Sport Tracker

Projekt z predmetu Pokročilé databázové technológie

Bc. Miroslav Lehotský

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

1 Úvod

V rámci tohto predmetu som sa rozhodol ako projekt realizovať webovú aplikáciu, ktorá najlepšie poslúži športovo aktívnym ľudom. Môže im poslúžiť pre vyhľadávanie im zaujímavých športových objektov s následným zobrazením na mape, pričom samotné vyhľadávanie možno špecifikovať príslušnými dostupnými filtrami v jednotlivých sekciách. Športové objekty možno v aktuálnej verzii zobrazovať v meste San Francisco, táto náležitosť je ale variabilná a závisí len na dátach v pripojenej databáze.

2 Dáta

Veľmi dôležitým aspektom aplikácie sú dáta, na ktorých je aplikácia postavená a s ktorými pracuje. V tejto sekcii je teda opísaný výber vhodných dát a proces ich následného sprístupnenia prostredníctvom databázy.

2.1 Výber dát pre mapu

Ako jeden z prvých krokov, o ktorých som musel rozhodnúť, bolo aké dáta v aplikácii využijem. Ako prvá možnosť sa ponúkal dostupný docker image s existujúcou PostGIS databázou obsahujúcou geo-dáta mesta Bratislava. Kvôli požiadavke na zakomponovanie dodatočného datasetu do aplikácie som musel ale túto možnosť eliminovať, keďže vhodný dataset s geo-dátami v oblasti Bratislavy sa mi nepodarilo nájsť.

Pri rozhodovaní o oblasti, z ktorej použijem dáta som teda prihliadal aj na dostupnosť vhodných datasetov pre danú oblasť. Ako jedna z oblastí, ku ktorej sa mi podarilo nájsť datasety s geo-dátami bolo mesto San Francisco. Rozhodol som sa teda pre využitie tejto oblasti.

2.2 Výber datasetu

Dataset, ktorý by som mohol vhodným spôsobom využiť, som vyberal s ohľadom na typ aplikácie, ktorú som realizoval. Nakoniec som sa rozhodol pre dataset "SF Large Utility Excavation Permits" [1]. Tento dataset obsahuje geo-dáta pre povolenia výkopových prác v meste San Francisco. Celkovo obsahuje 3504 takýchto záznamov. Dataset som stiahol vo formáte CSV, v ktorom zaberal 1.6MB.

2.3 Vytvorenie databázy

1. Stiahnutie PostgreSQL databázy s nainštalovaním PostGIS rozšírenia.
2. Vytvoril som databázu gis a aktivoval pre ňu rozšírenie postgis.
3. Pre zisk dát som použil nástroj OpenStreetMap, v ktorom som nastavil viewport na oblasť môjho záujmu a teda mesta San Francisco, následne som dal dáta z oblasti exportovať a stiahnuť.
4. Stiahnuté dáta z OpenStreetMap boli vo formáte XML, stiahol a nainštaloval som nástroj osm2pgsql, ktorý z XML dát vytvoril príslušné tabuľky v mnou vytvorenej databáze gis.

2.4 Import datasetu

Stiahnutý dataset vo formáte CSV som ešte upravil vymazaním atribútov, ktoré som pre svoju prácu nepotreboval. Zo všetkých atribútov, ktoré obsahovali záznamy datasetu som vybral nasledovné:

- Permit_Number
- Permit_Status
- Permit_Start_Date
- Permit_End_Date
- latitude
- longitude

Takto upravené CSV som skopíroval do novo vytvorenej tabuľky "excavations" vrámci databázy gis. Schéma vytvorenej tabuľky obsahuje vyššie uvedené stĺpce, navyše som pridal jeden stĺpec "way", ktorý je typu geometry a je dynamicky vytvorený z hodnôt latitude a longitude v príslušnom riadku. Týmto som dostal stĺpec konzistentný s geometrickým stĺpcom evidovaným v ostatných tabuľkách .

3 Architektúra aplikácie

Aplikáciu som implementoval v jazyku Java s využitím aplikačného frameworku Spring Boot s embedded Tomcat serverom. Backendová časť využíva PostgreSQL driver pre komunikáciu s databázou, všetku komunikáciu s databázou zabezpečuje trieda implementujúca vzor repozitár. Výsledky poskytované repozitárom sú spracované triedou typu služba, ktorá požadované spracované výstupy poskytuje kontroléru pre jednotlivý REST endpoint, ktorý obdržal HTTP požiadavku.

Frontendová časť je realizovaná prostredníctvom frameworku thymeleaf a v ňom vytvorenej šablóny. Šablóna pozostáva z HTML obohateného o špeciálne výrazy frameworku thymeleaf, štýl šablóny je prispôsobený za pomoci CSS (Bootstrap, Custom), na zobrazenie mapy a objektov na nej je použitá javascriptová knižnica MapBox GL. Pre zobrazenie mapy využívam publikovaný custom štýl vytvorený pomocou MapBox štúdia, na ktorý sa odkazujem v html šablóne s využitím knižnice MapBox GL. Thymeleaf šablóna pristupuje k dynamickému obsahu prostredníctvom špeciálneho dátového objektu model, ktorý napĺňa kontrolér.

4 Scenáre

Aplikácia pozostáva z 3 oddelených scenárov, z ktorých každý má vlastnú sekciu filtrov v ľavom paneli. Každý z týchto scenárov je bližšie opísaný v tejto sekcii.

4.1 Zobrazenie športovísk

Prvý scenár je určený pre zobrazenie športových objektov na mape podľa zadaných preferencií týkajúcich sa polohy a športu, "filterSportFacilities" je REST GET endpoint dedikovaný pre tento scenár. Pre tento scenár možno v ľavom paneli špecifikovať adresu, vzdialenosť a šport. Adresa a vzdialenosť (udávaná v metroch) udávajú polohu a vzdialenosť od tejto polohy, vrámci ktorej budú zobrazené existujúce športové objekty. Pre preklad adresy na súradnice je použité Google API. Selekt, v ktorom sú vylistovné všetky športy, pre ktoré existuje aspoň jedno športovisko v celej evidovanej oblasti, umožňuje ďalej objekty filtrovať aj podľa konkrétneho športu. Tento scenár s prázdnymi hodnotami filtrovacích položiek je použitý aj ako úvodná stránka, zobrazuje teda všetky existujúce športoviská na mape.

Obr. 1. Prvý scenár s použitím filtra polohy aj športu.

Zisk dát pre šport selekt

Aj keď filtrovanie podľa lokácie aj športu je súčasťou jedného a toho istého filtra, ktorý sa odosiela súčasne, som sa rozhodol v selekte zobrazovať vždy všetky športy, bez ohľadu na zmenu dostupnosti jednotlivých športov pri vymedzení oblasti prvou časťou filtra pracujúcou s lokáciou. Selekt listujúci dostupné športy je teda nemenný, vždy zobrazuje všetky športy, pre ktoré existuje aspoň jedno zodpovedajúce športovisko v databáze. Pre optimálny zisk dostupných športov z databázy som vytvoril špeciálnu tabuľku "sports", ktorá obsahuje len jeden stĺpec s názvami športov. Pre vytvorenie tejto tabuľky som použil nasledovný SQL dopyt, bez zbytočného riešenia optimalizácie, z dôvodu, že tento dopyt bude spustený len raz, pre vytvorenie tabuľky:

```
CREATE TABLE sports AS (
  SELECT DISTINCT(UNNEST(STRING_TO_ARRAY(sport, ';'))) AS
      sport
  FROM planet_osm_point
  UNION
  SELECT DISTINCT(UNNEST(STRING_TO_ARRAY(sport, ';')))
  FROM planet_osm_polygon
  ORDER BY sport ASC
)
```

Zisk dát pre mapu

Príklad SQL dopytu, ktorý získava z databázy dáta potrebné pre vykreslenie bodov na mape podľa nastaveného filtra:

```
SELECT osm_id, name, leisure, sport, ST_AsGeoJSON(
    ST_Transform(way, 4326)) AS geo_way
FROM planet_osm_point
WHERE ST_DWithin(ST_Transform(way, 4326)::geography,
    ST_GeomFromGeoJSON('{"type": "Point", "coordinates":
    [-122.4849882, 37.7762817]}')::geography, 1000.0)
AND STRING_TO_ARRAY(sport, ';') @> ARRAY['basketball']::
    text[]
UNION
SELECT osm_id, name, leisure, sport, ST_AsGeoJSON(
    ST_Transform(ST_Centroid(way), 4326)) AS geo_way
FROM planet_osm_polygon
WHERE ST_DWithin(ST_Transform(ST_Centroid(way), 4326)::
    geography, ST_GeomFromGeoJSON('{"type": "Point", "
    coordinates": [-122.4849882, 37.7762817]}')::geography
    , 1000.0)
AND STRING_TO_ARRAY(sport, ';') @> ARRAY['basketball']::
    text[]
```

Pôvodná hodnota **Total Cost** pre hore uvedený SQL dopyt bez vytvorenia indexov bola **44377.78**. Pre optimalizáciu tohto dopytu som vytvoril 4 samostatné indexy po 2 zhodné na 2 tabuľkách:

```
CREATE INDEX polygon_sports ON planet_osm_polygon USING
    GIN(string_to_array(sport, ';'))
CREATE INDEX point_sports ON planet_osm_point USING GIN(
    string_to_array(sport, ';'))
CREATE INDEX polygon_geography ON planet_osm_polygon
    USING GIST((ST_Transform(ST_Centroid(way), 4326)::
    geography))
```

```
CREATE INDEX point_geography ON planet_osm_point USING
    GIST((ST_Transform(way, 4326)::geography))
```

Vďaka týmto indexom bolo možné znížiť hodnotu **Total Cost** pre celý plán na **2297.38**.

4.2 Zobrazenie vonkajších ihrísk

Druhý scenár sa venuje zobrazeniu vonkajších ihrísk. Takisto mu je venovaná samostatná filtrovacia sekcia v ľavom paneli aplikácie. V tejto sekcii je možné špecifikovať typ povrchu ihriska ako aj jeho plochu. Môžme teda zobrazovať vonkajšie ihriská, ktoré vyfiltrujeme podľa požadovaného povrchu. Pre výber povrchov je použitý selektor, ktorý obsahuje všetky povrchy, pre ktoré existuje aspoň jedno príslušné ihrisko. Na mape sú zobrazované kontúry ihrísk, oproti bezrozmerným bodom zobrazeným v prvom scenári.

Obr. 2. Druhý scenár zobrazujúci trávnaté ihriská s rozlohou 1000 - 2000m^2.

Zisk dát pre surface selekt

Rovnako ako vo filtri pre prvý scenár, aj v tomto filtry je použity selekt, ktorý listuje všetky dostupné povrchy, pre ktoré existuje aspoň jedno zodpovedajúce ihrisko. Platí takisto, že tento selekt je nemenný v závislosti od prvej časti filtru, ktorá pracuje s veľkosťou plochy ihrísk. Samostatnú tabuľku pre "surfaces" som vytvoril podobne ako v prvom scenári, tentokrát pre efektívny zisk dostupných povrchov. Pre vytvorenie tejto tabuľky som použil dopyt:

```
CREATE TABLE surfaces AS (
  SELECT DISTINCT(surface) AS surface
  FROM planet_osm_polygon
  WHERE surface IS NOT NULL
  AND leisure = 'pitch'
  ORDER BY surface ASC
)
```

Zisk dát pre mapu

Príklad SQL dopytu, ktorý získava z databázy dáta potrebné pre vykreslenie vonkajších ihrísk na mape podľa nastaveného filtra:

```
SELECT osm_id, name, surface, ST_AsGeoJSON(ST_Transform(
    way, 4326)) AS geo_json
FROM planet_osm_polygon
WHERE leisure = 'pitch'
AND ST_Area(ST_Transform(way, 4326)::geography) >= 1000.0
AND ST_Area(ST_Transform(way, 4326)::geography) <= 2000.0
AND surface = 'grass'
```

V tomto dopyte posledná podmienka na hodnotu stĺpcu "surface" je variabilná časť dopytu. Táto podmienka je pridaná v závislosti od výberu konkrétneho povrchu zo selektu povrchov, ak žiadny konkrétny nebol vybraný a teda zobrazujeme všetky povrchy, potom táto podmienka v dopyte nebude.

Bez vytvorenia akýchkoľvek indexov bola hodnota **Total Cost** pre celý plán hore uvedeného dopytu **37071.36** najmä v dôsledku použitia sekvenčného skenovania použitého na tabuľke "planet_osm_polygon". Pre optimalizáciu som vytvoril jeden multi-column index pre stĺpcec "leisure", "surface" a pre vypočítanú hodnotu plochy polygónu:

```
CREATE INDEX polygon_leisure_area_surface ON
    planet_osm_polygon(leisure, ST_Area(ST_Transform(way,
    4326)::geography), surface)
```

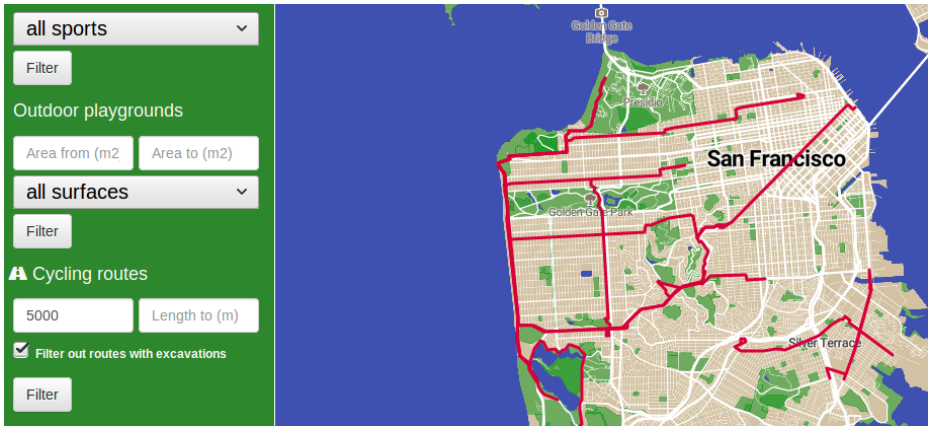
Použitím tohto indexu bol plánovač možný uskutočniť prístup "Index Scan" a hodnota **Total Cost** v dôsledku toho klesla na **10.96**. Vyskúšal som všetky kombinácie poradia jednotlivých stĺpcov pre tento index, pričom sa jednotlivé verzie o moc nelíšili ale hore uvedená verzia sa javila ako optimálna z pohľadu oboch možných verzií dopytu (s aj bez poslednej podmienky na surface).

4.3 Zobrazenie cyklistických ciest

V treťom scenári je realizované zobrazenie cyklistických ciest. Tomuto scenáru je opäť ako v predošlých dvoch prípadoch venovaná filtrovacia sekcia v ľavom paneli aplikácie. V prípade cyklistických ciest sme schopní filtrovať na základe dĺžky cyklistickej cesty, či už dolným alebo horným limitom. Tento scenár využíva externý dataset importovaný do databázy, pre zabezpečenie dodatočnej možnosti filtrovania ciest, v blízkosti ktorých môžu prebiehať výkopové práce. Tento filter je možné zapnúť/vypnúť pomocou checkboxu s legendou, ktorá vysvetluje jeho vplyv.

Zisk dát pre mapu

Príklad SQL dopytu, ktorý získava z databázy dáta potrebné pre vykreslenie cyklistických ciest na mape podľa nastaveného filtra:



Obr. 3. Tretí scenár. Zobrazuje cyklistické cesty, ktoré sú dlhé aspoň 5km a neprebieha v ich blízkosti žiadna výkopová práca.

```
WITH excavations_geo as (
  SELECT way::geography as excavation_geo
  FROM excavations
  WHERE permit_start_date <= now()
  AND permit_end_date >= now()
  AND permit_status = 'ACTIVE'
) , lines_geo AS (
  SELECT osm_id, name, ST_Transform(way, 4326)::geography
      AS line_geo
  FROM planet_osm_line
  WHERE route LIKE 'bicycle'
  AND ST_Length(ST_Transform(way, 4326)::geography) >=
      5000.0
  AND ST_Length(ST_Transform(way, 4326)::geography) <=
      1.7976931348623157E308
)
SELECT osm_id, name, ST_AsGeoJSON(line_geo) AS geo_json
FROM lines_geo
EXCEPT
SELECT DISTINCT osm_id, name, ST_AsGeoJSON(lines_geo.
    line_geo) AS geo_json
FROM excavations_geo CROSS JOIN lines_geo
WHERE ST_DWithin(lines_geo.line_geo, excavations_geo.
    excavation_geo, 5)
```

Časť dopytu od EXCEPT nižšie je variabilná časť, ktorá je prítomná, len ak bol vo filtri zakliknutý zodpovedajúci checkbox pre vynechanie ciest s možnosťou

prebiehajúcich výkopových prác v ich blízkosti. Pri optimalizácii som bral ohľad na optimalizovanie dopytu pre oba tieto varianty. Bez vytvorenia akýchkoľvek indexov bola hodnota **Total Cost** pre celý plán hore uvedeného dopytu **12713.93**. Pre optimalizáciu som vytvoril nasledovné indexy:

```
CREATE INDEX line_route ON planet_osm_line(route)
CREATE INDEX line_length ON planet_osm_line(ST_Length(
    ST_Transform(way, 4326)::geography))
CREATE INDEX excavations_status_start_end ON excavations(
    permit_status, (permit_start_date::date), (
    permit_end_date::date))
```

Vďaka týmto indexom sa zlepšila hodnota **Total Cost** pre daný dopyt na **263.29**.

5 Záver

Vytvorená webová aplikácia s názvom "Sport Tracker" je zameraná najmä pre športujúcu verejnosť v oblasti San Francisca. Umožňuje užívateľovi získať prehľad o dostupných športoviskách, obmedziť vyhľadanie športovísk na základe polohy a športu. Takisto umožňuje rýchly náhľad pre vonkajšie ihriská na základe vybraných povrchov. Cyklisti nájdu aplikácii využitie v podobe vyhľadania cyklistických ciest s možnosťou ich filtrovania na základe dĺžky cesty a na základe možných prebiehajúcich výkopových prác na ceste.

Literatúra

1. Kaggle. *SF Large Utility Excavation Permits.* `https://www.kaggle.com/san-francisco/sf-large-utility-excavation-permits`.