Advanced Databases

Phase 5 – Advanced Topic: Spatial Database with PostGIS

Project: Balkan geotourism

Bora Alili 231504
Esra Bekiri 231521
Marko Smilevski 231557
Fisnik Mamuti 231508

6. Advanced Topic - Spatial Data Analysis with PostGIS

For the advanced phase of the project, spatial database functionality was implemented by integrating the PostGIS extension into PostgreSQL. Since the application manages tourism-related information such as hotels, attractions, airports, and travel destinations, geographical data plays an important role in providing useful services to users. Traditional SQL databases can store latitude and longitude values, but they cannot efficiently perform geographic calculations such as finding nearby objects, measuring

distances, or identifying the closest location. PostGIS extends PostgreSQL with spatial data types, functions, operators, and indexing mechanisms that enable efficient processing of geographic information.

The purpose of implementing PostGIS is to improve the tourism database by supporting location-based queries that can be used in real-world travel planning and recommendation systems.

Purpose of the Advanced Topic

The tourism application contains many entities that represent real physical locations, including:

• Hotels

• Attractions

• Airports

• Travel destinations

Users of such an application often need answers to questions such as:

• Which attractions are close to my hotel?

• Which airport is closest to my accommodation?

• How far is a destination from another location?

Without spatial functionality, these calculations would require complex mathematical formulas and would be inefficient for large datasets.

By integrating PostGIS, the database can perform these operations directly using optimized spatial algorithms.

Installation and Database Preparation

The first step was enabling the PostGIS extension inside PostgreSQL:

```
CREATE EXTENSION IF NOT EXISTS postgis;
```

This installs all spatial data types and geographic functions required for spatial processing.

Next, the existing coordinates table was extended by adding a new geography column:

```
ALTER TABLE coordinates
ADD COLUMN geom GEOGRAPHY(Point, 4326);
```

The GEOGRAPHY(Point, 4326) data type stores geographic coordinates on the Earth's surface using the WGS84 coordinate reference system (SRID 4326), which is the standard used by GPS systems.

After creating the column, every existing latitude and longitude pair was converted into a geographic point:

```
UPDATE coordinates
SET geom =
ST_SetSRID(
ST_MakePoint(longitude, latitude),
4326
)::geography;
```

The function ST_MakePoint() creates a spatial point from longitude and latitude, while ST_SetSRID() assigns the correct spatial reference system.

Finally, a GiST spatial index was created:

```
CREATE INDEX idx_coordinates_geom
ON coordinates
USING GIST (geom);
```

The GiST index significantly improves the performance of spatial searches by avoiding full table scans when executing geographic queries.

Implemented Spatial Queries

Query 1 – Finding Attractions Within 5 Kilometers of a Hotel

The first implemented query searches for all tourist attractions located within a radius of

five kilometers from a selected hotel.

The query joins the hotel with its coordinates and compares them against the coordinates of every attraction. The spatial function ST_DWithin() checks whether two geographic points are within a specified distance.

ST_DWithin(hc.geom, ac.geom, 5000)

The distance is specified in meters because the GEOGRAPHY data type automatically performs calculations on the Earth's surface.

Additionally, the exact distance between the hotel and each attraction is calculated using:

ST_Distance(hc.geom, ac.geom)

The results are ordered by distance so that the nearest attractions appear first.

Practical Usage

This query can be used for several features within the tourism application:

- recommending nearby attractions to hotel guests;
- displaying points of interest on a hotel details page;
- assisting users during trip planning;
- suggesting sightseeing activities within walking distance;

- improving the overall tourist experience by providing location-aware recommendations.

Query 2 – Finding the Nearest Airport for Every Hotel

The second implemented query determines the closest airport for every hotel stored in the database.

Each hotel is compared with every airport, and the spatial distance between them is calculated using **ST_Distance().**

The PostgreSQL clause:

DISTINCT ON (h.id)

ensures that only the airport with the minimum distance is returned for each hotel.

The calculated distance is converted from meters into kilometers for easier interpretation.

Practical Usage

This functionality can support several real-world scenarios:

- helping tourists identify the most convenient airport;
- assisting travel agencies when organizing airport transfers;
- estimating transportation distances;
- improving hotel search results by including airport proximity;
- supporting logistics and travel planning.

Spatial Functions Used

Several important PostGIS functions were used during implementation:

ST_MakePoint()

Creates a geographic point from longitude and latitude coordinates.

ST_SetSRID()

Assigns the appropriate spatial reference system (SRID 4326) to each geographic point.

ST_DWithin()

Checks whether two geographic objects are within a specified distance.

This function is particularly efficient because it can utilize the GiST spatial index.

ST_Distance()

Calculates the real geographic distance between two locations stored as geography objects.

Unlike simple Euclidean distance, this calculation accounts for the Earth's curvature, providing accurate real-world measurements.

Benefits of the Implementation

Integrating PostGIS extends the tourism database with advanced geographic capabilities that would otherwise require complex calculations in application code.

The implemented solution provides several advantages:

- efficient execution of spatial searches;
- accurate real-world distance calculations;
- support for location-based recommendations;
- improved scalability through spatial indexing;
- enhanced user experience by enabling geographic search functionality.

These capabilities significantly increase the practical value of the tourism application and demonstrate how spatial databases can be applied in modern travel information systems.

6.4 Spatial Query 3 – Nearest Transport Station for Every Hotel

This query determines the closest transport station (bus, railway, etc.) for each hotel by calculating the geographic distance between hotels and transport stations.

Purpose

- Recommend the nearest public transportation option.
- Improve accessibility information for travelers.
- Demonstrate another practical application of spatial distance calculations.

Conclusion

The PostGIS extension enhances the database by enabling location-aware functionality that would not be possible with standard SQL alone. The implemented spatial queries demonstrate three practical tourism scenarios:

- locating nearby attractions,
- identifying the nearest airport,
- finding the nearest transport station.

Together, these examples illustrate how spatial databases can improve travel planning and recommendation systems while maintaining efficient query performance through spatial indexing.