

## Performance Tuning / Coordinate Conversion

Performance Without Indices

### Info:

- Crime dataset has ~390,000 unique records
- Bus stop dataset has ~8,000 stops
- Calculating nearest bus stop for each crime with raw coordinate data is infeasible (would take days)

### Our Solution:

- Use a spatially indexed nearest neighbor join for massive speedup
- PostGIS extension to convert coordinate data to geom object

```
QUERY PLAN
-----
Sort (cost=22282263.73..22282263.74 rows=1 width=28) (actual time=118616.018..118616.034 rows=539 loops=1)
  Sort Key: (count(*)) DESC
  Sort Method: quicksort  Memory: 62kB
  -> GroupAggregate (cost=22281779.66..22282263.72 rows=1 width=28) (actual time=118611.980..118615.952 rows=539 loops=1)
    Group Key: bus_data.bsrid, bus_data.lng, bus_data.lat
    Filter: (count(*) > 20)
    Rows Removed by Filter: 1978
    -> Sort (cost=22281779.66..22281876.47 rows=38724 width=20) (actual time=118611.971..118613.235 rows=39126 loops=1)
      Sort Key: bus_data.bsrid, bus_data.lng, bus_data.lat
      Sort Method: quicksort  Memory: 356kB
      -> Nested Loop (cost=5739.68..22278828.71 rows=38724 width=20) (actual time=5.325..118603.013 rows=39126 loops=1)
        Filter: (count(*) > 20)
        -> Seq Scan on crime_data c (cost=0.00..14589.10 rows=38724 width=32) (actual time=0.009..58.875 rows=39126 loops=1)
          Filter: is_violent
          Rows Removed by Filter: 352684
        -> Limit (cost=5739.68..5739.68 rows=1 width=60) (actual times=3.029..3.029 rows=1 loops=39126)
          -> Sort (cost=5739.68..5760.65 rows=8387 width=60) (actual time=3.029..3.029 rows=1 loops=39126)
            Sort Key: ((c.geom <-> bus_data.geom))
            Sort Method: top-N heapsort  Memory: 25kB
            -> Seq Scan on bus_data (cost=0.00..5697.74 rows=8387 width=60) (actual time=0.028..2.243 rows=8387 loops=39126)
              Planning Time: 0.111 ms
              Execution Time: 118616.191 ms
              (21 rows)
```

Performance With Indices

```
-- Enable PostGIS extension and create new geometry columns
1 CREATE EXTENSION IF NOT EXISTS postgis;
2 ALTER TABLE crime_data ADD COLUMN geom geometry(Point, 4326);
3 UPDATE crime_data
4 SET geom = ST_SetSRID(ST_MakePoint(longitude, latitude), 4326);
5
6 ALTER TABLE bus_data ADD COLUMN geom geometry(Point, 4326);
7 UPDATE bus_data
8 SET geom = ST_SetSRID(ST_MakePoint(longitude, latitude), 4326);
9
10 -- Create indexes to speed up join immensely
11 CREATE INDEX crimes_gix ON crime_data USING GIST (geom);
12 CREATE INDEX bus_stops_gix ON bus_data USING GIST (geom);
```

```
QUERY PLAN
-----
Sort (cost=54141.01..54141.02 rows=1 width=28) (actual time=1052.823..1052.839 rows=540 loops=1)
  Sort Key: (count(*)) DESC
  Sort Method: quicksort  Memory: 62kB
  -> HashAggregate (cost=54140.99..54141.00 rows=1 width=28) (actual time=1052.597..1052.766 rows=540 loops=1)
    Group Key: bus_data.bsrid, bus_data.lng, bus_data.lat
    Filter: (count(*) > 20)
    Batches: 1  Memory Usage: 649kB
    Rows Removed by Filter: 1978
    -> Nested Loop (cost=0.15..53753.75 rows=38724 width=20) (actual time=0.115..1045.210 rows=39126 loops=1)
      -> Seq Scan on crime_data c (cost=0.00..14589.10 rows=38724 width=32) (actual time=0.017..54.705 rows=39126 loops=1)
        Filter: is_violent
        Rows Removed by Filter: 352684
      -> Limit (cost=0.15..0.99 rows=1 width=60) (actual time=0.025..0.025 rows=1 loops=39126)
        -> Index Scan using bus_stops_gix on bus_data (cost=0.15..7056.80 rows=8387 width=60) (actual time=0.025..0.025 rows=1 loops=39126)
          Index Scan using crimes_gix on crime_data (cost=0.00..5697.74 rows=8387 width=60) (actual time=0.028..2.243 rows=8387 loops=39126)
            Order By: (geom <-> c.geom)
            Planning Time: 0.167 ms
            Execution Time: 1052.877 ms
            (17 rows)
```

# >100x Speed Up!

# Geospatial Analysis of Denver Crime Data around RTD Bus Stops

## Visualizing Violent Denver Crime by Stops

- Classify crimes as violent vs. non-violent.
- For each violent crime, find the nearest RTD bus stop using a nearest-neighbor geospatial search
- Group crimes by their closest stop and count them
- Keep only stops with > 20 violent crimes nearby
- Export results to CSV for Python + Folium to build a bubble-map visualization

[Demo Link](#)

