Úvod do zpracování prostorových dat
semestrální práce

Anna Kratochvílová
Nikola Němcová
Václav Petráš

14. května 2010

1 Úvod

1.1 Zadání

- Navrhněte a vytvořte tématické vrstvy (např. vodní toky, vodní plochy, lesy, silnice, železnice apod.) na základě dat *OpenStreetMap* (viz cvičná databáze `pgis_student` schéma `osm`). Pro tento účel byla na serveru `josef` založena databáze `pgis_osm`.
- Aplikujte testy datové integrity a odstraňte případné nekonzistence v datech.
- Vytvořte tutoriál pro výuku PostGIS - tj. sadu atributových a prostorových dotazů nad databází pgis_osm.

1.2 Výběr tématu

Vybrali jsme si témetické vrstvy lesy, vojenské újezdy a bunkry. Lesy jsme si vybrali, protože jsou kompletní pro celou republiku narozdíl od ostatních polygonových vrstev. Vzhledem k množství záznamů je i hodně nevalidních polygonů, proto jsme se rozhodli zaměřit naši práci především na validaci této vrstvy.

2 Výběr vrstev

V databázi `pgis_osm`, schématu `public` jsme z tabulky `czech_polygon` vybrali polygony pro vrstvu lesů, vojenských újezdů a bunkrů. Dávka pro vytvoření vrstvy `lesni_porosty` vypadala takto:

```
CREATE TABLE f10.lesni_porosty AS
   SELECT osm_id,name,way
     FROM czech_polygon
    WHERE landuse = 'forest';

ALTER TABLE f10.lesni_porosty ADD COLUMN gid SERIAL;
ALTER TABLE f10.lesni_porosty ADD PRIMARY KEY (gid);

SELECT Populate_geometry_columns
            ('f10.lesni_porosty'::regclass);
CREATE INDEX lesni_porosty_gist
    ON f10.lesni_porosty USING GIST (way);
```

Dávka pro vytvoření vrstvy vojenských újezdů z vrstvy `czech_polygon`:

```
CREATE TABLE f10.ujezdy AS
  SELECT name,military,way
    FROM czech_polygon
```

```
   WHERE military = 'danger_area'
     AND name like '%újezd%';

ALTER TABLE f10.ujezdy ADD COLUMN gid serial;
ALTER TABLE f10.ujezdy ADD PRIMARY KEY (gid);

SELECT Populate_geometry_columns('f10.ujezdy'::regclass);
CREATE INDEX f10_ujezdy_gist ON f10.ujezdy USING GIST (way);
```

Pro vrstvu bunkrů byly použity vrstvy `czech_polygon` a `czech_point`. V atributech *name* a *ref* byl zmatek, proto jsme se rozhodli nejdříve vytvořit prázdnou tabulku a do ní následně vkládat záznamy s těmito atributy tak, aby dávaly smysl. Polygonová vrstva měla atributy v pořádku, jen jsme převedli polygony na body, což má u bunkrů větší logiku. V bodové vrstvě se v atributu *name* vyskytovalo číselné označení bunkru, proto jsme jej vložili do atributu *ref* (všechny hodnoty byly NULL) na základě lomítka obsaženého v textu. V některých případech je v *name* obsaženo jméno i číselné označení dohromady, to bylo tak i ponecháno.

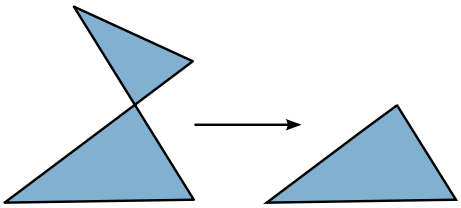
```
CREATE TABLE f10.bunkry(
      gid SERIAL PRIMARY KEY,
   osm_id INTEGER,
     name TEXT,
      ref TEXT,
      way GEOMETRY);

INSERT INTO f10.bunkry (osm_id,ref,way)
  SELECT osm_id, name, way
    FROM czech_point
   WHERE military = 'bunker'
     AND name LIKE '%/%';

INSERT INTO f10.bunkry (osm_id,name,way)
  SELECT osm_id, name, way
    FROM czech_point
   WHERE military = 'bunker'
     AND (name NOT LIKE '%/%' OR name IS NULL);

INSERT INTO f10.bunkry (osm_id,name,ref,way)
  SELECT osm_id, name,ref, ST_Centroid(way)
    FROM czech_polygon
   WHERE military = 'bunker';
```

Data jsme si zobrazovali v programu *QGis*.



Obrázek 1: Chování bufferu při *self-intersection*

3 Validace

Bylo zjištěno, že ze 67146 je 289 polygonů nevalidních. K zjištění validity slouží funkce `ST_IsValid`. Důvody nevalidity a problematické místo (avšak ne úplně dobře vyjádřené) můžeme zjistit pomocí funkce `ST_IsValidReason`. Ve vrstvě `lesni_porosty` se objevily tyto problémy:

- Self-intersection
- Hole lies outside shell
- Ring Self-intersection
- Duplicate Rings
- Holes are nested

Toto rozdělení je však pouze orientační, protože v jednom polygonu často docházelo ke kombinaci více problémů. Z toho vychází fakt, že problémy je třeba řešit všechny najednou. Pro validaci polygonů se často doporučuje použít funkci `ST_Buffer(geometry, 0.0)`, která vytvoří obalovou zónu o šíři nula. Toto řešení není použitelné, jakmile nastává křížení (například polygon má tvar osmičky). Dochází ke ztrátě dat (viz obr. 1) a u některých polygonů dokonce docházelo k ukončení spojení databázovým serverem. Hledali jsme tedy jiné řešení a našli jsme funkci `cleanGeometry` od Dr. Horst Duester, jejíž zdrojový kód psaný v PL/pgSQL je v příloze A, a stránku `http://trac.osgeo.org/postgis/wiki/UsersWikiCleanPolygons`, kde je řešen problém typu *self-intersection*. Tímto jsme se při sestavování validačních příkazů inspirovali. Nakonec jsme dospěli k následující dávce příkazů:

```
CREATE TABLE f10.valid AS

SELECT gid, ST_BuildArea(ST_Collect(way)) AS way
  FROM
    (SELECT
      gid,
      ST_BuildArea(ST_Union(ring.way, ST_StartPoint(ring.way)))
         AS way
```

```
       FROM
        (SELECT DISTINCT
          gid,
          ST_ExteriorRing((ST_DumpRings(way)).geom) AS way
           FROM f10.lesni_porosty
          WHERE NOT ST_IsValid(way))
             AS ring)
         AS after_union
GROUP BY gid;

ALTER TABLE f10.valid ADD PRIMARY KEY (gid);
SELECT Populate_Geometry_Columns('f10.valid'::regclass);
CREATE INDEX valid_gist ON f10.valid USING GIST (way);
```

Tímto jsme zároveň vytvořili tabulku `valid`, která obsahuje opravené polygony.

3.1 Průběh validace

Vstupem jsou nevalidní polygony mající většinou několik vnitřních ringů, z nichž některé ovšem leží vně polygonu. Ringy polygonů se často protínají navzájem či protínají samy sebe.

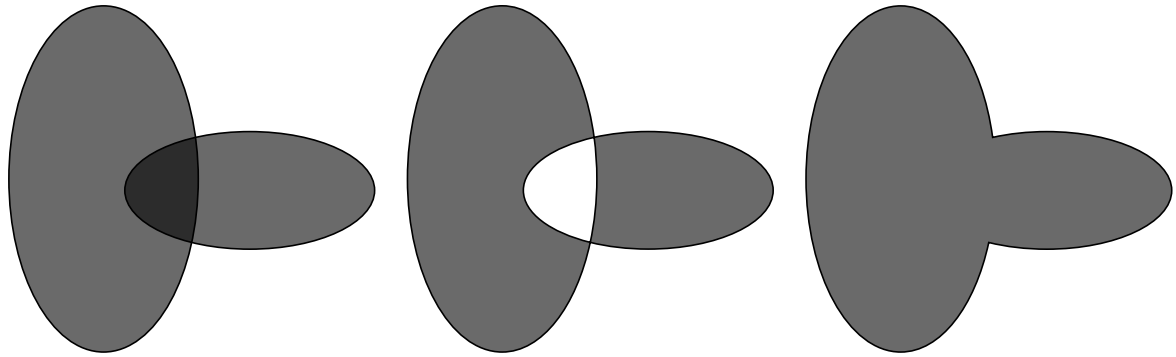
Funkce `ST_DumpRings` slouží k získání jednotlivých ringů ve formě polygonů. Z POLYGONu se dále funkcí `ST_ExteriorRing` získá LINESTRING, případná duplicita ringů je odstraněna pomocí DISTINCT. Dochází-li v rámci LINESTRINGu k *self-intersection*, `ST_Union` spočítá průsečíky a vznikne MULTILINESTRING. Z každého MULTILINESTRINGu funkce `ST_BuildArea` vytvoří POLYGON, případně MULTIPOLYGON. Tyto (MULTI)POLYGONy dále seskupíme podle sloupce `gid`, tedy podle příslušnosti k danému polygonu lesa. Voláme agregační funkci `ST_Collect`, která v tomto případě vrací GEOMETRYCOLLECTION, z níž `ST_BuildArea` vytvoří konečný (MULTI)POLYGON.

Je třeba uvést, jak se zachová funkce `ST_BuildArea`, volaná s parametrem GEOMETRYCOLLECTION, když obsahuje (MULTI)POLYGONy, oproti tomu, jak by se zachovala funkce `ST_Union`. Zatímco `ST_Union` při průniku dvou polygonů tento průnik zachová, funkce `ST_BuildArea` na místě průniku vytvoří díru či prázdný prostor, viz obrázek č. 2.

3.2 Kontrola validace

Po zvalidování je třeba zkontrolovat, zda se u polygonů nezměnila geometrie ve smyslu tvaru. Tabulku s porovnáním polygonů před validací a po validaci naleznete v příloze B. Tabulka byla vygenerována příkazem:

```
SELECT n.gid,
ST_NRings(n.way) AS r_neval,
ST_NRings(v.way) AS r_val,
```



(a) Vstup (b) ST_BuildArea (c) ST_Union

Obrázek 2: Porovnání funkcí

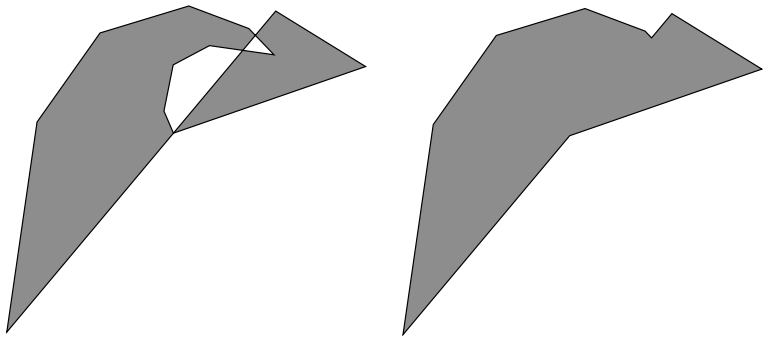
```
ST_NRings(n.way) - ST_NRings(v.way) AS roz,
ROUND( ( (ST_Area(n.way) - ST_Area(v.way)
               )  / ST_Area(n.way) )*100) AS pl_proc,
ST_GeometryType(v.way) AS type,
ST_IsValid(n.way) AS b,
ST_IsValid(v.way) AS a,
ST_IsValidReason(n.way) AS reason

FROM f10.lesni_porosty AS n
JOIN f10.valid AS v
ON v.gid = n.gid

ORDER BY n.gid;
```

Tento výpis jsme používali pro ověření toho, že při validaci nedošlo k nežádoucím změnám dat. Ve sloupci `pl_proc` je změna plochy (výměry) v procentech. Funkce `ST_Area` však špatně počítá plochu nevalidních polygonů, např. započítá jen část polygonu či interpretuje část plochy jako díru a odečte ji. Výměra tedy nemohla být pro ověření správnosti použita a je tedy uvedena pouze formálně.

Vzhledem ke struktuře vstupních dat jsme se při ověřování řídili především sloupci s údaji o počtu ringů. Počet ringů by měl zůstat zachován u polygonů, které mají vnější ringy (především chyba *hole lies outside shell*), nebo zvětšen u polygonů, kde dochází k *self-intersection*. Ke zmenšení počtu ringů by mělo dojít pouze tehdy, když jeden polygon obsahuje některý z ringů vícekrát, tedy když nastává chyba *duplicate rings*. Zjistili jsme, že volání `ST_BuildArea(geometry)`, kde `geometry` jsou MULTILINESTRINGy v některých případech z nám neznámých důvodů špatně vyhodnotí ringy, které jsou obsaženy vícekrát. Proto jsme do dotazu vybírajícího jednotlivé ringy vložili DISTINCT. Tím se každý ring vybere jen jednou a vyhodnotí se správně. V šesti případech došlo ke zmenšení počtu ringů o jeden či dva, tyto polygony jsme vizuálně zkontrolovali, zda jsou v pořádku. Namátkou



Obrázek 3: Polygon č. 312 nevalidní (vlevo) a zvalidovaný (vpravo)

jsme ještě vyhledali několik polygonů, které se nám zdály z různých důvodů podezřelé (například při validaci vznikl POLYGON a ne MULTIPOLYGON, který by měl při validaci převážně vznikat). Takto jsme narazili na polygon, který se při validaci změnil – zanikly díry a zvětšila se tak celková plocha. Ukázalo se, že tento polygon (`gid` 312) je natolik atypický, že ho náš postup nemohl zvalidovat správně. *QGis* jej zobrazil, jak je vidět na obrázku č. 3 (obrázek je zjednodušen). Plocha, která tímto přibyla, je vůči celkové ploše polygonu malá. Neumíme zjistit, u kolika dalších polygonů tento nebo podobný problém nastal. Zkoušeli jsme tedy problém odhalit vizuálně v programu *QGis*, avšak to lze pouze v omezené míře. Žádný takový problém jsme už neodhalili. Pokud k tomu tedy ještě někde došlo, rozhodně nejde o rozsáhlou plochu. Otázkou stále zůstává, jak se mají nevalidní data interpretovat a zda zobrazení v *QGisu* odpovídá úmyslu pořizovatele dat (zdroj dat nebyl uveden).

3.3 Update lesů

Bylo nutné nahradit nevalidní geometrie v tabulce `lesni_porosty` geometrií validní. Navíc bylo potřeba všechny polygony přetypovat na typ MULTIPOLYGON, což znamenalo upravit i omezení geometrie. To jsme provedli touto sérií příkazů:

```
ALTER TABLE f10.lesni_porosty DROP CONSTRAINT enforce_geotype_way;

UPDATE f10.lesni_porosty
   SET way = f10.valid.way
  FROM f10.valid
 WHERE f10.lesni_porosty.gid = f10.valid.gid;

UPDATE f10.lesni_porosty
   SET way = ST_Multi(way);

SELECT Populate_geometry_columns('f10.lesni_porosty'::regclass);
```

3.4 Překryt lesů

Vrstva lesů byla tedy zvalidována, ale to pouze z pohledu *PostGISu*. Lesy se totiž na mnoha místech překrývají, což by samozřejmě neměly. Překryty jsou často velmi malé a dochází k nim při okrajích, nicméně mnoho polygonů se překrývá větší částí své plochy, či jsou dokonce totožné.

Tento problém jsme se rozhodli řešit tak, že pro každý polygon zjistíme, kde se překrývá s polygony s vyšším číslem id (aby se překrytí nevyskytovalo vícekrát). Toto překrytí poté od něj odečteme a tím získáme nový polygon pro následný update. S vyhledáním překrývajících se polygonů nám pomohl Ing. Martin Landa (jeho příkaz viz příloha C). Počet překrývajících se polygonů byl 1336.

Diskutabilní je samozřejmě výběr toho polygonu, od kterého se bude odečítat. Lze vybírat na základě výměry, to však selhává u multipolygonů, kde by bylo nutné pracovat s jednotlivými samostatnými polygony. Dále lze vybírat na základě původu dat (důvěryhodnosti zdroje). O původu jsme však neměli informace. A v neposlední řadě by také šlo vybírat podle hodnot ostatních atributů, ty však většinou měli hodnotu NULL. Závěrem je, že při rozdílných datech lze jen obtížně najít řešení, to by muselo být navíc velmi komplexní. Vzhledem k tomu, že jsme o datech neměli moc informací, rozhodli jsme se zvolit řešení relativně jednodušší. K našemu způsobu řešení je nutné poznamenat ještě jednu věc. Přikazy jsou sestaveny tak, aby tam, kde jednou les zakreslen je, les zůstal. To tedy znamená, že když je nějaký les uložen duplicitně, plně se tedy překrývají dva polygony a zároveň jeden z nich obsahuje díry a druhý na stejném místě nikoliv, díry ve výsledném polygonu zmizí. Stručně řečeno, ve výsledku je plocha lesa všude tam, kde byla alespoň v jednom ze vstupních polygonů. Tomu odpovídá i kontrola správnosti odstranění překrytů (viz 3.6).

```
CREATE TABLE f10.lesy_intersect AS
 SELECT g1.gid AS g1_gid,
        ST_Union(ST_Buffer(ST_Intersection(g1.way, g2.way),0.0))
             AS inter
   FROM f10.lesni_porosty g1, f10.lesni_porosty g2
  WHERE g1.gid < g2.gid
        AND ST_Intersects(g1.way, g2.way)
        AND NOT ST_Touches(g1.way, g2.way)
        GROUP BY g1.gid;
```

Při řešení tohoto problému jsme objevili postupně i několik dalších komplikací. Funkce `ST_Intersection` v některých specifických případech vrací geometrii typu GEOMETRYCOLLECTION, viz obrázek č. 5. Proto je použita funkce `ST_Buffer(geometry, 0.0)`, která vybere pouze polygony a zbytek zahodí. Další problém představovala funkce `ST_Union`, která v průběhu výpočtu nahlásila:

```
NOTICE:  TopologyException: found non-noded intersection between ...
ERROR:  Union returned a NULL geometry.
```

To nás překvapilo, vzhledem k tomu, že tato funkce by si měla poradit se vzájemně se překrývajícími polygony. Tento problém by se mohl napravit, dle obecných doporučení, pomocí funkce `ST_SnapToGrid`, čímž by se uzly přesunuly na jiná místa a problém by mohl zmizet. Použití přichycení na mřížku s sebou jednak nese riziko ztráty přesnosti dat při špatné volbě velikosti mřížky, možné znevalidnění dat a také problémy spojené s tím, kdy funkci zavolat. To totiž ovlivní, v jakém vztahu bude výsledek překrytí a původní polygon, od kterého budeme odečítat, a také okolní polygony. Může se tudíž stát (a také se nám stalo), že výsledek rozdílu polygonu a intersektu bude obsahovat nějaké zbytkové plochy vzniklé v důsledku toho, že něco bylo a něco nebylo uchyceno na mřížku. Také se může stát, že se výsledný polygon bude stále protínat s nějakým sousedním. Tomu všemu jsme chtěli zabránit tím, že celou vrstvu lesů přichytíme na grid. Tím se ovšem problém pouze přesunul na jiné místo, neboť, jak již bylo uvedeno výše, uzly polygonů se tím posunou, jedna situace se tedy může změnit k lepšímu, jiná však k horšímu. Zjistili jsme, že problém s *non-noded intersection* nastává pouze u jednoho polygonu a to polygonu s `osm_id` -422770 (náš `gid` 40686). Byl proto ze základní množiny vyjmut a dále byl řešen samostatně. Do klausule WHERE u příkazu pro vytváření intersektů byla přidána podmínka `AND g1.gid != 40686`.

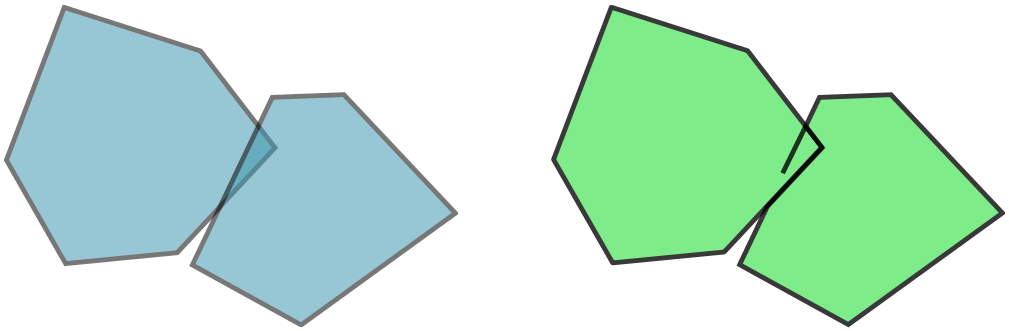
Pro získání překrytu polygonu s `osm_id` -422770 byl použit následující příkaz:

```
INSERT INTO f10.lesy_intersect
 SELECT g1.gid AS g1_gid,
        ST_Union(ST_Buffer(ST_Intersection(
                           ST_SnapToGrid(g1.way, 0.001), g2.way
                           ), 0.0)) AS inter
  FROM f10.lesni_porosty g1, f10.lesni_porosty g2
 WHERE g1.gid < g2.gid
       AND ST_Intersects(g1.way, g2.way)
       AND NOT ST_Touches(g1.way, g2.way)
       AND g1.gid = 40686
 GROUP BY g1.gid;
```

Použití funkce `ST_SnapToGrid` má za následek změnu situace a k vyvolání výjimky již v tomto případě nedojde.

Následujícím krokem je výpočet odečtení překrytu od základního polygonu. Tím se získá polygon, který se už nepřekrývá s okolními. Některé z takto vytvořených polygonů jsou nevalidní, patrně po použití funkce `ST_Difference`. Dochází především k *self–intersection* a ve dvou případech došlo také k *too few points in geometry component*. Celkem bylo 76 nevalidních polygonů. Další problém i u validních polygonů byl ten, že došlo ke vzniku zbytků po odečtení. Jednalo se o malé výběžky s nulovou plochou (viz obr. 4). Po několika neúspěšných pokusech jsme dospěli k řešení pomocí `ST_Buffer`, kde jsme jako velikost bufferu zvolili malou (vzhledem k očekávané přesnosti dat) avšak nenulovou hodnotu (konkrétně 0.01, při 0.1 docházelo k vyvolání výjimky).

```
CREATE TABLE f10.lesy_no_intersect_buffer AS
```



(a) Výchozí situace (b) Po odečtení

Obrázek 4: Vznik výběžků s nulovou plochou

```
SELECT A.gid,
       ST_Multi(ST_Difference(
                      A.way, ST_Buffer(B.intersect, 0.01)
                      )) AS way
  FROM f10.lesni_porosty AS A
  JOIN f10.lesy_intersect AS B
       ON A.gid = B.g1_gid
```

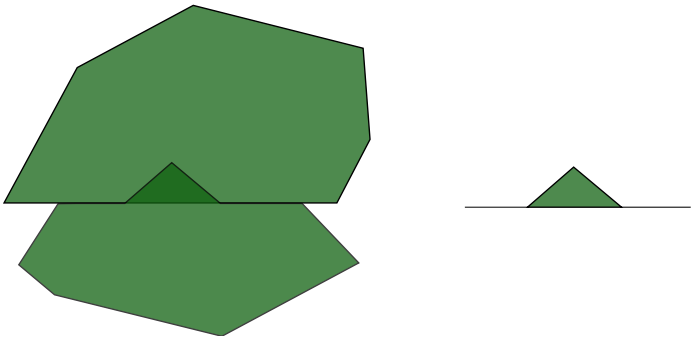
Odečítaný překryt je tedy větší než skutečný. Díky tomu nedojde ke vzniku nevalidních polygonů a polygonů s malými výběžky (viz výše). Pokud dojde k tomu, že překryt je větší než polygon, funkce `ST_Difference` vrací `GEOMETRYCOLLECTION EMPTY`, což je validní geometrie s nulovou výměrou. Navrácené geometrie jsou tedy validní a kontrolu lze provést příkazem:

```
SELECT gid,
       ST_GeometryType(way),
       ST_Area(way),
       ST_NPoints(way),
       ST_IsValidReason(way)
 FROM f10.lesy_no_intersect_buffer
WHERE ST_GeometryType(way) <> 'ST_MultiPolygon'
      OR ST_Area(way) < 10
      OR NOT ST_IsValid(way);
```

Příkazem se vyberou i geometrie, které nejsou MULTIPOLYGONy a geometrie s podezřele malou výměrou, tyto všechny je třeba zkontrolovat, než se bude pokračovat s opravou.

3.5 Další update lesů

Následujícím krokem byla aktualizace tabulky `lesni_porosty`, ze které jsme si nejdříve udělali zálohu, pro pozdější porovnání. Aktualizace proběhla dvěma způsoby. Jeden byl UPDATE polygonů, které se zmenšily. Těch byla většina, konkrétně 1293. Druhým způsobem



(a) Překrývající se polygony

(b) GEOMETRY-COLLECTION

Obrázek 5: Výsledek překrytí u některých polygonů

byl DELETE polygonů, pro které vyšla prázdná geometrie, tedy s nulovou výměrou. Nutno poznamenat, že u následujících příkazů UPDATE a DELETE je nutná pouze podmínka týkající se plochy, ostatní jsou jen pro jistotu.

```
UPDATE f10.lesni_porosty SET way = A.way
  FROM f10.lesy_no_intersect_buffer AS A
 WHERE A.gid = f10.lesni_porosty.gid
       AND ST_GeometryType(A.way) = 'ST_MultiPolygon'
       AND ST_Area(A.way) > 1.0
       AND ST_IsValid(A.way);

DELETE FROM f10.lesni_porosty
 USING f10.lesy_no_intersect_buffer AS A
 WHERE A.gid = f10.lesni_porosty.gid
       AND ST_Area(A.way) <= 1.0;
```

3.6 Kontrola nové vrstvy a další iterace

U obnovené vrstvy `lesni_porosty` jsme kontrolovali validitu (vše nutně validní), celkovou výměru všech lesů a také, zda byly všechny překryty odstraněny. Výměru jsme kontrolovali na základě toho, že celková plocha, která je pokrytá lesem, měla zůstat zachována. Kontrola celkové výměry byla provedena ve třech oblastech (s různým obdélníkem) příkazem:

```
SELECT
(
 SELECT SUM(ST_Area(way))
   FROM f10.lesni_porosty
  WHERE ST_Within(way,
                  ST_SetSRID(ST_MakeBox2D(
                        ST_Point(1500000, 6300000),
```

```
                        ST_Point(1560000, 6360000)
                    ) , 900913) )
) AS new
,
(
 SELECT ST_Area(ST_Union(way))
   FROM f10.lesni_porosty_winter
  WHERE ST_Within(way,
                    ST_SetSRID(ST_MakeBox2D(
                        ST_Point(1500000, 6300000),
                        ST_Point(1560000, 6360000)
                    ) , 900913) )
) AS old
;
```

Tabulka 1: Porovnání výměr

má býti	jest	rozdíl	relativní v %
660708927	660708306	621	0,000094
876496415	876496402	13	0,000001
1408442061	1408440929	1132	0,000080

Výsledek kontroly celkové výměry je velmi uspokojivý, viz tabulka 1. Co se týče kontroly překrytů, k našemu velkému překvapení bylo následujícím příkazem nalezeno 15 polygonů, které se překrývaly s jedním nebo více polygony.

```
 SELECT COUNT(g1.gid)
   FROM f10.lesni_porosty g1, f10.lesni_porosty g2
  WHERE g1.gid < g2.gid
        AND ST_Intersects(g1.way, g2.way)
        AND NOT ST_Touches(g1.way, g2.way)
        GROUP BY g1.gid;
```

Aplikovali jsme tedy znovu postup uvedený v 3.4 a 3.5. Další kontrolou byly nalezeny 4 polygony s překrytem. Znovu jsme tedy zopakovali celý postup. Potom jsme opět provedli kontrolu, ze které vyšly 3 polygony, které se překrývaly. Jejich překryty však měly nulovou výměru (byly to `GEOMETRYCOLLECTION EMPTY`). Celkem byly tedy třeba 3 iterace k odstranění všech překrytů (intersektů). Proč nejsou všechny překryty nalezeny a odstraněny již v prvním kroku, nám není známé.

4 Dotazy

4.1 Kolik bunkrů je v lese?

SQL příkaz:

```
SELECT COUNT(*)
  FROM f10.bunkry AS bunkry
  JOIN f10.lesni_porosty AS lesy
    ON ST_Within(bunkry.way, lesy.way);
```

Odpověď:

```
count
-------
   127
(1 row)
```

4.2 Který újezd má největší rozlohu a kolik?

SQL příkaz:

```
SELECT name, ROUND(ST_Area(way)/1e6) AS area
  FROM f10.ujezdy AS ujezdy
 ORDER BY area DESC
 LIMIT 1;
```

Odpověď:

```
          name           | area
-------------------------+------
 Vojenský újezd Hradiště |  809
(1 row)
```

Poznámka: Skutečná rozloha je 332 km^2. Rozdíl způsoben nezavedením opravy ze zobrazení (platí pro všechny otázky na plochu).

4.3 Kolik procent vojenského újezdu Hradiště pokrývá les?

SQL příkaz:

```
SELECT ROUND(((lesy_area/area)*100)::NUMERIC, 2)
FROM (
 SELECT SUM(ST_Area(ST_Intersection(ujezdy.way, lesy.way))) AS lesy_area,
        ST_Area(ujezdy.way) AS area
   FROM f10.ujezdy AS ujezdy
```

```
   JOIN f10.lesni_porosty AS lesy
     ON ujezdy.name = 'Vojensky ujezd Hradiste'
        AND ST_Intersects(ujezdy.way, lesy.way)
  GROUP BY ujezdy.way
) AS lesy_hrad;
```

Odpověď:

```
round
-------
 16.86
(1 row)
```

4.4 Jaká je vzdálenost mezi nejzápadnějším a nejvýchodnějším bunkrem v ČR?

SQL příkaz:

```
SELECT ROUND(ST_Distance(
       (SELECT way
          FROM f10.bunkry AS bunkry
         ORDER BY ST_X(bunkry.way) DESC
         LIMIT 1),
       (SELECT way
          FROM f10.bunkry
         ORDER BY ST_X(bunkry.way) ASC
         LIMIT 1)
        )/1e3) AS vzdalenost;
```

Odpověď:

```
 vzdalenost
------------
        524
(1 row)
```

Poznámka: Jistě je zde také rozdíl oproti skutečnosti způsobený nezavedením opravy ze zobrazení (platí pro všechny otázky na vzdálenost).

4.5 Který vojenský újezd má největší poměr obvodu ku ploše (poměr uveďte na 5 desetinných míst)?

SQL příkaz:

```
SELECT name, ROUND((ST_Perimeter(way)/ST_Area(way))::NUMERIC,5) AS hodnota
  FROM f10.ujezdy AS ujezdy
 ORDER BY hodnota DESC LIMIT 1;
```

Odpověď:

```
          name          | hodnota
------------------------+---------
 Vojenský újezd Březina | 0.00045
(1 row)
```

4.6 Vypište souřadnice bunkru, který se nachází nejdál od vojenského újezdu Brdy.

SQL příkaz:

```
SELECT ST_X(bunkr.way) AS X,
       ST_Y(bunkr.way) AS Y
  FROM (
    SELECT bunkry.way,ST_Distance(bunkry.way, ujezdy.way) AS vzdalenost
      FROM f10.bunkry AS bunkry
      JOIN f10.ujezdy AS  ujezdy
        ON ujezdy.name = 'Vojenský újezd Brdy'
     ORDER BY vzdalenost DESC LIMIT 1) AS bunkr;
```

Odpověď:

```
        x         |        y
------------------+------------------
 1996460.74801165 | 6437929.75357603
(1 row)
```

4.7 Kolik lesů má celou svou plochu do 10 km od vojenského újezdu Brdy?

SQL příkaz:

```
 SELECT COUNT(*)
   FROM f10.lesni_porosty AS lesy
   JOIN ST_Buffer(
          (SELECT way
             FROM f10.ujezdy
            WHERE name='Vojenský újezd Brdy'),1e4) AS buffer
     ON ST_Within(lesy.way, buffer);
```

Odpověď:

```
 count
-------
   322
(1 row)
```

Poznámky: Příkaz se dlouho se vykonává (josef: 79153,033 ms). Opět problém s jednotkami, tentokrát však už ve znění otázky.

4.8 Jaká je výměra lesa ve vzdálenosti 10 km kolem nejsevernějšího bunkru?

SQL příkaz:

```
SELECT ROUND(SUM(ST_Area(ST_Intersection(lesy.way, buffer))/1e6))
  FROM f10.lesni_porosty AS lesy
  JOIN ST_Buffer(
        (SELECT way
           FROM f10.bunkry
           ORDER BY ST_Y(way) DESC LIMIT 1)
        ,1e4) AS buffer
    ON ST_Intersects(lesy.way, buffer);
```

Odpověď:

```
 round
-------
   270
(1 row)
```

4.9 Jaká je výměra nejmenšího lesa v ČR?

SQL příkaz:

```
SELECT ROUND(ST_Area(lesy.way)) AS area
  FROM f10.lesni_porosty AS lesy
 ORDER BY area ASC LIMIT 1;
```

Odpověď:

```
 area
------
  106
(1 row)
```

4.10 Jaká je celková plocha všech újezdů v ČR?

SQL příkaz:

```
SELECT ROUND(SUM(ST_Area(way))/1e6) AS area
  FROM f10.ujezdy;
```

Odpověď:

```
 area
-------
  3058
(1 row)
```

5 Závěr

Během naší práce jsme se potýkali s nedostatečnou dokumentací *PostGISu*, především pak u hlášení chyb. Nedostatek informací se však týká i standardního chování funkcí. Vzhledem k tomu, že je *PostGIS* otevřený a zadarmo, nemůžeme marnit síly zoufáním nad dokumentací, pokud ke zdokumentování sami nepřispějeme.

Co se týče neočekávaného chování funkcí, vše již bylo uvedeno dříve a proto problémy pouze shrneme.

- Funkce `ST_Union` vyvolávala výjimku *non-noded intersection*, ačkoli by měla intersekt dopočítat.
- Funkce `ST_Difference` je schopna vrátit nevalidní geometrii.
- Funkce `ST_BuildArea` se chová jinak pro LINESTRINGy a jinak pro POLYGONY předané jako GEOMETRYCOLLECTION. Z POLYGONů, které jsou uvnitř jiného vytváří díry, zatímco LINESTRINGy ignoruje (alespoň v jednom případě k tomuto došlo).
- Funkce `ST_Dump` a `ST_DumpRings` byly pro naše potřeby použitelné pouze jako součást vnitřního selektu (jinak byly hlášeny chyby, že jsou volány ve špatném kontextu), na což jsme museli přijít pokusy a omyly.

Výsledkem naší práce je především validní vrstva lesních porostů bez vzájemných překrytů.

A Funkce cleanGeometry

```
-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
--
-- $Id: cleanGeometry.sql 2008-04-24 10:30Z Dr. Horst Duester $
--
-- cleanGeometry - remove self- and ring-selfintersections from
--                 input Polygon geometries
-- http://www.sogis.ch
-- Copyright 2008 SO!GIS Koordination, Kanton Solothurn, Switzerland
-- Version 1.0
-- contact: horst dot duester at bd dot so dot ch
--
-- This is free software; you can redistribute and/or modify it under
-- the terms of the GNU General Public Licence. See the COPYING file.
-- This software is without any warrenty and you use it at your own risk
--
-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


CREATE OR REPLACE FUNCTION cleanGeometry(geometry)
RETURNS geometry AS
$BODY$DECLARE
inGeom ALIAS for $1;
outGeom geometry;
tmpLinestring geometry;

Begin

outGeom := NULL;

-- Clean Process for Polygon
IF (GeometryType(inGeom) = 'POLYGON' OR GeometryType(inGeom) = 'MULTIPOLYGON') THEN

-- Only process if geometry is not valid,
-- otherwise put out without change
if not isValid(inGeom) THEN

  -- create nodes at all self-intersecting lines by union the polygon boundaries
  -- with the startingpoint of the boundary.
  tmpLinestring := st_union(st_multi(st_boundary(inGeom)),st_pointn(boundary(inGeom),1));
outGeom = buildarea(tmpLinestring);
IF (GeometryType(inGeom) = 'MULTIPOLYGON') THEN
RETURN st_multi(outGeom);
ELSE
RETURN outGeom;
END IF;
else
  RETURN inGeom;
END IF;


------------------------------------------------------------------------------
-- Clean Process for LINESTRINGS, self-intersecting parts of linestrings
-- will be divided into multiparts of the mentioned linestring
------------------------------------------------------------------------------
ELSIF (GeometryType(inGeom) = 'LINESTRING') THEN

-- create nodes at all self-intersecting lines by union the linestrings
-- with the startingpoint of the linestring.
outGeom := st_union(st_multi(inGeom),st_pointn(inGeom,1));
RETURN outGeom;
ELSIF (GeometryType(inGeom) = 'MULTILINESTRING') THEN
outGeom := multi(st_union(st_multi(inGeom),st_pointn(inGeom,1)));
RETURN outGeom;
```

```
ELSE
RAISE NOTICE 'The input type % is not supported',GeometryType(inGeom);
RETURN inGeom;
END IF;
End;$BODY$
LANGUAGE 'plpgsql' VOLATILE;
```

B Tabulka porovnání nevalidních a validních polygonů

gid	r_neval	r_val	roz	pl_proc	type	b	a	reason
3	4	4	0	-19	ST_MultiPolygon	f	t	Hole lies outside shell[1.34978e+06 6.48724e+06]
6	2	3	-1	-13	ST_MultiPolygon	f	t	Self-intersection[1.34964e+06 6.50055e+06]
26	1	3	-2	-2	ST_MultiPolygon	f	t	Self-intersection[1.35407e+06 6.48515e+06]
41	1	2	-1	-166	ST_MultiPolygon	f	t	Self-intersection[1.35469e+06 6.48335e+06]
57	1	2	-1	-49	ST_MultiPolygon	f	t	Self-intersection[1.35603e+06 6.48251e+06]
68	1	2	-1	-0	ST_MultiPolygon	f	t	Self-intersection[1.35698e+06 6.50107e+06]
69	1	2	-1	-1	ST_MultiPolygon	f	t	Self-intersection[1.35634e+06 6.4883e+06]
72	1	2	-1	-5	ST_MultiPolygon	f	t	Self-intersection[1.35693e+06 6.48171e+06]
80	9	10	-1	-0	ST_MultiPolygon	f	t	Self-intersection[1.3624e+06 6.46395e+06]
200	1	3	-2	-34	ST_MultiPolygon	f	t	Self-intersection[1.36967e+06 6.47665e+06]
232	3	4	-1	-0	ST_MultiPolygon	f	t	Self-intersection[1.37262e+06 6.4772e+06]
312	1	1	0	-15	ST_Polygon	f	t	Self-intersection[1.40405e+06 6.51491e+06]
320	1	4	-3	-4	ST_MultiPolygon	f	t	Self-intersection[1.38223e+06 6.448e+06]
329	72	74	-2	-74	ST_MultiPolygon	f	t	Self-intersection[1.39713e+06 6.3938e+06]
372	12	12	0	-43	ST_MultiPolygon	f	t	Hole lies outside shell[1.38662e+06 6.49008e+06]
438	1	3	-2	-1	ST_MultiPolygon	f	t	Self-intersection[1.38919e+06 6.41906e+06]
531	1	3	-2	-3	ST_MultiPolygon	f	t	Self-intersection[1.39388e+06 6.38504e+06]
605	3	3	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.39696e+06 6.4537e+06]
724	1	3	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.39856e+06 6.37906e+06]
767	39	42	-3	0	ST_Polygon	f	t	Self-intersection[1.40855e+06 6.47005e+06]
768	6	6	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.4067e+06 6.38336e+06]
798	9	10	-1	-0	ST_MultiPolygon	f	t	Ring Self-intersection[1.41807e+06 6.35481e+06]
799	2	3	-1	0	ST_Polygon	f	t	Ring Self-intersection[1.41807e+06 6.35481e+06]
1856	4	4	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.42165e+06 6.35748e+06]
1861	23	23	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.43393e+06 6.5089e+06]
2129	10	10	0	-40	ST_MultiPolygon	f	t	Hole lies outside shell[1.4307e+06 6.5223e+06]
2258	18	18	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.43211e+06 6.38901e+06]
2332	3	5	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.42807e+06 6.44999e+06]
2366	8	10	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.43681e+06 6.43471e+06]
2513	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.43138e+06 6.47703e+06]
2623	4	4	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.43199e+06 6.44626e+06]
2791	2	3	-1	1	ST_Polygon	f	t	Self-intersection[1.43465e+06 6.34557e+06]
2911	21	22	-1	-0	ST_MultiPolygon	f	t	Self-intersection[1.44701e+06 6.52082e+06]
2945	35	35	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.45084e+06 6.44438e+06]
3044	1	3	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.43892e+06 6.35461e+06]
3062	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.43875e+06 6.39369e+06]
3350	3	3	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.44528e+06 6.40266e+06]
3476	15	17	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.45288e+06 6.42756e+06]
3518	9	9	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.44687e+06 6.38301e+06]
3530	1	3	-2	-2	ST_MultiPolygon	f	t	Self-intersection[1.44723e+06 6.32871e+06]
3917	27	27	0	-0	ST_MultiPolygon	f	t	Hole lies outside shell[1.46753e+06 6.29924e+06]
3994	2	2	0	-90	ST_MultiPolygon	f	t	Hole lies outside shell[1.45901e+06 6.52333e+06]
4343	5	6	-1	-0	ST_Polygon	f	t	Ring Self-intersection[1.47052e+06 6.53713e+06]
4500	4	4	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.46842e+06 6.39319e+06]

4540	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.46514e+06 6.46032e+06]
4643	36	38	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.47572e+06 6.5239e+06]
4655	4	4	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.46917e+06 6.44623e+06]
4896	136	140	-4	-0	ST_MultiPolygon	f	t	Self-intersection[1.50897e+06 6.28372e+06]
5473	6	7	-1	0	ST_Polygon	f	t	Self-intersection[1.48532e+06 6.32059e+06]
5586	1	6	-5	-0	ST_MultiPolygon	f	t	Self-intersection[1.49975e+06 6.55126e+06]
5709	16	16	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.48798e+06 6.30615e+06]
5912	6	6	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.49025e+06 6.45085e+06]
6126	3	5	-2	0	ST_Polygon	f	t	Self-intersection[1.49184e+06 6.31958e+06]
6128	22	24	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.49916e+06 6.29519e+06]
6243	2	4	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.49162e+06 6.50667e+06]
6322	4	4	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.49312e+06 6.38454e+06]
6624	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.49724e+06 6.3933e+06]
6661	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.49636e+06 6.49683e+06]
6843	5	7	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.50152e+06 6.30555e+06]
6961	9	9	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.50785e+06 6.45866e+06]
7277	1	2	-1	-0	ST_MultiPolygon	f	t	Self-intersection[1.50349e+06 6.55591e+06]
7735	28	28	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.51739e+06 6.30116e+06]
8366	8	8	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.52025e+06 6.35145e+06]
8400	1	3	-2	-5	ST_MultiPolygon	f	t	Self-intersection[1.52322e+06 6.57239e+06]
8614	1	3	-2	0	ST_Polygon	f	t	Self-intersection[1.51784e+06 6.31026e+06]
8788	28	31	-3	-0	ST_MultiPolygon	f	t	Self-intersection[1.52271e+06 6.27296e+06]
8884	3	5	-2	-1	ST_MultiPolygon	f	t	Self-intersection[1.52305e+06 6.25516e+06]
9120	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.52238e+06 6.34421e+06]
9171	1	2	-1	-0	ST_MultiPolygon	f	t	Self-intersection[1.52518e+06 6.31349e+06]
9243	4	4	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.52482e+06 6.34436e+06]
10151	4	4	0	-0	ST_MultiPolygon	f	t	Hole lies outside shell[1.53483e+06 6.56637e+06]
10241	1	2	-1	-0	ST_MultiPolygon	f	t	Self-intersection[1.54486e+06 6.2621e+06]
10498	9	9	0	-44	ST_MultiPolygon	f	t	Hole lies outside shell[1.56947e+06 6.21467e+06]
10515	3	3	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.53555e+06 6.28885e+06]
10653	1	2	-1	-37	ST_MultiPolygon	f	t	Self-intersection[1.5358e+06 6.29888e+06]
11191	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.54015e+06 6.52961e+06]
11351	7	7	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.54745e+06 6.3507e+06]
11484	25	27	-2	-8	ST_MultiPolygon	f	t	Self-intersection[1.56028e+06 6.5872e+06]
11969	87	85	2	-1098	ST_MultiPolygon	f	t	Self-intersection[1.56423e+06 6.26019e+06]
12121	9	9	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.54955e+06 6.28938e+06]
12790	3	3	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.55711e+06 6.54426e+06]
13026	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.55662e+06 6.29266e+06]
13209	4	4	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.55774e+06 6.29109e+06]
14114	4	4	0	-2	ST_MultiPolygon	f	t	Hole lies outside shell[1.56594e+06 6.33511e+06]
14401	4	4	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.5746e+06 6.57853e+06]
14498	1	2	-1	-0	ST_MultiPolygon	f	t	Self-intersection[1.57018e+06 6.20615e+06]
14598	11	11	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.57167e+06 6.36691e+06]
14839	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.57087e+06 6.54496e+06]
15001	10	10	0	-4	ST_MultiPolygon	f	t	Hole lies outside shell[1.57204e+06 6.3436e+06]
15153	20	22	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.59104e+06 6.22806e+06]
15436	25	25	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.58411e+06 6.2676e+06]
15701	10	14	-4	0	ST_Polygon	f	t	Self-intersection[1.58907e+06 6.20361e+06]

15708	12	12	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.58025e+06 6.35105e+06]
16675	3	3	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.58667e+06 6.33832e+06]
16785	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.58651e+06 6.38897e+06]
16881	25	25	0	798	ST_MultiPolygon	f	t	Hole lies outside shell[1.6071e+06 6.59631e+06]
17452	2	2	0	-88	ST_MultiPolygon	f	t	Hole lies outside shell[1.59004e+06 6.61937e+06]
17649	2	2	0	-51	ST_MultiPolygon	f	t	Hole lies outside shell[1.59214e+06 6.62873e+06]
17996	3	3	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.59324e+06 6.37643e+06]
18797	10	12	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.60153e+06 6.22397e+06]
18865	1	6	-5	-133	ST_MultiPolygon	f	t	Self-intersection[1.5982e+06 6.20257e+06]
19029	3	3	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.59836e+06 6.53873e+06]
19096	5	4	1	-0	ST_Polygon	f	t	Duplicate Rings[1.61332e+06 6.60388e+06]
20176	3	3	0	-1	ST_MultiPolygon	f	t	Hole lies outside shell[1.60674e+06 6.34534e+06]
20177	3	3	0	-1	ST_MultiPolygon	f	t	Hole lies outside shell[1.60674e+06 6.34534e+06]
20428	2	2	0	-291	ST_MultiPolygon	f	t	Hole lies outside shell[1.6038e+06 6.33643e+06]
20904	1	2	-1	-1273	ST_MultiPolygon	f	t	Self-intersection[1.60825e+06 6.21488e+06]
21291	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.61301e+06 6.50367e+06]
21439	97	104	-7	-0	ST_MultiPolygon	f	t	Self-intersection[1.63357e+06 6.52907e+06]
21555	1	3	-2	-1	ST_MultiPolygon	f	t	Self-intersection[1.61226e+06 6.21388e+06]
21872	1	2	-1	-0	ST_MultiPolygon	f	t	Self-intersection[1.61462e+06 6.26095e+06]
22034	8	8	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.61592e+06 6.28255e+06]
22453	1	2	-1	-32	ST_MultiPolygon	f	t	Self-intersection[1.61716e+06 6.21028e+06]
22644	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.6215e+06 6.55758e+06]
22686	1	5	-4	-1	ST_MultiPolygon	f	t	Self-intersection[1.6377e+06 6.20756e+06]
23261	8	8	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.62374e+06 6.54622e+06]
23930	2	2	0	-287	ST_MultiPolygon	f	t	Hole lies outside shell[1.62647e+06 6.36244e+06]
24193	4	4	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.62959e+06 6.42756e+06]
24844	1	2	-1	-0	ST_MultiPolygon	f	t	Self-intersection[1.63255e+06 6.36892e+06]
24868	9	9	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.63344e+06 6.543e+06]
25632	1	1	0	0	ST_Polygon	f	t	Self-intersection[1.63816e+06 6.3721e+06]
25765	2	3	-1	0	ST_Polygon	f	t	Ring Self-intersection[1.64242e+06 6.34736e+06]
26397	13	17	-4	-0	ST_MultiPolygon	f	t	Self-intersection[1.6528e+06 6.55998e+06]
27086	3	3	0	-0	ST_MultiPolygon	f	t	Hole lies outside shell[1.64833e+06 6.35261e+06]
27580	21	21	0	-17	ST_MultiPolygon	f	t	Hole lies outside shell[1.65146e+06 6.36019e+06]
27708	5	5	0	-155	ST_MultiPolygon	f	t	Hole lies outside shell[1.65451e+06 6.24399e+06]
27810	21	21	0	-5	ST_MultiPolygon	f	t	Hole lies outside shell[1.66679e+06 6.58318e+06]
27981	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.65626e+06 6.59428e+06]
28336	13	13	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.66975e+06 6.31835e+06]
28608	1	4	-3	-1	ST_MultiPolygon	f	t	Self-intersection[1.66235e+06 6.59832e+06]
28656	13	13	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.6628e+06 6.26046e+06]
28823	4	4	0	581	ST_MultiPolygon	f	t	Hole lies outside shell[1.66234e+06 6.33366e+06]
28882	15	16	-1	-7	ST_MultiPolygon	f	t	Self-intersection[1.67326e+06 6.2779e+06]
29556	1	4	-3	-68	ST_MultiPolygon	f	t	Self-intersection[1.66907e+06 6.61945e+06]
30014	1	3	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.67057e+06 6.6193e+06]
30705	7	7	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.67613e+06 6.2896e+06]
31952	37	39	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.69305e+06 6.27797e+06]
32680	2	3	-1	-0	ST_MultiPolygon	f	t	Self-intersection[1.69074e+06 6.58306e+06]
33203	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.69148e+06 6.3112e+06]
33328	3	3	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.69208e+06 6.29461e+06]

34624	8	8	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.70147e+06 6.56867e+06]
34673	3	2	1	0	ST_Polygon	f	t	Self-intersection[1.70032e+06 6.31087e+06]
35568	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.70598e+06 6.55454e+06]
35743	5	5	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.70821e+06 6.55599e+06]
36865	53	59	-6	-1	ST_MultiPolygon	f	t	Self-intersection[1.7305e+06 6.57225e+06]
38069	2	2	0	-3	ST_MultiPolygon	f	t	Self-intersection[1.72154e+06 6.26655e+06]
38561	2	2	0	-18	ST_MultiPolygon	f	t	Self-intersection[1.72637e+06 6.55569e+06]
39428	22	22	0	-0	ST_MultiPolygon	f	t	Hole lies outside shell[1.74263e+06 6.43102e+06]
39824	22	21	1	-0	ST_Polygon	f	t	Duplicate Rings[1.74574e+06 6.53124e+06]
39844	114	114	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.75043e+06 6.55356e+06]
41278	9	9	0	-0	ST_MultiPolygon	f	t	Hole lies outside shell[1.74846e+06 6.45807e+06]
41410	5	5	0	-8	ST_MultiPolygon	f	t	Holes are nested[1.74757e+06 6.33776e+06]
41647	2	4	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.74962e+06 6.46281e+06]
41743	9	11	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.76391e+06 6.25152e+06]
41750	23	25	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.76442e+06 6.56957e+06]
41759	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.75084e+06 6.33635e+06]
43352	14	16	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.78538e+06 6.46755e+06]
43354	3	5	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.76474e+06 6.4787e+06]
43379	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.76537e+06 6.55549e+06]
43458	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.76677e+06 6.30371e+06]
44175	5	5	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.77502e+06 6.42486e+06]
44440	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.77429e+06 6.30002e+06]
45564	6	6	0	-0	ST_MultiPolygon	f	t	Hole lies outside shell[1.78623e+06 6.3575e+06]
45953	6	8	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.788e+06 6.55538e+06]
46583	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.79044e+06 6.40629e+06]
46956	4	4	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.79346e+06 6.4909e+06]
47473	10	10	0	-157	ST_MultiPolygon	f	t	Hole lies outside shell[1.79924e+06 6.34037e+06]
47596	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.8021e+06 6.27481e+06]
48045	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.80108e+06 6.40135e+06]
48266	3	3	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.80267e+06 6.31339e+06]
48706	3	4	-1	-0	ST_MultiPolygon	f	t	Self-intersection[1.80865e+06 6.49437e+06]
48768	1	3	-2	-7	ST_MultiPolygon	f	t	Self-intersection[1.80609e+06 6.5238e+06]
49067	23	23	0	-191	ST_MultiPolygon	f	t	Hole lies outside shell[1.815e+06 6.33064e+06]
49107	3	3	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.80848e+06 6.33344e+06]
49205	17	17	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.81625e+06 6.36316e+06]
49217	2	2	0	-25	ST_MultiPolygon	f	t	Hole lies outside shell[1.80852e+06 6.51641e+06]
49265	5	5	0	-427	ST_MultiPolygon	f	t	Hole lies outside shell[1.81205e+06 6.31772e+06]
49621	2	4	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.81403e+06 6.53151e+06]
49875	1	3	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.81265e+06 6.51075e+06]
50297	4	4	0	-251	ST_MultiPolygon	f	t	Hole lies outside shell[1.81476e+06 6.5572e+06]
50365	3	3	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.81727e+06 6.24846e+06]
50468	1	2	-1	-4	ST_MultiPolygon	f	t	Self-intersection[1.8197e+06 6.26048e+06]
50544	4	4	0	-68	ST_MultiPolygon	f	t	Hole lies outside shell[1.81425e+06 6.30619e+06]
50556	31	31	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.82274e+06 6.35643e+06]
50576	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.81879e+06 6.45821e+06]
50694	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.82026e+06 6.35393e+06]
50698	17	17	0	455	ST_MultiPolygon	f	t	Hole lies outside shell[1.81674e+06 6.51023e+06]
50864	5	5	0	383	ST_MultiPolygon	f	t	Hole lies outside shell[1.8212e+06 6.55682e+06]

50967	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.82242e+06 6.35502e+06]
51256	1	2	-1	-195	ST_MultiPolygon	f	t	Self-intersection[1.82484e+06 6.23061e+06]
51338	1	2	-1	-22	ST_MultiPolygon	f	t	Self-intersection[1.82709e+06 6.25787e+06]
51364	1	2	-1	-9	ST_MultiPolygon	f	t	Self-intersection[1.82801e+06 6.25192e+06]
51503	10	10	0	-3	ST_MultiPolygon	f	t	Hole lies outside shell[1.82872e+06 6.30828e+06]
51604	4	4	0	-3	ST_MultiPolygon	f	t	Hole lies outside shell[1.83265e+06 6.3184e+06]
51815	3	3	0	-7	ST_MultiPolygon	f	t	Hole lies outside shell[1.834e+06 6.30602e+06]
51950	1	3	-2	-3	ST_MultiPolygon	f	t	Self-intersection[1.83492e+06 6.27915e+06]
51951	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.8357e+06 6.36014e+06]
52555	1	2	-1	-0	ST_MultiPolygon	f	t	Self-intersection[1.83873e+06 6.24735e+06]
52918	16	16	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.84779e+06 6.31939e+06]
53634	4	4	0	0	ST_MultiPolygon	f	t	Self-intersection[1.85124e+06 6.31583e+06]
53954	1	3	-2	-1	ST_MultiPolygon	f	t	Self-intersection[1.85551e+06 6.23536e+06]
54249	13	13	0	-671	ST_MultiPolygon	f	t	Hole lies outside shell[1.85068e+06 6.33138e+06]
54364	16	17	-1	0	ST_Polygon	f	t	Self-intersection[1.87326e+06 6.45371e+06]
54472	8	8	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.87163e+06 6.23734e+06]
54679	1	2	-1	-4	ST_MultiPolygon	f	t	Self-intersection[1.87076e+06 6.46461e+06]
54752	8	8	0	-0	ST_MultiPolygon	f	t	Hole lies outside shell[1.87296e+06 6.38675e+06]
55404	1	3	-2	-3	ST_MultiPolygon	f	t	Self-intersection[1.88235e+06 6.2257e+06]
55640	6	8	-2	0	ST_Polygon	f	t	Self-intersection[1.88692e+06 6.4491e+06]
55650	4	4	0	-12	ST_MultiPolygon	f	t	Hole lies outside shell[1.89534e+06 6.39072e+06]
55686	52	56	-4	-335	ST_MultiPolygon	f	t	Self-intersection[1.90328e+06 6.48431e+06]
55796	1	2	-1	-21	ST_MultiPolygon	f	t	Self-intersection[1.88914e+06 6.45537e+06]
56078	28	30	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.90866e+06 6.4835e+06]
56167	16	16	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.91059e+06 6.4314e+06]
56190	1	2	-1	-0	ST_MultiPolygon	f	t	Self-intersection[1.90119e+06 6.2336e+06]
56251	3	11	-8	-5	ST_MultiPolygon	f	t	Self-intersection[1.90174e+06 6.36519e+06]
56621	36	38	-2	0	ST_MultiPolygon	f	t	Self-intersection[1.92403e+06 6.4653e+06]
56823	15	18	-3	0	ST_Polygon	f	t	Self-intersection[1.92538e+06 6.28882e+06]
56946	3	3	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.91265e+06 6.26938e+06]
56983	8	6	2	199	ST_Polygon	f	t	Self-intersection[1.91119e+06 6.26298e+06]
57122	25	27	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.92715e+06 6.48923e+06]
57229	3	3	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.91719e+06 6.41422e+06]
57636	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.92569e+06 6.33714e+06]
57849	2	4	-2	-1	ST_MultiPolygon	f	t	Self-intersection[1.92972e+06 6.34585e+06]
57936	12	12	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.9352e+06 6.48661e+06]
58276	2	3	-1	0	ST_Polygon	f	t	Self-intersection[1.93751e+06 6.28074e+06]
58316	20	20	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.9464e+06 6.47755e+06]
58344	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.9373e+06 6.25261e+06]
58407	1	2	-1	-0	ST_MultiPolygon	f	t	Self-intersection[1.93809e+06 6.42762e+06]
58599	1	2	-1	0	ST_Polygon	f	t	Self-intersection[1.94417e+06 6.39632e+06]
58668	2	2	0	-51	ST_MultiPolygon	f	t	Hole lies outside shell[1.94358e+06 6.26916e+06]
58685	6	6	0	-91	ST_MultiPolygon	f	t	Hole lies outside shell[1.94464e+06 6.49231e+06]
58948	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.94908e+06 6.4463e+06]
59001	3	4	-1	-0	ST_MultiPolygon	f	t	Self-intersection[1.96405e+06 6.404e+06]
59273	3	5	-2	0	ST_Polygon	f	t	Self-intersection[1.95704e+06 6.30019e+06]
59626	1	9	-8	-2	ST_MultiPolygon	f	t	Self-intersection[1.9643e+06 6.47642e+06]
59635	3	4	-1	0	ST_Polygon	f	t	Self-intersection[1.96291e+06 6.30832e+06]

59714	1	2	-1	-2082	ST_MultiPolygon	f	t	Self-intersection[1.96101e+06 6.45113e+06]
59764	82	88	-6	0	ST_Polygon	f	t	Self-intersection[1.9857e+06 6.34063e+06]
59953	4	6	-2	0	ST_Polygon	f	t	Self-intersection[1.96943e+06 6.30379e+06]
60297	10	10	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.9743e+06 6.39808e+06]
60390	1	2	-1	-0	ST_MultiPolygon	f	t	Self-intersection[1.96988e+06 6.49862e+06]
60407	7	9	-2	0	ST_Polygon	f	t	Self-intersection[1.97638e+06 6.29969e+06]
60442	5	7	-2	0	ST_Polygon	f	t	Self-intersection[1.97219e+06 6.30462e+06]
60526	30	35	-5	0	ST_Polygon	f	t	Self-intersection[1.98336e+06 6.34064e+06]
60544	1	2	-1	-4	ST_MultiPolygon	f	t	Self-intersection[1.9722e+06 6.48155e+06]
60556	3	3	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.97274e+06 6.42396e+06]
60561	5	5	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.97394e+06 6.48659e+06]
60566	4	5	-1	0	ST_Polygon	f	t	Self-intersection[1.97833e+06 6.41297e+06]
60720	1	2	-1	-1	ST_MultiPolygon	f	t	Self-intersection[1.97434e+06 6.29711e+06]
61050	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.98312e+06 6.39215e+06]
61102	78	78	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.99579e+06 6.33672e+06]
61122	12	12	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.98369e+06 6.2978e+06]
61230	1	2	-1	-2	ST_MultiPolygon	f	t	Self-intersection[1.98305e+06 6.43514e+06]
61360	12	13	-1	0	ST_Polygon	f	t	Self-intersection[1.98886e+06 6.31604e+06]
61488	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.98813e+06 6.40008e+06]
61655	1	3	-2	-0	ST_MultiPolygon	f	t	Self-intersection[1.98907e+06 6.36688e+06]
61656	3	3	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.98943e+06 6.39537e+06]
62026	17	17	0	-0	ST_MultiPolygon	f	t	Self-intersection[2.00189e+06 6.41577e+06]
62157	24	24	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.99804e+06 6.29164e+06]
62229	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[1.99764e+06 6.35709e+06]
62302	38	38	0	-0	ST_MultiPolygon	f	t	Self-intersection[2.00752e+06 6.34324e+06]
62471	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[2.00145e+06 6.31609e+06]
62589	1	3	-2	-491	ST_MultiPolygon	f	t	Self-intersection[2.00306e+06 6.40567e+06]
62636	84	84	0	-0	ST_MultiPolygon	f	t	Self-intersection[2.01588e+06 6.33678e+06]
62645	29	29	0	-0	ST_MultiPolygon	f	t	Self-intersection[2.005e+06 6.32657e+06]
62833	10	10	0	-0	ST_MultiPolygon	f	t	Self-intersection[2.01001e+06 6.3497e+06]
63006	1	2	-1	-11	ST_MultiPolygon	f	t	Self-intersection[2.00984e+06 6.32251e+06]
63526	4	3	1	0	ST_Polygon	f	t	Self-intersection[2.0175e+06 6.33538e+06]
63538	18	18	0	-0	ST_MultiPolygon	f	t	Self-intersection[2.01838e+06 6.34619e+06]
63539	1	3	-2	-6	ST_MultiPolygon	f	t	Self-intersection[2.01612e+06 6.41913e+06]
63580	28	28	0	-0	ST_MultiPolygon	f	t	Self-intersection[2.01721e+06 6.33431e+06]
63768	85	85	0	-0	ST_MultiPolygon	f	t	Self-intersection[2.02954e+06 6.34372e+06]
63969	35	35	0	-0	ST_MultiPolygon	f	t	Self-intersection[2.0301e+06 6.35362e+06]
63996	1	2	-1	-2	ST_MultiPolygon	f	t	Self-intersection[2.02398e+06 6.42222e+06]
64027	5	5	0	-0	ST_MultiPolygon	f	t	Self-intersection[2.02392e+06 6.33009e+06]
64317	2	4	-2	-0	ST_MultiPolygon	f	t	Self-intersection[2.02784e+06 6.37607e+06]
64384	102	102	0	-0	ST_MultiPolygon	f	t	Self-intersection[2.04836e+06 6.3395e+06]
64812	1	5	-4	-1	ST_MultiPolygon	f	t	Self-intersection[2.05416e+06 6.33614e+06]
64951	1	2	-1	-0	ST_MultiPolygon	f	t	Self-intersection[2.04053e+06 6.41301e+06]
65220	98	98	0	-6	ST_MultiPolygon	f	t	Hole lies outside shell[2.05504e+06 6.34183e+06]
65936	1	2	-1	5	ST_Polygon	f	t	Self-intersection[2.05672e+06 6.3826e+06]
66288	2	2	0	-0	ST_MultiPolygon	f	t	Self-intersection[2.06963e+06 6.4083e+06]
66454	30	33	-3	0	ST_Polygon	f	t	Self-intersection[2.07639e+06 6.3624e+06]
66516	1	3	-2	-1	ST_MultiPolygon	f	t	Self-intersection[2.06827e+06 6.42986e+06]

```
 67044 |       1 |     3 |  -2 |       0 | ST_Polygon      | f | t | Self-intersection[2.08657e+06 6.38492e+06]
 67079 |      34 |    39 |  -5 |      -0 | ST_MultiPolygon | f | t | Self-intersection[2.09931e+06 6.36814e+06]
 67120 |      86 |    86 |   0 |     234 | ST_MultiPolygon | f | t | Hole lies outside shell[2.08758e+06 6.38502e+06]
 67129 |       1 |     2 |  -1 |      -1 | ST_MultiPolygon | f | t | Self-intersection[2.09688e+06 6.37222e+06]
 67142 |       1 |     2 |  -1 |     -23 | ST_MultiPolygon | f | t | Self-intersection[2.09912e+06 6.3688e+06]
(289 rows)
```

C Příkaz Ing. Martina Landy

```
CREATE TABLE ilesy AS
SELECT
      COUNT(DISTINCT g1.osm_id) AS count1,
      COUNT(DISTINCT g2.osm_id) AS count2,
      array_accum(DISTINCT g1.osm_id) AS array_accum1,
      array_accum(DISTINCT g2.osm_id) AS array_accum2,
      st_collect(DISTINCT g1.way) AS st_collect1,
      st_collect(DISTINCT g2.way) AS st_collect2,
      st_intersection(g1.way, g2.way),
      st_area(st_intersection(g1.way, g2.way))
FROM lesy AS g1,
     lesy AS g2
WHERE g1.osm_id < g2.osm_id
      AND st_intersects(g1.way, g2.way)    -- jen ty co se protinaji
    --AND st_isvalid(g1.way) AND st_isvalid(g2.way)
GROUP BY st_intersection(g1.way, g2.way)
ORDER BY COUNT(DISTINCT g1.osm_id),
          st_area(st_intersection(g1.way, g2.way));
```