Managing Mobility Data in PostgreSQL

Third International Workshop in Big Mobility Data Analytics BMDA 2020

Contact: Esteban Zimányi (ezimanyi@ulb.ac.be)
Mahmoud SAKR (mahmoud.sakr@ulb.ac.be)

MobilityDB

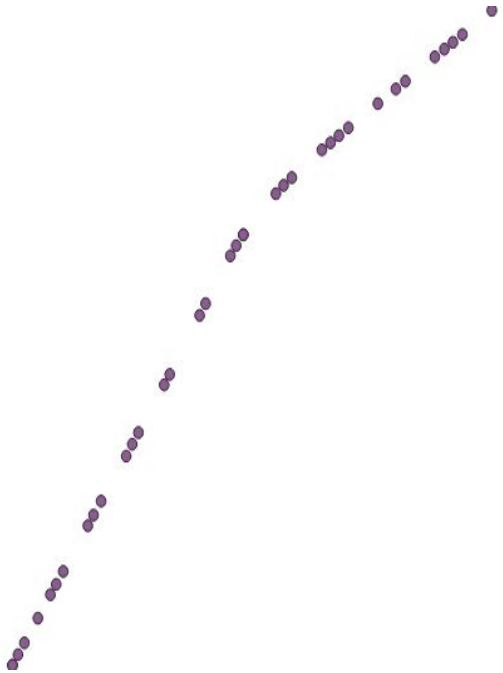
- A moving object database MOD
- Builds on PostgreSQL and PostGIS
- Developed by a team in Université libre de Bruxelles
- Open source extension
- Compliant with OGC standards on Moving Features, and in particular the OGC Moving Features Access

ULB

OGC®

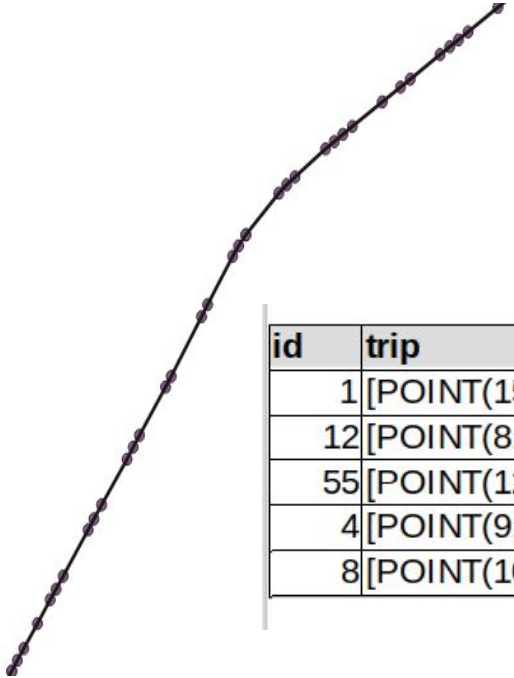
Associate Member

Mobility Data: PostGIS



id	geom	t
1	POINT(15.839728 55.836783)	2018-04-01 19:34:49+00
1	POINT(15.831427 55.83403)	2018-04-01 19:36:19+00
1	POINT(15.823145 55.831307)	2018-04-01 19:37:49+00
1	POINT(15.820398 55.830398)	2018-04-01 19:38:19+00
1	POINT(15.817642 55.829467)	2018-04-01 19:38:49+00
1	POINT(15.816722 55.829165)	2018-04-01 19:38:59+00
1	POINT(15.814793 55.828537)	2018-04-01 19:39:20+00
1	POINT(15.80575 55.825483)	2018-04-01 19:40:59+00
1	POINT(15.798323 55.823005)	2018-04-01 19:42:20+00
1	POINT(15.797487 55.822735)	2018-04-01 19:42:29+00
1	POINT(15.792805 55.821195)	2018-04-01 19:43:20+00
1	POINT(15.791978 55.820913)	2018-04-01 19:43:29+00
1	POINT(15.786472 55.81908)	2018-04-01 19:44:29+00
1	POINT(15.784457 55.818405)	2018-04-01 19:44:50+00
1	POINT(15.779068 55.816605)	2018-04-01 19:45:50+00
1	POINT(15.776327 55.815688)	2018-04-01 19:46:20+00
1	POINT(15.775412 55.815377)	2018-04-01 19:46:29+00
1	POINT(15.774503 55.815063)	2018-04-01 19:46:39+00
1	POINT(15.772762 55.814488)	2018-04-01 19:46:59+00
1	POINT(15.770842 55.813838)	2018-04-01 19:47:20+00
1	POINT(15.76726 55.812655)	2018-04-01 19:47:59+00
1	POINT(15.764525 55.811742)	2018-04-01 19:48:29+00

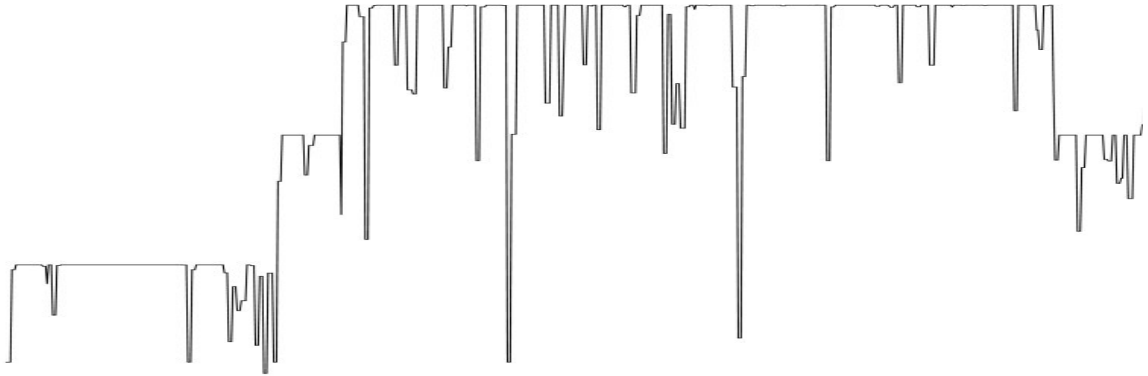
Mobility Data: Trajectories



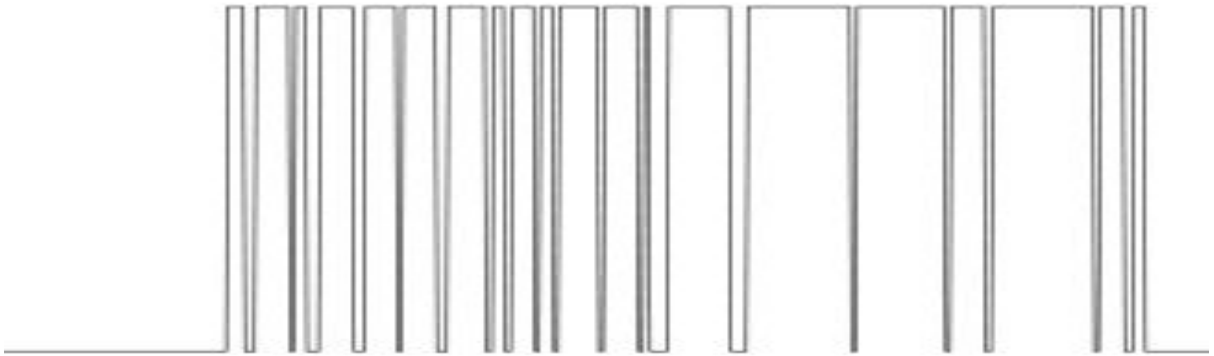
id	trip
1	[POINT(15.839728 55.836783)@2018-04-01 19:34:49+00, POINT(15.831427 55.83403)@2018
12	[POINT(8.067513 57.851652)@2018-04-01 07:35:06+00, POINT(8.073813 57.848518)@2018-
55	[POINT(12.446722 54.689387)@2018-04-01 00:00:00+00, POINT(12.447155 54.689822)@201
4	[POINT(9.752845 55.544552)@2018-04-01 08:34:16+00, POINT(9.75391 55.545305)@2018-0
8	[POINT(10.141707 55.152783)@2018-04-01 07:11:16+00, POINT(10.141707 55.152783)@201

Mobility Data: Temporal Types

tfloat: speed(Trip)



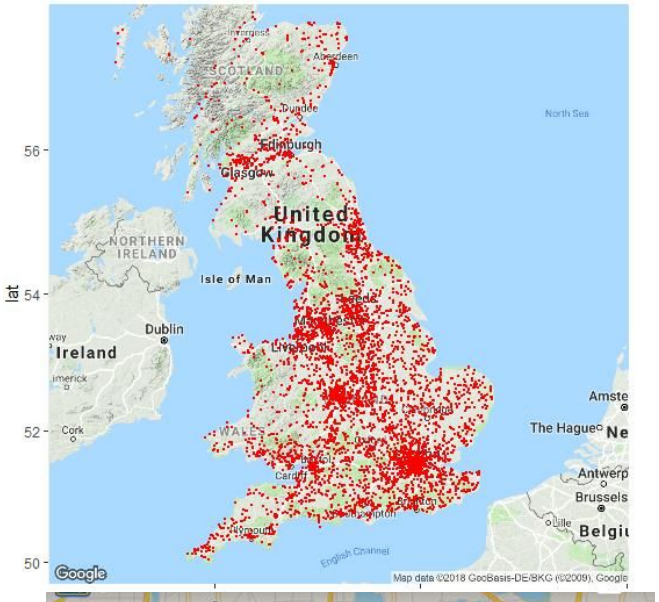
tbool: speed(Trip) > 90



Mobility Data: Points

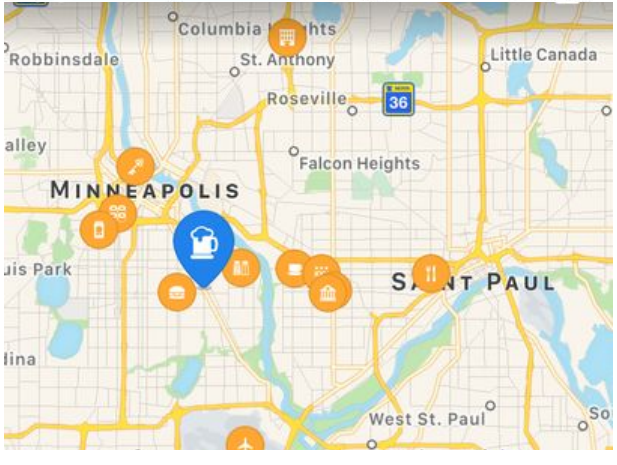
tgeompoint(inst): UK road accidents 2012-14

https://www.kaggle.com/daveianhickey/2000-16-traffic-flow-england-scotland-wales

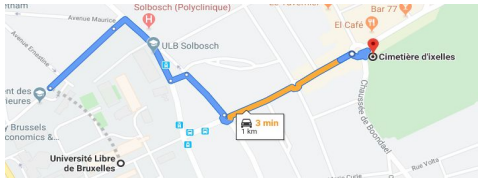


tgeompoint(instants): social network check-ins

https://support.foursquare.com/



MobilityDB: Architecture



MobilityDB

tgeompoint, tgeogpoint,
tint, tfloat, ttext, tbool

PostGIS

geometry, geography

PostgreSQL

numeric, monetary, character,
data/time, boolean, enum,
arrays, range,
XML, JSON, ...

Data Management: Application vs. DBMS Support

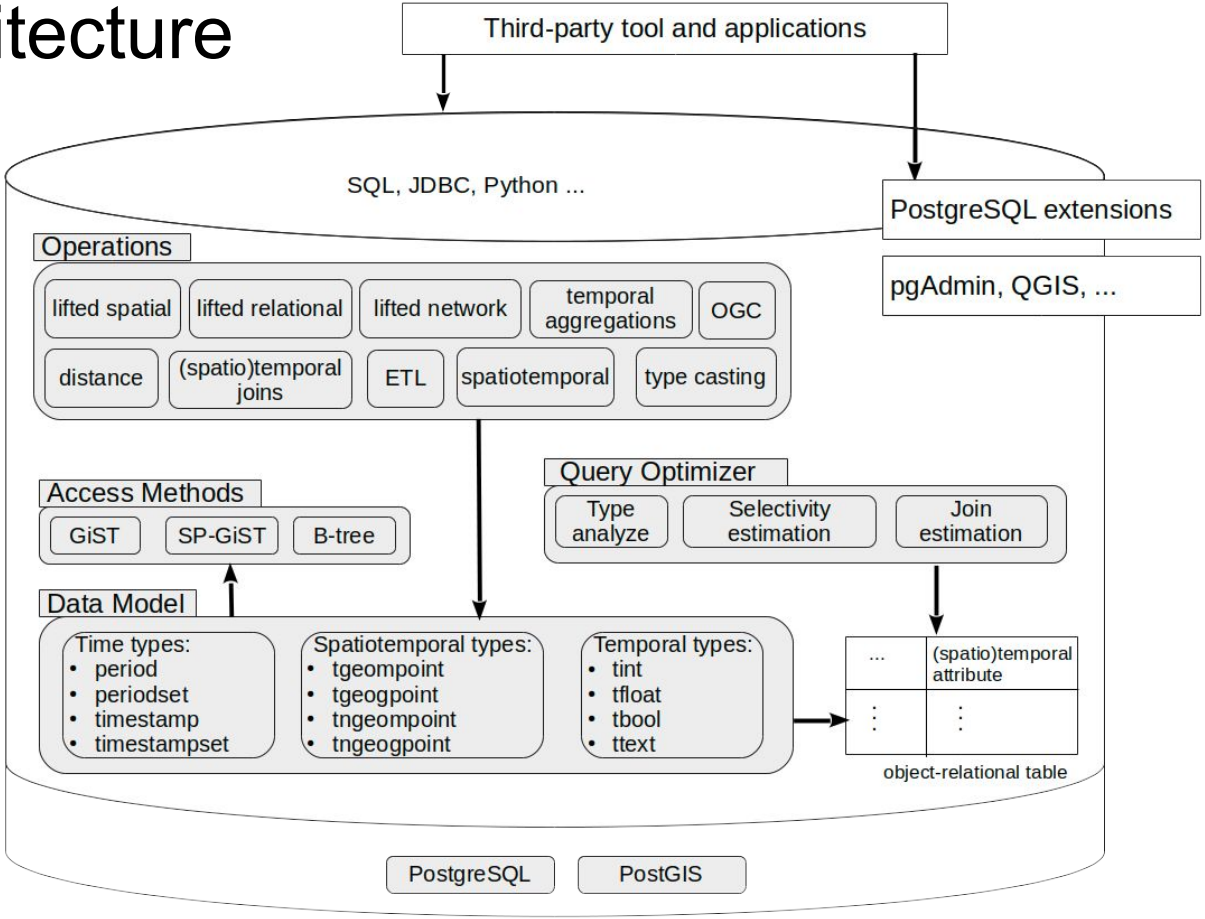
```
WITH TripSegs AS (
     SELECT TripId, Point AS P1, T AS T1,
          LAG (Point) OVER W AS P2,
          LAG (T) OVER W AS T2
     FROM TripPoints WINDOW W AS (
          PARTITION BY TripId ORDER BY T)
)
SELECT DISTINCT(TripId)
FROM TripSegs
WHERE ST_Distance(Point2, Point1) /
     (T2 - T1) > 90
```

Speed calculation using PostGIS

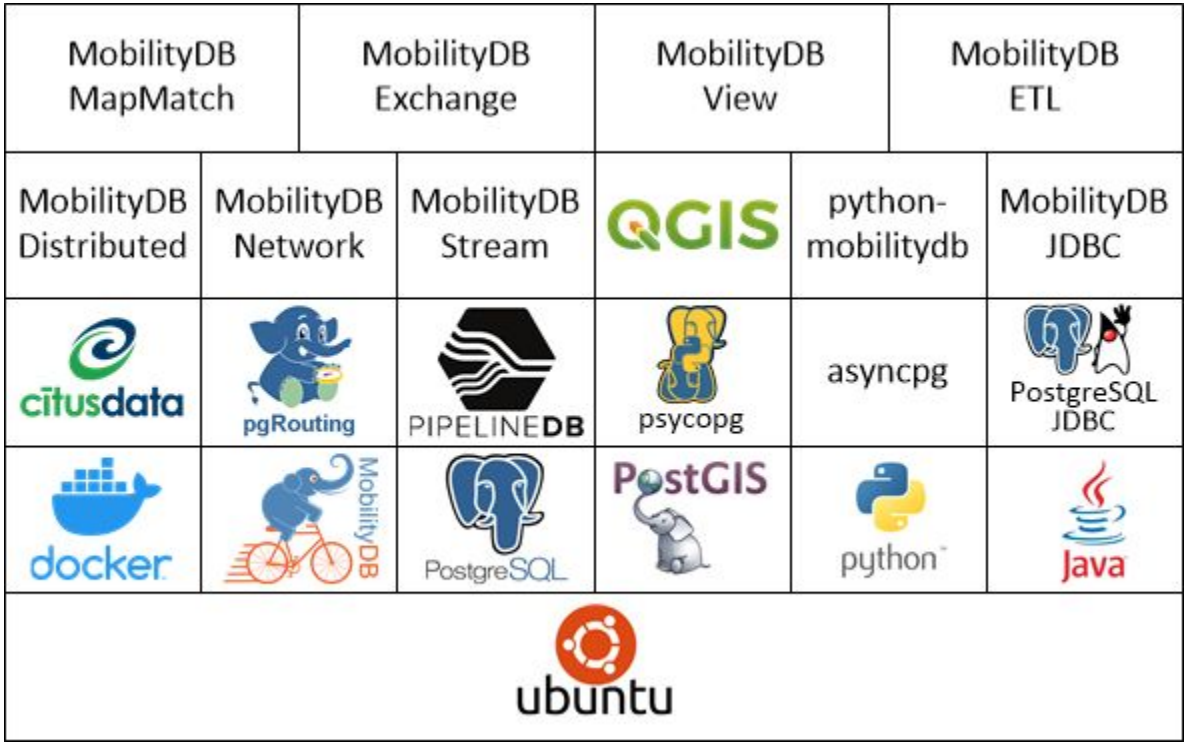
```
SELECT TripId
FROM Trips t
WHERE speed(trip) @> 90
```

Using MobilityDB

MobilityDB Architecture



MobilityDB Ecosystem



Loading Data: CSV Example

```
CREATE TABLE TripsInput (
  CarId integer REFERENCES Cars,
  TripId integer,
  Lon float,
  Lat float,
  T timestamptz,
  PRIMARY KEY (CarId, TripId, T) );
```

```
CREATE TABLE Trips (
  CarId integer NOT NULL,
  TripId integer NOT NULL,
  Trip tgeompoint,
  PRIMARY KEY (CarId, TripId),
  FOREIGN KEY (CarId)
    REFERENCES Cars (CarId) );
```

```
COPY TripsInput(CarId, TripId, Lon, Lat, T) FROM '/home/mobilitydb/data/trips.csv'
    DELIMITER  ',' CSV HEADER;
```

```
INSERT INTO Trips
  SELECT CarId, TripId,
    tgeompointseq(array_agg(tgeompointinst(
      ST_Transform(ST_SetSRID(ST_MakePoint(Lon,Lat), 4326), 5676), T) ORDER BY T))
FROM TripsInput
GROUP BY CarId, TripId;
```

Loading Data: GTFS Example

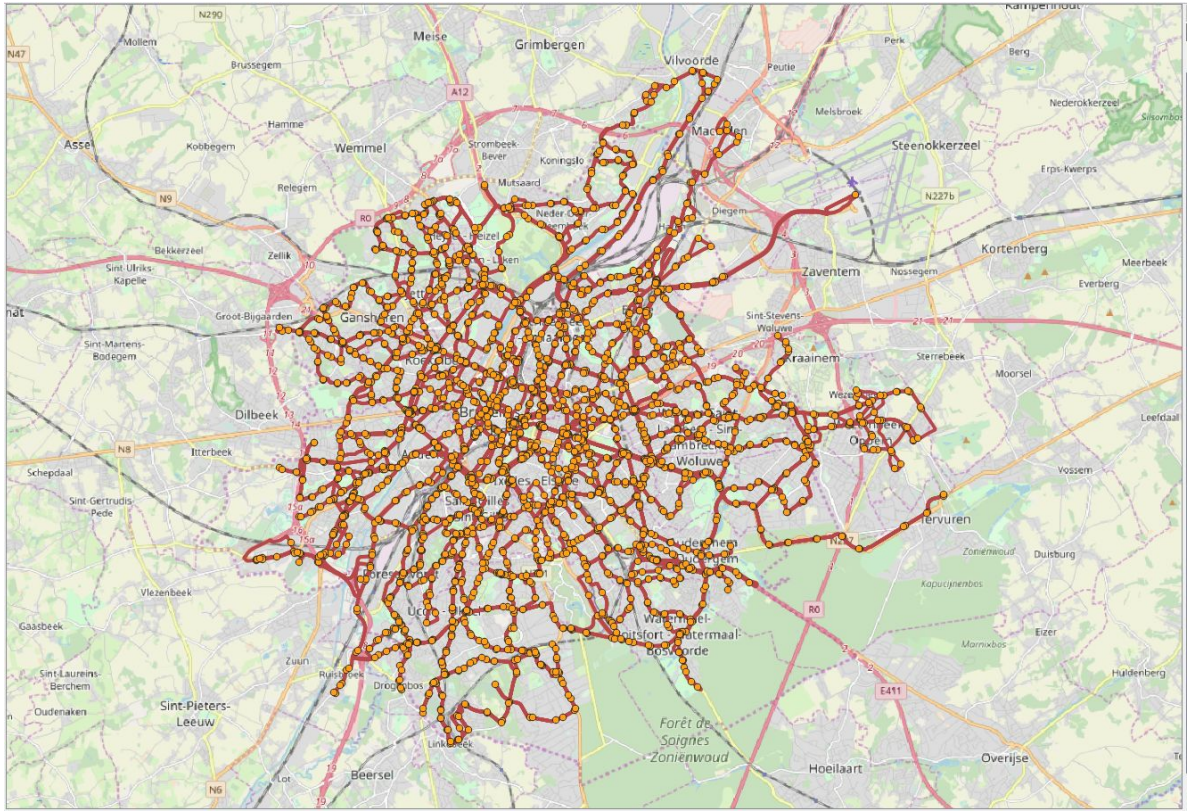
Source: STIB, Brussels

Duration: 28 days

7 Oct- 3 Nov 2019

#Trips: 445,187

DB size: 9 GB



https://docs.mobilitydb.com/nightly/workshop/ch02.html

Loading Data: Google Location Data

Source: Personal Google data

Duration: 6 years with time gaps

JSON size: 144 MB

CSV size: 8 MB converted with jq

#Trips: One per day

https://docs.mobilitydb.com/nightly/workshop/ch03.html

Loading Data: Google Location Data

USE RESPONSIBLY

Source: Personal Google data

Duration: 6 years with time gaps

JSON size: 144 MB

CSV size: 8 MB converted with jq

#Trips: One per day

https://docs.mobilitydb.com/nightly/workshop/ch03.html

Loading Data: Maritime Data (AIS)

Source: Danish Maritime Authority

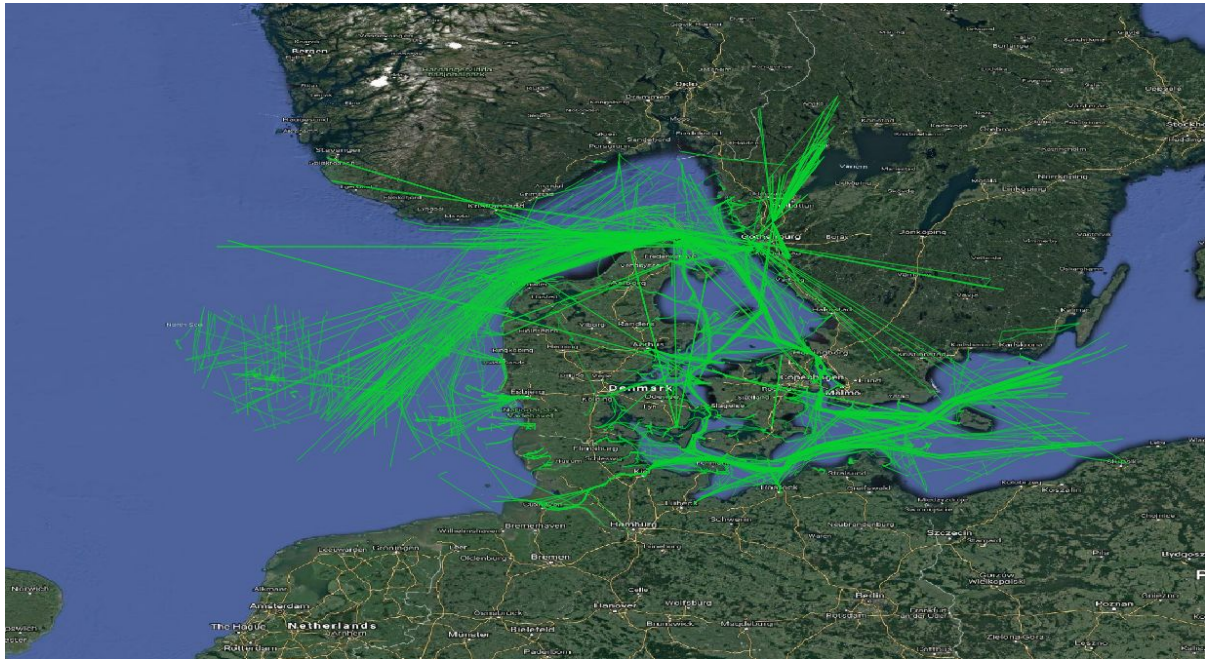
Duration: one day

April 1st 2018

#Rows: 10M

#Trips: 2,995

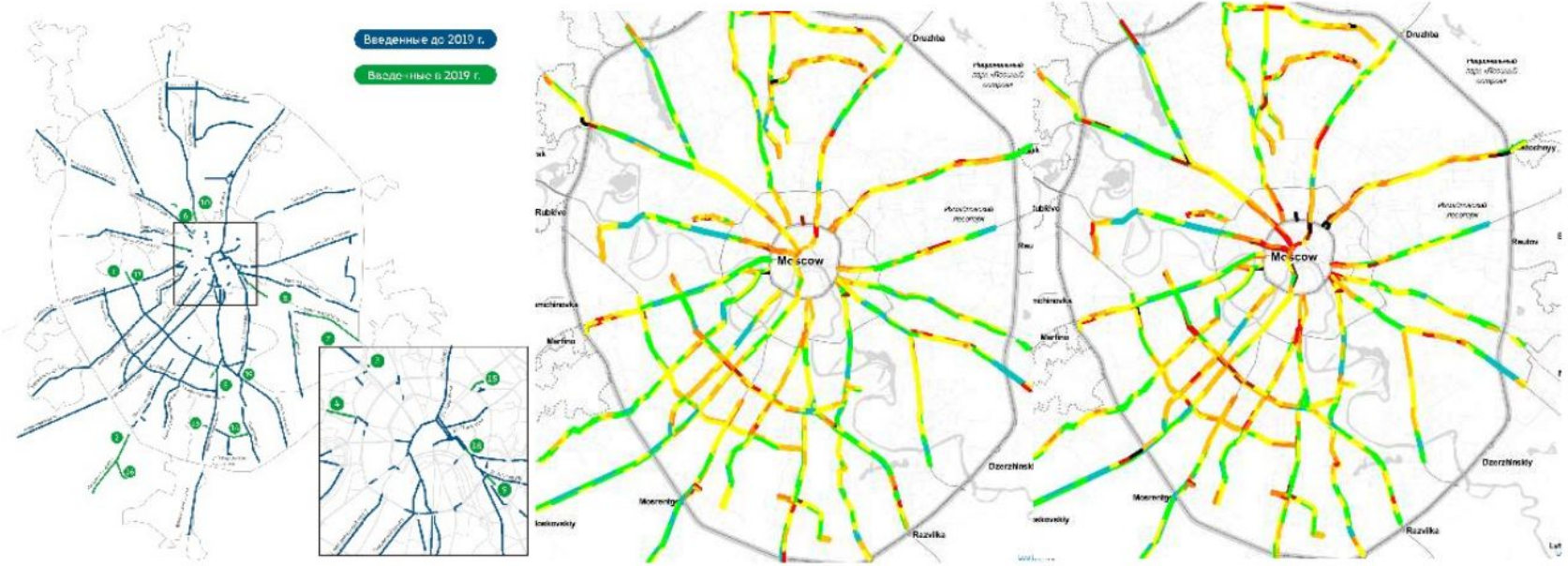
DB size: 1 GB



https://docs.mobilitydb.com/nightly/workshop/ch01.html

Data analysis – velocity maps

Moscow bus lanes

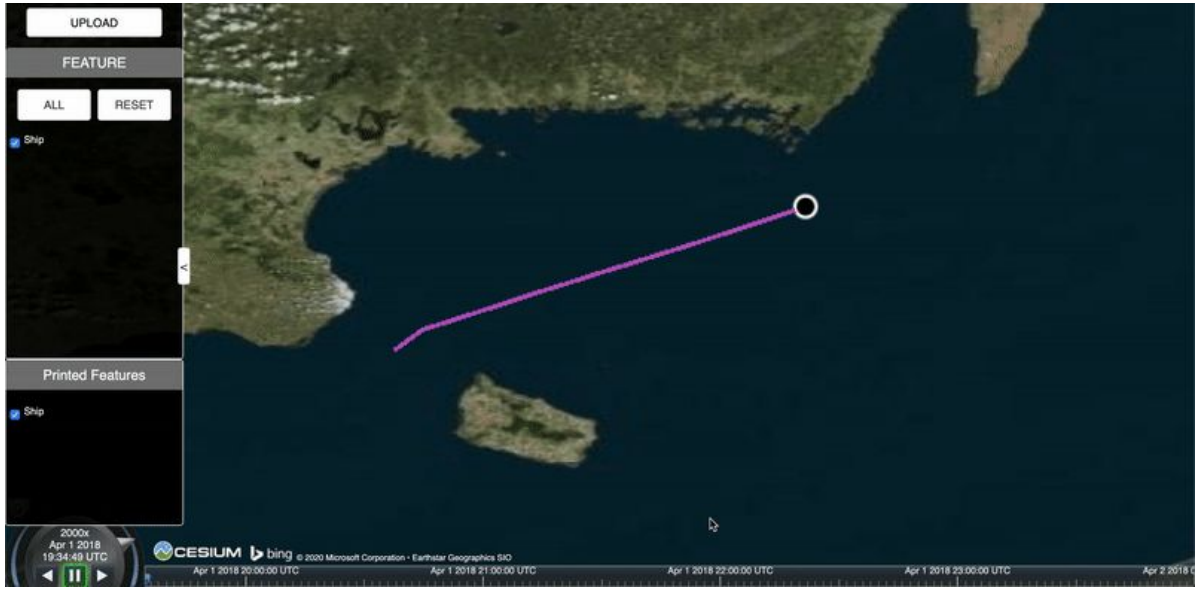


Nina Belyavskaya, https://pgconf.ru/en/2020/265266

Visualization: Cesium MF-JSON

Cesium extension for MF-JSON visualization

Kyoung-Sook KIM et al., Artificial Intelligence Research Center (AIRC) of AIST in Japan.



https://www.opengeospatial.org/pressroom/pressreleases/3083

Example: Spatial Projection

```
Ships(mmsi integer, trip tgeompoint, sog tfloat, cog tfloat, traj geometry)
```

List the ships that commute between the ports Rødby and Puttgarden

```
CREATE INDEX Ships_trip_idx ON Ships USING GiST(trip);
SELECT *
FROM Ships
WHERE intersects( trip, ST_MakeEnvelope(...) ) AND
    intersects( trip, ST_MakeEnvelope(...) )
```

The intersects function is index supported


Rødbyhavn
Bjørnæs
Grüner Brink
PUTTGARDEN

Example: Temporal Operations

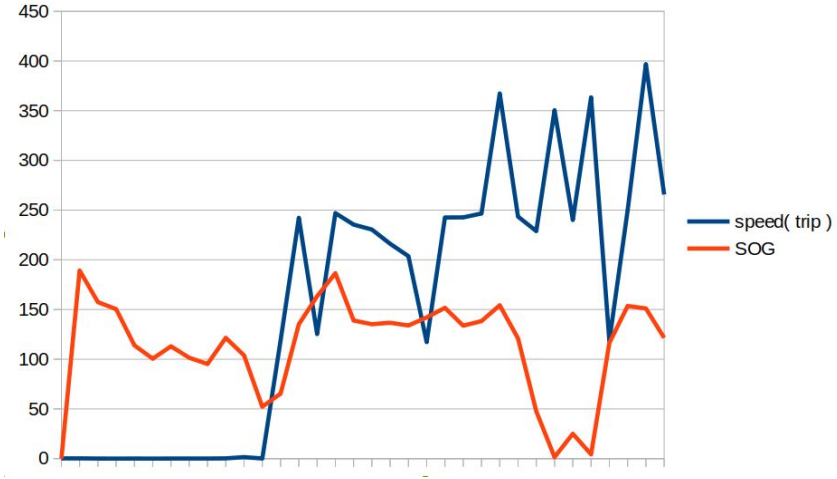
```
Ships(mmsi integer, trip tgeompoint, sog tfloat, cog tfloat, traj geometry)
```

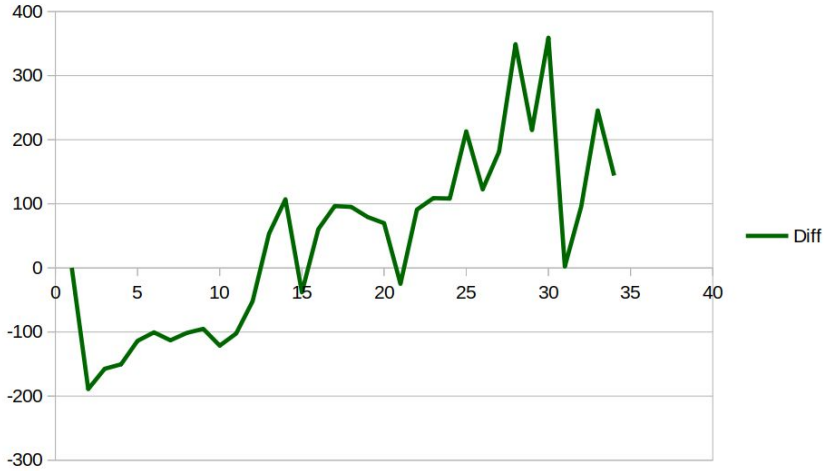
Find all the trips that report SOG very different from the speed calculated from their trajectories (noise, broken sensor, ...).

```
SELECT *
FROM Ships
WHERE twavg ( ( speed( trip ) * 3.6 ) - ( sog * 1.852 ) ) > 10
```

Example: Temporal Operations

```
SELECT *
FROM Ships
WHERE twavg ( ( speed( trip ) * 3.6 ) - ( sog * 1.852 ) ) > 10
```





Example: Aggregation

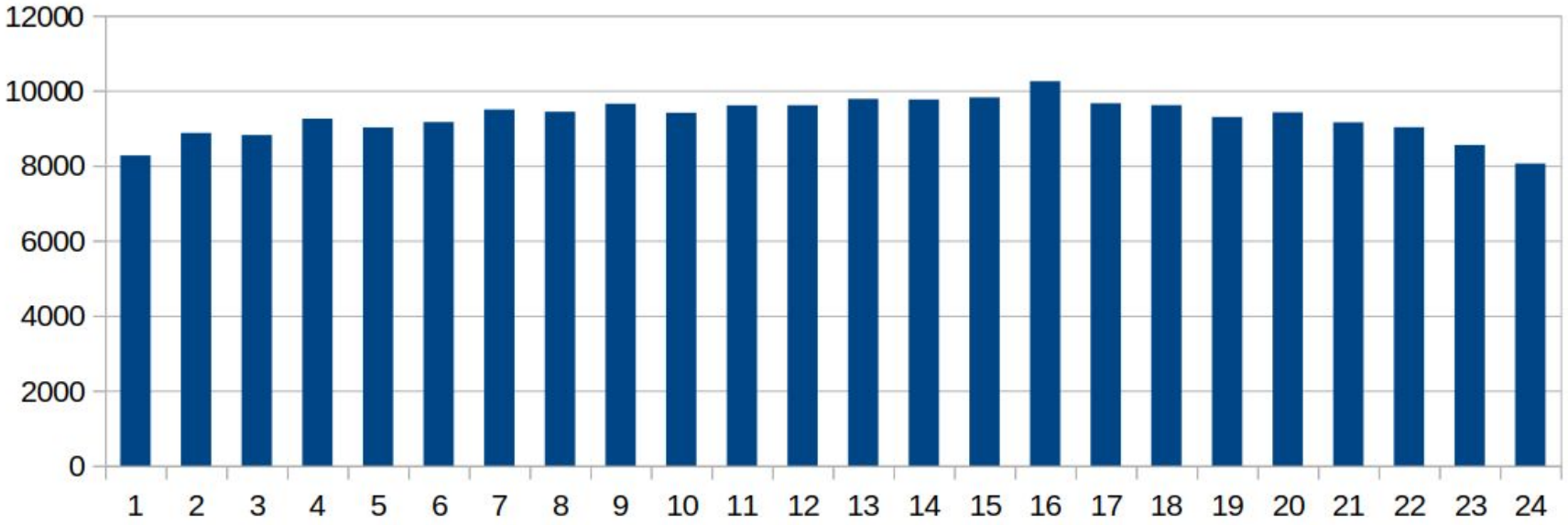
```
Ships(mmsi integer, trip tgeompoint, sog tfloat, cog tfloat, traj geometry)
```

Total distance travelled by ships per hour

```
WITH TimeSplit(Period) AS (
    SELECT period(H, H + interval '1 hour')
    FROM generate_series(timestamptz '2018-04-01',
        timestamptz '2018-04-02', interval '1 hour') AS H )
SELECT Period, SUM( length( atPeriod( Trip, Period) ) )/1000 travelledKms
FROM TimeSplit T, Ships S
WHERE T.Period && S.Trip
GROUP BY T.Period
ORDER BY T.Period;
```

Example: Aggregation

What is the total distance travelled by ships per hour

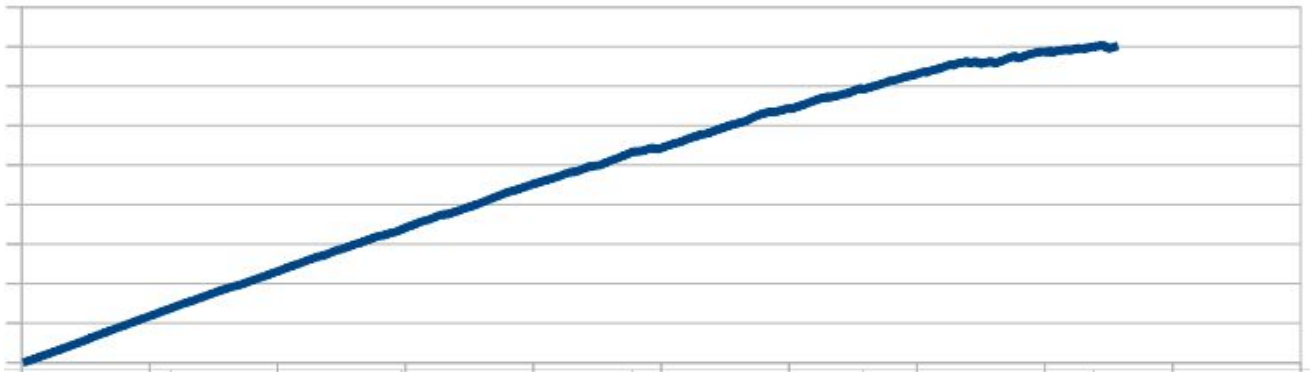


Example: Temporal Aggregation

```
Ships(mmsi integer, trip tgeompoint, sog tfloat, cog tfloat, traj geometry)
```

Cumulative distance travelled by the ships at each instant during one week

```
SELECT tsum( cumulativeLength( Trip) ) traveled
FROM Ships;
```

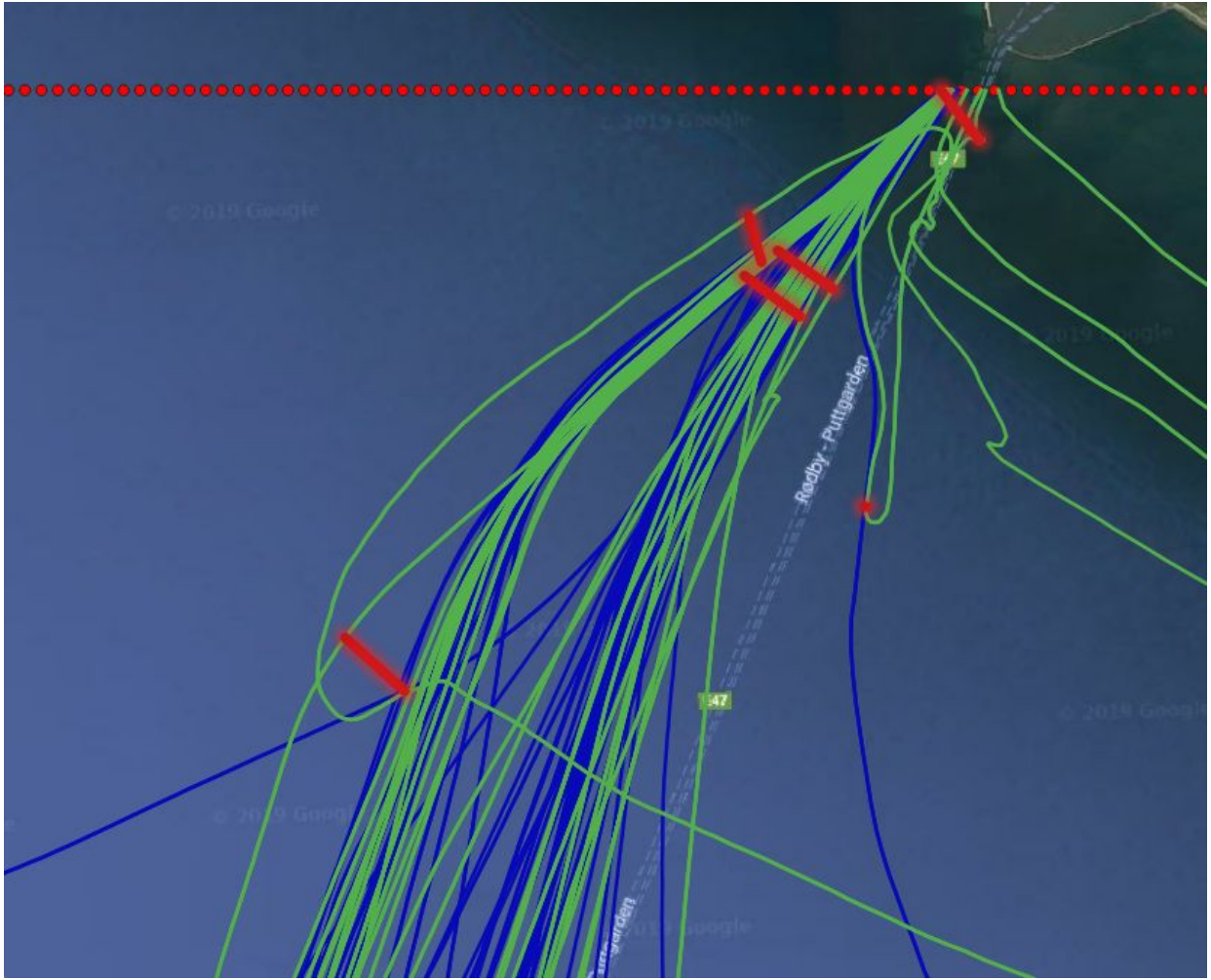


Example: Spatiotemporal Join

```
Ships(mmsi integer, trip tgeompoint, sog tfloat, cog tfloat, traj geometry)
```

Ships that come closer than 300 meters to one another

```
SELECT S1.MMSI, S2.MMSI, S1.Traj, S2.Traj,
    shortestLine(S1.trip, S2.trip) Approach
FROM Ships S1, Ships S2
WHERE S1.MMSI > S2.MMSI AND
    dwithin(S1.trip, S2.trip, 300)
```


© 2019 Google
Rødby - Puttgarden
47

Example: Trajectory Simplification

Compute simplified trip (with max. 100 m distance and 10 km/h speed difference)

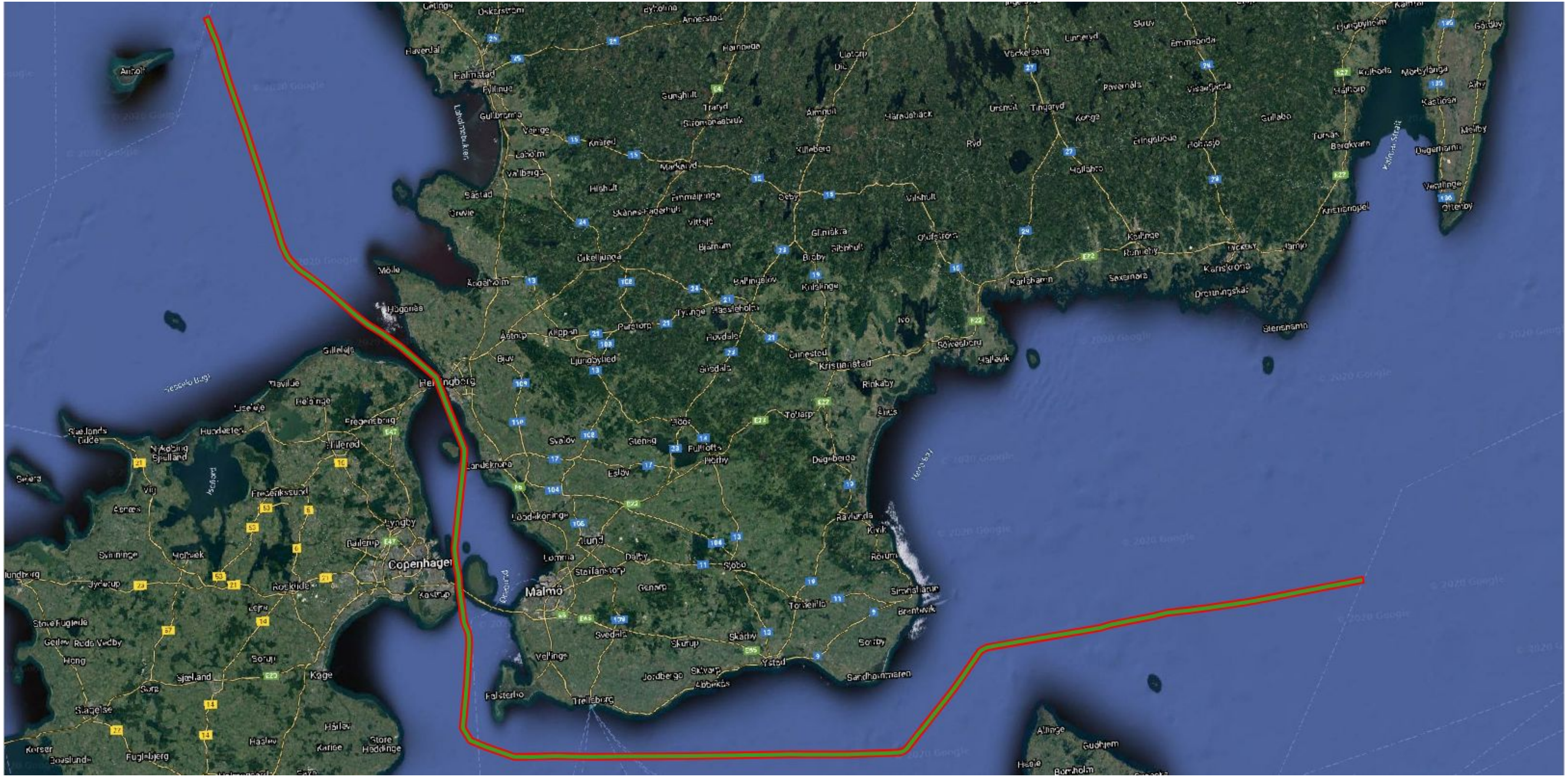
```
UPDATE Ships SET SimpTrip = simplify(Trip, 100, 10 / 3.6);
```

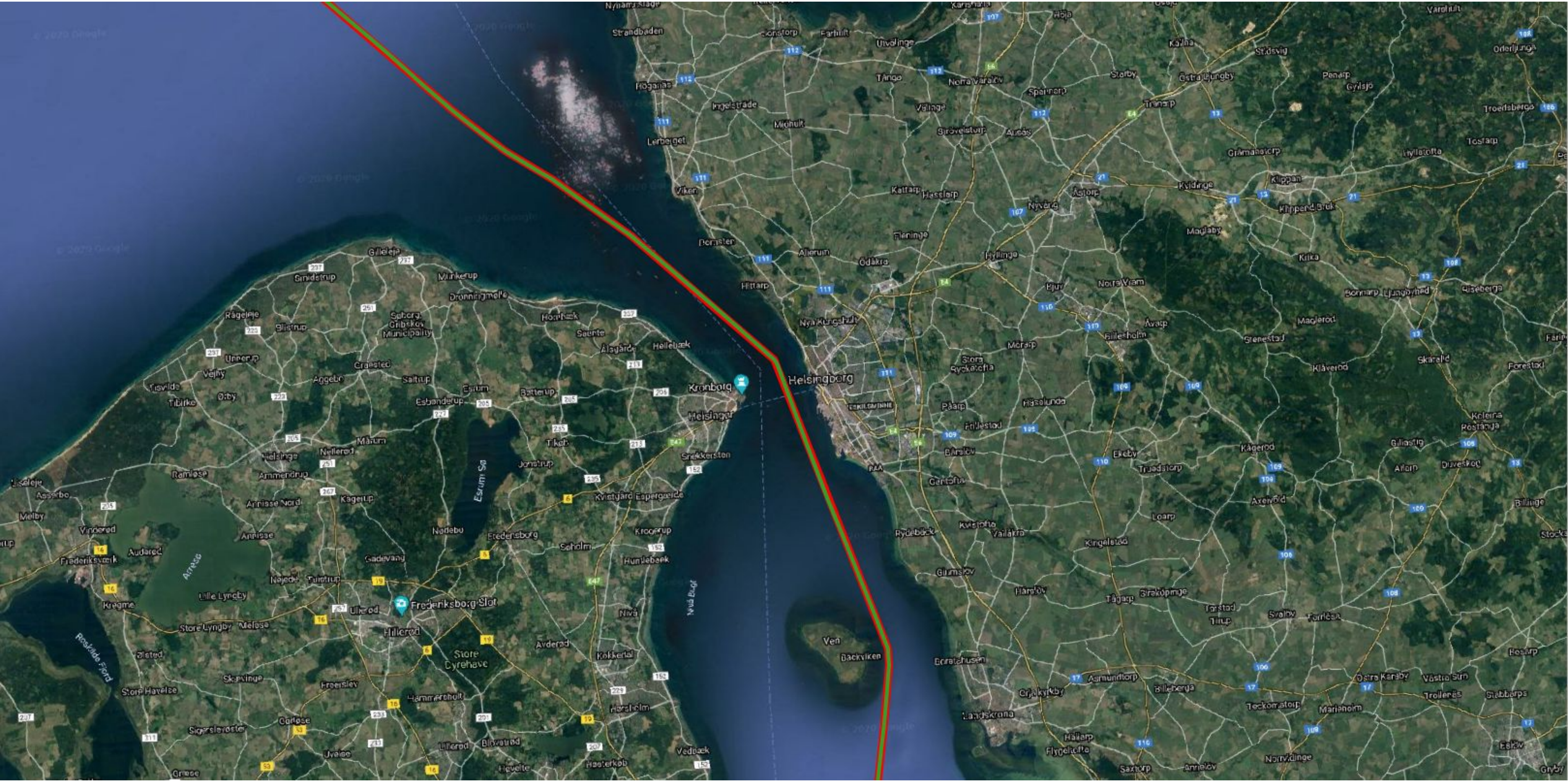
Minimum, maximum, and average number of instants in trips/simplified trips

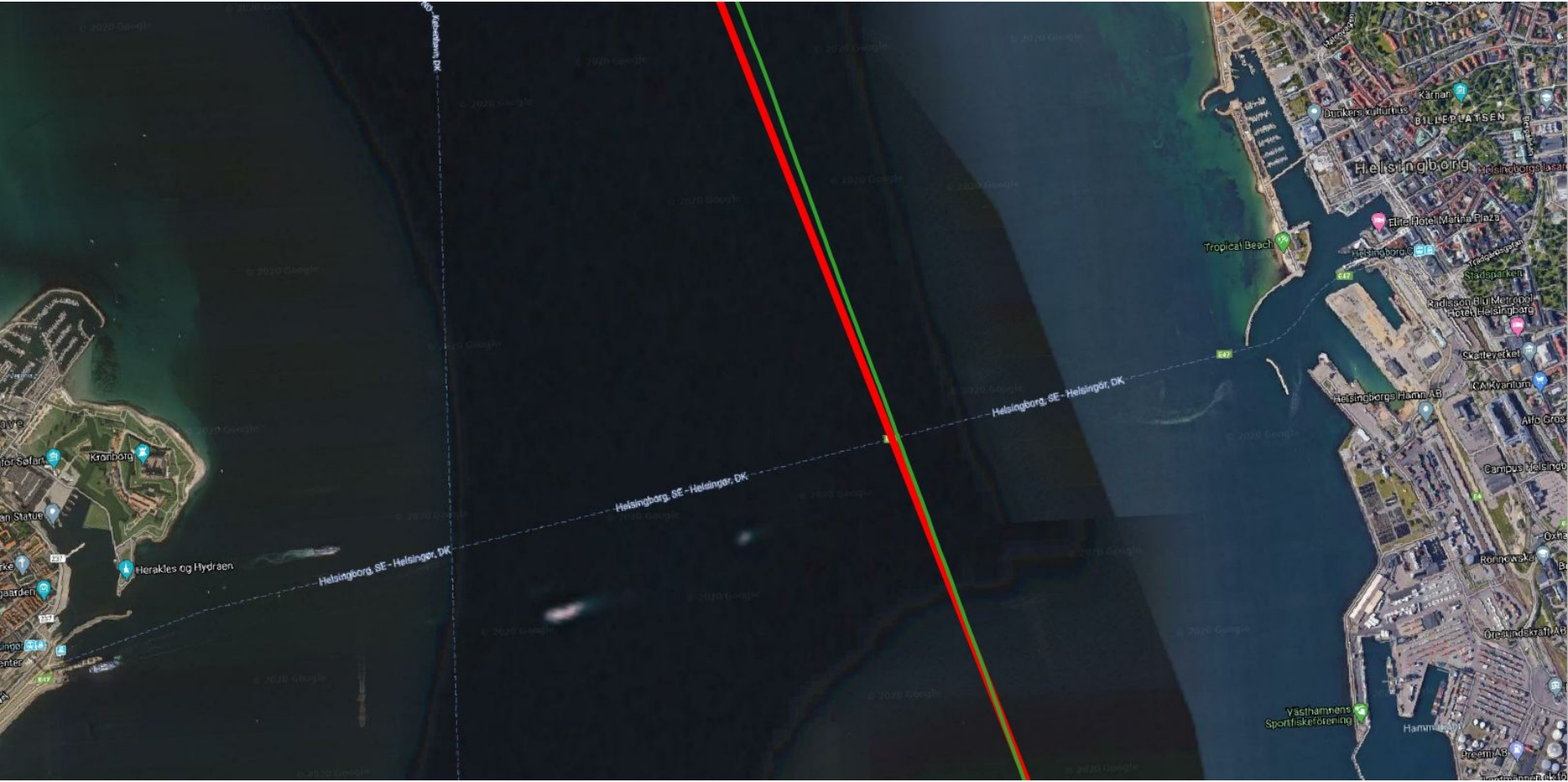
```
SELECT MIN(numInstants(Trip)), MAX(numInstants(Trip)),
    AVG(numInstants(Trip)), MIN(numInstants(SimpTrip)),
    MAX(numInstants(SimpTrip)), AVG(numInstants(SimpTrip))
FROM Ships;
-- 2     23654    3490.44      2        7492     58.55
```

Column size trips/simplified trips

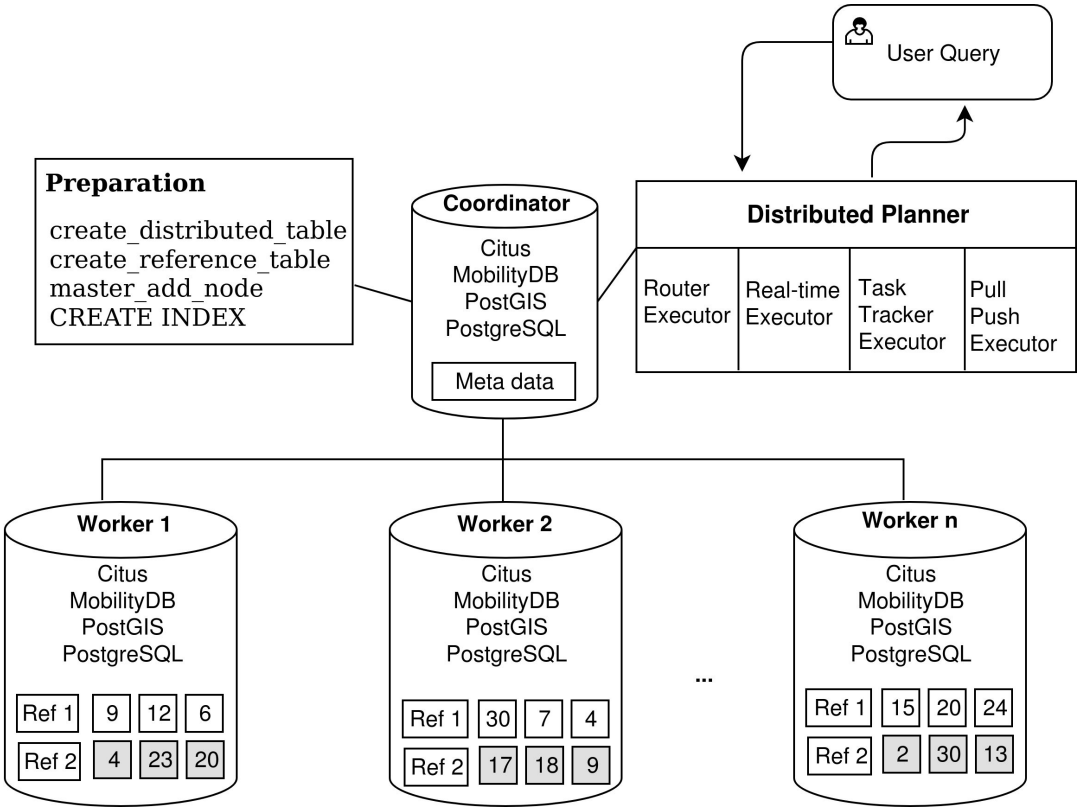
```
SELECT sum(pg_column_size(Trip)), sum(pg_column_size(SimpTrip))
FROM Ships;
-- "322 MB"  "6554 kB"
```


Halmstad
Laholm
Båstad
Ängelholm
Markaryd
Hässleholm
Kristianstad
Karlshamn
Ronneby
Karlskrona
Sölvesborg
Helsingborg
Landskrona
Lund
Malmö
Trelleborg
Ystad
Simrishamn
Copenhagen
Roskilde
Køge
Slagelse
Hillerød
Kastrup
Falsterbo
Eslöv
Höör
Tomelilla
Bornholm
Gudhjem


Helsingborg
Helsingør
Kronborg
Frederiksborg Slot
Hillerød
Ven
Bäckviken
Landskrona
Höganäs
Viken
Hittarp
Arresø
Esrum Sø
Roskilde Fjord
Store Dyrehave
Fredensborg
Gilleleje
Snekkersten
Humlebæk
Nivå
Nivå Bugt
Rydebäck
Råå
Ängelholm
Klippan
Åstorp
Kågeröd
Svalöv
Eslöv
Tisvilde
Frederiksværk
Kvistgård
Espergærde
Hornbæk
Hellebæk


Helsingborg, SE - Helsingør, DK
Helsingborg, SE - Helsingør, DK
Helsingborg, SE - Helsingør, DK
Kronborg
Herakles og Hydraen
Helsingborg
Tropical Beach
Elite Hotel Marina Plaza
Stadsparken
Radisson Blu Metropol Hotel Helsingborg
Helsingborgs Hamn AB
Västhamnens Sportfiskeförening

Distributed MobilityDB Using Citus



Citus Distributed Query Planner: Query Classes

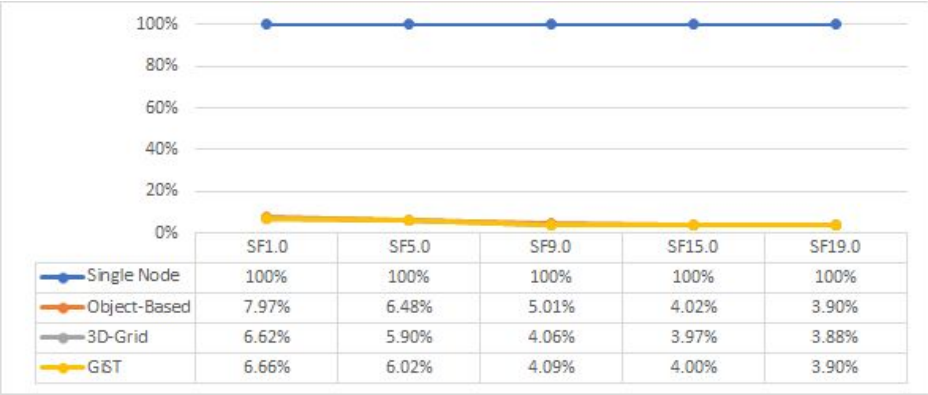
- **Routable queries:** Queries that can be fully evaluated on a subset of workers, the final result is a simple concatenation of the workers results
- Query sent to worker nodes, which optimize it using the regular PostgreSQL planner, executes it, and returns the result to the route executor

Query	Workers	Coordinator
`SELECT *` `FROM Trips` `WHERE length(Trip) > 10000`	`SELECT *` `FROM Trips_1` `WHERE length(Trip) > 10000`	`SELECT *` `FROM Result_1` `UNION` `SELECT *` `FROM Result_2` `...`

Performance

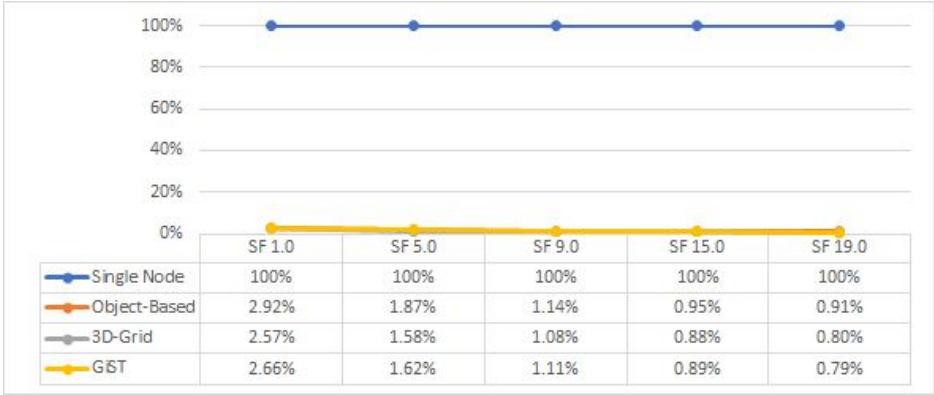
- Dataset generated by BerlinMOD, a benchmark for MOD
  - Simulated trips: to work, from work, leisure
  - Size can be controlled by a scale factor
- Workload: 17 BerlinMOD/R range queries of four categories
  - Object, Temporal, Spatial, Spatiotemporal

Experimental Results: Overall Gain



	SF1.0	SF5.0	SF9.0	SF15.0	SF19.0
Single Node	100%	100%	100%	100%	100%
Object-Based	7.97%	6.48%	5.01%	4.02%	3.90%
3D-Grid	6.62%	5.90%	4.06%	3.97%	3.88%
GiST	6.66%	6.02%	4.09%	4.00%	3.90%

Run time gain on a cluster of 4 nodes



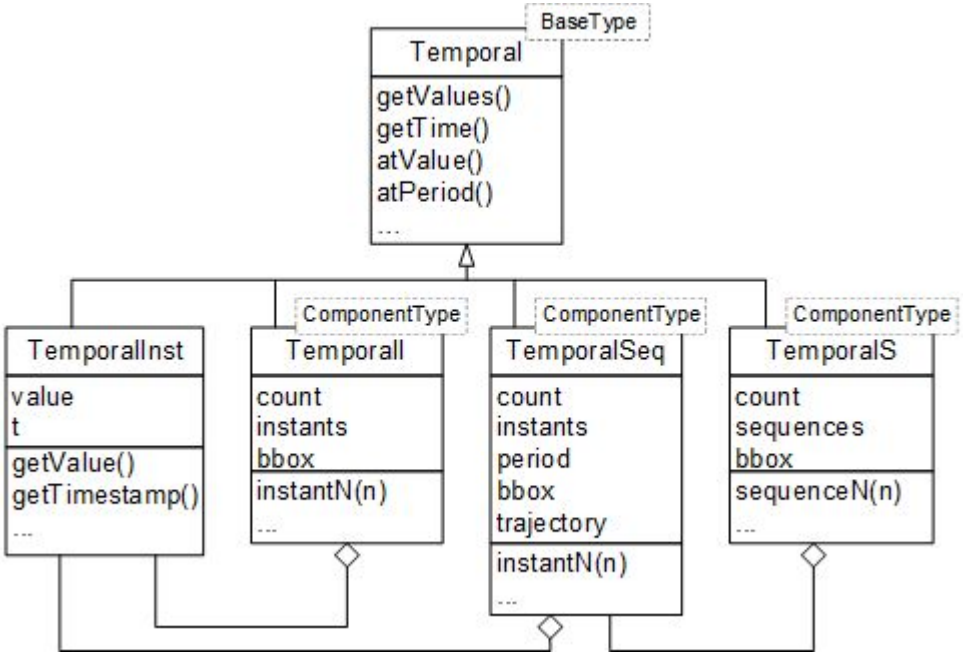
	SF 1.0	SF 5.0	SF 9.0	SF 15.0	SF 19.0
Single Node	100%	100%	100%	100%	100%
Object-Based	2.92%	1.87%	1.14%	0.95%	0.91%
3D-Grid	2.57%	1.58%	1.08%	0.88%	0.80%
GiST	2.66%	1.62%	1.11%	0.89%	0.79%

Run time gain on a cluster of 28 nodes

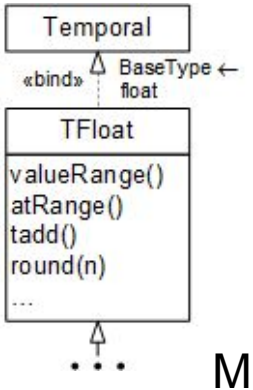
Python Support

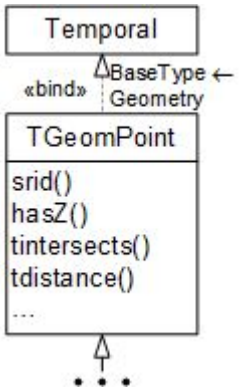
- python-mobilitydb: database adapter to access MobilityDB from Python
- Open source, developed by MobilityDB Team
- Available on Github
- Supports both psycopg2 and asyncpg for PostgreSQL
- Uses postgis adapter for PostGIS
- An adapter for SQLAlchemy has been independently developed
- Also available on Github

Python Classes: UML Diagram



Template Classes



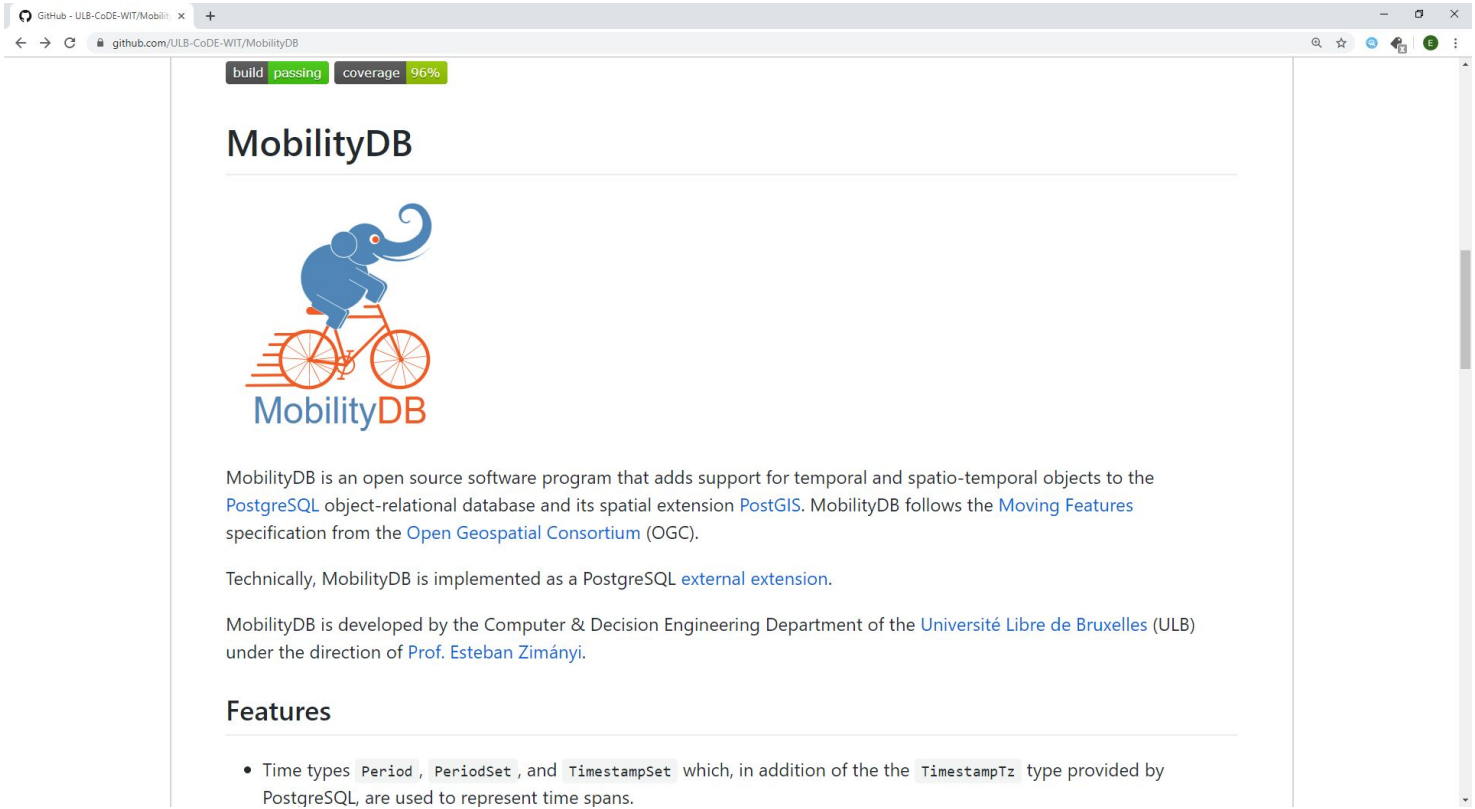


Main Classes

Future Work: Roadmap

- Distribution
  - Enabling non-co-located spatial and spatiotemporal joins
  - Supporting MobilityDB temporal aggregate functions
  - Extending the distributed planner of Citus
- Supporting multiple versions of PostgreSQL/PostGIS
- Development of other modules of the ecosystem
  - Visualization
  - ETL
  - Network-constrained points
  - Generic geometries/geographies
  - Mobility streams
  - ….

MobilityDB on Github



Thanks for listening !

Questions ?