Laboratorio 04: Uso de GiST para Consultas Espaciales

Prof. Heider Sanchez
ACLs: Ana María Accilio, Sebastián Loza

Introducción

En aplicaciones modernas como sistemas de navegación, mapas interactivos, logística, planificación urbana o geolocalización en apps móviles, es común responder preguntas como:

- ¿Qué ciudades se encuentran dentro de un radio determinado?
- ¿Cuáles son las ciudades más cercanas a una ubicación específica?
- ¿Cuál es el punto más cercano dentro de un conjunto de ubicaciones?

Estas consultas espaciales no son eficientes con índices tradicionales (B-Tree), ya que se basan en operaciones de igualdad o rangos lineales. En cambio, GiST (**Generalized Search Tree**) permite indexar estructuras complejas como geometrías, habilitando búsquedas por proximidad, intersección o distancia.

En este laboratorio se le pide usar **GiST con PostGIS** para acelerar consultas espaciales en una tabla de ciudades con coordenadas geográficas.

Diseño del Experimento

Se solicita crear una tabla para almacenar más de 140 mil ciudades del mundo, convirtiendo sus coordenadas en geometrías espaciales. A partir de estos datos, se generará un índice espacial GiST y se realizarán consultas para identificar ciudades dentro de un radio específico o cercanas a una ubicación dada. Finalmente, se comparará el rendimiento de las consultas con y sin el uso del índice.

1. Preparación del entorno

```
CREATE DATABASE ciudades;
\c ciudades
CREATE EXTENSION postgis; -- Es importante tener instalado PostGIS en su computadora
```

2. Crear la tabla base

```
CREATE TABLE cities (
    id int PRIMARY KEY,
    name varchar(100),
    state_id INT,
    state_code varchar(10),
    state_name varchar(100),
    country_id INT,
    country_code varchar(10),
    country_name varchar(100),
    latitude DOUBLE PRECISION,
    longitude DOUBLE PRECISION,
    wikiDataId varchar(15)
);
```

3. Importar el CSV

```
COPY cities(id, name, state_id, state_code, state_name, country_id, country_code,
country_name, latitude, longitude)
FROM '/ruta/al/archivo/cities.csv'
DELIMITER ','
CSV HEADER;
```

4. Agregar geometría

```
ALTER TABLE cities ADD COLUMN ubicacion GEOMETRY(Point, 4326);
UPDATE cities SET ubicacion = ST_SetSRID(ST_MakePoint(longitude, latitude), 4326);
```

5. Crear índice GiST

```
CREATE INDEX idx_cities_geom_gist ON cities USING GIST (ubicacion);
```

Consultas Espaciales

1. Ciudades dentro de un radio

```
SELECT name, country_name
FROM cities
WHERE ST_DWithin(
    ubicacion::geography,
    ST_MakePoint(-78.91667,  -8.08333)::geography,
    10000 -- metros
);
```

2. KNN – 5 ciudades más cercanas

```
SELECT c.name, c.country_name,
       ST_Distance(c.ubicacion::geography, qp.ubicacion::geography) AS distance
FROM cities c,
     (SELECT ST_SetSRID(ST_MakePoint(-78.91667, -8.08333), 4326) AS ubicacion) qp
ORDER BY c.ubicacion <#> qp.ubicacion
LIMIT 5;
```

- `<#>` calcula la distancia geográfica (considera la curvatura de la tierra).
- `<->` calcula la distancia geométrica (aproximada en grados).

Comparación de Rendimiento

- Crear una copia de la columna geométrica que permita realizar consultas sin aprovechar el índice espacial:

```
ALTER TABLE cities ADD COLUMN ubicacion2 GEOMETRY(Point, 4326);
UPDATE cities SET ubicacion2 = ubicacion;
```

- Aplicar la siguiente consulta sobre ambas columnas —la indexada y la no indexada— para comparar su rendimiento:

```
EXPLAIN ANALYZE
SELECT name
FROM cities
WHERE ST_DWithin(
    ubicacion::geography,
    ST_MakePoint(-78.91667,  -8.08333)::geography,
    100000
);
```

> ¿Quién obtuvo mejores resultados?

Ejercicios Propuestos para el Alumno

1. (1 pts) Consulta por Rango

Encontrar todas las ciudades dentro del rectángulo delimitado por:

- Coordenadas: (lon: -79.4742, lat: -8.6159) y (lon: -78.4742, lat: -7.6159)

```
-- Tu consulta aquí
```

2. (1 pts) Top 10 ciudades más cercanas a Lima

Encontrar las 10 ciudades más cercanas a Lima. Se debe mostrar el nombre de la ciudad, el país y la distancia en kilómetros.

- Coordenadas de Lima: (-77.0282, -12.0432)

```
-- Tu consulta aquí usando ORDER BY <-> y LIMIT
```

3. (4 pts) Función de vecinos más cercanos

Implementar una función o procedimiento almacenado que, dado el nombre de una ciudad, devuelva sus **N ciudades más cercanas**. Aplicar indice B-Tree adecuadamente para acelerar la consulta.

```
-- CREATE FUNCTION ...
```

- Realice pruebas y analice su funcionamiento con el plan de ejecución.

4. (4 pts) Análisis de rendimiento

Medir y comparar el tiempo de ejecución de una consulta espacial con y sin el índice GiST. ¿Cuánto mejora?

```
-- Usa EXPLAIN ANALYZE con y sin índice
```

- Debe mostrar los resultados en un cuadro comparativo para diferentes tamaños de la tabla (20k, 40k, 60k, ...)

5. (4 pts) Detección de ciudades aisladas

Escribe una consulta que devuelva ciudades que **no tienen ninguna otra ciudad a menos de 50 km**.

- Realice pruebas y analice su funcionamiento con el plan de ejecución.

Tip: Trabajar en base al vecino más cercano de cada ciudad.

6. (6 pts) Replique el experimento en otro dataset

- Cargar en una tabla de PostgreSQL el siguiente conjunto de datos: Geolocated photographs of tourists in Cusco, Peru
- Crear un atributo geoespacial a partir de las coordenadas y aplicar un índice espacial GiST.
- Analizar el dominio de datos y proponer tres consultas espaciales relevantes para el caso de estudio, incluyendo búsquedas por rango y por proximidad (KNN).
- Mostrar evidencia del funcionamiento y un cuadro comparativo del rendimiento con y sin índice.

Conclusiones

- Ventajas de GiST:
  - Mejora significativa en rendimiento para consultas espaciales
  - Ideal para búsquedas por proximidad y distancia
  - Efectivo con grandes volúmenes de datos (>100k registros)
- Operadores Optimizados:
  - <->: Mejor rendimiento para KNN search
  - ST_DWithin: Eficiente para búsquedas por radio
  - ST_MakeEnvelope: Óptimo para búsquedas rectangulares