

Dane przestrzenne (GIS) — punkty odbioru i strefy dostaw

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Zapisywanie położenia w kolumnach POINT i POLYGON, obliczanie odległości na kuli ziemskiej, wyszukiwanie najbliższego punktu odbioru i sprawdzanie, czy adres leży w strefie dostawy, oraz użycie indeksu SPATIAL.

Po zajęciach potrafisz:

- zapisuje i odczytuje punkty i wielokąty (ST_GeomFromText, POINT, ST_AsText, ST_X, ST_Y)
- oblicza odległości funkcją ST_Distance_Sphere i znajduje najbliższy punkt dla każdego klienta
- sprawdza przynależność punktu do obszaru funkcją ST_Contains
- oblicza długość trasy z kolejnych przystanków (LAG + ST_Distance_Sphere)
- sprawdza w EXPLAIN użycie indeksu SPATIAL i eksportuje dane do GeoJSON

Sprzęt i materiały

- komputer z XAMPP (MariaDB 10.4) lub MySQL 8.4, dostęp do internetu (podgląd mapy)
- plik `r2_26_dostawy.sql` z folderu Pliki_do_zadan

Oprogramowanie

- **XAMPP (MariaDB, phpMyAdmin)** — phpMyAdmin pokazuje kolumny geometryczne jako dane binarne — używaj ST_AsText
apachefriends.org
- **geojson.io** — podgląd GeoJSON na mapie w przeglądarce, bez rejestracji
geojson.io
- **Mapa internetowa (OpenStreetMap)** — odczyt współrzędnych: prawy przycisk myszy → Pokaż adres
openstreetmap.org

Przygotowanie przed zajęciami

- Przypomnij sobie z geografii: długość geograficzna (wschód–zachód) i szerokość geograficzna (północ–południe). Kraków leży ok. 20° E i 50° N.

Bezpieczeństwo

- Pracuj w bazie `dostawy`. Położenie klientów to dane osobowe — w zadaniu są fikcyjne; w prawdziwej aplikacji nie publikuj ich na mapie ani w GeoJSON bez potrzeby.
- Do geojson.io wklejaj tylko dane z zadania (punkty odbioru, strefy), nie dane klientów.

Przebieg zadania

1 Punkty i odległości

Zaimportuj plik `r2_26_dostawy.sql`. Współrzędne zapisano jako POINT(długość szerokość) bez układu odniesienia (SRID 0). Funkcja ST_Distance_Sphere(p1, p2) zwraca odległość w metrach po powierzchni Ziemi.

```
-- dwa sposoby zapisu tego samego punktu
ST_GeomFromText('POINT(19.9373 50.0617)')
POINT(19.9373, 50.0617)
```

1. Z1. Wyświetl nazwę, zapis tekstowy położenia (ST_AsText), długość (ST_X) i szerokość (ST_Y) punktów odbioru w Krakowie.
2. Z2. Odczytaj na mapie OpenStreetMap współrzędne dowolnego miejsca w Krakowie (np. szkoły lub dworca) i dodaj punkt odbioru o nazwie „Punkt Dworzec” lub własnej. Uwaga: OpenStreetMap podaje najpierw szerokość, potem długość.
3. Z3. Wyświetl odległość w km (2 miejsca po przecinku) każdego punktu w Krakowie od punktu Stare Miasto (19.9373, 50.0617), od najbliższego.
4. Z4. Znajdź jeden najbliższy punkt odbioru dla klienta Zielińskiego (odległość w metrach).
5. Z5. Dla każdego klienta znajdź najbliższy punkt odbioru (ROW_NUMBER() OVER (PARTITION BY ... ORDER BY odległość)).
6. Z6. Wyświetl punkty odbioru w promieniu 2 km od klientki Nowak.
7. Z7. Oblicz odległości w linii prostej między punktami w Krakowie (Stare Miasto), Warszawie, Gdańsku i Wrocławiu — każdą parę miast raz (złączenie tabeli z samą sobą). Porównaj jedną z odległości z trasą samochodową na mapie.

2 Strefy dostawy (wielokąty)

Tabela strefy zawiera dwa prostokątne obszary (POLYGON). Pierwszy i ostatni punkt wielokąta muszą być takie same.

1. Z8. Dla każdego klienta wyświetl strefę dostawy i opłatę, a dla klientów spoza stref napis „poza strefami” (LEFT JOIN ... ON ST_Contains(obszar, polozenie)).
2. Z9. Policz punkty odbioru w każdej strefie.
3. Z10. Wyeksportuj punkty odbioru w Krakowie i strefę „Centrum” funkcją ST_AsGeoJSON. Wklej wynik dla strefy do geojson.io (zamień zawartość okna na obiekt Polygon) i zrób zrzut mapy. Dodaj wielokąt własnej strefy dostawy (np. wokół szkoły) poleceniem INSERT i sprawdź go tak samo.

3 Trasa kuriera i indeks SPATIAL

Tabela kurs_przystanki zawiera kolejne przystanki kursu nr 1, a tabela automaty — 5000 fikcyjnych automatów paczkowych z indeksem SPATIAL.

1. Z11. Wyświetl przystanki kursu 1 z długością odcinka od poprzedniego przystanku w km (funkcja LAG), a następnie oblicz długość całej trasy.
2. Z12. Policz automaty leżące w prostokącie 19,5–20,5° E i 49,8–50,3° N (MBRContains z wielokątem). Porównaj EXPLAIN tego zapytania bez wskazówek i z `IGNORE INDEX (sp_automat)`.
3. Z13. Policz automaty w promieniu 20 km od Starego Miasta warunkiem na ST_Distance_Sphere i sprawdź EXPLAIN. Następnie napisz zapytanie, które najpierw zawęży wyszukiwanie prostokątem (MBRContains — korzysta z indeksu), a potem sortuje po odległości i zwraca 3 najbliższe automaty.

W sprawozdaniu umieść

- ☐ plik `zadanie26.sql` z zapytaniami Z1–Z13 opisanymi komentarzami

- ☐ zrzuty ekranu wyników Z5, Z8 i Z11
- ☐ zrzut mapy z geojson.io ze strefą „Centrum” i własną strefą (Z10)
- ☐ porównanie planów EXPLAIN z Z12 i Z13
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Dlaczego odległość z ST_Distance_Sphere jest mniejsza niż trasa samochodowa i do czego mimo to się przydaje?
2. Czym różni się MBRContains od ST_Contains? Kiedy wynik może być inny?
3. Dlaczego warunek ST_Distance_Sphere(...) < 20000 nie korzysta z indeksu SPATIAL i jak to obejść?
4. Podaj trzy pytania, na które odpowiada aplikacja z danymi przestrzennymi w sklepie internetowym lub firmie kurierskiej.

Kryteria oceny

Kryterium	Punkty
Z1–Z4: zapis punktów i obliczanie odległości	4
Z5–Z7: najbliższy punkt dla każdego klienta, promień, odległości między miastami	4
Z8–Z10: strefy (ST_Contains), GeoJSON i własny wielokąt na mapie	4
Z11–Z13: długość trasy i analiza indeksu SPATIAL	5
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).