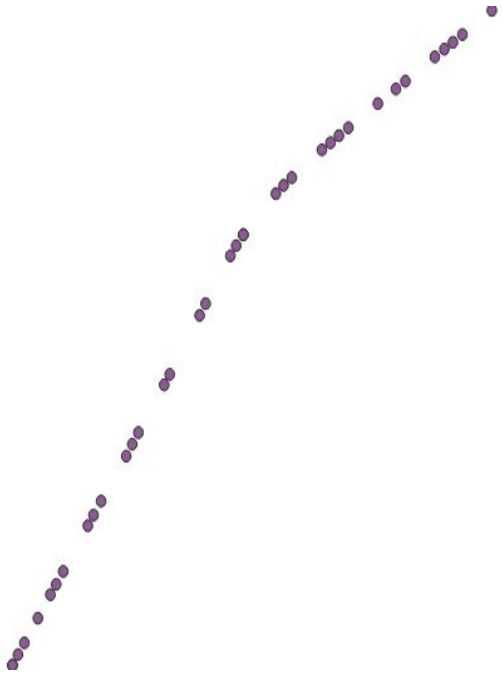
Extending PostgreSQL With Spatiotemporal Data Management

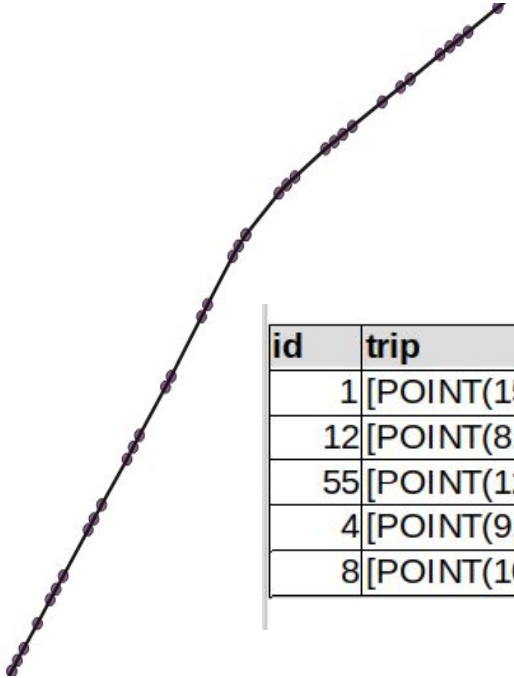
Contacts: Esteban Zimanyi (ezimanyi@ulb.ac.be)
Mahmoud SAKR (mahmoud.sakr@ulb.ac.be)

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

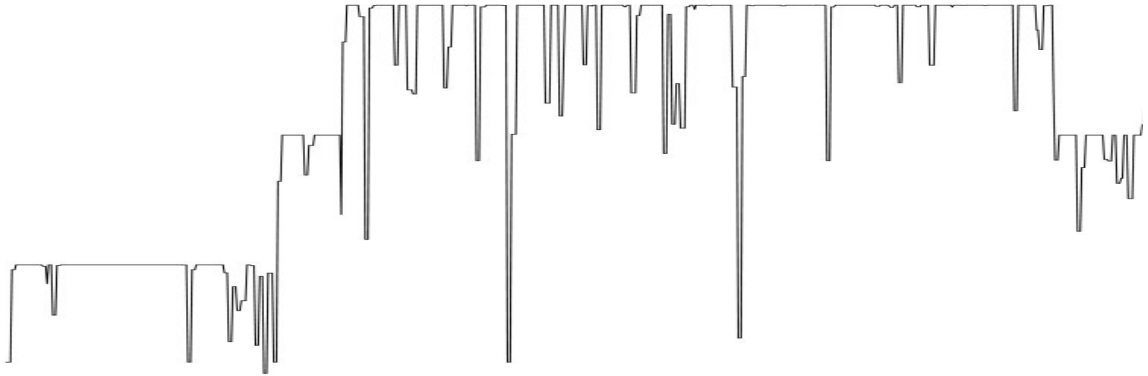
Mobility Data: Trajectories



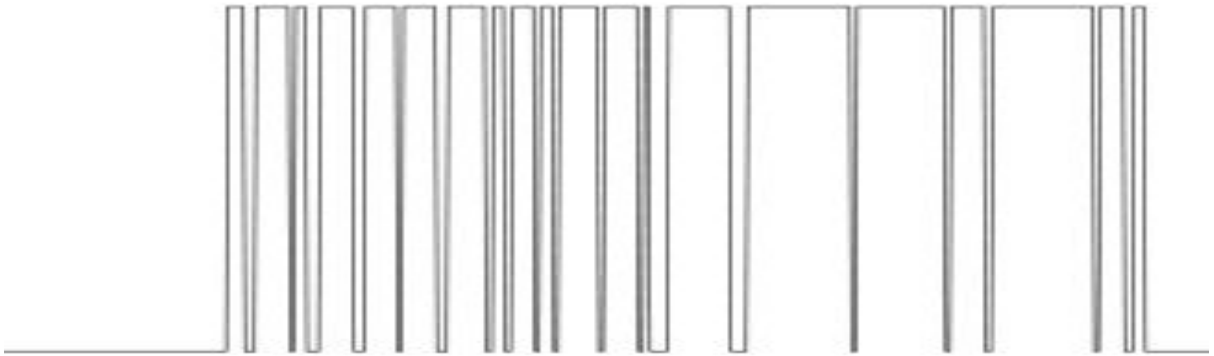
id	trip
1	[POINT(15.839728 55.836783)@2018-04-01 19:34:49+00, POINT(15.831427 55.83403)@2018
12	[POINT(8.067513 57.851652)@2018-04-01 07:35:06+00, POINT(8.073813 57.848518)@2018-
55	[POINT(12.446722 54.689387)@2018-04-01 00:00:00+00, POINT(12.447155 54.689822)@201
4	[POINT(9.752845 55.544552)@2018-04-01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



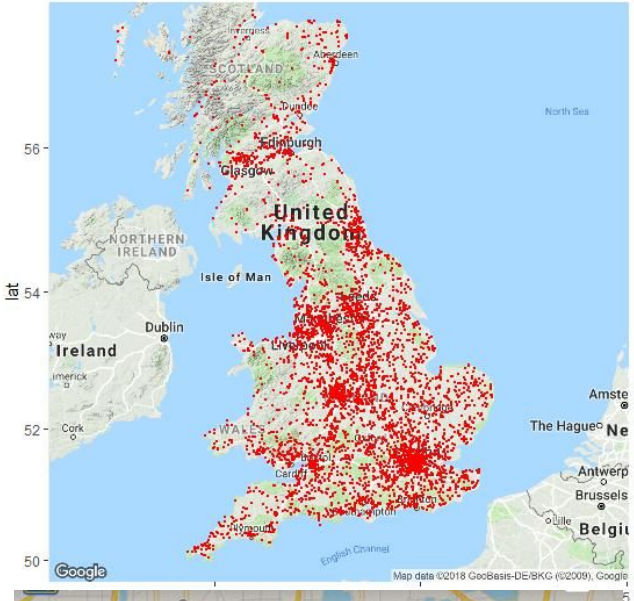
tbool: speed(Trip) > 90



Mobility Data: Points

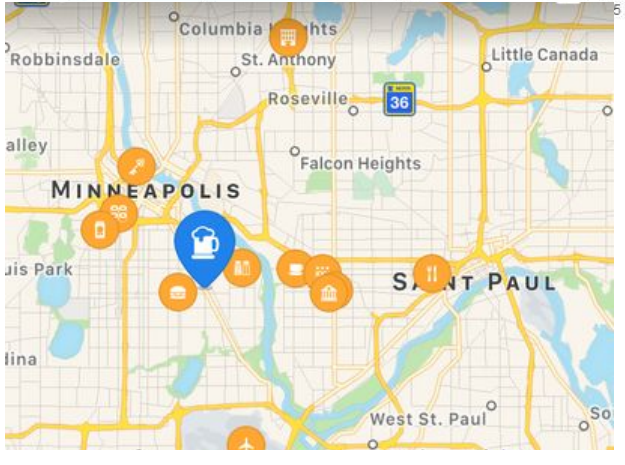
tgeompoint(inst): UK road accidents 2012-14

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



tgeompoint(instants): foursquare check-ins

https://support.foursquare.com/



MobilityDB

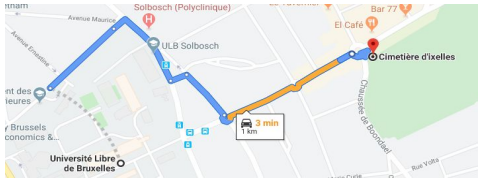
- A moving object database MOD
- Builds on PostgreSQL and PostGIS
- Developed by a team in Université libre de Bruxelles
- OPEN SOURCE extension
- Compliant with OGC standards on Moving Features, and in particular the OGC Moving Features Access

ULB

OGC®

Associate Member

MobilityDB: Architecture



MobilityDB

tgeompoint, tgeogpoint,
tint, tfloat, ttext, tbool

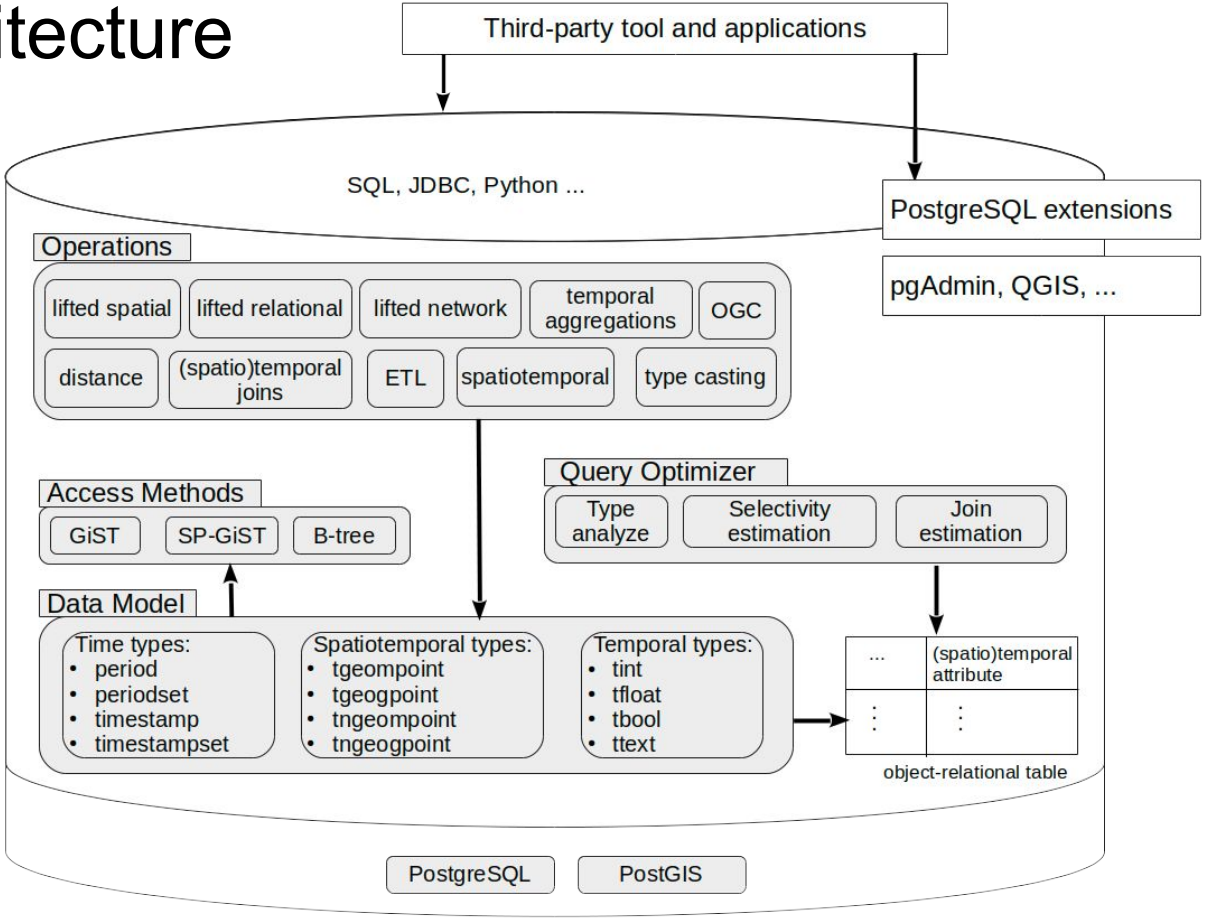
PostGIS

geometry, geography

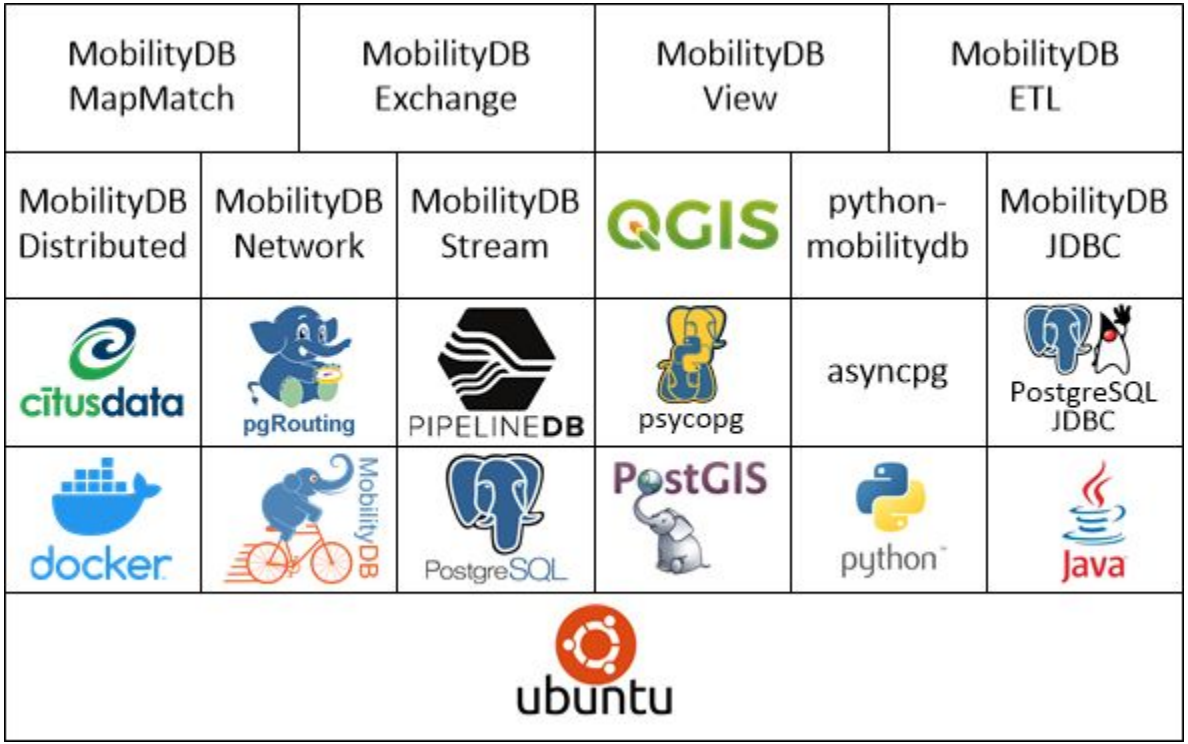
PostgreSQL

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

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

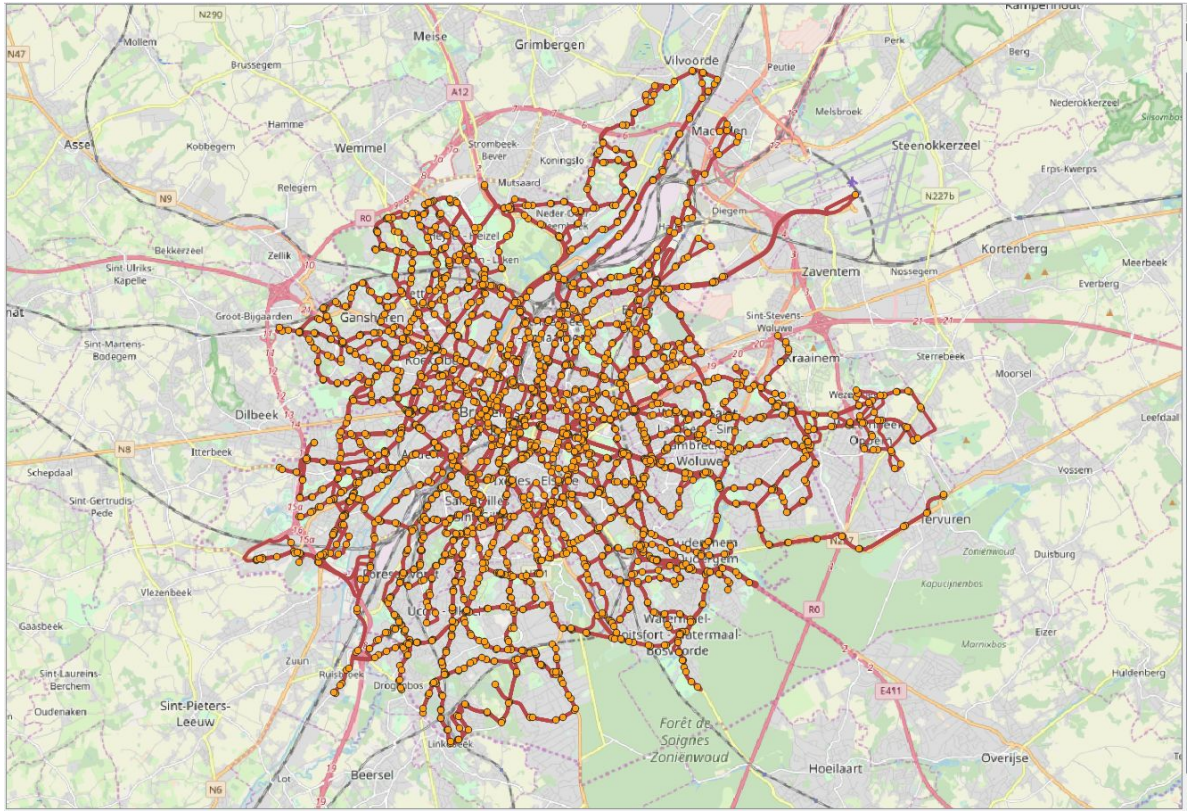
Source: STIB, Brussels

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: Maritime Data (AIS)

Source: Danish Maritime Authority

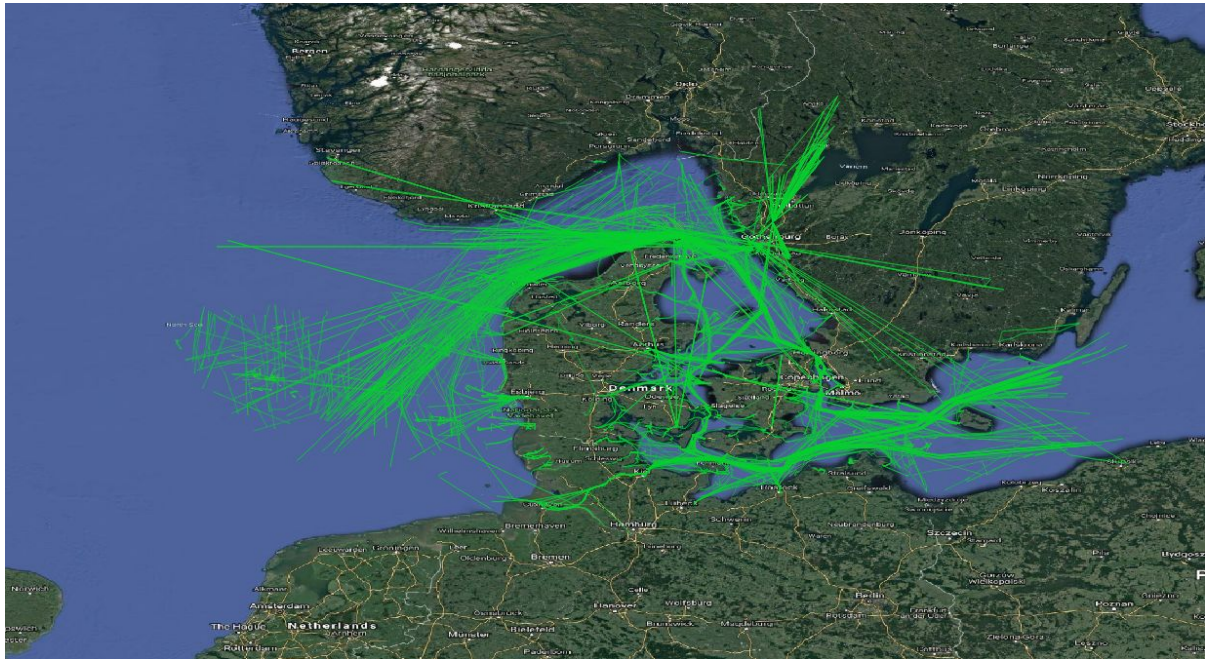
Duration: one day

April 1st 2018

#Rows: 10M

#Trips: 2,995

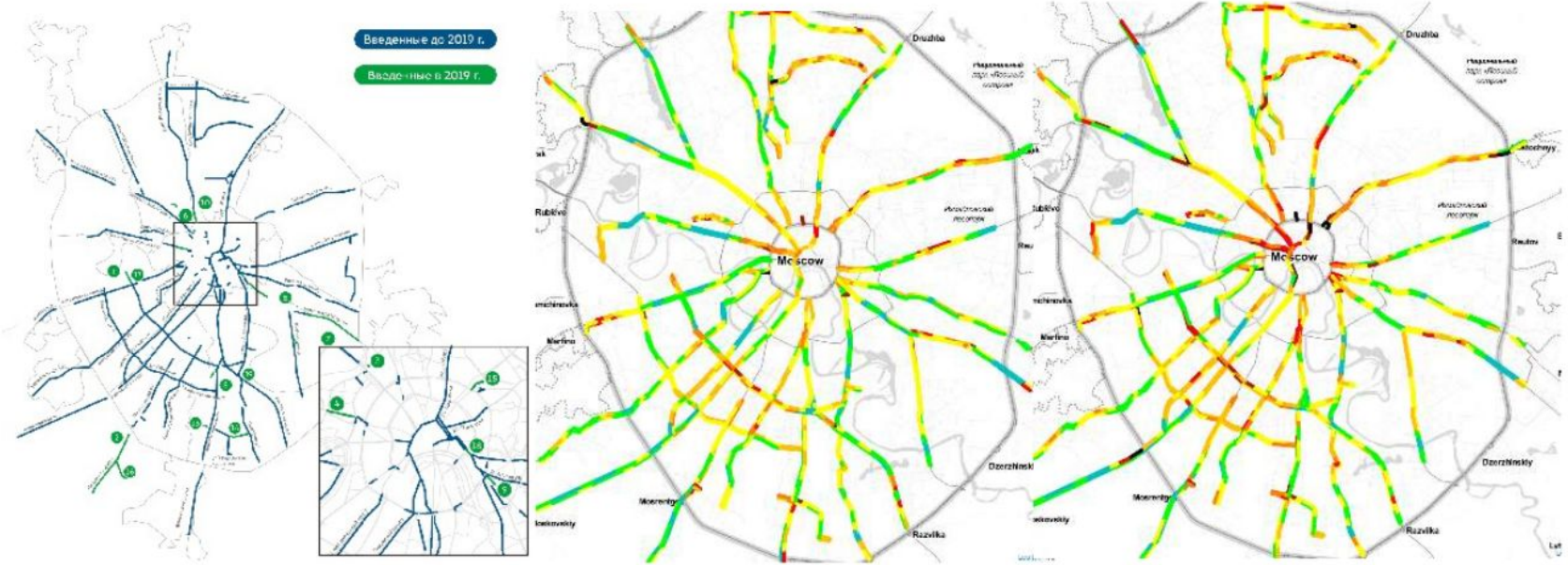
DB size: 1 GB



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

Data analysis – velocity maps

Moscow bus lanes



https://pgconf.ru/en/2020/265266

Quick Example: Spatial Projection

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

List the ships that commute between the ports Rødby and Puttgarden.

```
CREATE INDEX Ships_tripETRS_idx ON Ships USING GiST(tripETRS);


SELECT *
FROM Ships
WHERE intersects( tripETRS, ST_MakeEnvelope(...) ) AND
    intersects( tripETRS, ST_MakeEnvelope(...) )
```

The intersects function is index supported, i.e.,


Rødbyhavn
Bjernæs
Grüner Brink
PUTTGARDEN

Quick Example: Temporal Operations

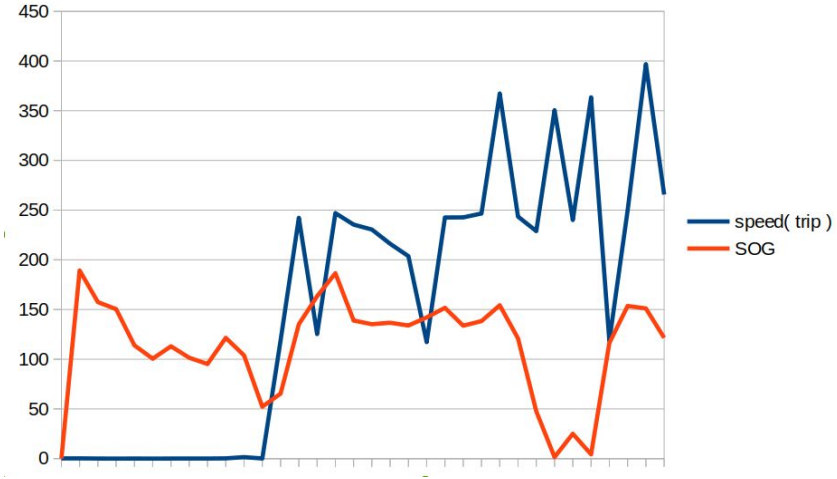
```
TABLE Ships(mmsi integer, trip tgeompoint, sog tfloat, cog tfloat, traj
geometry, tripETRS tgeompoint )
```

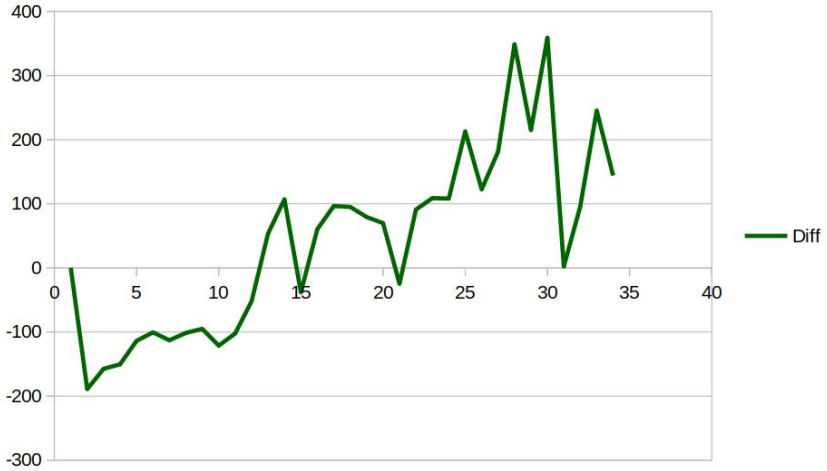
Find all the trips that report SOG very different from the speed calculated from their trajectories (noise, broken sensor, ...).

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

Quick Example: Temporal Operations

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





Quick Example: Aggregation

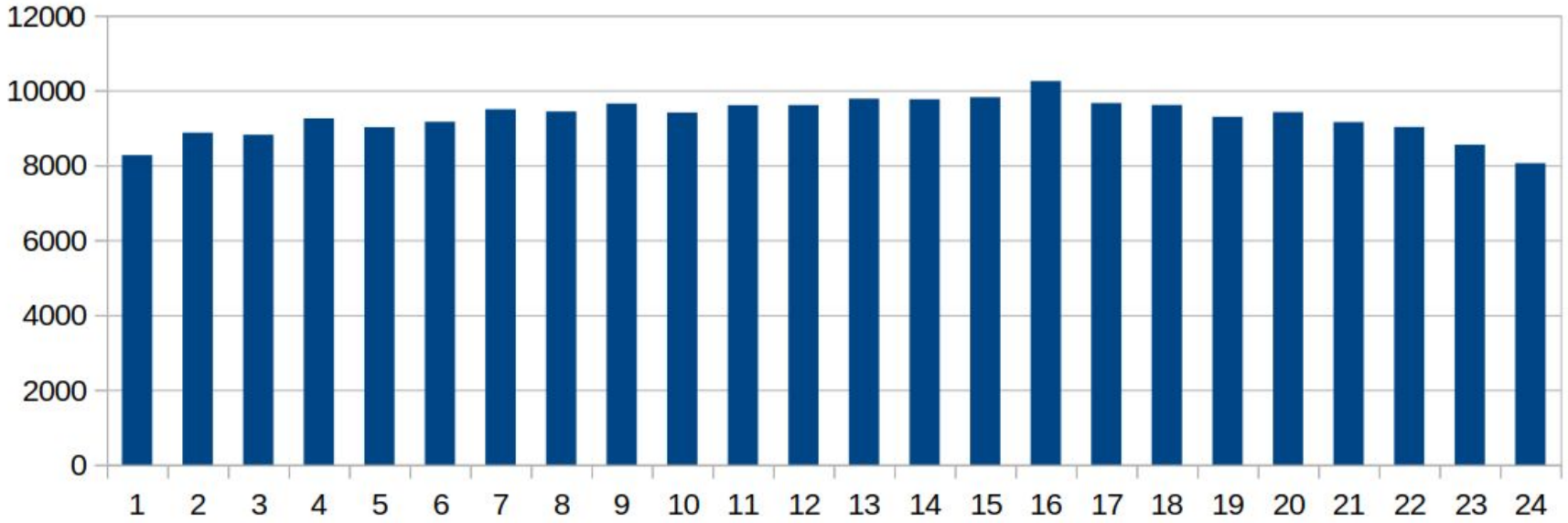
```
TABLE Ships(mmsi integer, trip tgeompoint, sog tfloat, cog tfloat, traj
geometry, tripETRS tgeompoint )
```

What is the 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 00:00:00',
        timestamptz '2018-04-02 00:00:00', interval '1 hour') AS H )
SELECT Period, SUM( length( atPeriod( TripETRS, Period) ) )/1000 travelledKms
FROM TimeSplit T, Ships S
WHERE T.Period && S.Trip
GROUP BY T.Period
ORDER BY T.Period;
```

Quick Example: Aggregation

What is the total distance travelled by ships per hour

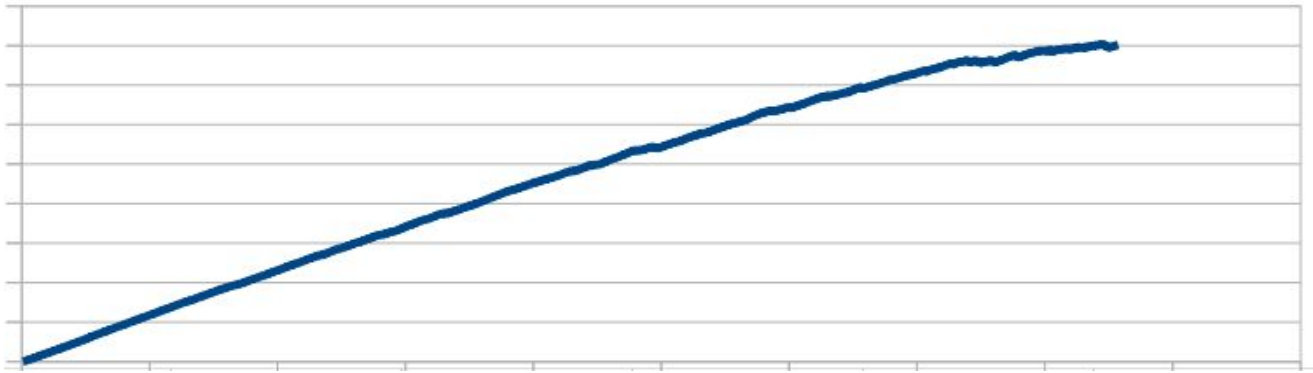


Quick Example: Temporal Aggregation

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

What is the cumulative distance travelled by the company busses at each instant during one week.

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

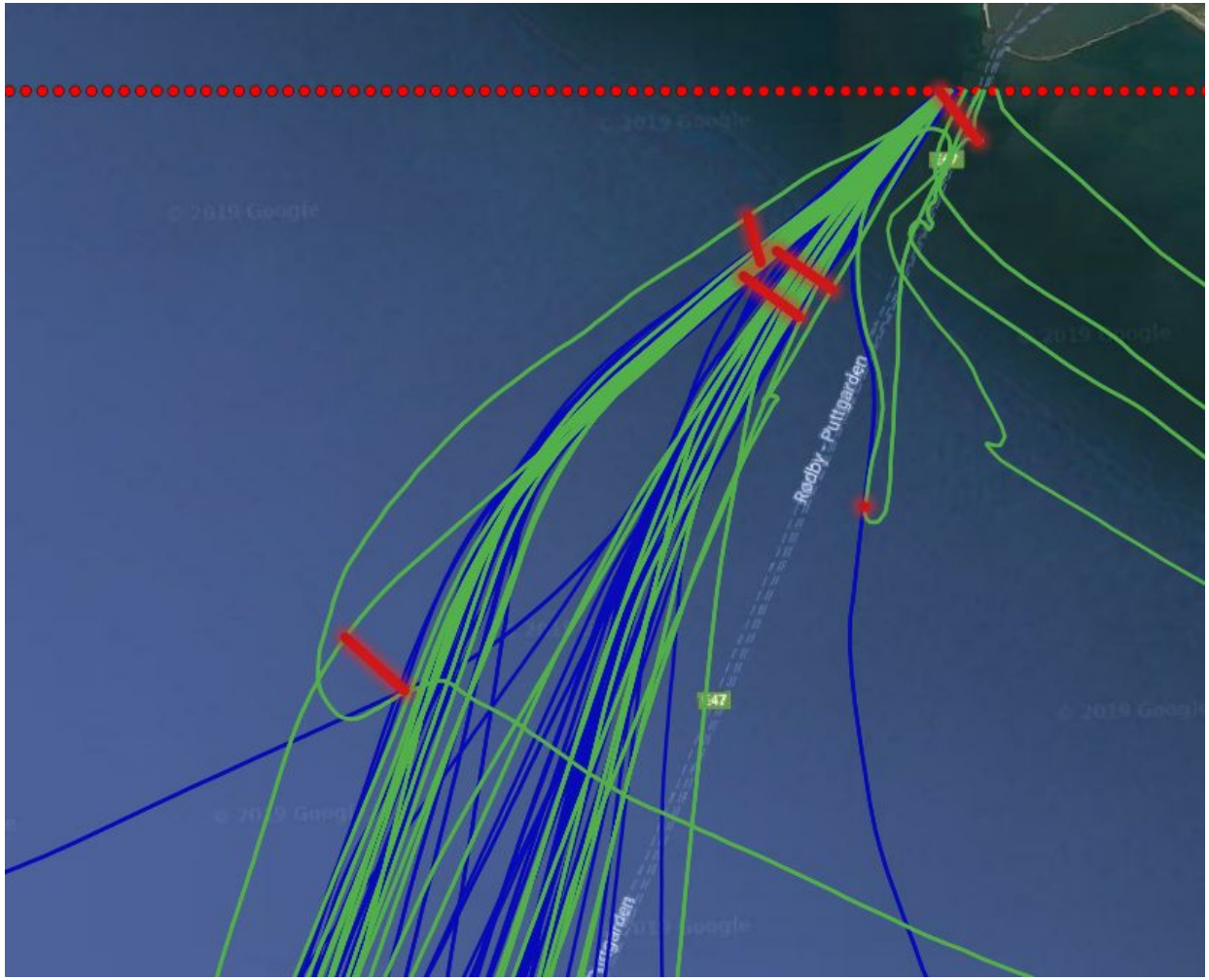


Quick Example: Spatiotemporal Join

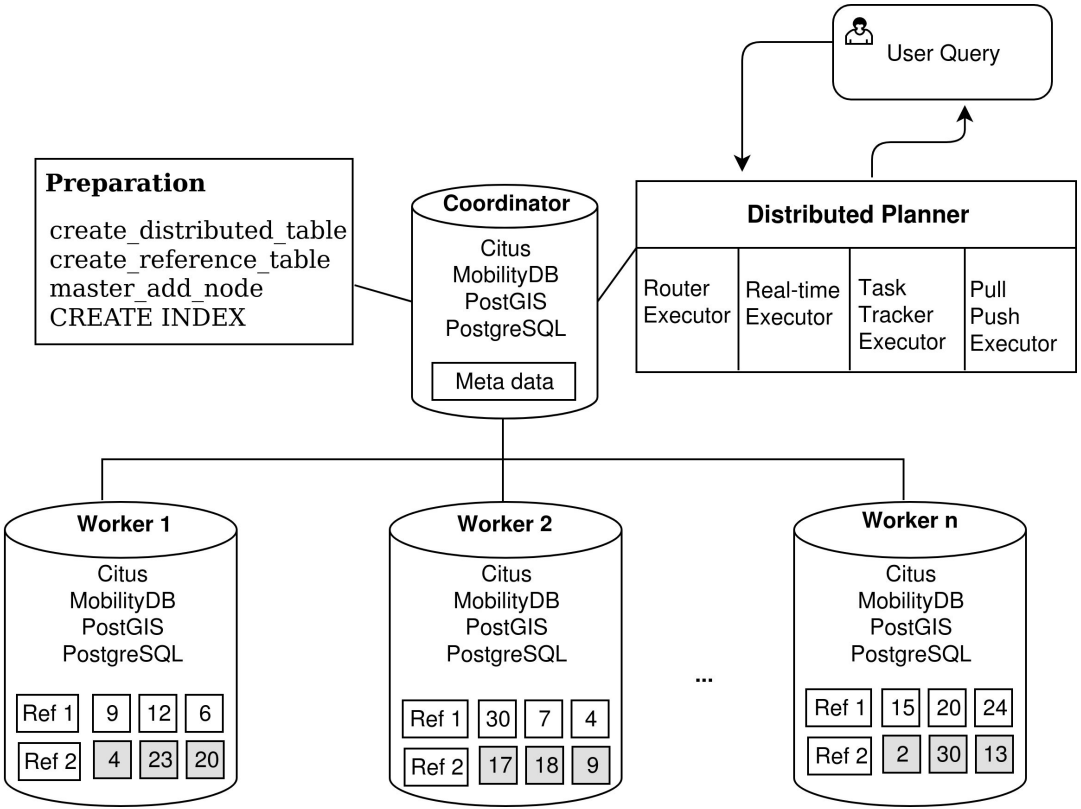
```
TABLE Ships(mmsi integer, trip tgeompoint, sog tfloat, cog tfloat, traj
geometry, tripETRS tgeompoint )
```

Ships that come closer than 300 meters to one another.

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


Rødby - Puttgarden
147

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

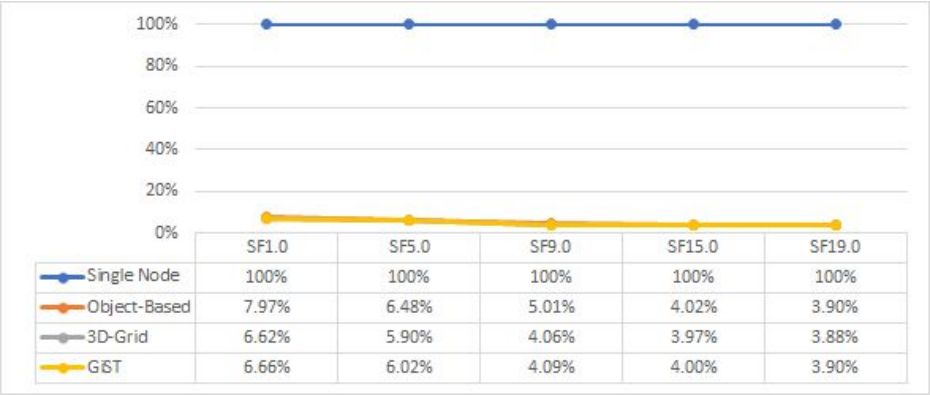
Query	Workers	Coordinator

`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

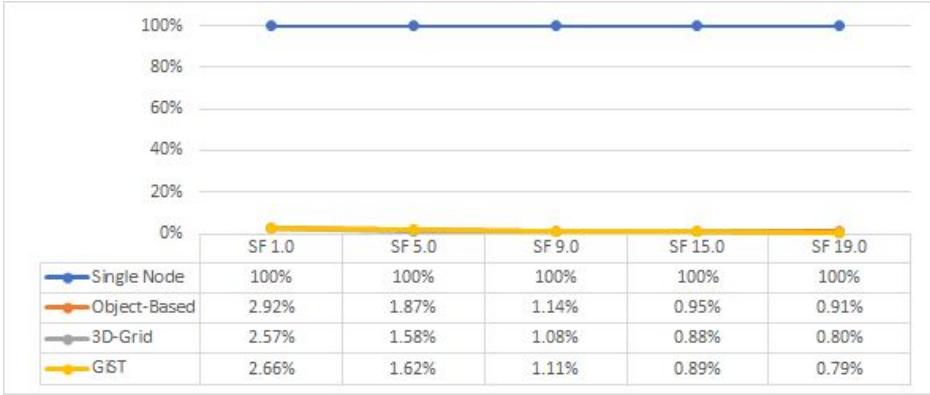
- Dataset generated by BerlinMOD, a benchmark for MOD
  - Simulated trips: to work, from work, leisure
  - Size can be controlled by a scale factor
- Workload: 17 BerlinMOD/R range queries of four categories
  - Object, Temporal, Spatial, Spatiotemporal

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



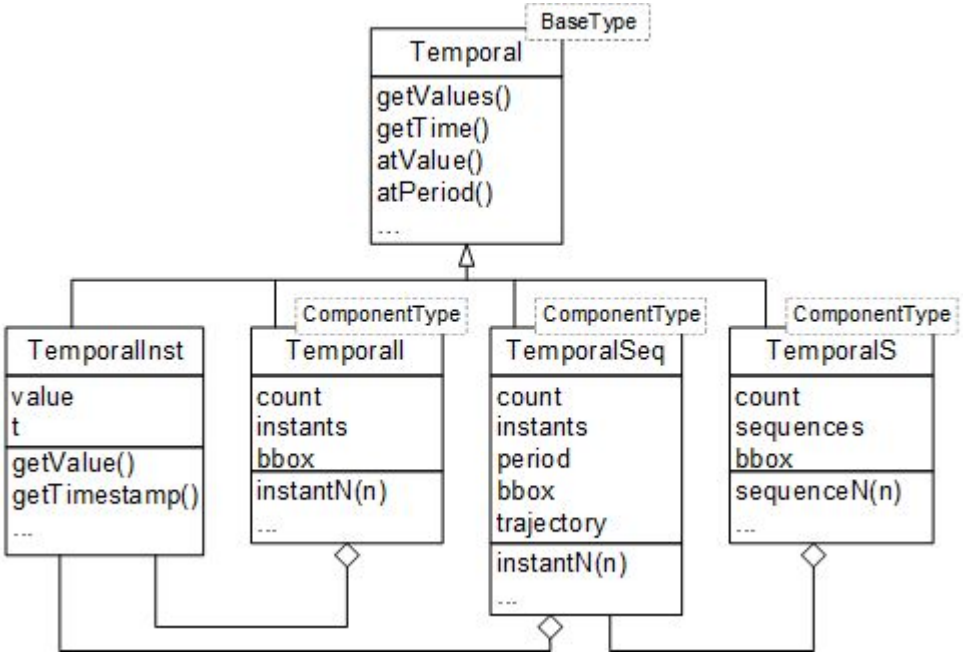
	SF 1.0	SF 5.0	SF 9.0	SF 15.0	SF 19.0
Single Node	100%	100%	100%	100%	100%
Object-Based	2.92%	1.87%	1.14%	0.95%	0.91%
3D-Grid	2.57%	1.58%	1.08%	0.88%	0.80%
GiST	2.66%	1.62%	1.11%	0.89%	0.79%

Run time gain on a cluster of 28 nodes

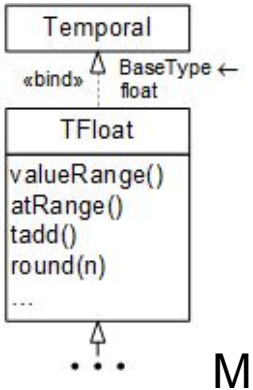
Python Support

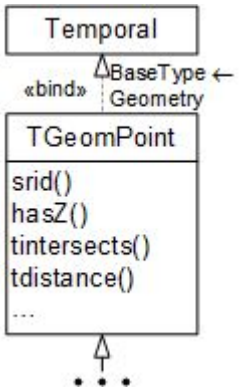
- python-mobilitydb: database adapter to access MobilityDB from Python
- Open source, developed by MobilityDB Team
- Available on Github
- Supports both psycopg2 and asyncpg for PostgreSQL
- Uses postgis adapter for PostGIS
- An adapter for SQLAlchemy has been independently developed
- Also available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

Python Classes: UML Diagram



Template Classes





Main Classes

Python: Usage Example (1)

```
import psycopg2
from mobilitydb.psycopg import register

# Set the connection parameters to PostgreSQL
connection = psycopg2.connect(host='localhost', database='test',
        user='mobilitydb', password='')
connection.autocommit = True

# Register MobilityDB data types
register(connection)

# Open a cursor to perform database operations
cursor = connection.cursor()
```

Python: Usage Example (2)

```
# Insert a row
carid = 1
tripid = 2
trip = TGeomPointSeq(['POINT(1.0 1.0)@2019-09-01',
        'POINT(2.0 2.0)@2019-09-02'], srid=4326)
insert_query = "INSERT INTO trips(carid, tripid, trip) VALUES(%s, %s, %s)"
result = cursor.execute(insert_query, (carid, tripid, trip))
connection.commit()
print(cursor.rowcount, "record(s) inserted successfully into trips table")

# Close the connection
if connectionObject:
        connectionObject.close()
```

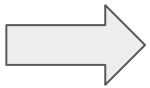
Future Work: Roadmap

- 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
- Continue development of other modules of the ecosystem
  - Visualization, ETL, generic geometries/geographies, streaming ….

anytemporal

- PostgreSQL has a closed list of pseduo-types: any, anyelement, anyarray, anyrange.
- It would be nice to make this list extensible, so that one can add anytemporal.
- Significant reduction in the number of MobilityDB functions.

```
CREATE FUNCTION startValue(tgeompoint)
CREATE FUNCTION startValue(tgeogpoint)
CREATE FUNCTION startValue(tbool)
CREATE FUNCTION startValue(tint)
CREATE FUNCTION startValue(tfloat)
CREATE FUNCTION startValue(ttext)
```



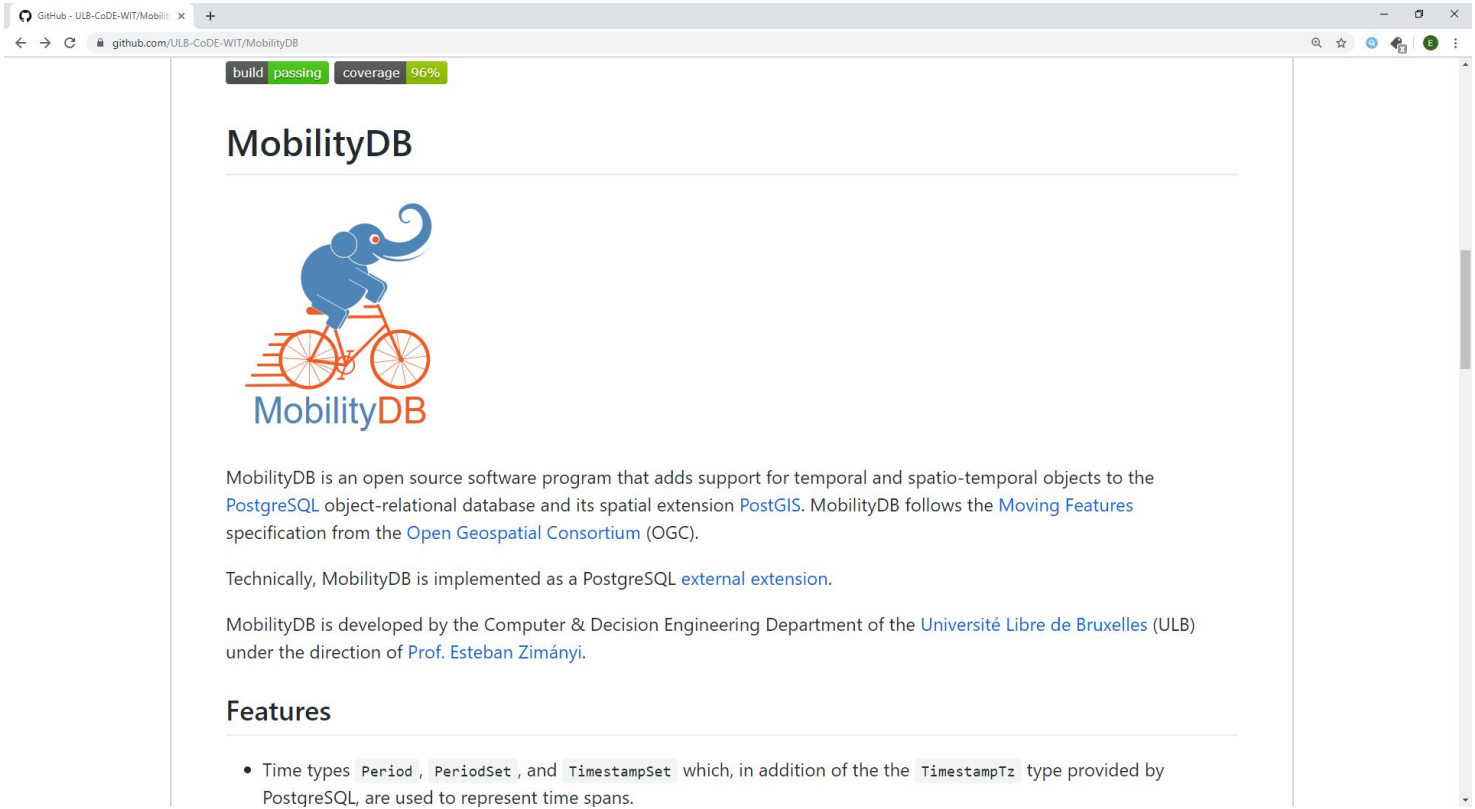
```
CREATE FUNCTION startValue(anytemporal)
```

store_function

- Memory structure different than persistent desk structure.
- A temporal point caches the spatial trajectory as a linstring.
- Eager or lazy ?

```
CREATE TYPE name (
    INPUT = input_function,
    OUTPUT = output_function
    [ , RECEIVE = receive_function ]
    [ , SEND = send_function ]
    …
```

MobilityDB on Github



Thanks for listening !

Questions ?