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
Points of Interest (POI) that Enrich Spatial Graph Data

```sql
-- Just the basic queries

-- SELECT COUNT(*) FROM POI_COLLECTION;

-- SELECT * FROM POI_COLLECTION LIMIT 10;

SELECT id, name, type, amenity, description, location FROM POI_COLLECTION LIMIT 10;
```

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>type</th>
<th>amenity</th>
<th>description</th>
<th>location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>London Eye</td>
<td>ATtraction</td>
<td></td>
<td></td>
<td>Point</td>
</tr>
<tr>
<td>2</td>
<td>Big Ben</td>
<td>ATtraction</td>
<td></td>
<td></td>
<td>Point</td>
</tr>
<tr>
<td>3</td>
<td>Buckingham P</td>
<td>ATtraction</td>
<td></td>
<td></td>
<td>Point</td>
</tr>
<tr>
<td>4</td>
<td>Tower Bridge</td>
<td>ATraction</td>
<td></td>
<td></td>
<td>Point</td>
</tr>
<tr>
<td>5</td>
<td>Tower Bridge</td>
<td>ATraction</td>
<td></td>
<td></td>
<td>Point</td>
</tr>
<tr>
<td>6</td>
<td>Tower Bridge</td>
<td>ATraction</td>
<td></td>
<td></td>
<td>Point</td>
</tr>
<tr>
<td>7</td>
<td>Tower Bridge</td>
<td>ATraction</td>
<td></td>
<td></td>
<td>Point</td>
</tr>
<tr>
<td>8</td>
<td>Tower Bridge</td>
<td>ATraction</td>
<td></td>
<td></td>
<td>Point</td>
</tr>
<tr>
<td>9</td>
<td>Tower Bridge</td>
<td>ATraction</td>
<td></td>
<td></td>
<td>Point</td>
</tr>
<tr>
<td>10</td>
<td>Tower Bridge</td>
<td>ATraction</td>
<td></td>
<td></td>
<td>Point</td>
</tr>
</tbody>
</table>
Snap POIs to **Street Network Vertices**

- **Street vertex**
- **Voronoi Cell**
- **Street segment**
- **POIs**
Snap POIs to Street Network Vertices
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
Calculate Drive-Time Isochrone Map

```sql
-- 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_ amenity, WORKDAY(0) AS 'clinic',  
   IN_ maxDriveTime, DOUBLE AS 360.0  
)  
RETURNS TABLE("ID" BIGINT, "CELL_32638" ST_GEOMETRY(32638), "DRIVE_TIME" DOUBLE) BEGIN  

-- 1 get the 73 clinics
startPOIs = SELECT "VERTEX_O Sid" AS "START_VERTEX" FROM "POI_SHAPE" WHERE "AMENITY" = :i_ amenity;  
-- 2 parallel SPOA calculations for the clinic
o_SPOA_ Results = MAP_REDUCE(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"  
   FROM o_SPOA_Results  
   WHERE "RANK" = 1)  
   ORDER BY "DRIVE_TIME" ASC  
   LIMIT 1;  
-- 4 apply spatial clustering on the street vertices
RETURN  
SELECT ST_ClusterID() AS "ID", ST_ClusterCentroid() AS "CELL_32638", CAST(AVG("DRIVE_TIME") AS DOUBLE) AS "DRIVE_TIME" FROM (  
   SELECT v."START_VERTEX", v."SHAPE_32638", v."DRIVE_TIME"  
   FROM o_vertices_minDriveTime AS v LEFT JOIN "LONDON_VERTICES" AS v ON v."END_VERTEX" = lv."oidId"  
   GROUP CLUSTER BY "SHAPE_32638" USING HEXAGON X CELLS 30)  
END
```

© 2021 SAP SE or an SAP affiliate company. All rights reserved. | PUBLIC
Drive-Time Isochrones for 73 Clinics in London