Tutorial: Urban Trajectory Visualization

Data Model and Management

Ye Zhao and Farah Kamw

Outline

- Urban Data and Availability
- Urban Trajectory Data Types
- Data Preprocessing and Data Registration
- Urban Trajectory Data and Query Model
- Spatial Database and Indexing Schemes
- Urban Trajectory Data Management with Examples

Urban Data

Urban structures: Defining urban space

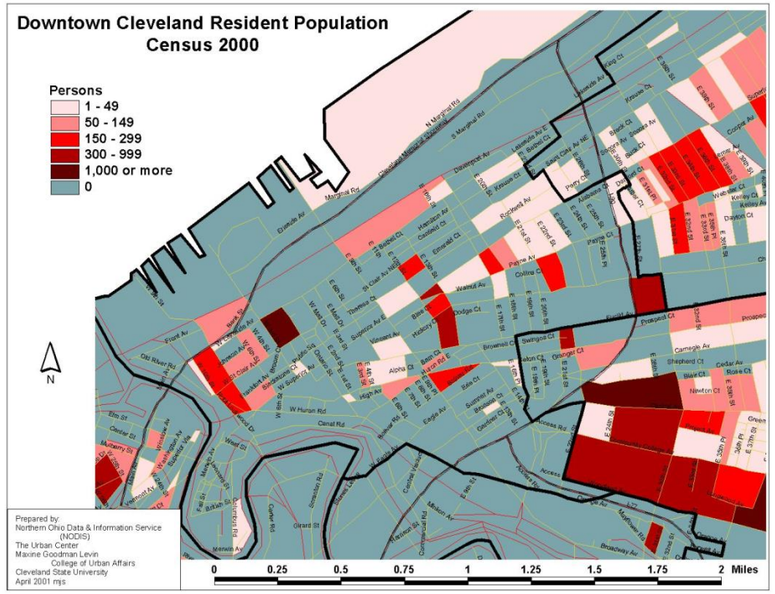
- **Road Network**
  - Roads may be categorized such as Primary, Secondary, Residential, …
- **POIs**

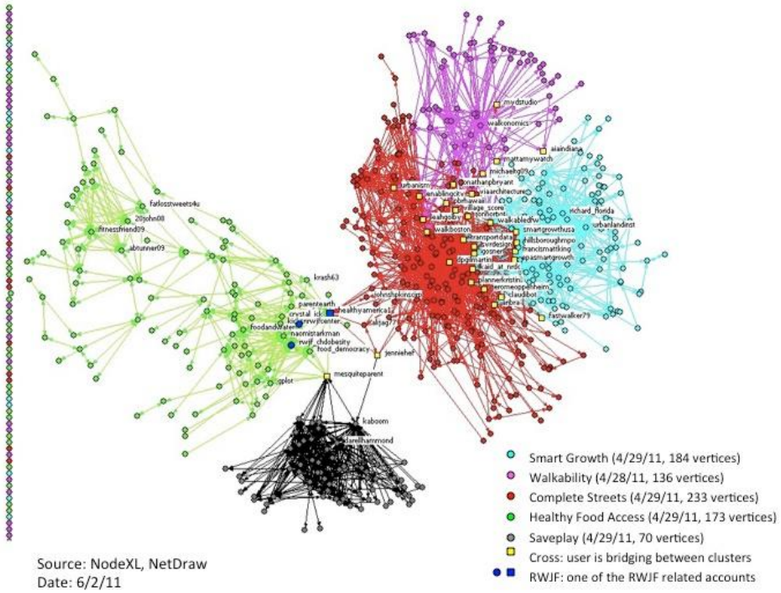
  POIs may be categorized into hierarchical levels.

Urban Data

Other Related Data

- **News, public reports, statistics…**
- **Social Media**
  - **Blogs, tweets, …**



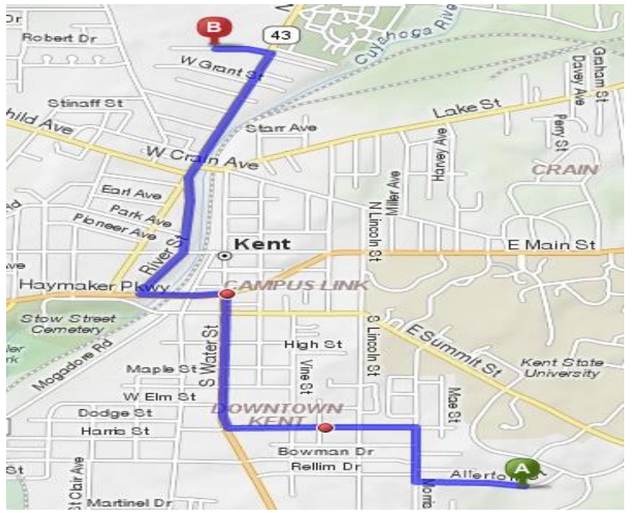


Urban Mobility Data

Urban trajectories: recording human behavior traces

- **Humans, vehicles, fleets, public transits, …**
  - GPS, Wi-Fi, Cellular, RFID, etc.

	T1
DATE/TIME	06/17/12 08:19:11
STATUS	1
SPEED	50KM/H
LATITUDE	22.533
LONGITUDE	114.044



Trajectory Data Sources

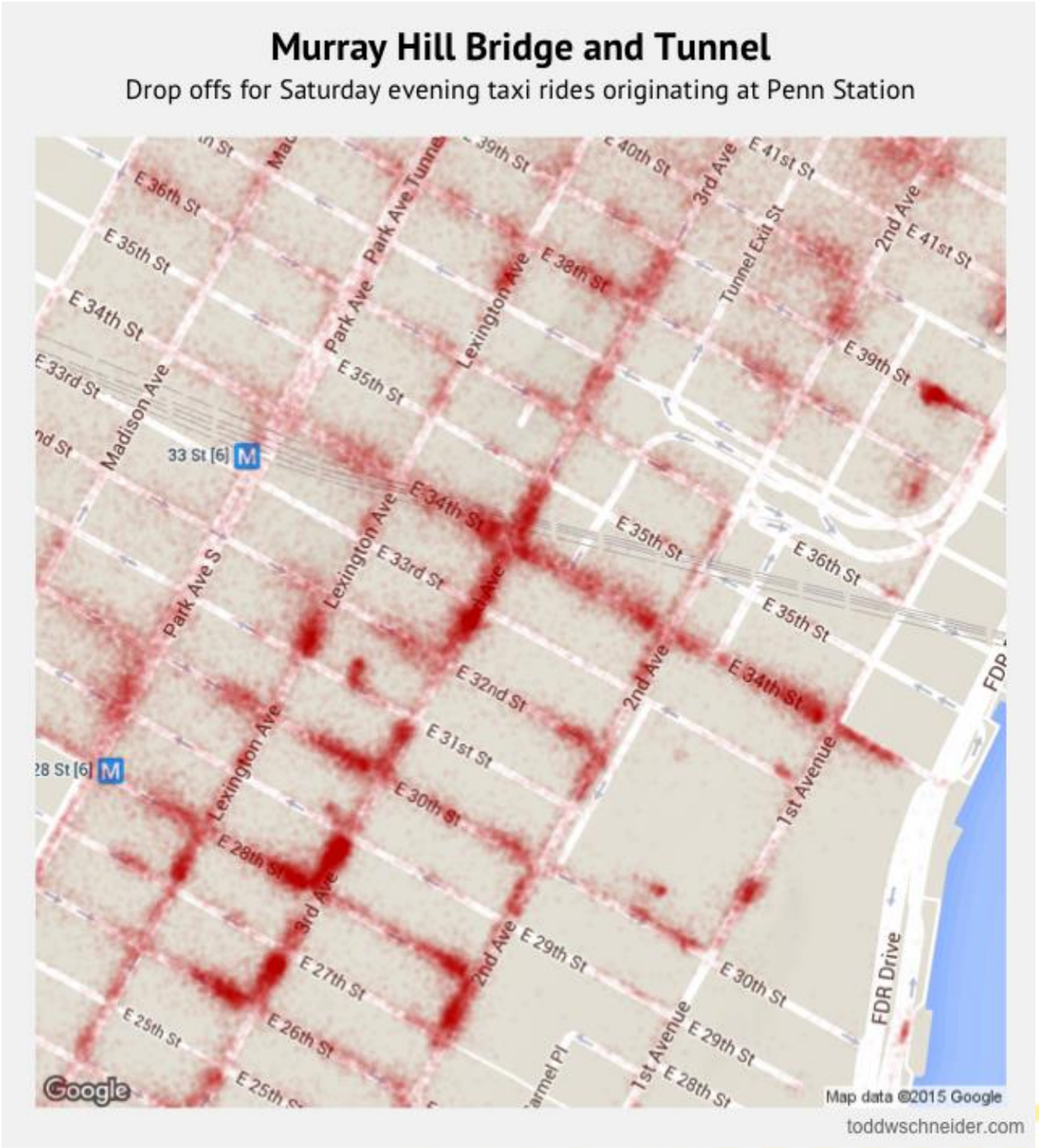
- Personal positioning devices
- Vehicle positioning devices
- Cellular stations
- Wifi access points
- RFID detectors
- Geo-tagged messages
- Others

Availability

- Private: data recorders, administrators, businesses, etc.
  - Privacy protection
  - Human subject sensitivity
  - Business properties and values
- Possibly acquire as collaborative partners
- Public: mostly anonymized data for research use
  - Examples:
    - Dataset of Trajectories of Taxi Cabs in Porto, Portugal.
    - Dataset of Trajectories of Taxi Cabs in Rome, Italy.
    - Dataset of OD of Taxi Cabs in NYC, USA.

Trajectory Data Types

- GPS Point Samples
  - E.g., Origin and destination of taxi trips



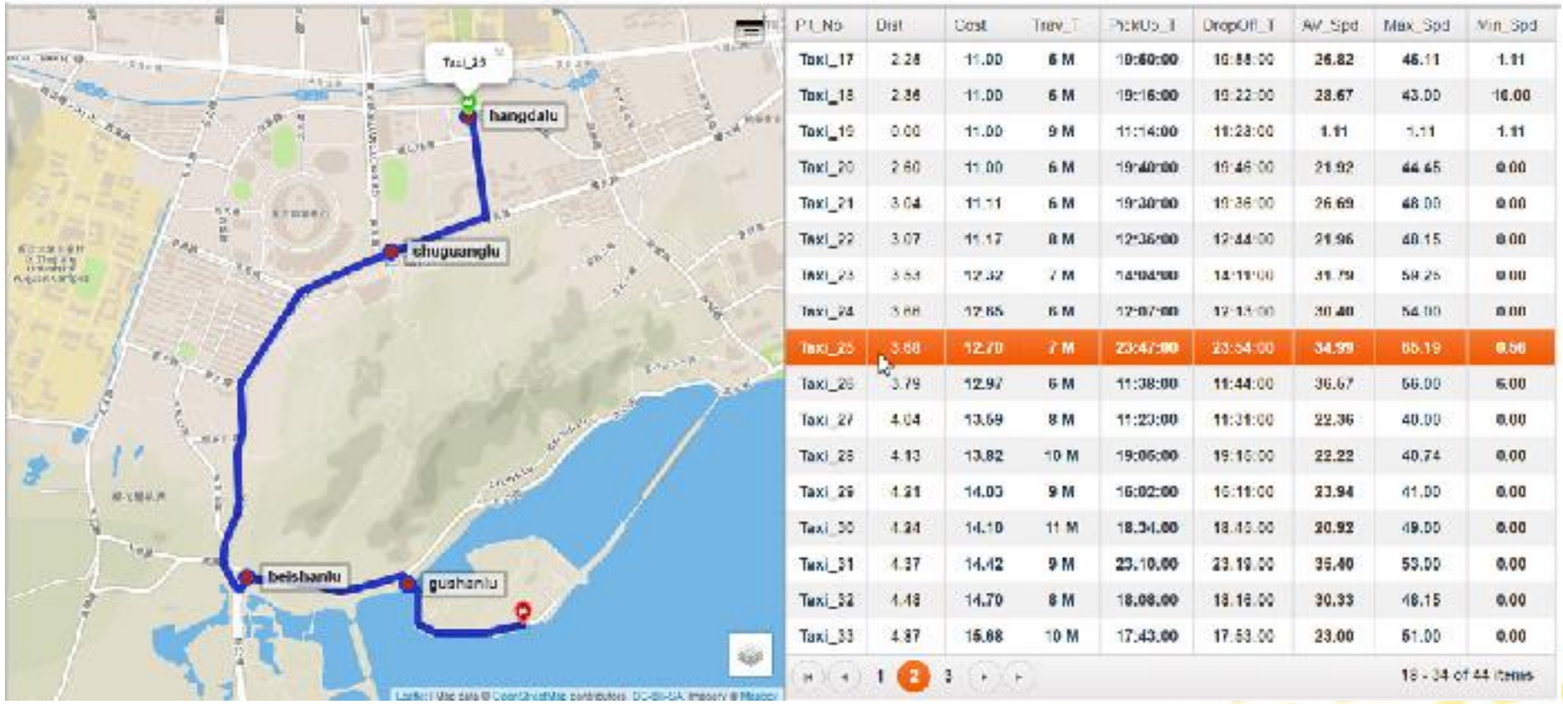
Trajectory Data Types

- POI Point Samples
  - E.g., Bus check-in and check-out data

Subway usage in Paris, a data visualization made by Data-Publica

Trajectory Data Types

- Polyline Trajectory Data
  - Linked point samples

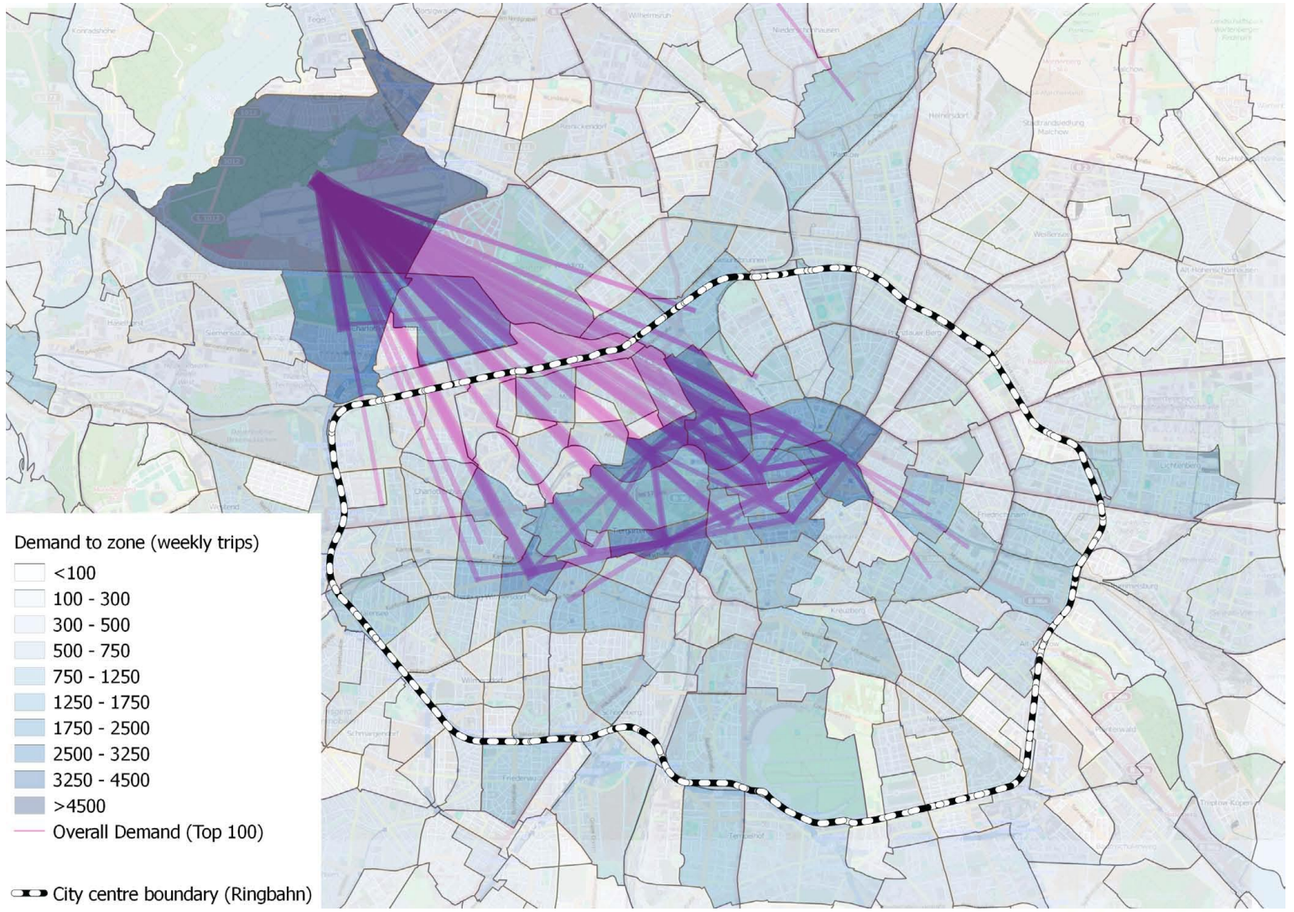


Pt_No	Dist	Cost	Trav_T	PickUp_T	DropOff_T	Av_Spd	Max_Spd	Min_Spd
Taxi_17	2.28	11.00	6 M	10:50:00	10:56:00	26.82	46.11	1.11
Taxi_18	2.36	11.00	6 M	10:16:00	10:22:00	28.67	43.00	16.00
Taxi_19	0.00	11.00	9 M	11:14:00	11:23:00	1.11	1.11	1.11
Taxi_20	2.60	11.00	6 M	10:40:00	10:46:00	21.92	44.45	0.00
Taxi_21	3.04	11.11	6 M	10:30:00	10:36:00	26.69	48.00	0.00
Taxi_22	3.07	11.17	8 M	12:36:00	12:44:00	21.96	40.15	0.00
Taxi_23	3.53	12.32	7 M	14:04:00	14:11:00	31.79	58.25	0.00
Taxi_24	3.66	12.65	6 M	12:57:00	13:03:00	30.40	54.00	0.00
Taxi_25	3.68	12.70	7 M	23:47:00	23:54:00	34.99	65.19	0.56
Taxi_26	3.79	12.97	6 M	11:38:00	11:44:00	36.67	56.00	5.00
Taxi_27	4.04	13.59	8 M	11:23:00	11:31:00	22.36	40.00	0.00
Taxi_28	4.13	13.82	10 M	19:06:00	19:16:00	22.22	40.74	0.00
Taxi_29	4.21	14.03	9 M	16:02:00	16:11:00	23.94	41.00	0.00
Taxi_30	4.24	14.10	11 M	18:34:00	18:45:00	20.92	49.00	0.00
Taxi_31	4.37	14.42	9 M	23:10:00	23:19:00	35.40	53.00	0.00
Taxi_32	4.48	14.70	8 M	18:08:00	18:16:00	30.33	48.15	0.00
Taxi_33	4.87	15.68	10 M	17:43:00	17:53:00	23.00	51.00	0.00

1 2 3 18 - 34 of 44 items

Trajectory Data Types

- Aggregated Data over Geospatial Spaces
  - Group motion behavior among regions, streets, POIs, …
  - Less sensitive, relatively easy to acquire and share

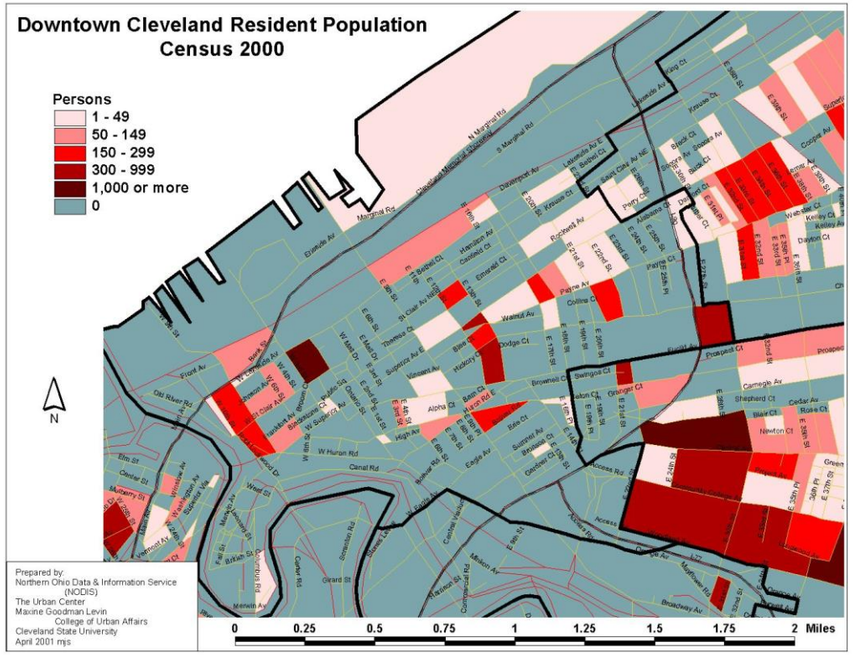


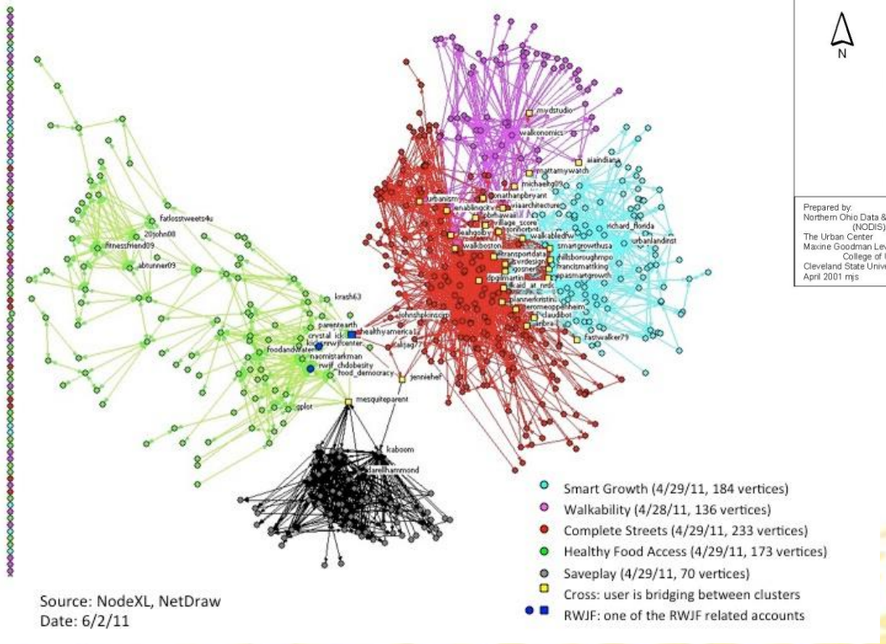
Taxi trips Berlin, by TU Berlin

Related Geospatial-Temporal Data

Inherently linked to trajectory data

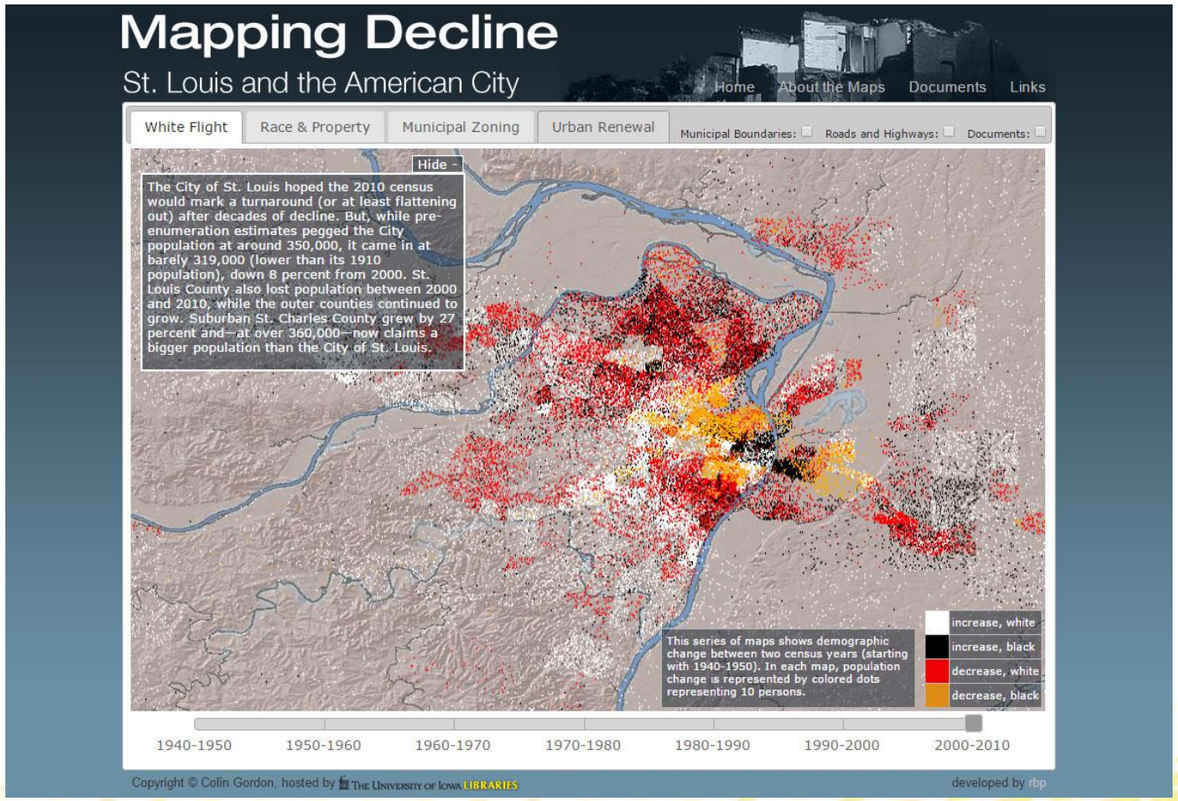
- Demographics
- News, public reports, business data…
- Social Media
  - Blogs, tweets …

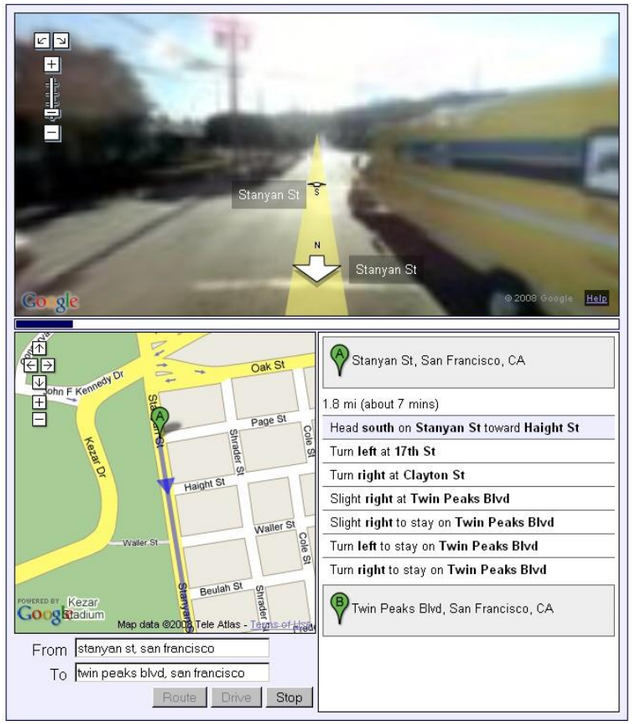




Related Geospatial-Temporal Data

- Law enforcement reports
- Street view pictures and videos





http://blogoscoped.com/files/google-street-view-api-large.jpg

Trajectory Data Issues

- Inaccuracy and error
  - Sampling errors
  - Transformation/transfer errors
  - Missing data
- Cleaning
  - Removing
  - Correcting and Interpolation

Data Cleaning

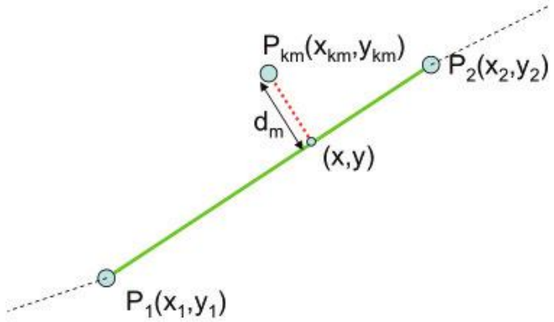
Trajectory_ID	latitude	longitude	date/time	speed
1	41.1414	-8.61864	7/1/2013 0:00	2.813648
1	41.1414	-8.6185	7/1/2013 0:01	47.05709
1	41.1425	-8.62033	7/1/2013 0:01	50.41256
1	41.1438	-8.62215	7/1/2013 0:01	39.55936
	41.1444	-8.62395	41456.00137	55.89229
	41.1448	-8.62668	41456.00154	14.12102
1	41.1447	-8.62737	7/1/2013 0:02	59.00425
2			41456.00582	10.39434
2	41.1599	-8.64035	7/1/2013 0:08	37.55105
2	41.1601	-8.6422	7/1/2013 0:08	46.44954
2	41.1605	-8.64445	7/1/2013 0:09	50.76127
2	41.1609	-8.64692	7/1/2013 0:09	63.91959
2	41.1615	-8.65		65.07201
2	41.162	-8.65317	7/1/2013 0:09	67.42575
2	41.1626	-8.65643	7/1/2013 0:10	77.02357
2	41.1632		41456.00721	60.35919
3	41.1404	-8.61296	7/1/2013 0:02	8.441072
3	41.1404	-8.61338	7/1/2013 0:02	16.8933
3	41.1403	-8.61421	7/1/2013 0:03	11.56684
3	41.1404	-8.61477	7/1/2013 0:03	22.91148
3	41.1404	-8.61591	7/1/2013 0:03	15.04686
3	41.1406	-8.61661	7/1/2013 0:03	43.04855
3	41.1414	-8.61847	7/1/2013 0:04	57.12187

Removing Errors →

Trajectory_ID	latitude	longitude	date/time	speed
1	41.1414	-8.61864	7/1/2013 0:00	2.813648
1	41.1414	-8.6185	7/1/2013 0:01	47.05709
1	41.1425	-8.62033	7/1/2013 0:01	50.41256
1	41.1438	-8.62215	7/1/2013 0:01	39.55936
1	41.1447	-8.62737	7/1/2013 0:02	59.00425
2	41.1599	-8.64035	7/1/2013 0:08	37.55105
2	41.1601	-8.6422	7/1/2013 0:08	46.44954
2	41.1605	-8.64445	7/1/2013 0:09	50.76127
2	41.1609	-8.64692	7/1/2013 0:09	63.91959
2	41.162	-8.65317	7/1/2013 0:09	67.42575
2	41.1626	-8.65643	7/1/2013 0:10	77.02357
3	41.1404	-8.61296	7/1/2013 0:02	8.441072
3	41.1404	-8.61338	7/1/2013 0:02	16.8933
3	41.1403	-8.61421	7/1/2013 0:03	11.56684
3	41.1404	-8.61477	7/1/2013 0:03	22.91148
3	41.1404	-8.61591	7/1/2013 0:03	15.04686
3	41.1406	-8.61661	7/1/2013 0:03	43.04855
3	41.1414	-8.61847	7/1/2013 0:04	57.12187

Trajectory Map Matching

- Map GPS sampling points to roads
- A research topic in GIS with many algorithms

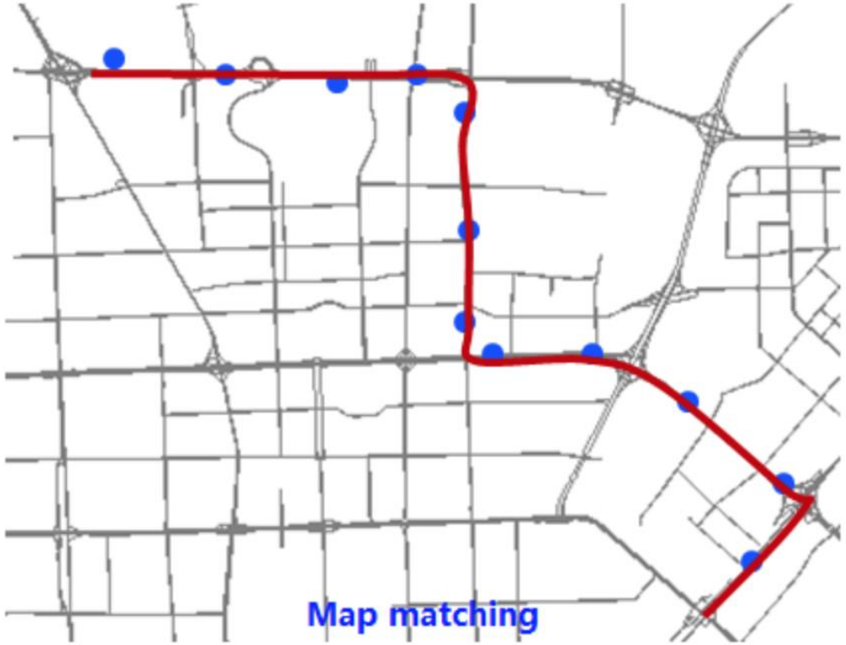


Technological Issues in the Design of Cost-Efficient Electronic Toll Collection Systems by José Santa, Rafael Toledo-Moreo, Benito Ubeda, Antonio Skarmeta

A Dilution-matching-encoding compaction of trajectories over road networks, by Ranit GotsmanYaron Kanza

Trajectory Data Registration

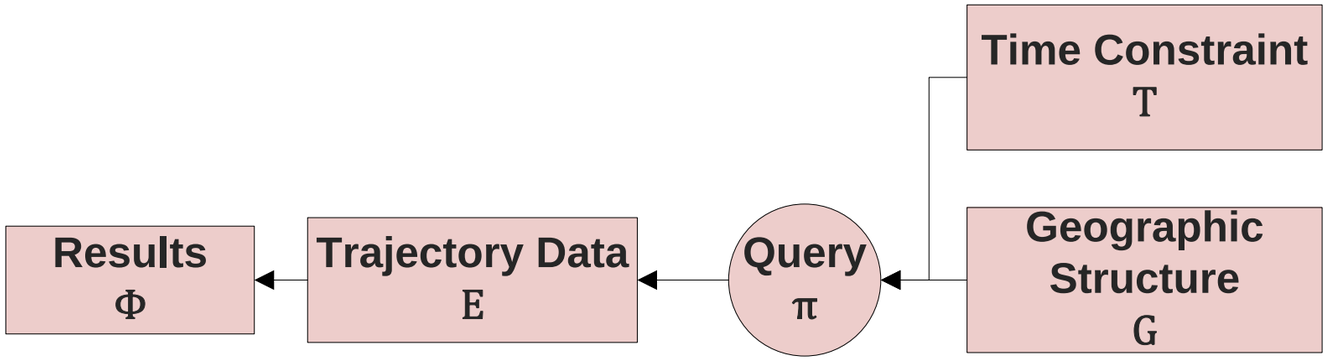
- Also mapped to POIs, geo-regions and other geospatial elements
- A critical process in most visualization tasks
- Link trajectory data to urban structures/objects and then urban attributes



From Trajectory Data Mining: A Review of Methods and Applications by *Jean Damascène Mazimpaka, Sabine Timpf*

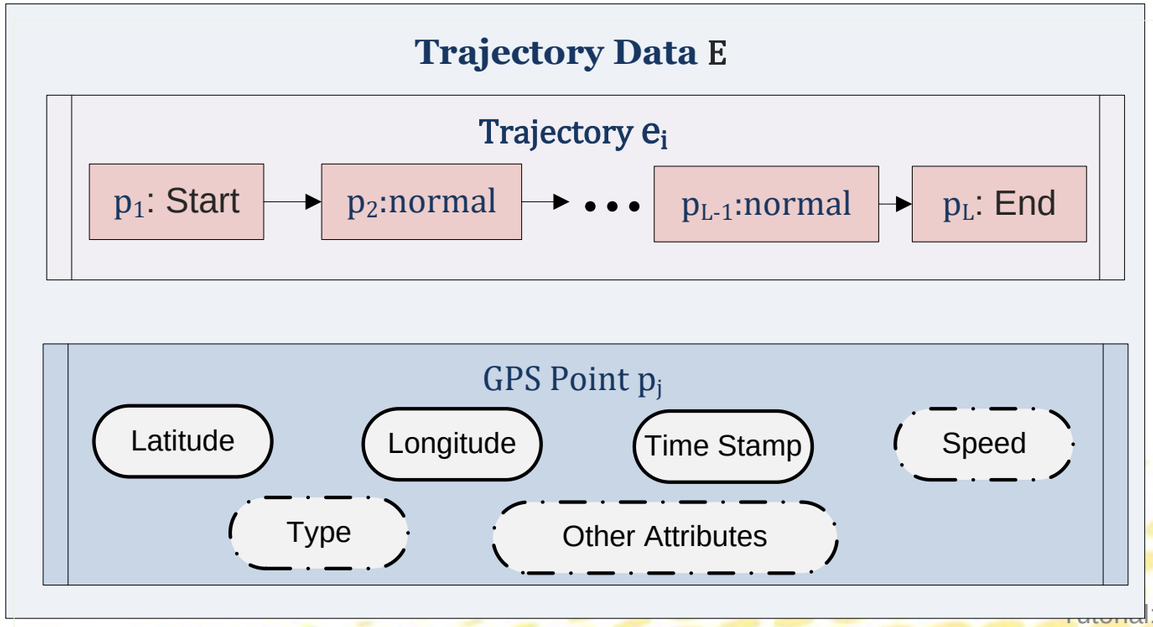
Urban Trajectory Data Query Model

- Accessing trajectory data involves five components
  - Trajectory data
  - Geographic structures
  - Time constraints
  - Query modes
  - Query results



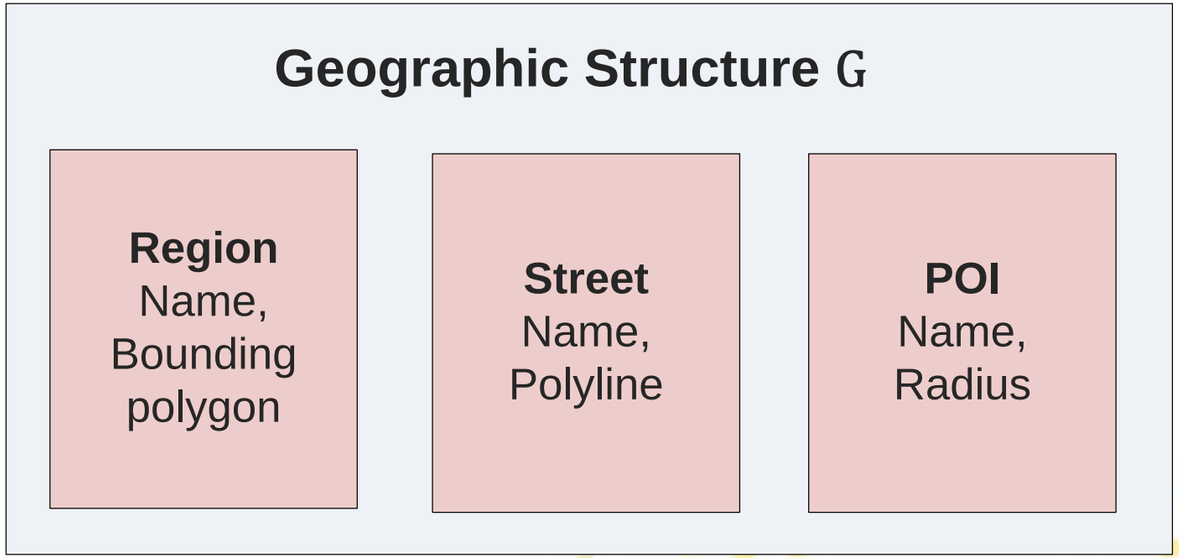
Trajectory Data Elements

- Linking GPS points to form a trajectory
- May only know Start/End points
- May not have accurate Latitude and Longitude
  - Approximate region (cellular tower region)
  - Street address
  - POI



Geographic Structures

- Provide meaningful conditions for various data access
- Highly related to GIS information of
  - Regions (zipcodes, demographics, …)
  - Street networks
  - POI
- Geometric information
- Semantic information



Time Constraints

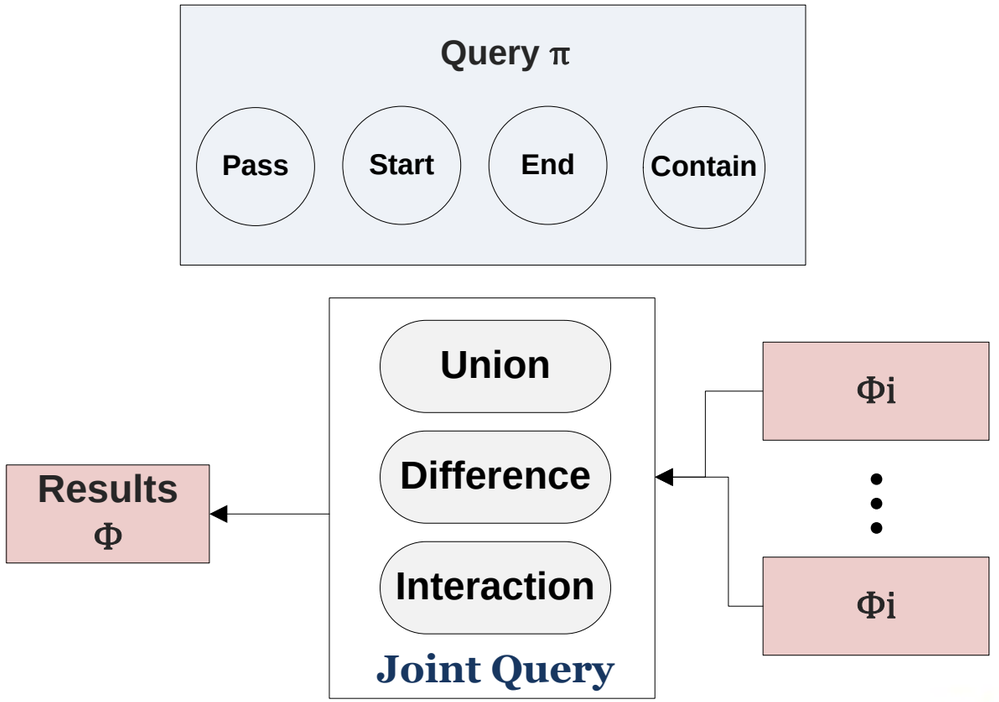
- Time Period
- In-week distribution
  - Monday, Tuesday, … Sunday
- In-Day distribution
  - Morning, Noon, Afternoon, Night
  - Hourly

Query Modes

- A query retrieves trajectory data items Φi that
  - pass
  - start from
  - end at
  - are contained inside

given geographic structure

- Joint query
  - Start from A end at B
  - … …



Query Results

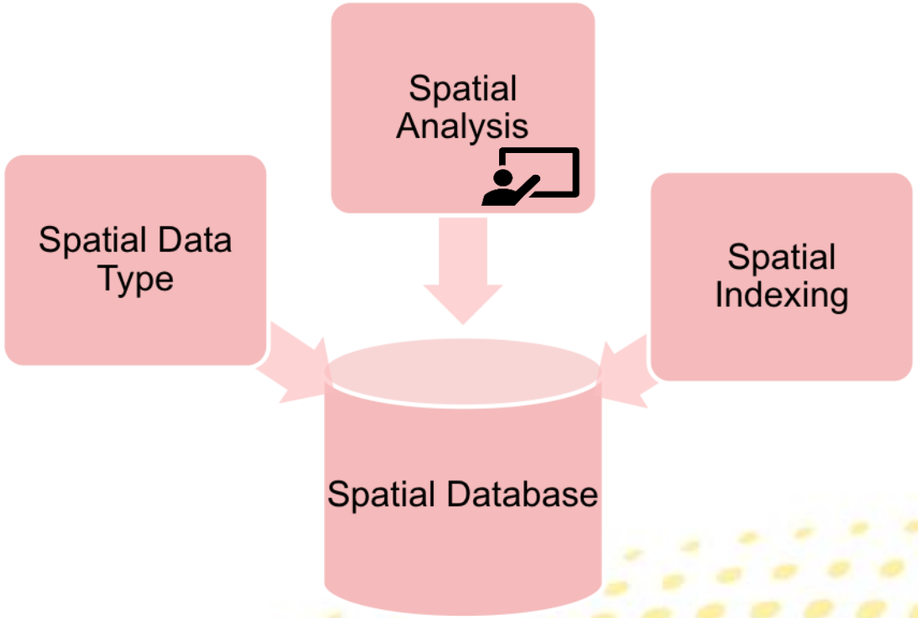
- A series of trajectory elements including points, traces and their geographical structures (e.g., streets)
- To be visualized
  - Visualize trajectories directly
  - Visualize specific points
  - Project and aggregate over street/region/POI

Supporting Data Retrieval

- Spatial database specifically designed for trajectory data
  - Data indexing structure and algorithm
  - Data aggregation
- Many challenges
  - Performance
  - Web transfer and communication
  - Big data issues

Spatial Database

- Database that is optimized for storing and querying a geometric spatial data.
- It allows the representation of Spatial Data Types SDT such as points, lines, polygons.



Spatial Database

- Supports SDT with:
  - Spatial indexing.
  - Efficient query language.

    Relationships among geographic structures
  - Efficient algorithms for spatial joins.

Spatial Database Systems

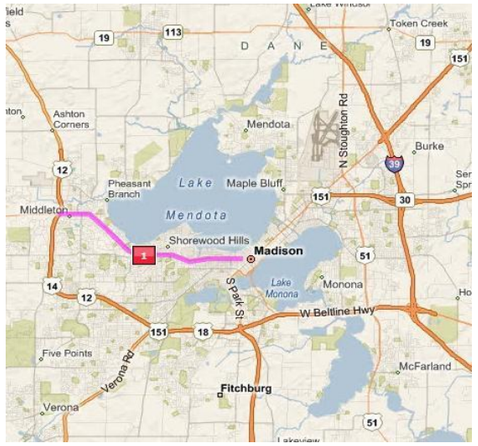
- PostGIS extension with Postgresql database.
- MonetDB/GIS extension for MongoDB.
- Spatial extension with MySQL.
- Oracle Spatial.
- Geocouch extension with CouchDB.
- Microsoft SQL Server.
- Others.

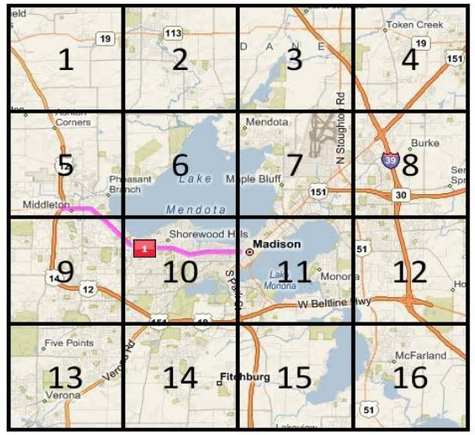
Spatial Indexing

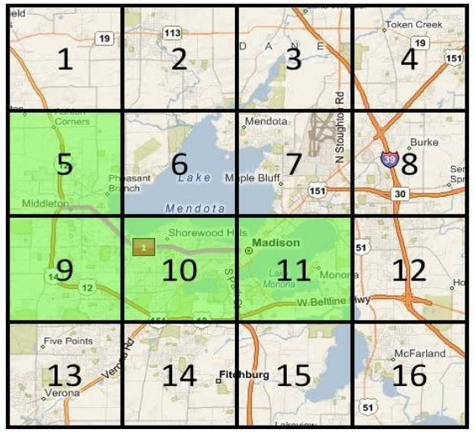
- **Spatial index** is a type of extended index that allows you to index a spatial data type to optimize spatial queries.
  - Speed up spatial queries
  - Efficient for modification
  - Redundancy but not too much space
- **Algorithms:**
  - Grid (Spatial Index)
  - R-tree/R+ tree/R* tree
  - Quadtree
  - kd-tree
  - …

Grid (Spatial Index)

- A **grid** is a tessellation of 2-D surface.
- It divides the surface into a series of contiguous cells.
- Cells is assigned unique identifiers.



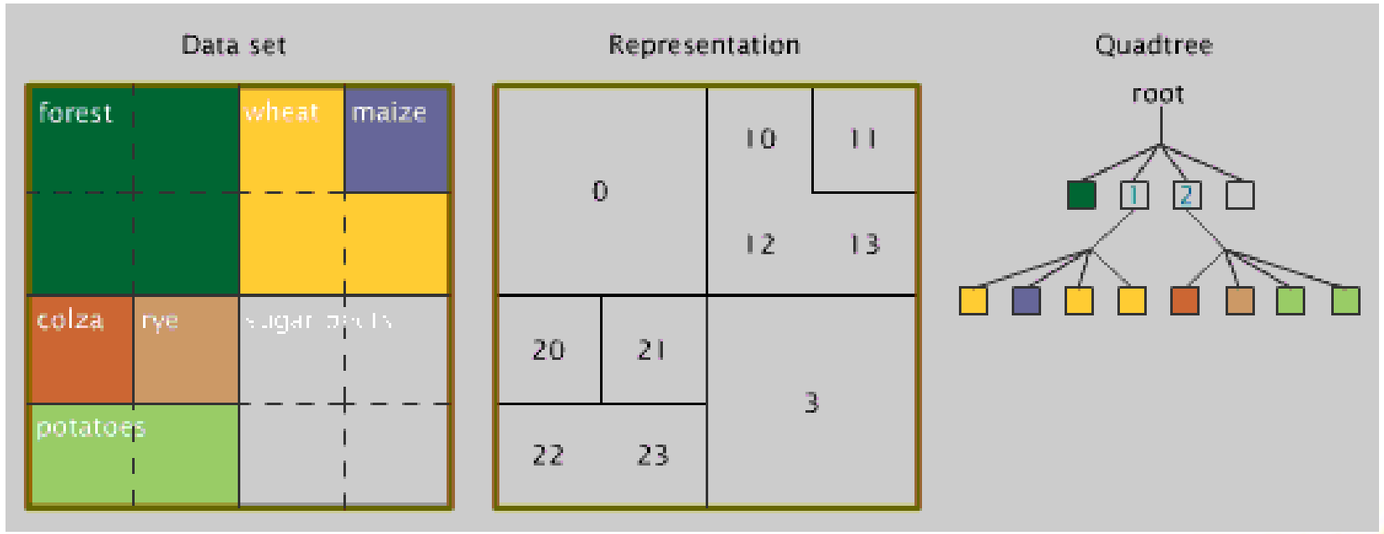




https://blogs.msdn.microsoft.com/isaac/2007/11/27/spatial-indexing-part-2-a-simple-spatial-indexing-scheme/

Quadtree

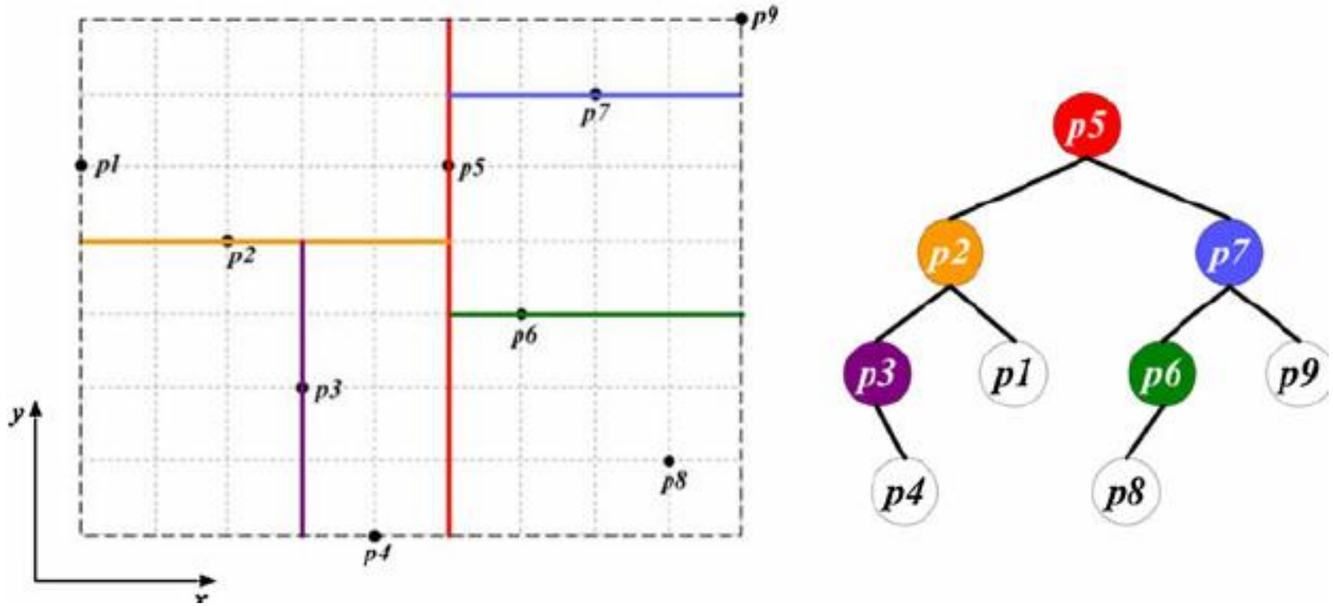
- A quadtree is a tree data structure in which each node has zero or four children.
- It is recursively dividing a flat 2-D space into four quadrants.



http://www.gitta.info/SpatPartitio/en/html/RegDecomp_learningObject3.

KD-Tree

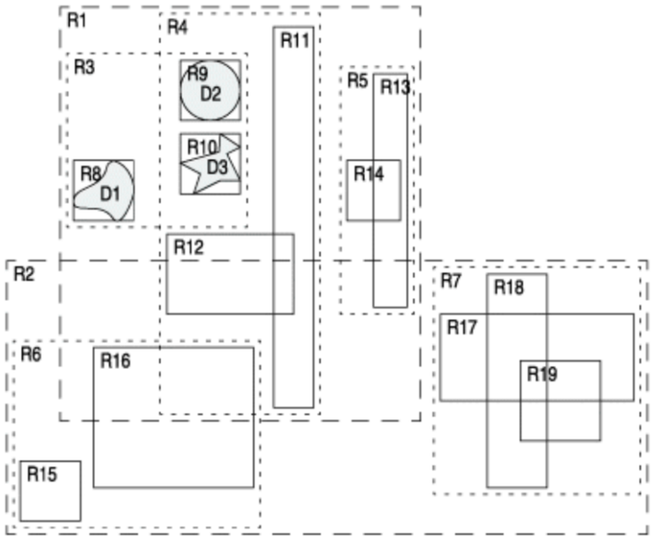
- A k-d tree is a space-partitioning data structure for organizing points in a *k*-dimensional space.

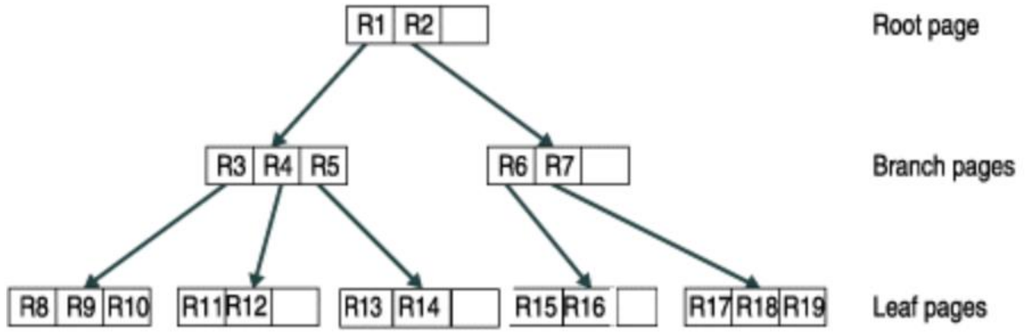


https://www.researchgate.net/figure/Diagram-of-the-KD-tree-structure_fig1_266556783

R-tree

- The R-tree indexing method organizes data in a tree-shaped structure.
- The index uses a bounding box.
- Bounding box is a rectilinear shape that completely contains the data objects or other bounding boxes.





https://www.ibm.com/support/knowledgecenter/en/SSGU8G_11.50.0/com.ibm.rtree.doc/sii-overview-27706.htm

GIST Indexing

- **GIST** stands for Generalized Search Tree.
- A data structure and API used in PostGIS.
- For spatial indexing, R-Tree is integrated into Gist

Spatial Queries

- A **spatial query** is a special type of database **query** supported by spatial databases and indexing, such as
  - Finding all geographic structures within a given region boundary
  - Finding all points inside a given geometric region
  - Finding all trajectories intersected by a query linestring

Urban Trajectory Data Management

- Urban trajectory data tables
- Spatial queries over trajectory data
- Examples with PostGIS and PostGreSQL database

Postgres Example

- Input GPS points of trips data set.
  - ✓Create table.

```
CREATE TABLE Porto_GPS_Points
(
    tripid integer,
    latitude double precision,
    longitude double precision,
    pdateTime timestamp without time zone,
    speed double precision
);
```

  - ✓Import from csv file.

```
COPY Porto_GPS_Points FROM 'File_Path\File_Name.csv'
DELIMITERS ',' CSV HEADER;
```

Raw Trajectory Dataset

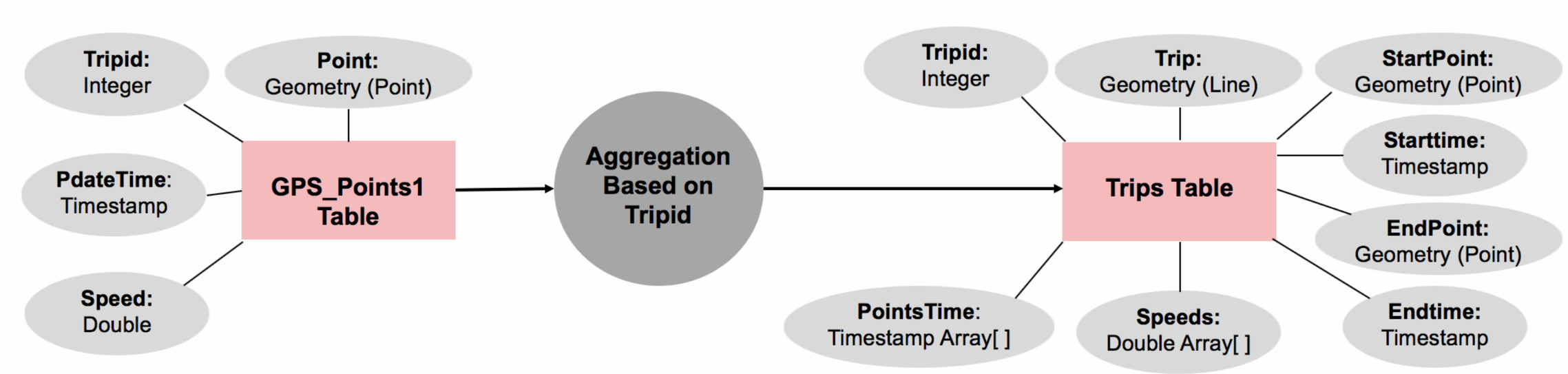
- GPS points
- Each has Trip ID indicating the trajectory it belongs to
- Trajectory (i.e., Trip) attributes vary by different applications
  - Time
  - Speed
  - Others
- A csv file from Porto datasets:
  - Taxi trajectories by all the 442 taxis running in the city of Porto, Portugal

tripid	latitude	longitude	pdatetime	speed
1	41.1414	-8.61864	7/1/2013 0:00	2.813648
1	41.1414	-8.6185	7/1/2013 0:01	47.05709
1	41.1425	-8.62033	7/1/2013 0:01	50.41256
1	41.1438	-8.62215	7/1/2013 0:01	39.55936
1	41.1444	-8.62395	7/1/2013 0:01	55.89229
2	41.1598	-8.63985	7/1/2013 0:08	10.39434
2	41.1599	-8.64035	7/1/2013 0:08	37.55105
2	41.1601	-8.6422	7/1/2013 0:08	46.44954
2	41.1605	-8.64445	7/1/2013 0:09	50.76127
2	41.1609	-8.64692	7/1/2013 0:09	63.91959
2	41.1615	-8.65	7/1/2013 0:09	65.07201
2	41.162	-8.65317	7/1/2013 0:09	67.42575
3	41.1404	-8.61296	7/1/2013 0:02	8.441072
3	41.1404	-8.61338	7/1/2013 0:02	16.8933
3	41.1403	-8.61421	7/1/2013 0:03	11.56684
3	41.1404	-8.61477	7/1/2013 0:03	22.91148
4	41.152	-8.57468	7/1/2013 0:00	2.735915
4	41.1519	-8.57471	7/1/2013 0:01	0.200943
4	41.1519	-8.5747	7/1/2013 0:01	2.787091
5	41.1805	-8.64599	7/1/2013 0:04	0.80342
5	41.1805	-8.64595	7/1/2013 0:05	13.49371
5	41.18	-8.64605	7/1/2013 0:05	32.9951
5	41.1789	-8.6468	7/1/2013 0:05	55.27292
5	41.1785	-8.6495	7/1/2013 0:05	54.87539

Moreira-Matias, L., Gama, J., Ferreira, M., Mendes-Moreira, J., Damas, L., ”Predicting Taxi–Passenger Demand Using Streaming Data”. In: IEEE Transactions on Intelligent Transportation Systems, vol.14, no.3, pp.1393-1402, September (2013)

Overview of Data Table Scheme

- Create tables from raw data
  - Table of Point
  - Table of Trips



GPS_Points Table

- Each record is one sampling point form a trajectory
- A GIST indexing is created together

```
CREATE TABLE GPS_Points1 AS
(
        SELECT tripid,
        ST_MakePoint(longitude, latitude) AS
        point,
        pdateTime timestamp,
        Speed double,
        FROM Porto_GPS_Points //csv file
);
CREATE INDEX gps_points1_Index ON GPS Points
USING GIST (point);
```

Edit Data - TrajAnalytics (vis.cs.kent.edu:5432) - traja_porto_example_1_1_2 - gps_points1

File Edit View Tools Help

100 rows

	tripid integer	point geometry	pdatetime timestamp without time zone	speed double precision
1	1	0101000020E6100000B0FECF61BE3C21C0C3D32B6519924440	2013-07-01 00:00:58	2.81364787161369
2	1	0101000020E6100000E9263108AC3C21C0C3D32B6519924440	2013-07-01 00:01:13	47.0570876893036
3	1	0101000020E61000009274CDE49B3D21C00AD7A3703D924440	2013-07-01 00:01:28	50.4125585274583
4	1	0101000020E6100000AD69DE718A3E21C01895D40968924440	2013-07-01 00:01:43	39.5593607725613
5	1	0101000020E6100000ACADD85F763F21C06DC5FEB27B924440	2013-07-01 00:01:58	55.8922906714891
6	2	0101000020E6100000F2B0506B9A4721C04DF38E5374944440	2013-07-01 00:08:23	10.3943369020428
7	2	0101000020E6100000B9FC87F4DB4721C0B1506B9A77944440	2013-07-01 00:08:38	37.5510503578942
8	2	0101000020E61000007FFB3A70CE4821C0780B24287E944440	2013-07-01 00:08:53	46.4495371644818
9	2	0101000020E61000007DD0B359F54921C006819543 8B944440	2013-07-01 00:09:08	50.7612715519619
10	2	0101000020E6100000B3412619394B21C094F6065F98944440	2013-07-01 00:09:23	63.9195870821666
11	2	0101000020E6100000CDCCCCCCCC4C21C0E9263108AC944440	2013-07-01 00:09:38	65.0720134799632
12	2	0101000020E6100000E674594C6C4E21C0DBF97E6ABC944440	2013-07-01 00:09:53	67.4257525162536
13	3	0101000020E61000002783A3E4D53921C0E02D90A0F8914440	2013-07-01 00:02:31	8.44107231852244
14	3	0101000020E61000007C0A80F10C3A21C0E02D90A0F8914440	2013-07-01 00:02:46	16.8932998468339
15	3	0101000020E610000098C0ADBB793A21C07DD0B359F5914440	2013-07-01 00:03:01	11.5668369716513
16	3	0101000020E6100000B41F2922C33A21C0E02D90A0F8914440	2013-07-01 00:03:16	22.9114820072356
17	4	0101000020E6100000A054FB743C2621C0FA7E6ABC74934440	2013-07-01 00:00:54	2.7359154227332
18	4	0101000020E61000004A5E9D63402621C096218E7571934440	2013-07-01 00:01:09	0.200942668349696
19	4	0101000020E6100000BC0512143F2621C096218E7571934440	2013-07-01 00:01:24	2.78709139803541
20	5	0101000020E610000009168733BF4A21C0C976BE9F1A974440	2013-07-01 00:04:51	0.803419931910827
21	5	0101000020E6100000D0B359F5B94A21C0C976BE9F1A974440	2013-07-01 00:05:06	13.4937061991521
22	5	0101000020E61000005F29CB10C74A21C0D7A3703D0A974440	2013-07-01 00:05:21	32.9951025163521
23	5	0101000020E6100000091B9E5E294B21C090A0F831E6964440	2013-07-01 00:05:36	55.2729189114686
24	5	0101000020E610000068195438B4C21C0022B8716D9964440	2013-07-01 00:05:51	54.8753885314205

Trajectory Tables

- Each record is a complete trajectory as a connected line of its sampling points.
- Use LineString fields for trajectory geometry, start/end point
    - In PostGIS, a Linestring accommodates point, multipoint, or line geometries

```
CREATE TABLE Trips AS
      SELECT *, ST_StartPoint(trip) AS startpoint, ST_EndPoint(trip) AS endpoint,
      FROM
      (
            SELECT tripid, ST_MakeLine(point order by pdatetime) AS trip,
            min(pdatetime) AS starttime, max(pdatetime) AS endtime,
            array_agg(speed order by pdatetime) AS speeds,
            array_agg(pdateTime order by pdatetime) AS pointstime
            FROM GPS_Points1
            GROUP BY tripid
      )
```

Indexing Trajectory Tables

- Create spatial index Gist over them

```
CREATE INDEX Trips_index1 ON Trips USING GIST (trip);
CREATE INDEX Trips_index2 ON Trips USING GIST (startpoint);
CREATE INDEX Trips_index3 ON Trips USING GIST (endPoint);
```

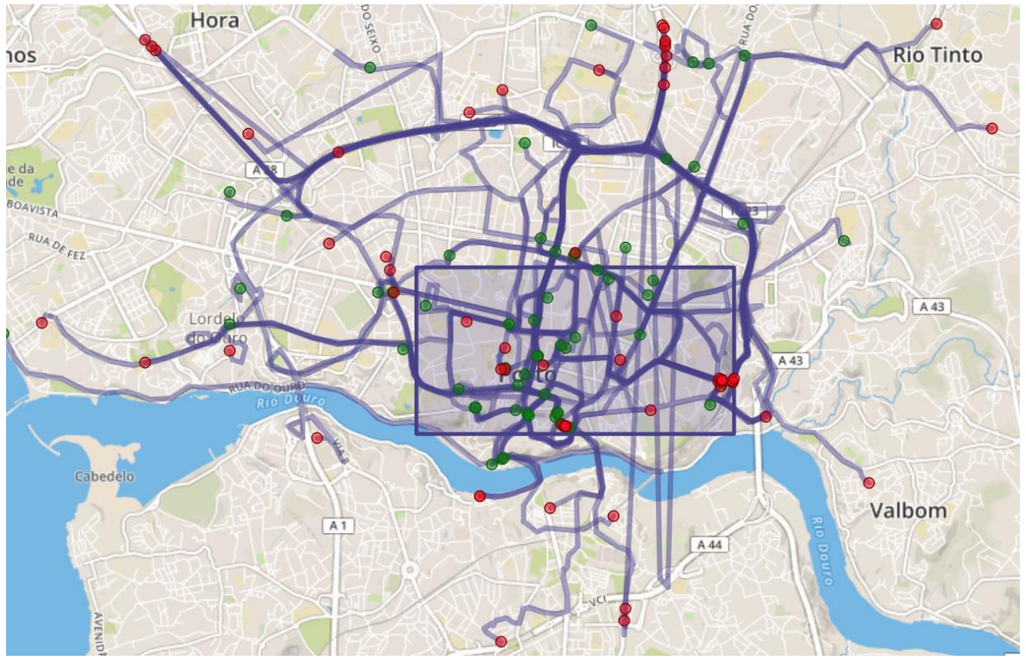
	tripid integer	trip geometry	startpoint geometry	starttime timestamp without time	endpoint geometry	endtime timestamp without tim	speeds double precision[]	pointstime timestamp without time zone[]
1	1	0102000020E6100000050(	0101000020E6100	2013-07-01 00:00:58	0101000020E61	2013-07-01 00:01:58	{2.81364787161369,47.0570876893036,50.4125585274583,39.5593607725613,55.8922906714891}	{"2013-07-01 00:00:58","2013-07-01 00:01:13","20
2	2	0102000020E6100000070(	0101000020E6100	2013-07-01 00:08:23	0101000020E61	2013-07-01 00:09:53	{10.3943369020428,37.5510503578942,46.4495371644818,50.7612715519619,63.9195870821666,65.0720134799632,67.4257525162536}	{"2013-07-01 00:08:23","2013-07-01 00:08:38","20
3	3	0102000020E6100000040(	0101000020E6100	2013-07-01 00:02:31	0101000020E61	2013-07-01 00:03:16	{8.44107231852244,16.8932998468339,11.5668369716513,22.9114820072356}	{"2013-07-01 00:02:31","2013-07-01 00:02:46","20
4	4	0102000020E6100000030(	0101000020E6100	2013-07-01 00:00:54	0101000020E61	2013-07-01 00:01:24	{2.7359154227332,0.200942668349696,2.78709139803541}	{"2013-07-01 00:00:54","2013-07-01 00:01:09","20
5	5	0102000020E6100000050(	0101000020E6100	2013-07-01 00:04:51	0101000020E61	2013-07-01 00:05:51	{0.803419931910827,13.4937061991521,32.9951025163521,55.2729189114686,54.8753885314205}	{"2013-07-01 00:04:51","2013-07-01 00:05:06","20

Queries over Data Tables

1. Pass Query

Input parameters:

1. Query Region.
2. [Time1,Time2].




```
SELECT Trajectory ID, Trajectory FROM Trajectories
WHERE Start Time BETWEEN (Time1 AND Time2)
AND(Query Region INTERSECT Trajectory).
```

Pass Query Example

- **SELECT** tripid,**ST_AsText**(trip) **AS** trajectory **FROM** **Trips** **WHERE** starttime **BETWEEN** '2013-07-05 00:00:00' **AND** '2013-07-01 09:00:00' **AND** **ST_Intersects**(**ST_GeomFromText**('POLYGON((-8.624954223632814 41.1437074383407,-8.624954223632814 41.1597351586533,-8.586158752441408 41.1597351586533,-8.586158752441408 41.1437074383407,-8.624954223632814 41.1437074383407))',4326),trip);

Data Output | Explain | Messages | History

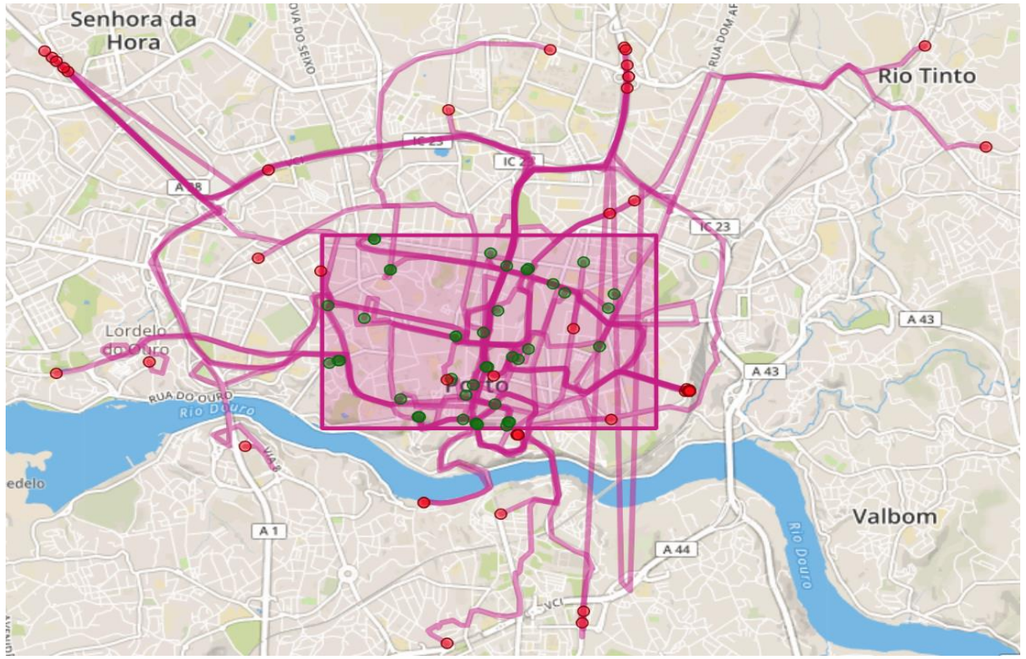
	tripid integer	trajectory text
1	296	LINESTRING(-8.58264 41.1806,-8.58264 41.1805,-8.58226 41.1803,-8.58253 41.1788,-8.58327 41.177,-8.58424 41.1751,-8.58521 41.1731,-8.58653 41.1705,-8.58793 41.1679,-8.58898 41.1653,-8.59068 41.1628,-8.59233 41.1604,-8.59288 41.1595,-8.59327 41.1588,-8.59408
2	369	LINESTRING(-8.60657 41.1447,-8.60739 41.1454,-8.60712 41.1463,-8.60687 41.1468,-8.60687 41.1469,-8.60728 41.1471,-8.60855 41.1474,-8.6093 41.1476,-8.61085 41.148,-8.61309 41.1483,-8.62226 41.1477,-8.62484 41.1485,-8.62582 41.1502,-8.62608 41.152,-8.62607 4
3	310	LINESTRING(-8.60476 41.1612,-8.6053 41.1604,-8.60575 41.1591,-8.60451 41.1587,-8.60476 41.1564,-8.60472 41.1543,-8.60595 41.1535,-8.6089 41.1535,-8.6098 41.153,-8.61043 41.1511,-8.61109 41.1504,-8.61101 41.1506,-8.61125 41.15,-8.61126 41.1486,-8.61145 41.1
4	382	LINESTRING(-8.61071 41.1455,-8.61077 41.1455,-8.6108 41.1457,-8.61051 41.146,-8.60934 41.1466,-8.60864 41.1471,-8.60878 41.1473,-8.60909 41.1476,-8.61081 41.1479,-8.61062 41.1489,-8.61038 41.1496,-8.60962 41.1511,-8.61016 41.1523,-8.60986 41.1546,-8.60962
5	308	LINESTRING(-8.60648 41.1446)
6	347	LINESTRING(-8.60635 41.1446,-8.60637 41.1446,-8.60668 41.1447,-8.60741 41.1457,-8.60702 41.1466,-8.60687 41.1468,-8.60744 41.1472,-8.6085 41.1474,-8.60975 41.1477,-8.61204 41.148,-8.62033 41.1476,-8.62074 41.1486,-8.6207 41.1487,-8.6207 41.1487,-8.6207 41.
7	304	LINESTRING(-8.60271 41.1835,-8.59997 41.1827,-8.5973 41.1819,-8.59823 41.18,-8.59945 41.1786,-8.60202 41.1793,-8.60474 41.1801,-8.60514 41.1802,-8.60613 41.1788,-8.608 41.177,-8.60845 41.1755,-8.6068 41.1749,-8.60589 41.1737,-8.6059 41.1737,-8.60536 41.172
8	297	LINESTRING(-8.63139 41.1794,-8.63153 41.1794,-8.63338 41.1797,-8.63566 41.181,-8.63572 41.1808,-8.63365 41.1797,-8.63136 41.1792,-8.62843 41.1795,-8.62542 41.1798,-8.62335 41.1793,-8.62249 41.1773,-8.62257 41.1747,-8.6225 41.1724,-8.62325 41.1731,-8.62043
9	338	LINESTRING(-8.62839 41.1574,-8.62804 41.1576,-8.62624 41.1574,-8.6252 41.1572,-8.62513 41.1572,-8.62511 41.1564,-8.62492 41.1555,-8.62309 41.1558,-8.62123 41.1558,-8.61952 41.1553,-8.61943 41.1553,-8.61917 41.1553,-8.6176 41.155,-8.61565 41.1546,-8.61365 4
10	321	LINESTRING(-8.60178 41.1596,-8.60178 41.1596,-8.60179 41.1596,-8.6018 41.1596,-8.6018 41.1596,-8.60179 41.1596,-8.60112 41.1594,-8.6018 41.1577,-8.6027 41.1554,-8.60301 41.1547,-8.60445 41.1548,-8.60455 41.1548,-8.60476 41.1537,-8.60605 41.1535,-8.60615 41
11	378	LINESTRING(-8.61556 41.1407,-8.61522 41.1408,-8.61443 41.142,-8.61363 41.1432,-8.61182 41.1446,-8.6108 41.1456,-8.60989 41.1466,-8.60872 41.1474,-8.61046 41.1479,-8.6107 41.1479,-8.61106 41.1479,-8.61305 41.1482,-8.6225 41.1478,-8.62484 41.1485,-8.62501 41
12	324	LINESTRING(-8.61757 41.1462,-8.61728 41.145,-8.61575 41.1454,-8.61405 41.146,-8.61252 41.146,-8.61187 41.146,-8.61084 41.1457,-8.61088 41.1455,-8.61086 41.145,-8.61033 41.1434,-8.60863 41.1432,-8.60665 41.1423,-8.60523 41.1433,-8.60513 41.1436,-8.60561 41.
13	390	LINESTRING(-8.61376 41.1501,-8.61377 41.1501,-8.61378 41.1501,-8.61379 41.1501,-8.61378 41.1501,-8.61378 41.1501,-8.61378 41.1501,-8.61379 41.1501,-8.61378 41.1501,-8.61422 41.15,-8.61421 41.1501,-8.61428 41.15,-8.61433 41.15,-8.61428 41.15,-8.61427 41.15,
14	329	LINESTRING(-8.63027 41.1574,-8.63031 41.1574,-8.6299 41.1572,-8.62834 41.1573,-8.62798 41.1577,-8.62797 41.158,-8.62679 41.1589,-8.62467 41.1598,-8.62286 41.1604,-8.62123 41.1603,-8.62106 41.1613,-8.61972 41.162,-8.61817 41.1625,-8.61572 41.1623,-8.61309 4
15	362	LINESTRING(-8.58719 41.1465,-8.58714 41.1465,-8.58755 41.1468,-8.5866 41.1472,-8.58408 41.1461,-8.58172 41.1448,-8.57938 41.1459,-8.58004 41.1467,-8.57889 41.1491,-8.57997 41.152,-8.58263 41.1547,-8.58263 41.1583,-8.58111 41.1621,-8.58272 41.1658,-8.58696
16	377	LINESTRING(-8.60691 41.1457,-8.60706 41.1459,-8.6072 41.146,-8.60728 41.1457,-8.60707 41.1467,-8.60704 41.147,-8.60867 41.1473,-8.60954 41.1476,-8.61059 41.1479,-8.61061 41.1479,-8.61093 41.1482,-8.61032 41.1495,-8.60967 41.1512,-8.61 41.1525,-8.60946 41.1
17	361	LINESTRING(-8.61085 41.1456,-8.61053 41.1461,-8.60905 41.1469,-8.6081 41.1483,-8.60715 41.1486,-8.60674 41.1484,-8.60633 41.1483,-8.60454 41.1478,-8.60336 41.1474,-8.60225 41.1472,-8.60227 41.1471,-8.60225 41.1472,-8.60224 41.1472,-8.60223 41.1471,-8.60036
18	374	LINESTRING(-8.61127 41.172,-8.61125 41.172,-8.61123 41.1715,-8.61092 41.1707,-8.60943 41.1699,-8.60869 41.1693,-8.60974 41.1685,-8.60938 41.1678,-8.60921 41.1669,-8.60927 41.1658,-8.60846 41.1651,-8.60849 41.1646,-8.60848 41.1646,-8.60848 41.1645,-8.60872
19	384	LINESTRING(-8.62102 41.1611,-8.62138 41.1605,-8.62112 41.1603,-8.62088 41.1614,-8.61918 41.1622,-8.61724 41.1624,-8.61661 41.1624,-8.61489 41.1621,-8.61271 41.1619,-8.6102 41.1616,-8.60742 41.161,-8.60458 41.1604,-8.60172 41.1596,-8.59918 41.1589,-8.59885
20	322	LINESTRING(-8.6073 41.1615,-8.60715 41.1615,-8.60713 41.1615,-8.60703 41.162,-8.60693 41.162,-8.60692 41.1621,-8.6078 41.1623,-8.60905 41.1624,-8.60937 41.1621,-8.60891 41.1613,-8.60656 41.1608,-8.60536 41.1606,-8.604 41.1602,-8.60204 41.1597,-8.60133 41.1
21	372	LINESTRING(-8.61986 41.148,-8.62 41.1479,-8.62073 41.1482,-8.62084 41.1495,-8.62102 41.1504,-8.62092 41.1521,-8.62092 41.1535,-8.62081 41.1553,-8.62057 41.1556,-8.61923 41.1554,-8.61793 41.1551,-8.61519 41.1546,-8.61347 41.1542,-8.61327 41.1539,-8.61304 41
22	330	LINESTRING(-8.61325 41.1544,-8.61328 41.1543,-8.61329 41.1543,-8.61329 41.1543,-8.61328 41.1543,-8.61324 41.1538,-8.61086 41.1536,-8.6082 41.1535,-8.60737 41.1534,-8.60607 41.1529,-8.60705 41.1519,-8.60694 41.151,-8.6069 41.151,-8.60685 41.151,-8.60617 41.
23	342	LINESTRING(-8.59457 41.1586,-8.59457 41.1586,-8.59456 41.1586,-8.59456 41.1586,-8.59456 41.1586,-8.59435 41.1586,-8.59384 41.1584,-8.59384 41.1578,-8.59439 41.1569,-8.59438 41.1568,-8.59439 41.1567,-8.59496 41.1558,-8.59558 41.1547,-8.59559 41.1547,-8.5958

Queries over Data Tables

2. **Start From** Query

Input parameters:

1. Query Region.
2. [Time1,Time2].




```
SELECT Trajectory ID, Trajectory FROM Trajectories
WHERE Start Time BETWEEN (Time1 AND Time2)
AND(Query Region CONTAINS Start Point).
```

Start From Query Example

- **SELECT** tripid,**ST_AsText**(trip) **AS** trajectory **FROM** **Trips** **WHERE** starttime **BETWEEN** '2013-07-01 05:00:00' **AND** '2013-07-01 09:00:00' **AND** **ST_Contains**(**ST_GeomFromText**('POLYGON((-8.624954223632814 41.1437074383407,-8.624954223632814 41.1597351586533, -8.586158752441408 41.1597351586533,-8.586158752441408 41.1437074383407,-8.624954223632814 41.1437074383407))',4326), startpoint);

Data Output | Explain | Messages | History

	tripid integer	trajectory text
1	330	LINESTRING(-8.61325 41.1544,-8.61328 41.1543,-8.61329 41.1543,-8.61329 41.1543,-8.61328 41.1543,-8.61324 41.1538,-8.61086 41.1536,-8.6082 41.1535,-8.60737 41.1534,-8.60607 41.1529,-8.60705 41.1519,-8.60694 41.151,-8.6069 41.151,-8.60685 41.151,-8.60617 41.
2	369	LINESTRING(-8.60657 41.1447,-8.60739 41.1454,-8.60712 41.1463,-8.60687 41.1468,-8.60687 41.1469,-8.60728 41.1471,-8.60855 41.1474,-8.6093 41.1476,-8.61085 41.148,-8.61309 41.1483,-8.62226 41.1477,-8.62484 41.1485,-8.62582 41.1502,-8.62608 41.152,-8.62607 4
3	382	LINESTRING(-8.61071 41.1455,-8.61077 41.1455,-8.6108 41.1457,-8.61051 41.146,-8.60934 41.1466,-8.60864 41.1471,-8.60878 41.1473,-8.60909 41.1476,-8.61081 41.1479,-8.61062 41.1489,-8.61038 41.1496,-8.60962 41.1511,-8.61016 41.1523,-8.60986 41.1546,-8.60962
4	308	LINESTRING(-8.60648 41.1446)
5	347	LINESTRING(-8.60635 41.1446,-8.60637 41.1446,-8.60668 41.1447,-8.60741 41.1457,-8.60702 41.1466,-8.60687 41.1468,-8.60744 41.1472,-8.6085 41.1474,-8.60975 41.1477,-8.61204 41.148,-8.62033 41.1476,-8.62074 41.1486,-8.6207 41.1487,-8.6207 41.1487,-8.6207 41.
6	342	LINESTRING(-8.59457 41.1586,-8.59457 41.1586,-8.59456 41.1586,-8.59456 41.1586,-8.59456 41.1586,-8.59456 41.1586,-8.59435 41.1586,-8.59384 41.1584,-8.59384 41.1578,-8.59439 41.1569,-8.59438 41.1568,-8.59439 41.1567,-8.59496 41.1558,-8.59558 41.1547,-8.59559 41.1547,-8.5958
7	336	LINESTRING(-8.62417 41.1562,-8.62417 41.1562,-8.62415 41.1562,-8.62393 41.1569,-8.62353 41.1583,-8.62256 41.1581,-8.62274 41.1569,-8.62303 41.1559,-8.62302 41.156,-8.62303 41.156,-8.62298 41.1559,-8.62155 41.156,-8.62035 41.1555,-8.61827 41.1551,-8.61569 4
8	359	LINESTRING(-8.61088 41.1456,-8.6108 41.1459,-8.60951 41.1467,-8.60874 41.1472,-8.60861 41.1473,-8.60805 41.1484,-8.60686 41.1485,-8.60655 41.1484,-8.60548 41.1481,-8.60383 41.1474,-8.60378 41.1466,-8.60381 41.1466,-8.6045 41.1461,-8.60452 41.1453,-8.60483
9	321	LINESTRING(-8.60178 41.1596,-8.60178 41.1596,-8.60179 41.1596,-8.6018 41.1596,-8.6018 41.1596,-8.60179 41.1596,-8.60112 41.1594,-8.6018 41.1577,-8.6027 41.1554,-8.60301 41.1547,-8.60445 41.1548,-8.60455 41.1548,-8.60476 41.1537,-8.60605 41.1535,-8.60615 41
10	389	LINESTRING(-8.60822 41.1569,-8.6082 41.1569,-8.6084 41.1569,-8.60902 41.1568,-8.60897 41.1577,-8.60828 41.1592,-8.60811 41.1601,-8.60952 41.1603,-8.60982 41.1604,-8.60999 41.159,-8.61001 41.1574,-8.61 41.156,-8.60995 41.1543,-8.60989 41.1521,-8.61077 41.15
11	344	LINESTRING(-8.60542 41.1443,-8.60541 41.1443,-8.60542 41.1443,-8.60543 41.1443,-8.60544 41.1443,-8.60544 41.1443,-8.60544 41.1443,-8.60544 41.1443,-8.60544 41.1443,-8.60544 41.1443,-8.60543 41.1443,-8.60542 41.1443,-8.60541 41.1443,-8.60541 41.1443,-8.60543 41.1443,-8.6054
12	324	LINESTRING(-8.61757 41.1462,-8.61728 41.145,-8.61575 41.1454,-8.61405 41.146,-8.61252 41.146,-8.61187 41.146,-8.61084 41.1457,-8.61088 41.1455,-8.61086 41.145,-8.61033 41.1434,-8.60863 41.1432,-8.60665 41.1423,-8.60523 41.1433,-8.60513 41.1436,-8.60561 41.
13	390	LINESTRING(-8.61376 41.1501,-8.61377 41.1501,-8.61378 41.1501,-8.61379 41.1501,-8.61378 41.1501,-8.61378 41.1501,-8.61378 41.1501,-8.61379 41.1501,-8.61378 41.1501,-8.61422 41.15,-8.61421 41.1501,-8.61428 41.15,-8.61433 41.15,-8.61428 41.15,-8.61427 41.15,
14	362	LINESTRING(-8.58719 41.1465,-8.58714 41.1465,-8.58755 41.1468,-8.5866 41.1472,-8.58408 41.1461,-8.58172 41.1448,-8.57938 41.1459,-8.58004 41.1467,-8.57889 41.1491,-8.57997 41.152,-8.58263 41.1547,-8.58263 41.1583,-8.58111 41.1621,-8.58272 41.1658,-8.58696
15	377	LINESTRING(-8.60691 41.1457,-8.60706 41.1459,-8.6072 41.146,-8.60728 41.1457,-8.60707 41.1467,-8.60704 41.147,-8.60867 41.1473,-8.60954 41.1476,-8.61059 41.1479,-8.61061 41.1479,-8.61093 41.1482,-8.61032 41.1495,-8.60967 41.1512,-8.61 41.1525,-8.60946 41.1
16	361	LINESTRING(-8.61085 41.1456,-8.61053 41.1461,-8.60905 41.1469,-8.6081 41.1483,-8.60715 41.1486,-8.60674 41.1484,-8.60633 41.1483,-8.60454 41.1478,-8.60336 41.1474,-8.60225 41.1472,-8.60227 41.1471,-8.60225 41.1472,-8.60224 41.1472,-8.60223 41.1471,-8.60036
17	372	LINESTRING(-8.61986 41.148,-8.62 41.1479,-8.62073 41.1482,-8.62084 41.1495,-8.62102 41.1504,-8.62092 41.1521,-8.62092 41.1535,-8.62081 41.1553,-8.62057 41.1556,-8.61923 41.1554,-8.61793 41.1551,-8.61519 41.1546,-8.61347 41.1542,-8.61327 41.1539,-8.61304 41

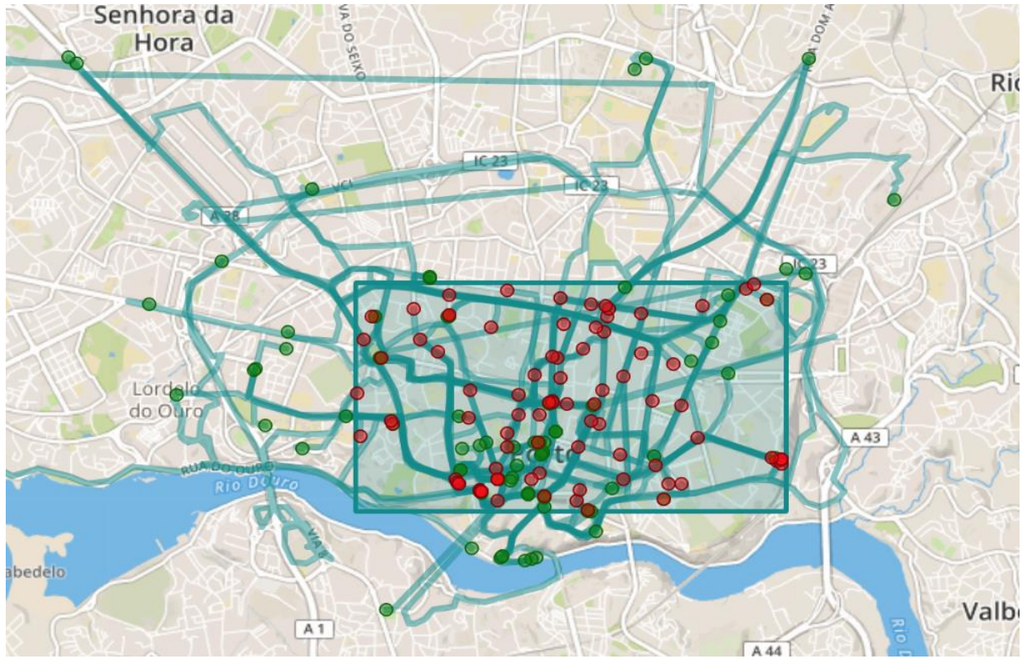
Queries over Data Tables

3. **End At** Query

Input parameters:

1. Query Region.
2. [Time1,Time2].

```
SELECT Trajectory ID, Trajectory FROM Trajectories
WHERE End Time BETWEEN (Time1 AND Time2)
AND(Query Region CONTAINS End Point).
```



End At Query Example

- **SELECT** tripid,**ST_AsText**(trip) **AS** trajectory **FROM Trips WHERE** endtime **BETWEEN** '2013-07-01 05:00:00' **AND** '2013-07-01 09:00:00’ **AND ST_Contains**(**ST_GeomFromText**('POLYGON((-8.624954223632814 41.1437074383407,-8.624954223632814 41.1597351586533, -8.586158752441408 41.1597351586533,-8.586158752441408 41.1437074383407,-8.624954223632814 41.1437074383407))',4326), endpoint);

Data Output | Explain | Messages | History

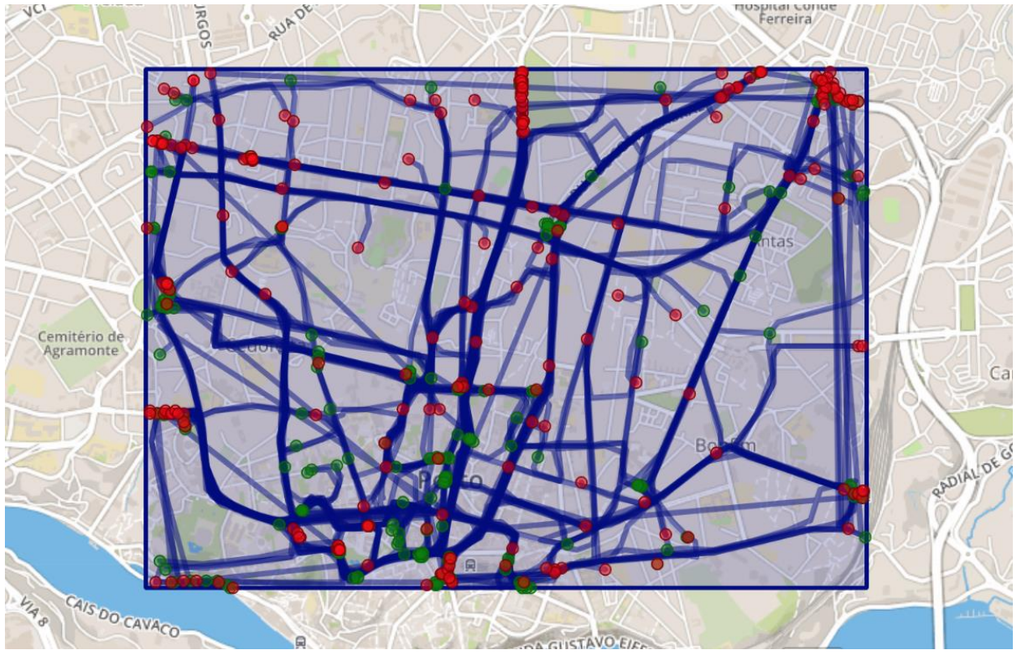
	tripid integer	trajectory text
1	321	LINESTRING(-8.60178 41.1596,-8.60178 41.1596,-8.60179 41.1596,-8.6018 41.1596,-8.6018 41.1596,-8.60179 41.1596,-8.60112 41.1594,-8.6018 41.1577,-8.6027 41.1554,-8.60301 41.1547,-8.60445 41.1548,-8.60455 41.1548,-8.60476 41.1537,-8.60605 41.1535,-8.60615 41
2	296	LINESTRING(-8.58264 41.1806,-8.58264 41.1805,-8.58226 41.1803,-8.58253 41.1788,-8.58327 41.177,-8.58424 41.1751,-8.58521 41.1731,-8.58653 41.1705,-8.58793 41.1679,-8.58898 41.1653,-8.59068 41.1628,-8.59233 41.1604,-8.59288 41.1595,-8.59327 41.1588,-8.59408
3	308	LINESTRING(-8.60648 41.1446)
4	324	LINESTRING(-8.61757 41.1462,-8.61728 41.145,-8.61575 41.1454,-8.61405 41.146,-8.61252 41.146,-8.61187 41.146,-8.61084 41.1457,-8.61088 41.1455,-8.61086 41.145,-8.61033 41.1434,-8.60863 41.1432,-8.60665 41.1423,-8.60523 41.1433,-8.60513 41.1436,-8.60561 41.
5	390	LINESTRING(-8.61376 41.1501,-8.61377 41.1501,-8.61378 41.1501,-8.61379 41.1501,-8.61378 41.1501,-8.61378 41.1501,-8.61378 41.1501,-8.61379 41.1501,-8.61378 41.1501,-8.61422 41.15,-8.61421 41.1501,-8.61428 41.15,-8.61433 41.15,-8.61428 41.15,-8.61427 41.15,
6	304	LINESTRING(-8.60271 41.1835,-8.59997 41.1827,-8.5973 41.1819,-8.59823 41.18,-8.59945 41.1786,-8.60202 41.1793,-8.60474 41.1801,-8.60514 41.1802,-8.60613 41.1788,-8.608 41.177,-8.60845 41.1755,-8.6068 41.1749,-8.60589 41.1737,-8.6059 41.1737,-8.60536 41.172
7	361	LINESTRING(-8.61085 41.1456,-8.61053 41.1461,-8.60905 41.1469,-8.6081 41.1483,-8.60715 41.1486,-8.60674 41.1484,-8.60633 41.1483,-8.60454 41.1478,-8.60336 41.1474,-8.60225 41.1472,-8.60227 41.1471,-8.60225 41.1472,-8.60224 41.1472,-8.60223 41.1471,-8.60036

Queries over Data Tables

4. **Contain** Query

Input parameters:

1. Query Region.
2. [Time1,Time2].




```
SELECT Trajectory ID, Trajectory FROM Trajectories
WHERE Start Time BETWEEN (Time1 AND Time2)
AND(Query Region CONTAINS Trajectory).
```

Contain Query Example

- **SELECT** tripid,**ST_AsText**(trip) **AS** trajectory **FROM Trips WHERE** starttime **BETWEEN** '2013-07-01 05:00:00' **AND** '2013-07-01 09:00:00' **AND ST_Contains**(**ST_GeomFromText**('POLYGON((-8.624954223632814 41.1437074383407,-8.624954223632814 41.1597351586533, -8.586158752441408 41.1597351586533,-8.586158752441408 41.1437074383407,-8.624954223632814 41.1437074383407))',4326), trip);

Data Output | Explain | Messages | History

	tripid integer	trajectory text
1	330	LINESTRING(-8.61325 41.1544,-8.61328 41.1543,-8.61329 41.1543,-8.61329 41.1543,-8.61328 41.1543,-8.61324 41.1538,-8.61086 41.1536,-8.6082 41.1535,-8.60737 41.1534,-8.60607 41.1529,-8.60705 41.1519,-8.60694 41.151,-8.6069 41.151,-8.60685 41.151,-8.60617 41.
2	369	LINESTRING(-8.60657 41.1447,-8.60739 41.1454,-8.60712 41.1463,-8.60687 41.1468,-8.60687 41.1469,-8.60728 41.1471,-8.60855 41.1474,-8.6093 41.1476,-8.61085 41.148,-8.61309 41.1483,-8.62226 41.1477,-8.62484 41.1485,-8.62582 41.1502,-8.62608 41.152,-8.62607 4
3	382	LINESTRING(-8.61071 41.1455,-8.61077 41.1455,-8.6108 41.1457,-8.61051 41.146,-8.60934 41.1466,-8.60864 41.1471,-8.60878 41.1473,-8.60909 41.1476,-8.61081 41.1479,-8.61062 41.1489,-8.61038 41.1496,-8.60962 41.1511,-8.61016 41.1523,-8.60986 41.1546,-8.60962
4	308	LINESTRING(-8.60648 41.1446)
5	347	LINESTRING(-8.60635 41.1446,-8.60637 41.1446,-8.60668 41.1447,-8.60741 41.1457,-8.60702 41.1466,-8.60687 41.1468,-8.60744 41.1472,-8.6085 41.1474,-8.60975 41.1477,-8.61204 41.148,-8.62033 41.1476,-8.62074 41.1486,-8.6207 41.1487,-8.6207 41.1487,-8.6207 41.
6	342	LINESTRING(-8.59457 41.1586,-8.59457 41.1586,-8.59456 41.1586,-8.59456 41.1586,-8.59456 41.1586,-8.59435 41.1586,-8.59384 41.1584,-8.59384 41.1578,-8.59439 41.1569,-8.59438 41.1568,-8.59439 41.1567,-8.59496 41.1558,-8.59558 41.1547,-8.59559 41.1547,-8.5958
7	336	LINESTRING(-8.62417 41.1562,-8.62417 41.1562,-8.62415 41.1562,-8.62393 41.1569,-8.62353 41.1583,-8.62256 41.1581,-8.62274 41.1569,-8.62303 41.1559,-8.62302 41.156,-8.62303 41.156,-8.62298 41.1559,-8.62155 41.156,-8.62035 41.1555,-8.61827 41.1551,-8.61569 4
8	359	LINESTRING(-8.61088 41.1456,-8.6108 41.1459,-8.60951 41.1467,-8.60874 41.1472,-8.60861 41.1473,-8.60805 41.1484,-8.60686 41.1485,-8.60655 41.1484,-8.60548 41.1481,-8.60383 41.1474,-8.60378 41.1466,-8.60381 41.1466,-8.6045 41.1461,-8.60452 41.1453,-8.60483

Queries over Data Tables

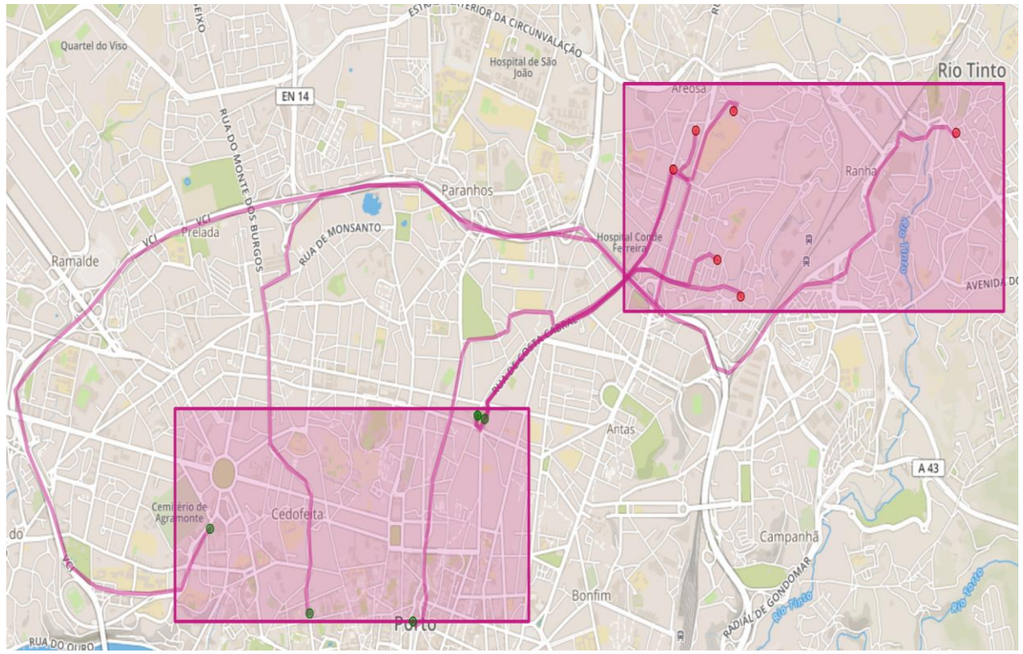
5. Joint Query

Combine query conditions

For example: Trajectories **start at** Region1 **and end at** Region2

Input parameters:

1. Query Region1 &Region2
2. [Time1,Time2].



Aggregation Queries

- Temporal aggregation: grouping trajectory data on
  - Week days.
  - Day hours.
- Spatial aggregation: grouping trajectory data on
  - Roads.
  - POIs.
  - Regions.

Temporal Aggregation Queries

- Week days aggregation with Pass query.
- Retrieve the number of trips in each week day.

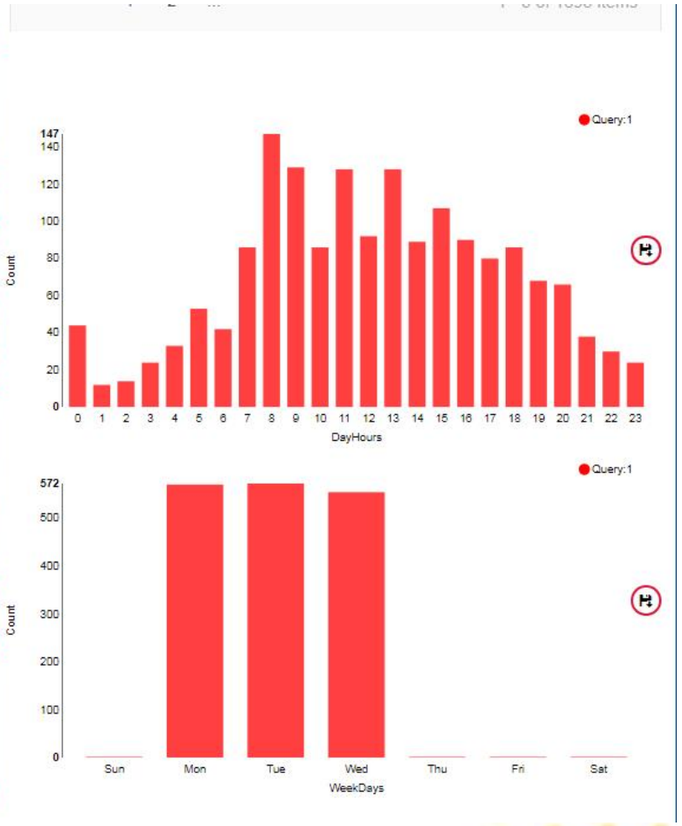
```
SELECT startday, COUNT(*) FROM
(
    SELECT tripid, EXTRACT(DOW FROM starttime) AS startday FROM Trips WHERE starttime BETWEEN
    '2013-06-16 00:00:00' AND '2013-06-18 23:59:59' AND
    ST_Intersects(ST_GeomFromText('POLYGON((-8.624954223632814 41.1437074383407,-
    8.624954223632814 41.1597351586533,-8.586158752441408 41.1597351586533,-8.586158752441408
    41.1437074383407,-8.624954223632814 41.1437074383407))',4326),trip);
) GROUP BY startday ORDER BY startday";
```

Temporal Aggregation Queries

- Day hours aggregation with Pass query.
- Retrieve the number of trips in each hour in a day.

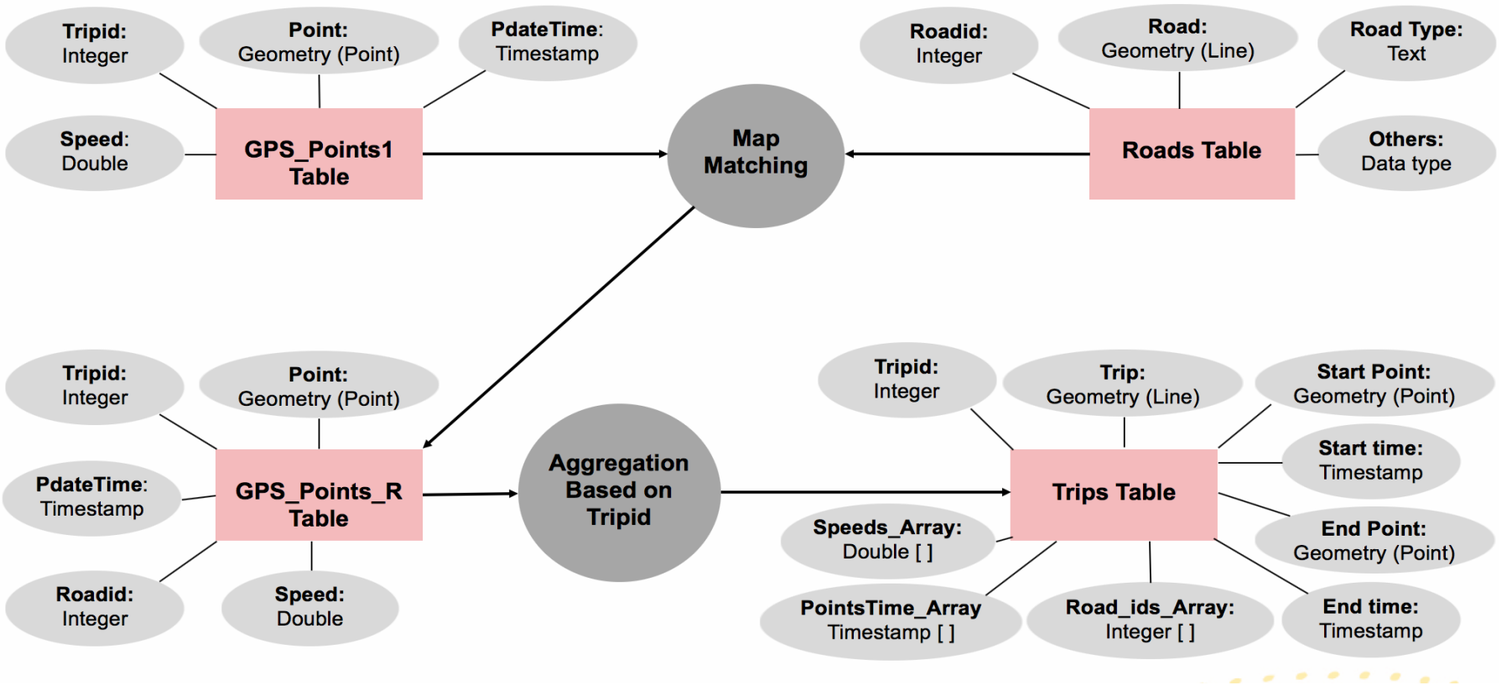
```
SELECT starthour, COUNT(*) FROM
(
    SELECT tripid, EXTRACT(Hour FROM starttime) AS starthour FROM Trips WHERE starttime BETWEEN
    '2013-06-16 00:00:00' AND '2013-06-18 23:59:59' AND
    ST_Intersects(ST_GeomFromText('POLYGON((-8.624954223632814 41.1437074383407,-
    8.624954223632814 41.1597351586533,-8.586158752441408 41.1597351586533,-8.586158752441408
    41.1437074383407,-8.624954223632814 41.1437074383407))',4326),trip);
) GROUP BY starthour ORDER BY starthour ";
```

Temporal Aggregation Result



Create Tables with Road/Region Information

- Create data tables based on map-matching results over roads (and regions)



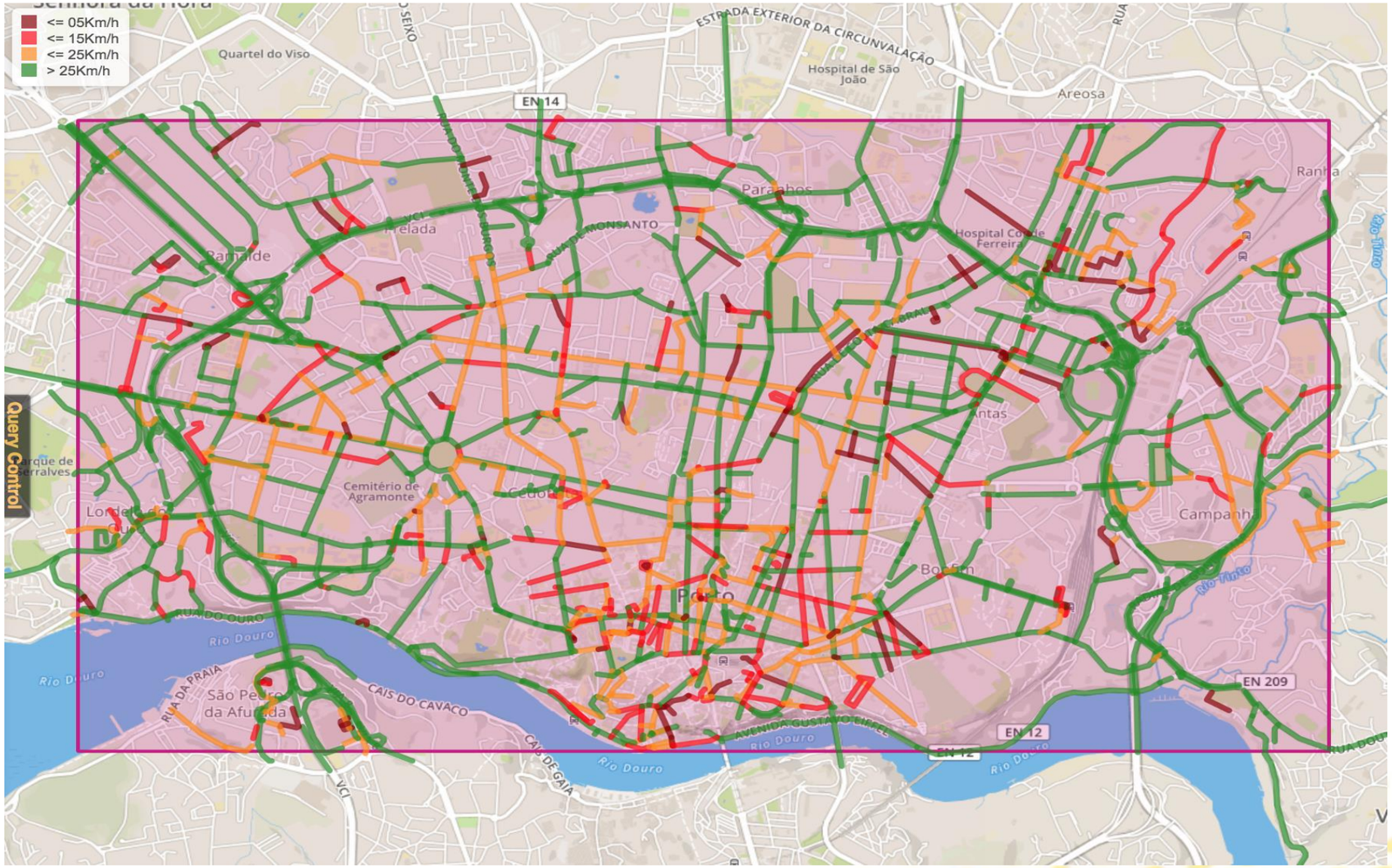
Spatial Aggregation Queries

Retrieve aggregated attributes on roads

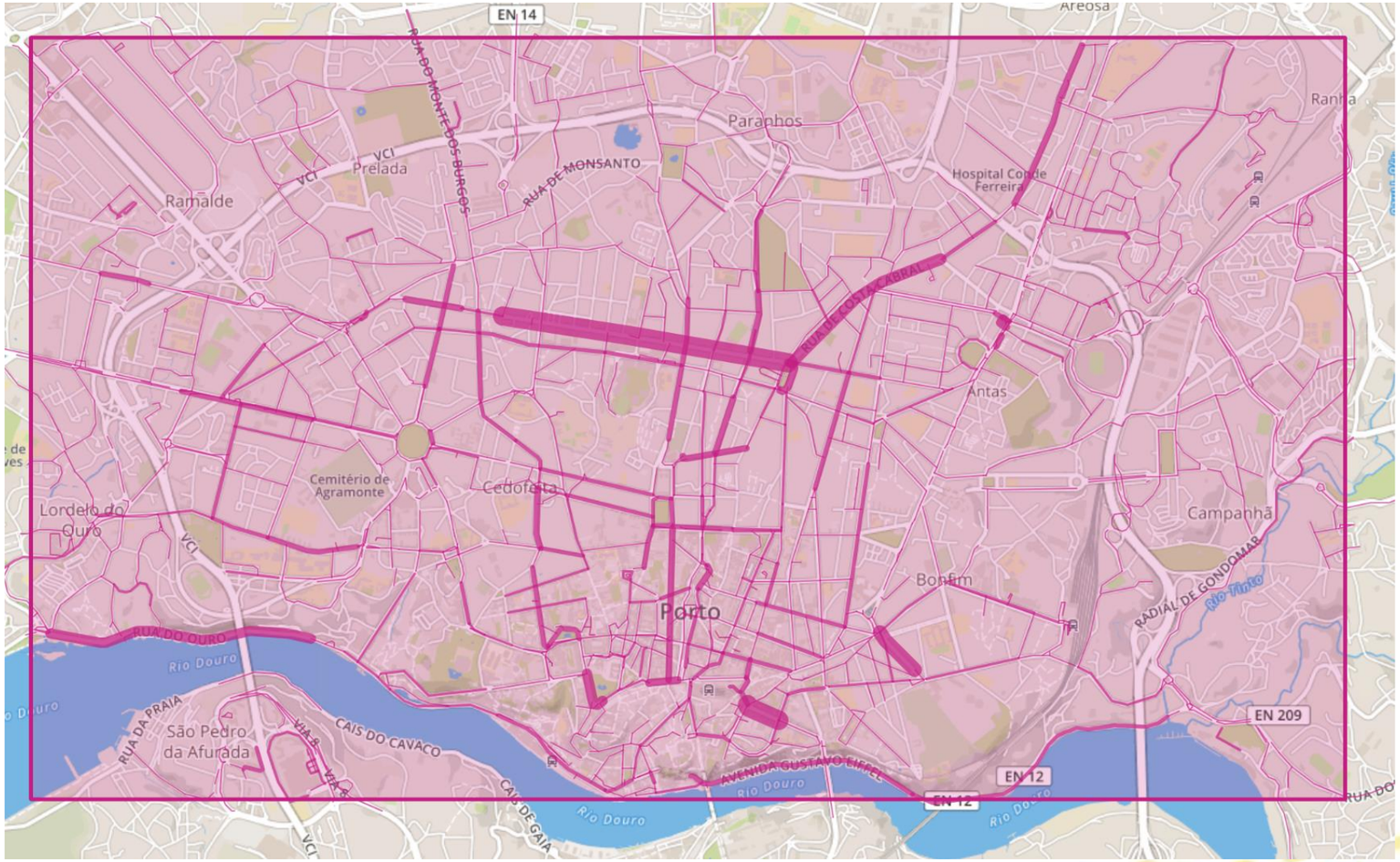
- Counts of trajectories
- Average speeds (max, min, etc.)

```
SELECT road_id, COUNT(*) AS count, AVG(speed) AS average_speed FROM
(
        SELECT unnest(road_ids_array) AS road_id, unnest(speeds_array) AS speed
        FROM Trips WHERE starttime BETWEEN '2013-07-01 05:00:00' AND '2013-07-01 09:00:00' AND
        ST_Contains(ST_GeomFromText('POLYGON((-8.624954223632814 41.1437074383407,-
        8.624954223632814 41.1597351586533, -8.586158752441408 41.1597351586533,-8.586158752441408
        41.1437074383407,-8.624954223632814 41.1437074383407))',4326), trip);
) GROUP BY road_id
```

Traffic Speed on Roads



Counts of Taxi Trips on Roads



Data Performance and Issues

- Data aggregation on the fly
  - Easy implementation
  - Slow when the amount of trajectories is large
- Potential solution
  - Precomputing aggregations
  - Create caching structures (e.g. data cubes) in database

Thanks!