

POLITECHNIKA WARSZAWSKA
Wydział Elektroniki i Technik Informacyjnych

ZAAWANSOWANE PROBLEMY BAZ DANYCH

Optymalizacja danych przestrzennych pod kątem wykorzystania w aplikacjach mobilnych

Sprawozdanie końcowe

Autorzy:

Przemysław BARCIKOWSKI
Maciej SUCHECKI

Prowadzący:

dr inż. Robert BEMBENIK

16 czerwca 2015

1 Treść zadania

Tytuł Optymalizacja danych przestrzennych pod kątem wykorzystania w aplikacjach mobilnych

Opis Załadować przestrzenne dane z konkursu „Dane po Warszawsku” (<https://api.um.warszawa.pl>) do wybranego SZPBD razem z odpowiednimi danymi nieprzestrzennymi pod kątem ich wykorzystania w aplikacjach mobilnych (optymalizacja pod kątem częstego korzystania/częstej aktualizacji danych). Przygotować zapytania przestrzenne pokazujące różne aspekty danych.

2 Opis rozwiązania

W ramach zadania skupiliśmy się na danych o stacjach wypożyczania rowerów „Veturilo” znajdujących się na terenie Warszawy. W celu pobrania danych przy użyciu API wspomnianego w treści zadania został napisany skrypt w języku Python, wykorzystujący biblioteki *requests* (obsługa zapytań REST oraz formatu JSON) oraz *psycopg2* (komunikacja z bazą danych). Skrypt po uruchomieniu próbuje połączyć się z bazą danych. Po uzyskaniu połączenia odpytuje on wspomniane API tak, aby pobrać dane o wszystkich zapisanych w bazie stacjach. Przykładowy fragment skryptu zamieszczony jest poniżej.

```
1 def store_station_in_db(data, cursor, connection):
2     object_id = data["featureMemberProperties"][0]["OBJECTID"]
3     location = data["featureMemberProperties"][0]["LOKALIZACJA"]
4     station_nr = data["featureMemberProperties"][0]["NR_STACJI"]
5     bikes = data["featureMemberProperties"][0]["ROWERY"]
6     stands = data["featureMemberProperties"][0]["STOJAKI"]
7     latitude = data["featureMemberCoordinates"][0]["latitude"]
8     longitude = data["featureMemberCoordinates"][0]["longitude"]
9     coordinates = "POINT(%s %s)" % (latitude, longitude)
10    sql = "INSERT INTO bike_stations_"
11    sql += "(object_id, location, station_nr, bikes, stands, station_coordinates)_"
12    sql += "VALUES(%s, %s, %s, %s, %s, ST_Transform(ST_GeomFromText(%s, 4326), 2059));"
13    cursor.execute(sql, (object_id, location, station_nr, bikes, stands, coordinates))
14    connection.commit()
15
16 # load the bike stations
17 stations_loaded = 0
18 for number in range(MIN_STATION_NR, MAX_STATION_NR + 1):
19     successful = False
20     url = API_URL + "?id=" + MAP_ID + "&apikey=" + API_KEY
21     url += "&filter=" + FILTER_PREFIX + str(number) + FILTER_SUFFIX
22     while not successful:
23         response = requests.get(url)
24         if is_response_valid(response):
25             store_station_in_db(response.json()["result"], cursor, connection)
26             stations_loaded += 1
27             successful = True
28     else:
29         print("ERROR: ", end="")
30         if response.status_code == 200:
31             print(response.json()["result"])
32         else:
33             print(str(response.status_code))
```

Listing 1: Fragment skryptu służącego do pobierania danych.

Po pobraniu każdej stacji jest ona zapisywana w bazie danych. W celu przechowywania danych została wykorzystana baza PostgreSQL wraz z wtyczką PostGIS. Każda ze stacji zawiera następujące dane:

- `object_id` – identyfikator stacji (typ *INT*),
- `location` – krótki opis dotyczący lokalizacji stacji (typ *VARCHAR(100)*),
- `station_nr` – numer stacji w systemie w zakresie 6300 - 6467 (typ *INT*),
- `bikes` – informacja o aktualnej liczbie rowerów znajdujących się na stacji (typ *VARCHAR(10)* z racji na możliwość wystąpienia wartości tekstowej, np. „5+”),
- `stands` – informacja o dostępnej liczbie stojaków na stacji (typ *INT*),
- `station_coordinates` – współrzędne geograficzne stacji (typ *GEOMETRY(POINT,2059)*).

Dane o stacjach są przechowywane w tabeli *bike_stations*. W celu optymalizacji zapytań na dane założony został następujący indeks przestrzenny:

```
1 CREATE INDEX bike_stations_gix ON bike_stations USING GIST (station_coordinates);
```

Listing 2: Indeks przestrzenny założony na danych w tabeli.

3 Testy

W celu przetestowania tak utworzonej bazy danych zostały utworzone następujące przykładowe zapytania SQL, wykonujące obliczenia na podstawie danych przestrzennych:

Następnie zapytania zostały wielokrotnie wykonywane odpowiednio z założonym i zdjętym indeksem w celu przetestowania jego wpływu na wydajność zapytań przestrzennych. Wyniki zostały zamieszczone w poniższej tabeli.

4 Wnioski