KATHMANDU UNIVERSITY

Dhulikhel, Kavre

SDBMS: GEOM 318

A Project Report on

Spatial Database Management System (SDBMS) Integration for Enhanced Decision-Making in Windmill Site Selection.

Prepared By:

Pragyan Baral - 07

Abhinav Chand - 15

Shisir Kharel -27

Rishav Khatiwada -29

Saurav Nepal -36

GE III/I

Supervisor:

Er. Ajay Kumar Thapa

Lecturer, DoGE

15th January 2024

TABLE OF CONTENTS

1. INTRODUCTION

Renewable energy, particularly wind power, stands as a cornerstone in the pursuit of sustainable and environmentally conscious energy solutions (Global Wind Energy Council, 2022). This project focuses on harnessing the potential of Geographic Information Systems (GIS) and Spatial Database Management System (SDBMS) to identify optimal locations for windmill sites within a specified region. The study area's exploration into wind energy potential necessitates a meticulous analysis of diverse spatial parameters to pinpoint suitable sites for windmill installations.

Our project leverages GIS to integrate datasets comprising road networks, land use and land cover (LULC), airports, transmission lines, slope, built-up areas, and wind speed. These criteria collectively contribute to a comprehensive evaluation of potential windmill sites. The project unfolds with a series of simple queries addressing specific aspects such as average distances, SRID updates, and filtering based on wind speed. (Li & Wu, 2016)

As the complexity of our analysis deepens, we introduce complex and nested queries, employing spatial operations and conditions to identify areas with specific attributes (Jokar Arsanjani et al., 2013). These include slope gradients, distances from roads, proximity to key infrastructures, and environmental considerations like elevation. The intricate nature of these queries demonstrates the capability of SDBMS and GIS to handle nuanced spatial relationships, providing decision-makers with valuable insights for wind energy project planning.

Our report details the methodology, data sources, and outcomes of these spatial queries, offering a comprehensive guide for future endeavors in identifying suitable locations for windmill sites. This project marks a crucial step towards sustainable energy planning and underscores the importance of leveraging GIS and SDBMS for informed decision-making in renewable energy initiatives.

2. OBJECTIVES

1. Spatial Analysis for Suitability Mapping: Utilize GIS and SDBMS to analyze road networks, land use, airports, transmission lines, slopes, built-up areas, and wind speed.

2. Advanced Spatial Criteria Integration: Apply advanced GIS queries to factor in complex spatial relationships, including slope gradients, infrastructure proximity, and environmental considerations. Use nested queries for refined site selection, enhancing decision-making for sustainable wind energy planning.

3. QUERIES

In this section, our methodology revolves around harnessing GIS and SDBMS tools to evaluate and select suitable locations for windmill sites in Gandaki Province. The five simple queries perform fundamental tasks such as calculating average distances, updating spatial references for wind turbines, and filtering windmills based on wind speed. These queries establish the groundwork for subsequent analyses. On the other hand, the seven complex queries delve into nuanced spatial relationships and criteria integration. They identify areas based on slope gradients, road proximity, airport and electricity grid distance, land use, settlement considerations, and wind speed. By employing nested queries, our approach ensures a refined site selection process, factoring in elevation and exclusion criteria. This comprehensive methodology enables a thorough assessment of potential wind energy sites in the region, combining simplicity for foundational insights with complexity for detailed spatial analysis.

3.1 Simple Queries

3.1.1. Query that represents the average distance from road to airport.

SELECT AVG (ST_Distance(roads_table.geom, airports_table.geom))
AS avg_distance
FROM roads_table, airports_table;

In the above query, 'ST_Distance' function is used to calculate the distance between geometries of tables 'roads_table' and 'airports_table'. Then the 'AVG()' function is used to calculate the average distance as 'avg_distance'.

3.1.2. Query that updates the windturbines' SRID to '32645' which is 'WGS 84/UTM zone 45N'.

SELECT UpdateGeometrySRID ('turbines_table', 'Geom', '32645');

Here, the geometries of 'turbines_table' are provided with SRID '32645' which is WGS 84/UTM Zone 45N.

3.1.3. Query to represent the windmills where the windspeed is greater than 9 m/s.

*SELECT * FROM wind_table where windspeed>9;*

Here, those data of the table 'wind_table' are selected that have the 'windspeed' column value greater than 9m/s.

3.1.4. Query to count the number of turbines from WindTurbine Table.

SELECT COUNT (DISTINCT turbines) from turbines_table;

Here, unique turbines of the 'turbines' column of the table 'turbines_table' are counted.

3.1.5. Query that selects the area from LULC table having region name ‘openarea’ or ‘bareground’.

*SELECT * FROM lulc_table*
WHERE lulcregion IN ('openarea', 'bareground');

In the above query, from the table 'lulc_table', only those data are selected that have column 'lulcregion' value either 'openarea' or 'bareground'.

3.2. Complex/Nested Queries:

3.2.1. Query that represents the area with slope less than 15% and lies in between 500m and 5000m from road boundary.

```
SELECT gd.area
FROM gd_table as gd
JOIN roads_table rd ON ST_DWithin(gd.geom, rd.geom, 100)
WHERE ST_Slope(gd.geom) < 15
AND ST_Length(rd.geom) > 500
AND ST_Length(rd.geom) < 5000;
```

In this query, spatial join operation is done between 'gd_table' and 'roads_table' based on the condition that the geometries ('gd.geom' and 'rd.geom') are within a distance of 100 units from each other. Also,
WHERE ST_Slope(gd.geom) < 15: Filters the results to include only those rows where the slope of the geometry in "gd_table" is less than 15.

AND ST_Length(rd.geom) > 500: Adds another condition to filter the results, including only rows where the length of the geometry in "roads_table" is greater than 500.

AND ST_Length(rd.geom) < 5000: Adds a final condition to further filter the results, including only rows where the length of the geometry in "roads_table" is less than 5000.

3.2.2. Query to find suitable areas meeting all criteria and again refines the selection based on elevation and exclusion criteria

```
WITH suitable_areas AS (
  SELECT area
  FROM gd_table
  WHERE ST_Slope(geom) < 15
    AND ST_Length(road_geom) > 500
    AND ST_Length(road_geom) < 5000
    AND ST_DWithin(geom, airport_geom, 5000)
    AND ST_DWithin(geom, electricity_grid_geom, 2000)
    AND land_use IN ('Open Space', 'Farmland')
```

```
   AND settlement_distance(geom) > 1000
   AND wind_speed > 8)
SELECT sa.area
FROM suitable_areas sa
WHERE elevation > (
  SELECT AVG (elevation)
  FROM gd_table
  WHERE ST_Within(geom, ST_Buffer((SELECT ST_Centroid(geom) FROM suitable_areas),
1000)))
AND area NOT IN (
  SELECT area
  FROM excluded_areas);
```

In this query, first of all table named 'suitable_areas' is created having 'area' column which is created using multiple criteria. Then, 'area' from the 'suitable_areas' table is selected such that 'elevation' has a certain criterion. This way, suitable areas are found out or filtered using query.

In summary, this query aims to find suitable areas meeting various conditions (slope, road length, proximity to airport and electricity grid, land use, settlement distance, and wind speed) and then further refines the selection based on elevation and exclusion criteria. The query utilizes spatial functions like 'ST_DWithin', 'ST_Within', and 'ST_Buffer' for geometric comparisons.

3.2.3. Query that represents the area where the windspeed is less than 4.8 m/s and elevation is less than 1000m.

```
SELECT
w.geom AS windspeed_geom,
e.geom AS elevation_geom,
w.windspeed,
e.elevation
FROM wind_table as w
JOIN elevation_table as e
ON ST_Intersects(ST_Buffer(w.geom, 500), e.geom)
WHERE w.windspeed < 4.8 AND e.elevation < 1000;
```

Firstly, geometries of the tables 'wind_table' and 'elevation_table' are selected. Column 'windspeed' from table 'wind_table' and column 'elevation' from table 'elevation_table' are also selected. Then, earlier mentioned two table are joined using the 'ST_Intersects' function, which checks if the buffered geometry of the wind_table intersects with the geometry of the elevation_table. 'WHERE w.windspeed < 4.8 AND e.elevation < 1000’ filters the result set to include only rows where the windspeed in the wind_table is less than 4.8 and the elevation in the elevation_table is less than 1000.

3.2.4. Query that calculates the distance from each airport to the point where the wind speed is less than 9m/s and greater than 25m/s.

SELECT gd.area, gd.wind_speed,
ST_Distance(gd.geom, a.airport_geom)
AS distance_to_airport
FROM gd_table gd
JOIN airports_table a
ON ST_DWithin(gd.geom, a.airport_geom, 5000)
gd.wind_speed > 9
AND gd.wind_speed < 25;

Here, at first, the data of the columns 'area', 'wind_speed' and distance between the geometries of 'gd_table' and 'airports_table' are selected from 'gd_table'. Then 'gd_table' and 'airports_table' are joined in such a way that geometries of 'gd_table' and 'airports_table' are within 5000 metres and value of 'windspeed' column is greater than 9 m/s and less than 25 m/s.

3.2.5. Query that calculates the distance between each airport.

SELECT a.geom, st_distance(a.geom,b.geom)
AS dist,
a.placename as aname,
b.placename as bname
FROM airports_table as a, airports_table as b
WHERE a.airports<>b.airports
ORDER BY aname,
dist ASC;

This query retrieves pairs of airports with their names, calculates the spatial distance between them, and presents the results in ascending order of airport names and distance. The self-join is used to compare each airport with every other airport, excluding self-comparisons.

3.2.6. Query that locate the wind turbines into high and highest suitability zones.

SELECT wt.geom AS turbine_location
FROM turbines_table as wt
JOIN SuitabilityMap sm
ON ST_Contains(sm.geom, wt.geom)
WHERE sm.suitability
IN ('Highest Suitability', 'High Suitability');

In this query, firstly, geometries of the table 'turbines_table' are selected. Then, 'SuitabilityMap' table is joined with 'turbines_table' based on 'ST_Contains' functions in which geometries of 'turbines_tables' must be contained in geometries of 'SuitabilityMap' table where 'suitability' column of the table 'SuitabilityMap' have values Highest and High Suitability.

3.2.7. Query that represents the windmills where the elevation is less than 1000m and distance to road from windmills is less than 500m but it excludes the area if it contains airport.

```
SELECT turbines_table.windmill_id , turbines_table.geometry
FROM turbines_table w
JOIN elevationtable.elevation_criteria e
ON w.windmill_id = e.windmill_id
JOIN road_table.road_criteria rp
ON ST_DWithin(w.geometry, rp.road_geometry, 500)
LEFT JOIN aiports_table.airport_criteria ap
ON ST_DWithin(w.geometry, ap.airport_geometry, 1000)
WHERE e.elevation <= 1000
  AND rp.distance_to_road <= 500
  AND (ap.airport_id IS NULL OR ap.distance_to_airport > 1000);
```

- The provided query involves joining several tables related to windmills, elevation criteria, road proximity criteria, and airport proximity criteria. The query filters windmills based on their elevation, proximity to roads, and absence of nearby airports.
- **FROM turbines_table** : Specifies the primary table as **turbines_table** and assigns the alias w to it.
- **JOIN elevationtable.elevation_criteria e ON w.windmill_id** = e.windmill_id: Joins the elevation_criteria table with the windmill table based on the windmill_id.
- **JOIN road_proximity_criteria rp ON ST_DWithin(w.geometry, rp.road_geometry, 500):** Joins the road_proximity_criteria table based on the spatial condition that the windmill's geometry is within 500 units of the road's geometry.
- **LEFT JOIN airport_proximity_criteria ap ON ST_DWithin(w.geometry, ap.airport_geometry, 1000):** Left joins the airport_criteria table based on the spatial condition that the windmill's geometry is within 1000 units of the airport's geometry.
- **WHERE e.elevation <= 1000:** Filters the result to include only rows where the windmill's elevation is less than or equal to 1000 units.
- **AND rp.distance_to_road <= 500:** Additional filtering to include only windmills where the distance to the road is less than or equal to 500 units.
- **(ap.airport_id IS NULL OR ap.distance_to_airport > 1000):** Ensures that windmills are included in the result either if there is no corresponding airport proximity information (ap.airport_id IS NULL) or if the distance to the airport is greater than 1000 units.

3.2.8. Query to create the buffer zone around road, transmission lines, and built-up areas and combine all the regions into a separate buffer area.

```
WITH roadbuffer_cte AS (
    SELECT ST_Buffer(roads_table.geom, 500) AS rdgeom
    FROM road_table),
transmissionlinesbuffer_cte AS (
```

```
    SELECT ST_Buffer(transmissionline_table.geom, 500) AS tlgeom
    FROM transmissionlines_table),
Builtupareasbuffer_cte AS (
    SELECT ST_Buffer(builtuparea_table.geom, 2000) AS bageom
    FROM builtupareas_table ),
final_union AS (
    SELECT ST_Union(geom) AS geom
    FROM (
        SELECT rdgeom FROM roadbuffer_table
        UNION
        SELECT tlgeom FROM transmissionlinesbuffer_table
        UNION
        SELECT bageom FROM builtupareasbuffer_table
    ) AS buffers
)
SELECT * FROM finalunion_cte;
```

The provided query performs spatial operations to create buffer zones around roads, transmission lines, and built-up areas, and then combines these buffer zones into a single geometry using the **ST_Union** function. Let's break down the query step by step:

- **Roadbuffer_cte**: Creates buffer zones of 500 units around geometries in the roads_table.
- **transmissionlinesbuffer_cte**: Creates buffer zones of 500 units around geometries in the transmissionlines_table.
- **builtupareasbuffer _cte:** Creates buffer zones of 2000 units around geometries in the builtupareas _table.
- **finalunion_cte:** Combines the buffer zones from roadbuffer, transmissionlinesbuffer, and builtupareas_cte into a single geometry using the **ST_Union** function.
- **SELECT * FROM finalunion_cte:** Retrieves and displays the resulting geometry from the combined buffer zones. This could be visualized or used for further analysis.

3.2.9. Query to create the buffer around airport that contains LULC (forest, builtuparea and waterbody) and the windspeed is less than 4.8 m/s and greater than 25 m/s.

```
SELECT * FROM wind_table, lulc_table
WHERE ST_Contains ((SELECT ST_Buffer(airports_table.geom, 15000) AS ageom
FROM airports_table
WHERE lulc_table.lulcregion IN (‘forest’, ‘builtuparea’, ‘waterbody’)), wind_table.geom)
(SELECT wind_table.geom FROM wind_table WHERE wind_table.windspeed <= 4.8 AND
wind_table.windspeed >= 25));
```

Query attempts to select records from wind_table and lulc_table where the wind speed falls within a specified range and the point is contained within a buffer zone around airports within specific land use and land cover (LULC) regions.

- **SELECT * FROM wind_table, lulc_table:** This part of the query specifies that you want to retrieve all columns from the tables wind_table and lulc_table. This is done using a Cartesian product, which essentially combines every row from wind_table with every row from lulc_table.
- **WHERE ST_Contains(...):** This is the spatial condition that filters the rows based on whether the geometry of a wind point is contained within a buffered zone around airports within specific land use and land cover (LULC) regions.
- **(SELECT ST_Buffer(airports_table.geom, 15000) AS ageom ...):** This subquery creates a buffer (circular in this case) around airport geometries (airports_table.geom) with a radius of 15000 units. The result is aliased as ageom.
- **FROM airports_table WHERE lulc_table.lulcregion IN ('forest', 'builtuparea', 'waterbody'):** These filters airports based on their LULC region, including only those in the specified regions ('forest', 'builtuparea', 'waterbody').
- **wind_table.geom:** This is the geometry column from wind_table, representing the wind points.
- **AND wind_table.windspeed <= 4.8 AND wind_table.windspeed >= 25:** This part of the query includes additional conditions on wind speed, ensuring that wind speed is both less than or equal to 4.8 and greater than or equal to 25. This is a bit contradictory, as wind speed cannot be simultaneously both less than or equal to 4.8 and greater than or equal to 25. You might want to adjust these conditions based on your actual requirements.

3.2.10. Query that represents the aggregate information about windspeed within different landcover site within the specified bounding box and meeting specified criteria. (Use of aggregate spatial analysis)

SELECT lulctable.lulcregion, AVG (wind_data.windspeed) AS avg_windspeed
FROM gd_table
JOIN lulc_table ON ST_Within(gd_table.geom, ST_MakeEnvelope(min_lon, min_lat, max_lon, max_lat, 32645))
JOIN wind_data ON gd_table.windmill_id = wind_data.windmill_id
JOIN road_table ON ST_DWithin(gd_table.geom, road_table.geom, roadbufferdistance)
WHERE wind_data.windspeed BETWEEN 4.8 AND 25
AND wind_data.elevationdata BETWEEN 1500 AND 4750
GROUP BY lulc_table.lulcregion;

This query calculates the average wind speed (windspeed) for different land use and land cover (LULC) regions within a specified geographic area, considering additional conditions such as wind speed range, elevation range, and proximity to roads.

SELECT lulctable.lulcregion, AVG(wind_data.windspeed) AS avg_windspeed: This part of the query specifies that you want to retrieve the LULC region (lulctable.lulcregion) and the average wind speed (AVG(wind_data.windspeed)) for each unique LULC region.

FROM gd_table: This indicates that the main table being queried is gd_table.

JOIN lulc_table ON ST_Within(gd_table.geom, ST_MakeEnvelope(min_lon, min_lat, max_lon, max_lat, 4326)): This join condition checks whether the geometry (geom) in gd_table is within the specified bounding box defined by ST_MakeEnvelope. It joins with the lulc_table based on this spatial relationship.

JOIN wind_data ON gd_table.windmill_id = wind_data.windmill_id: This joins the wind data in wind_data to the corresponding windmills in gd_table based on the windmill_id.

JOIN road_table ON ST_DWithin(gd_table.geom, road_table.geom, roadbufferdistance): This join checks whether the geometry of each record in gd_table is within a specified distance (roadbufferdistance) of the geometry in road_table. It joins with the road_table based on this spatial relationship.

WHERE wind_data.windspeed BETWEEN 4.8 AND 25: This filters the results to include only those records where the wind speed is between 4.8 and 25.

AND wind_data.elevationdata BETWEEN 1500 AND 4750: This adds another condition, filtering the results to include only those records where the elevation data is between 1500 and 4750.

GROUP BY lulc_table.lulcregion: This groups the results by the LULC region, so the average wind speed is calculated for each unique LULC region.

4. CONCLUSION

In conclusion, our use of SDBMS tools, particularly PostgreSQL with spatial extensions, has been critical in conducting a robust and efficient analysis for windmill site selection in our project. The use of SQL queries with spatial functions has enabled us to efficiently process and interpret geospatial data. Simple queries, such as updating SRID for wind turbines and filtering by wind speed, complement the complexity of spatial join operations, allowing for more nuanced assessments of slope, road proximity, and environmental criteria. The systematic approach to complex queries, such as proximity to multiple features and nested conditions, improves the granularity of our site suitability assessment.

This SDBMS implementation not only enables a thorough examination of potential wind energy sites, but it also lays the groundwork for future scalability and adaptability to similar projects. It emphasizes the importance of spatial databases in supporting geospatial decision-making processes, as well as the need for accuracy and efficiency when managing and analyzing spatial data for renewable energy projects.

5. REFRENCES

Global Wind Energy Council. (2022). Global Wind Report 2022.

Goodchild, M. F. (2007). Citizens as Voluntary Sensors: Spatial Data Infrastructure in the World of Web 2.0. *International Journal of Spatial Data Infrastructures Research, 2*, 24-32.

Jokar Arsanjani, J., Helbich, M., & Bakillah, M. (2013). Towards a multi-dimensional segmentation of crowded neighborhoods using very high resolution remotely sensed imagery and GIS. *Computers, Environment and Urban Systems, 41*, 91-102.

Li, Z., & Wu, B. (2016). A GIS-based decision support system for site selection of tidal stream power plants. *Renewable Energy, 97*, 407-417.

Shekhar, S., Chawla, S., & Lu, C.-T. (2003). Spatial Databases: Accomplishments and Research Needs. *GeoInformatica, 7*(4), 251–273.