GEO-SERIES | EPISODE #13

Technical Session

Handling large spatial data using PostgreSQL and DuckDB

Speaker

Er. Kshitij Raj Sharma

Open Source Geospatial Developer,
Product Owner (Map Data Access Services) at
Humanitarian OpenStreetMap Team (HOTOSM)

Kshitij Raj Sharma is a product owner for map data access services at HOTOSM from Nepal, He is a passionate spatial developer with a love for open-source software and open data. His expertise in spatial data and deep interest in mapping led him to explore the potential of AI. Kshitij has been experimenting and advocating for FOSS and contributing to open data initiatives since past seven years . He is also a maintainer of several opensource tools including Free and open source AI tool: fAIr, OSM Export Tool, Raw Data API, geojson2osm, OSMSG, etc and an integral member of our own OSGEO Nepal

Moderator

Er. Tek Kshetri

M.Sc. at University of Calgary

26th April, 2024

7:00 PM-7:45 PM (NST)

NEPAL GEOMATICS ENGINEERING SOCIETY

OSGeo Nepal

Organizers

Handling Large Spatial Data using PostgreSQL and DuckDB

Kshitij Raj Sharma

Contents

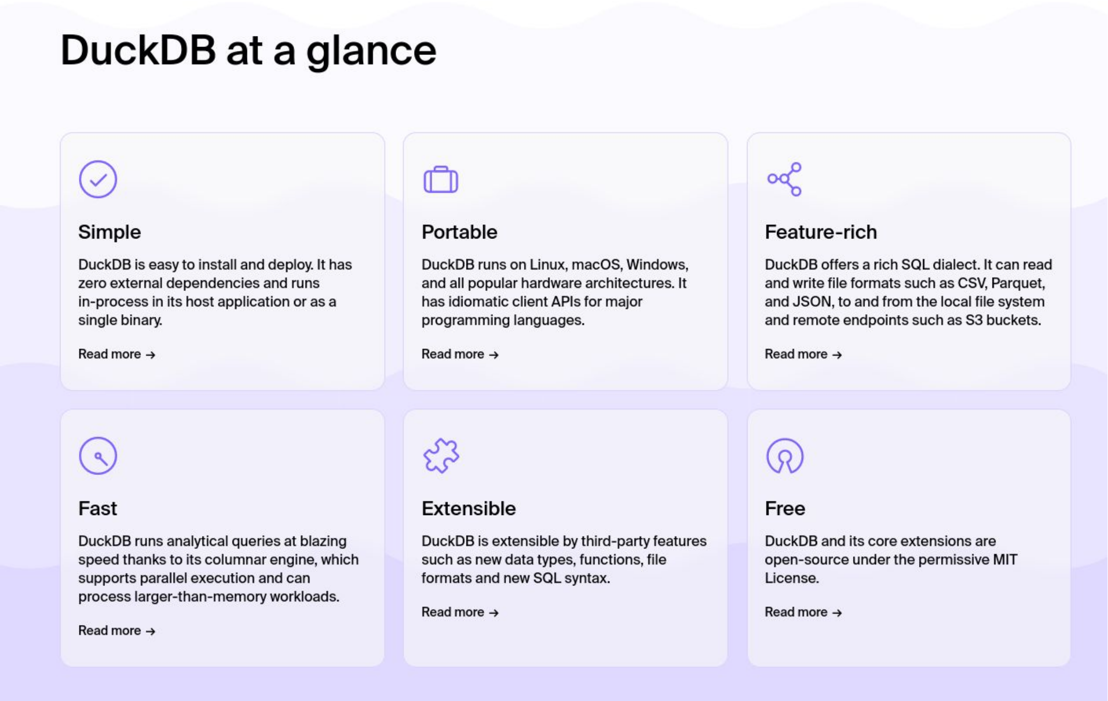
1. Introduction
2. Postgresql
3. DuckDB
4. Handling large spatial data
5. Experiments
6. DuckDB Usage
7. Walkthrough & Demo
8. Contact

PostgreSQL

- Most popular open source database
- Reliable
- Feature Rich
- Powerful indexes
- Large dev support group
- Production ready database support in multiple commercial cloud service provider
- Rich in spatial extensions like postgis , pg_raster

DuckDB

https://duckdb.org/



DuckDB at a glance

Simple

DuckDB is easy to install and deploy. It has zero external dependencies and runs in-process in its host application or as a single binary.

Read more →

Portable

DuckDB runs on Linux, macOS, Windows, and all popular hardware architectures. It has idiomatic client APIs for major programming languages.

Read more →

Feature-rich

DuckDB offers a rich SQL dialect. It can read and write file formats such as CSV, Parquet, and JSON, to and from the local file system and remote endpoints such as S3 buckets.

Read more →

Fast

DuckDB runs analytical queries at blazing speed thanks to its columnar engine, which supports parallel execution and can process larger-than-memory workloads.

Read more →

Extensible

DuckDB is extensible by third-party features such as new data types, functions, file formats and new SQL syntax.

Read more →

Free

DuckDB and its core extensions are open-source under the permissive MIT License.

Read more →

- Spatial Support

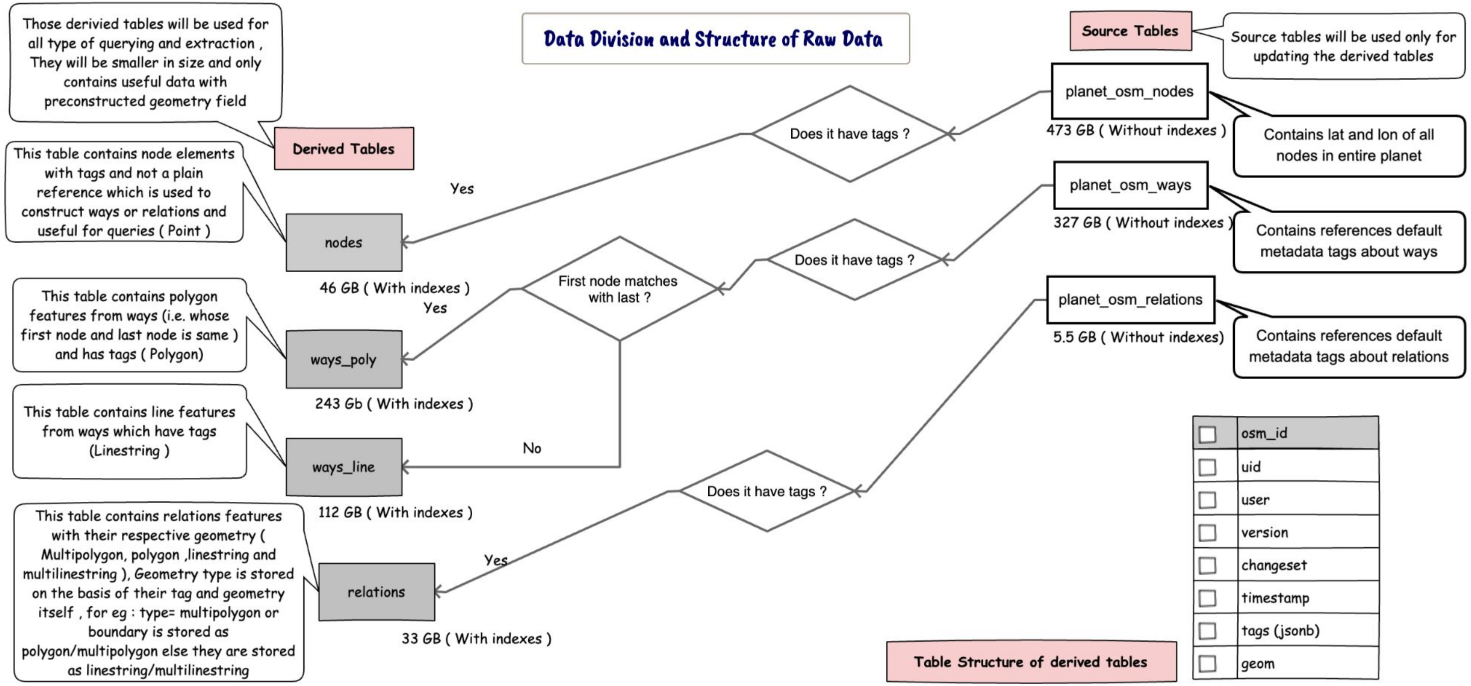
Nepal

Handling Large spatial data

Tips and Tricks :

- Understanding and organizing data
- Processing raw data
- Choice of database based on need
- Schema
- Managing Indexes
- Query plan
- Scalability

Experiments



Data Division and Structure of Raw Data

Source Tables

Source tables will be used only for updating the derived tables

planet_osm_nodes

473 GB (Without indexes)

Contains lat and lon of all nodes in entire planet

planet_osm_ways

327 GB (Without indexes)

Contains references default metadata tags about ways

planet_osm_relations

5.5 GB (Without indexes)

Contains references default metadata tags about relations

Derived Tables

Those derivied tables will be used for all type of querying and extraction , They will be smaller in size and only contains useful data with preconstructed geometry field

This table contains node elements with tags and not a plain reference which is used to construct ways or relations and useful for queries (Point)

nodes

46 GB (With indexes)

This table contains polygon features from ways (i.e. whose first node and last node is same) and has tags (Polygon)

ways_poly

243 Gb (With indexes)

This table contains line features from ways which have tags (Linestring)

ways_line

112 GB (With indexes)

This table contains relations features with their respective geometry (Multipolygon, polygon ,linestring and multilinestring), Geometry type is stored on the basis of their tag and geometry itself , for eg : type= multipolygon or boundary is stored as polygon/multipolygon else they are stored as linestring/multilinestring

relations

33 GB (With indexes)

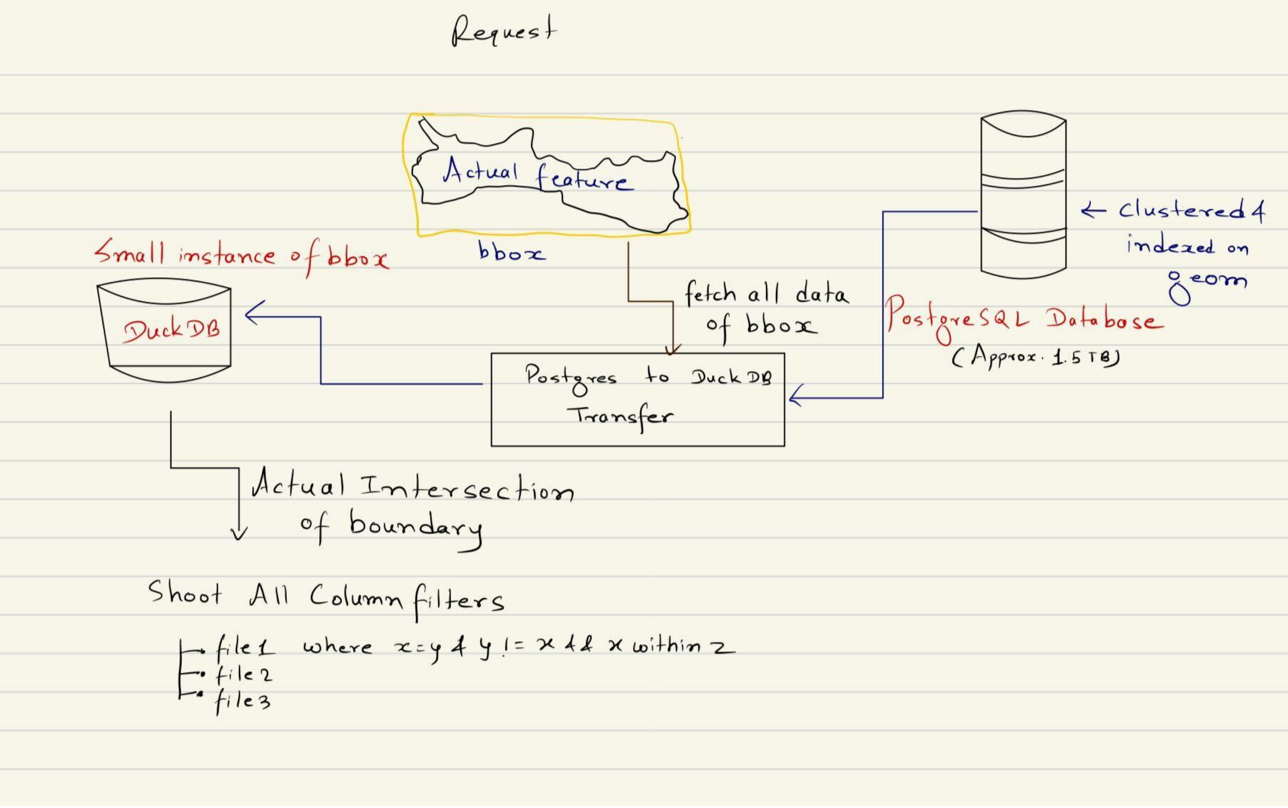
Table Structure of derived tables

□	osm_id
□	uid
□	user
□	version
□	changeset
□	timestamp
□	tags (jsonb)
□	geom

DuckDB Usage

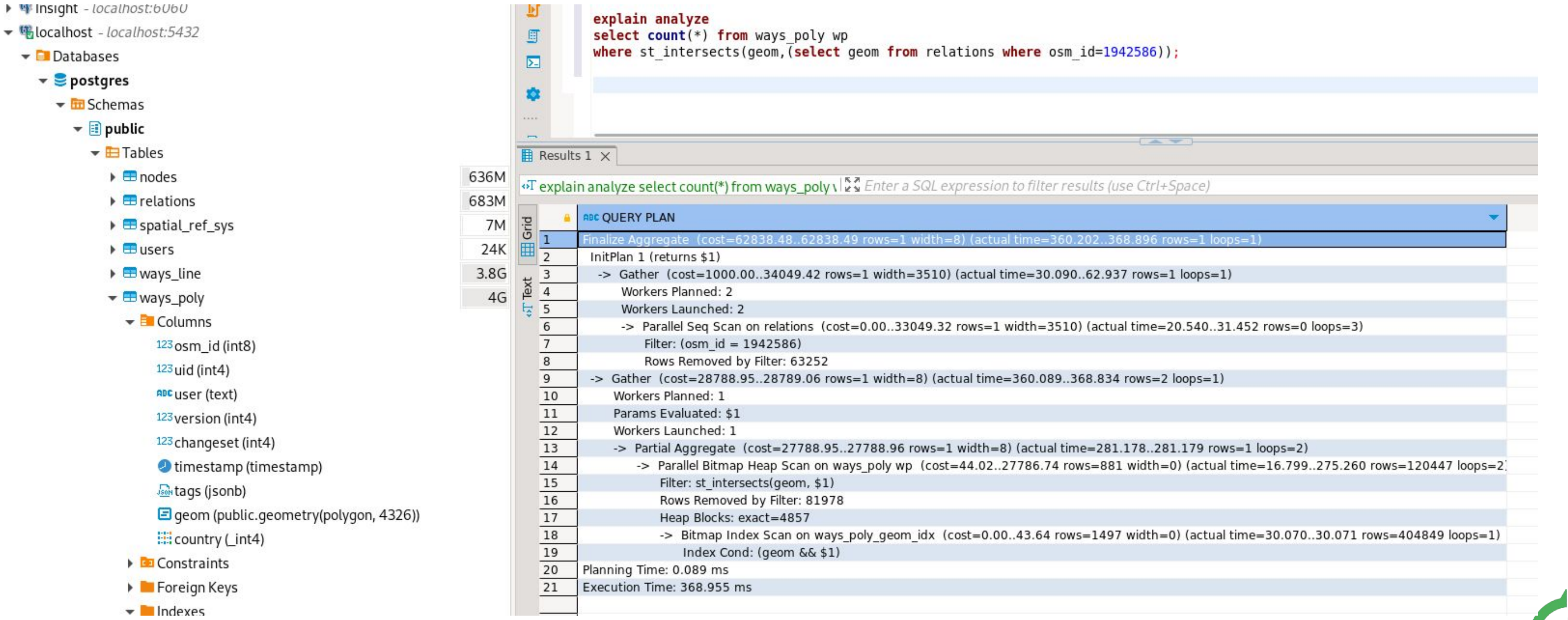
- DuckDB is really useful when you want to do quick spatial operations in large files in your disk without having to setup whole database
- Faster columnar query as compared to postgresql without indexes
- Relatively new and doesn’t contain lots of spatial datasets
- Currently focused on vector datasets
- Supports both spatial and non spatial query
- Has support of OGR formats enabling larger GIS format export support directly from the database
- Can be useful on google buildings , overture datasets / your custom csv datasets

PostgreSQL + DuckDB



Nepal

Walkthrough & Demo



```sql
explain analyze
select count(*) from ways_poly wp
where st_intersects(geom,(select geom from relations where osm_id=1942586));
```

explain analyze select count(*) from ways_poly \

```
Finalize Aggregate  (cost=62838.48..62838.49 rows=1 width=8) (actual time=360.202..368.896 rows=1 loops=1)
  InitPlan 1 (returns $1)
    ->  Gather  (cost=1000.00..34049.42 rows=1 width=3510) (actual time=30.090..62.937 rows=1 loops=1)
          Workers Planned: 2
          Workers Launched: 2
          ->  Parallel Seq Scan on relations  (cost=0.00..33049.32 rows=1 width=3510) (actual time=20.540..31.452 rows=0 loops=3)
                Filter: (osm_id = 1942586)
                Rows Removed by Filter: 63252
  ->  Gather  (cost=28788.95..28789.06 rows=1 width=8) (actual time=360.089..368.834 rows=2 loops=1)
        Workers Planned: 1
        Params Evaluated: $1
        Workers Launched: 1
        ->  Partial Aggregate  (cost=27788.95..27788.96 rows=1 width=8) (actual time=281.178..281.179 rows=1 loops=2)
              ->  Parallel Bitmap Heap Scan on ways_poly wp  (cost=44.02..27786.74 rows=881 width=0) (actual time=16.799..275.260 rows=120447 loops=2
                    Filter: st_intersects(geom, $1)
                    Rows Removed by Filter: 81978
                    Heap Blocks: exact=4857
                    ->  Bitmap Index Scan on ways_poly_geom_idx  (cost=0.00..43.64 rows=1497 width=0) (actual time=30.070..30.071 rows=404849 loops=1)
                          Index Cond: (geom && $1)
Planning Time: 0.089 ms
Execution Time: 368.955 ms
```

Nepal

Contact Me :

Kshitij Raj Sharma

LinkedIn / GitHub / Mastodon :
@kshitijrajsharma

Email : skshitizraj@gmail.com