Enabling Mobility Data in Multiple Computing Environments

Esteban Zimányi
Université libre de Bruxelles, Belgium

Invited talk at Data-driven Smart Cities (DASC) Workshop @ ICDE 2023
Anaheim, CA, USA, March 2023

Types of Mobility Data: Time-Varying Attributes

Position: tgeompoint, tgeogpoint

Speed: tfloat

Speed > 50: tbool

Gear: tint

Road type (e.g., 'highway', 'primary' ...): ttext

Connected Car Data

timestamp	moving_state	trip_id	sp1	ssa	driver_gender	driver_age	speed_label
10:10.1	1	c521b0b0	13.64	309	F	32	city
10:10.2	1	c521b0b0	14.2	288	F	32	city
10:10.3	1	c521b0b0	14.56	256.5	F	32	city
10:10.4	1	c521b0b0	14.87		F	32	city
10:10.5	1	c521b0b0	15.19		F	32	city
10:10.6	1	c521b0b0	15.68		F	32	city
10:10.7	1	c521b0b0	16.15		F	32	city
10:10.8	1	c521b0b0	16.7	93	F	32	city
10:10.9	1	c521b0b0	16.99	69	F	32	city
10:11.0	1	c521b0b0	17.55	49.5	F	32	city
10:11.1	1	c521b0b0	18.14		F	32	city
10:11.2	1	c521b0b0	18.7		F	32	city
10:11.3	1	c521b0b0	18.86		F	32	city
10:11.4	1	c521b0b0	19.58		F	32	city
10:11.5	1	c521b0b0	20.21	-25.5	F	32	city
10:11.6	1	c521b0b0	20.63	-36	F	32	city
10:11.7	1	c521b0b0	21.02	-37.5	F	32	city

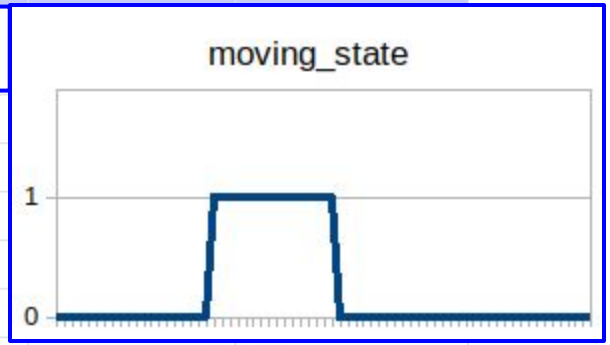
Connected Car Data

timestamp	moving_state	trip_id	sp1	ssa	driver_gender	driver_age	speed_label
10:10.1	1	:521b0b0	13.64	309	F	32	city
10:10.2	1	:521b0b0	14.2	288	F	32	city
10:10.3	1	:521b0b0	14.56	256.5	F	32	city
10:10.4	1	:521b0b0	14.87		F	32	city
10:10.5	1	:521b0b0	15.19		F	32	city
10:10.6	1	:521b0b0	15.68		F	32	city
10:10.7	1	:521b0b0	16.15		F	32	city
10:10.8	1	:521b0b0	16.7	93	F	32	city
10:10.9	1	:521b0b0	16.99	69	F	32	city
10:11.0	1	:521b0b0	17.55	49.5	F	32	city
10:11.1	1	:521b0b0	18.14		F	32	city
10:11.2	1	:521b0b0	18.7		F	32	city
10:11.3	1	:521b0b0	18.86		F	32	city
10:11.4	1	:521b0b0	19.58		F	32	city
10:11.5	1	:521b0b0	20.21	-25.5	F	32	city
10:11.6	1	:521b0b0	20.63	-36	F	32	city
10:11.7	1	:521b0b0	21.02	-37.5	F	32	city

Connected Car Data

timestamp	moving_state	trip_id	sp1	s				
10:10.1	1	:521b0b0	13.64					
10:10.2	1	:521b0b0	14.2	288	F			
10:10.3	1	:521b0b0	14.56	256.5	F			
10:10.4	1	:521b0b0	14.87		F			
10:10.5	1	:521b0b0	15.19		F			
10:10.6	1	:521b0b0	15.68		F			
10:10.7	1	:521b0b0	16.15		F		32	city
10:10.8	1	:521b0b0	16.7	93	F		32	city
10:10.9	1	:521b0b0	16.99	69	F		32	city
10:11.0	1	:521b0b0	17.55	49.5	F		32	city
10:11.1	1	:521b0b0	18.14		F		32	city
10:11.2	1	:521b0b0	18.7		F		32	city
10:11.3	1	:521b0b0	18.86		F		32	city
10:11.4	1	:521b0b0	19.58		F		32	city
10:11.5	1	:521b0b0	20.21	-25.5	F		32	city
10:11.6	1	:521b0b0	20.63	-36	F		32	city
10:11.7	1	:521b0b0	21.02	-37.5	F		32	city

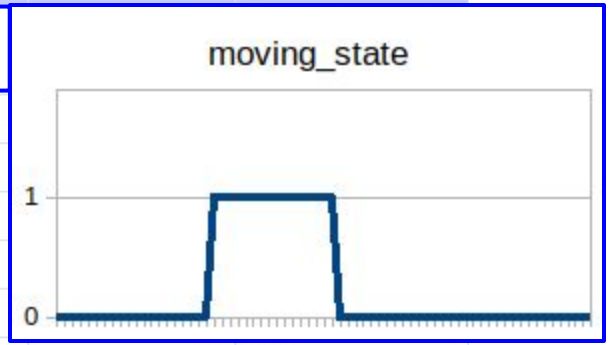
Temporal Boolean



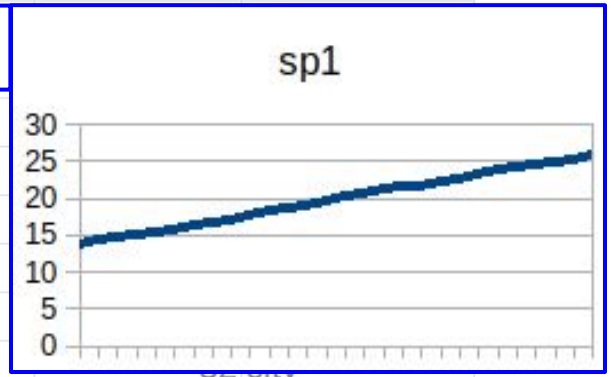
Connected Car Data

timestamp	moving_state	trip_id	sp1	s				
10:10.1	1	c521b0b0	13.64					
10:10.2	1	c521b0b0	14.2	288	F			
10:10.3	1	c521b0b0	14.56	256.5	F			
10:10.4	1	c521b0b0	14.87		F			
10:10.5	1	c521b0b0	15.19		F			
10:10.6	1	c521b0b0	15.68		F			
10:10.7	1	c521b0b0	16.15		F		32	city
10:10.8	1	c521b0b0	16.7					
10:10.9	1	c521b0b0	16.99					
10:11.0	1	c521b0b0	17.55	49.5	F			
10:11.1	1	c521b0b0	18.14		F			
10:11.2	1	c521b0b0	18.7		F			
10:11.3	1	c521b0b0	18.86		F			
10:11.4	1	c521b0b0	19.58		F			
10:11.5	1	c521b0b0	20.21	-25.5	F			
10:11.6	1	c521b0b0	20.63	-36	F		32	city
10:11.7	1	c521b0b0	21.02	-37.5	F		32	city

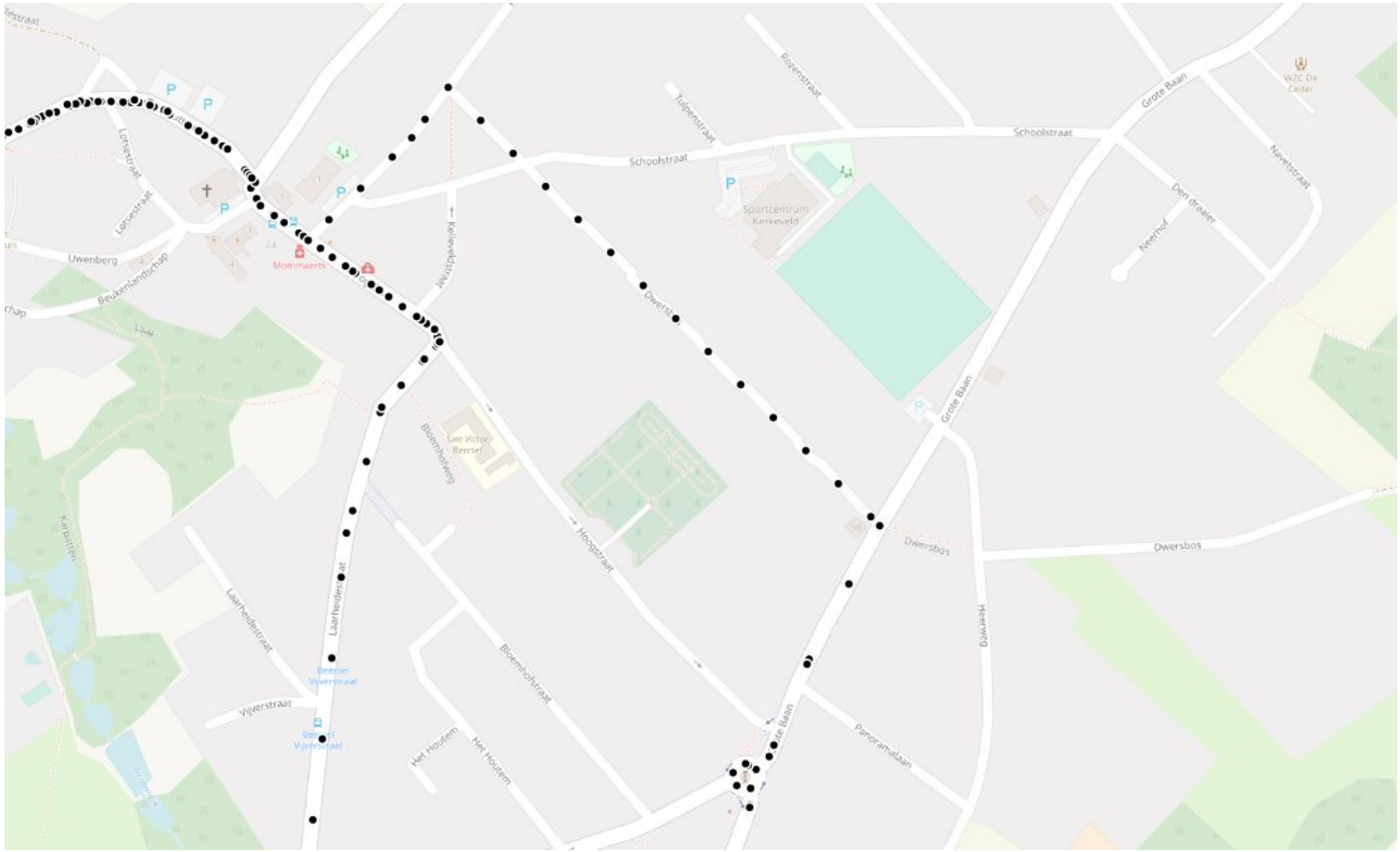
Temporal Boolean



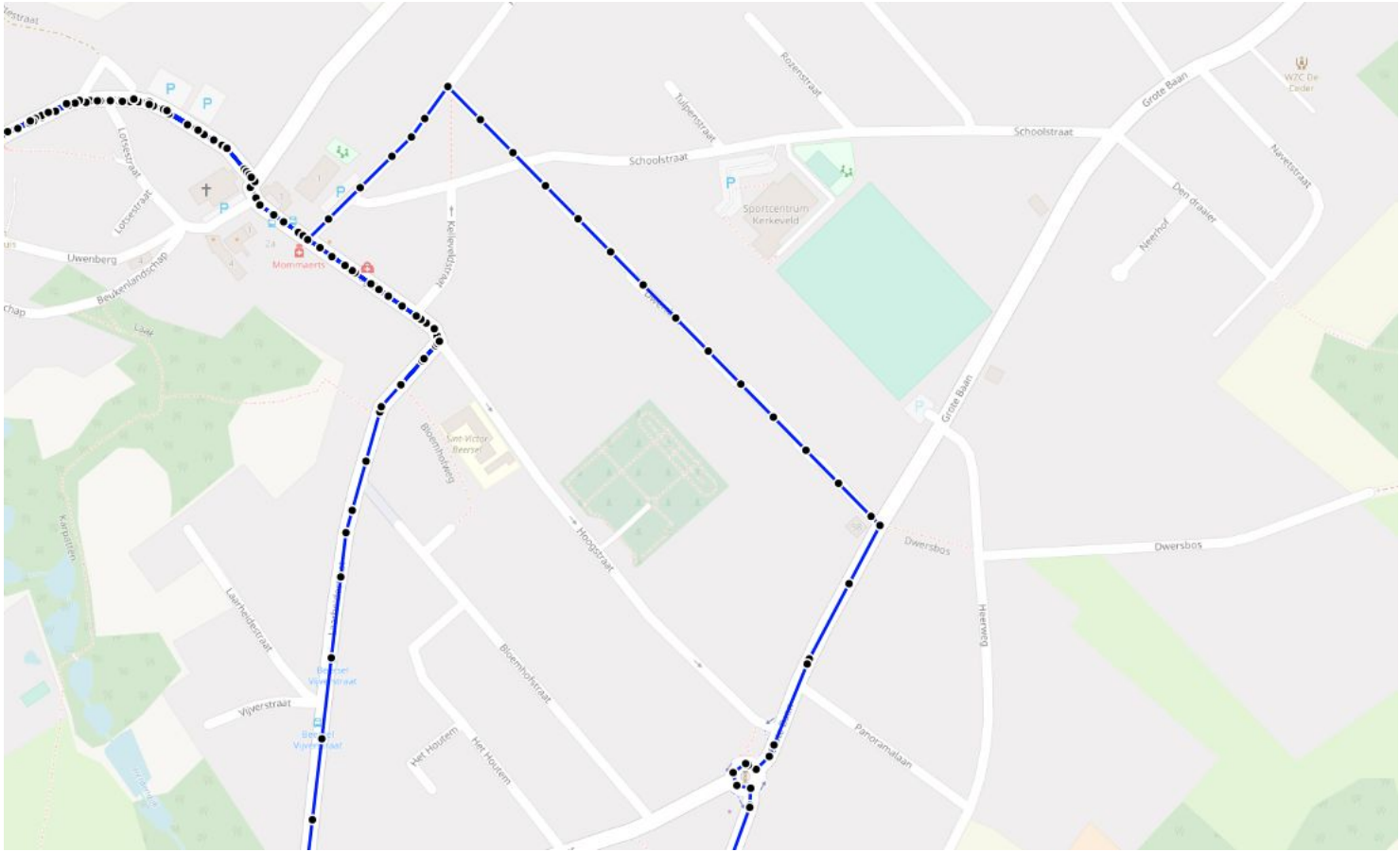
Temporal float



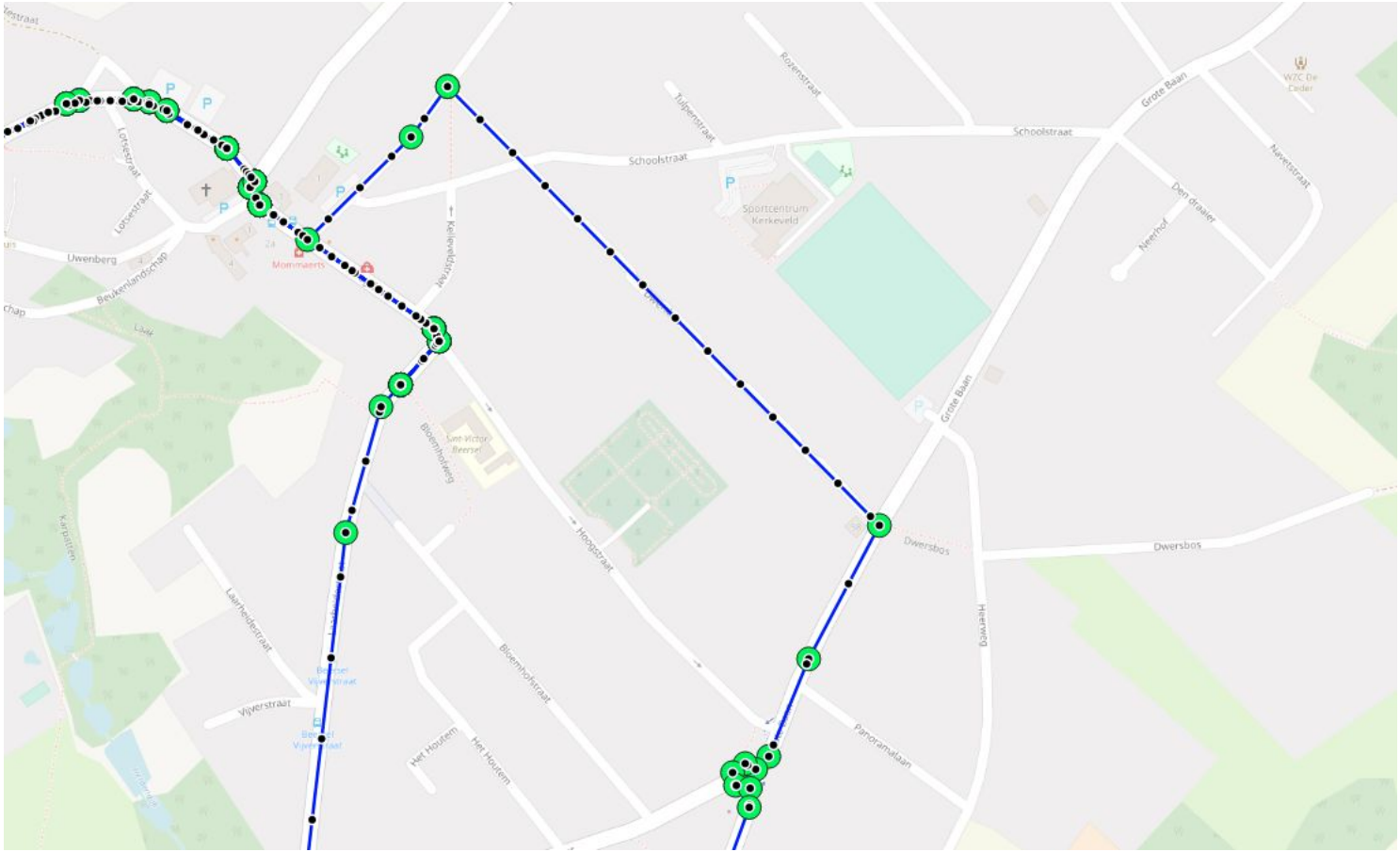
Mobility Data: Traditional Data Format



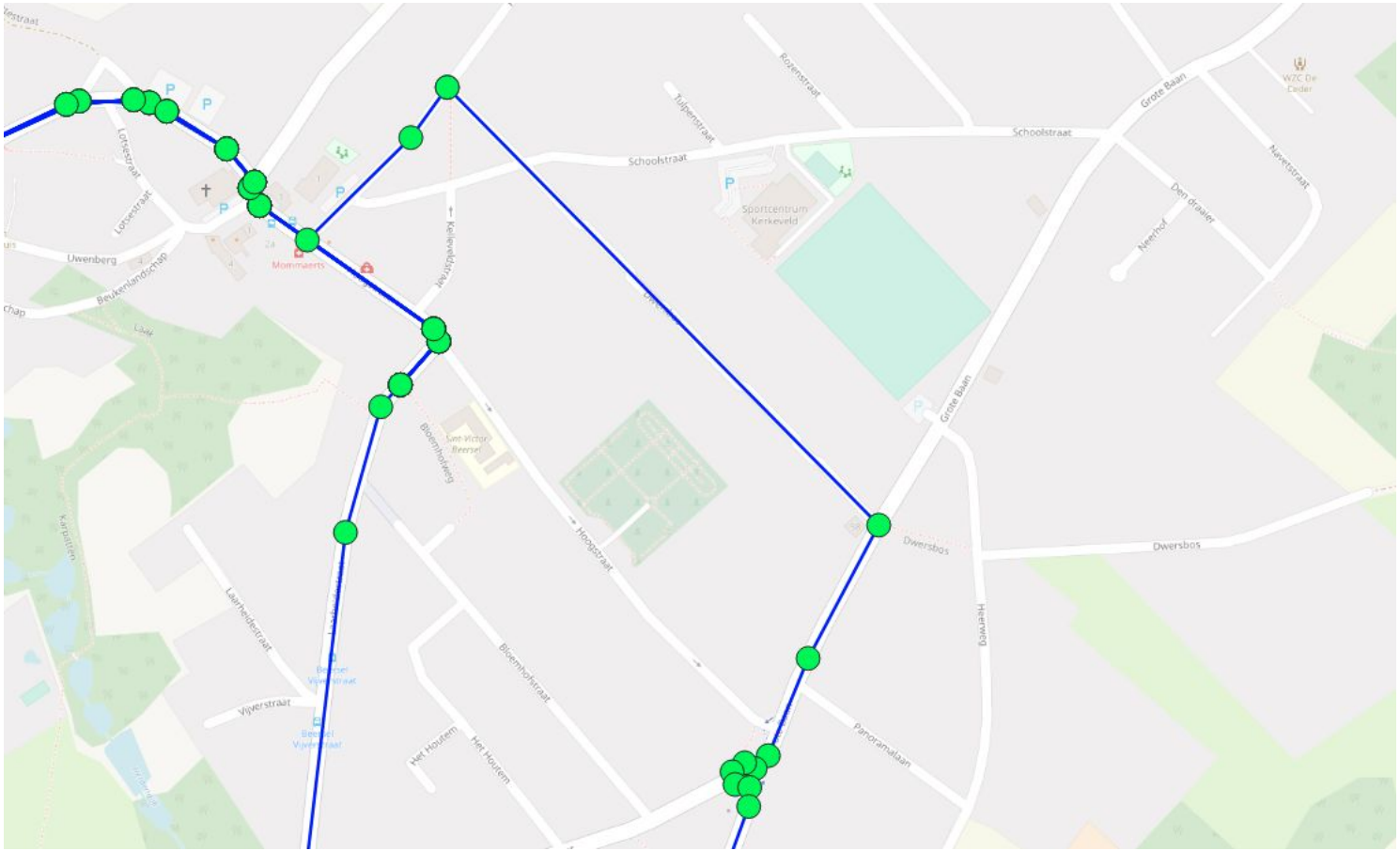
MobilityDB Compact Format



MobilityDB Compact Format



MobilityDB Compact Format



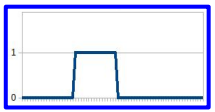
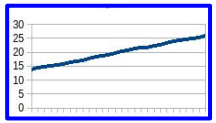
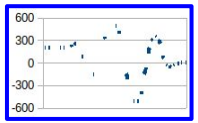
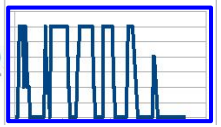
Migrating Connected Car Data into MobilityDB

Compression rate 450%

timestamp	moving_state	trip_id	sp1	ssa	driver_gender	driver_age	speed_label
10:10.1	1	c521b0b0	13.64	309	F	32	city
10:10.2	1	c521b0b0	14.2	288	F	32	city
10:10.3							ity
10:10.4							ity
10:10.5							ity
10:10.6							ity
10:10.7	1	c521b0b0	16.15		F	32	city
10:10.8	1	c521b0b0	16.7	93	F	32	city
10:10.9	1	c521b0b0	16.99	69	F	32	city
10:11.0	1	c521b0b0	17.55	49.5	F	32	city

16,000 records per trip

100 trip_id
1.6 million rows
165 MB

moving_state	trip_id	sp1	ssa	driver_gender	driver_age	speed label
	c521b0b0			F	32	

100 trip_id
100 rows
37 MB

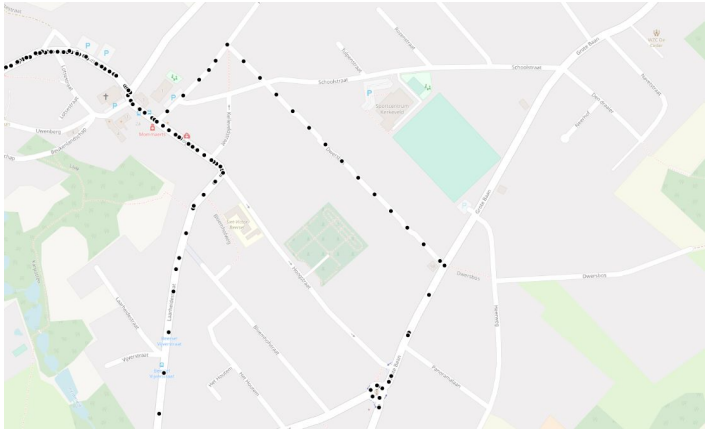
MobilityDB Compact Format

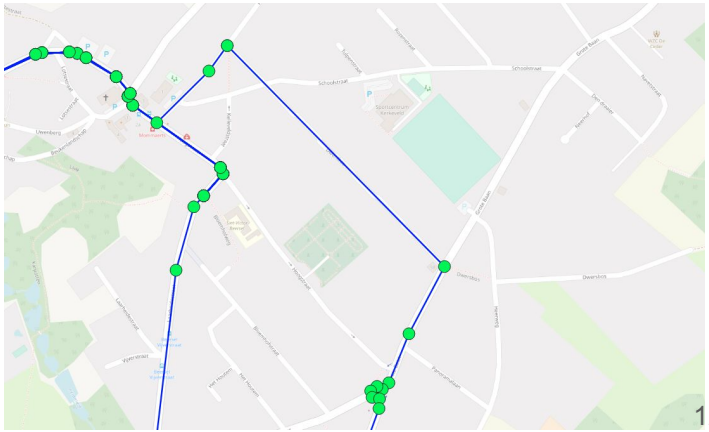
Improvement over existing tools

- Compact data storage with lossless high compression rate
- Query speedup
- Compatible with PostgreSQL ecosystem

Novel benefits

- Interpolation
- Native mobility data types
- Rich mobility analytics API
- Significant reduction in development time





Rich Mobility In-Database Analytics

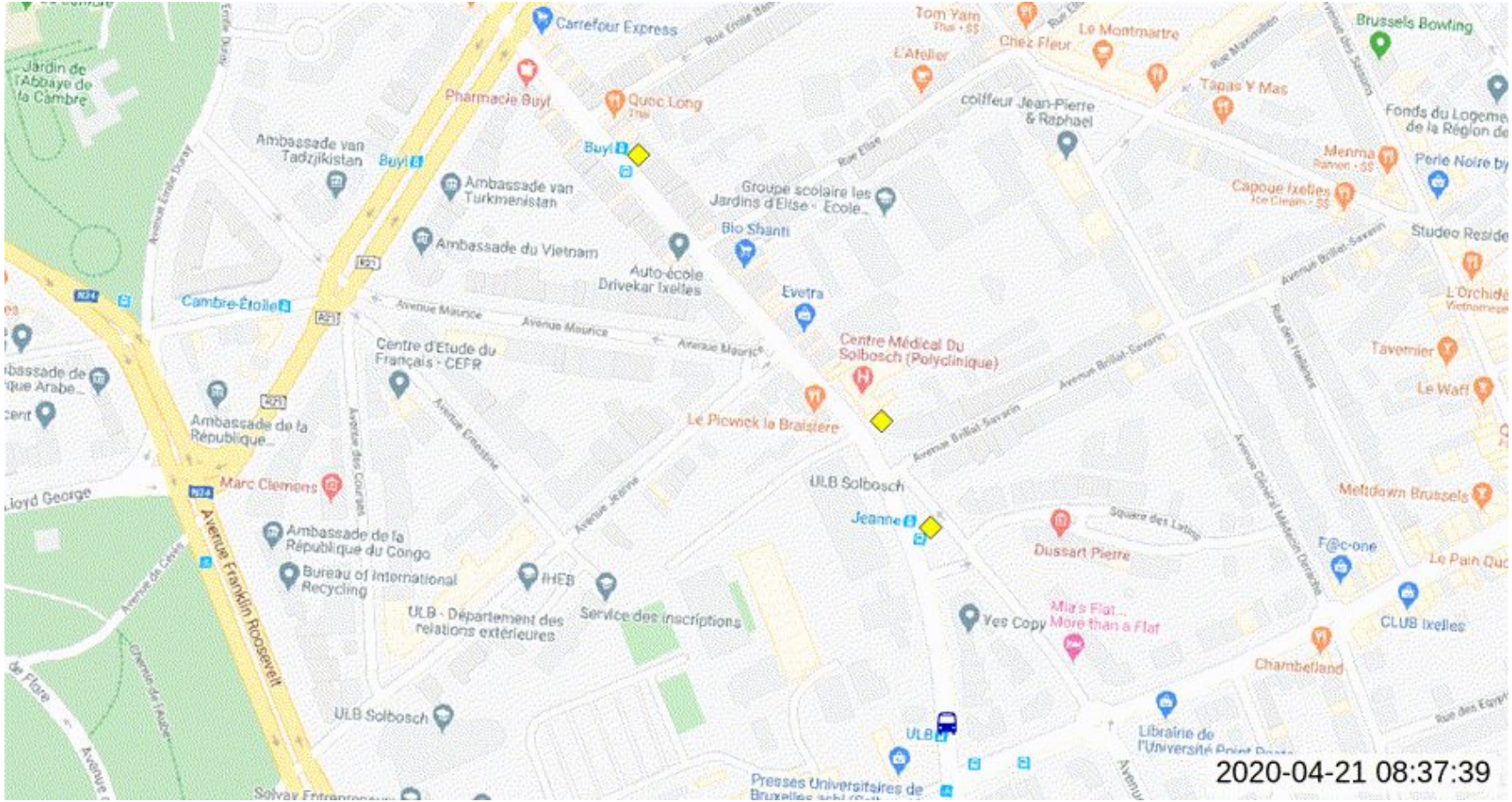
```
SELECT MIN(fix_altitude), MAX(fix_altitude),
  AVG(fix altitude)
FROM trips_feb;
```

```
+---+-----+-----------------+
|min|max  |avg              |
+---+-----+-----------------+
|0  |175.8|39.23260869565219|
+---+-----+-----------------+
```

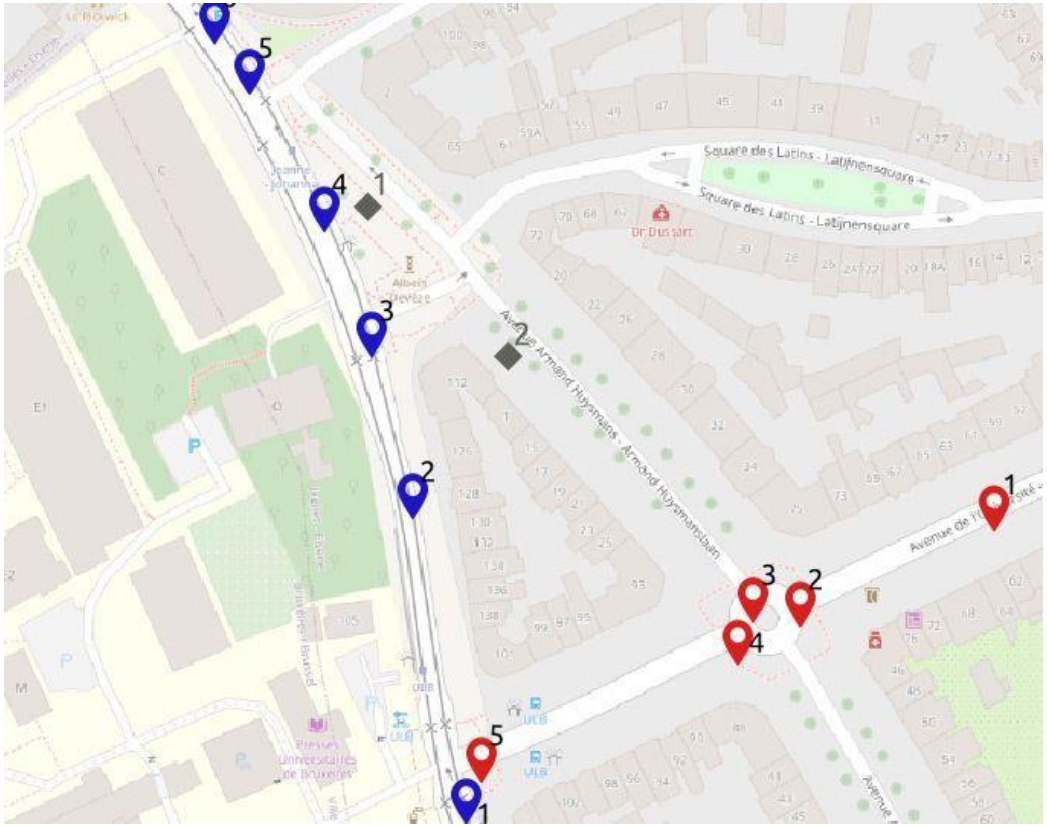
```
SELECT trip id, twAvg(speed(transform(
  setSRID(trip, 4326), 31370)) - fix_speed) * 3.6 AS SpeedDiff
FROM trips_feb
```

```
+--------+------------------+
|trip id |speeddiff         |
+--------+------------------+
|2cd61e90|-5.728162725492423 |
|516c534e|5.342059044492264  |
|6ac04b1c|-15.430977498010625|
|6c24f0a7|11.738490603453421 |
|6cb55b37|-14.722568277901456|
+--------+------------------+
```

Spatiotemporal Proximity: Smart Advertising



Spatiotemporal Proximity: Smart Advertising



https://techcommunity.microsoft.com/t5/azure-database-for-postgresql/analyzing-gps-trajectories-at-scale-with-postgres-mobilitydb-amp/ba-p/1859278

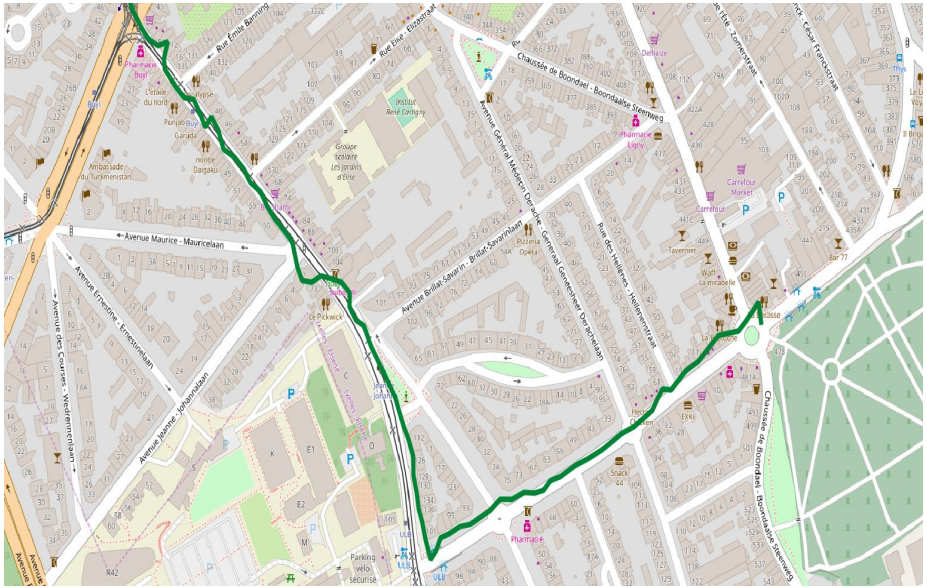
```
WITH pointPair AS (
  SELECT tripID, pointID AS p1, t AS t1, geom AS geom1,
    lead(pointID, 1) OVER (PARTITION BY tripID ORDER BY pointID) AS p2,
    lead(t, 1) OVER (PARTITION BY tripID ORDER BY pointID) AS t2,
    lead(geom, 1) OVER (PARTITION BY tripID ORDER BY pointID) AS geom2
  FROM gpsPoint
), segment AS (
  SELECT tripID, p1, p2, t1, t2, st_makeline(geom1, geom2) AS geom
  FROM pointPair WHERE p2 IS NOT NULL
), approach AS (
  SELECT tripID, p1, p2, t1, t2, a.geom, st_intersection(
    a.geom, st_exteriorRing(st_buffer(b.geom, 30))) AS visibilityTogglePoint
  FROM segment a, billboard b WHERE st_dwithin(a.geom, b.geom, 30) )
SELECT tripID, p1, p2, t1, t2, geom, visibilityTogglePoint,
  (st_lineLocatePoint(geom, visibilityTogglePoint) * (t2 - t1)) + t1 visibilityToggleTime
FROM approach;
```

MobilityDB: Connecting the Dots

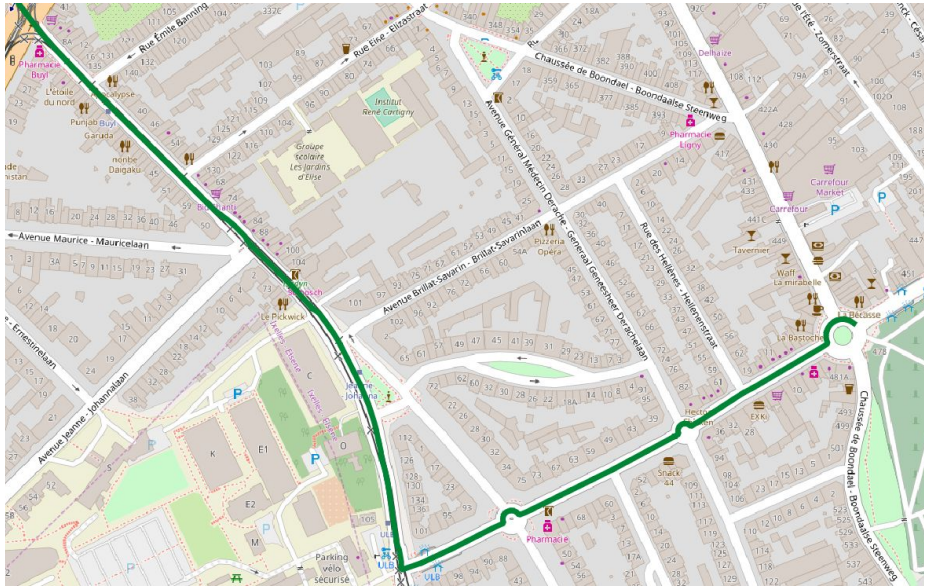

```
SELECT tripID, pointID, billboardID
FROM gpsPoint a, billboard b
WHERE st_dwithin(a.geom, b.geom, 30);
```

Spatiotemporal Proximity: Marine Data

Map Matching: Continuous Trajectories

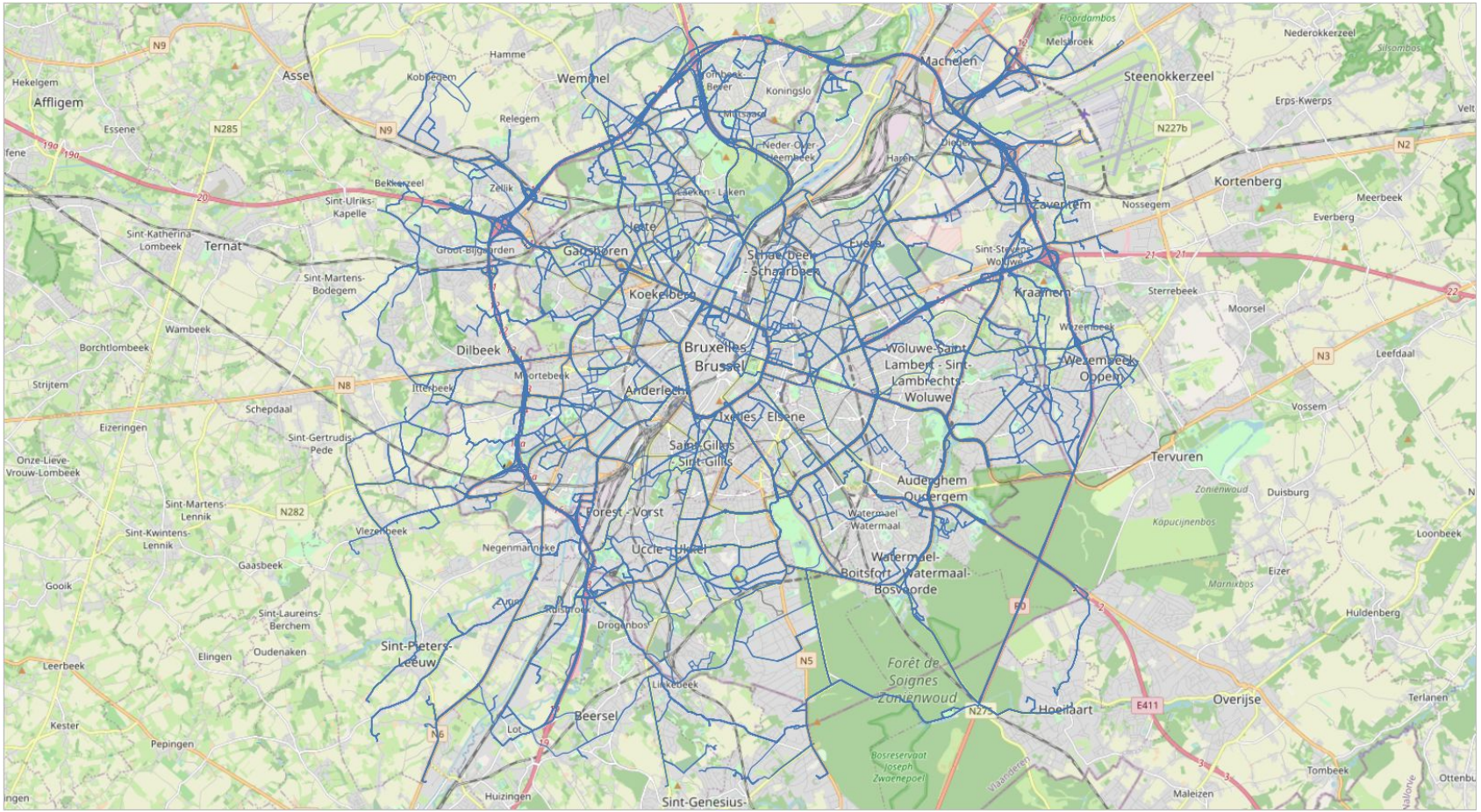


Original observations

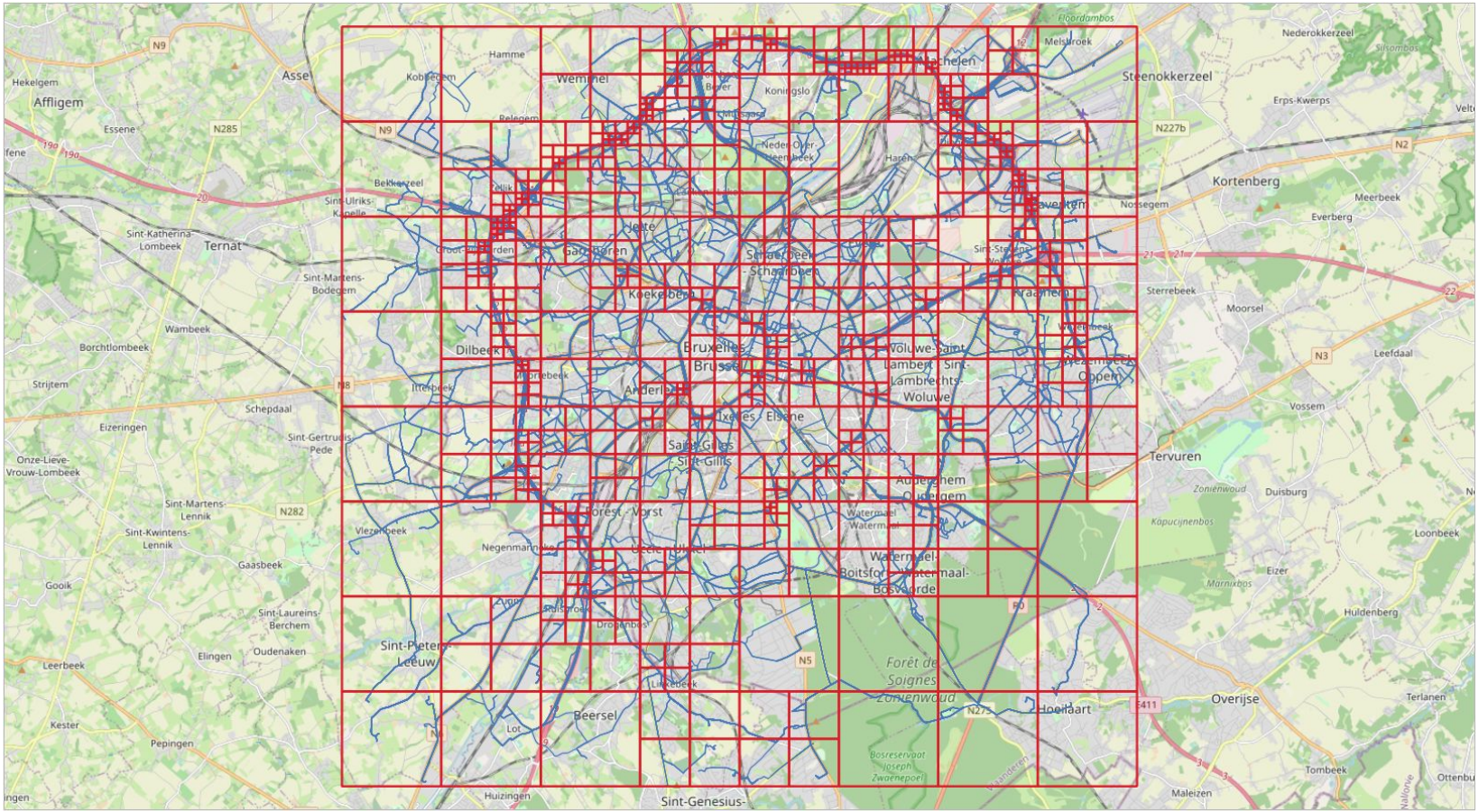


Map-matched observations

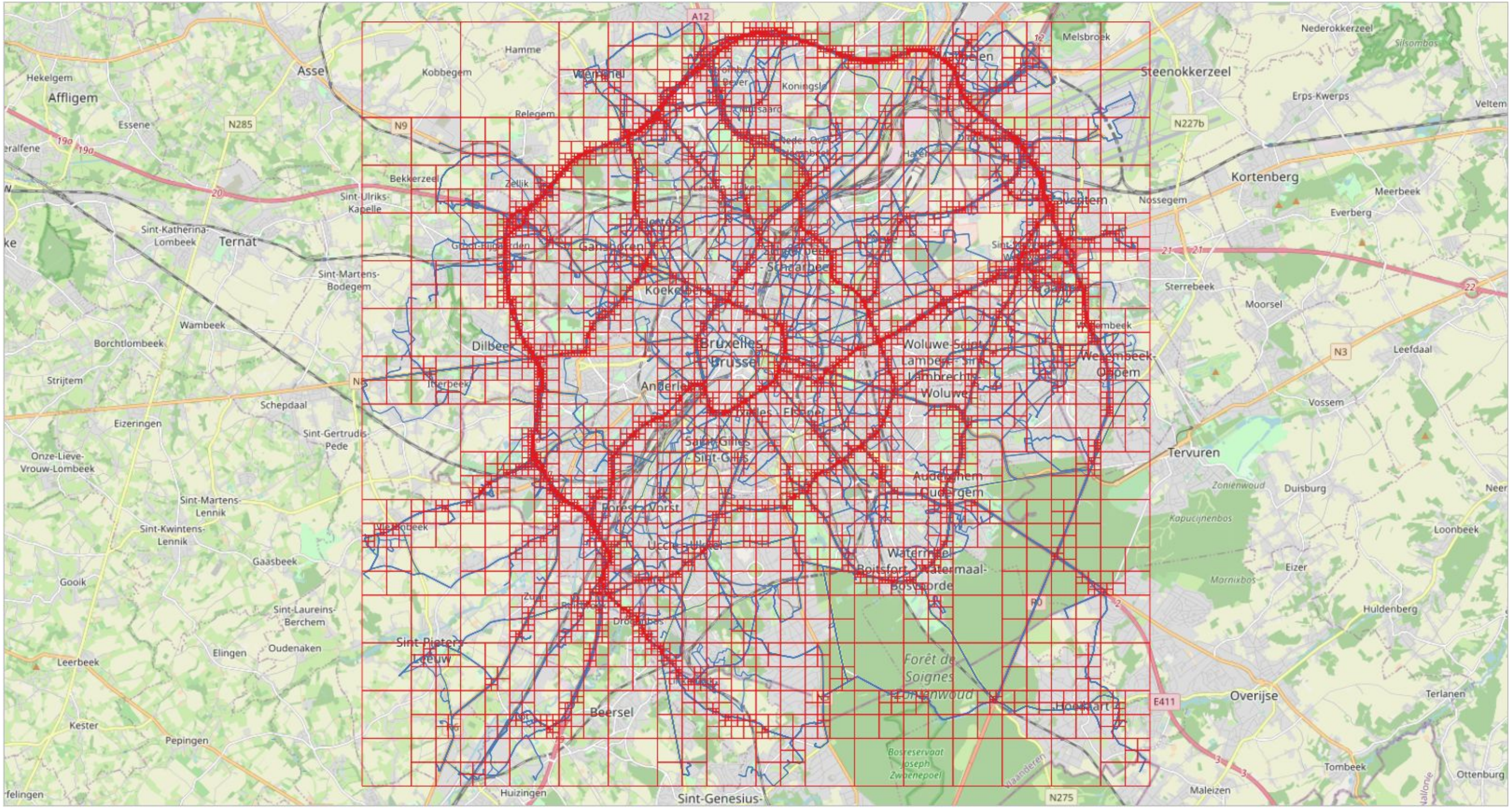
BerlinMOD Data Generator



Multiresolution Grid: 10K instants per cell



Multiresolution Grid: 1K instants per cell



MobilityDB: Open Source based on OGC Standards

README.md

Main Build passing coverage 97%

code quality A chat on gitter

MobilityDB

An open source geospatial trajectory data management & analysis platform

MobilityDB

https://github.com/MobilityDB/MobilityDB

OSGeo

Home » Projects » MobilityDB

MobilityDB

An open source geospatial trajectory data management & analysis platform

Back to projects

MobilityDB

https://www.osgeo.org/projects/mobilitydb/

Open Geospatial Consortium

Menu

Moving Features SWG

Chair(s):
Ishimaru, Nobuhiro (Ishimaru, Nobuhiro) - Co-Chair,
Kim, Kyoung-Sook (National Institute of Advanced Industrial Science & Technology (AIST)) - Co-Chair,
SAKR, Mahmoud (Université Libre de Bruxelles (ULB)) - Co-Chair

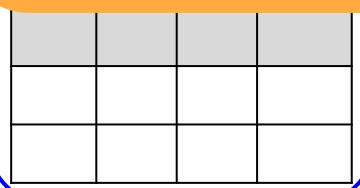
Group Charter:
Download Charter document

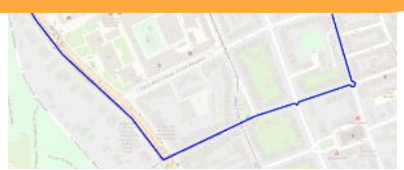
Group Description:
Moving Features
With the development of communication and positioning technology such as Global Navigation Satellite System (GNSS), Wi-Fi, and Beacon, collecting movement data for moving features,

https://www.ogc.org/projects/groups/movfeatswg

	PostgreSQL	PostGIS	MobilityDB
SQL Optimization	Scalar statistics & selectivity estimation	Grid-based statistics	Spatial grid + period bound histograms
Indexes	B-tree, hash, GiST, SP-GiST, GIN, BRIN	GiST, SP-GiST, BRIN	GiST, SP-GiST
Operations	Comparison, transformation, casting, ...etc	topological, CRS, properties, overlay, ...etc.	trajectory, temporal properties, lifted predicates, aggregations
Type System	numeric, character, date/time, bool, xml, json	Geometry, geography	tgeompoint, tgeogpoint, tint, tfloat, ttext, tbool

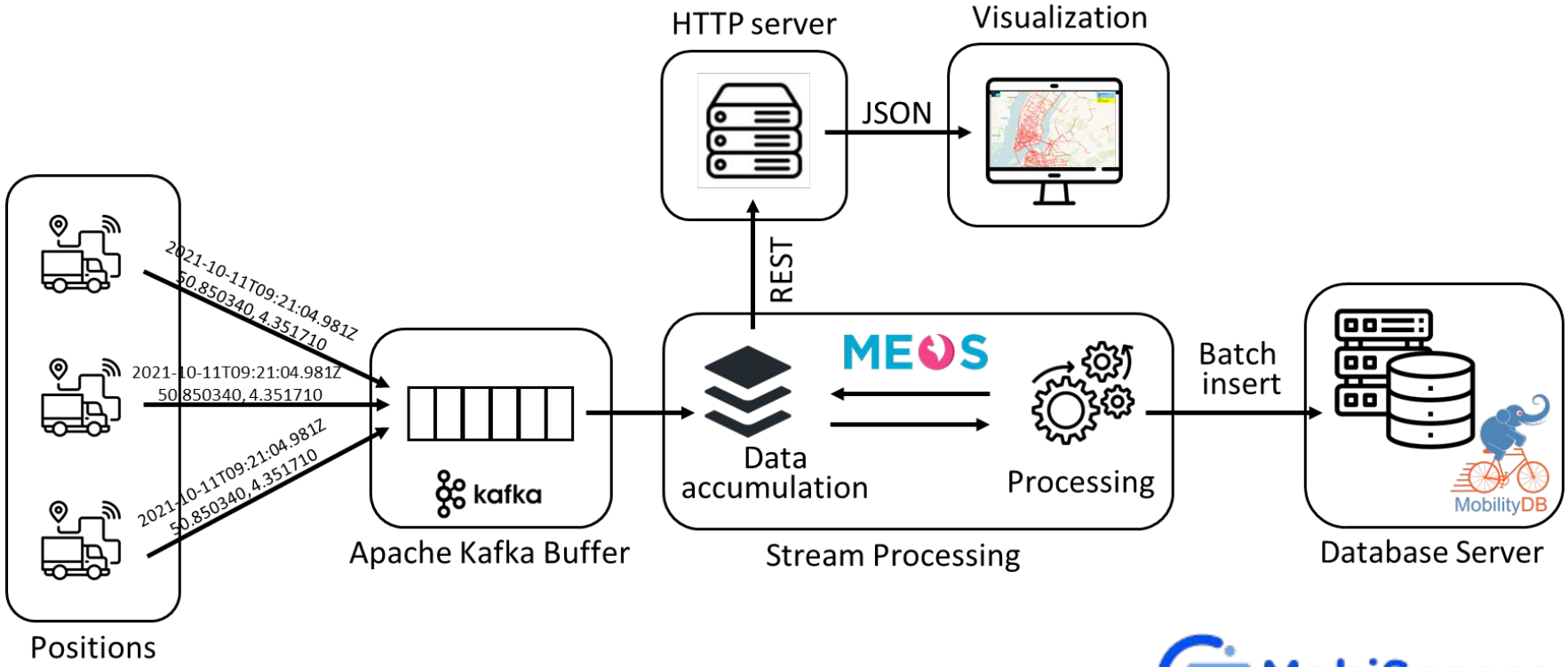
MEOS





https://mobilitydb.com/

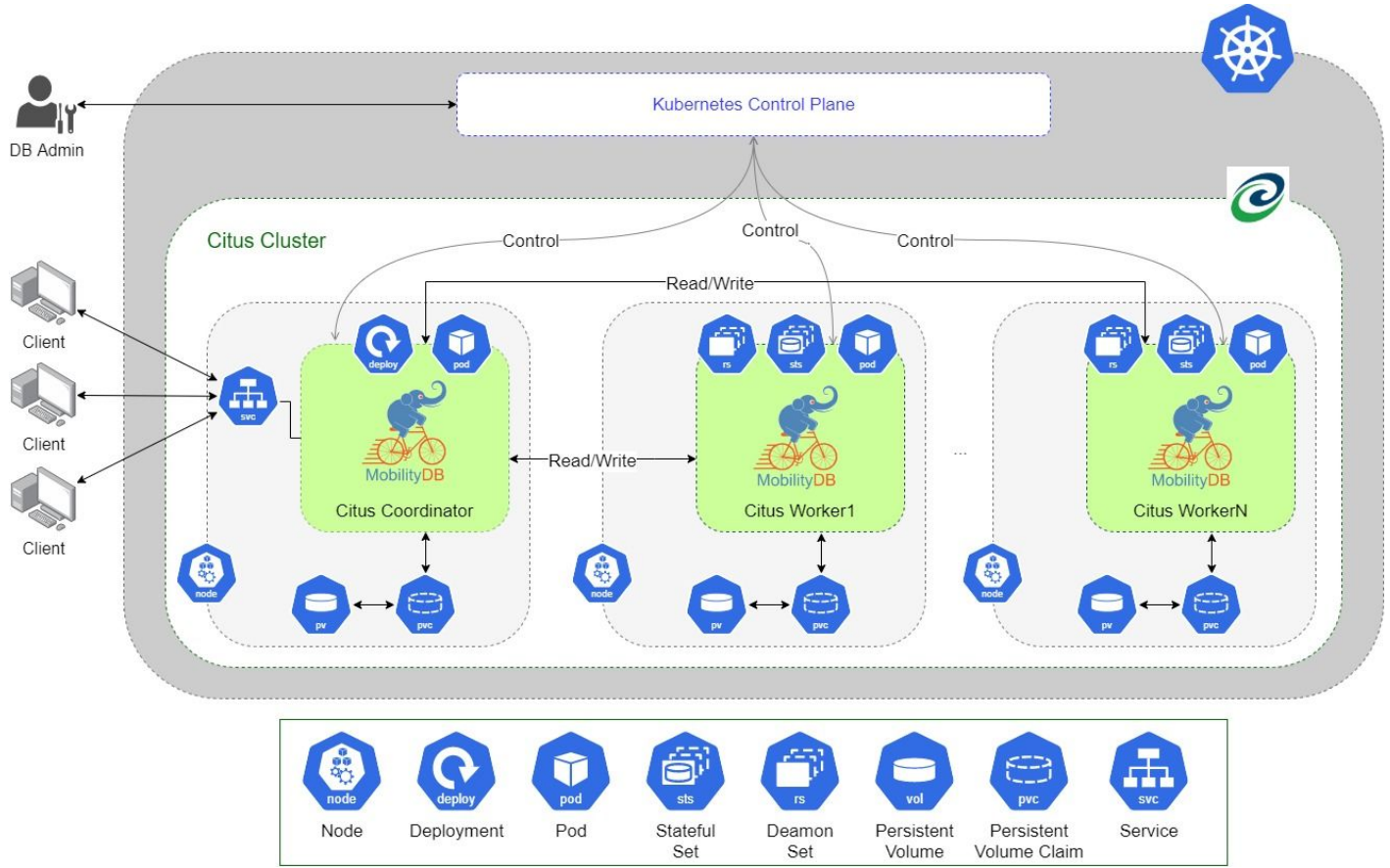
Mobility Data in Stream and Edge Processing



https://mobispaces.eu/

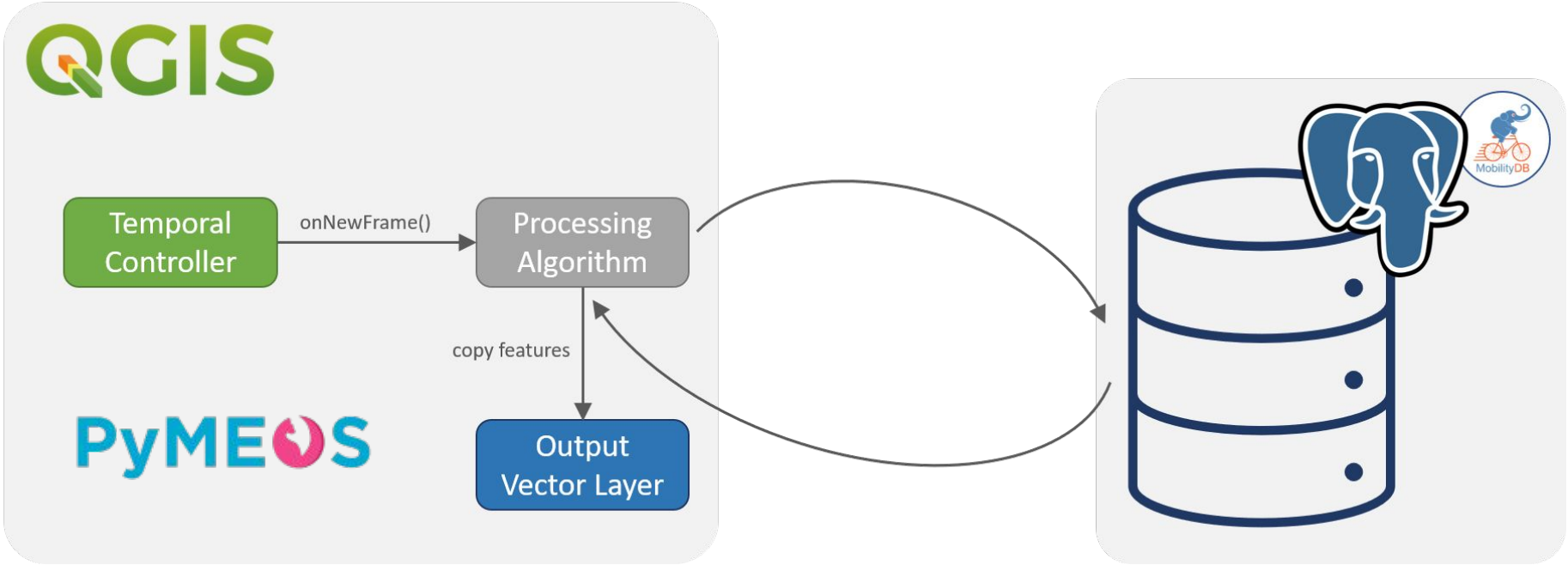
MobiSpaces
new data spaces for green mobility

Mobility Data in the Cloud



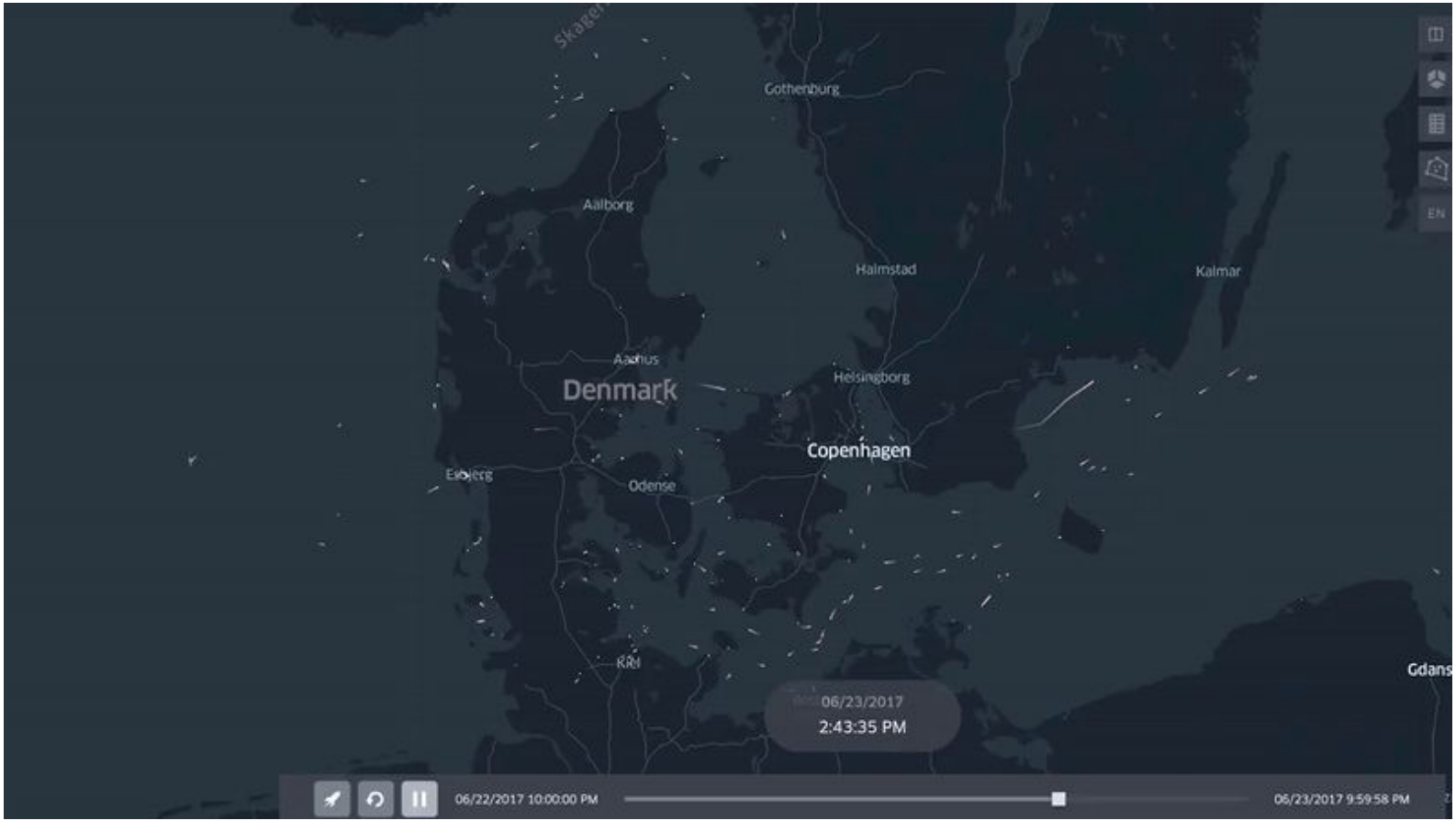
https://github.com/MobilityDB/MobilityDB-Azure

Visualizing Mobility Data in QGIS



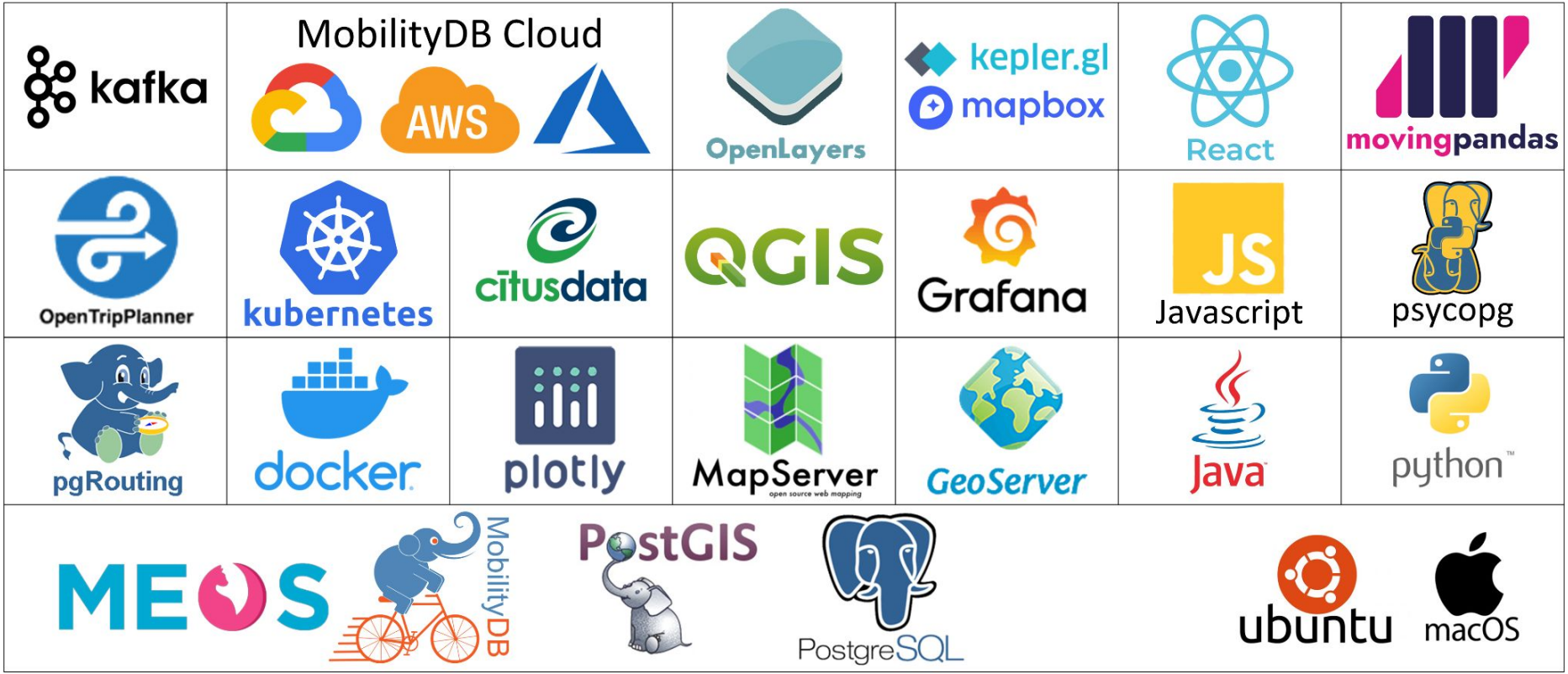
https://github.com/MobilityDB/MobilityDB-QGIS

Visualizing Mobility Data in Deck.gl



https://github.com/MobilityDB/MobilityDB-Deck

MobilityDB Ecosystem



MEOS

MEOS (Mobility Engine, Open Source) is a C library and its associated API for manipulating temporal and spatiotemporal data. It is the core component of MobilityDB, an open source geospatial trajectory data management & analysis platform built on top of PostgreSQL and PostGIS.

MEOS extends the ISO 19141:2008 standard (Geographic information — Schema for moving features) for representing the change of non-spatial attributes of features. It also takes into account the fact that when collecting mobility data it is necessary to represent "temporal gaps", that is, when for some period of time no observations were collected due, for instance, to signal loss.

MEOS is heavily inspired by a similar library called GEOS (Geometry Engine, Open Source) — hence the name. A first version of the MEOS library written in C++ has been proposed by Krishna Chaitanya Bommakanti. However, due to the fact that MEOS codebase is actually a subset of MobilityDB codebase, which is written in C and in SQL, the current version of the library allows us to evolve both programming environments simultaneously.

MEOS aims to be the base library on which other projects can be built. For example, the following projects are built on top of MEOS:

- PyMEOS is a Python binding to MEOS using CFFI
- MobilityDB is a PostgreSQL extension that enables storing and manipulating the temporal types provided by MEOS.

Other projects can built on top of MEOS, for example, Java or C# drivers for MEOS or implementing MEOS on other DBMSs such as MySQL.

GEOS

GEOS is a C/C++ library for computational geometry with a focus on algorithms used in geographic information systems (GIS) software. It implements the OGC *Simple Features* geometry model and provides all the spatial functions in that standard as well as many others. GEOS is a core dependency of PostGIS, QGIS, GDAL, and Shapely.

If you need support using the GEOS library or would like to get involved in the community check out the Support page.

Capabilities

Spatial Model and Functions

- **Geometry Model**: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, GeometryCollection
- **Predicates**: Intersects, Touches, Disjoint, Crosses, Within, Contains, Overlaps, Equals, Covers
- **Operations**: Union, Distance, Intersection, Symmetric Difference, Convex Hull, Envelope, Buffer, Simplify, Polygon Assembly, Valid, Area, Length,
- **Prepared geometry** (using internal spatial indexes)
- **Spatial Indexes**: STR (Sort-Tile-Recursive) packed R-tree spatial index
- **Input/Output**: OGC Well Known Text (WKT) and Well Known Binary (WKB) readers and writers.

API Features

MEOS Family

- MEOS enables a single code base for manipulating moving features in multiple programming environments and languages
- Thin wrappers for several programming languages

6.4.1 Input/Output Functions

- Get the Well-Known Text (WKT) representation

`asText({tpoint,tpoint[],geo[]}) → {text,text[]}`

```
SELECT asText(tgeompoint 'SRID=4326;[Point(0 0 0)@2001-01-01, Point(1 1 1)@2001-01-02)');
-- [POINT Z (0 0 0)@2001-01-01, POINT Z (1 1 1)@2001-01-02)
SELECT asText(ARRAY[geometry 'Point(0 0)', 'Point(1 1)']);
-- {"POINT(0 0)","POINT(1 1)"}
```

- Get the Extended Well-Known Text (EWKT) representation

`asEWKT({tpoint,tpoint[],geo[]}) → {text,text[]}`

```
SELECT asEWKT(tgeompoint 'SRID=4326;[Point(0 0 0)@2001-01-01, Point(1 1 1)@2001-01-02)');
-- SRID=4326;[POINT Z (0 0 0)@2001-01-01, POINT Z (1 1 1)@2001-01-02)
SELECT asEWKT(ARRAY[geometry 'SRID=5676;Point(0 0)', 'SRID=5676;Point(1 1)']);
-- {"SRID=5676;POINT(0 0)","SRID=5676;POINT(1 1)"}
```

- Get the Moving Features JSON representation

`asMFJSON(tpoint,options=0,flags=0,maxdecdigits=15) → bytea`

The `options` argument can be used to add BBOX and/or CRS in MFJSON output:

  - 0: means no option (default value)
  - 1: MFJSON BBOX
  - 2: MFJSON Short CRS (e.g., EPSG:4326)
  - 4: MFJSON Long CRS (e.g., urn:ogc:def:crs:EPSG::4326)

https://docs.mobilitydb.com/MobilityDB/develop/

Connection between MobilityDB and MEOS

MobilityDB SQL definition

```
CREATE FUNCTION asText(tgeompoint)
  RETURNS text
  AS 'MODULE_PATHNAME', 'Tpoint_as_text'
  LANGUAGE C IMMUTABLE STRICT PARALLEL SAFE;
CREATE FUNCTION asText(tgeogpoint)
  RETURNS text
  AS 'MODULE_PATHNAME', 'Tpoint_as_text'
  LANGUAGE C IMMUTABLE STRICT PARALLEL SAFE;
```

MobilityDB C definition

```
PGDLLEXPORT Datum
Tpoint_as_text(PG_FUNCTION_ARGS)
{
  Temporal *temp = PG_GETARG_TEMPORAL_P(0);
  int dbl_dig_for_wkt = OUT_DEFAULT_DECIMAL_DIGITS;
  if (PG_NARGS() > 1 && ! PG_ARGISNULL(1))
    dbl_dig_for_wkt = PG_GETARG_INT32(1);
  char *str = tpoint_as_text(temp, dbl_dig_for_wkt);
  text *result = cstring2text(str);
  pfree(str);
  PG_FREE_IF_COPY(temp, 0);
  PG_RETURN_TEXT_P(result);
}
```

MEOS

https://github.com/MobilityDB/MobilityDB/wiki/Building-MobilityDB-and-MEOS

Connection between PyMEOS and MEOS

PyMEOS Classes

```python
class TPoint(Temporal[shp.Point, TG, TI, TS, TSS], ABC):
    """
    Abstract base class for both Geographic and Geometric types of any temporal subtype.
    """

    def as_wkt(self, precision: int = 15):
        """Return the string representation of the content of ``self``."""
        return tpoint_as_text(self._inner, precision)

    def as_ewkt(self, precision: int = 15):
        return tpoint_as_ewkt(self._inner, precision)
```

PyMEOS

PyMEOS CFFI Interface

```python
def tpoint_as_text(temp: 'const Temporal *', maxdd: int) -> str:
    temp_converted = _ffi.cast('const Temporal *', temp)
    result = _lib.tpoint_as_text(temp_converted, maxdd)
    result = _ffi.string(result).decode('utf-8')
    return result if result != _ffi.NULL else None
```

MEOS

https://github.com/MobilityDB/PyMEOS/wiki/PyMEOS-Architecture

MEOS API

Modules

Functions for PostgreSQL types
Functions for PostgreSQL types.

Functions for PostGIS types
Functions for PostGIS types.

Functions for set and span types
Functions for set and span types.

Functions for box types
Functions for box types.

Functions for temporal types
Functions for temporal types.

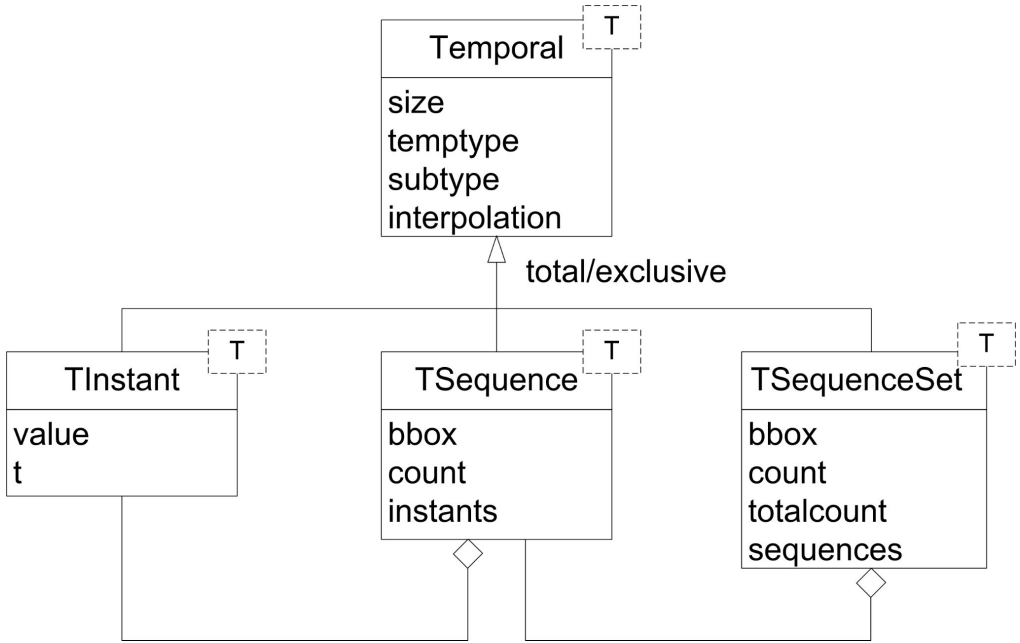
Functions

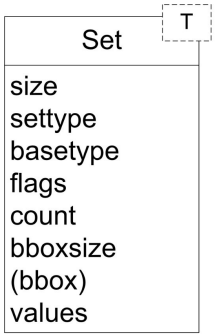
Temporal *	**tbool_in** (const char *str) Return a temporal boolean from its Well-Known Text (WKT) representation. More...
char *	**tbool_out** (const **Temporal** *temp) Return a temporal boolean from its Well-Known Text (WKT) representation. More...
char *	**temporal_as_hexwkb** (const **Temporal** *temp, uint8_t variant, size_t *size_out) Return the WKB representation of a temporal value in hex-encoded ASCII. More...
char *	**temporal_as_mfjson** (const **Temporal** *temp, bool with_bbox, int flags, int precision, char *srs) Return the MF-JSON representation of a temporal value. More...
uint8_t *	**temporal_as_wkb** (const **Temporal** *temp, uint8_t variant, size_t *size_out) Return the WKB representation of a temporal value. More...
Temporal *	**temporal_from_hexwkb** (const char *hexwkb) Return a temporal value from its HexEWKB representation. More...
Temporal *	**temporal_from_mfjson** (const char *mfjson) Return a temporal point from its MF-JSON representation. More...

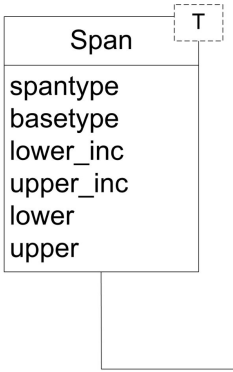
. . .

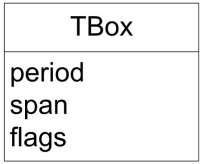
https://docs.mobilitydb.com/libmeos/html/modules.html

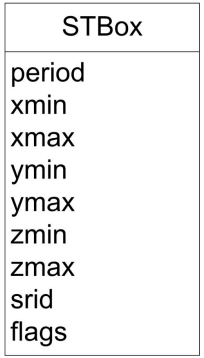
Temporal Model: Data Structures











https://libmeos.org/documentation/datastructures/

MobilityDB and OGC's Moving Features SWG

- MobilityDB aims at being 100% conformant with OGC's MF standards
- MF-JSON support for many years
  Extended for all temporal types: tbool, tint, tfloat, ttext
- OGC's Simple Feature Access WKT and WKB extended for all temporal types
- WKB is essential in distributed environments such as the cloud:
  Processes need an efficient way to exchange information
- MF-API support is being done in the European project EMERALDS
- Essential component for enabling Mobility as a Service (MaaS)
- MobilityDB provides an open-source testbench for designing and implementing MF-SWG standards

https://emeralds-horizon.eu/

References

1. A. Vaisman, E. Zimányi. Mobility Data Warehouses. ISPRS International Journal of Geo-Information, 8(4): 170, 2019.
2. E. Zimányi, M. Sakr, A. Lesuisse, M. Bakli, MobilityDB: A Mainstream Moving Object Database System. Proc. of SSTD 2019, p. 206-209.
3. M. Bakli, M. Sakr, E. Zimányi. Distributed Moving Object Data Management in MobilityDB. Proc. of BigSpatial '19, 2019.
4. M. Bakli, M. Sakr, E. Zimányi. Distributed Mobility Data Management in MobilityDB. Proc. of MDM 2020, pages 238-239, 2020.
5. E. Zimányi, M. Sakr, A. Lesuisse. MobilityDB: A Mobility Database based on PostgreSQL and PostGIS. ACM Transactions on Database Systems, 45(4), 2020.
6. M. Bakli, M. Sakr, E. Zimányi. Distributed Spatiotemporal Trajectory Query Processing in SQL. Proc. of SIGSPATIAL'20, pages 87-98, 2020.
7. M. Schoemans, M. Sakr, E. Zimányi. Implementing Rigid Temporal Geometries in Moving Object Databases. Proc. of ICDE 2021.
8. G. Rovinelli, S. Matwin, et al. Multiple aspect trajectories: a case study on fishing vessels in the Northern Adriatic Sea. Proc. of the Workshops of EDBT/ICDT 2021.

References

9. E. Zimányi, M. Sakr, M. Bakli, et al. MobilityDB: hands on tutorial on managing and visualizing geospatial trajectories in SQL. In SpatialAPI@SIGSPATIAL 2021.
10. M. Schoemans, M. Sakr, E. Zimányi. MOVE: Interactive Visual Exploration of Moving Objects. Proc. of EDBT/ICDT Workshops, 2022.
11. J. Godfrid, P. Radnic, A. Vaisman, E. Zimányi. Analyzing Public Transport in the City of Buenos Aires with MobilityDB. Public Transport, 14(2): 287-321, 2022.
12. A. Broniewski, M.I. Tirmizi, E. Zimányi, M. Sakr. Using MobilityDB and Grafana for Aviation Trajectory Analysis. Proc. of the 10th OpenSky Symposium, 2022.
13. M. Sakr, C. Ray, C. Renso. Big mobility data analytics: recent advances and open problems. GeoInformatica, 26(4): 541–549, 2022.
14. B. Brandoli, et al. From multiple aspect trajectories to predictive analysis: a case study on fishing vessels in the Northern Adriatic sea. GeoInformatica. 26(4): 551–579, 2022.
15. A. Broniewski, M. I. Tirmizi, E. Zimányi, M. Sakr. Using MobilityDB and Grafana for Aviation Trajectory Analysis. Engineering Proceedings, 28(1), 17, 2022.

Home The Project Publications Documentation Installation Tutorials Contact

MobilityDB

An open source geospatial trajectory data management & analysis platform.

View on Github

Many thanks for your attention !

Location tracking devices, such as GPS, are nowadays widely used in smartphones, and in vehicles. As a result, geospatial trajectory data are currently being collected and used in many application domains. MobilityDB provides the necessary database support for storing and querying such geospatial trajectory data

MobilityDB is implemented as an extension to PostgreSQL and PostGIS. It implements persistent database types, and query operations for managing geospatial trajectories and their time-varying properties.

Learn More

OPEN CHAT