

JSON, Spatial, Graph –
Multi-model Workloads
with SAP HANA Cloud

Demo Scenario

London

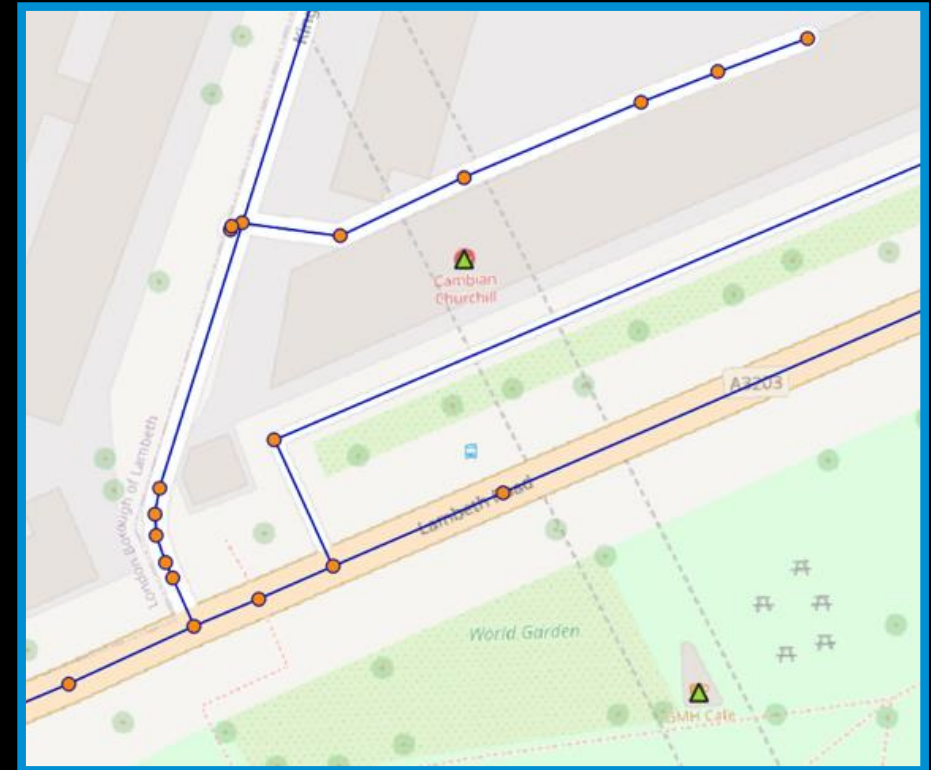
- Analyze **Points of Interest (POI)** locations and distribution
- Calculate POI relations, distance, and reachability based on the **street network**

Utility Networks, e.g. power grids

- Analyze **assets**, failures and work orders
- Calculate dependencies and simulate impact of maintenance work in a **utility network**

Supply and production networks

- Analyze and plan **material** supply and production
- Identify risks and alternative “**routes**” in case of supply shortage and production failures



Street network,
POIs

Points of Interest (POI) that Enrich Spatial Graph Data

<Demo> 2021 Q3 London POI 5 min LDBC.sql

```
-- just the basic queries

SET SCHEMA "MM_DEMO";
/*****
-- INSPECT THE DATA
*****/
-- There are 28.000 POI JSON Documents in POI_COLLECTION
-- The POI geo locations are stored in the POI_SHAPE table
-- We can join the JSON collection to the relational table
WITH "POI_COLLECTION" AS (SELECT TO_BIGINT("id") AS "id", * FROM "POI_COLLECTION")
SELECT PC.*, PS."SHAPE_32630"
  FROM "POI_COLLECTION" AS PC
 LEFT JOIN "POI_SHAPE" AS PS ON PC."id" = PS."ID"
 ORDER BY "ID" LIMIT 100;

/*****
-- Inspect OSM street network data
-- Both tables have a geometry - 350.000 EDGES have LINESSTRINGs and 180.000 VERTICES have POINTS
SELECT "osmid", "u", "v", "highway", "length", "maxspeed", "SHAPE_32630", *
  FROM "LONDON_EDGES"
 LIMIT 100;
SELECT "osmid", "y", "x", "SHAPE_32630"
  FROM "LONDON_VERTICES"
 LIMIT 100;
```

POI_COLLECTION(+)
LONDON_EDGES 1 (2)
LONDON_VERTICES 1 (3)
POI_SHAPE(+)
LONDON_EDGES 1 (5)
LONDON_VERTICES 1 (6)
SYS_SS_TBL_157736_RET 1 (7)
SYS_SS_TBL_157736_RET(+)
DO 1 (9)
Statistics 1

WITH "POI_COLLECTION" AS (SELECT TO_BIGINT("id") AS "id", * FROM "POI_COLLECTION")

id	POI_COLLECTION	SHAPE_32630
1	99,884 ("type": "node", "id": 99884, "lat": 51.5243642, "lon": -0.1528165, "tags": {"amenity": "waste_basket"})	POINT (697507.1098556519 5711982.02281189)
2	108,042 ("type": "node", "id": 108042, "lat": 51.5235613, "lon": -0.1355134, "tags": {"addr:housenumber": "31", "addr:post": "POINT (698710.6856002808 5711939.621986389)"	
3	108,539 ("type": "node", "id": 108539, "lat": 51.5291251, "lon": -0.0933878, "tags": {"amenity": "bicycle_rental", "brand": "S: POINT (701607.8062133789 5712673.505599976)"	
4	109,575 ("type": "node", "id": 109575, "lat": 51.5282624, "lon": -0.1431214, "tags": {"amenity": "advice", "name": "Citizens a POINT (698162.5935592651 5712441.660400391)"	
5	110,075 ("type": "node", "id": 110075, "lat": 51.5342569, "lon": -0.1402758, "tags": {"amenity": "bicycle_parking", "capacity": "POINT (698333.8701400757 5713115.858970642)"	
6	13,799,212 ("type": "node", "id": 13799212, "lat": 51.5210756, "lon": -0.1156142, "tags": {"amenity": "library", "name": "Holboi POINT (700101.740524292 5711717.508140564)"	
7	15,262,028 ("type": "node", "id": 15262028, "lat": 51.5164813, "lon": -0.1698266, "tags": {"addr:country": "GB", "addr:housenu POINT (696361.2669067383 5711059.7802734375)"	
8	20,599,885 ("type": "node", "id": 20599885, "lat": 51.5302738, "lon": -0.117551, "tags": {"amenity": "disused_pub", "name": "G POINT (699927.0894393921 5712734.8862838745)"	
9	20,821,133 ("type": "node", "id": 20821133, "lat": 51.5004234, "lon": -0.1777428, "tags": {"amenity": "pub", "name": "The Unio POINT (695881.0193862915 5709253.191612244)"	
10	20,973,316 ("type": "node", "id": 20973316, "lat": 51.4727261, "lon": -0.1811598, "tags": {"amenity": "pub", "name": "The Wat POINT (695762.5447616577 5706164.582572937)"	
11	21,392,280 ("type": "node", "id": 21392280, "lat": 51.5154842, "lon": -0.1049292, "tags": {"addr:city": "London", "addr:houseni POINT (700867.4713058472 5711125.133056641)"	
12	21,563,447 ("type": "node", "id": 21563447, "lat": 51.5125836, "lon": -0.1067767, "tags": {"amenity": "post_box", "collection_tir POINT (700752.0694732666 5710797.567710876)"	
13	21,565,294 ("type": "node", "id": 21565294, "lat": 51.517885, "lon": -0.1091282, "tags": {"amenity": "bicycle_parking", "bicycle POINT (700565.6381988525 5711380.535858154)"	
14	21,593,232 ("type": "node", "id": 21593232, "lat": 51.5153316, "lon": -0.1117428, "tags": {"addr:city": "London", "addr:houseni POINT (700395.4865570068 5711089.478096008)"	
15	21,593,236 ("type": "node", "id": 21593236, "lat": 51.5172349, "lon": -0.1192033, "tags": {"addr:city": "London", "addr:country POINT (699869.6185913086 5711280.678359985)"	
16	21,593,237 ("type": "node", "id": 21593237, "lat": 51.5173465, "lon": -0.1189566, "tags": {"addr:city": "London", "addr:houseni POINT (699886.2425689697 5711293.760398865)"	

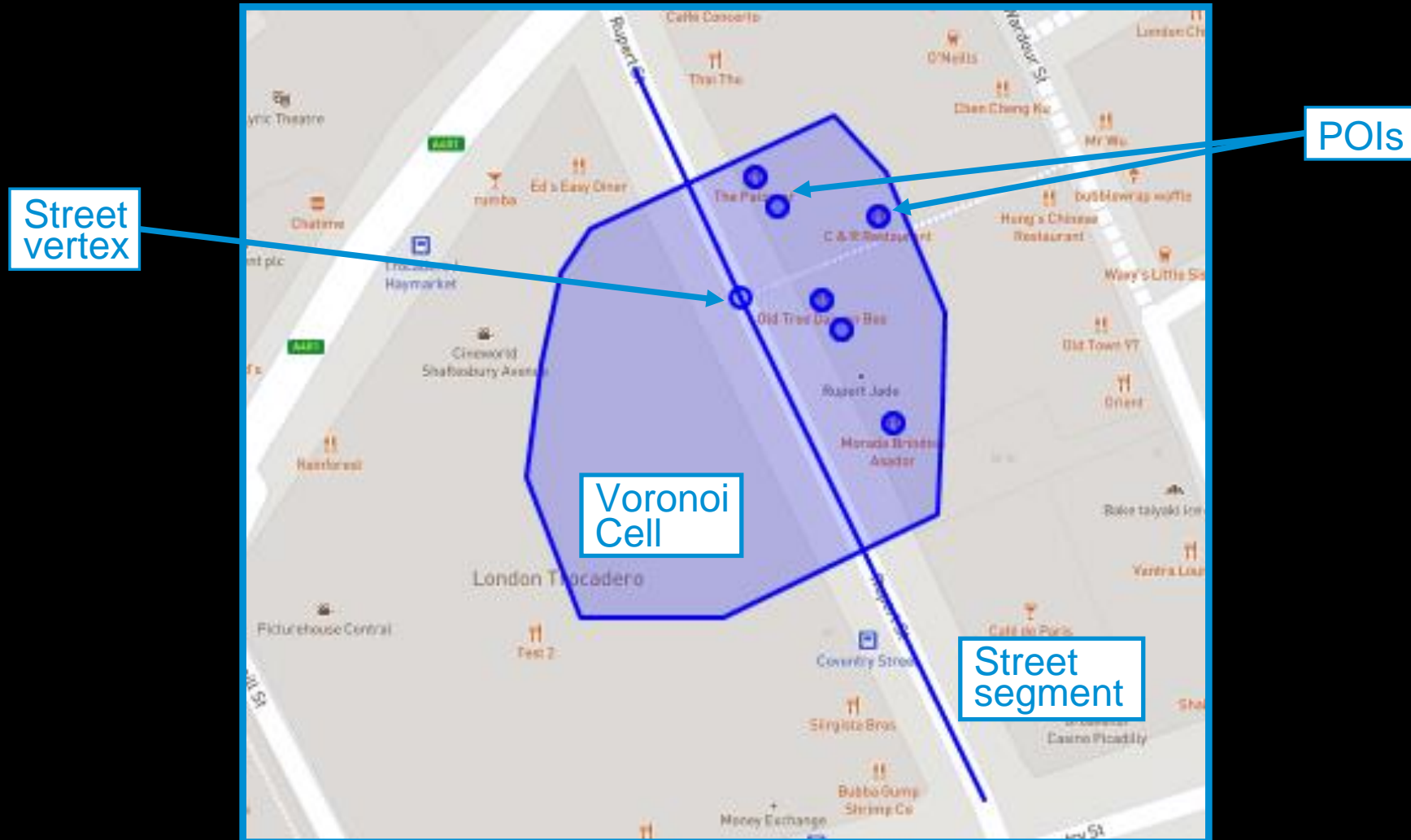
Leaflet | © OpenStreetMap contributors

200 row(s) fetched - 3.247s (+)

Save Cancel Script 200 100 Rows: 1

CET en Writable Smart Insert 15:21:650 Sel: 0|0

Snap POIs to Street Network Vertices



Snap POIs to Street Network Vertices

<Demo> 2021 Q3 London POI 5 min LDDBC.sql

```

/*****
-- Task: Snap POIs to street network vertices
-- The street network will become the graph, but the POIs need to be snapped to the graph's vertices
-- Solution: We used a spatial function to generate Voronoi cells and find for each POI the nearest street vertex

-- Each of the 28.000 POIs are snapped to the nearest street network vertex
-- Let's inspect the clinic POIs and the street vertices they are snapped to
SELECT PS."ID", PS."NAME", PS."SHAPE_32630" AS "POI_LOC", LV."VORONOI_CELL", LV."SHAPE_32630" AS "VERTEX_LOC"
FROM "POI_SHAPE" AS PS
LEFT JOIN "LONDON_VERTICES" LV ON PS."VERTEX_OSMID" = LV."osmid"
WHERE "AMENITY" = 'clinic';

/*****
-- Create a graph model, i.e. a GRAPH WORKSPACE on the street network - LONDON_VERTICES and LONDON_EDGES
-- LONDON_EDGES contains "u" and "v" which are the source and the target of the edges,
-- pointing to source/target vertices
SELECT "ID", "u", "v", "highway", "SHAPE_32630" FROM "LONDON_EDGES";
SELECT "osmid", "SHAPE_32630" FROM "LONDON_VERTICES";

CREATE GRAPH WORKSPACE "LONDON_GRAPH"
EDGE TABLE "LONDON_EDGES"

```

POI_SHAPE(+ 1 (4) | LONDON_EDGES 1 (5) | LONDON_VERTICES 1 (6) | _SYS_SS_TBL_157736_RET 1 (7) | _SYS_SS_TBL_157736_RET(+ 1 (8) | DO 1 (9) | Statistics 1

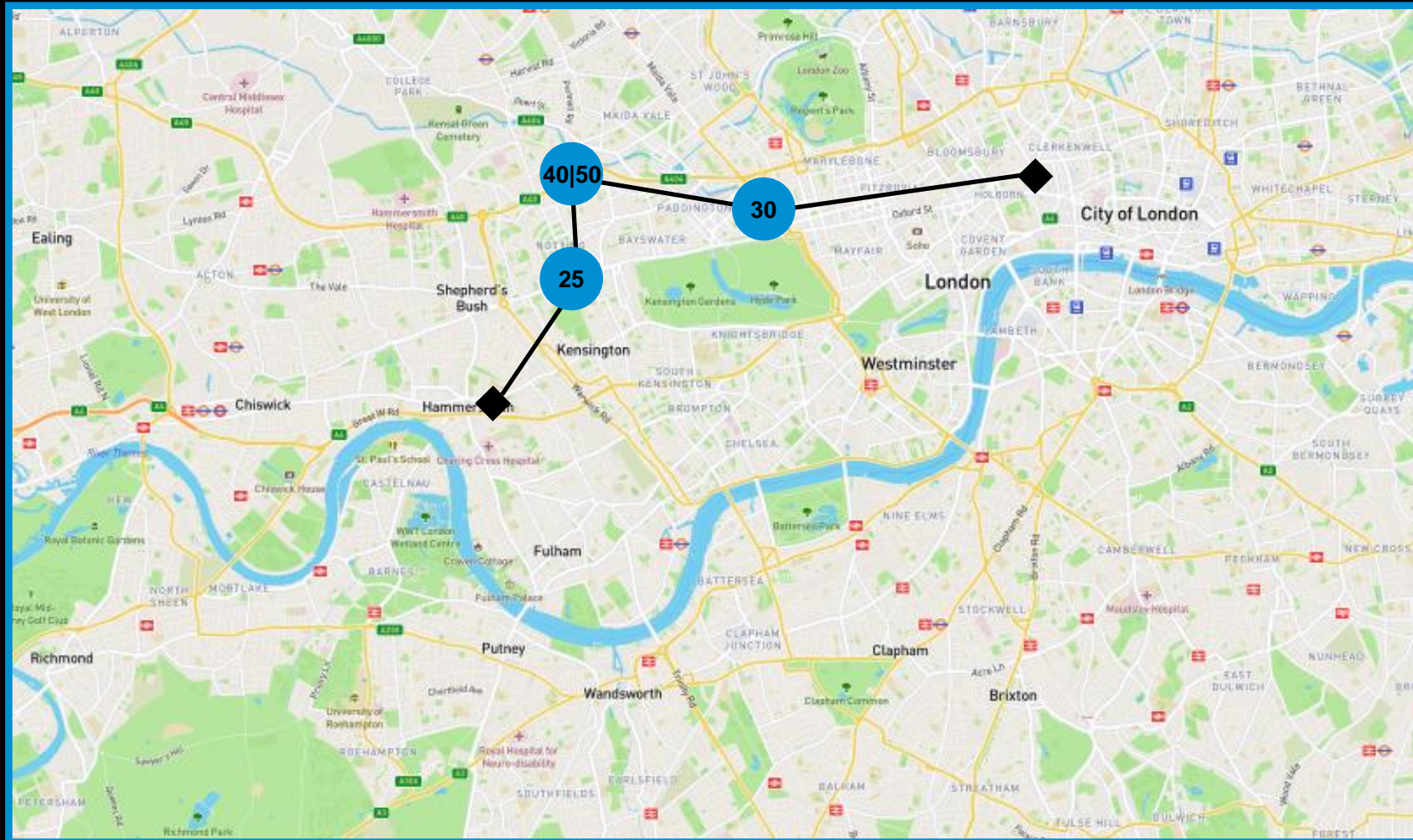
SELECT PS."ID", PS."NAME", PS."SHAPE_32630" AS "POI_LOC", LV."VORONOI_CELL", LV."VERTEX_LOC"

ID	NAME	POI_LOC	VORONOI_CELL	VERTEX_LOC
10	4,193,845,924 Physical Health Clinic	POINT (701826.7266769409 5711771.909843445	POLYGON ((701822.1925354004 5711759.20715332, 7	POINT (701841.8167266846
11	4,320,155,180 Pentonville Road Dental	POINT (699840.4238357544 5712800.633010864	POLYGON ((699837.8786392212 5712799.864051819,	POINT (699854.2830505371
12	3,986,315,089 Smokers Clinic	POINT (704022.2379989624 5711357.829330444	POLYGON ((704000.9874725342 5711352.608093262,	POINT (704037.2683944702
13	3,986,315,090 Health and Wellness Centre	POINT (703968.9671173096 5711389.830635071	POLYGON ((703944.7946472168 5711434.29750061,	POINT (703977.4290237427
14	3,997,671,020 Tobacco Dependence Research	POINT (704011.6937408447 5711377.80657196)	POLYGON ((703979.1680603027 5711355.446983337,	POINT (704023.1275787354
15	7,838,168,098 Battersea Respiratory Clinic	POINT (696778.7986984253 5706282.687263488	POLYGON ((696740.9356994629 5706256.070007324,	POINT (696772.7700195312
16	3,227,939,099 CREATE Fertility	POINT (701424.925201416 5711084.849273682)	POLYGON ((701436.2924194336 5711096.856460571,	POINT (701414.7802200317
17	4,117,637,266 Queenstown Medical Centre	POINT (697919.7998428345 5705599.976715088	POLYGON ((697943.6403579712 5705606.345504761,	POINT (697910.8900222778
18	4,434,592,817 Institute of Sport Exercise and	POINT (698683.8702926636 5711833.08695209	POLYGON ((698653.2125015259 5711808.588874817,	POINT (698666.4082107544
19	6,914,948,964 Six Physio	POINT (702225.7821426392 5710641.005027771	POLYGON ((702230.2641906738 5710643.690216064,	POINT (702218.3608856201
20	4,254,744,158 St. Katharine Docks Practice	POINT (703450.9754486084 5710303.356651306	POLYGON ((703447.7867126465 5710304.136421204,	POINT (703461.6271743774
21	3,270,345,413 The Village Practice	POINT (700316.4515609741 5716024.469062805	POLYGON ((700310.5713729858 5716026.501693726,	POINT (700323.4581604004
22	7,908,351,156 Clapham Park Group Practice	POINT (699001.5477294922 5703983.414207458	POLYGON ((698965.3582458496 5704014.163261414,	POINT (698982.7301940918
23	5,172,768,852 Millbank Medical Centre	POINT (699436.3929367065 5708692.721191406	POLYGON ((699412.0360565186 5708717.051277161,	POINT (699452.8197555542
24	5,174,812,335 Randox Health	POINT (702244.7620391846 5711403.794044495	POLYGON ((702282.9957275391 5711388.09059906,	POINT (702256.6359405518

Map view showing a street network with a highlighted polygon and a red dot representing a POI. Labels include Shuttleworth Road, Bridge Lane Nursery, and Bridge Lane Health Centre.

Save | Cancel | Script | 200 | 74 | Rows: 1 | CET en | Writable | Smart Insert | 55 : 50 : 1972 | Sel: 0 | 0 | 74 row(s) fetched

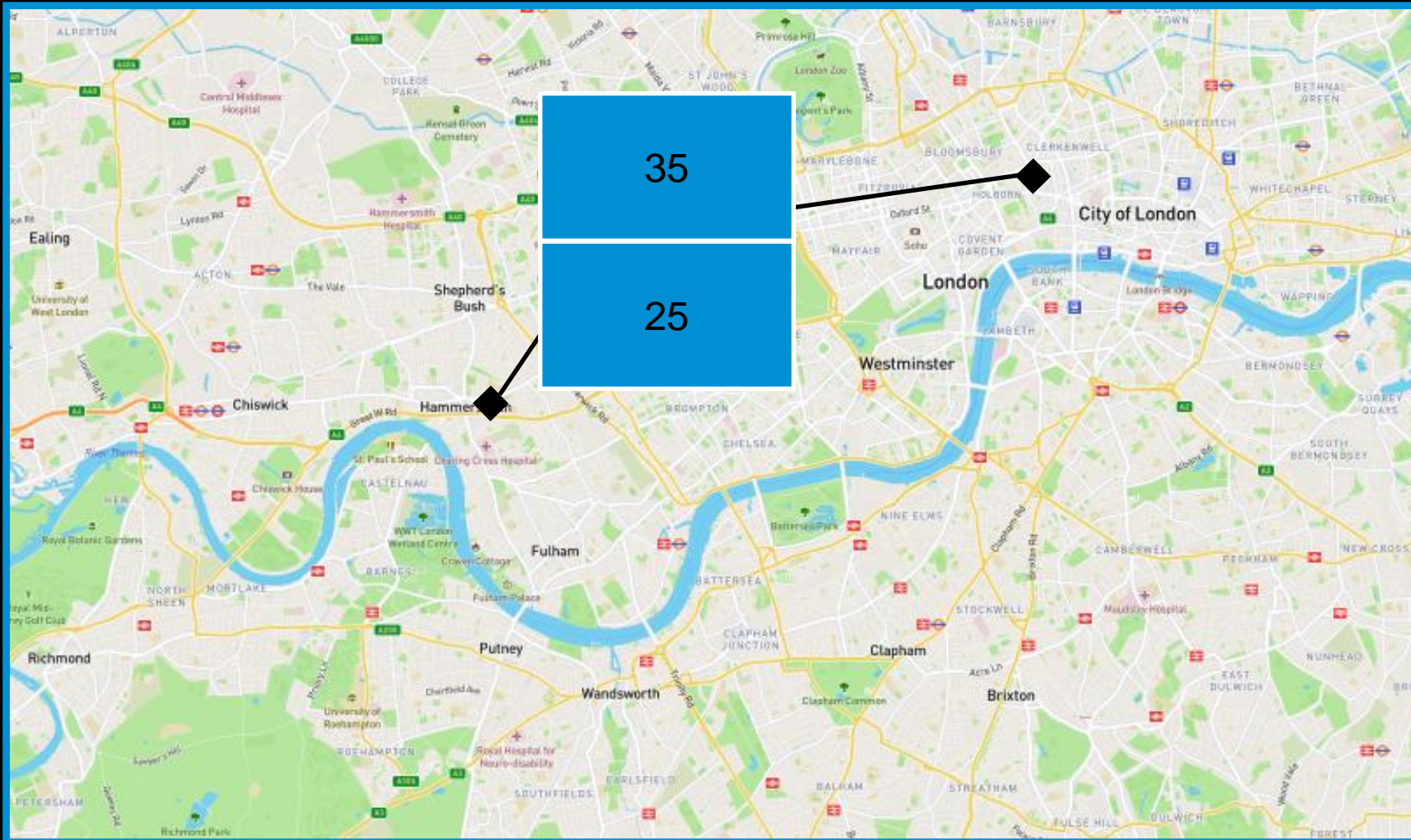
Isochrone Map



1 select POIs

2 calculate drive time distances

Isochrone Map



1 select POIs

2 calculate drive time distances

3 select min distance for each vertex

4 spatial clustering and calculate average

Calculation of Drive-Time Distances using GraphScript

```
<Demo> 2021 Q3 London POI 5 min LDDBC.sql
/*****
-- ANALYZE DATA
*****/
-- Graph database procedures

-- Now, one of the core operations on a graph is to find paths.
-- In HANA, Shortest Paths One-to-All is one of the built-in algorithms.

-- Given start, the procedure returns all reachable vertices and their drive-time distance
-- This procedure is the building block for subsequent analysis steps:
-- we will calculate drive-time isochrones for the 73 clinics in London

CREATE OR REPLACE PROCEDURE "GS_SPOA"(
  IN i_startVertex BIGINT, -- INPUT: the key of the start vertex
  IN i_maxDriveTime DOUBLE, -- INPUT: the maximum drive time
  OUT o_vertices "TT_SPOA_VERTICES" -- OUTPUT: a table structure
)
LANGUAGE GRAPH READS SQL DATA AS
BEGIN
  GRAPH g = Graph("MM_DEMO", "LONDON_GRAPH");
  VERTEX v_start = Vertex(:g, :i_startVertex);
  GRAPH g_spoa = SHORTEST_PATHS_ONE_TO_ALL(:g, :v_start, "DRIVE_TIME",
    (EDGE e, DOUBLE current_drive_time) => DOUBLE {
      IF(:current_drive_time <= :i_maxDriveTime) {
        RETURN :e."length"/(DOUBLE(:e."SPEED_MPH") * 0.44704); -- length is in m, speed is mph, the result is in seconds
      }
      ELSE {
        END TRAVERSE;
      }
    });
  o_vertices = SELECT :v."osmid", :v."DRIVE_TIME" FOREACH v IN Vertices(:g_spoa);
END;

-- We can call the procedure via a SQL Table Function
-- Start at Fern Skin Clinic 812348430
SELECT * FROM "FT_SPOA" (i_startVertex => 812348430, i_maxDriveTime => 120.0) ORDER BY "DRIVE_TIME" ASC;

-- ... and join the SPOA result to our JSON collection.
WITH "POI_COLLECTION" AS (SELECT TO_BIGINT("id") AS "id", * FROM "POI_COLLECTION")
SELECT "DRIVE_TIME", "NAME", "AMENITY", "SHAPE_32630", "POI_COLLECTION"
FROM "FT_SPOA" (812348430, 120.0) AS "SPOA"
LEFT JOIN "POI_SHAPE" AS PS ON SPOA."END_VERTEX" = PS."VERTEX_OSMID"
LEFT JOIN "POI_COLLECTION" AS PC ON PS."ID" = PC."id"
ORDER BY "DRIVE_TIME" ASC;
```


Calculate Drive-Time Isochrone Map

```
<Demo> 2021 Q3 London POI 5 min LDBC.sql
ORDER BY "DRIVE_TIME" ASC;

/*****/
-- Calculate drive-time isochrone map
-- The query below
-- 1. takes 73 "clinic" POIs
-- 2. runs parallel calculations of SPOA based on drive-time
-- 3. merges the SPOA results, identifies minimum drive-time distance to next clinic for each vertex
-- 4. runs spatial clustering to generate a isochrone map

@SELECT * FROM SQL FUNCTION (
  IN i_amenity NVARCHAR(500) => 'clinic',
  IN i_maxDriveTime DOUBLE => 360.0
)
  RETURNS TABLE("ID" BIGINT, "CELL_32630" ST_GEOMETRY(32630), "DRIVE_TIME" DOUBLE)
  BEGIN
-- 1 get the 73 clinics
startPOIs = SELECT "VERTEX_OSMID" AS "START_VERTEX" FROM "POI_SHAPE" WHERE "AMENITY" = :i_amenity;
-- 2 parallel SPOA calculations for the clinics
o_SPOA_Results = MAP_MERGE(:startPOIs, "FT_SPOA"(:startPOIs."START_VERTEX", :i_maxDriveTime));
-- 3 keep only minimum drive-time distance for each street vertex
o_vertices_minDriveTime = SELECT "START_VERTEX", "END_VERTEX", "DRIVE_TIME" FROM (
  SELECT "START_VERTEX", "END_VERTEX", "DRIVE_TIME",
    RANK() OVER(PARTITION BY "END_VERTEX" ORDER BY "DRIVE_TIME" ASC) AS "RANK"
  FROM :o_SPOA_Results
) WHERE "RANK" = 1;
-- 4 apply spatial clustering on the street vertices
RETURN
  SELECT ST_ClusterID() AS "ID", ST_ClusterCell() AS "CELL_32630", CAST(AVG("DRIVE_TIME") AS DOUBLE) AS "DRIVE_TIME" FROM (
    SELECT v."START_VERTEX", lv."SHAPE_32630", v."DRIVE_TIME"
    FROM :o_vertices_minDriveTime AS v LEFT JOIN "LONDON_VERTICES" AS lv ON v."END_VERTEX" = lv."osmid")
  GROUP CLUSTER BY "SHAPE_32630" USING HEXAGON X CELLS 50;
END
;
```

Drive-Time Isochrones for 73 Clinics in London

