Multimodel Data Processing with SAP HANA

Prof. Dr. Ingo Claßen

Dienstag, 7. September 2021 13:20

- Verbindung von Geo- und Graphdatenverarbeitung
- Geo-Visualisierungen über Python/Leaflet in Jupyter Notebooks
- OpenStreetMap Daten von London
- GraphScript-Prozeduren
- Basiert auf https://github.com/SAP-samples/teched2020-DAT260

LONDON_POI (90202 Datensätze)

osmid	nvarchar	PK, OpenStreetMap ID
name	nvarchar	Name des POI
amenity	nvarchar	Schule, Bar, …
geometry_GEO	ST_GEOMETRY	Punkt oder Polygon in SRS 4326
SHAPE	ST_GEOMETRY	Punkt oder Polygon in SRS 32630 (planar)

```
select "amenity", count(*)
from adbkt.LONDON_POI
group by "amenity";
```

LONDON_Vertices (793.834 Datensätze)

osmid	BIGINT	PK, OpenStreetMap ID
highway	nvarchar	Art des Punktes (z.B. Kreuzung)
geometry_GEO	ST_GEOMETRY	Punkt SRS 4326
SHAPE	ST_GEOMETRY	Punkt SRS 32630 (planar)

```
select "highway", count(*)
from adbkt.LONDON_VERTICES
group by "highway";
```

LONDON_EDGES (1.574.532 Datensätze)

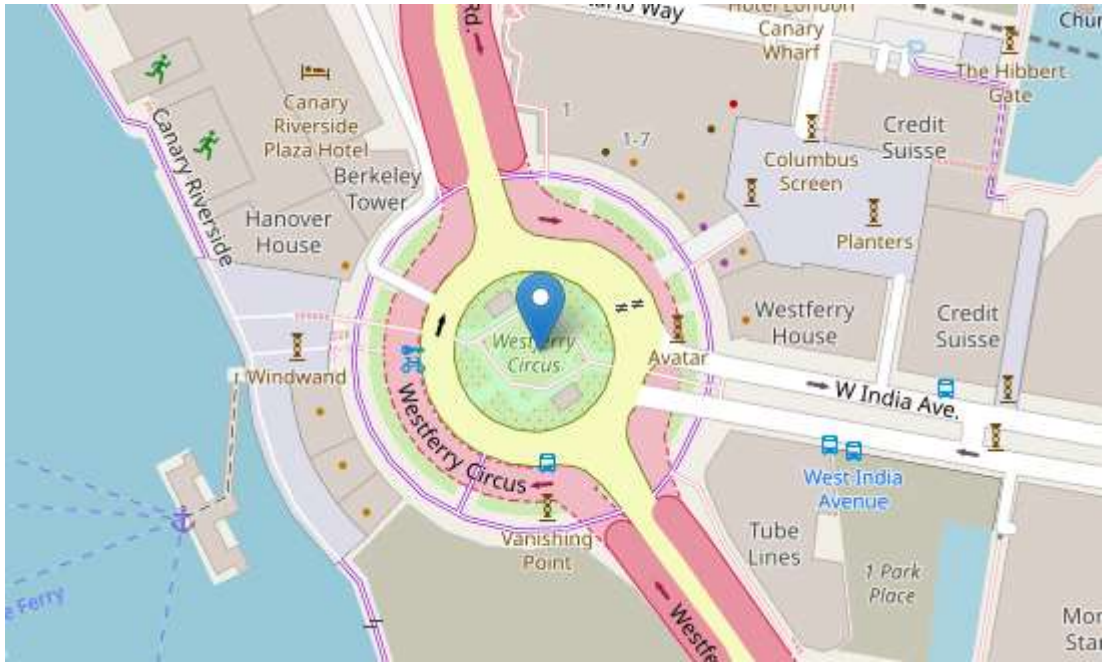
ID	nvarchar	PK
SOURCE	BIGINT	FK nach LONDON_Vertices
TARGET	BIGINT	FK nach LONDON_Vertices
osmid	nvarchar	OpenStreetMap ID
length	double	Länge in Meter
highway	nvarchar	Art der Verbindung (z.B. Straße)
maxspeed	nvarchar	Maximale Geschwindigkeit
geometry_GEO	ST_GEOMETRY	Punkt SRS 4326
SHAPE	ST_GEOMETRY	Punkt SRS 32630 (planar)

Geo-Visualisierung

Prof. Dr. Ingo Claßen

Dienstag, 7. September 2021 14:41

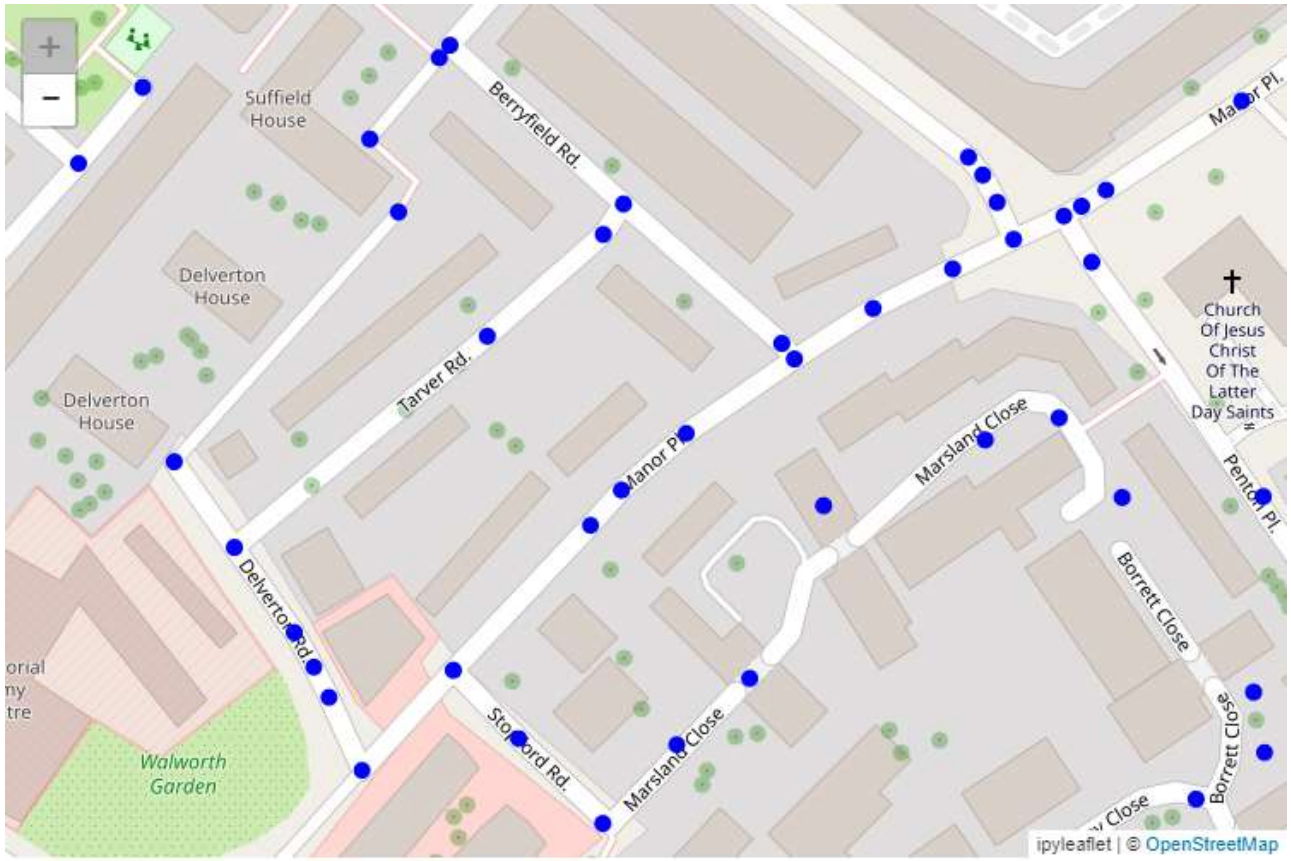
```
import pandas as pd
from hana_ml import ConnectionContext
from ipywidgets import Layout
from ipyleaflet import Map, basemaps, Marker, DivIcon, WKTLayer
import cred
latlng_london = (51.5370654, -0.1412396)


def div_marker(lat, lng, text):
    icon = DivIcon(html=f'{text}')
    marker = Marker(location=(lat, lng), icon=icon)
    return marker

def wkt_from_hdf(hdf, geo_col, srid, diag=False):
    hdf1 = hdf.select(


(f'ST_CollectAggr({geo_col}).ST_Transform({srid}).ST_AsWKT()',
      'wkt'))
    wkt_string = hdf1.collect().wkt.values[0]
    wkt_layer = WKTLayer(
      wkt_string=wkt_string,
      hover_style={"fillColor": "red"})
    return wkt_layer

defaultLayout=Layout(width='750px', height='750px')
m = Map(
    center=latlng_london,
    zoom=12,
    basemap=basemaps.OpenStreetMap.DE,
    layout=defaultLayout
)
m
```




```
sql = '''
select ST_GeomFromText('POINT(-0.026859 51.505748)', 4326) as geo
from DUMMY;
'''
m.remove_layer(map_layer)
with ConnectionContext(cred.host, cred.port, cred.user, cred.password) as cc:
    hdf = cc.sql(sql)
    map_layer = wkt_from_hdf(hdf, 'geo', 4326)
    m.add_layer(map_layer)
```

Arbeitsstelle

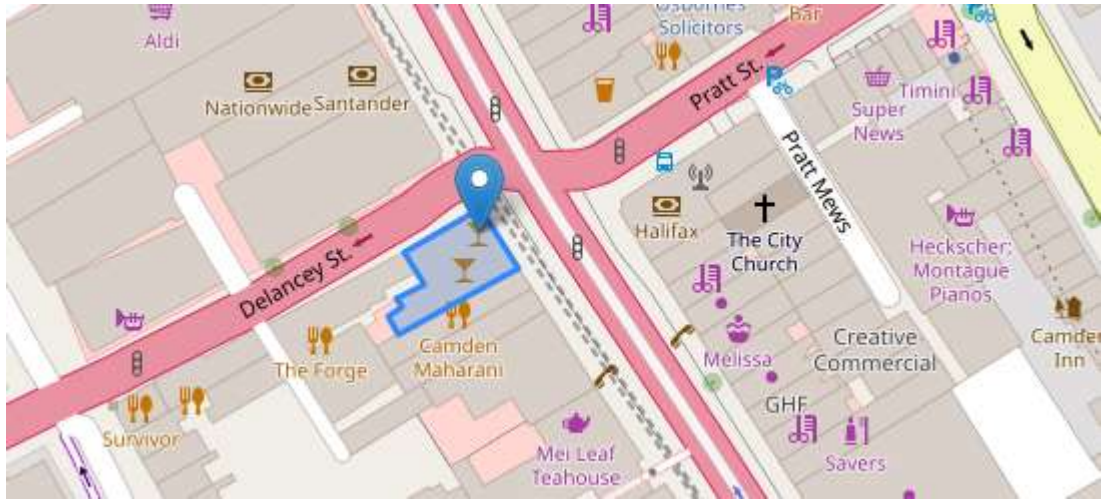
```
with center as (
  select
   ST_EnvelopeAggr(SHAPE)
   .ST_Centroid() as c
  from adbkt.LONDON_POI
)
select
 lv.shape.st_transform(4326) as geo
from adbkt.LONDON_VERTICES lv, center
where center.c.st_distance(lv.shape)<200;
```



```
select "osmid", to_varchar(SHAPE.ST_Transform(4326).ST_AsWKT()) as geo
from adbkt.LONDON_POI lp
where lower("name") like'%blues kitchen%' and "amenity" = 'bar';
```

*	osmid	GEO
1	6274057185	POINT (-0.14123969996928326 51.53706540003013)
2	457644936	POLYGON ((-0.14113280004710366 51.53700960000624,-0.14123339995
3	165488538	POLYGON ((-0.08026889999320642 51.52657160002017,-0.08026000003
4	321923639	POLYGON ((-0.11903039998323908 51.46045399998399,-0.11915050003

```
select shape as geo
from adbkt.LONDON_POI lp
where
  "osmid"='6274057185' or
  "osmid"='457644936';
```



Bar

Zwischen Westrferry Circus und Blues Kitchen

```
with points as (
  select
    ST_GeomFromText(
    'POINT (706327.107445 5710259.94449)', 32630
   ) AS START_PT, SHAPE asTARGET_PT
  from adbkt.LONDON_POI lp
  where "osmid" = 6274057185
)
select ST_MakeLine(START_PT, TARGET_PT) AS geo
from points
union
select
 ST_MakeLine(START_PT,TARGET_PT)
 .ST_LineInterpolatePoint(0.5) AS geo
from points
union
select
 ST_MakeLine(START_PT,TARGET_PT)
 .ST_LineInterpolatePoint(0.5)
 .ST_Buffer(4835) AS geo
from
points
```

Kennzeichnung aller Knoten im Bereich

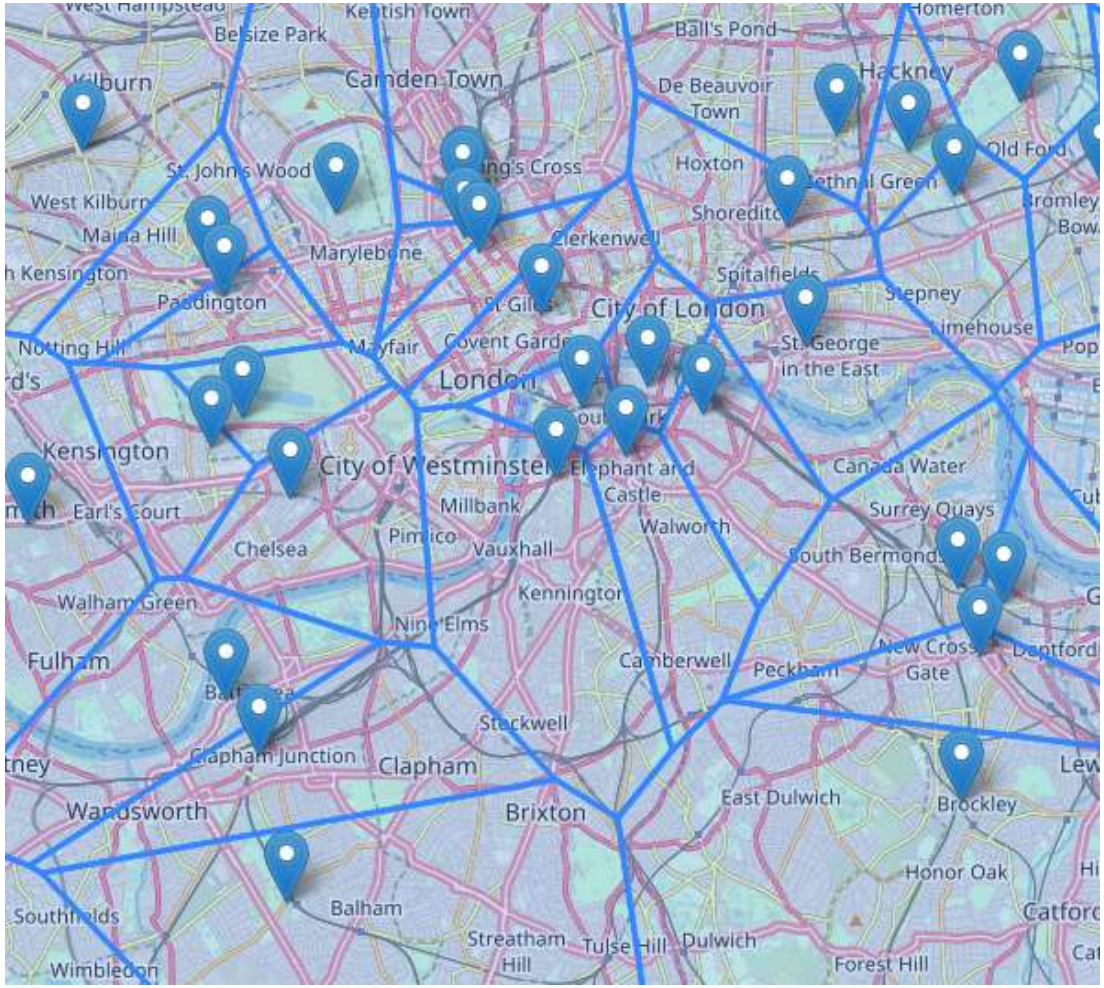
Prof. Dr. Ingo Claßen

Mittwoch, 8. September 2021 10:29

```
merge into LONDON_VERTICES lv
using
(
  select
    ST_MakeLine(
        ST_GeomFromText(
          'POINT (706327.107445 5710259.94449)',
          32630),
        SHAPE
    )
    .ST_LineInterpolatePoint(0.5)
    .ST_Buffer(5000) AS AREA
    from LONDON_POI
    where "osmid" = 6274057185
) circle on 1=1
when matched then update
  set lv.IN_SCOPE =
   CIRCLE.AREA.ST_Intersects(SHAPE);
```


```
    select shape as geo
    from adbkt.LONDON_VERTICES
    where IN_SCOPE = 1 LIMIT 1000;
```

```
select shape as geo
from adbkt.LONDON_POI
where "amenity" = 'bicycle_repair_station'
union
select
    ST_VoronoiCell(SHAPE, 10.0) OVER () as geo
from adbkt.LONDON_POI
where
 "amenity" like 'bicycle_repair_station';
```



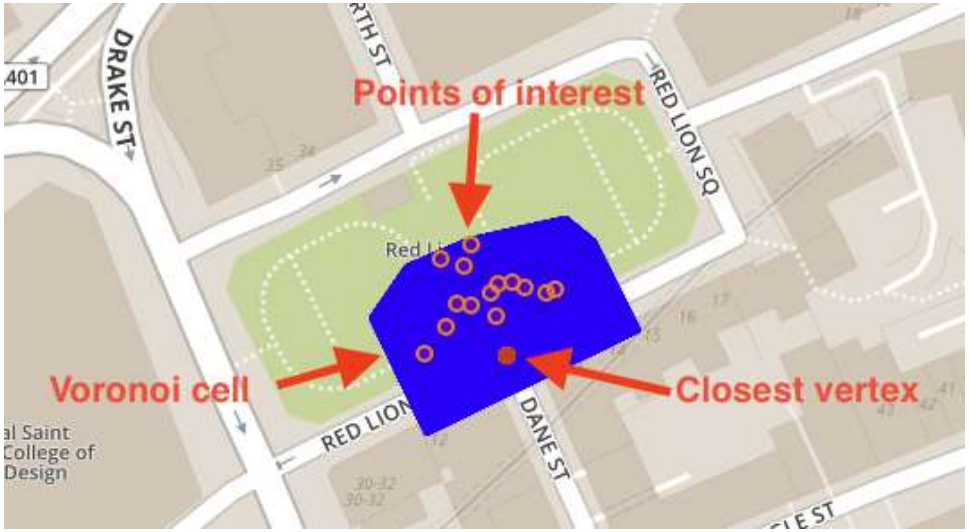

```
merge into LONDON_VERTICES
using
(
    select "osmid", ST_VoronoiCell(shape, 10.0) OVER () as CELL
    from LONDON_VERTICES
) v on LONDON_VERTICES."osmid" = v."osmid"
when matched then update
 set LONDON_VERTICES.VORONOI_CELL = v.CELL;
```

Verbindung POI mit Knoten im Wegenetz

Prof. Dr. Ingo Claßen

Mittwoch, 8. September 2021 12:02

```
update LONDON_POI
set SHAPE_CENTROID =
    case
        when SHAPE.ST_GeometryType() = 'ST_Point'
        then SHAPE
        else SHAPE.ST_Centroid()
    end;


merge into LONDON_POI lp
using LONDON_VERTICES lv on lv.VORONOI_CELL.ST_Intersects(lp.SHAPE_CENTROID) = 1
when matched then update
 set lp.VERTEX_OSMID = lv."osmid";
```

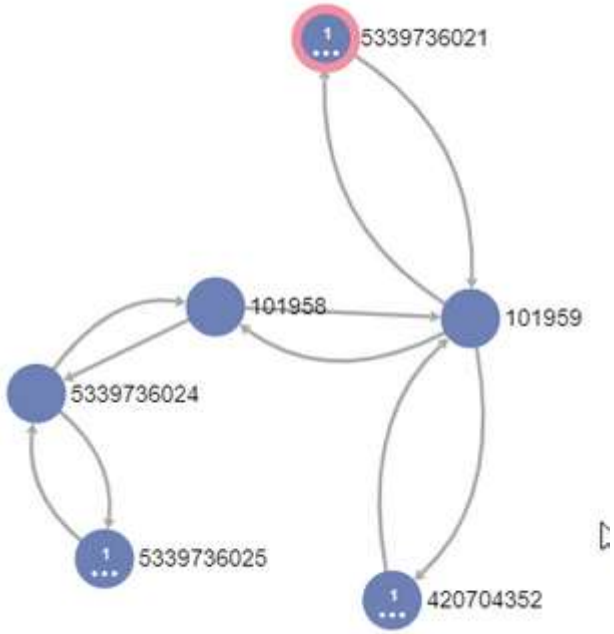


Create a Graph Workspace

Prof. Dr. Ingo Claßen

Mittwoch, 8. September 2021 12:06

```
CREATE GRAPH WORKSPACE "LONDON_GRAPH"
     EDGE TABLE adbkt.LONDON_EDGES
          SOURCE COLUMN "SOURCE"
          TARGET COLUMN "TARGET"
          KEY COLUMN "ID"
     VERTEX TABLE adbkt.LONDON_VERTICES
          KEY COLUMN "osmid";
```



```
CREATE TYPE TT_SPOO_EDGES AS TABLE (
  "ID" NVARCHAR(5000), "SOURCE" BIGINT, "TARGET" BIGINT, "EDGE_ORDER" BIGINT, "length" DOUBLE);


@delimiter ++;
CREATE OR REPLACE PROCEDURE "GS_SPOO"(
     IN i_startVertex BIGINT,
     IN i_endVertex BIGINT,
     IN i_direction VARCHAR(10),   -- OUTGOING (default), INCOMING, ANY
     OUT o_edges TT_SPOO_EDGES
     )
LANGUAGE GRAPH READS SQL DATA AS BEGIN
  GRAPH g_all = Graph("LONDON_GRAPH");

  GRAPH g = SubGraph(:g_all, v IN Vertices(:g_all) WHERE :v."IN_SCOPE" == 1);

  VERTEX v_start = Vertex(:g, :i_startVertex);
  VERTEX v_end   = Vertex(:g, :i_endVertex);

  WeightedPath<BIGINT> p = Shortest_Path(:g, :v_start, :v_end, :i_direction);
  o_edges = SELECT :e."ID", :e."SOURCE", :e."TARGET", :EDGE_ORDER, :e."length"
            FOREACH e IN Edges(:p) WITH ORDINALITY AS EDGE_ORDER;
END;
++
@delimiter ;++
```

Ausführung der Berechnung

Prof. Dr. Ingo Claßen

Mittwoch, 8. September 2021 12:16

```
CALL "GS_SPOO"(
  i_startVertex => 1433737988,
  i_endVertex => 1794145673,
  i_direction => 'ANY',
  o_path_length => ?,
  o_edges => ?
);

-- oder in Kurzform

CALL "GS_SPOO"(1433737988, 1794145673, 'ANY', ?, ?);
```

Log | 1: CALL "GS_SPOO"(i_startVertex ... [464]

*	ID	SOURCE	TARGET	EDGE_ORDER	length
1	1433737988-1433738110-0	1433737988	1433738110	1	7.165
2	1433738110-6619046707-0	1433738110	6619046707	2	2.268
3	6619046707-3762571738-0	6619046707	3762571738	3	30.391
4	3762571738-95956791-0	3762571738	95956791	4	15.97
5	95956791-3762571736-0	95956791	3762571736	5	20.404
6	3762571736-95956792-0	3762571736	95956792	6	7.386
7	95956792-453053943-0	95956792	453053943	7	9.846
8	453053943-1433738038-0	453053943	1433738038	8	33.179
9	1433738038-1433738127-0	1433738038	1433738127	9	20.252
10	1433738127-27084462-0	1433738127	27084462	10	11.176
11	27084462-107227-0	27084462	107227	11	12.767
12	107227-1433738148-0	107227	1433738148	12	26.795
13	1433738148-1433738161-0	1433738148	1433738161	13	10.088
14	1433738161-1433738150-0	1433738161	1433738150	14	13.466
15	1433738150-1433738152-0	1433738150	1433738152	15	16.112
16	1433738152-1433738012-0	1433738152	1433738012	16	9.47
17	1433738012-1888959092-0	1433738012	1888959092	17	4.942
18	1888959092-5345938034-0	1888959092	5345938034	18	10.779
19	5345938034-1433738062-0	5345938034	1433738062	19	6.177
20	1433738062-2018469849-0	1433738062	2018469849	20	3.746
21	2018469849-107226-0	2018469849	107226	21	18.704

```
SELECT VERTEX_OSMID
FROM "LONDON_POI"
WHERE "name" = 'Blues Kitchen' AND "osmid" = 6274057185;
```

*	VERTEX_OSMID
1	1794145673

```
update "LONDON_EDGES"
  set "SPEED_MPH" = TO_INT(REPLACE("maxspeed", ' mph', ''))
  where REPLACE("maxspeed", ' mph', '') <> "maxspeed" ;

select "SPEED_MPH", COUNT(*) as C from "LONDON_EDGES" group by "SPEED_MPH" order by C desc;

update "LONDON_EDGES" set "SPEED_MPH" = 30 where "SPEED_MPH" is null;
```

*	SPEED_MPH	C
1	(null)	906156
2	20	455414
3	30	182843
4	40	14766
5	5	4639
6	50	4015
7	10	3539
8	15	1082
9	60	934
10	70	572
11	12	526
12	4	36
13	25	10

```
CREATE TYPE "TT_SPOO_WEIGHTED_EDGES" AS TABLE (
  "ID" NVARCHAR(5000), "SOURCE" BIGINT, "TARGET" BIGINT, "EDGE_ORDER" BIGINT, "length" DOUBLE, "SPEED_MPH" INT);

@delimiter ++;
CREATE OR REPLACE PROCEDURE "GS_SPOO_WEIGHTED"(
      IN i_startVertex BIGINT,
      IN i_endVertex BIGINT,
      IN i_direction VARCHAR(10),   -- OUTGOING (default), INCOMING, ANY
      OUT o_edges TT_SPOO_WEIGHTED_EDGES
      )
LANGUAGE GRAPH READS SQL DATA AS BEGIN
  GRAPH g_all = Graph("LONDON_GRAPH");

  GRAPH g = SubGraph(:g_all, v IN Vertices(:g_all) WHERE :v."IN_SCOPE" == 1);


  VERTEX v_start = Vertex(:g, :i_startVertex);
  VERTEX v_end   = Vertex(:g, :i_endVertex);

  WeightedPath<DOUBLE> p = Shortest_Path(:g, :v_start, :v_end,
    (Edge e) => DOUBLE{ return :e."length"/DOUBLE(:e."SPEED_MPH"); }, :i_direction);


  o_edges = SELECT :e."ID", :e."SOURCE", :e."TARGET", :EDGE_ORDER, :e."length", :e."SPEED_MPH"
            FOREACH e IN Edges(:p) WITH ORDINALITY AS EDGE_ORDER;
END;
++
@delimiter ;++

CALL "GS_SPOO_WEIGHTED"(1433737988, 1794145673, 'ANY', ?);
```

Shortest Path - Hohe Gewichtungen für Fahrradwege

Prof. Dr. Ingo Claßen

Montag, 13. September 2021 13:32

```
WeightedPath<DOUBLE> p = Shortest_Path(:g, :v_start, :v_end,
  (EDGE e)=> DOUBLE{
    IF(:e."highway" == 'cycleway' {
      RETURN :e."length"/10.0;
    } ELSE {
      RETURN :e."length";
    }
```

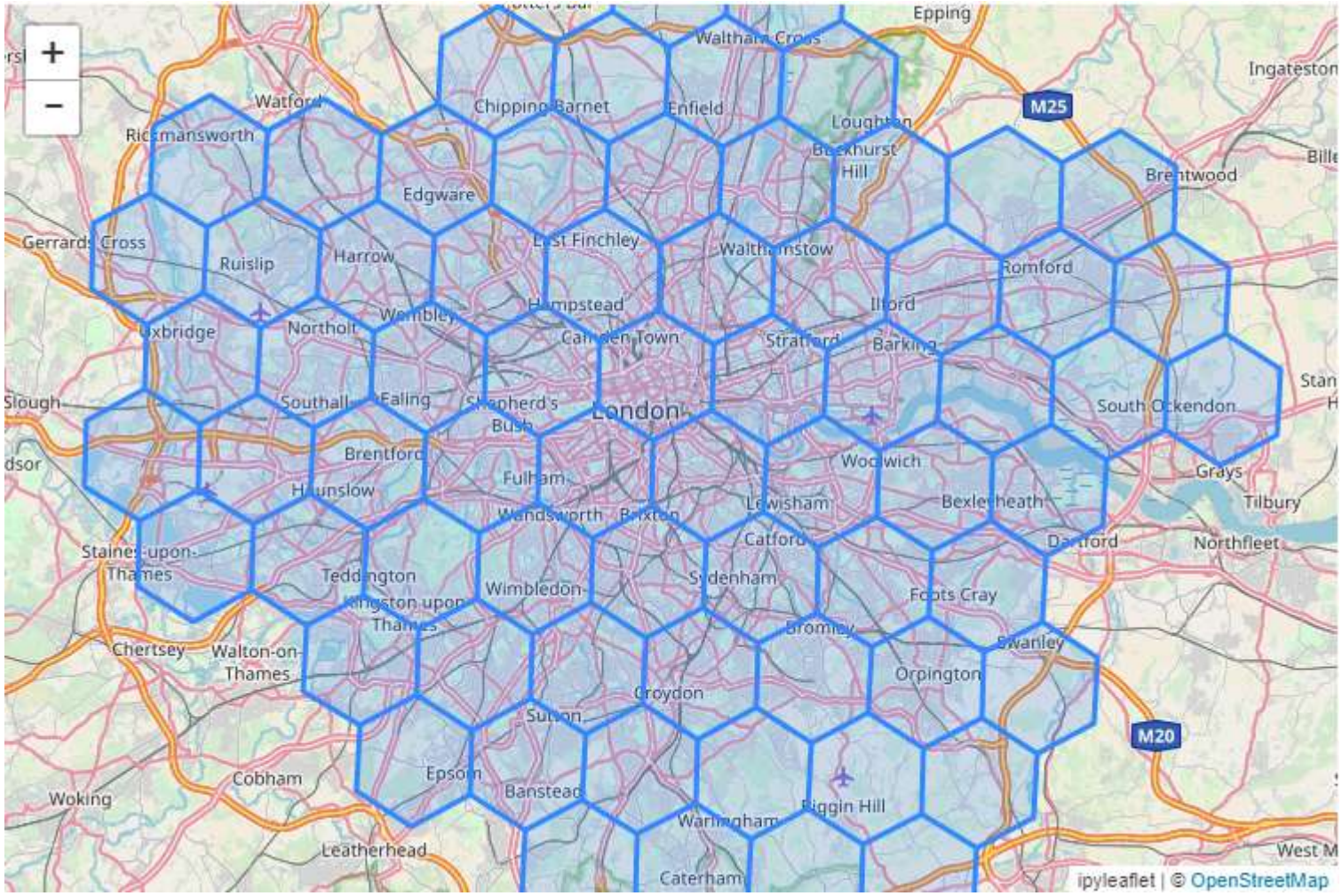
```
@delimiter ++;
CREATE OR REPLACE FUNCTION "F_SPOO_EDGES"(
  IN i_startVertex BIGINT,
  IN i_endVertex BIGINT,
  IN i_direction VARCHAR(10)
    )
 RETURNS "TT_SPOO_WEIGHTED_EDGES"
 LANGUAGE SQLSCRIPT READS SQL DATA AS
BEGIN
  declare o_edges "TT_SPOO_WEIGHTED_EDGES";
  CALL "GS_SPOO_WEIGHTED"(:i_startVertex, :i_endVertex, :i_direction, o_edges);
  o_edges = SELECT * from :o_edges;
  RETURN :o_edges;
END;
++
@delimiter ;++
```

```
select "geometry_GEO" as geo
from "F_SPOO_EDGES"(1433737988, 1794145673, 'ANY') fso
     join adbkt.LONDON_EDGES le
       on le.SOURCE=fso.SOURCE and le.TARGET=fso.TARGET;
```

Geht nur auf Hana Cloud

https://github.com/SAP-samples/teched2020-DAT260/blob/main/exercises/ex9/README.md

```
select ST_ClusterCell() as geo
from adbkt.london_vertices
group cluster by shape
using hexagon x cells 10;
```



```
select id, count(*) as no_of_schools
from
  (select ST_ClusterID() as id, ST_ClusterCell() as cell
   from adbkt.london_vertices
   group cluster by shape using hexagon x cells 10) h
  cross join
  (select shape_centroid
   from adbkt.london_poi
   where "amenity"='school') s
where h.cell.st_contains(s.shape_centroid)=1
group by h.id
order by no_of_schools desc;
```

*	ID	NO_OF_SCHOOLS
1	54	158
2	45	152
3	46	135
4	55	134
5	56	129
6	66	126
7	44	110
8	65	105
9	35	103
10	34	86

53	71	9
54	59	8
55	17	6
56	13	5
57	41	4
58	6	3
59	61	2
60	31	2
61	18	1