Technical Spotlight

Inland Standing Water Polygons from OS MasterMap

MAY 2020

Continuing with the theme of last months' feature, we are looking at customer query to identify standing water (static) polygons within England. Similar, to the last article, the best way to do this is through extraction of Topographic area features fromOS MasterMap Topography Layer. Additionally, as the features need to be limited to England, we will use a boundary dataset of combined regions that constitute England – from BoundaryLine.

What do you need to do to answer this query?

Follow this 3-step script building workflow to execute as a query within a spatial database. In this example we have loaded both products into the spatial database beforehand:

1.

The first step is to create a spatial index on the geometry field for both OS MasterMap Topography & BoundaryLine datasets. This helps the spatial database in its speed of identifying and categorising features to utilise in a spatial query thereafter.

```
-- Create indexes on geometry fields

create index osmm_topo_geom on osmm_topo.topographicarea using GIST (geom);
vacuum analyze osmm_topo.topographicarea;

create index boundaryline_region_geom on boundaryline.european_region using GIST
(geom);
vacuum analyze boundaryline.europeanregion;
```

Note:

OS MasterMap Topography dataset name in spatial database: osmm_topograhicarea
BoundaryLine dataset name in spatial database: boundaryline.europeanregion

2.

Now that we spatial indexes generated, we can move on to restricting features in both datasets to only features we require for the spatial query. To do this, we use the following script that looks up topographic areas and limits features to only those that have 'Static Water' in their 'DescriptiveTerm' field. Following this, the script will look up within the 'European Region' boundary of the BoundaryLine dataset and limit features to only those that are regions of England.

```
-- Limit Topography features to Static Water & Limit BoundaryLine features to
combined England polygon
```

```
SELECT toid, featurecode, "version", versiondate, theme, calculatedareavalue,
changedate, reasonforchange, descriptivegroup, descriptiveterm, make,
physicallevel, physicalpresence, a.geom, first_change_date, last_change_date,
style_description, style_code
FROM osmm_topo.topographicarea
WHERE
  'Static Water' = ANY(osmm_topo.topographicarea.descriptiveterm) as a;

SELECT "name", b.geom
FROM boundaryline.european_region
where boundaryline.european_region."name" = 'East Midlands Euro
Region' or boundaryline.european_region."name" = 'Eastern Euro
Region' or boundaryline.european_region."name" = 'London Euro
Region' or boundaryline.european_region."name" = 'North East Euro
Region' or boundaryline.european_region."name" = 'North West Euro
Region' or boundaryline.european_region."name" = 'South East Euro
Region' or boundaryline.european_region."name" = 'South West Euro
Region' or boundaryline.european_region."name" = 'West Midlands Euro
Region' or boundaryline.european_region."name" = 'Yorkshire and the Humber Euro
Region' as b;
```

Note:

This script now creates an alias for each dataset. Calling Topography dataset as ‘a’ and BoundaryLine dataset as ‘b’ (which can be seen in the last part of each section of query). This will help keep the spatial query manageable and less repeatable.

3.

In the third step, we are now able to run a spatial query to find the Standing Water features that reside within England. To do this, we can create a spatial query that checks which features *intersect* with each other. Once identified, these will be automatically placed in a brand-new table of the spatial database.

```
-- Run an intersection query and create a new table of resulting Standing Water
Polygon features

create table osmm_topo.england_standing_water_features as
SELECT toid, featurecode, "version", versiondate, theme, calculatedareavalue,
changedate, reasonforchange, descriptivegroup, descriptiveterm, make,
physicallevel, physicalpresence, a.geom, first_change_date, last_change_date,
style_description, style_code FROM a,
(SELECT geom FROM b)
WHERE ST_Intersects(a.geom,b.geom);
```

Note:

As described in Step 2 notes, a & b in the script refer to the aliases created.

We now have a new table of the resulting features. At this point, we used a GIS software package to connect to the new table in the spatial database. Once connected, we outputted the features to a spatial format for use in the customers system.