ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta stavební
Katedra mapování a kartografie

DOKUMENTACE

Úvod do zpracování prostorových dat

SKUPINA D:

Štěpán Kaiser
David Hanousek

2013

OBSAH

1 ÚVOD

1.1 Zadání

- Navrhněte a vytvořte tematické vrstvy (např. vodní toky, vodní plochy, lesy, silnice, železnice) 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**.

1.2 Výběr tématu

Cílem tohoto projektu je vytvořit zjednodušenou databázi pro lyžaře, která bude obsahovat základní informace o sjezdovkách na území České Republiky, které budou získány z rastru DMT z datasetu FreeGeoDataCZ . Dále v této databázi bude možné nalézt informace o ubytovacích zařízeních, možnosti vlakové dopravy za použití nadstavby pgRouting a pár dalších údajů, které budou podrobněji popsány níže.

2 Použitá data, Použitý software

2.1 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.

(*Zdroj: wiki.openstreetmap.org)*

OpenStreetMap byl pro tento projekt jedním ze dvou použitých zdrojů vektorových dat. Druhý zdroj dat byla databáze gis1, používaná při výuce předmětu GIS1 na FSv ČVUT v Praze.

2.2 DMT z datasetu FreeGeoDataCZ

```
r_table_catalog  | pgis_uzpd
r_table_schema   | public
r_table_name     | dmt
r_raster_column  | rast
srid             | 2065
scale_x          | 59.9999373525213
scale_y          | -89.991627410955
blocksize_x      | 7881
blocksize_y      | 3246
same_alignment   | t
regular_blocking | t
num_bands        | 1
pixel_types      | {32BF}
nodata_values    |
out_db           | {f}
```

Tento rastr DMT České Republiky byl použit pro zjištění výškových informací.

2.3 PostGIS

PostGIS je open source software. Přidává podporu pro geografické objekty objektově-relačnímu databázovému systému PostgreSQL. PostGIS implementuje specifikaci "Simple Features for SQL" konsorcia Open Geospatial Consortium.

(*Zdroj: cs.wikipedia.org)*

2.4 PgRouting

PgRouting je rozšíření pro PostGIS určené pro síťové analýzy. Jsou zde implementovány tyto síťové analýzy:

- Vyhledání nejkratší cesty
- Problém obchodního cestujícího

- Dojezdová vzdálenost

(*Zdroj: geo.fsv.cvut.cz/freegis)*

V tomto projektu byla použita pouze možnost vyhledání nejkratší cesty.

2.5 PostGIS Raster

PostGIS Raster je rozšíření pro PostGIS umožňující uložení, manipulaci, zpracování a dotazování rastrových dat v prostředí PostGIS.

(*Zdroj: geo.fsv.cvut.cz/freegis)*

2.6 PostGIS Topology

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

(*Zdroj: geo.fsv.cvut.cz/freegis)*

2.7 Quantum GIS

Quantum GIS (zkráceně QGIS) je svobodný a multiplatformní geografický informační systém (GIS).

Vývoj, který započal roku 2002, zajišťuje skupina dobrovolníků, verze s označením 1.0 vyšla 5. ledna 2009. QGIS je psán v jazyku C++, grafické uživatelské rozhraní je postaveno na knihovně Qt. Zásuvné moduly je možno vytvářet v C++ nebo Pythonu.

QGIS umožňuje zejména prohlížení, tvorbu a editaci rastrových a vektorových vrstev, zpracování GPS dat a tvorbu map. Funkčnost rozšiřují zásuvné moduly, významný je modul zpřístupňující funkce GRASS GISu – QGIS tak může sloužit jako jeho nadstavba.

(*Zdroj: cs.wikipedia.org)*

Quantum GIS byl použit pro vizualizaci vytvořených dat.

2.8 pgAdmin

pgAdmin je free a open source grafické administrační rozhraní pro PostgreSQL, podporovaný na většině populárních platforem. Program je dostupný pro více než tucet jazyků. První prototyp, nazvaný pgManager, byl napsán pro PostgreSQL 6.3.2 v roce 1998, a po několika měsících přepsán a vydán pod GPL licencí jako pgAdmin. Druhá inkarnace (nazvaná pgAdmin II) byla opět kompletním přepsáním, a byla vydána 16 ledna 2002. Aktuální verze je pgAdmin III, původně vydaná pod Artistic License a nyní je vydávána pod stejnou licencí jako PostgreSQL. Na rozdíl od předchozích verzí, psaných v jazyce Visual Basic, je pgAdmin III napsán v C++ pomocí frameworku wxWidgets což mu umožňuje běžet na nejběžnějších operačních systémech.

(*Zdroj: cs.wikipedia.org)*

2.9 Přehled verzí použitých softwarů

- pgAdmin – 1.16.1
- Quantum GIS – 1.8.0 Lisboa
- PostGIS – 2.0
- pgRouting – 1.05

3 Tvorba tematických vrstev

Po průzkumu dat z OpenStreetMap a témat použitých v minulých letech, bylo vybráno téma lyžování a s ním související vrstvy:

- Bodové:
  - Obce
  - Vlaky
  - Ubytování
- Liniové:
  - Železnice
  - Vleky
- Polygonové:
  - Kraje
  - Obce_pol

Níže bude popsáno vytvoření jednotlivých vrstev a řešení případných problémů při jejich vzniku.

3.1 Obce

- Překopírování bodové vrstvy ze schémata gis1 do schémata d13:

```
CREATE TABLE        obce
AS SELECT           ogc_fid, kodob, nazob_eng, nazorp_eng, nk, geom FROM obce_b;
```

- Nastavení primárního klíče a prostorového indexu

```
ALTER TABLE         obce
ADD PRIMARY KEY     (ogc_fid);
CREATE INDEX        obce_geom
ON                  obce
USING               gist (geom);
```

- Transformace ze systému S-JTSK (2065) do systému Google Mercator (900913)

```
ALTER TABLE         obce
RENAME COLUMN       geom
TO                  geom_beta;
 SELECT             AddGeometryColumn('obce', 'geom', 900913, 'point', 2);
UPDATE              obce
SET                 geom = ST_Transform(geom_beta, 900913);
SELECT              DropGeometryColumn('obce','geom_beta');
```

U této vrstvy nebylo třeba řešit žádný problém, pouze bylo potřeba změnit souřadnicový systém.

3.2 Vlaky

Pod vrstvou vlaky se skrývají železniční stanice, které jsou stejně jako obce zkopírovány ze schématu gis1. Tvorba této vrstvy je naprosto totožná jako tvorba vrstvy obce, proto zde nebude dále rozebírána.

3.3 Ubytování

Bodová vrstva ubytování, byla vybrána z tabulky czech_point ze schématu osm (OpenStreetMap). Několik ubytovacích zařízení se nalézalo i v tabulce czech_polygon. Polygonové záznamy byly převedeny na bodovou vrstvu a sloučena s body získanými z tabulky czech_point.

- Výběr hotelů, hostelů, chat a apartmánů z tabulky czech_point:

```
CREATE TABLE    ubytovani
AS SELECT       osm_id, geom, name AS nazev, tourism AS typ
FROM            czech_point
WHERE           tourism IN ('chalet', 'guest_house', 'hostel', 'hotel');
```

- Výběr hotelů, hostelů, chat a apartmánů z tabulky czech_polygon a jejich následný převod na body:

```
CREATE  TABLE   ubytovani_pol
AS SELECT       osm_id, geom, name AS nazev, tourism AS typ
FROM            czech_polygon AS cp
WHERE           tourism IN ('chalet', 'guest_house', 'hostel', 'hotel')
AND             osm_id NOT IN (     SELECT cz_p.osm_id
                                    FROM   ubytovani AS ub
                                    JOIN   czech_polygon AS cz_p
                                    ON     ST_contains(cz_p.geom,ub.geom)
                                    WHERE          cz_p.tourism IN ('chalet',
'guest_house', 'hostel', 'hotel'));
CREATE TABLE    ubytovani_bod
AS SELECT       osm_id, ST_Centroid (ubytovani_pol.geom) AS geom, nazev, typ
FROM            ubytovani_pol;
INSERT INTO     ubytovani(osm_id, geom, nazev, typ)
SELECT          *
FROM            ubytovani_bod;
DROP TABLE      ubytovani_pol;
DROP TABLE      ubytovani_bod;
```

Dále byl přidán primární klíč a vytvořen prostorový index stejně jako u vrstvy obce.

Nakonec byl ještě pro přehlednost přeložen atribut guest_house na apartman a chalet na chalupu.

```
UPDATE          ubytovani SET typ='apartman' WHERE typ='guest_house';
UPDATE          ubytovani SET typ='chalupa' WHERE typ='chalet';
```

3.4 Železnice

Liniová vrstva železnice byla vytvořena stejně jako bodové vrstvy obce a vlaky, proto zde její tvorba nebude více rozebírána. Železnice jsou do databáze přidány pro následnou ilustraci nadstavby pgRouting. Původně pro tuto ilustraci měly být použity silnice z OpenStreetMap, ale tato data byla zbytečně velká a navíc dost nekonzistentní, proto byly nakonec zvoleny železnice, se kterými se dá pracovat mnohem rychleji a navíc neobsahují žádné „díry“.

3.5 Vleky

Liniová vrstva vleky je hlavní vrstvou tohoto projektu. Této vrstvy se bude týkat většina atributových dotazů. Její tvorba probíhala velmi podobně jako tvorba vrstvy ubytování, až na pár drobností, které budou níže probrány. Ještě je třeba říci, že aby bylo možno tento projekt realizovat, jsou vleky brány jako sjezdovky, je možné že ne každý vlek je zároveň sjezdovkou, ale v 99 procentech případů to platí. Dále je třeba vědět, že databáze OpenStreetMap není zdaleka kompletní, proto je možné, že ne všechny výsledky, které budou na konci uvedeny v atributových a prostorových dotazech, budou korektní a souhlasící s realitou.

- Bylo třeba odstranit vleky ležící mimo území ČR:

```
DELETE FROM     vleky
WHERE           vleky.geom  NOT IN (  SELECT vleky.geom
                            FROM          vleky
                            JOIN          kraje
                            ON            ST_Within(vleky.geom,kraje.geom));
```

- Byla dopočítána délka každého vleku:

```
ALTER TABLE  vleky
ADD          delka float;
UPDATE       vleky
SET          delka = ST_LENGTH(geom);
```

- Také bylo třeba odstranit vleky, které logicky nemohou být sjezdovkou

```
DELETE FROM   vleky
WHERE         delka < 50;
```

- Nakonec byly opět přeloženy typy vleků do češtiny, podobně jako tomu bylo u vrstvy ubytování:

```
UPDATE vleky SET typ='lanovka' WHERE typ='cable_car';
UPDATE vleky SET typ='gondola' WHERE typ='gondola';
UPDATE vleky SET typ='sedackova_lanovka' WHERE typ='chair_lift';
UPDATE vleky SET typ='smisena_lanovka' WHERE typ='mixed_lift';
UPDATE vleky SET typ='poma' WHERE typ='t-bar';
UPDATE vleky SET typ='poma' WHERE typ='j-bar';
UPDATE vleky SET typ='kotva' WHERE typ='drag_lift';
```

3.6 Kraje

Tvorba polygonové vrstvy byla zřejmě nejsložitější, protože bylo třeba nejprve spojit polygonovou vrstvu obcí ze schématu gis1 na základě atributu nk (název kraje). Poté se ještě výsledná vrstva musela převést na MultiPolygony.

- Vytvoření vrstvy kraje:

```
CREATE TABLE   kraje (ogc_fid SERIAL PRIMARY KEY, nazev varchar(250), geom geometry);
INSERT INTO    kraje (geom,nazev)
SELECT         ST_Union(obce_pol.geom) AS geom, obce_pol.nk AS nazev
FROM           obce_pol
GROUP BY       nk;
```

- Převod na multipolygony:

```
CREATE TABLE   kraje_beta
AS SELECT      nazev, ST_Union(geom) AS geom
FROM           kraje
GROUP BY       nazev;
```

```
CREATE TABLE   kraje_gama
AS SELECT      *
FROM           kraje;

DELETE FROM    kraje_gama
WHERE          ctid
NOT IN         (SELECT MAX(ctid) FROM kraje GROUP BY nazev);
SELECT          DropGeometryColumn('kraje_gama', 'geom');
SELECT         AddGeometryColumn('kraje_gama', 'geom', 2065, 'MultiPolygon', 2);
UPDATE         kraje_gama AS a
SET            geom = ST_Multi(b.geom)
FROM           kraje_beta AS b WHERE a.nazev = b.nazev;
DROP TABLE     kraje_beta;
ALTER TABLE    kraje_gama
RENAME TO      kraje_kraje;
DROP TABLE     kraje;
ALTER TABLE    kraje_kraje
RENAME TO      kraje;
```

Tvorba primárního klíče, prostorového indexu a následná transformace do souřadnicového systému Google Mercator proběhla stejně jako u bodových vrstev obce a vlaky.

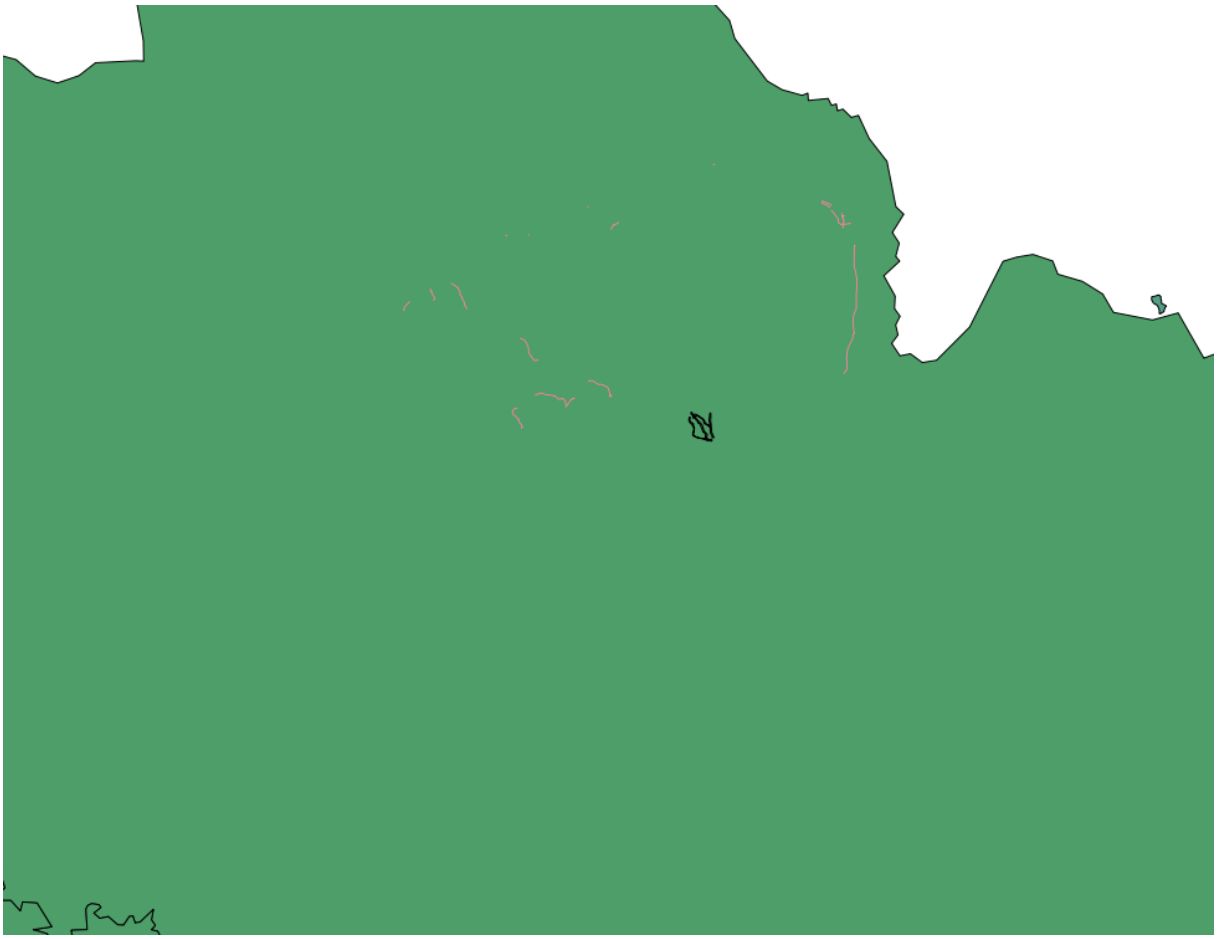
3.7 Obce_pol

Tvorba polygonové vrstvy obcí probíhala stejně jako tvorba vrstvy krajů, proto zde nebude více rozebrána.

3.8 Lyžování a lyžařské cesty

Nakonec byly ještě vytvořeny 2 vrstvy z OpenStreetMap. Lyžování, která měla znázorňovat oblasti, kde se lyžuje a lyžařské/běžkařské cesty. S těmito vrstvami se nakonec vůbec nepracovalo, protože byly

nekompletní a obsahovaly minimum dat. Pro ilustraci je níže přiložen obrázek Krkonoš, kde je vidět, že tyto vrstvy opravdu neobsahují téměř žádná data.



4 Validace dat

Pokud chceme v prostředí PostGIS provádět prostorové dotazy, je třeba zajistit, aby všechna data byla geometricky korektní. K zjištění zda jsou data korektní, slouží tento příkaz:

```
SELECT      osm_id, ST_IsValidReason(geom)
FROM        vleky
WHERE NOT   ST_IsValid(geom);
```

Funkce ST_IsValid vrací hodnotu TRUE, jestli je geometrie bez problémů, pokud ne tak vypíše důvod proč tomu tak není.

Obdobně byly takto testovány všechny vrstvy. Chyba byla nalezena pouze u polygonové vrstvy obce, kde nastal problém *ring self-intersection*. Za pomoci internetového fóra byl tento problém opraven takto: ještě před převodem obcí na multipolygony byl na tuto vrstvu aplikován „nulový buffer“.

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

To problém *ring self-intersection* odstranilo.

Dále byla na všechny vrstvy puštěna kontrola, zda v každé vrstvě existuje sloupec geometrie a zda má správná prostorová omezení:

```
SELECT      populate_geometry_columns('d13.vleky'::regclass);
```

Takto byly zkontrolovány všechny vrstvy. Nebyl zjištěn ani jeden problém.

5 Topologie

Tvorba topologie bude předvedena na vrstvě vleky. Stejně probíhala pro všechny ostatní vrstvy.

- Vytvoření nové topologie:

```
SELECT CreateTopology ('topo_vleky');
```

- Přidání sloupečku s topologií do vrstvy vleky:

```
SELECT AddTopoGeometryColumn ('topo_vleky', 'd13', 'vleky', 'topo', 'POLYLINE');
```

- Automatická tvorba topologie pomocí funkce toTopoGeom:

```
UPDATE vleky        SET topo = toTopoGeom (geom, 'topo_vleky', 1);
```

- Kontrola topologie:

```
SELECT      TopologySummary     ('topo_vleky');
SELECT      ValidateTopology    ('topo_vleky');
```

Topologie byla bez problémů vytvořena pro všechny vrstvy.

6 Přidání výškových informací vlekům pomocí PostGIS Raster

- Nejprve byl zkopírován dmt rastr do schématu d13

```
CREATE TABLE   d13.dmt
AS SELECT      *
FROM           dmt;
```

Nyní byla spuštěna transformace rastru do souřadnicového systému Google Mercator, ale ta ani po hodině nedoběhla. Proto byl zvolen opačný postup. Vrstva vleky byla převedena do S-JTSK (ve kterém byl i dmt rastr), v tomto souřadnicovém systému byly doplněny výškové informace a následně byla vrstva vleky zpětně převedena do systému Google Mercator.

- Dále následovalo přidání začátečních a koncových bodů každého vleku:

```
ALTER TABLE    vleky_jtsk
ADD COLUMN     sp GEOMETRY('point', 2065);
ALTER TABLE    vleky_jtsk
ADD COLUMN     ep GEOMETRY('point', 2065);
UPDATE              vleky_jtsk
SET            sp = ST_StartPoint(geom);
UPDATE              vleky_jtsk
SET            ep = ST_EndPoint(geom);
```

- Poté byla z rastru zjištěna výška každého počátečního a koncového bodu, tyto informace byly předány do vrstvy v systému Google Mercator a pomocná vrstva v S-JTSK byla smazána:

```
ALTER TABLE    vleky_jtsk
ADD COLUMN     sp_vyska FLOAT;
ALTER TABLE    vleky_jtsk
ADD COLUMN     ep_vyska FLOAT;
UPDATE              vleky_jtsk
```

```
SET         sp_vyska = st_value(rast, sp)
FROM        dmt;
UPDATE              vleky_jtsk
SET         ep_vyska = st_value(rast, ep)
FROM        dmt;

UPDATE              vleky AS v1
SET         sp_vyska = v2.sp_vyska
FROM        vleky_jtsk AS v2
WHERE       v1.osm_id = v2.osm_id;
UPDATE              vleky AS v1
SET         ep_vyska = v2.ep_vyska
FROM        vleky_jtsk AS v2
WHERE       v1.osm_id = v2.osm_id;
DROP TABLE  vleky_jtsk;
```

- Nakonec byly ještě dopočítány informace o sklonu a převýšení jednotlivých vleků:

```
ALTER TABLE  vleky
ADD COLUMN   prevyseni float;
UPDATE              vleky
SET          prevyseni = ABS(sp_vyska - ep_vyska);
ALTER TABLE  vleky
ADD COLUMN   sklon float;
UPDATE              vleky
SET          sklon = (prevyseni/delka)*100;
```

Je třeba pamatovat na to, že DMT má rozlišení 60x90 metrů. Proto krátké sjezdovky mohou mít i nulové převýšení. Informace je třeba brát spíše jako orientační.

7 pgRouting

Aby bylo možno pracovat s nadstavbou pgRouting, bylo třeba poupravit vrstvu železnice. Každá hrana musela dostat informaci o jejím začátečním a koncovém bodu a o indexu těchto bodů. Při řešení tohoto problému jsme se během tvorby našeho projekt inspirovali na stránce *http://anitagraser.com*. Po nastudování této stránky jsme přípravu pro pgRouting vyřešili takto:

```
CREATE VIEW    zeleznice_ext AS
SELECT         *, st_startpoint(geom), st_endpoint(geom)
FROM           zeleznice;
CREATE TABLE   node
AS SELECT      row_number() OVER (ORDER BY foo.p)::integer AS id,
        foo.p AS geom
FROM           (
                    SELECT DISTINCT st_startpoint AS p FROM zeleznice_ext
                    UNION
                    SELECT DISTINCT st_endpoint AS p FROM zeleznice_ext
               ) foo
GROUP BY       foo.p;
CREATE TABLE   zeleznice_network
AS SELECT      a.*, b.id as start_id, c.id as end_id
FROM           zeleznice_ext AS a
JOIN           node AS b
ON             a.st_startpoint = b.geom
JOIN           node AS c
ON             a.st_endpoint = c.geom;
ALTER TABLE    zeleznice_network
RENAME COLUMN      ogc_fid TO gid;
ALTER TABLE    zeleznice_network
RENAME COLUMN      geom TO the_geom;
```

```
ALTER TABLE zeleznice_network
RENAME COLUMN start_id TO source;
ALTER TABLE zeleznice_network
RENAME COLUMN end_id TO target;
ALTER TABLE    zeleznice_network
ADD COLUMN   length float;
UPDATE         zeleznice_network
SET              length = st_distance(st_startpoint, st_endpoint);
```

8 TUTORIAL – DOTAZY

8.1 Atributové

- Kolik je na území ČR sjezdovek delších než 2km?

```
SELECT        delka, typ
FROM          vleky
WHERE         delka > 2000;
```

Odpověď: **22**

- Které vesnice v Olomouckém kraji začínají na písmeno U?

```
SELECT        nazob_eng
FROM          obce
WHERE         nk = 'OL' AND nazob_eng LIKE 'U%';
```

Odpověď: **Uhelna, Ujezd, Unicov, Ustin, Urcice, Uhricice, Usti, Usov**

- Kolik ubytovacích zařízení z celkového počtu má v databázi uveden název?

```
SELECT        count(*)
FROM          ubytovani
WHERE         nazev IS NOT NULL;
```

```
SElECT      count(*)
FROM        ubytovani;
```

Odpověď: **2297/2477 (93%)**

- Ve kterém kraji se nachází nejvíce obcí?

```
SELECT      nk, count(*) AS pocet
FROM        obce
GROUP BY    nk
ORDER BY    pocet DESC
LIMIT       1;
```

Odpověď: **ST - Středočeský kraj (1146)**

8.2 Prostorové

B) PROSTOROVE DOTAZY

- Vyjmenujte 5 obci s největším počtem ubytovacích zařízeni (+jejich počet).

```
SELECT      nazev_eng AS mesto, count (DISTINCT u.osm_id) AS pocet
FROM        obce_pol AS o
JOIN        ubytovani AS u
ON          ST_Contains(o.geom, u.geom)
GROUP BY    mesto
ORDER BY    pocet DESC
LIMIT       5;
```

Odpověď: **Praha(286), Spindleruv Mlyn(114), Brno(53), Luhacovice(47), Bozi Dar(41)**

- Které ubytovací zařízení leží nejblíže nějaké sjezdovce? A ve které obci leží?

```
SELECT      ROUND(ST_Distance(u.geom, v.geom)) AS vzdalenost, u.nazev, u.typ
FROM        ubytovani AS u
JOIN        vleky AS v
ON          ST_Disjoint(u.geom, v.geom)
ORDER BY    vzdalenost
LIMIT       1;
```

```
SELECT      DISTINCT nazev_eng AS obec
FROM        obce_pol
JOIN        ubytovani
ON          ST_Contains(obce_pol.geom, (SELECT ubytovani.geom FROM ubytovani WHERE nazev='Apartmán Novako'));
```

Odpověď: Apartmán Novako, 26m od sjezdovky, obec Bozi Dar

- Která železniční stanice je nejvhodnější výstupní stanice pro lyžaře? (V jejím 2km okolí leží nejvíce sjezdovek?)

```
před optimalizaci       (ani po 5 minutách dotaz neproběhl):
```

```
SELECT      nazev, count(nazev) AS pocet_sjezdovek
FROM        vlaky
JOIN        vleky
ON          ST_Intersects(vleky.geom, ST_Buffer(vlaky.geom, 2e3))
GROUP BY    nazev
ORDER BY    pocet_sjezdovek DESC
LIMIT       2
```

```
-- po optimalizaci      (83 ms)
```

```
SELECT      nazev, count(nazev) AS pocet_sjezdovek
FROM        vlaky
JOIN        vleky
ON          ST_Expand(vlaky.geom, 2e3) && vleky.geom
AND         ST_Intersects(vleky.geom, ST_Buffer(vlaky.geom, 2e3))
GROUP BY    nazev
ORDER BY    pocet_sjezdovek DESC
LIMIT       2
```

Odpověď: **Velke Karlovice (11 sjezdovek do 2 km), Velke Karlovice zast. (10 sjezdovek do 2 km)**

- Na území kterého kraje je největší poměr obyvatel na počet ubytovacích zařízení?

```
CREATE TABLE  pomoc
AS SELECT     kraje.nazev,
              COUNT(ubytovani.nazev) AS pocet
              FROM        kraje
              JOIN        ubytovani
              ON          ST_Contains (kraje.geom, ubytovani.geom)
              GROUP BY    kraje.nazev;
ALTER TABLE   pomoc
ADD COLUMN    obyvatele float;
UPDATE              pomoc
SET           obyvatele = (  SELECT      SUM(obyv02)
                             FROM        obce_pol
                             WHERE       pomoc.nazev = obce_pol.nk
                             GROUP BY    nk)
FROM          obce_pol
WHERE         pomoc.nazev = obce_pol.nk;
```

```
SELECT      nazev, obyvatele/pocet AS foo
FROM        pomoc
ORDER BY    foo DESC;

DROP TABLE  pomoc;
```

Odpověď: Pardubický kraj

- Existuje v ČR nějaká sjezdovka, která se nachází na území dvou krajů?

```
SELECT      COUNT(DISTINCT osm_id)
FROM        vleky
JOIN        kraje
ON          ST_Crosses(vleky.geom, ST_Boundary(kraje.geom));
```

Odpověď: ne

- Nejblíže, které obci (bod), železniční zastávce a ubytovacímu zařízení se nachází sjezdovka s největším převýšením?

```
SELECT      osm_id
FROM        vleky
ORDER BY    prevyseni desc
LIMIT       2
```

(limit 2, protože první nemá určené převýšení)

```
SELECT      nazob_eng, St_Distance(o.geom, v.geom) AS vzdalenost
FROM        obce AS o
JOIN        vleky AS v
```

```
ON          v.osm_id = 27556621
ORDER BY    vzdalenost
LIMIT       1;

SELECT      nazev, St_Distance(o.geom, v.geom) AS vzdalenost
FROM        vlaky AS o
JOIN        vleky AS v
ON          v.osm_id = 27556621
ORDER BY    vzdalenost
LIMIT       1;

SELECT      nazev, o.typ,  St_Distance(o.geom, v.geom) AS vzdalenost
FROM        ubytovani AS o
JOIN        vleky AS v
ON          v.osm_id = 27556621 AND nazev LIKE '%'
ORDER BY    vzdalenost
LIMIT       1;
```

Odpověď: Rokytnice nad Jizerou, železniční stanice Rokytnice nad Jizerou a nejlépe se ubytovat v hotelu Rokytka

- Kde se ubytovat pokud chceme lyžovat na nejprudší sjezdovce ve vzdálenosti do 100 km od Prahy, která má alespoň 1500m ?

```
SELECT      osm_id
FROM        vleky
JOIN        obce
ON          ST_Expand(obce.geom, 1.5e5) && vleky.geom
AND         nazob_eng = 'Praha'
```

```
AND         ST_Intersects(vleky.geom, ST_Buffer(obce.geom, 1.5e5))
AND         delka > 1500
ORDER BY    sklon DESC
LIMIT       1;

--96934872

SELECT      nazev, o.typ,  St_Distance(o.geom, v.geom) AS vzdalenost
FROM        ubytovani AS o
JOIN        vleky AS v
ON          v.osm_id = 96934872 AND nazev LIKE '%'
ORDER BY    vzdalenost
LIMIT       1;

--hotel Florian


SELECT      o.nazev_eng
FROM        obce_pol AS o
JOIN        ubytovani AS u
ON          ST_Contains(u.geom, o. geom)
AND         u.nazev = 'Florián'
```

Odpověď: **obec Svetla pod Jestedem (skiareal Jested), Hotel Ještěd**

- Jaká je výměra území, které se nachází v nadmořské výšce nad 1350m.

```
CREATE TABLE   dmt1350
AS SELECT      ST_MapAlgebraExpr(rast, 1, NULL, 'CASE WHEN [rast] > 1350 THEN 1 ELSE 0 END') AS
rast
```

```
FROM        dmt;

SELECT      ST_ValueCount(rast) from dmt1350;
DROP TABLE  dmt1350;
```

Odpověď: 2797,74 ha

- Jaká je výměra území, které je ve vzdálenosti 2000m od jakékoliv sjezdovky a nachází se v horách (nad 1000 mnm)?

```
CREATE TABLE  vleky_jtsk
AS SELECT     *
FROM          vleky;

ALTER TABLE   vleky_jtsk
RENAME COLUMN     geom
TO            geom_beta;

SELECT        AddGeometryColumn('vleky_jtsk', 'geom', 2065, 'linestring', 2);

UPDATE              vleky_jtsk
SET           geom = ST_Transform(geom_beta, 2065);

SELECT        DropGeometryColumn('vleky_jtsk','geom_beta');

CREATE TABLE  vleky_buff
AS SELECT     ST_Union(ST_Buffer(geom, 2000)) AS geom
FROM          vleky_jtsk;
```

```
SELECT (ST_SummaryStats(rast)).* AS stats
FROM   dmt;

CREATE TABLE   dmt1000
AS SELECT      ST_Reclass(rast, 1, '1000-1594:1', '4BUI') AS rast
FROM           dmt;

CREATE TABLE   dmt_vleky_buff
AS SELECT      ST_Intersection(rast, geom) as geomval
FROM           dmt1000
CROSS JOIN     vleky_buff;

CREATE TABLE   dmt_vleky_geom
AS SELECT      (gv).geom AS geom, (gv).val AS val
FROM
(
SELECT         geomval AS gv
FROM           dmt_vleky_buff
) AS           foo;

SELECT         SUM(ST_Area(geom))
FROM           dmt_vleky_geom
WHERE          val = 1;

DROP TABLE     vleky_jtsk;
DROP TABLE     vleky_buff;
DROP TABLE     dmt1000;
DROP TABLE     dmt_vleky_buff;
DROP TABLE     dmt_vleky_geom;
```

Odpověď: **28134 ha**

-- 10) Jakou železniční cestou jet, pokud se chci dostat k nejbližší sjezdovce z Olomouce, která je alespoň 1000m dlouhá?

```
SELECT      osm_id, ST_Distance(v.geom, o.geom) AS vzd
FROM        vleky AS v
JOIN        obce AS o
ON          o.nazob_eng = 'Olomouc'
AND         delka > 2500
ORDER BY    vzd;
```

-- (91641416)

```
SELECT      nazev, ST_Distance(s.geom, v.geom) AS vzd
FROM        vlaky AS s
JOIN        vleky AS v
ON          v.osm_id = 91641416
ORDER BY    vzd
LIMIT       1;
```

-- (Kouty nad Desnou)

```
SELECT      z.source, ST_Distance(z.the_geom, v.geom) AS vzd
FROM        zeleznice_network AS z
JOIN        vlaky AS v
ON          v.nazev = 'Kouty nad Desnou'
ORDER BY    vzd
LIMIT       1;
```

-- target id = 594

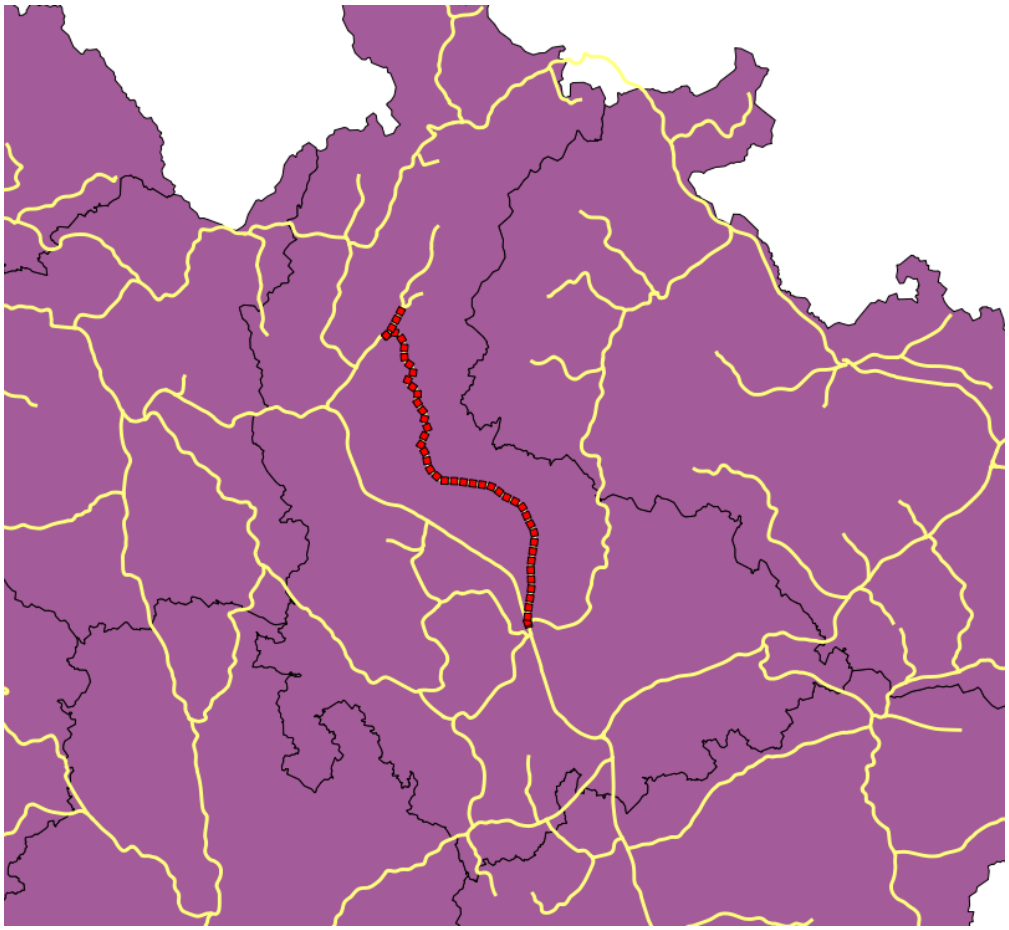
```
SELECT      z.source, nazev, ST_Distance(z.the_geom, v.geom) AS vzd
FROM        zeleznice_network AS z
JOIN        vlaky AS v
ON          v.nazev LIKE 'Olom%'
ORDER BY    vzd
LIMIT       1;


-- source id: 635


SELECT * FROM shortest_path('SELECT ogc_fid as id,start_id as source,end_id as target,length as cost
from zeleznice_network',


635, 594, FALSE, FALSE);


CREATE TABLE   olomouc_kouty
AS SELECT      gid, the_geom
FROM           dijkstra_sp('zeleznice_network', 635, 594);
```



- Kolik ubytovacích zařízení se nachází do 2 km od železniční trati Praha - Olomouc? A které je nejblíže kolejím?

```
-- source id 635 (z predchozi ulohy)

SELECT      z.source,nazev, ST_Distance(z.the_geom, v.geom) AS vzd
FROM        zeleznice_network AS z
JOIN        vlaky AS v
ON          v.nazev = 'Praha hl.n.'
ORDER BY    vzd
LIMIT       1;
```

```
--target id 258

CREATE TABLE   olomouc_praha
AS SELECT      gid, the_geom
FROM           dijkstra_sp('zeleznice_network', 635, 258);

CREATE TABLE   trat_buff
AS SELECT      ST_Union(ST_Buffer(the_geom, 2000)) AS geom
FROM           olomouc_praha;

SELECT         count(nazev)
FROM           ubytovani as u
JOIN           trat_buff as t
ON             ST_Intersects(u.geom, t.geom);

SELECT         nazev,typ, ST_Distance(u.geom, z.geom) AS vzd
FROM           ubytovani as u
JOIN           trat_buff as t
ON             ST_Intersects(u.geom, t.geom)
JOIN           zeleznice as z
ON             ST_Distance(z.geom, u.geom) > 0
ORDER BY       vzd
LIMIT          1;

SELECT         nazev_eng
FROM           obce_pol AS o
JOIN           ubytovani AS u
ON             ST_Within(u.geom, o.geom)
WHERE          u.nazev = 'Poprad'
```

Odpověď: **113, nejblíže je hotel Poprad, Ústí nad Orlici**

9 Závěr

Snažili jsme se vybrat si téma, které doposud v předmětu Úvod do Zpracování Prostorových dat nikdo nezpracoval, a předvést na něm všechny možnosti prostorové nadstavby databáze PostgreSQL, PostGIS. Dotazy jsme se snažili tvořit lehce pochopitelné a snadno představitelné. Myslím, že to se nám povedlo, protože během našeho tutoriálu se uživatel seznámí s atributovými tak i prostorovými dotazy a nakonec nahlédne i na práci s nadstavbami PostGIS Raster a pgRouting. Při tvorbě databáze je také ukázána tvorba topologie. Takže jsou zde ve zjednodušené formě předvedeny všechny oblasti prostředí PostGIS. Databáze bohužel není ani zdaleka kompletní, a aby se dala používat jako plnohodnotná příručka pro nadšené lyžaře, bylo by ji třeba doplnit a zpřesnit. Pro všeobecnou orientaci ale určitě postačí.