duckdb-geography: Global vector data in DuckDB

Dewey Dunnington, Wherobots Inc.

duckdb-geography

In a nutshell:

```python
import duckdb
con = duckdb.connect()

con.install_extension("geography", repository="community")

con.load_extension("geography")
con.table_function("s2_data_cities").limit(3)
```

name varchar	population int32	geog geography
Vatican City	832	POINT (12.4533865 41.90328219999999)
San Marino	29000	POINT (12.441770200000002 43.93609579999999)
Vaduz	5342	POINT (9.51666947 47.1337238)

Data types: Geography

A Geography is a Geometry whose edges follow a the shortest distance on the Earth rather than the shortest distance in Cartesian/planar space

If you want to represent something like Fiji, Russia, or Antarctica without splitting it into pieces, you need Geography!

Data types: Geography

Spherely array:

- Order is kept
- In memory data structures
- Requires explicit parallelization

DuckDB Geography:

- Order is sometimes kept
- Serialized data structures (can be slower)
- Automatic parallelization

Data types: S2 Cell

```
con.sql("""
SELECT s2_cellfromlonlat(7.64, 47.53).s2_cell_parent(level)
FROM (VALUES (5), (4), (3), (2), (1), (0)) levels(level)
""")
```

s2_cell_parent(s2_cellfromlonlat(7.64, 47.53), "level") s2_cell
2/03302
2/0330
2/033
2/03
2/0
2/

Data types: S2 Cell

S2 Cells are represented by `uint64_t` and have an important property: **containment can be calculated using a simple comparison**

```
con.sql("""
SELECT c, s2_cell_range_min(c), s2_cell_range_max(c)
FROM (VALUES ('2/0330'::S2_CELL)) cells(c)
""")
```

c s2_cell	s2_cell_range_min(c) s2_cell	s2_cell_range_max(c) s2_cell
2/0330	2/0330000000000000000000000000000	2/0330333333333333333333333333333

(…because computers are much better at simple comparisons than spatial ones!)

Data types: S2 Cell Center

```
(
    con
    .table_function("s2_data_cities")
    .select("geog")
    .limit(5)
)
```

geog geography
POINT (12.4533865 41.90328219999999)
POINT (12.441770200000002 43.93609579999999)
POINT (9.51666947 47.1337238)
POINT (31.199997100000004 -26.466667500000003)
POINT (6.13000281 49.61166039999999)

Data types: S2 Cell Center

```
(
    con
    .table_function("s2_data_cities")
    .select("geog::S2_CELL_CENTER AS cell")
    .limit(5)
)
```

cell s2_cell_center
0/212113230003023131001102313200
0/212112131211123020010233101310
2/033031212023000232111020023022
0/331313212213231033112021020221
2/033022221321121102131101113231

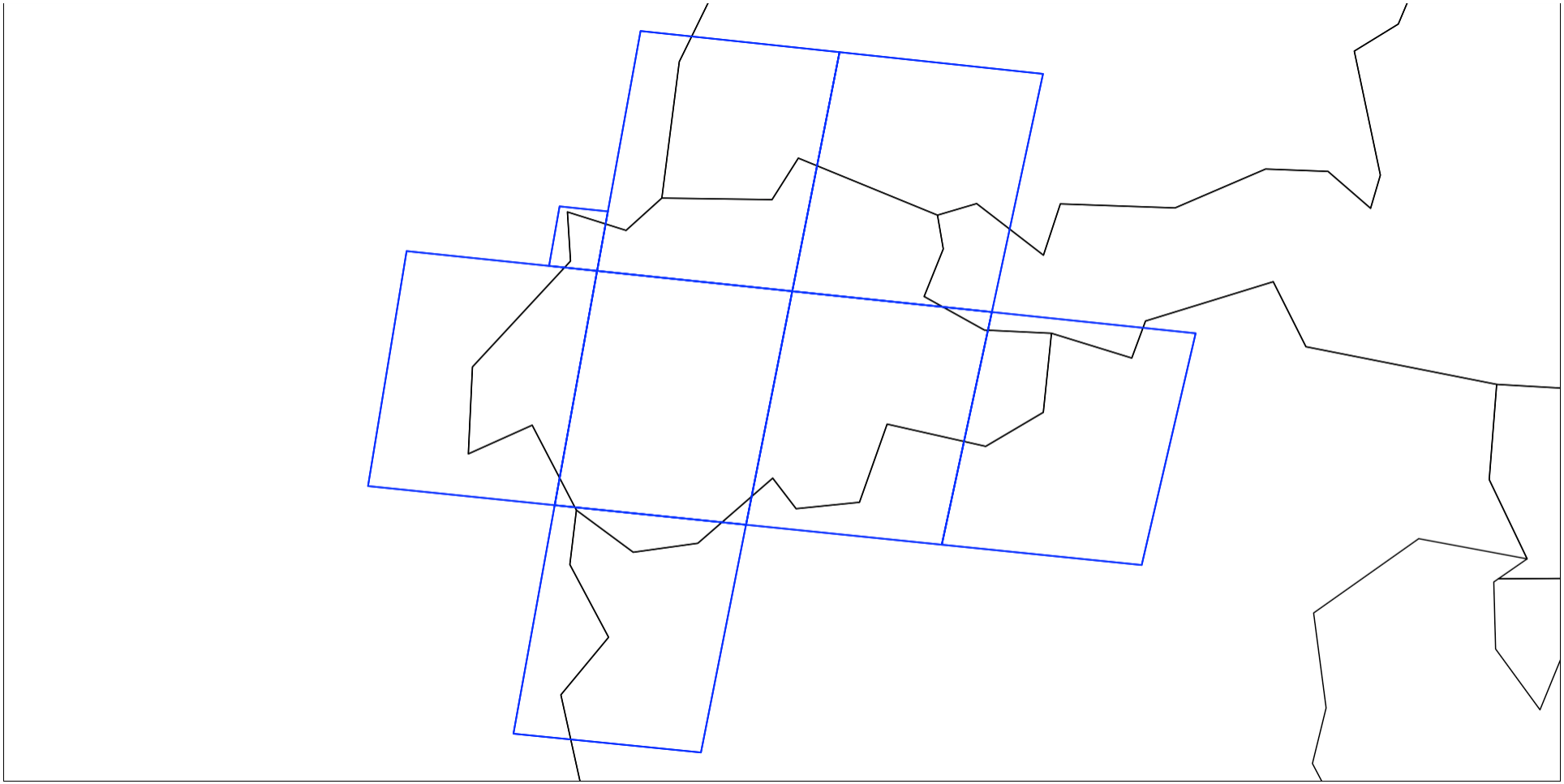
Data types: S2 Cell Union

```python
con.sql("SELECT s2_data_country('Switzerland').s2_covering()")
```

```
┌──────────────────────────────────────────────────────────────────────────────
│                    s2_covering(s2_data_country('Switzerland'))
│
│                                  s2_cell_union
│
├──────────────────────────────────────────────────────────────────────────────
│ [2/033001, 2/033002, 2/033010, 2/033012, 2/033013, 2/033020, 2/03302100,
2/033031] │
└──────────────────────────────────────────────────────────────────────────────
```

Data types: S2 Cell Union

```
con.sql("SELECT s2_data_country('Switzerland').s2_covering()")
```



Practical example: using S2 cells to do a join

- https://dewey.dunnington.ca/post/2024/wrangling-and-joining-130m-points-with-duckdb–the-open-source-spatial-stack/#duckdb-geography
- https://dewey.dunnington.ca/post/2024/partitioning-strategies-for-bigger-than-memory-spatial-data/#using-s2-cells-fixed-cell-level

duckdb-geography

- dewey.dunnington.ca/slides/geopython2025
- paleolimbot/duckdb-geography
- duckdb.org/community_extensions

```python
import duckdb
duckdb.install_extension("geography", repository="community")
```

```duckdb
INSTALL geography FROM community;
```