

Bancos de dados relacionais espaciais

Disciplina: Programação aplicada à engenharia cartográfica

Maurício C. M. de Paulo - D.Sc.

20 de fevereiro de 2026

Banco de Dados Relacional (BDR):

- Dados organizados em tabelas (relações)
- Linhas → tuplas
- Colunas → atributos
- Chaves primárias e estrangeiras

Modelo baseado em:

- Álgebra relacional
- SQL (Structured Query Language)

Exemplo de SGBD:

- **PostgreSQL database system**
- **MySQL database system**

Extensão Espacial

BDRs podem ser estendidos para armazenar geometrias.

Exemplo:

- **PostGIS** extensão do PostgreSQL

Novo tipo de dado:

- GEOMETRY
- GEOGRAPHY

Permite armazenar:

- Pontos
- Linhas
- Polígonos

E executar consultas espaciais.

Shapefile vs PostgreSQL/PostGIS vs DuckDB Spatial

	Shapefile	PostgreSQL + PostGIS	DuckDB + Spatial
Multiusuário	Não	Sim	Limitado
Transações ACID	Não	Sim	Sim
Cliente–Servidor	Não	Sim	Não
Instalação	Simples	Servidor dedicado	Arquivo único
Índice Espacial	.qix	GiST / SP-GiST	R-Tree
SQL Completo	Não	Sim	Sim
Escalabilidade	Baixa	Alta	Média

Posicionamento:

Shapefile → formato legado de troca.

PostgreSQL + PostGIS → infraestrutura robusta multiusuário.

DuckDB Spatial → banco analítico leve e embarcado.

Modelo Espacial

Cada feição espacial é composta por:

- Atributos (campos tabulares)
- Geometria (coluna espacial)

Estrutura típica:

```
        id | nome | area | geom
1 | Rio de Janeiro | 43,696 | MULTIPOLYGON(...)
```

A coluna geom armazena objetos como:

- POINT
- LINESTRING
- POLYGON

Consultas Espaciais

Além de consultas tradicionais:

- SELECT
- JOIN
- GROUP BY

Existem funções espaciais:

- ST_Buffer()
- ST_Intersects()
- ST_Contains()
- ST_Area()

Exemplo conceitual:

Selecionar municípios que intersectam um rio.

Índices Espaciais

Consultas espaciais são custosas.

Solução: Índices espaciais.

- R-Tree
- GiST (PostgreSQL)

Benefícios:

- Redução drástica de tempo de consulta
- Filtragem por bounding box

Sem índice → varredura completa da tabela.

PostgreSQL + PostGIS — Visão Geral

PostgreSQL:

- SGBD relacional cliente–servidor
- ACID completo
- Controle de concorrência (MVCC)
- Extensível

PostGIS:

- Extensão espacial do PostgreSQL
- Adiciona tipos GEOMETRY e GEOGRAPHY
- Implementa padrões OGC Simple Features

Resultado:

Banco relacional + engine espacial robusta

Conceitos Fundamentais do PostGIS

1. Tipo GEOMETRY

- Armazena POINT, LINESTRING, POLYGON, etc.
- Associado a um SRID (sistema de referência)

2. Funções Espaciais

3. Índices Espaciais

- GiST (Generalized Search Tree)
- Filtragem por bounding box
- Otimização de ST_Intersects, ST_Within, etc.

4. Modelo Cliente–Servidor

- Multiusuário
- Controle transacional
- Ideal para produção

1 Criar tabela espacial

```
CREATE TABLE municipios (  
  id SERIAL PRIMARY KEY,  
  nome TEXT,  
  populacao INTEGER,  
  geom GEOMETRY(MULTIPOLYGON, 4674)  
);
```

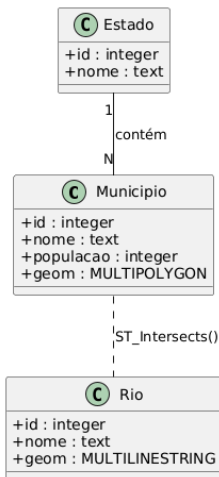
2 Criar índice espacial

```
CREATE INDEX idx_municipios_geom  
ON municipios  
USING GIST (geom);
```

Exercícios PostGIS

- 1 Instalar o PostGIS (Docker recomendado)
- 2 Criar um banco de dados
- 3 Acessar pelo QGIS
- 4 Abrir o DB Manager.
- 5 Criar uma tabela.
- 6 Importar um ShapeFile para o PostGIS.

Modelo ER Espacial



3 Municípios que intersectam um rio

```
SELECT m.nome
FROM municipios m
JOIN rios r
ON ST_Intersects(m.geom, r.geom)
WHERE r.nome = 'Rio X';
```

4 Área em km²

```
SELECT nome,
ST_Area(geom) / 1000000 AS area_km2
FROM municipios;
```

O que é DuckDB?

DuckDB é um banco de dados analítico embutido (embedded OLAP).

Características principais:

- Arquivo único (.duckdb)
- Sem servidor (in-process)
- Execução vetorizada (vectorized engine)
- Otimizado para leitura intensiva (OLAP)

Casos de uso típicos:

- Análise local de grandes datasets
- Ciência de dados
- Processamento de Parquet
- Integração com Python / R

Conceitos Arquiteturais Fundamentais

1. Embedded Database

- Executa dentro do processo Python
- Não há daemon ou serviço separado

2. Columnar Storage

- Armazenamento orientado a colunas
- Ideal para agregações e scans analíticos

3. Integração com formatos modernos

- Parquet (leitura direta)
- CSV
- Arrow
- Extensão Spatial (tipos GEOMETRY)

4. OLAP vs OLTP

- Excelente para consultas analíticas
- Não projetado para alta concorrência transacional

Exemplo — DuckDB + Spatial em Python

Instalação (uma vez):

```
pixi add duckdb
```

Exemplo em Python (parte 1):

```
import duckdb

# conecta (arquivo será criado se não existir)
con = duckdb.connect("dados.duckdb")

# instala e carrega extensão espacial
con.execute("INSTALL spatial;")
con.execute("LOAD spatial;")
```


Exemplo — DuckDB + Spatial em Python

```
# cria tabela espacial
con.execute("""
    CREATE TABLE municipios AS
    SELECT
        1 AS id,
        'Município A' AS nome,
        ST_GeomFromText(
            'POLYGON((-48 -15, -48 -16, -47 -16, -47 -15, -48 -15))'
        ) AS geom;
""")

# consulta área
result = con.execute("""
    SELECT nome,
    ST_Area(geom) AS area
    FROM municipios;
""").fetchall()

print(result)
```

Exemplo — Download de Shapefile + DuckDB Spatial

```
import requests, zipfile, io
import duckdb

# 1 Download (Natural Earth - países 110m, leve)
url = "https://naturalearth.s3.amazonaws.com/110m_cultural/ne_110m_
response = requests.get(url)
response.raise_for_status()

# 2 Extrai zip em memória
with zipfile.ZipFile(io.BytesIO(response.content)) as z:
    z.extractall("ne_countries")

# 3 Conecta ao DuckDB
con = duckdb.connect("world.duckdb")
con.execute("INSTALL spatial;")
con.execute("LOAD spatial;")
```

Exemplo — Download de Shapefile + DuckDB Spatial

```
# 4 Lê shapefile direto via GDAL
con.execute("""
    CREATE TABLE countries AS
    SELECT * FROM ST_Read(
        'ne_countries/ne_110m_admin_0_countries.shp');
""")

# 5 Calcula área (m² | CRS 4326->3857)
result = con.execute("""
    SELECT NAME,
    ST_Area(ST_Transform(geom, 3857)) AS area
    FROM countries
    ORDER BY area DESC
    LIMIT 5;
""").fetchall()

for row in result:
    print(row)
```