Slovenská technická univerzita
Fakulta informatiky a informačných technológii
Ilkovičova 3, 812 19 Bratislava

Pokročilé databázové technológie

Priestorové databázy – semestrálny projekt

Bc. Jozef Pazúrik

Cvičiaci: Ing. Miroslav Rác

Študijný odbor: Inteligentné softvérové systémy
Ročník: 2. Ing.
Akademický rok: 2018/2019

Zadanie úlohy

Úlohou zadania bolo vytvoriť mapovú aplikáciu ktorá umožňuje používateľovi vyhľadávať a zobrazovať lokalizačné dáta nad mapou. V tomto riešení aplikácia ponúka vyhľadávanie okolitých parkovacích miest podľa viacerých kritérií. Používateľ si môže zobraziť buď N najbližších parkovísk, či všetky parkoviská v danom rozsahu od konkrétneho bodu, či v danom okrese. Parkoviská sú navyše farebne rozlíšené podľa uvedeného prístupu, pričom privátne parkoviská sú zobrazené červenou farbou, verejné zelenou a ostatné modrou.

Použité technológie

Základ projektu tvorí Postgres databáza v.10 s rozšírením pre priestorové dáta – PostGis. Dáta pochádzajú z open source projektu https://www.openstreetmap.org, pre región západného Slovenska.

Serverovú časť tvorí REST API vytvorené v jazyku php vo frameworku Lumen. API poskytuje webové rozhranie na dopyty nad databázou ktoré vracia odpovede v podobe GeoJSON formátu. Frontendovú časť tvorí jednoduchá stránka v JavaScripte s využitím jQuery a doplnkov ako Leaflet ktorý pre zobrazenie mapy používa dáta zo služby MapBox.

Spracovanie dát

Dáta stiahnuté zo stránky https://www.openstreetmap.org bolo potrebné najprv importovať do databázy a spracovať pre ďalšie použitie. Pre import dát poslúžil nástroj Osm2pgsql, dostupný na adrese https://wiki.openstreetmap.org/wiki/Osm2pgsql. Následne bolo potrebné konvertovať typ geometrie na štandardnú priestorovú projekciu WGS 84 resp. EPSG:4326. Neskôr bol nad touto geometriou vytvorený priestorový index pre rýchlejšie vyhľadávanie dopytov.

Na zobrazenie tepelnej mapy dopravných hodnôt v okolí, bol použitý verejne dostupný dataset zo stránky kaggle: https://www.kaggle.com/silicon99/dft-accident-data#Accidents0515.csv

Opis API

REST API pozostáva z nasledovných adries:

Cesta	**parametre**	**popis**
/postgis_data/nearest	location, limit	Vracia N najbližších parkovísk v okolí daného bodu
/postgis_data/radius	Location, distance	Vracia všetky prakoviská, ktoré sa nachádzajú v rámci danej vzdialenosti od konkréteho bodu. Vzdialenosť sa údáva v metroch.

/postgis_data/region	osm_id	Parameter osm_id predstavuje id konkréteho polygónu okresu v databáze, v rámci, ktorého sa vyhľadávajú parkoviská
/postgis_data/get_regions		Vracia zoznam všetkých okresov spolu s atribútom osm_id v celej databáze
/postgis_data/get_accidents		Vracia súradnice dopravných nehôd spolu s počtom úmrtí v okolí 30 km od stredu mapy
postgis_data/save_parking		Vracia parkoviská s najmenším počtom nehôd v okolí 50 m okolo parkoviska.

Ukážky použitých dopytov nad databázov:

Úprava tabuľky a zmena geometrie stĺpca:

```
alter table planet_osm_polygon ADD COLUMN geom_data geometry(Geometry, 4326);

UPDATE planet_osm_polygon set geom_data = ST_Transform(way, 4326);
```

Vytvorenie indexu:

```
CREATE INDEX planet_osm_polygon_geom_data_index ON public.planet_osm_polygon USING gist(geom_data);
```

Všetky parkoviská v okruhu 1 Km od daného bodu:

```
SELECT p.*, st_asgeojson(p.geom_data) from planet_osm_polygon p
WHERE p.amenity = 'parking'
AND ST_Distance_Sphere(p.geom_data,
ST_SetSRID(ST_GeomFromText('POINT(-0.15206 51.49732)'), 4326)) < 1000;
```

Explain:

```
Gather  (cost=1000.00..238829.71 rows=3369 width=1258)
  Workers Planned: 2
  ->  Parallel Seq Scan on planet_osm_polygon p  (cost=0.00..232580.31 rows=1404 width=1258)
        Filter: ((amenity = 'parking'::text) AND (st_distance_sphere(geom_data, '0101000020E61000002DCF83BBB376C3BFC8D2872EA8BF4940'::geometry) < '1000'::double precision))
```

20 najbližších parkovísk od daného bodu:

```
SELECT p.*, st_asgeojson(p.geom_data) from planet_osm_polygon p
WHERE p.amenity = 'parking'
ORDER BY p.geom_data <-> ST_SetSRID(ST_GeomFromText('POINT(-0.15206 51.49732)'),
4326) LIMIT 20;
```

Explain:

```
Limit  (cost=0.29..502.85 rows=20 width=1266)
  ->  Index Scan using planet_osm_polygon_geom_data_index on
planet_osm_polygon p  (cost=0.29..253998.33 rows=10108 width=1266)
        Order By: (geom_data <->
'0101000020E61000002DCF83BBB376C3BFC8D2872EA8BF4940'::geometry)
        Filter: (amenity = 'parking'::text)
```

Všetky parkoviská v okrese Dartford:

```
WITH okres as (
SELECT * from planet_osm_polygon t where t.boundary='administrative' and
t.admin_level = '8' and t.name like '% Dartford %' limit 1
)
SELECT p.*, st_asgeojson(p.geom_data) from planet_osm_polygon p cross join okres
where (ST_WITHIN(p.geom_data, okres.geom_data) and p.amenity = 'parking');
```

Explain:

```
Nested Loop  (cost=52902.17..55139.58 rows=3 width=1258)
  CTE okres
    ->  Limit  (cost=1000.00..52869.61 rows=1 width=1226)
          ->  Gather  (cost=1000.00..52869.61 rows=1 width=1226)
                Workers Planned: 2
                ->  Parallel Seq Scan on planet_osm_polygon t
(cost=0.00..51869.51 rows=1 width=1226)
                      Filter: ((name ~~ '%Dartford%'::text) AND (boundary =
'administrative'::text) AND (admin_level = '8'::text))
  ->  CTE Scan on okres  (cost=0.00..0.02 rows=1 width=32)
  ->  Bitmap Heap Scan on planet_osm_polygon p  (cost=32.55..2262.42 rows=3
width=1226)
        Recheck Cond: (okres.geom_data ~ geom_data)
        Filter: ((amenity = 'parking'::text) AND _st_contains(okres.geom_data,
geom_data))
        ->  Bitmap Index Scan on planet_osm_polygon_geom_data_index
(cost=0.00..32.55 rows=569 width=0)
              Index Cond: (okres.geom_data ~ geom_data)
```

Nakoľko tabuľka s dopravnými nehodami je obrovská – obsahuje viac ako 1,7 milióna záznamov, vytvorili sme nad ňou view pre ľahšie pracovanie.

Vytvorenie pohľadu nad tabuľkou dopravných nehôd v okolí 2 km od stredu:

```
create view accidents as
SELECT * from car_accidents
where ST_DistanceSphere(geom_data, ST_SetSRID(ST_GeomFromText('POINT(-0.15206
51.49732)'), 4326)) <= 2000;
```

10 parkovísk v okruhu 2Km od zadaného bodu, ktoré majú najmenší počet nehôd v okolí 50 metrov:

```
SELECT p.geom_data, p.osm_id, count(c.gid) as pocet from planet_osm_polygon p
LEFT JOIN accidents c on (ST_DistanceSphere(p.geom_data, c.geom_data) < 50)
WHERE p.amenity = 'parking' and ST_DistanceSphere(p.geom_data,
ST_SetSRID(ST_GeomFromText('POINT(-0.15206 51.49732)'), 4326)) <= 2000
group by p.geom_data, p.osm_id order by pocet asc limit 10;
```

Explain:

```
Limit  (cost=563500532.76..563500557.88 rows=10 width=233)
  ->  Result  (cost=563500532.76..563508997.37 rows=3369 width=233)
        ->  Sort  (cost=563500532.76..563500541.18 rows=3369 width=201)
              Sort Key: (count(car_accidents.gid))
              ->  HashAggregate  (cost=563500426.27..563500459.96 rows=3369
width=201)
                    Group Key: p.geom_data, p.osm_id
                    ->  Nested Loop Left Join  (cost=1000.00..558479748.75
rows=669423669 width=201)
                          Join Filter: (_st_distance(geography(p.geom_data),
geography(car_accidents.geom_data), '0'::double precision, false) <
'50'::double precision)
                          ->  Seq Scan on planet_osm_polygon p
(cost=0.00..199912.18 rows=3369 width=193)
                                Filter: ((amenity = 'parking'::text) AND
(_st_distance(geography(geom_data),
'0101000020E61000002DCF83BBB376C3BFC8D2872EA8BF4940'::geography, '0'::double
precision, false) <= '2000'::double precision))
                          ->  Materialize  (cost=1000.00..360010.93
rows=596103 width=40)
                                ->  Gather  (cost=1000.00..352372.42
rows=596103 width=40)
                                      Workers Planned: 2
                                      ->  Parallel Seq Scan on car_accidents
(cost=0.00..291762.12 rows=248376 width=40)
                                            Filter:
(_st_distance(geography(geom_data),
'0101000020E61000002DCF83BBB376C3BFC8D2872EA8BF4940'::geography, '0'::double
precision, false) <= '2000'::double precision)
```

Použitie k-means algoritmu na vytvorenie 2000 clustrov nad tabuľkou dopravných nehôd. Pre každý cluster sa vyberie centroid a jeho početnosť ako podklad pre zobrazenie tepelnej mapy na frontende.

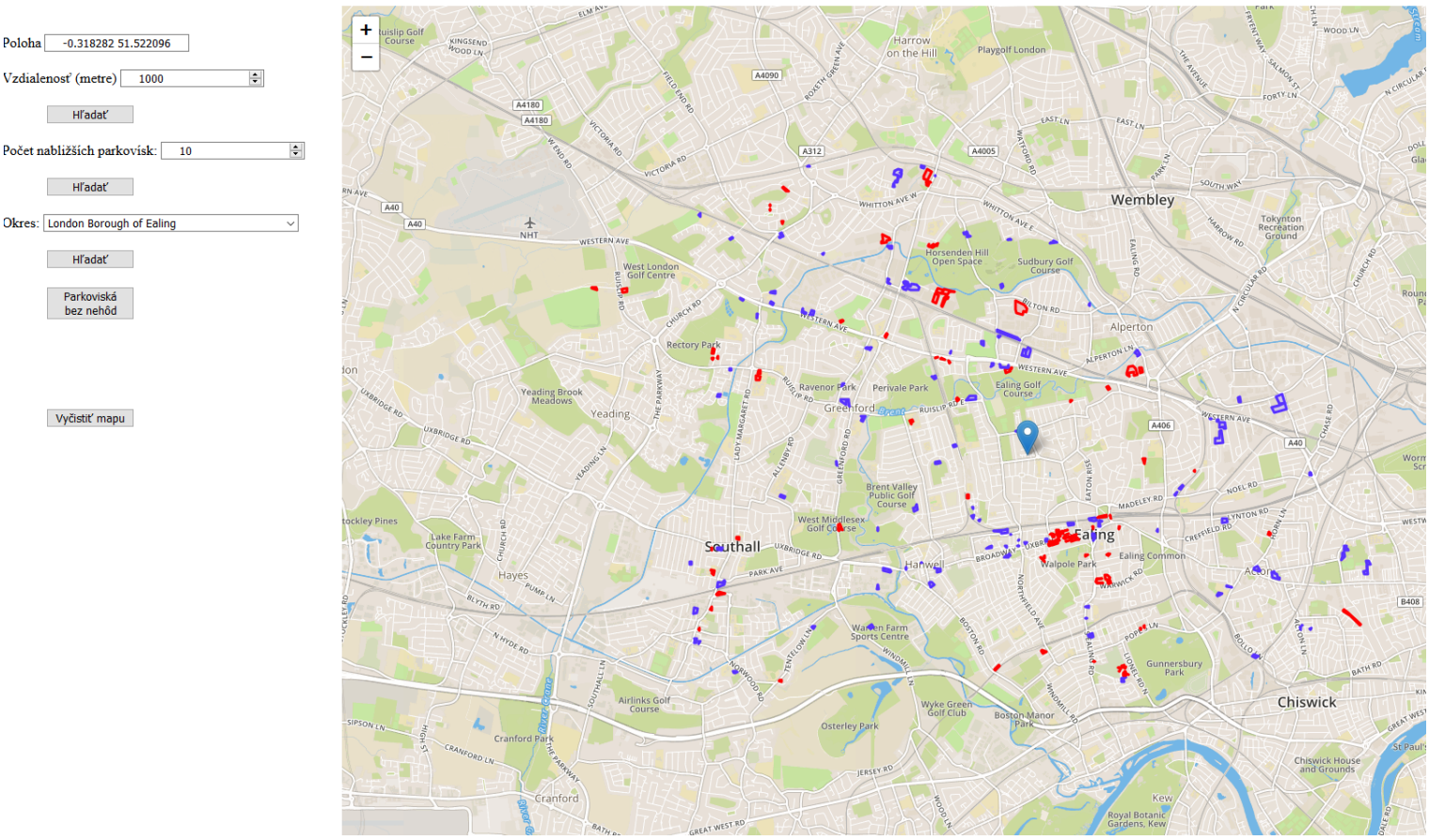
```
SELECT
kmeans_cid,
count(*), st_asgeojson(ST_Centroid(ST_Collect(geom_data))) AS centroid
FROM (
    SELECT
    ST_ClusterKMeans(geom_data, 2000) OVER () kmeans_cid,
    geom_data
    FROM accidents
) kmeans GROUP BY kmeans_cid;
```

Explain:

```
HashAggregate  (cost=586087.97..586590.97 rows=200 width=44)
  Group Key: st_clusterkmeans(car_accidents.geom_data, 2000) OVER (?)
  ->  WindowAgg  (cost=0.00..575656.17 rows=596103 width=36)
        ->  Seq Scan on car_accidents  (cost=0.00..568204.89 rows=596103 width=32)
              Filter: (_st_distance(geography(geom_data), '0101000020E61000002DCF83BBB376C3BFC8D2872EA8BF4940'::geography, '0'::double precision, false) <= '2000'::double precision)
```

Ukážka grafického rozhrania

Červená farba znázorňuje súkromné parkoviská.



Tepelná mapa dopravných nehôd v okolí