
pgis_osm-> AND name LIKE '%štejn%';
      name
-----------------
 Hrad Vildštejn
 Gutštejn
 Hauenštejn
```

```
 Pernštejn
 Branná Kolštejn
(5 rows)
Time: 10.939 ms
```

Vypište jména (name) a druh (amenity) restaurací se zahrádkou (biergarten) a seřaďte podle abecedy.

```
pgis_osm=> SELECT name, amenity FROM a10.obcerstveni
pgis_osm-> WHERE amenity = 'biergarten' ORDER BY name;
              name               |  amenity
---------------------------------+------------
 Forman - minipivovar Velichov | biergarten
 Hospůdka v Trněném újezdu     | biergarten
 Hříbek                        | biergarten
 Kukačka                       | biergarten
 Mexiko                        | biergarten
 Palma                         | biergarten
 Petrovický mlýn               | biergarten
 Pod Skalou                    | biergarten
 Sauna                         | biergarten
 Sobacovskej rybnik            | biergarten
 Tančírna                      | biergarten
 U Bobra                       | biergarten
 U Kristiána                   | biergarten
 U lávky                       | biergarten
 U Pohanků                     | biergarten
 U Starého pivovaru            | biergarten
 U trabanta                    | biergarten
 U Váchy                       | biergarten
(18 rows)

Time: 106.368 ms
```

Kolik je typů památek?

```
pgis_osm=> SELECT distinct historic FROM a10.pamatky;
 historic
----------
 castle
 monument
 ruins
(3 rows)

Time: 5.021 ms
```

Kolik je hradů v památkách?

```
pgis_osm=> SELECT COUNT(*) FROM a10.pamatky WHERE historic = 'castle';
 count
-------
   133
(1 row)

Time: 1.851 ms
```

Vypište všechny pizzerie (v názvu mají řetězec 'pizz').

```
pgis_osm=> SELECT name FROM a10.obcerstveni WHERE amenity='restaurant'
pgis_osm-> AND name LIKE '%pizz%';
         name
-----------------------
 pizzeria Kamenka
 Piccolo mondo pizza
 pizzeria Grado
 pizzeria Metamorfozzi
 pizzerie
 pizzerie Nico
 Chacharova pizza
 Mini pizzerie
(8 rows)

Time: 38.241 ms
```

5.2 Prostorové dotazy

Kolik km komunikací leží v lese?

```
pgis_osm=> SELECT SUM(st_length(st_intersection(lesy.way,kom.way))/1e3)
pgis_osm-> FROM a10.lesy AS lesy JOIN a10.komunikace AS kom
pgis_osm-> ON st_intersects(lesy.way, kom.way);
       sum
------------------
 5794.67539645508
(1 row)

Time: 304039.063 ms
```

Kolik památek leží do 5 km od silnice a současně do 5 km od železnice?

```
pgis_osm=> SELECT COUNT(distinct pamatky.name) AS pocet_pamatek
pgis_osm-> FROM a10.pamatky AS pamatky  JOIN a10.komunikace AS kom
pgis_osm-> ON st_dwithin(pamatky.way, kom.way, 5000) JOIN a10.zeleznice AS zelez
pgis_osm-> ON st_dwithin(pamatky.way, zelez.way, 5000);
 pocet_pamatek
---------------
           192
(1 row)

Time: 44286.695 ms
```

Jaká je celková výměra vodních ploch v km^2?

```
pgis_osm=> SELECT ROUND(SUM(st_area(way)/1e6)) AS area_km2 FROM a10.voda;
 area_km2
----------
     1198
(1 row)

Time: 584.367 ms
```

Kolik vodních ploch leží dál než 5km od obcí?

```
pgis_osm=> SELECT COUNT(voda.osm_id) FROM a10.voda AS voda
pgis_osm-> LEFT JOIN a10.obce AS obce ON st_dwithin(voda.way, obce.way, 5000)
pgis_osm-> WHERE obce.osm_id is null;
NOTICE:  LWGEOM_gist_joinsel called with incorrect join type
NOTICE:  LWGEOM_gist_joinsel called with incorrect join type
 count
-------
  2974
(1 row)

Time: 10166.215 ms
```

Kolik památek je v lese?

```
pgis_osm=> SELECT COUNT(*) AS pocet_pamatek FROM a10.pamatky AS p
pgis_osm-> JOIN a10.lesy AS l ON st_within(p.way, l.way);
 pocet_pamatek
---------------
            67
(1 row)

Time: 846.026 ms
```

Kolik občerstvení je u silnice (tj. do 100 m) ?

```
pgis_osm=> SELECT COUNT(distinct ob.name) AS obcerstveni_u_silnice
pgis_osm-> FROM a10.obcerstveni AS ob
pgis_osm-> JOIN a10.komunikace AS kom ON st_dwithin(ob.way, kom.way, 100);
 obcerstveni_u_silnice
-----------------------
                   980
(1 row)

Time: 561.105 ms
```

Na kolika místech se kříží železnice a silnice ?

```
pgis_osm=> SELECT COUNT(*) AS pocet_krizeni FROM a10.komunikace AS kom
pgis_osm-> JOIN a10.zeleznice AS zelez ON st_crosses(kom.way, zelez.way);
 pocet_krizeni
---------------
          4267
(1 row)

Time: 79561.890 ms
```

Jaká je výměra lesů, které leží do 10 km od Prahy, Brna a Ostravy?

```
pgis_osm=> CREATE VIEW a10.obce_buffer AS SELECT st_buffer(way,1e4) AS way
pgis_osm-> FROM a10.obce
pgis_osm-> WHERE name = 'Praha' or name = 'Brno' or name = 'Ostrava';
CREATE VIEW
Time: 591.859 ms
pgis_osm=> SELECT SUM(st_area(st_intersection(obce_buff.way,lesy.way))/1e6)
pgis_osm-> FROM a10.lesy AS lesy JOIN a10.obce_buffer AS obce_buff
pgis_osm-> ON st_intersects(obce_buff.way, lesy.way);
       sum
------------------
 99.8789516638121
(1 row)

Time: 3625.312 ms
```

Kolik vodních ploch a s jakou výměrou leží zcela v lese nebo se lesa dotýkají?

```
pgis_osm=> SELECT COUNT(voda.osm_id), SUM(st_area(voda.way)/1e6)
pgis_osm-> FROM a10.voda AS voda JOIN a10.lesy AS lesy
pgis_osm-> ON st_touches(voda.way,lesy.way) or st_within(voda.way, lesy.way);
 count |       sum
-------+------------------
  4632 | 59.3368699422716
```

```
(1 row)

Time: 960141.882 ms
```

Kolik % území, které je do 20 km od Ostravy, zaujímá les?

```
pgis_osm=> CREATE VIEW a10.ostr_buff AS SELECT st_buffer(way,2e4) AS way
pgis_osm-> FROM a10.obce WHERE name = 'Ostrava';
CREATE VIEW
Time: 9.022 ms
pgis_osm=> SELECT ROUND((lesy_area/ostr_area)*100)
pgis_osm-> FROM (SELECT SUM(st_area(st_intersection(ostrava.way, lesy.way)))
pgis_osm-> AS lesy_area, st_area(ostrava.way) AS ostr_area
pgis_osm-> FROM a10.ostr_buff AS ostrava
pgis_osm-> JOIN a10.lesy AS lesy
pgis_osm-> ON st_intersects(ostrava.way,lesy.way)
pgis_osm-> GROUP BY ostr_area) AS lesy_ostr;
 round
-------
    15
(1 row)

Time: 3370.085 ms
```

Jaký je poměr celkové délky komunikace/železnice?

```
pgis_osm=> SELECT (SELECT (SUM(st_length(kom.way))) FROM a10.komunikace AS kom)/
pgis_osm-> (SELECT (SUM(st_length(zelez.way))) FROM a10.zeleznice AS zelez);
 ?column?
------------------
 5.75873614520475
(1 row)

Time: 190.759 ms
```

Kolik km dálnic je v ČR?

```
pgis_osm=> SELECT ROUND(SUM(st_length(kom.way)/1e3)) AS delka_dalnic
pgis_osm-> FROM a10.komunikace AS kom WHERE highway = 'motorway';
 delka_dalnic
--------------
         3303
(1 row)

Time: 87.929 ms
```

Vypište 3 největší úseky železnice, které jsou v lese.
Uveďte id železnice, id lesa a délku.

```
pgis_osm=> SELECT z.osm_id AS zeleznice, l.osm_id AS les,
pgis_osm-> st_length(st_intersection(z.way, l.way))
pgis_osm-> FROM a10.zeleznice AS z join a10.lesy AS l
pgis_osm-> ON st_crosses(z.way, l.way)
pgis_osm-> ORDER BY st_length DESC LIMIT 3;
  osm_id  |  osm_id  |    st_length
----------+----------+-----------------
 25535786 |   -24325 | 9200.37950639252
 23308335 | 30939124 | 8546.75477499056
 14392929 |   -23849 | 6225.99442174813
(3 rows)

Time: 94892.877 ms
```

Vyberte nejbližší nevěstinec od hradu Kozel v km.

```
pgis_osm=> SELECT o.name, st_distance(p.way,o.way)/1e3 AS v
pgis_osm-> FROM a10.pamatky AS p JOIN a10.obcerstveni AS o
pgis_osm-> ON o.amenity = 'brothel' AND p.name = 'Kozel'
pgis_osm-> ORDER BY v DESC LIMIT 1;
    name    |        v
------------+-----------------
 Big Sister | 159.445678593174
```

```
(1 row)

Time: 84.747 ms
```

Vypište počet občerstvení ležící u lesa (do 100 m).

```
pgis_osm=> SELECT COUNT(DISTINCT(o.name)) FROM a10.obcerstveni AS o
pgis_osm-> JOIN a10.lesy AS l ON st_dwithin(o.way,l.way,100);
 count
-------
   147
(1 row)

Time: 822.976 ms
```

Kolik % plochy ČR zabírají lesy, když plocha republiky je 78866 km^2?

```
pgis_osm=> SELECT ((SELECT SUM(st_area(lesy.way))/1e6 FROM a10.lesy)/78866);
 ?column?
-------------------
 0.616849480161295
(1 row)

Time: 378.996 ms
```

Jaká je nejmenší vodní plocha a její výměra (její osm_id a výměra v m^2)?

```
pgis_osm=> SELECT osm_id, st_area(way) AS area
pgis_osm-> FROM a10.voda ORDER BY st_area(way) ASC LIMIT 1;
  osm_id  |       area
----------+------------------
 40428806 | 22.8508501052856
(1 row)

Time: 639.126 ms
```

Obvod nejmenšího lesa?

```
pgis_osm=> SELECT osm_id, st_perimeter(lesy.way) AS obvod
pgis_osm-> FROM a10.lesy ORDER BY obvod ASC LIMIT 1;
  osm_id  |      obvod
----------+------------------
 50001538 | 43.2777075280125
(1 row)

Time: 794.756 ms
```

Kolik obcí leží v lese?

```
pgis_osm=> SELECT COUNT(*) FROM a10.obce AS o
pgis_osm-> JOIN a10.lesy AS l ON st_contains(l.way,o.way);
 count
-------
    99
(1 row)

Time: 6632.177 ms
```

Najděte nejbližší obec u Prahy a její vzdálenost.

```
pgis_osm=> SELECT DISTINCT ON(o1.name) o1.name, o2.name, st_distance(o1.way,o2.way)
pgis_osm-> FROM a10.obce AS o1
pgis_osm-> JOIN a10.obce AS o2 ON(o1.name <> o2.name and o1.name ='Praha')
pgis_osm-> ORDER BY o1.name, st_distance ASC;
 name  |    name    |   st_distance
-------+------------+-----------------
 Praha | Horoměřice | 12541.8675988512
(1 row)

Time: 302.795 ms
```

V okruhu 5 km od obcí najděte pro každou obec nejbližší památku. Vypište prvních pět obcí, jméno památky a jejich vzdálenost.

```
pgis_osm=> SELECT DISTINCT ON(o1.name) o1.name , o2.name ,
pgis_osm-> st_distance(o1.way,o2.way)
pgis_osm-> FROM a10.obce AS o1 JOIN  a10.pamatky AS o2
pgis_osm-> ON st_dwithin(o1.way, o2.way, 5000)
pgis_osm-> ORDER BY o1.name, st_distance(o1.way, o2.way) LIMIT 5;
        name         |         name          |   st_distance
---------------------+-----------------------+-----------------
 Abertamy            | Modesgrund            | 1830.71988634395
 Adamov              | zřícenina hradu Ronov | 3538.92993308341
 Babčice             | Oblajovice            | 4647.98326916952
 Babice              | Morový sloup          | 4506.34087268406
 Babice nad Svitavou | zřícenina hradu Ronov | 3523.61615625423
(5 rows)

Time: 54.580 ms
```

Zde uvádíme příklad dotazu, který nemohl být realizován z důvodu nedostatečné kapacity zařízení.

```
pgis_osm=> SELECT COUNT(DISTINCT l.osm_id) FROM a10.voda AS l
pgis_osm-> JOIN a10.zeleznice AS z ON st_disjoint(z.way, l.way);
ERROR:  could not write block 413909 of temporary file: No space left ON device
HINT:  Perhaps out of disk space?
```

6 Závěr

Cílem této dokumentace je přiblížit Vám náš projekt, který vznikl v rámci předmětu Úvod do zpracování prostorových dat. Tato dokumentace by měla sloužit jako pomůcka/návod pro výuku PostGIS. Výsledkem naší práce je 7 tématických vrstev (3 bodové, 2 liniové a 2 polygonové), tedy 7 tabulek, které jsou ve schématu a10 v databázi `pgis_osm` a které obsahují pouze validní data, dále pak soubor atributových a prostorových dotazů nad těmito tabulkami. Hlavním cílem tohoto projektu bylo vyzkoušet si práci s PostGIS. Také jsme používali program QGIS, který jsme využili pro vizualizaci dat a který nám také pomohl při vyhledávání nevalidních objektů. Snažili jsme se odstranit nevalidní data, v případě že se nám to nepodařilo, nevalidní objekt jsme z tabulky vymazali. Naše snaha při tvorbě dotazů byla použít různé funkce, které PostGIS nabízí. Tento projekt byl pro nás zajímavý a využili jsme zde své znalosti z předmětu Databázové systémy, který jsme již absolvovali.

6.1 Použitý software

- MS Windows
- GNU/Linux
- PostgreSQL
- PostGIS
- QGIS
- LaTeX 2_ε

Reference

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

[2] PostGIS: Documentation:
`http://postgis.refractions.net/documentation/`

[3] Quantum GIS project:
`http://www.qgis.org/`

[4] PostGIS GeoWikiCZ:
http://gama.fsv.cvut.cz/wiki/index.php/PostGIS

[5] OpenStreetMap:
http://www.openstreetmap.org/

[6] PostgreSQL: Documentation:
http://www.postgresql.org/docs/

Rádi bychom poděkovali všem, kteří nám pomohli při vytváření našeho projektu.