Research Proposal

Jinming Ge

November 2, 2025

1 Introduction

Spatial queries are fundamental operations in geographic information systems (GIS) and spatial databases. In Assignment 1, two spatial query methods were implemented: *point-in-polygon* queries and *polygon-in-polygon* queries. To support these query operations, four core spatial functions are required, as defined below:

```
MAKE_POINT(latitude, longitude)
    -- Constructs a geometric point object from coordinates

MAKE_POLYGON(vertex_table_name, id_column, id_value)
    -- Constructs a polygon object by fetching and ordering
    -- vertices from the specified vertex table

ST_Contains(geometry1, geometry2)
    -- Returns TRUE if geometry1 spatially contains geometry2

ST_Distance(geometry1, geometry2)
    -- Computes the minimum distance between two geometries
```

Listing 1: Spatial Operation Definitions

However, without proper indexing, spatial queries become computationally expensive, particularly when dealing with large-scale geospatial datasets. This research proposes to compare the efficiency of different spatial indexing methods for the aforementioned query operations. Specifically, three prominent indexing techniques will be evaluated:

- **Z-order curve (Morton code):** A space-filling curve that maps multi-dimensional spatial data to one dimension by interleaving coordinate bits, preserving spatial locality.
- **R-tree:** A hierarchical structure organizing spatial objects using minimum bounding rectangles (MBRs) with balanced area and overlap.
- **R*-tree:** An optimized R-tree variant with improved node splitting strategies that minimize area, overlap, and margin.

The research will empirically evaluate these indexing methods in terms of query response time, index construction overhead, and memory consumption.

2 Implementation

2.1 Platform and Software

The implementation will be conducted using the following environment:

- **Database System:** PostgreSQL 15.x with PostGIS 3.x extension for spatial data management
- **Programming Language:** Python 3.10+ for data preprocessing and benchmarking
- **Development Tools:** Jupyter Notebook for analysis, pgAdmin for database management

2.2 Datasets

To ensure comprehensive evaluation, the following real-world geospatial datasets will be used:

- **OpenStreetMap (OSM) Data:** Urban building footprints and administrative boundaries
- **Scale Variation:** Datasets ranging from 10,000 to 1,000,000 spatial objects to evaluate scalability
- **Geometry Complexity:** Mix of simple (3-10 vertices) and complex polygons (100+ vertices)

2.3 Experimental Design

The experimental methodology comprises three phases:

2.3.1 Phase 1: Index Construction

Each indexing method will be implemented and constructed on identical datasets. Metrics include:

- Construction time
- Index size (memory footprint)
- Tree depth (for hierarchical structures)

2.3.2 Phase 2: Query Performance Evaluation

Two query workloads will be executed:

1. **Point-in-polygon queries:** Test with varying point distributions (uniform, clustered, random)
2. **Polygon-in-polygon queries:** Evaluate containment tests with different polygon sizes

For each query type, measure:

- Average query response time
- Query throughput (queries per second)
- Number of disk I/O operations

2.3.3 Phase 3: Comparative Analysis

Results will be compared across:

- Query selectivity (varying result set sizes)
- Data distribution patterns
- Update performance (insertion/deletion operations)

2.4 Evaluation Metrics

The comparative analysis will focus on:

- **Query Efficiency:** Response time for point-in-polygon and polygon-in-polygon queries
- **Scalability:** Performance degradation as dataset size increases
- **Space Overhead:** Index size relative to raw data size
- **Construction Cost:** Time required to build the index structure

3 Expected Outcomes

This research will provide empirical evidence on the relative strengths and weaknesses of each spatial indexing method, offering practical guidance for selecting appropriate indexing strategies based on specific query workload characteristics and data properties.