# **Лабораторна робота №2**

**Знайомство з функціями PostGIS по роботі з географічними об’єктами**

**Мета роботи:** ознайомитись з функціями створення географічних об’єктів, а також функціями вибірки та аналізу географічних даних.

## Хід роботи

### **Знаходження центру географічних об’єктів**

В якості першого завдання створили збережену процедуру *get\_centroids*, що виконує розрахунок центру кожного з географічних об’єктів вказаної таблиці, а також записує отримані результати до таблиці *objects\_centroids*.

SQL-запит на створення таблиці *objects\_centroids* та просторового індексу для атрибута, що зберігає географічні дані об’єкта, наведено нижче у лістингу 2.1, текст функції *get\_centroids* – у лістингу 2.2, а результати тестування – на рисунку 2.1.

Лістинг 2.1 – Створення таблиці *objects\_centroids*

|  |
| --- |
| CREATE TABLE IF NOT EXISTS public.objects\_centroids (  id\_centroid serial NOT NULL,  centroid\_geom geometry NOT NULL,  centroid\_text text NOT NULL,  id\_building bigint NOT NULL,  CONSTRAINT buildings\_centroid\_fk FOREIGN KEY (id\_building)  REFERENCES public.buildings (gid) MATCH SIMPLE  ON UPDATE CASCADE  ON DELETE CASCADE  );  CREATE INDEX centroids\_geom\_index ON public.objects\_centroids  USING GIST (centroid\_geom); |

Лістинг 2.2 – Функція розрахунку центру кожного об’єкту

|  |
| --- |
| CREATE OR REPLACE PROCEDURE get\_centroids()  AS $body$  BEGIN  INSERT INTO objects\_centroids(centroid\_geom, centroid\_text,  id\_building)  SELECT ST\_Centroid(geom), ST\_AsText(ST\_Centroid(geom)), gid  FROM buildings;  END;  $body$  LANGUAGE plpgsql;  CALL get\_centroids(); |

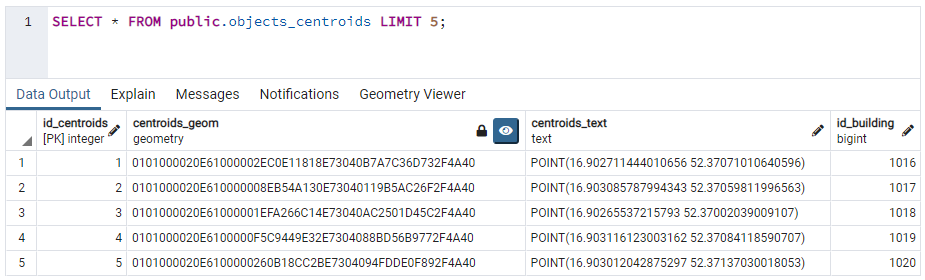


Рисунок 2.1 – Вміст таблиці objects\_centroids після виконання функції

Далі скористалися ПЗ QGIS для візуалізації даних, отриманих після виконання процедури *get\_centroids* (рисунок 2.2).

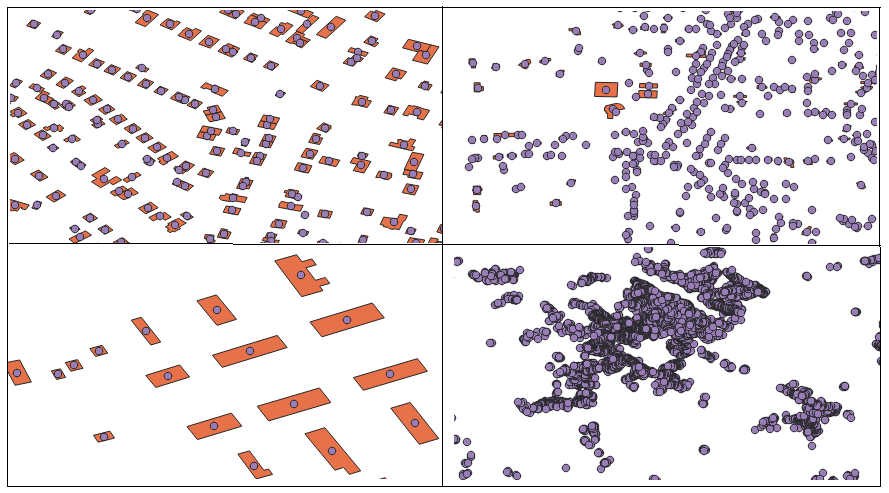


Рисунок 2.2 – Центри всіх географічних об’єктів на мапі міста Познань

### **Побудова найкоротшого маршруту між двома об’єктами**

В якості другого завдання створили функцію *build\_route*, яка будує найкоротший маршрут між двома об’єктами, що задані користувачем як дві довільні точки. Маршрут прокладається між центрами об’єктів, до яких належать задані точки. Для знаходження таких об’єктів і їх географічних центрів створили функцію *get\_object\_centre*, до якої у якості параметра передаються дані про точку. У випадку, якщо точка не належить до жодного з об'єктів БД, то видається відповідне сповіщення. Для збереження отриманих даних про маршрут і його довжину створили таблицю *route\_between\_objects*. Перед кожною вставкою нових даних таблиця видаляє всі наявні у ній записи (тригер *clean\_before\_insert*), а відстань розраховується та вставляється до таблиці окремо за допомогою тригера *calculate\_distance*.

SQL-запит на створення таблиці *route\_between\_objects* наведено нижче у лістингу 2.3, текст функції *get\_object\_centre* – у лістингу 2.4, функції *build\_route* – у лістингу 2.5, тригерів *clean\_before\_insert* і *calculate\_distance* у лістингах 2.6 і 2.7 відповідно, а результати тестування – на рисунках 2.3 та 2.4.

Лістинг 2.3 – Створення таблиці *route\_between\_objects*

|  |
| --- |
| CREATE TABLE IF NOT EXISTS public.route\_between\_objects  (  route geometry NOT NULL,  distance double precision  ); |

Лістинг 2.4 – Функція *get\_object\_centre*

|  |
| --- |
| CREATE OR REPLACE FUNCTION get\_object\_centre(point geometry)  RETURNS geometry  AS $body$  BEGIN  RETURN (SELECT oc.centroid\_geom  FROM buildings b, objects\_centroids oc  WHERE b.gid = oc.id\_building AND ST\_Contains(b.geom, point));  END;  $body$  LANGUAGE plpgsql; |

Лістинг 2.5 – Функція *build\_route*

|  |
| --- |
| CREATE OR REPLACE PROCEDURE build\_route(p1 double precision[], p2  double precision[])  AS $body$  DECLARE  point\_1 geometry := ST\_SetSRID(ST\_MakePoint(p1[1], p1[2]), 4326);  point\_2 geometry := ST\_SetSRID(ST\_MakePoint(p2[1], p2[2]), 4326);    centre\_1 geometry;  centre\_2 geometry;  BEGIN  centre\_1 := get\_object\_centre(point\_1);  centre\_2 := get\_object\_centre(point\_2);  IF (centre\_1 IS NOT NULL) AND (centre\_2 IS NOT NULL) THEN  INSERT INTO route\_between\_objects(route)  SELECT ST\_MakeLine(centre\_1, centre\_2);  ELSE |

Лістинг 2.5 – Функція *build\_route* (продовження)

|  |
| --- |
| RAISE EXCEPTION 'Точка вне объекта!';  END IF;  END;  $body$  LANGUAGE plpgsql; |

Лістинг 2.5 – Тригер *clean\_before\_insert*

|  |
| --- |
| CREATE OR REPLACE FUNCTION clean\_before\_insert() RETURNS TRIGGER AS $$  BEGIN  DELETE FROM route\_between\_objects;  RETURN NEW;  END;  $$  LANGUAGE plpgsql;  CREATE TRIGGER clear\_table\_trigger  BEFORE INSERT ON route\_between\_objects  FOR EACH ROW EXECUTE PROCEDURE clean\_before\_insert(); |

Лістинг 2.6 – Тригер *calculate\_distance*

|  |
| --- |
| CREATE OR REPLACE FUNCTION calculate\_distance() RETURNS TRIGGER  AS $body$  BEGIN  UPDATE route\_between\_objects rbo  SET distance = ST\_Length(NEW.route);  RETURN NEW;  END;  $body$  LANGUAGE plpgsql;  CREATE TRIGGER distance\_trigger AFTER INSERT ON route\_between\_objects  FOR EACH ROW EXECUTE PROCEDURE calculate\_distance(); |

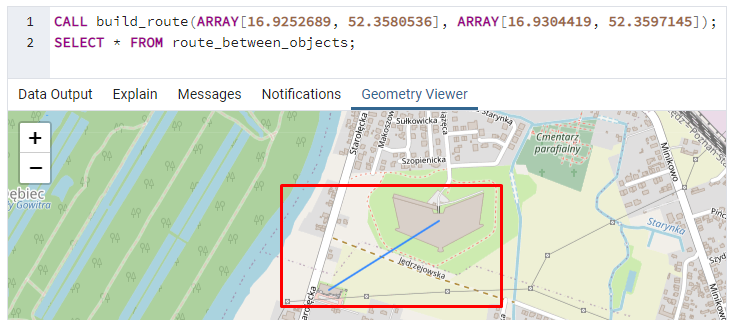


Рисунок 2.3 – Результати тестування (перегляд у PostgreSQL)

## 

Рисунок 2.4 – Результати тестування (перегляд у QGIS)

### **Побудова найкоротшого маршруту через довільну кількість об’єктів**

У третьому завданні створили функцію , подібну до раніше створеної *build\_route*, але при цьому вона дозволяє будувати найкоротший маршрут через довільну кількість об’єктів, що задаються користувачем. Текст функції *build\_route\_through\_several\_points* наведено у лістингу 2.7, а результати тестування – на рисунках 2.5 і 2.6.

Лістинг 2.7 – Функція *build\_route\_through\_several\_points*

|  |
| --- |
| CREATE OR REPLACE PROCEDURE build\_route\_through\_several\_points  (points\_string text) AS $body$  DECLARE  centre geometry;  geom\_point geometry;  geom\_points geometry[] := ARRAY(  SELECT ST\_PointN(column1, generate\_series(1, ST\_NPoints(column1)))  FROM (VALUES(ST\_GeomFromText(points\_string, 4326))) AS foo);  centre\_points geometry[];  BEGIN  FOREACH geom\_point IN ARRAY geom\_points LOOP  centre := get\_object\_centre(geom\_point);  IF centre IS NOT NULL THEN  centre\_points := centre\_points || centre;  ELSE  RAISE EXCEPTION 'Точка вне объекта!';  END IF;  END LOOP;    INSERT INTO route\_between\_objects(route) |

Лістинг 2.7 – Функція *build\_route\_through\_several\_points* (продовження)

|  |
| --- |
| SELECT ST\_MakeLine(centre\_points);  END;  $body$  LANGUAGE plpgsql; |

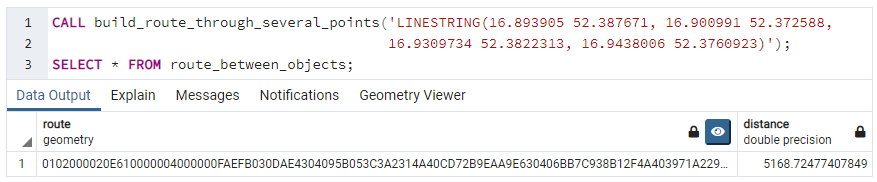


Рисунок 2.5 – Результати тестування (перегляд у PostgreSQL)

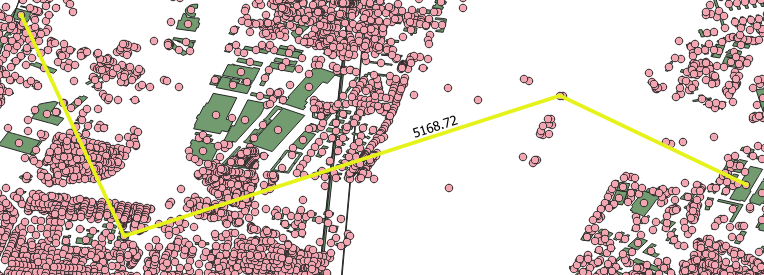


Рисунок 2.6 – Результати тестування (перегляд у QGIS)

## Висновки

Виконавши лабораторну роботу №2, ми ознайомились з функціями створення географічних об’єктів, вибірки та аналізу географічних даних. Створили три функції, які дозволяють знайти географічні центри всіх об’єктів, а також побудувати найкоротший маршрут як між двома точками, так і через їх довільну кількість.