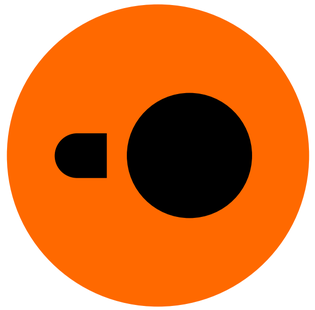


Spatial Analytics W/ DuckDB

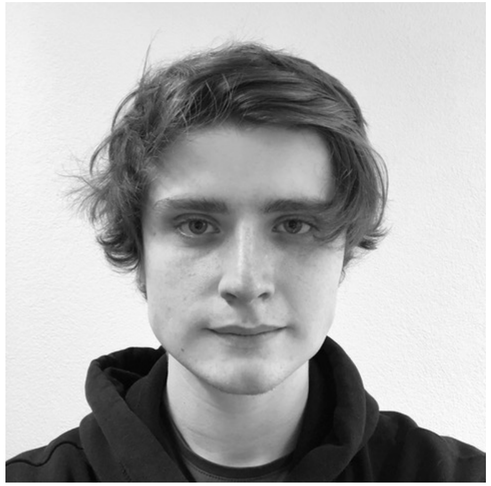
Max Gabrielsson
Software Engineer

DuckDB Labs

About me

Max Gabrielsson

- BsC from Uppsala University
- Fan turned Soft. Engineer @ DuckDB Labs
- All over the place & Spatial Extension



DuckDB Labs

- Founded by DuckDB creators
- Based in Amsterdam, ≈18 people
- Development & Support for DuckDB

DuckDB Labs

What is DuckDB?

- Analytical embedded SQL Database
- Originally research project @ CWI
- Free and open source (MIT)
- No external dependencies
- Version 1.0.0 as of a week ago 🎉
- Extensive python/client integrations
- Makes the most of your laptop

```
$ pip install duckdb
```

In-process

Portable

Fast

Open-source

The three sides of DuckDB

Execution

(Its fast)

Storage

(Its efficient)

UX

(Its friendly)

Execution: Query Processing

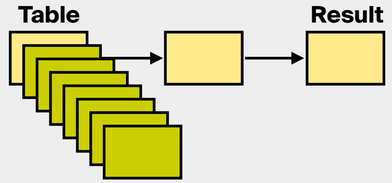
Row-at-a-Time

- Classic Database approach
- Low memory footprint
- High CPU overhead

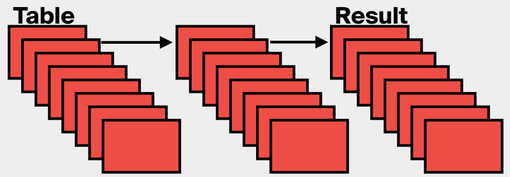
Column-at-a-Time

- Common in Dataframes
- Efficient CPU usage - SIMD
- Large memory footprint

Row-at-a-Time



Column-at-a-Time

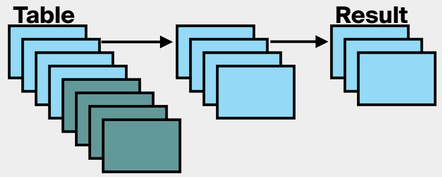


Execution: Query Processing

DuckDB is *Vector-at-a-Time*

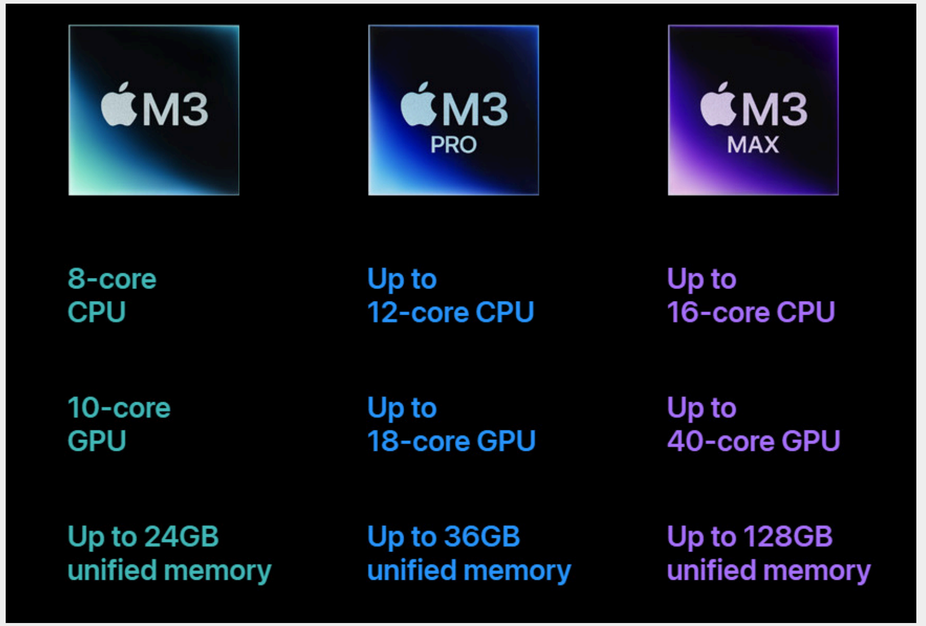
- Best of both worlds!
- Optimize for CPU-Cache

Vectorized Processing



DuckDB is *Multi-threaded*

- Natural to parallelize over vectors
- Parallelism is increasingly important!



Storage: Why it matters

- DuckDB is not an in-memory DB
- 1 Database = 1 File
- Updates, “ACID”, Transactions
- Multiple tables in one database
- Table columns stored individually
  - Fast search w/ per-column statistics
  - Compression: save 3-5x on storage!
  - Paradoxically improves query speed

Example: 1TB table with 100 columns

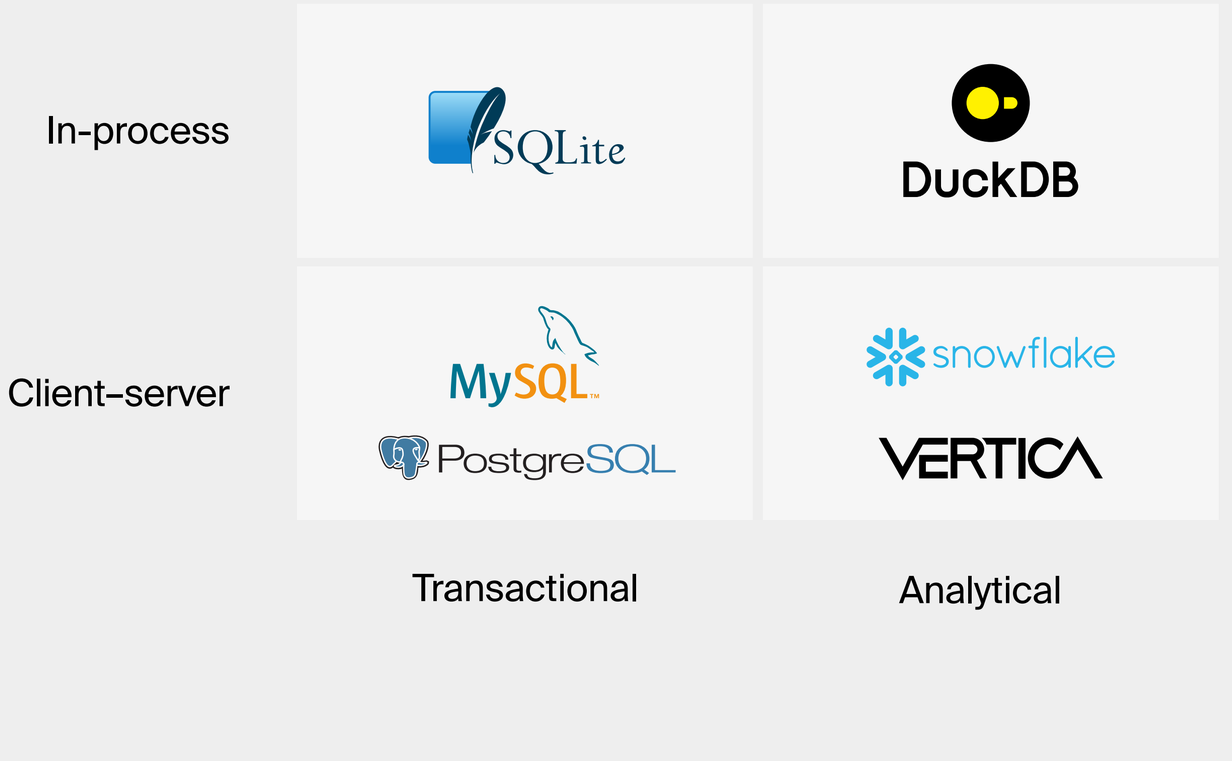
- Column-store:
  Read 5 columns (50GB) from disk ≅ 8 minutes
- Row-store:
  Read entire 1TB of data from disk at 100MB/s ≅ 3 hours

UX: In Process Deployment

- No separate server
- No config, no env, no docker
- Minimal transfer delay/overhead
- Run DuckDB as part of larger system
  - In the backend
  - In the client
  - In the browser (WASM)
  - In your phone

```
$ pip install duckdb
```

```
$ brew install duckdb
```

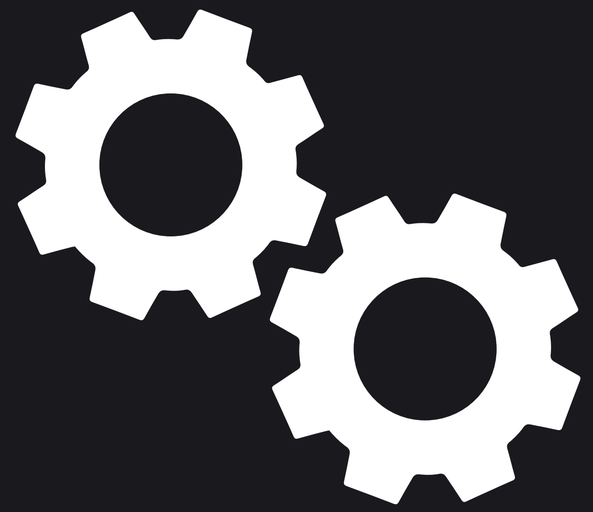


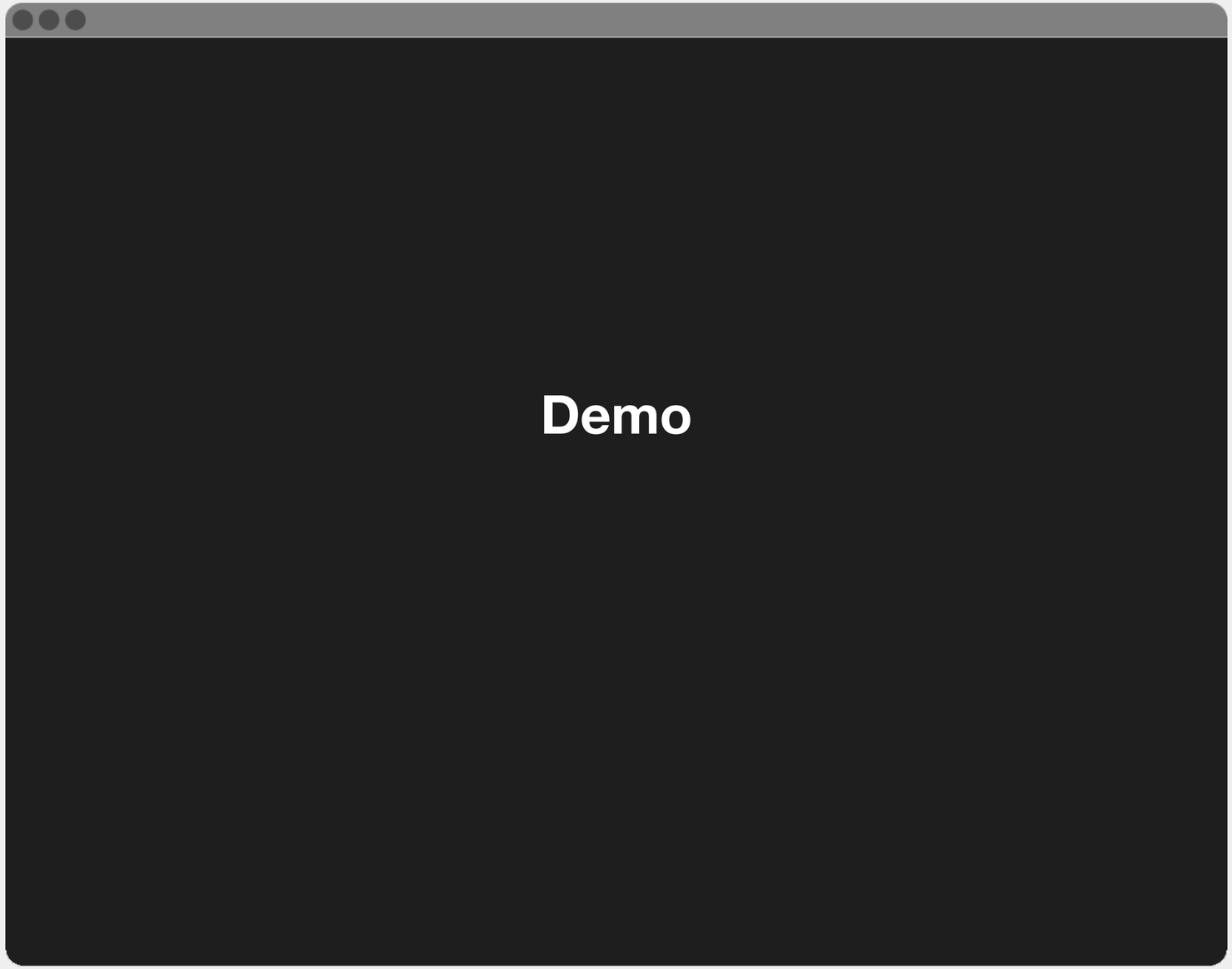
UX: “Friendly SQL”

- Superset of PostgreSQL dialect
- FROM-first syntax, trailing commas
- GROUP BY ALL, column selection
- Nested types, lambda functions

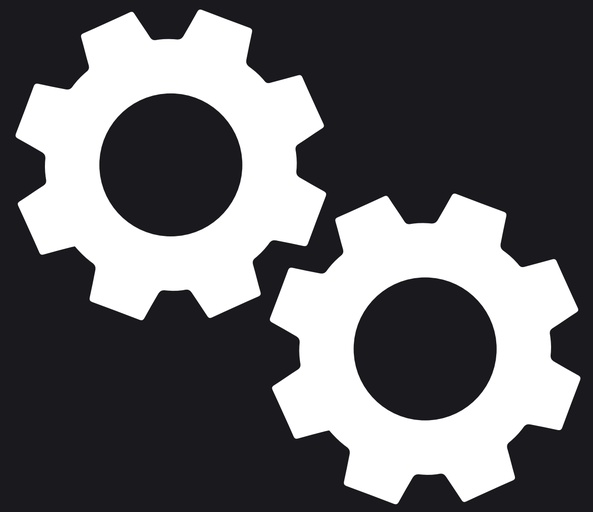
```
-- FROM-first syntax!
FROM my_table SELECT 1 + 2;

-- Trailing commas
SELECT a, b, c, FROM my_table;

-- Column selection
SELECT * EXCLUDE (id, name) FROM users

-- List comprehensions
SELECT [x + 1 for x in [1, 2, 3]];

- Unified function call syntax
SELECT name.upper() FROM users
```

SQL Refresher

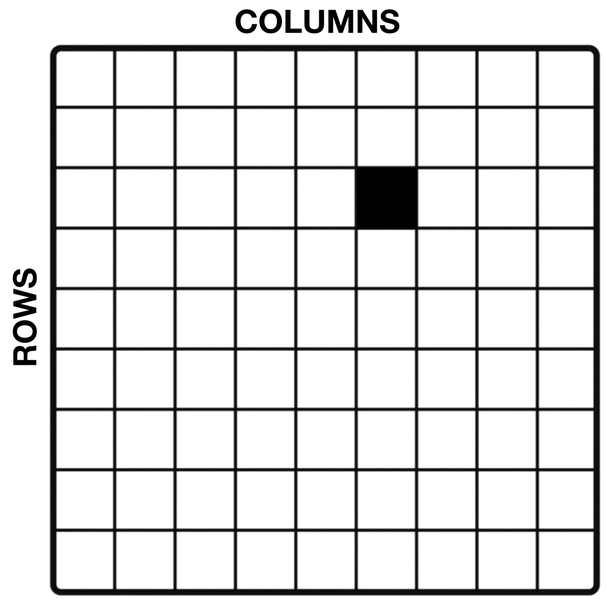



Demo

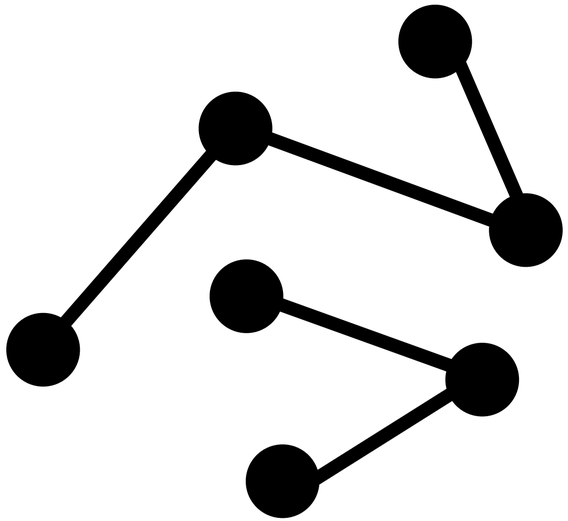
Geospatial Analytics



Raster vs Vector data

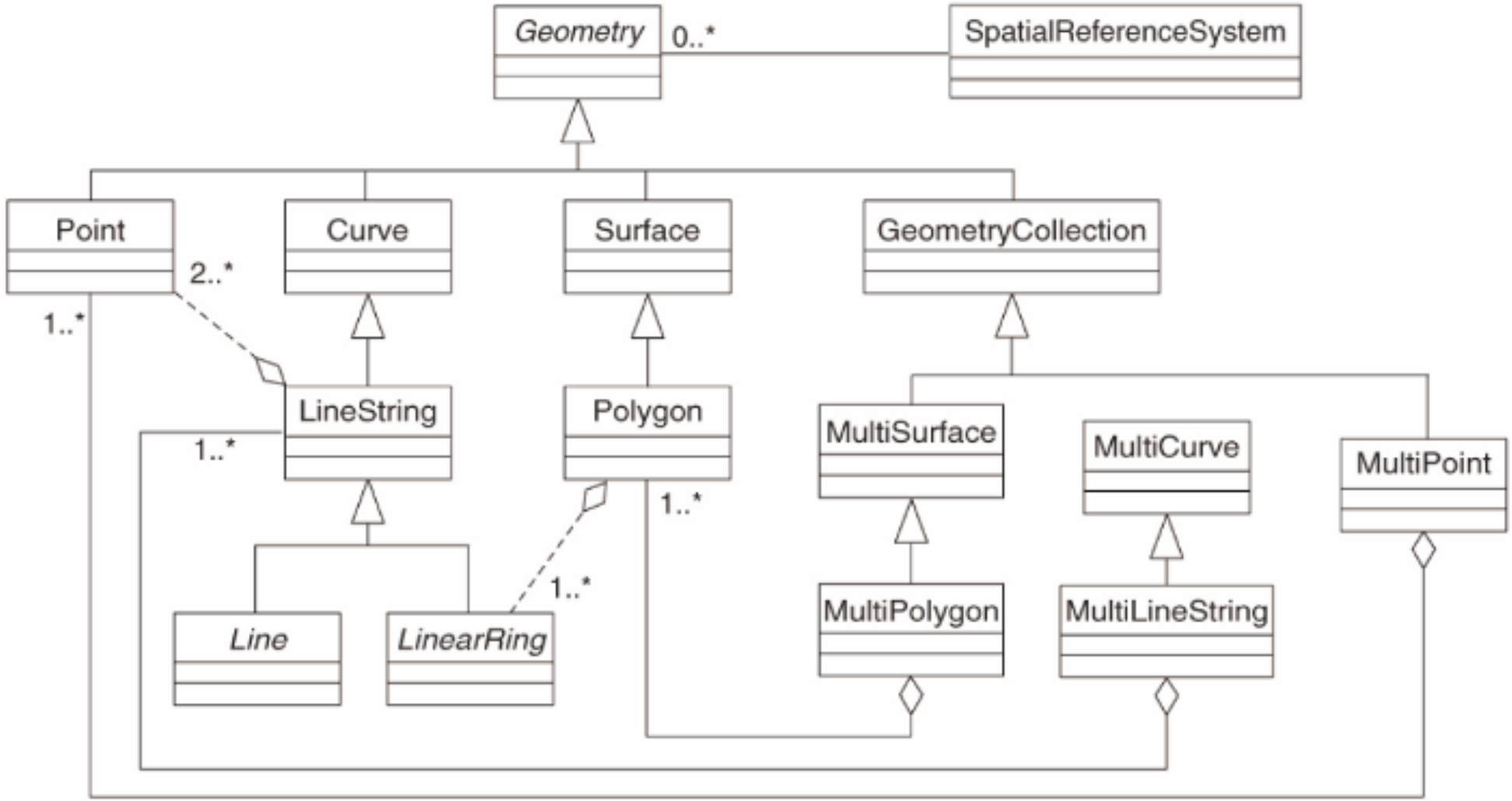


RASTER



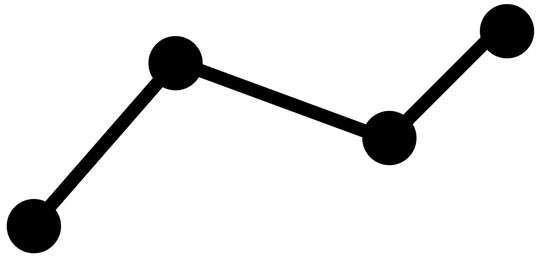
VECTOR

Simple Features (Vector Data)

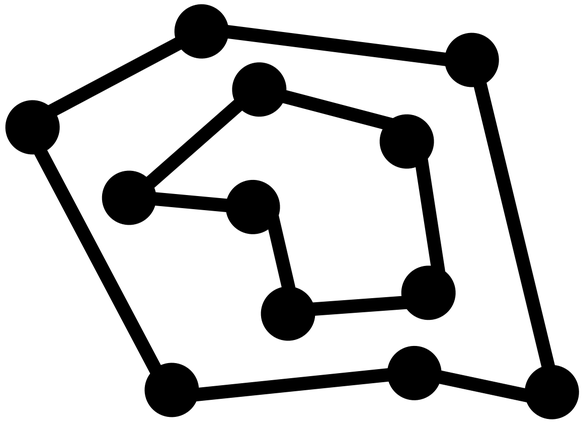


What is a “GEOMETRY” really?

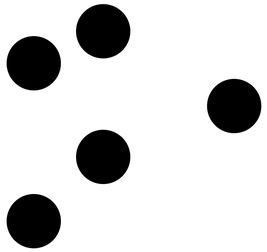
POINT



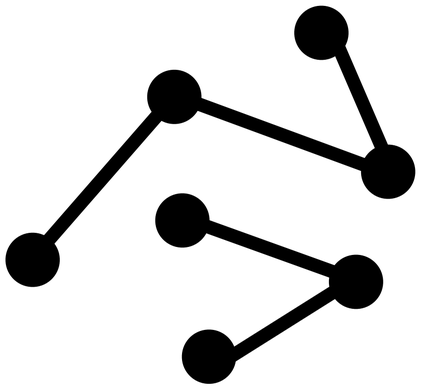
LINESTRING



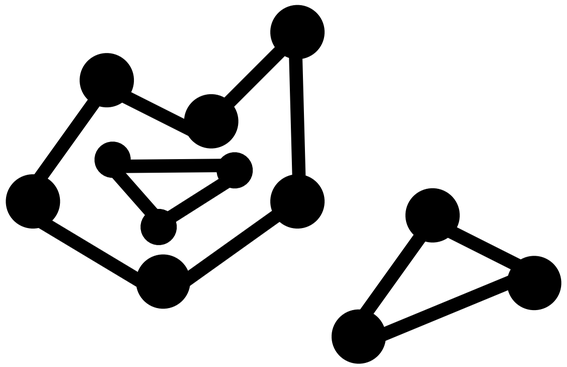
POLYGON



MULTIPOINT

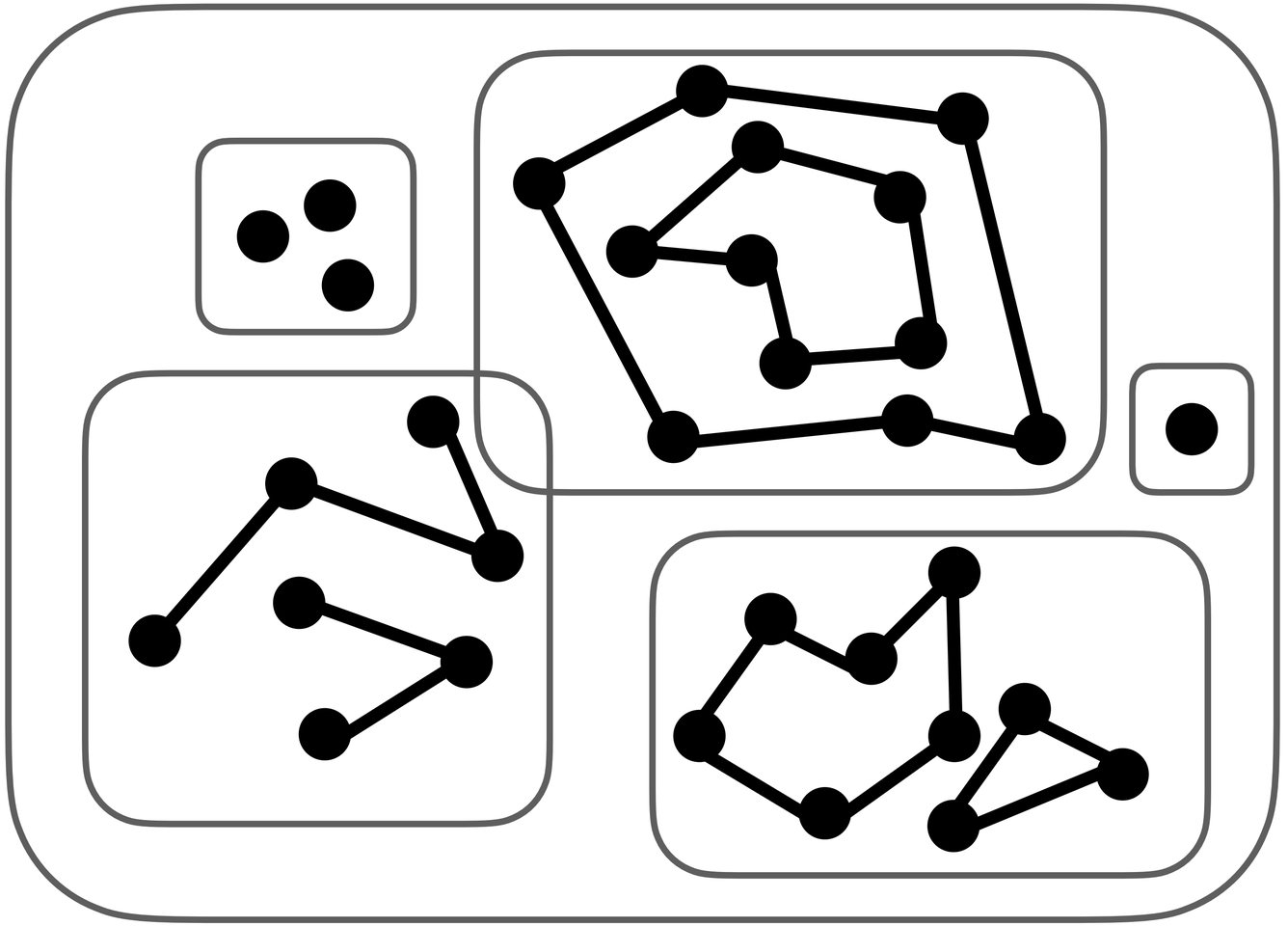


MULTILINESTRING



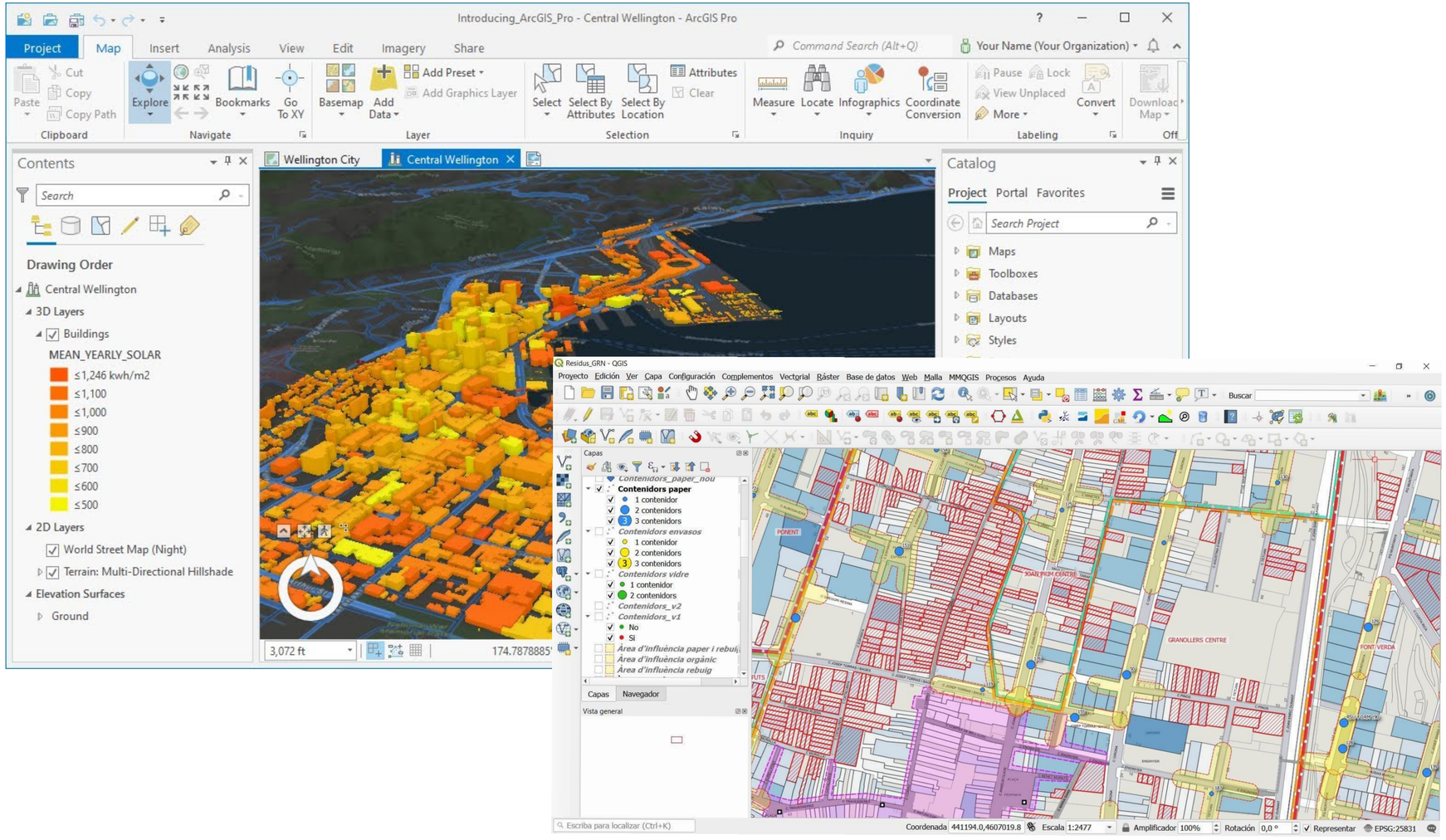
MULTIPOLYGON

What is a “GEOMETRY” really?



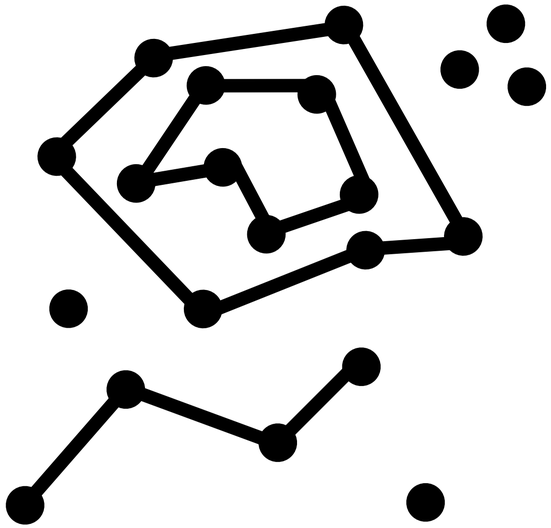
GEOMETRYCOLLECTION

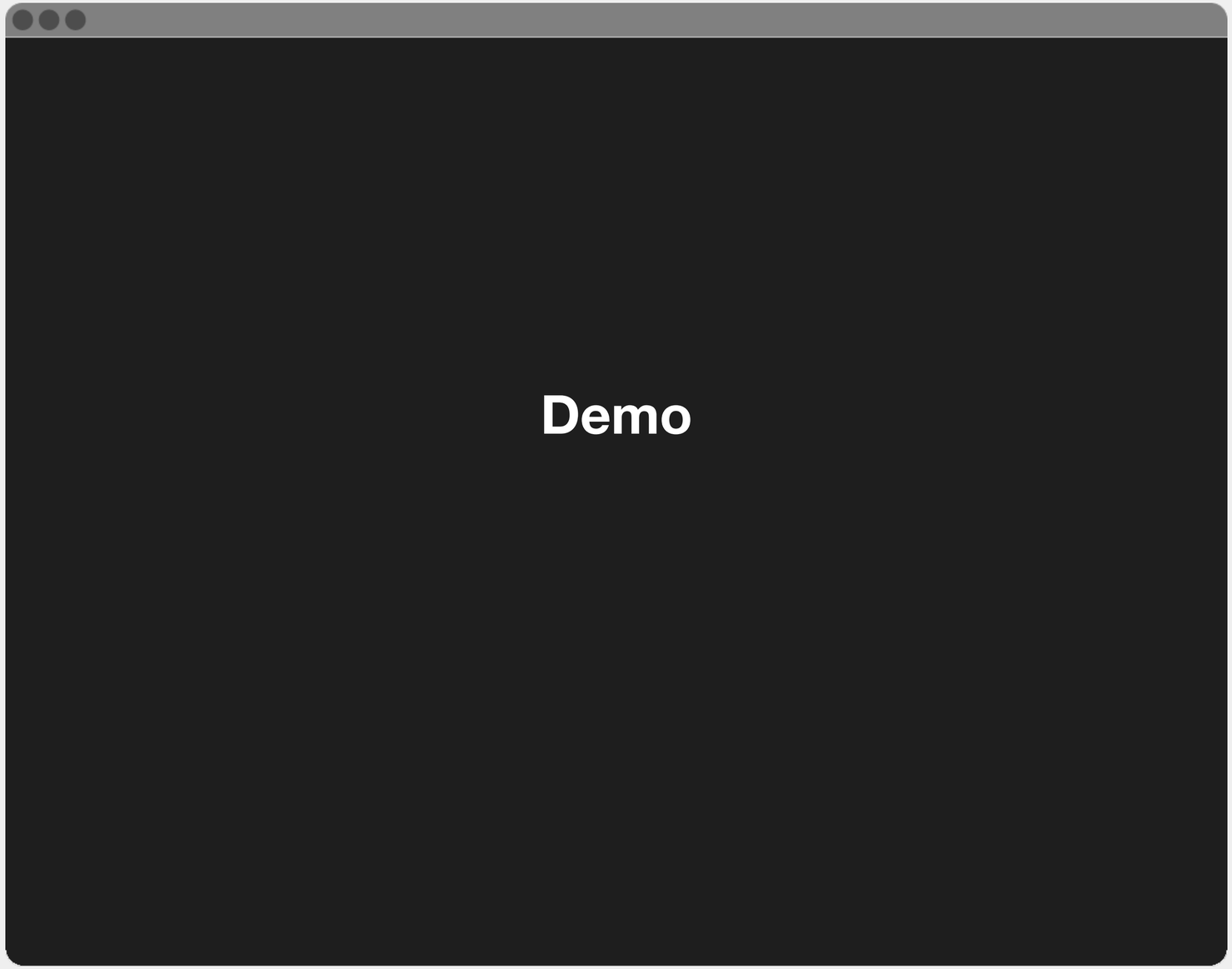
Geographic Information Systems (GIS)



DuckDB Spatial

- DuckDB supports “extensions” = plugins
- “Spatial” officially supported adding geospatial capabilities
- Adds “Simple Features” vector **`GEOMETRY`** type
- No runtime dependencies
- Modeled after PostGIS
- Adds 100+ spatial “ST_” functions for use in SQL




Demo

Coordinate Systems

- The world isn’t flat
  - But its really useful to pretend it is!
- **Problem:**
  - Inaccurate at large scales
  - More distortion further from the equator

thetruesize.com

Coordinate Systems

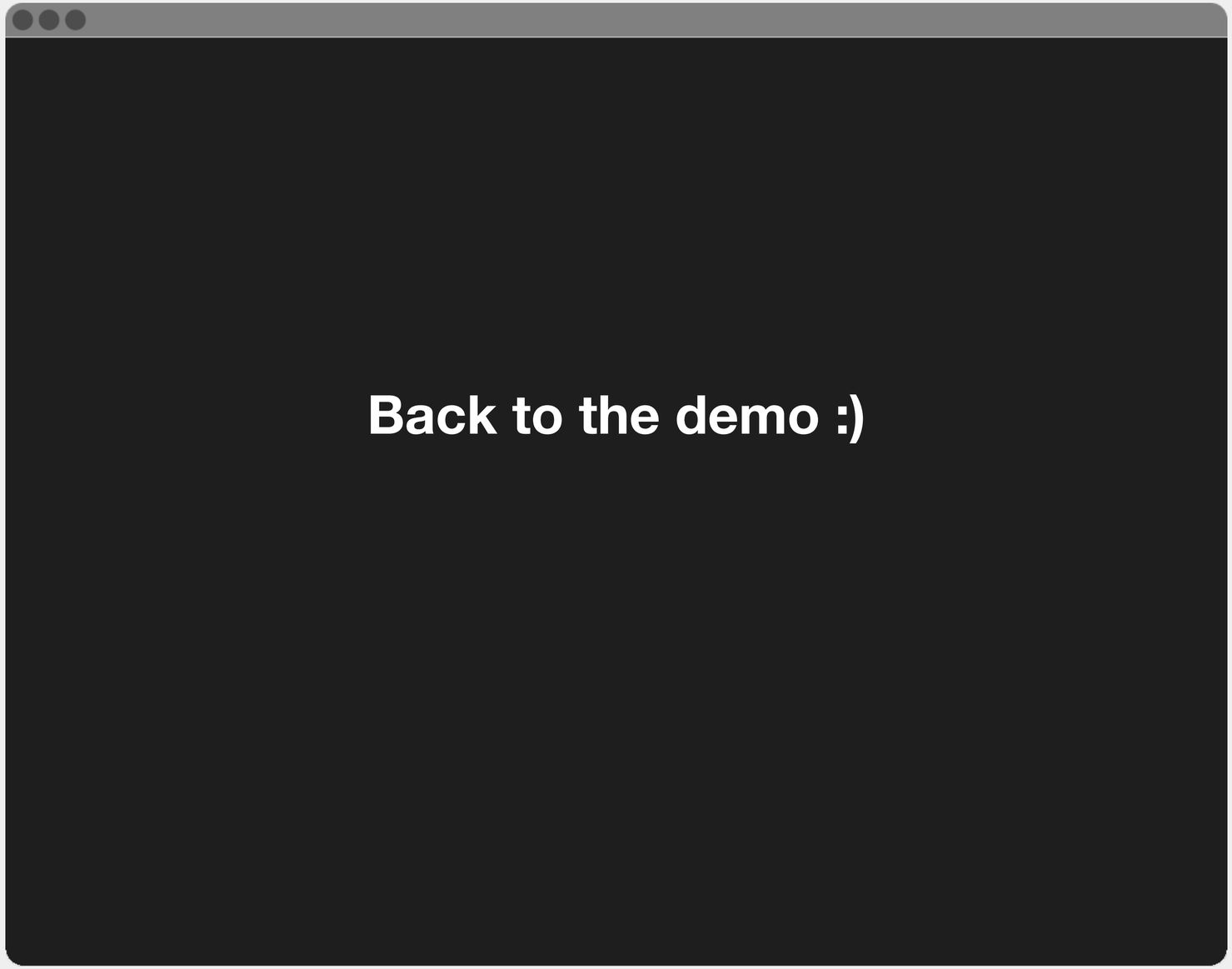
- **Solution: Use another “CRS”**
- Defined for a specific area of interest
- Mathematical expression on how to “project”
- Referred to with shorthand, “SRID”
  - **EPSG:4326/WGS84** = “latitude/longitude”, deg
  - **EPSG:3857** = All web-based maps, meters
  - **EPSG:28992** = Netherlands, meters

EPSG:28992 Area

Coordinate Systems

- **Good news!**
  - DuckDB recognizes 3000+ coordinate systems by name!
- **Bad news!**
  - Most systems* always assume **Longitude, Latitude** “axis order”

    ... ignoring the CRS authority.
  - E.g. GeoJSON is always WGS84 but Lon/Lat.


Back to the demo :)

How is this relevant?

Todays Challenge

Datasets: Amsterdam Pedestrian Road Network

- Extracted from OpenStreetMap
- Roads split into segments at every intersection
- Additional properties at each segment
  - Bridge, Tunnel, Road type, Length
- Forms a routable graph
- **Note: DuckDB can’t do “routing”!**

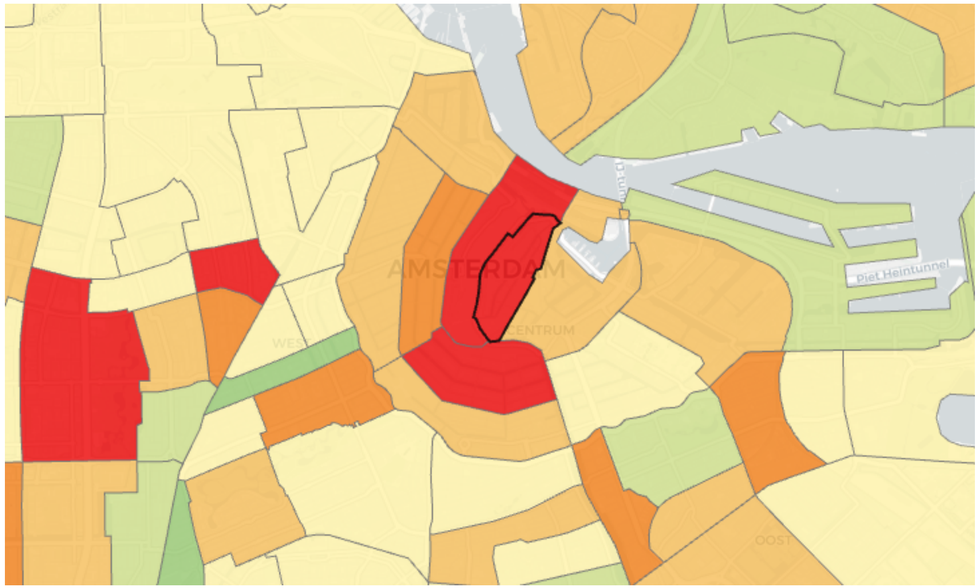
Datasets: Amsterdam Open Data: Maps

- Administrative zone polygons
  - Wijken (districts)
  - Buurten (neighborhoods)
- Positions of street lamps
- Positions of CCTV cameras
- Tons of other stuff to explore!
- All available as GeoJSON

Datasets: Amsterdam Open Data: Crime

- Multiple Datasets Per Wijk
  - Registered crime
  - Experience victimization
  - Experience nuisance
  - Insecurity experience
- Available as CSV/XLSX

  Except Burgwallen
- **No Wijk (district) is “worst” in all categories***

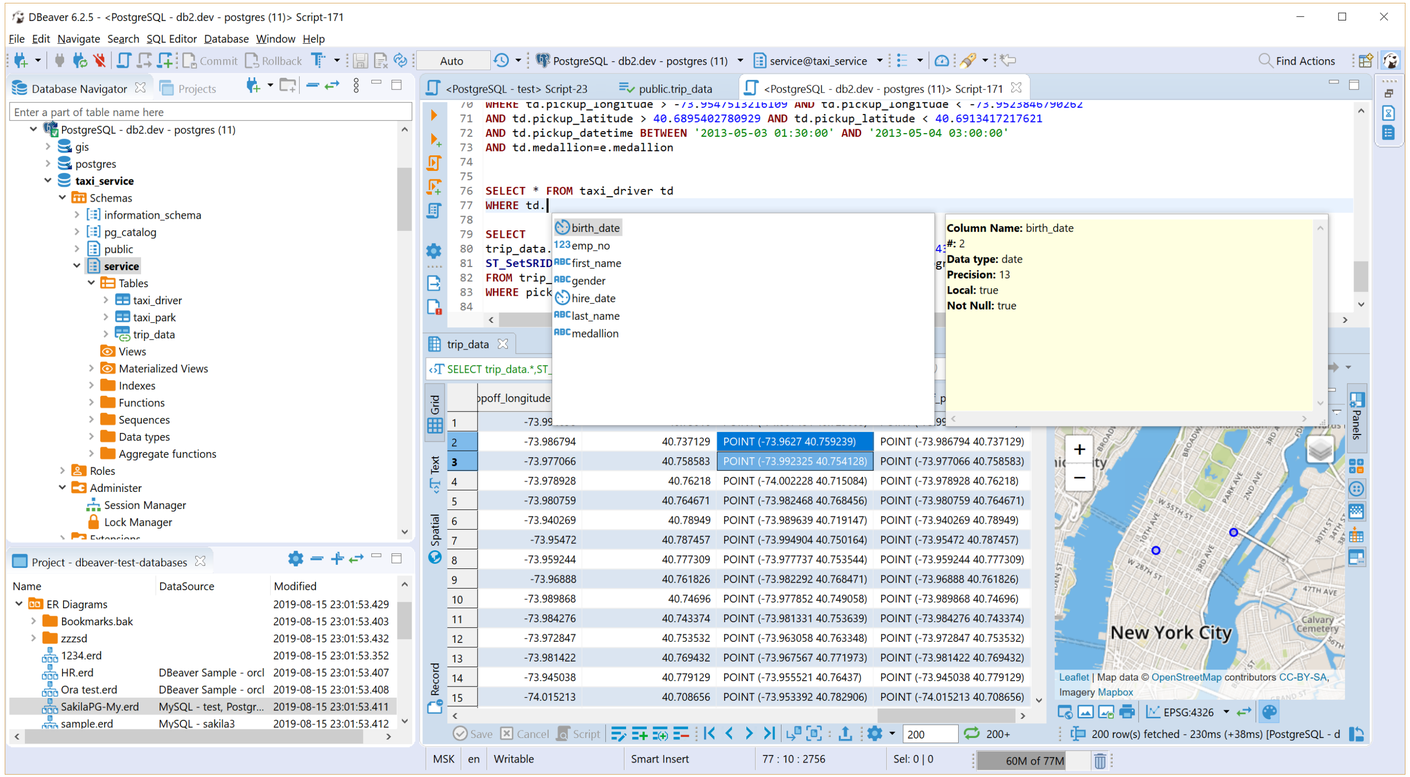


Datasets: Overture Maps

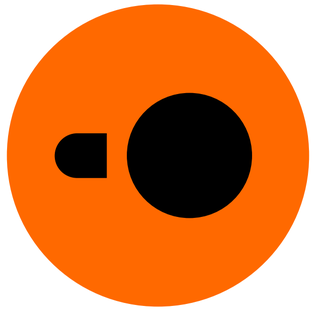
- “Curated”/processed distribution of OpenStreetMap data
- Backed by big tech, worldwide coverage, multiple “themes”
  - **Base:** water, land-use, land-cover, infrastructure
  - **Buildings:** polygons for buildings
  - **Places:** Points for facilities, amenities
- Slightly trickier to access but possible through DuckDB!
- Good as base data for creating your own maps

Note: Visualization

- **For project:**
  - DuckDB supports lots, but I recommend **GeoJSON**
    - Not very efficient, but simple
  - Web maps: Leaflet, OpenLayers
- **During Development:**
  - VSCode: Geo Data Viewer
  - GUI: DBeaver

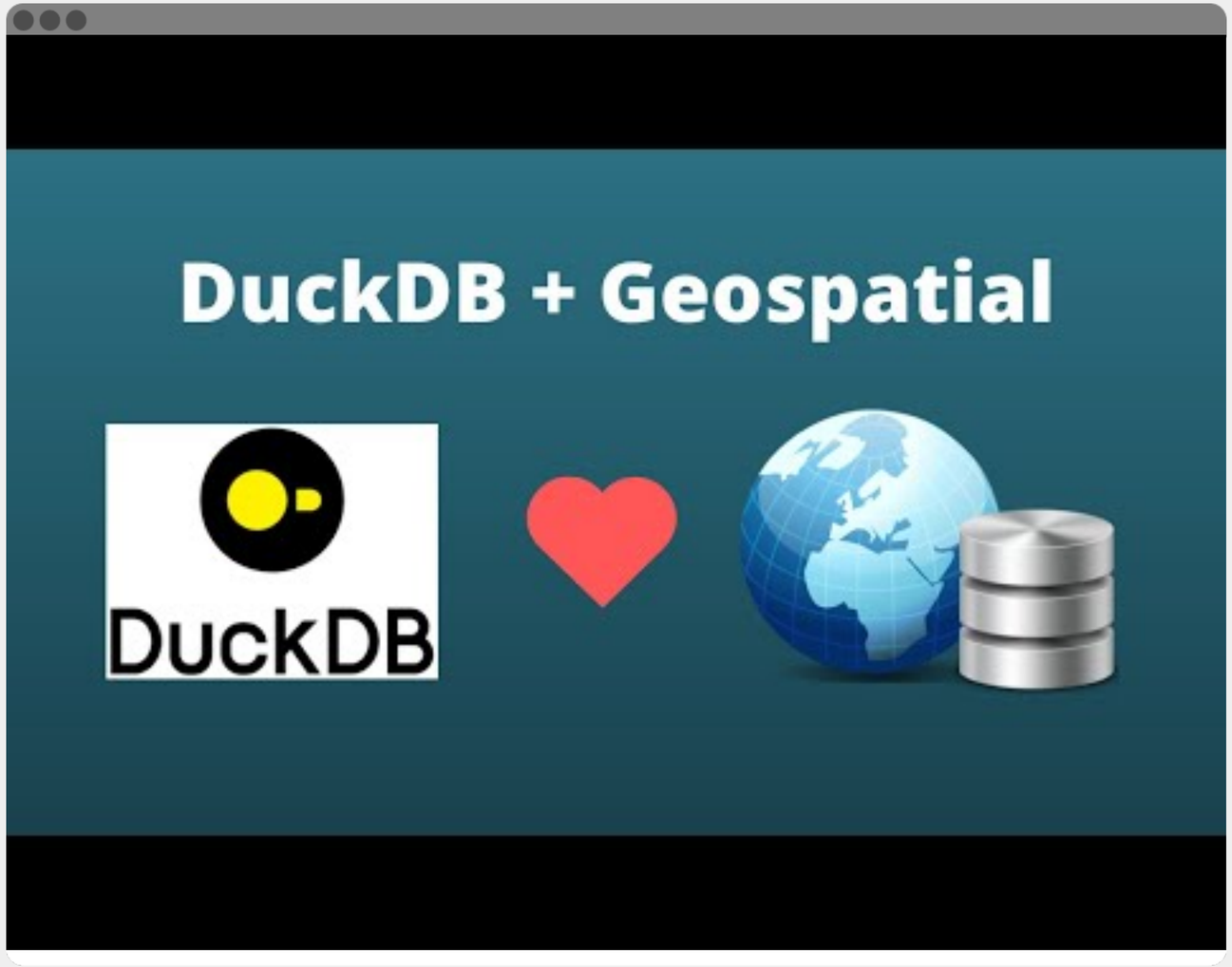



www.github.com/Maxxen/hack4her



Dr. Qiusheng Wu's Spatial Data Management

- University of Tennessee GEOG-414
- Course material available online
- Web-book + YouTube videos
- Python, GeeMap, DuckDB, PostGIS



https://geog-414.gishub.org/