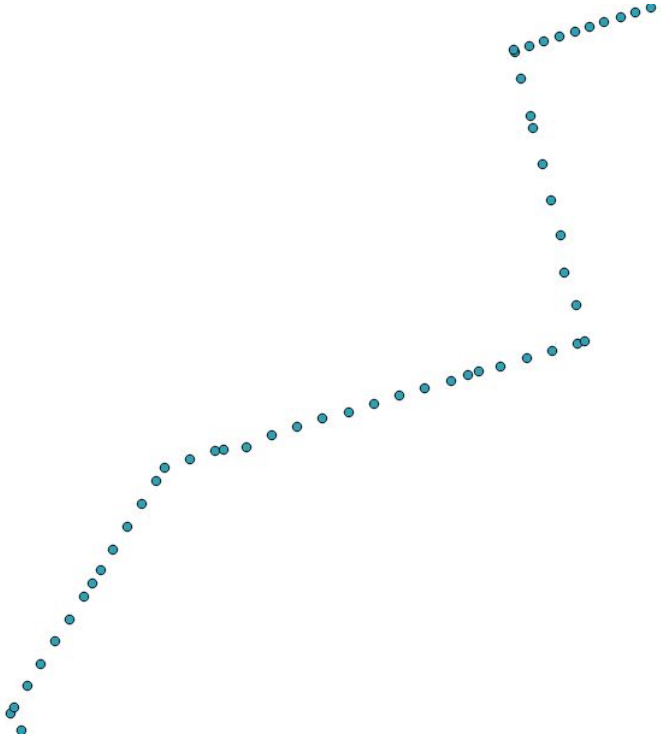
# MobilityDB

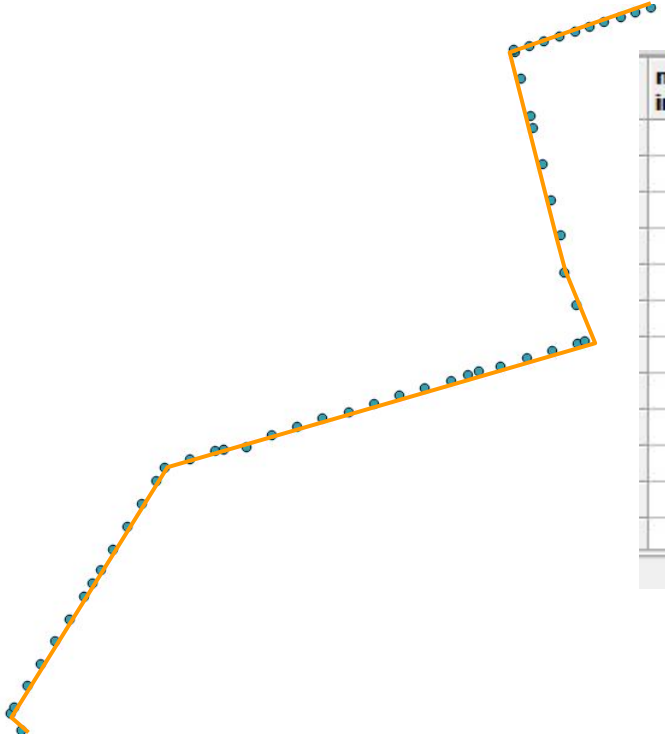
A PostgreSQL-PostGIS Extension for Mobility Data Management

# Mobility Data: Traditional Approach



| 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 |

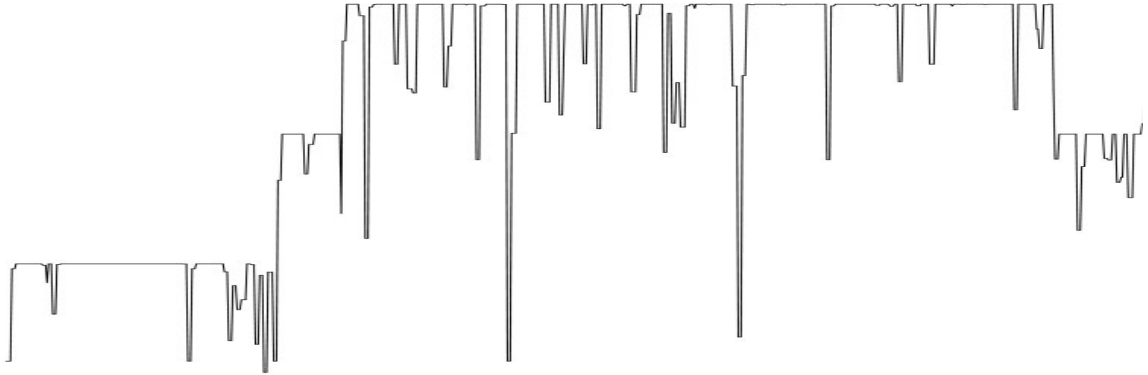
# Mobility Data: Trajectories



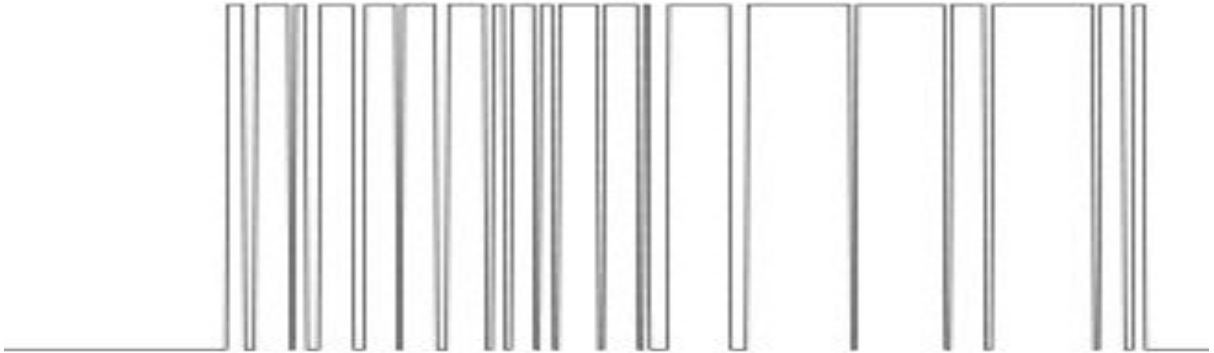
| moid integer | tripid integer | astext text |
|---|---|---|
| 6 | 163 | [POINT(2997192.88890412 5839689.91506735)@2007-05-28 06:00:00.001+00, POI |
| 6 | 165 | [POINT(2985654.50641456 5848965.14626724)@2007-05-28 16:09:12.824+00, POI |
| 8 | 235 | [POINT(3010311.09650771 5836055.09743228)@2007-05-28 07:19:01.864+00, POI |
| 8 | 237 | [POINT(2997958.79103681 5837131.44898043)@2007-05-28 16:05:50.982+00, POI |
| 8 | 241 | [POINT(2997958.79103681 5837131.44898043)@2007-05-29 17:11:03.19+00, POIN |
| 8 | 247 | [POINT(3010311.09650771 5836055.09743228)@2007-05-30 07:02:57.848+00, POI |
| 9 | 288 | [POINT(3001526.14852942 5837101.46991784)@2007-05-31 21:15:07.6+00, POINT |
| 9 | 290 | [POINT(3008321.78980041 5845720.9362808)@2007-05-31 22:47:38.444+00, POIN |
| 10 | 323 | [POINT(2993181.49144001 5853123.75533338)@2007-05-30 17:09:18.5+00, POINT |
| 10 | 325 | [POINT(2995709.23953211 5838172.58057013)@2007-05-31 07:01:19.697+00, POI |
| 13 | 422 | [POINT(3020510.76271993 5835681.48725136)@2007-05-28 06:32:00.131+00, POI |
| 13 | 424 | [POINT(2998220.90876918 5842741.02120682)@2007-05-28 17:21:02.64+00, POIN |

# Mobility Data: Temporal Types

tfloat: speed(Trip).



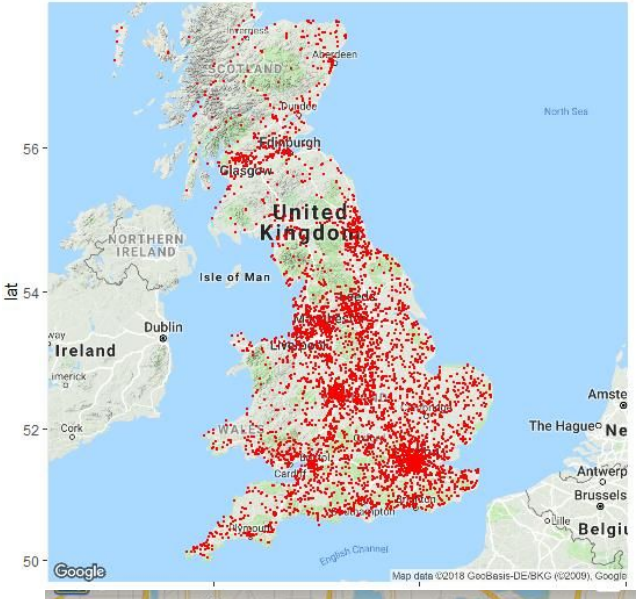
tbool: speed(Trip) > 90



# Mobility Data: Points

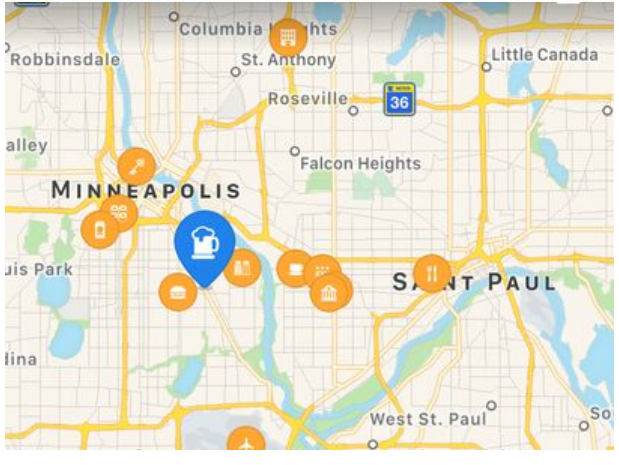
tgeogpoint(inst): UK road accidents 2012-14

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

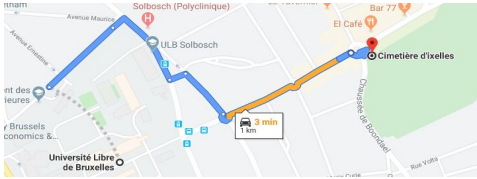



tgeogpoint(instants): foursquare check-ins

https://support.foursquare.com/



# MobilityDB: Architecture





| | | | |
|---|---|---|---|
| | | | |
| | | | |

MobilityDB

tgeompoint, tgeogpoint,
tint, tfloat, ttext, tbool

PostGIS

geometry, geography

PostgreSQL

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

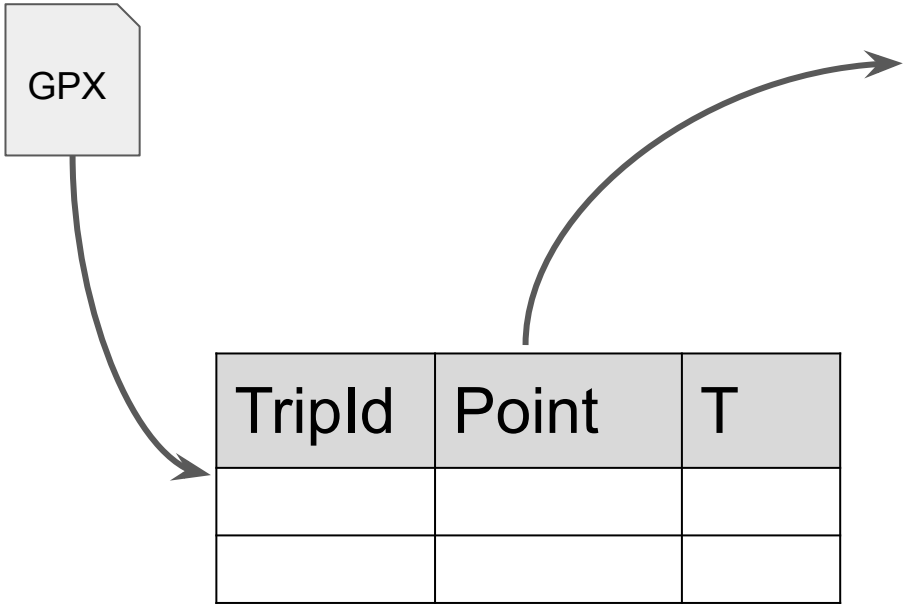
# MobilityDB: Querying


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

...

```
CREATE TABLE Trips AS
  SELECT CarId, TripId, tgeompointseq(
    array_agg(
      tgeompointinst( ST_Point(lon, lat), t) ORDER BY t
    )) AS Trip
FROM gpx
GROUP BY CarId, TripId
ORDER BY CarId, TripId
```

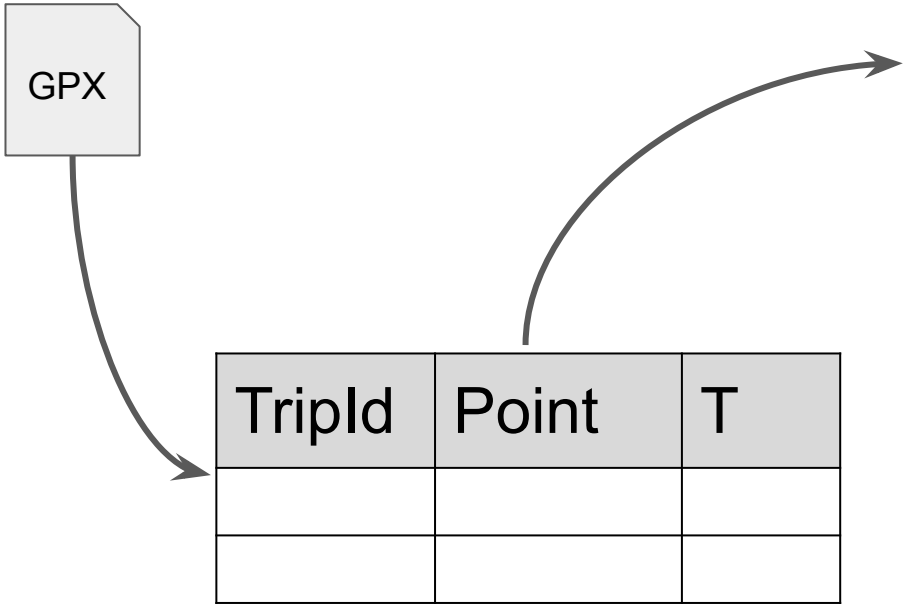
# Querying without MobilityDB




```
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
```

...

# Querying without MobilityDB

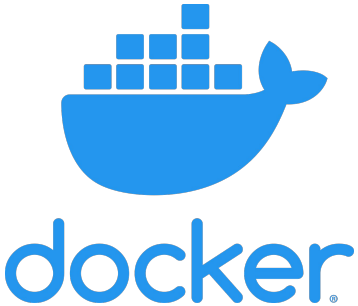


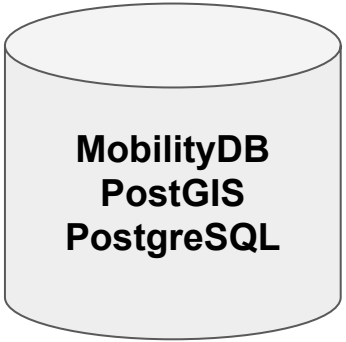

```
WITH TripSegs AS (
     SELECT TripId, Point AS P1, T AS T1,
          LAG (Point) OVER W AS P2,
          LAG (T) OVER W AS T2
     FROM TripPoints WINDOW W AS (
          PARTITION BY TripId ORDER BY T)
)
SELECT DISTINCT(TripId)
FROM TripSegs
WHERE ST_Distance(Point2, Point1) /
     (T2 - T1) >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

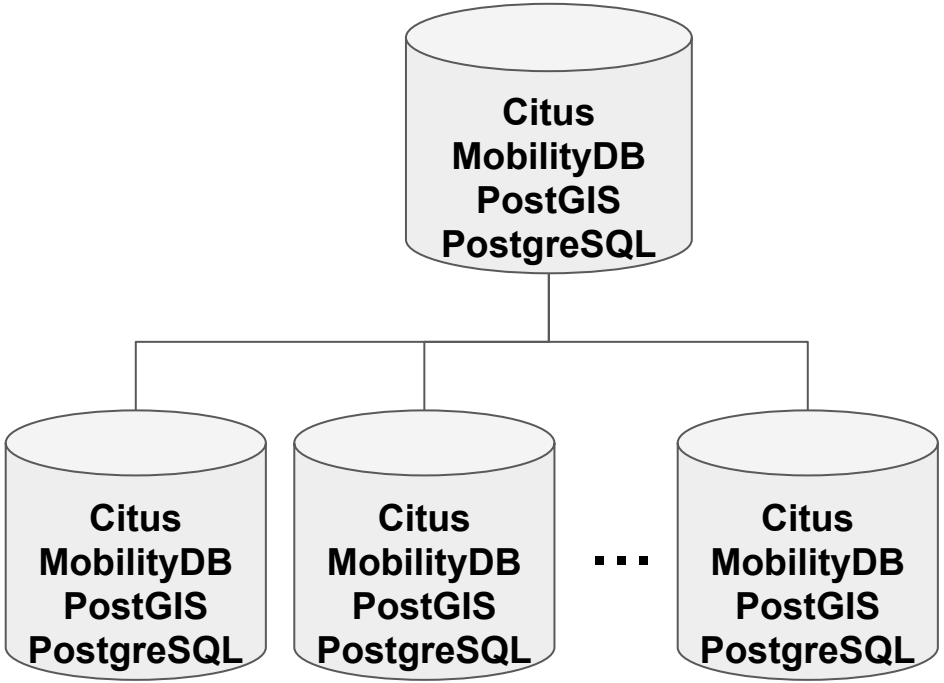
...

What if the trips contain temporal gaps ?
(e.g. GPS signal lost in tunnels)

# Alternative 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

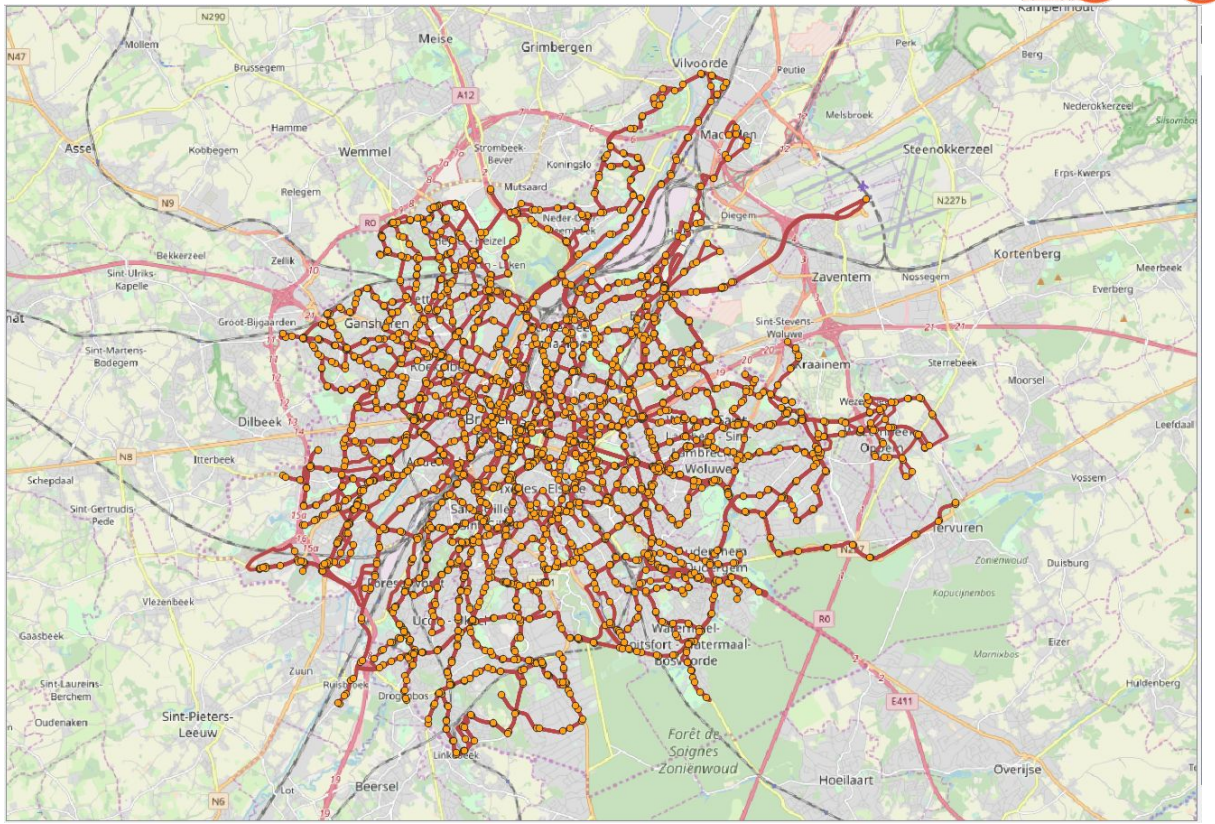
**Source:** STIB, Brussels

**Duration:** 28 days

7 Oct- 3 Nov 2019

**#Trips:** 445,187

**DB size:** 9 GB



# Quick Example: Spatial Projection

```
TABLE Bus( TripNo integer, LineNo 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( TripNo integer, LineNo 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( TripNo integer, LineNo 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 week,

FROM Bus

GROUP BY week;
```

# Quick Example: Temporal Aggregation

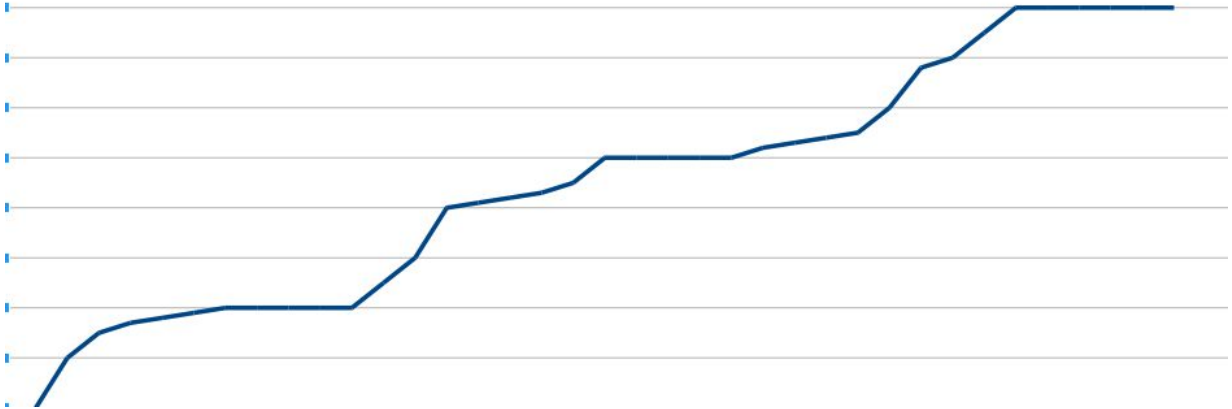
```
TABLE Bus( TripNo integer, LineNo integer, Route tgeompoint( Sequence, Point, 3812 ) );
```

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 week,

FROM Bus

GROUP BY week;
```



# Quick Example: Spatiotemporal Join

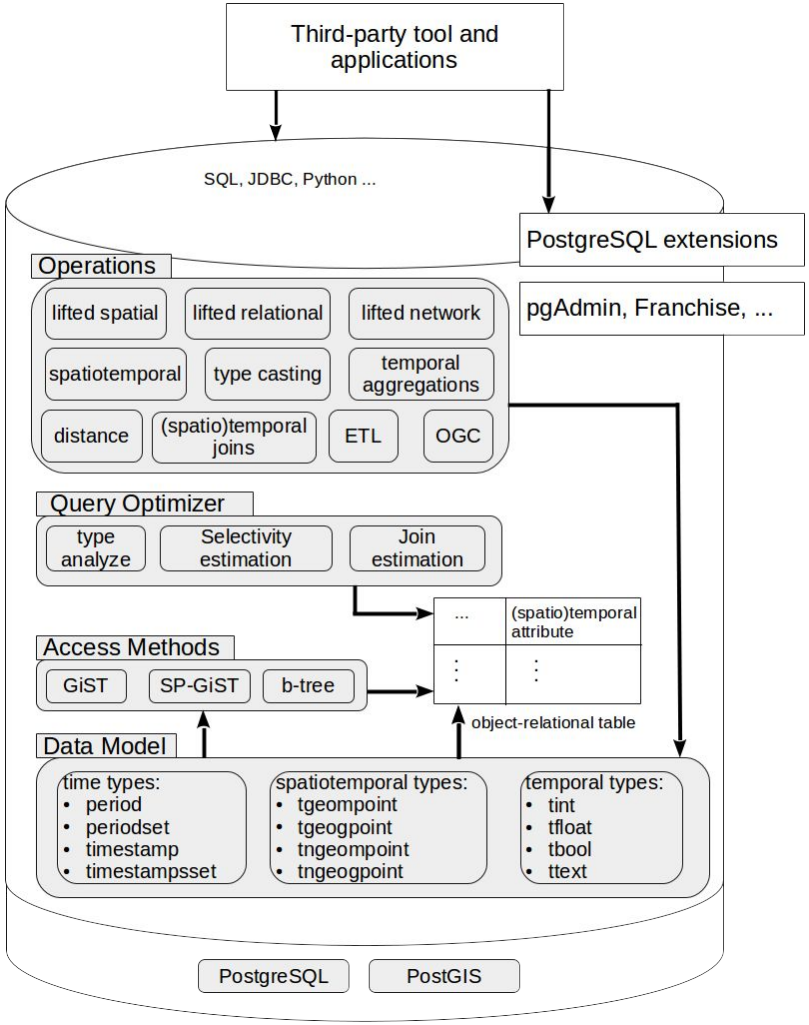
```
TABLE Bus( TripNo integer, LineNo integer, Route tgeompoint( Sequence, Point, 3812) );

TABLE Stops( StopNo integer, Name TEXT, Geo GEOMETRY(POINT, 3812) );
```

List all transit opportunities, that 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
```

# MobilityDB Features



# MobilityDB Ecosystem

<table>
<tr><td colspan="2">MobilityDB<br>MapMatch</td><td colspan="2">MobilityDB<br>Exchange</td><td colspan="2">MobilityDB<br>ETL</td><td colspan="2">MobilityDB<br>View</td></tr>
<tr><td>MobilityDB<br>Distributed</td><td colspan="2">MobilityDB<br>Network</td><td colspan="2">MobilityDB<br>Stream</td><td>MobilityDB<br>Python</td><td>MobilityDB<br>JDBC</td></tr>
<tr><td>Citus</td><td colspan="2">PgRouting</td><td colspan="2">PipelineDB</td><td>Psycopg<br>2.8</td><td>PostgreSQL<br>JDBC<br>42.2.6</td></tr>
<tr><td colspan="2">MobilityDB</td><td colspan="2">PostgreSQL 11<br>PostGIS 2.5</td><td colspan="2">Python 3.7</td><td colspan="2">Java 11</td></tr>
<tr><td colspan="8">Ubuntu 18.04.2 LTS</td></tr>
</table>

# 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

# MobilityDB on Github

GitHub - ULB-CoDE-WIT/Mobilit

github.com/ULB-CoDE-WIT/MobilityDB

build passing | coverage 96%

## MobilityDB

MobilityDB

MobilityDB is an open source software program that adds support for temporal and spatio-temporal objects to the PostgreSQL object-relational database and its spatial extension PostGIS. MobilityDB follows the Moving Features specification from the Open Geospatial Consortium (OGC).

Technically, MobilityDB is implemented as a PostgreSQL external extension.

MobilityDB is developed by the Computer & Decision Engineering Department of the Université Libre de Bruxelles (ULB) under the direction of Prof. Esteban Zimányi.

### Features

- Time types `Period`, `PeriodSet`, and `TimestampSet` which, in addition of the the `TimestampTz` type provided by PostgreSQL, are used to represent time spans.

# Thanks for listening !

Questions ?