PostGIS and PostgreSQL

GIS Data, Queries, and Performance

RustProof Labs

bringing you data

Ryan Lambert

- **RustProof Labs**
- Author: *Mastering PostGIS and OpenStreetMap* (https://postgis-osm.com/)
- Blog: https://blog.rustprooflabs.com/

Ryan Lambert

- **RustProof Labs**
- Author: *Mastering PostGIS and OpenStreetMap* (https://postgis-osm.com/)
- Blog: https://blog.rustprooflabs.com/

Day Job

- Director, Data Science and Institutional Research
- MS SQL Server

Julie Lambert

- **RustProof Labs**
- Director - Drone Division
- RPiC (Remote Pilot in Control)

Places to Find Us

- RustProof Labs Blog
- Mastodon
- Discord - People, Postgres, Data

https://mastodon.social/@rustprooflabs

https://discord.com/channels/710918545906597938/953833675655639050/953833703124135946

Today’s Agenda

- PostGIS Intro
- Data Sources and Formats
- Spatial Joins, Buffers, and Common Operations
- SRIDs, Explain, and Hexes
- Routing
- Configuration and Nuances

Resources Available

https://blog.rustprooflabs.com/2023/11/pass-2023-precon--gis-queries-performance

Participate Your Way

- Listen and Learn
- Follow along using Demo DB
- Follow along on your hardware

Demo Database: OpenStreetMap

- General Washington details
- Detailed Seattle and Spokane schemas
- Routing schema (Seattle roads)

Demo Database: OpenStreetMap

- General Washington details
- Detailed Seattle and Spokane schemas
- Routing schema (Seattle roads)

- 175 MB (gzipped)
- 639 MB (unzipped)
- 787 MB (in Postgres)

PostGIS Advantages

- Data Types
- Indexes
- Spatial Analysis
- You already know SQL!

Spatial data is Special!

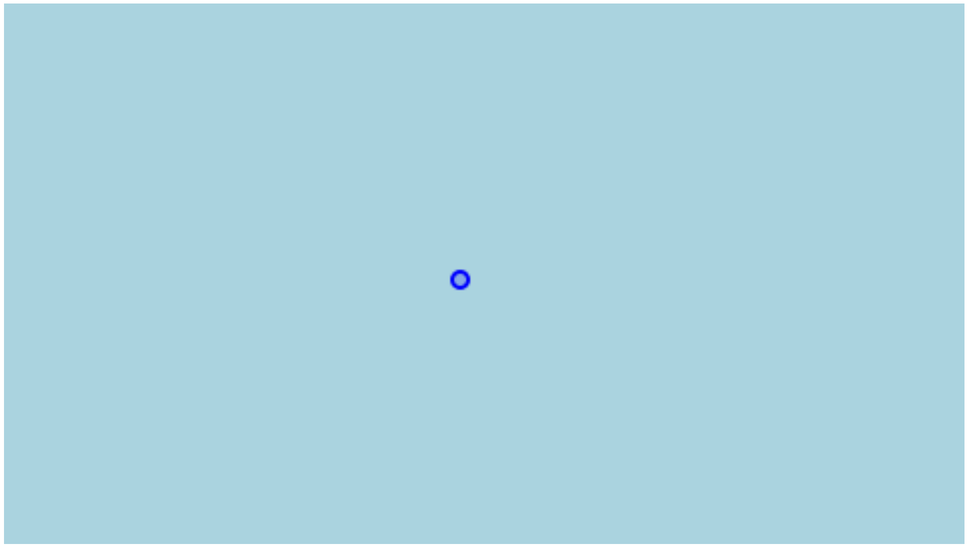
Spatial data is Special!

Is it?

Spatial data without attributes…

Spatial data without attributes...

are just shapes!



Attributes are key to spatial data

- `BIGINT`
- `NUMERIC`
- `TEXT`
- `TIMESTAMPTZ`
- `JSONB`

Spatial data is just data!

Spatial data example

Spatial data example

osm_id	osm_type	name	geom
29546940	city	Seattle	POINT(-122.33 47.60)

© OpenStreetMap Contributors

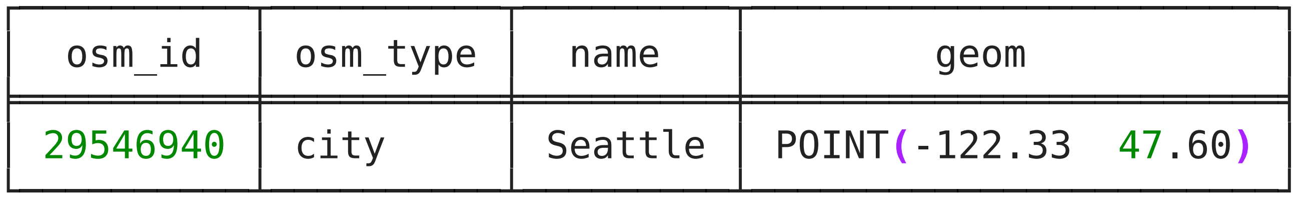
Use the right tools

Use the right tools

> `psql` *is great, but...*

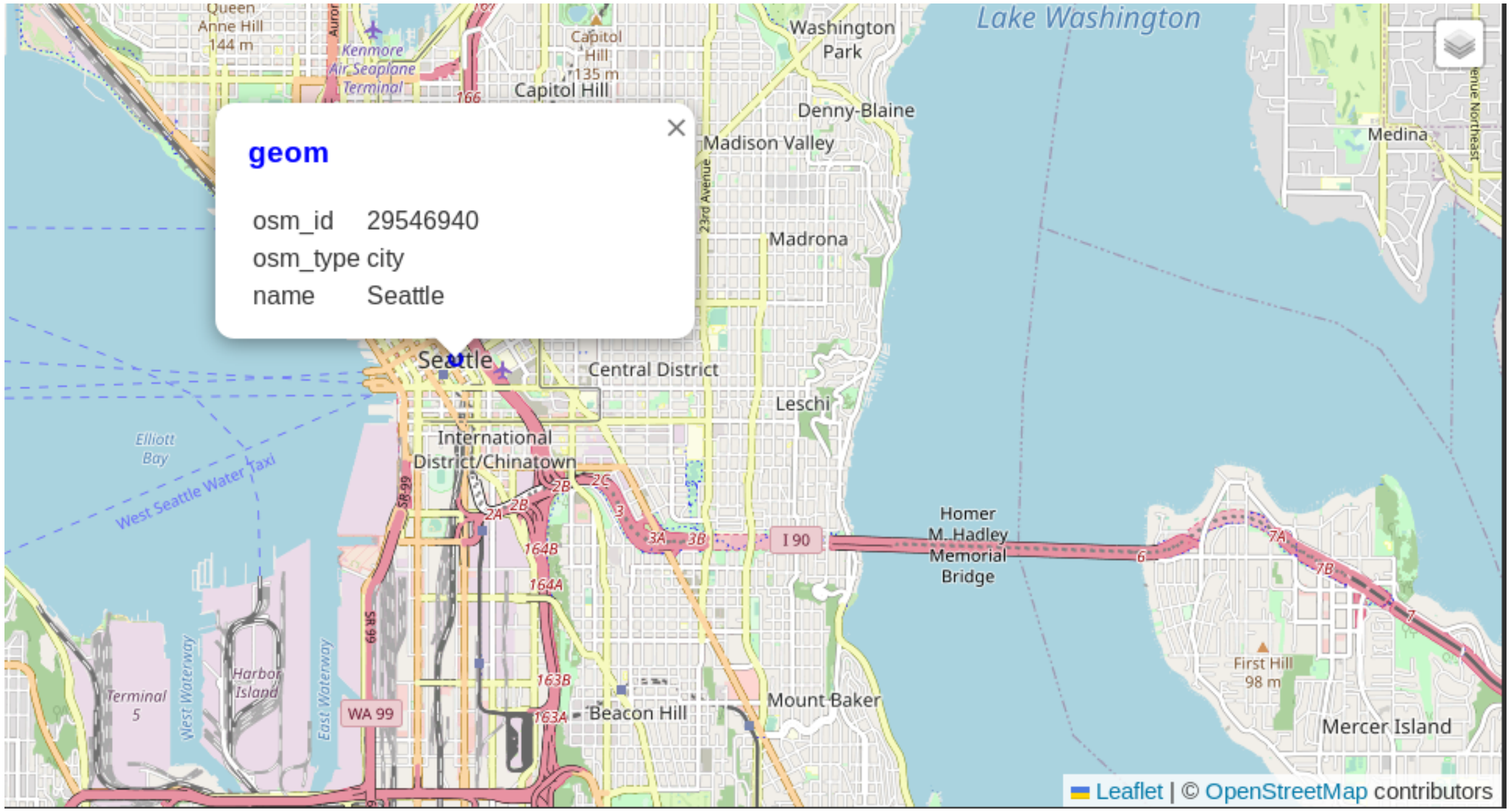
Use the right tools

> `psql` *is great, but...*



```
┌──────────┬──────────┬─────────┬───────────────────────┐
│  osm_id  │ osm_type │  name   │         geom          │
╞══════════╪══════════╪═════════╪═══════════════════════╡
│ 29546940 │ city     │ Seattle │ POINT(-122.33  47.60) │
└──────────┴──────────┴─────────┴───────────────────────┘
```

DBeaver Geometry Viewer


geom

osm_id 29546940
osm_type city
name Seattle

https://blog.rustprooflabs.com/2019/06/dbeaver-geometry-viewer

Use the right tools for PostGIS

DBeaver

- More like SSMS than `psql`
- General querying
- Spatial viewer

QGIS

- Polished visual outputs
- Powerful CAD for GIS
- Drag & Drop or Custom SQL

Use the right tools

Operating System

- Run Postgres on *Nix
- Docker okay

Why avoid Windows?

- Memory strategy
- Tooling
- Community Expertise

Temperature Check

Is anyone running MS SQL Server on Linux?

👍 🤔 👎

PostGIS is an Extension

https://postgis.net/

Extensions used today

These will need to be installed if following along on your DB!

- PostGIS
- pgDD https://github.com/rustprooflabs/pgdd
- Convert https://github.com/rustprooflabs/convert
- pg_stat_statements
- h3-pg https://github.com/zachasme/h3-pg
- pgRouting

Shameless Plug

PostgreSQL: Extensions Shape the Future

Wednesday 11/15 – 10:15 - 11:30 AM

https://passdatacommunitysummit.com/sessions/2014/

Spatial Data is just data

Spatial Data is just data

It has meta data

- Type of GIS data
- Data size
- Projection (SRID) of the data

How to explore meta data

- PostGIS Functions
- Internal catalog (`pg_catalog`)
- PgDD extension

Extensions in `pg_catalog`

```
In [5]: sql_raw = """
        SELECT extname, extversion
            FROM pg_catalog.pg_extension
            WHERE extname IN ('postgis', 'h3', 'convert', 'pgdd', 'pgrouting')
            ORDER BY extname;
        """
        pd.read_sql(sql_raw, get_db_conn())
```

Out[5]:

	extname	extversion
0	convert	0.0.3
1	h3	4.1.2
2	pgdd	0.5.1
3	pgrouting	3.5.0
4	postgis	3.4.0

Spatial Data is just data

Spatial Data is just data

Joins

```sql
SELECT foo.this, bar.that
    FROM foo
    JOIN bar ON <boolean expression>
;
```

Relational Join

```
FROM foo
JOIN bar ON foo.id = bar.id
```

Relational Join

```
FROM foo
JOIN bar ON foo.id = bar.id
```

Join with Function

```
FROM foo
JOIN bar ON check_foo_bar_ids(foo.id, bar.id)
```

Temperature Check

How do you feel about functions in joins?

👍 🤔 👎

Spatial Join

```
FROM foo
JOIN bar ON ST_Contains(foo.geom, bar.geom)
```

Spatial Join

```sql
FROM foo
JOIN bar ON ST_Contains(foo.geom, bar.geom)
```

Spaital Joins w/out functions

```sql
FROM foo
-- bounding box join
JOIN bar ON foo.geom && bar.geom
```

PostGIS functions and indexes

> `ST_Contains()` *"automatically includes a bounding box comparison that* ***makes use of any spatial indexes*** *that are available on the geometries"*

https://www.postgis.net/docs/ST_Contains.html

Small Group Brain Break

3 - 5 minutes

- Stand up & Group Up
- Name
- Peanut Butter: Creamy or Crunchy?
- Key takeaway so far

OpenStreetMap is Maptastic

- Open source, volunteer driven
- Bootstrap any spatial project
- Breadth of Data
- Worldwide coverage

PgOSM Flex Supports

- Custom Layers (tables)
- Custom Indexes
- Replication (diff updates)
- Intended to be modified (like Postgres!)

PgOSM Flex Resources

Has a lot of documentation

- In-Docker https://pgosm-flex.com/quick-start.html
- External Pg https://pgosm-flex.com/postgres-external.html

PgOSM Flex Resources

Explains regions, subregions, layersets, etc.

https://pgosm-flex.com/common-customization.html

https://pgosm-flex.com/layersets.html

More PgOSM Flex Resources

https://blog.rustprooflabs.com/2023/08/load-right-amount-of-openstreetmap

https://blog.rustprooflabs.com/2023/04/pgosm-flex-production-openstreetmap

https://blog.rustprooflabs.com/category/pgosm-flex

PostGIS and OpenStreetMap

- PostgresConf session: Intro to PostGIS and OpenStreetMap: https://youtu.be/l98YREUSJs4

Spatial Data Types

Geometry Type	Constructed with	Size on Disk*
Point	`(x, y)`	16 bytes
Line	2 or more points	16 bytes per point
Polygon	4 or more points, closed	16 bytes per point

Types (cont'd)

- MULTIPOINT
- MULTILINE
- MULTIPOLYGON
- GEOMETRYCOLLECTION

> *`ST_GeometryType()` is your meta-friend*

Geometry types in `DDL`

```sql
CREATE TABLE geom_examples
(
   id BIGINT NOT NULL PRIMARY KEY,
   geom_generic GEOMETRY,
   geom_point GEOMETRY(POINT),
   geom_point_3857 GEOMETRY(POINT, 3857),
   geom_line GEOMETRY(LINESTRING, 3857),
   geom_multiline GEOMETRY(MULTILINESTRING, 4326)
);
```

Adjusted Listing 2.11 from *Mastering PostGIS and OpenStreetMap*

Spatial Design Best Practices

- Design with restrictions in mind
- Limit columns to single SRID
- Store Points/Lines/Polygons in individual tables

Watch out for Gotchas

Watch out for Gotchas

> *x = longitude, y = latitude!*

Gotcha

> *A triangle is constructed with 4 points*

Gotcha

> *Complex polygons add up quickly*

`ST_MakePoint()`

```sql
WITH x_y AS (
    SELECT -122.33 AS x,
            47.60 AS y
)
SELECT ST_SetRID(
           ST_MakePoint(x, y)
               , 4326
           ) AS geom,
        x AS longitude, y AS latitude
    FROM x_y;
```

`ST_MakeLine()`

- Input: PostGIS Points
- Order matters!

`ST_MakeLine()`

- Input: PostGIS Points
- Order matters!

`ST_MakePolygon()`

- Input: PostGIS line
- Must be valid & closed

Creating Geometries via SQL is (generally) Rare

Types (cont’d)

Of course there is more!

- `GEOMETRY` - Euclidean coordinate system (cartesian plane)
- `GEOGRAPHY` - Geodetic coordinate system (ellipsoid)

`GEOMETRY`

- ✅ Fast calculations
- ✅ All PostGIS Functions
- ❌ Accuracy is tricky (SRIDs!)

`GEOGRAPHY`

- ✅ Accurate calculations
- ❌ Limited PostGIS functions
- 🤔 Calculations slow at scale

PostGIS Math

PostGIS can `CAST` (`::`) between `GEOMETRY` and `GEOGRAPHY`

```
SELECT ST_Transform(geom, 4326)::GEOGRAPHY,
        ST_Transform(geog::GEOMETRY, 3857)
    ...
```

It’s just Math

Ellipsoid and Cartesian Coordinates Conversions

European Space Agency

https://gssc.esa.int/navipedia/index.php
/Ellipsoidal_and_Cartesian_Coordinates_Conversion

Types (Still more!)

- Rasters
- 3D (x, y, z)
- Trajectories (x, y, m) (2D plus time)
- ☝️ Both native and via MobilityDB extension
- 4D (x, y, z, m)

Spatial Indexes with `GIST`

```sql
CREATE INDEX ON my_table USING GIST (geom);
```

Spatial Indexes with `GIST`

- Spatial indexes are bounding boxes
- `ST_Envelope()`
- Operators can use indexes: `&&`, `@`, `<->`
- Functions can use operators that use indexes

https://www.postgresql.org/docs/current/gist-intro.html

https://postgis.net/docs/reference.html#Operators

Spatial Indexes + Spatial Joins == Amazing

It’s Quiz Time!

Not a Temperature Check

Which Polygon is Bigger?

Which Polygon is Bigger?

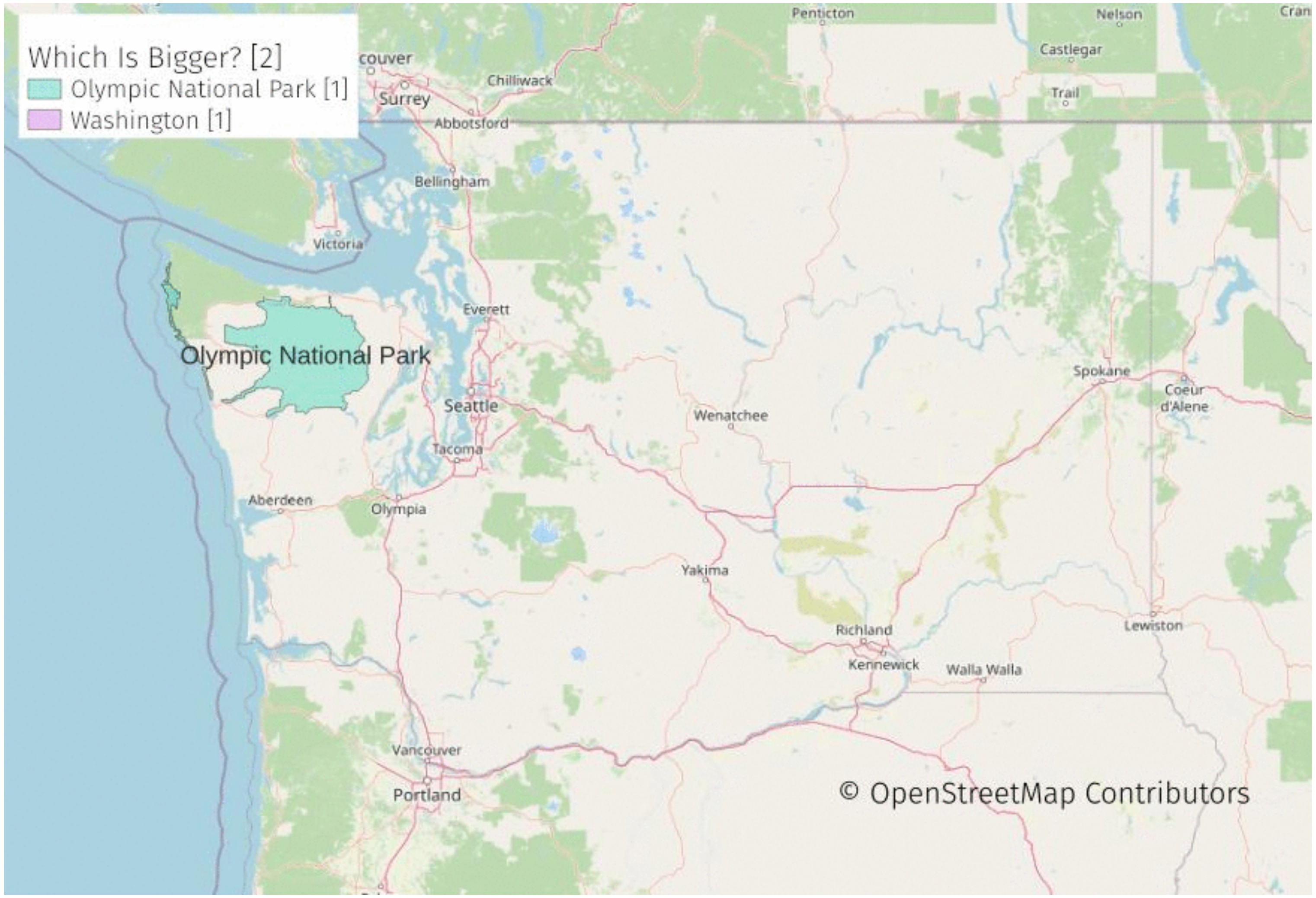
(It's a trick question)

Which Polygon is Bigger?

Washington State or Olympic National Park?

```
SELECT osm_id, name, geom
    FROM osm_wa.place_polygon
    WHERE osm_id IN (-165479, -163769)
;
```

Which Polygon is Bigger?



Which Polygon is Bigger?

Define ”Bigger”

- Area
- Number of points (aka size on disk)

Which Polygon is Bigger?

Define ”Bigger”

- Area
- Number of points (aka size on disk)

Size Related Functions

- `ST_Area()`
- `ST_NPoints()` and/or `pg_column_size()`

In [4]:

```
sql_raw = """
SELECT name,
        -- Calc on SRID 3857 -- not accurate, but not important for thi
        convert.area_m2_to_km2(ST_Area(geom))::BIGINT AS km2,
        ST_NPoints(geom) AS point_count,
        pg_size_pretty(pg_column_size(geom)::BIGINT) AS data_size
    FROM osm_wa.place_polygon
    WHERE osm_id IN (-165479, -163769)
;
"""
```

In [5]:

```
pd.read_sql(sql_raw, get_db_conn(in_docker=False))
```

Out[5]:

	name	km2	point_count	data_size
0	Washington	402907	1363	21 kB
1	Olympic National Park	8191	15197	238 kB

Which is Bigger?

- Washington is larger in area
- Olympic National Park is larger on disk!

Each point takes 16 bytes on disk

Beware: Geopolitical Boundaries!

In [7]:
```
sql_raw = """
SELECT COUNT(*) AS polygon_count,
        MIN(ST_NPoints(geom)) AS min_point_count,
        AVG(ST_NPoints(geom))::BIGINT AS avg_point_count,
        MAX(ST_NPoints(geom)) AS max_point_count
    FROM osm_wa.place_polygon
;
"""
pd.read_sql(sql_raw, get_db_conn(in_docker=False))
```

Out[7]:

	polygon_count	min_point_count	avg_point_count	max_point_count
0	3181	4	228	15197

Simplify Geometry

When Precise Detail not Required

```
ST_Simplify(geom, tolerance)
```

Simplify Geometry

When Precise Detail not Required

```
ST_Simplify(geom, tolerance)
```

Units for `tolerance` determined by the data's SRID

More Examples in DBeaver

(02 sql)

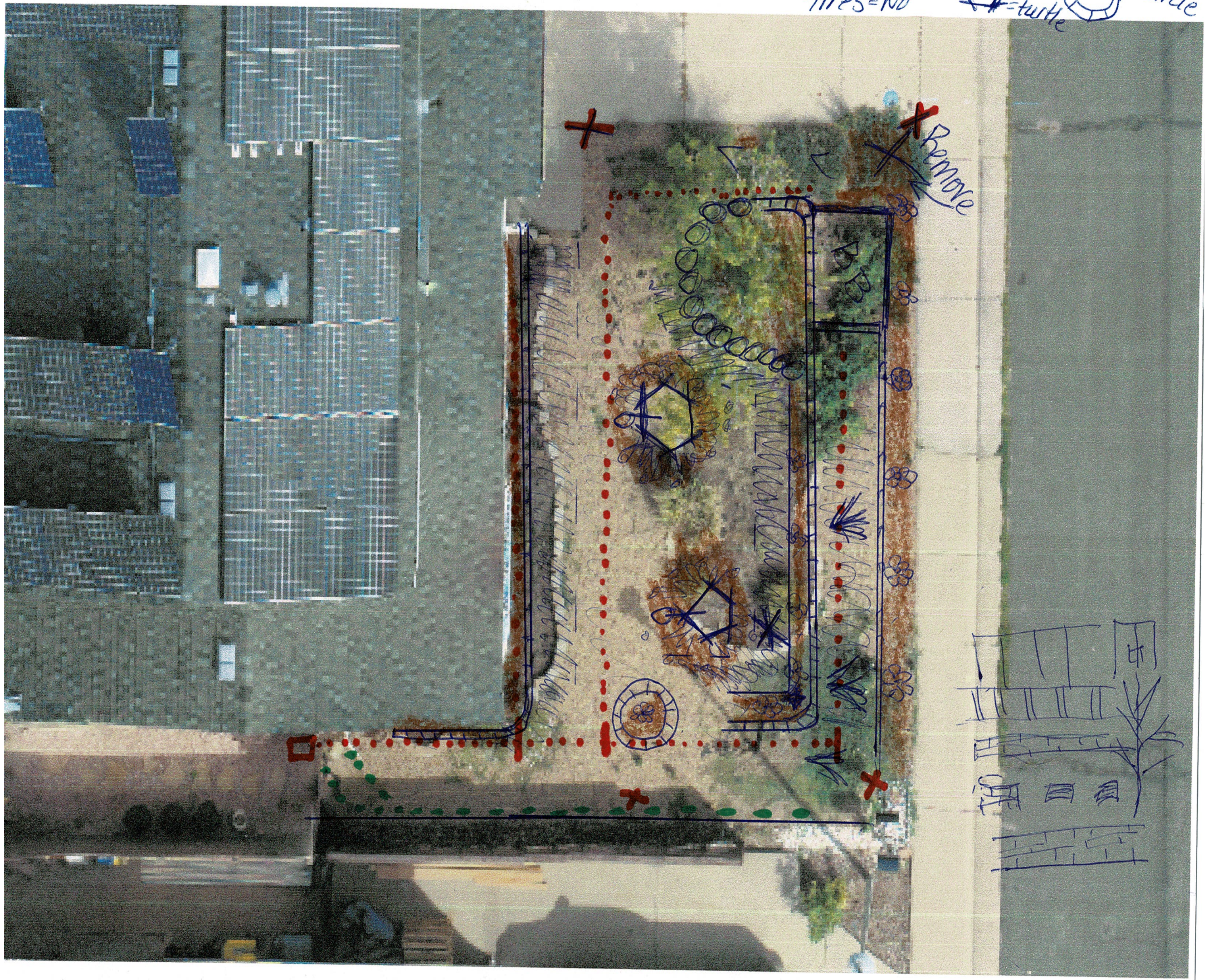
Other Data Sources

- gpx traces (mobile apps, GPS units, etc)
- Drone imagry / processing
- Private sensor networks
- etc

Crowd Sourcing

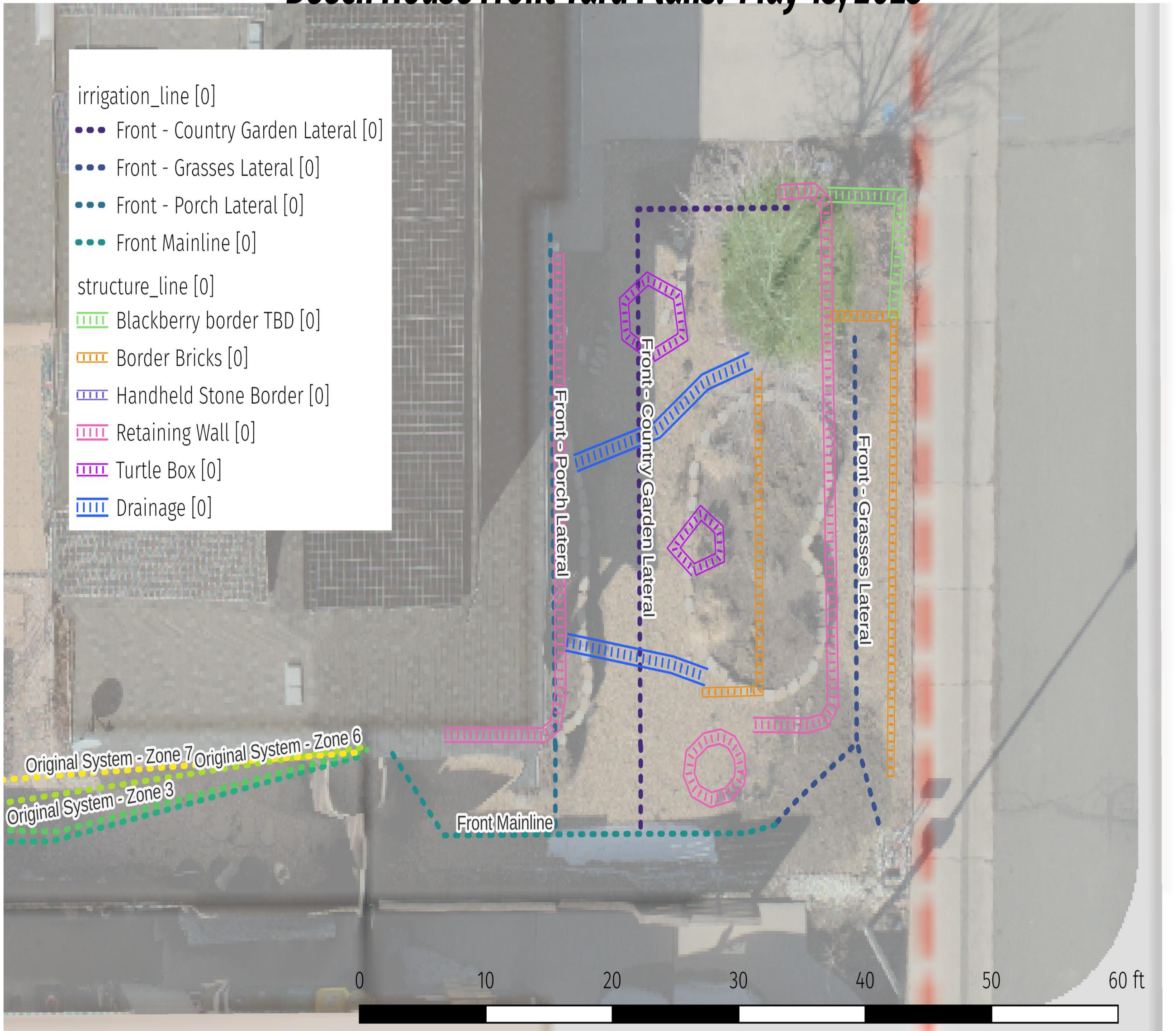
- What other Data Sources can you think of?

X = remove   pinwheels → ✳   tires = No   = turtle   = circle



● sump pump dams   ····· irrigation   — walls & boxes   mulch

Beech House Front Yard Plans: May 13, 2023



Geospatial Data Formats

- GeoJSON
- KML / GPX
- Shapefile
- Geodatabase
- WKT
- `.osm.pbf`
- csv (longitude, latitude)
- GeoTIFF

Geospatial Data Formats

- GeoJSON
- KML / GPX
- Shapefile
- Geodatabase
- WKT
- `.osm.pbf`
- csv (longitude, latitude)
- GeoTIFF

Geospatial data tools

- `ogr2ogr`
- `osm2pgsql`
- `shp2pgsql`
- `psql`
- `pg_dump`
- DBeaver
- QGIS
- MapLibre