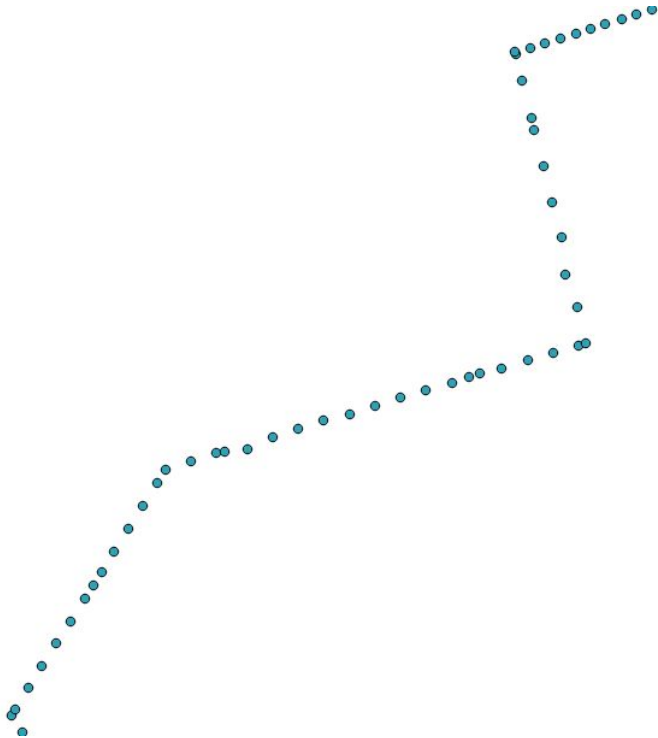




# MobilityDB

A PostgreSQL-PostGIS Extension for Mobility Data  
Management

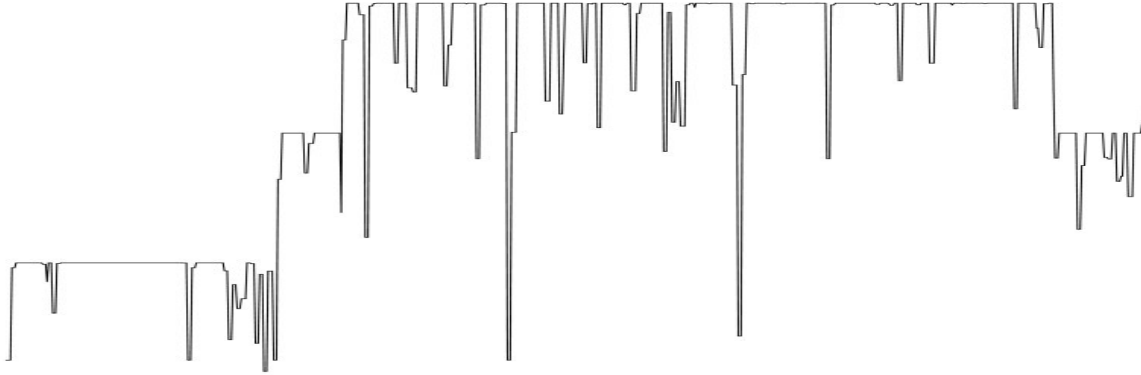
# What is Mobility Data ?



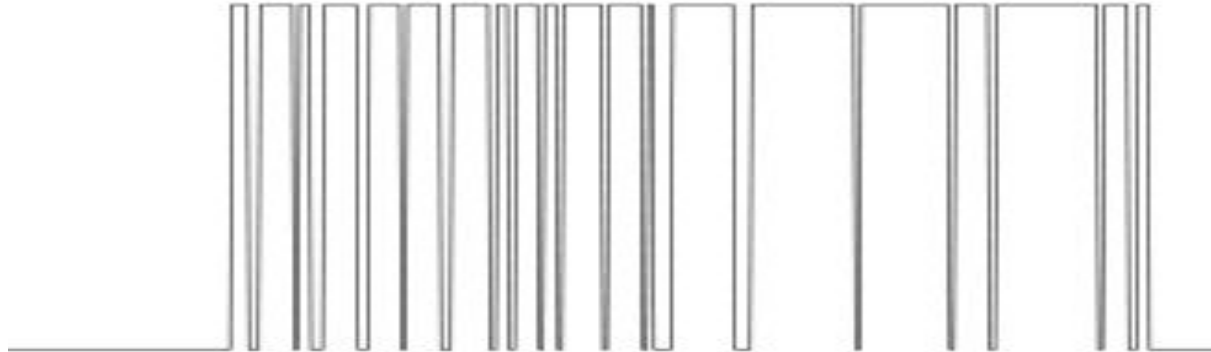
moid	tripid	tstart	xstart	ystart
1	2	2007-05-28T08:36:47	13.43593	52.41721
1	2	2007-05-28T08:36:49	13.43605	52.41723
1	2	2007-05-28T08:36:51	13.43628	52.41727
1	2	2007-05-28T08:36:53	13.43652	52.4173
1	2	2007-05-28T08:36:55	13.43676	52.41734
1	2	2007-05-28T08:36:57	13.437	52.41737
1	2	2007-05-28T08:36:59	13.43719	52.41741
1	2	2007-05-28T08:37:01	13.43739	52.41744
1	2	2007-05-28T08:37:03	13.43762	52.41747
1	2	2007-05-28T08:37:05	13.43786	52.41751
1	2	2007-05-28T08:37:07	13.43809	52.41755

# But Also

tfloat: speed(Trip).



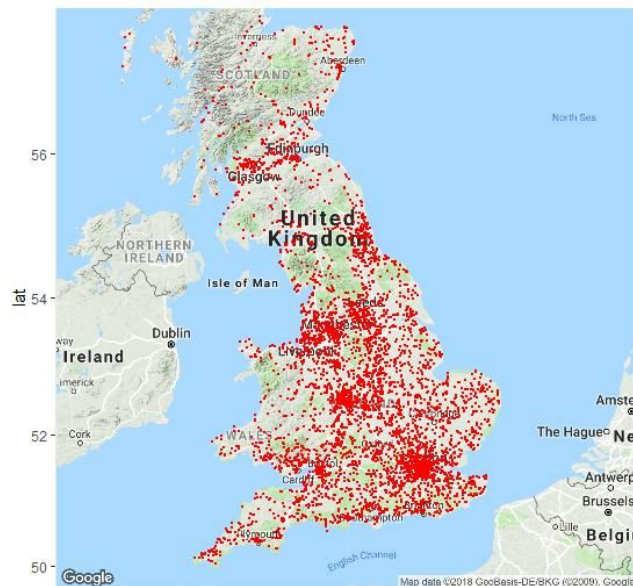
tbool: speed(Trip) > 90



# But Also

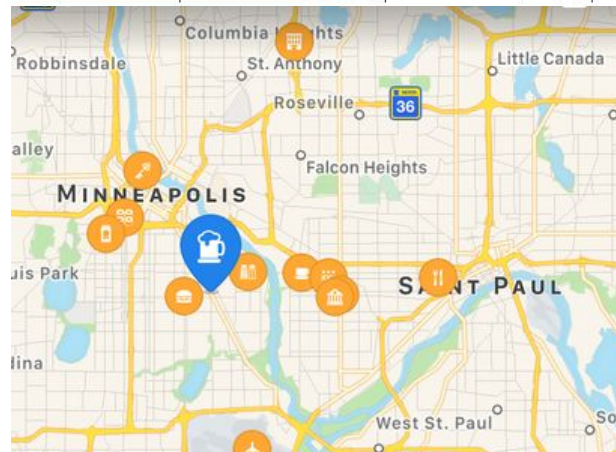
tgeogpoint(inst): UK road accidents 2012-14

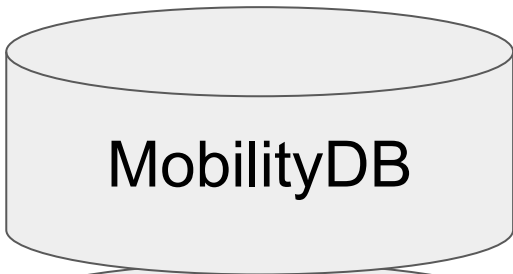
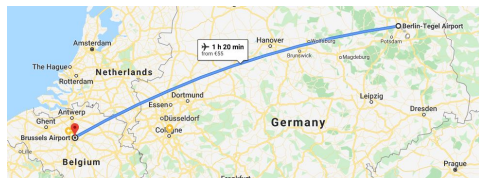
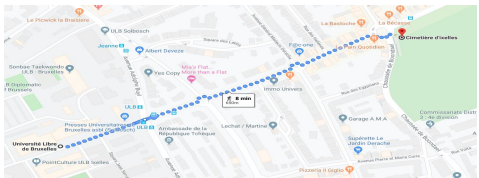
<https://www.kaggle.com/daveianhickey/2000-16-traffic-flow-england-scotland-wales>



tgeogpoint(instants): foursquare check-ins

<https://support.foursquare.com/>

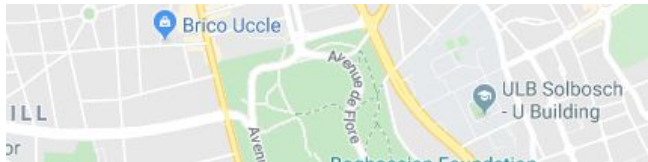




tgeompoint, tgeogpoint,  
tint, tfloat, ttext, tbool

geometry, geography

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




GPX

```
graph TD; GPX[GPX] --> CreateTable[CREATE TABLE Trips AS...]; CreateTable --> FilterQuery[SELECT tripid FROM trips t WHERE speed(trip) @> 90]; FilterQuery --> Ellipsis[...];
```

```
CREATE TABLE Trips AS
  SELECT vehicleId, tripId, tgeompointseq(
    array_agg(
      tgeompointinst( ST_Point(lon, lat), t) order by t
    )) AS trip
FROM gpx
GROUP BY vehicleId, tripId
ORDER BY vehicleId, tripId
```

```
SELECT tripid
FROM trips t
WHERE speed(trip) @> 90
```

...

# Without MobilityDB

GPX

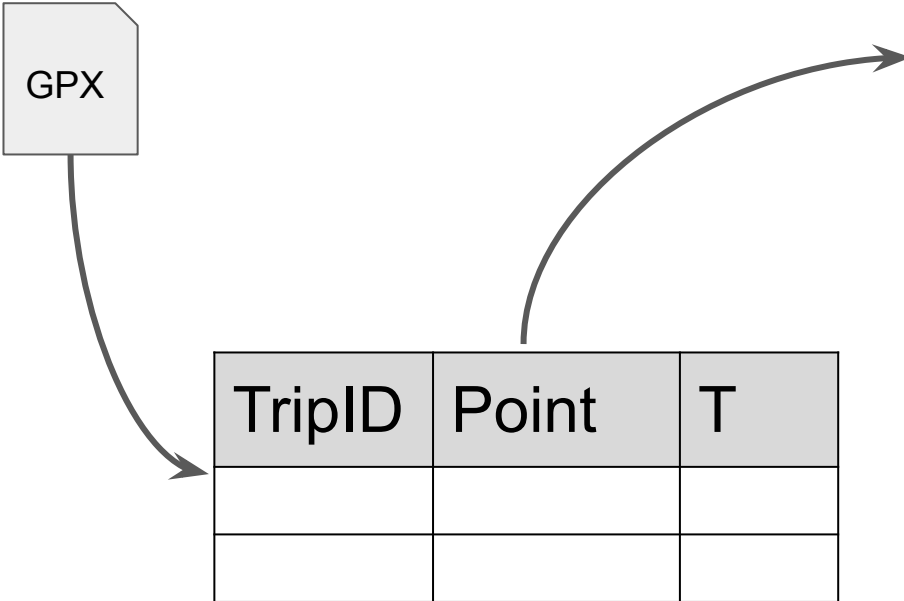
TripID	Point	T

```
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 TripPoints  
WHERE ST_Distance(Point2, Point1) /  
      (T2 - T1) > 90
```

...

# Without MobilityDB

GPX



TripID	Point	T

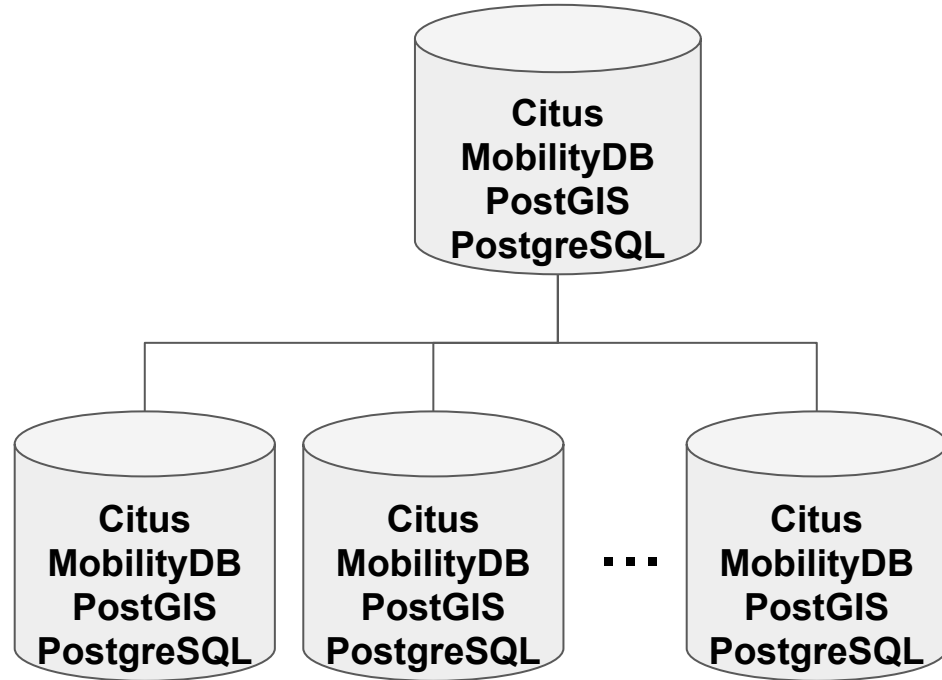
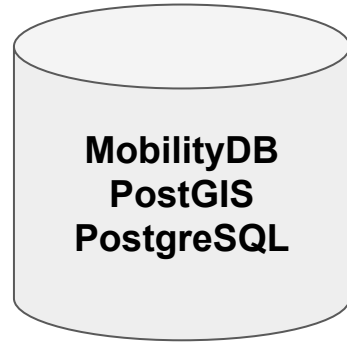
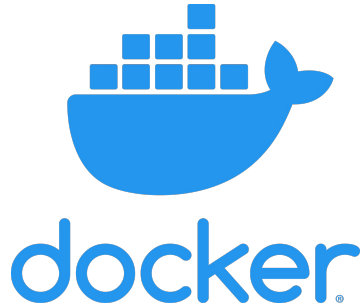
```
WITH TripSegs AS (  
    SELECT TripID, Point AS P1, T AS T1  
           LAG (Point) OVER W AS P2,  
           LAG (T) OVER W AS T2,  
    FROM TripPoints WINDOW W AS (  
        PARTITION BY TripID ORDER BY T)  
    )  
SELECT DISTINCT(TripID)  
FROM TripPoints  
WHERE ST_Distance(Point2, Point1)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

...

What if the trips contain temporal gaps ?



# Installation Options



# Loading Data: CSV, LocationHistory, GPX, GTFS

```
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

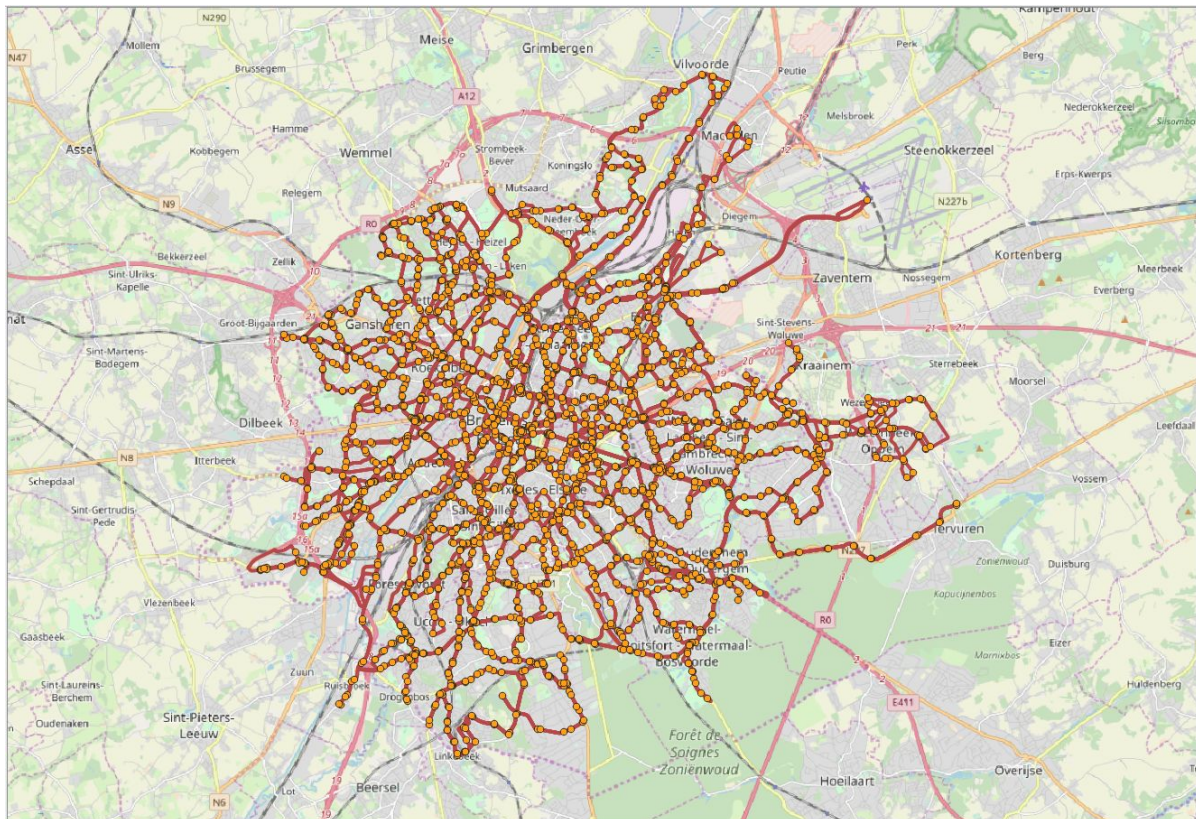
**Source:** STIB, Brussels

**Duration:** 28 days

7 Oct- 3 Nov 2019

**#Trips:** 445,187

**DB size:** 9 GB



# Quick Example: Spatial Projection

```
TABLE Bus( LineNo integer, TripNo integer, Route tgeompoint( Sequence, Point, 3812 ) );
```

```
TABLE POI ( POINo integer, Name TEXT, Geo GEOMETRY(POINT, 3812) );
```

List the bus lines that traverse Louise square.

```
SELECT TripNo
```

```
FROM Bus B, (SELECT P.Geo FROM POI P WHERE P.Name = 'Place Louise' LIMIT 1) T
```

```
WHERE intersects(B.Route, T.Geo)
```

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

```
'SELECT $1 OPERATOR(@extschema@.&&) $2 AND @extschema@._intersects($1,$2)'
```

# Quick Example: Temporal Predicate

```
TABLE Bus( LineNo integer, TripNo integer, Route tgeompoint( Sequence, Point, 3812) );
```

```
TABLE Network( LineNo integer, Route GEOMETRY(LINESTRING, 3812) );
```

Find all the trips that did not deviate from their line routes.

```
SELECT TripNO
```

```
FROM Bus B, Network N
```

```
WHERE st_buffer(N.Route, 20) && B.Route AND
```

```
    tcontains(st_buffer(N.Route, 20), B.Route) @= TRUE
```

The && operator performs a bounding box index filtering.

# Quick Example: Traditional Aggregation

```
TABLE Bus( LineNo integer, TripNo integer, Route tgeompoint( Sequence, Point, 3812 ) );
```

What is the total distance travelled by the company buses per week.

```
SELECT SUM( length(Trip) ) travelled, date_part('week', startTimestamp(Trip)) AS weekly,  
FROM Bus  
GROUP BY weekly;
```

# Quick Example: Temporal Aggregation

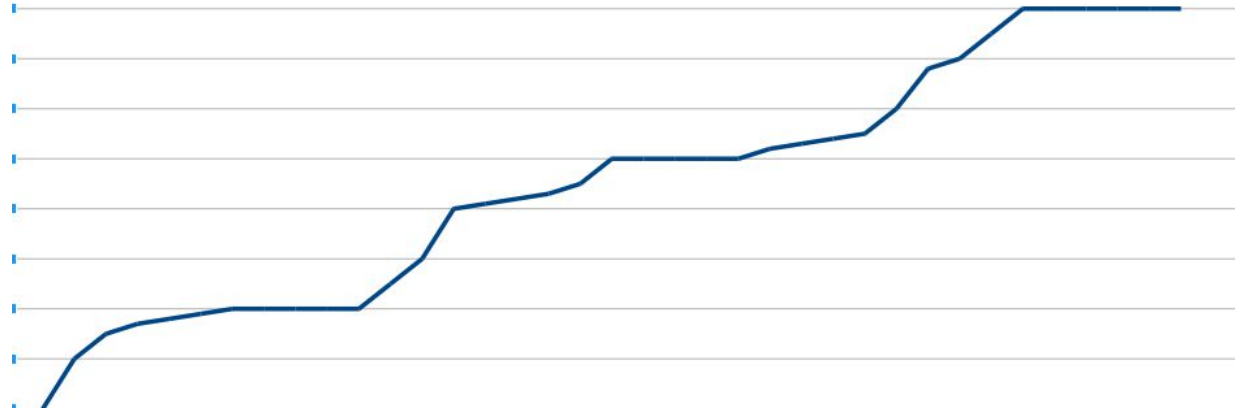
```
TABLE Bus( LineNo integer, TripNo integer, Route tgeompoint( Sequence, Point, 3812 ) );
```

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

```
SELECT tsum( cumulativeLength(Trip) ) travelled, date_part('week', startTimestamp(Trip)) AS weekly,
```

```
FROM Bus
```

```
GROUP BY weekly;
```



# Quick Example: Spatiotemporal Join

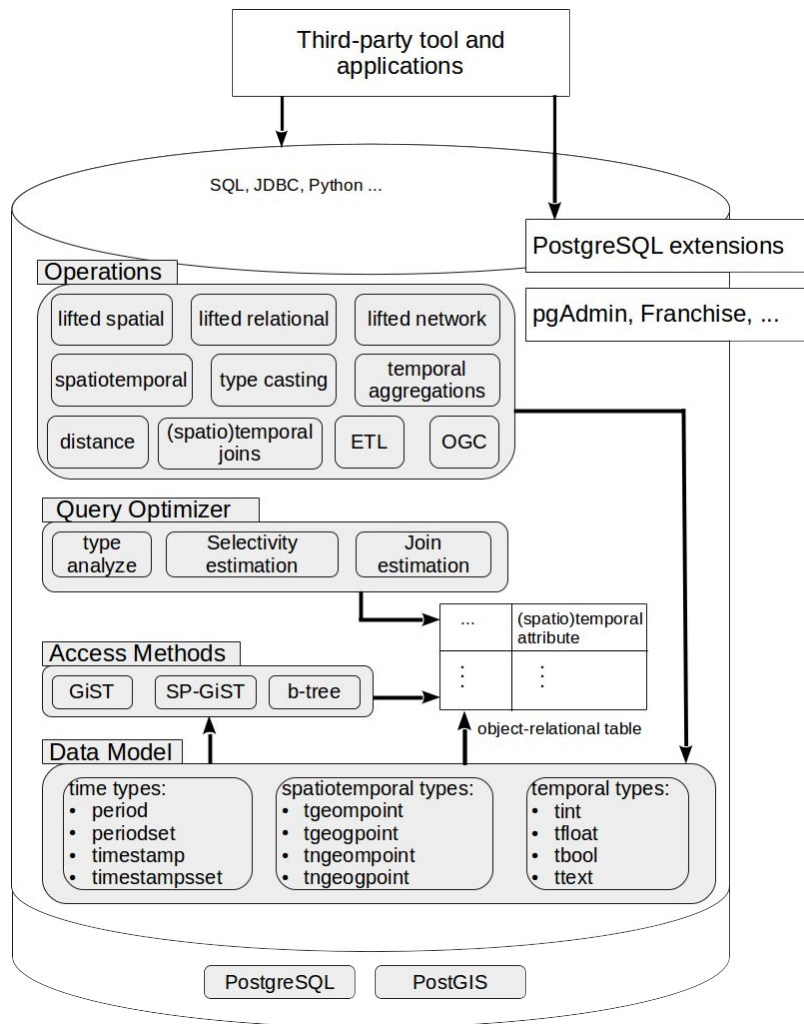
```
TABLE Bus( LineNo integer, TripNo integer, Route tgeompoint( Sequence, Point, 3812) );
```

List all transit opportunities during Jan 2019. A transit opportunity is when two buses from different lines meet at a station, so the passenger have the opportunity to immediately change the line.

```
WITH BusStops AS(
SELECT TripNO, atGeometry(B.Route, S.Geo) RestrictedRoute
FROM Bus B, Stops S )
  SELECT A.TripNO, B.TripNO
  FROM BusStops A, BusStops B
  WHERE
    A.LineNO < B.LineNO AND A.TripNO < B.TripNO AND
    twithin(A.RestrictedRoute, B.RestrictedRoute, 50) &= TRUE
```



# Features





# MobilityDB Eco-system

MobilityDB MapMatch		MobilityDB Exchange		MobilityDB ETL		MobilityDB View	
MobilityDB Distributed	MobilityDB Network		MobilityDB Stream	MobilityDB Python		MobilityDB JDBC	
Citius	PgRouting		PipelineDB	Psycopg 2.8		PostgreSQL JDBC 42.2.6	
MobilityDB		PostgreSQL 11 PostGIS 2.5		Python 3.7		Java 11	
Ubuntu 18.04.2 LTS							



# MobilityDB

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

# Thanks for listening !

Questions ?