Guillaume Minette de Saint Martin

Advanced data Technologies

Lab 5

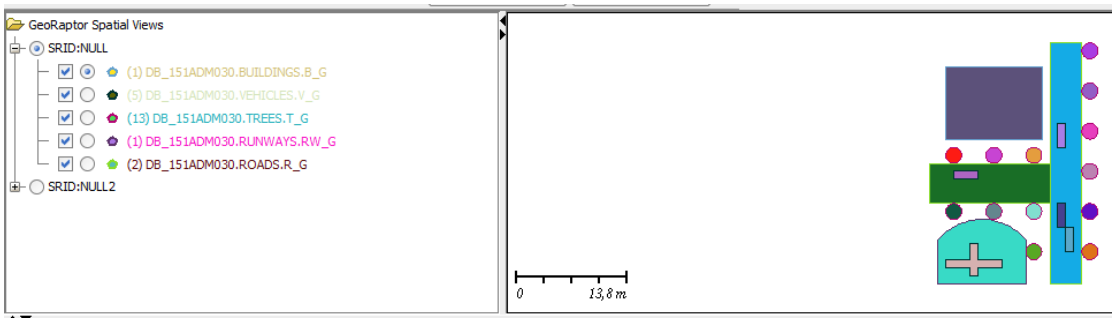
Table of contents

Drawings

The designed tables will describe informations about the geography of an airport.

For this, we can identify runways and airplanes but also roads and cars as well as buildings and trees.

There is the Georaptor display of the airport drawing



Tables creation

Tables

In order to store informations in database, several layers are created to store the different objects.

These layouts are stored in tables which correspond to the different types of objects represented in the database.

Tables are the following:

- Roads
- Buildings
- Trees
- Runways
- Vehicles

SQL code

Each table contains 3 columns: an id, a name and the geometry item.

Sequences are used to automatically fill the id fields during inserts.

Roads

```
CREATE TABLE roads (
  id NUMBER,
  name VARCHAR2(50),
  r_g SDO_GEOMETRY );
```

Buildings

```
CREATE TABLE buildings (
  id NUMBER,
  name VARCHAR2(50),
  b_g SDO_GEOMETRY );
```

Trees

```
CREATE TABLE trees (
  id NUMBER,
  name VARCHAR2(50),
  t_g SDO_GEOMETRY );
```

Runways

```
CREATE TABLE runways (
  id NUMBER,
  name VARCHAR2(50),
  rw_g SDO_GEOMETRY );
```

Vehicles

```
CREATE TABLE vehicles (
  id NUMBER,
  name VARCHAR2(50),
  v_g SDO_GEOMETRY );
```

Sequences

```
create sequence roads_seq;
create sequence buildings_seq;
create sequence trees_seq;
create sequence runways_seq;
create sequence vehicles_seq;
```

Metadata insertion

For each table, metadata are inserted in order to specify the dimensions of the two axis used to display the spacial figures.

Roads

```
INSERT  INTO  USER_SDO_GEOM_METADATA(TABLE_NAME, COLUMN_NAME,
DIMINFO, SRID)
  VALUES('roads', 'r_g',MDSYS.SDO_DIM_ARRAY(
  MDSYS.SDO_DIM_ELEMENT('X', 0, 30, 1),
  MDSYS.SDO_DIM_ELEMENT('Y', 0, 30, 1)),
  NULL);
```

Buildings

```
INSERT  INTO  USER_SDO_GEOM_METADATA(TABLE_NAME, COLUMN_NAME,
DIMINFO, SRID)
  VALUES('buildings', 'b_g',MDSYS.SDO_DIM_ARRAY(
  MDSYS.SDO_DIM_ELEMENT('X', 0, 30, 1),
  MDSYS.SDO_DIM_ELEMENT('Y', 0, 30, 1)),
  NULL);
```

Trees

```
INSERT  INTO  USER_SDO_GEOM_METADATA(TABLE_NAME, COLUMN_NAME,
DIMINFO, SRID)
  VALUES('trees', 't_g',MDSYS.SDO_DIM_ARRAY(
  MDSYS.SDO_DIM_ELEMENT('X', 0, 30, 1),
  MDSYS.SDO_DIM_ELEMENT('Y', 0, 30, 1)),
  NULL);
```

Runways

```
INSERT  INTO  USER_SDO_GEOM_METADATA(TABLE_NAME, COLUMN_NAME,
DIMINFO, SRID)
  VALUES('runways', 'rw_g',MDSYS.SDO_DIM_ARRAY(
  MDSYS.SDO_DIM_ELEMENT('X', 0, 30, 1),
  MDSYS.SDO_DIM_ELEMENT('Y', 0, 30, 1)),
```

```
  NULL);
```

Vehicles

```
INSERT  INTO  USER_SDO_GEOM_METADATA(TABLE_NAME, COLUMN_NAME,
DIMINFO, SRID)
  VALUES('vehicles', 'v_g',MDSYS.SDO_DIM_ARRAY(
  MDSYS.SDO_DIM_ELEMENT('X', 0, 30, 1),
  MDSYS.SDO_DIM_ELEMENT('Y', 0, 30, 1)),
  NULL);
```

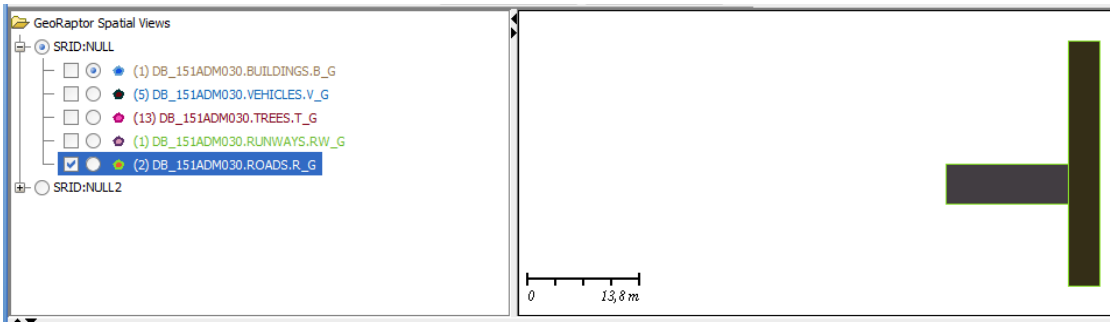
Data insertion

In each table, at least one item is inserted.

Roads

This table uses rectangles to draw roads, it Is therefore needed to input two points to draw it: the lower left and the upper right: example sdo_ordinate_array (15,0, 19,30).

```
INSERT INTO roads (id, name, r_g) VALUES (roads_seq.nextval,
'road1',sdo_geometry (2003, null, null,
  sdo_elem_info_array (1,1003,3),
  sdo_ordinate_array (15,0, 19,30)));
INSERT INTO roads (id, name, r_g) VALUES (roads_seq.nextval,
'road2',sdo_geometry (2003, null, null,
  sdo_elem_info_array (1,1003,3),
  sdo_ordinate_array (0,10, 15,15)));
```



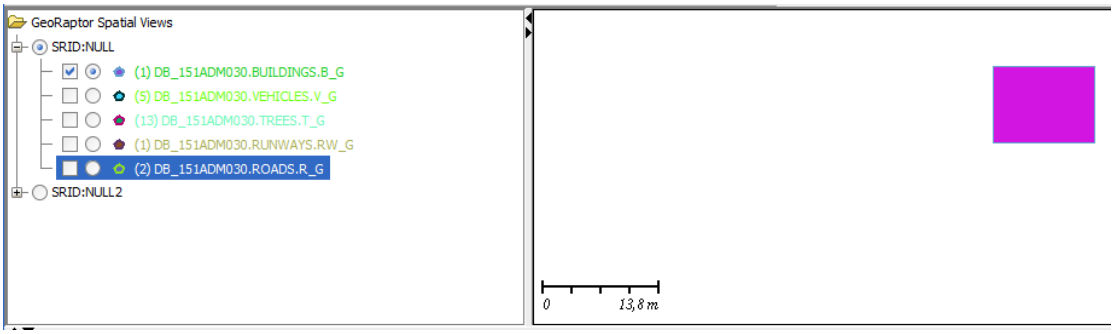
Roads graphical display

Buildings

This table uses rectangles to draw buildings, buildings are then also drawn according to their lower left and upper right point: example: sdo_ordinate_array (2,18, 14,27).

```
INSERT INTO buildings (id, name, b_g) VALUES (buildings_seq.nextval,
'building1',sdo_geometry (2003, null, null,
  sdo_elem_info_array (1,1003,3),
```

```
  sdo_ordinate_array (2,18, 14,27)));
```



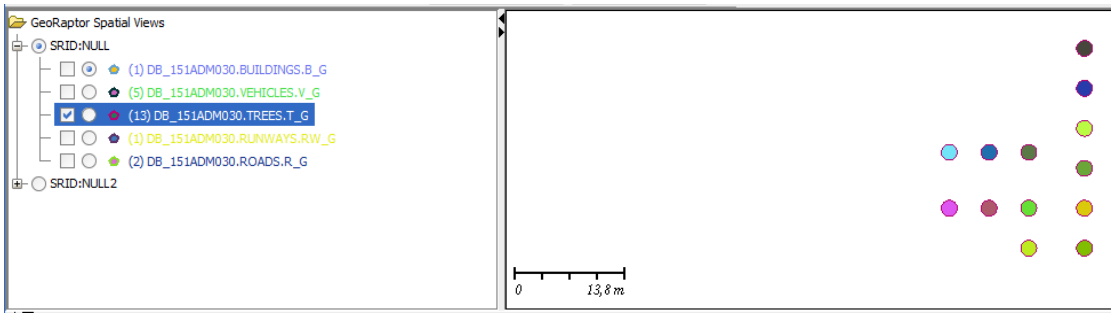
Buildings graphical display

Trees

This table uses circles to draw trees, trees are then drawn according to their diameter's coordinates as well as the coordinates of their centre point: example: sdo_ordinate_array (3,17, 4,16, 3,15).

```
INSERT INTO trees (id, name, t_g) VALUES (trees_seq.nextval,
'tree1',sdo_geometry (2003, null, null,
  sdo_elem_info_array (1,1003,4),
  sdo_ordinate_array (3,17, 4,16, 3,15)));
INSERT INTO trees (id, name, t_g) VALUES (trees_seq.nextval,
'tree2',sdo_geometry (2003, null, null,
  sdo_elem_info_array (1,1003,4),
  sdo_ordinate_array (8,17, 9,16, 8,15)));
INSERT INTO trees (id, name, t_g) VALUES (trees_seq.nextval,
'tree3',sdo_geometry (2003, null, null,
  sdo_elem_info_array (1,1003,4),
  sdo_ordinate_array (13,17, 14,16, 13,15)));
INSERT INTO trees (id, name, t_g) VALUES (trees_seq.nextval,
'tree4',sdo_geometry (2003, null, null,
  sdo_elem_info_array (1,1003,4),
  sdo_ordinate_array (3,10, 4,9, 3,8)));
INSERT INTO trees (id, name, t_g) VALUES (trees_seq.nextval,
'tree5',sdo_geometry (2003, null, null,
  sdo_elem_info_array (1,1003,4),
  sdo_ordinate_array (8,10, 9,9, 8,8)));
INSERT INTO trees (id, name, t_g) VALUES (trees_seq.nextval,
'tree6',sdo_geometry (2003, null, null,
  sdo_elem_info_array (1,1003,4),
  sdo_ordinate_array (13,10, 14,9, 13,8)));
INSERT INTO trees (id, name, t_g) VALUES (trees_seq.nextval,
'tree7',sdo_geometry (2003, null, null,
  sdo_elem_info_array (1,1003,4),
  sdo_ordinate_array (13,5, 14,4, 13,3)));
INSERT INTO trees (id, name, t_g) VALUES (trees_seq.nextval,
'tree8',sdo_geometry (2003, null, null,
  sdo_elem_info_array (1,1003,4),
```

```
  sdo_ordinate_array (20,5, 21,4, 20,3)));
INSERT INTO trees (id, name, t_g) VALUES (trees_seq.nextval,
'tree9',sdo_geometry (2003, null, null,
  sdo_elem_info_array (1,1003,4),
  sdo_ordinate_array (20,10, 21,9, 20,8)));
INSERT INTO trees (id, name, t_g) VALUES (trees_seq.nextval,
'tree10',sdo_geometry (2003, null, null,
  sdo_elem_info_array (1,1003,4),
  sdo_ordinate_array (20,15, 21,14, 20,13)));
INSERT INTO trees (id, name, t_g) VALUES (trees_seq.nextval,
'tree11',sdo_geometry (2003, null, null,
  sdo_elem_info_array (1,1003,4),
  sdo_ordinate_array (20,20, 21,19, 20,18)));
INSERT INTO trees (id, name, t_g) VALUES (trees_seq.nextval,
'tree12',sdo_geometry (2003, null, null,
  sdo_elem_info_array (1,1003,4),
  sdo_ordinate_array (20,25, 21,24, 20,23)));
INSERT INTO trees (id, name, t_g) VALUES (trees_seq.nextval,
'tree13',sdo_geometry (2003, null, null,
  sdo_elem_info_array (1,1003,4),
  sdo_ordinate_array (20,30, 21,29, 20,28)));
```

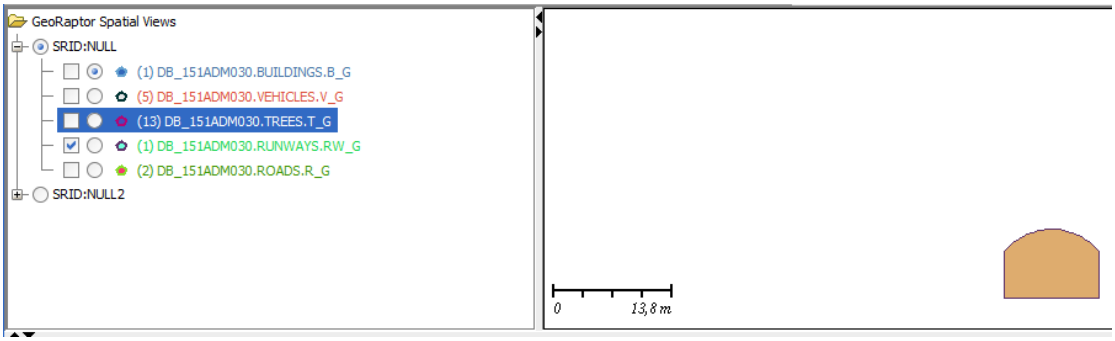


Trees graphical display

Runways

This table uses compound polygons (to include curves) to draw runways, runways are then drawn according to their different point's coordinates as well as the coordinates describing the curved line: example: sdo_ordinate_array (1,5.5, 1,0, 12,0, 12,5.5, 6,8, 1,5.5).

```
INSERT INTO runways (id, name, rw_g) VALUES (runways_seq.nextval,
'runway1',sdo_geometry (2003, null, null,
  sdo_elem_info_array (1,1005,2, 1,2,1, 7,2,2),
  sdo_ordinate_array (1,5.5, 1,0, 12,0, 12,5.5, 6,8, 1,5.5)));
```



Runway graphical display

Vehicles

This table uses simple polygons to draw vehicles, vehicles are then drawn according to their different point's coordinates: example: sdo_ordinate_array (17,4, 18,4, 18,7, 17,7, 17,4).

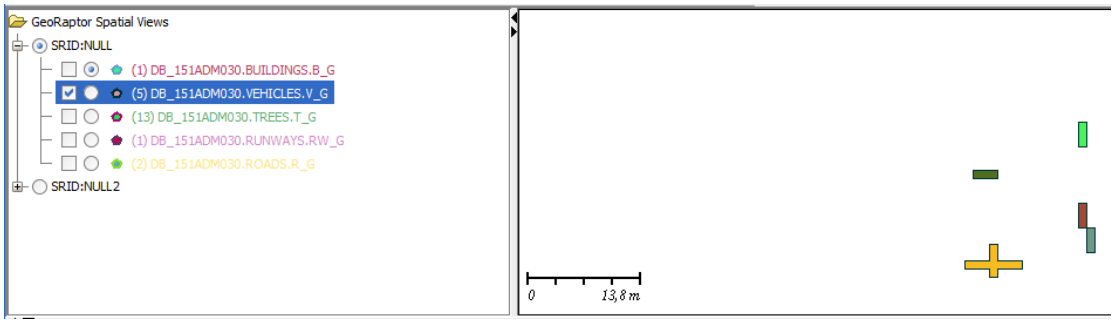
```
INSERT INTO vehicles (id, name, v_g) VALUES (vehicles_seq.nextval,
'car1',sdo_geometry (2003, null, null,
  sdo_elem_info_array (1,1003,1),
  sdo_ordinate_array (17,4, 18,4, 18,7, 17,7, 17,4)));

INSERT INTO vehicles (id, name, v_g) VALUES (vehicles_seq.nextval,
'car2',sdo_geometry (2003, null, null,
  sdo_elem_info_array (1,1003,1),
  sdo_ordinate_array (16,7, 17,7, 17,10, 16,10, 16,7)));

INSERT INTO vehicles (id, name, v_g) VALUES (vehicles_seq.nextval,
'car3',sdo_geometry (2003, null, null,
  sdo_elem_info_array (1,1003,1),
  sdo_ordinate_array (16,17, 17,17, 17,20, 16,20, 16,17)));

INSERT INTO vehicles (id, name, v_g) VALUES (vehicles_seq.nextval,
'car4',sdo_geometry (2003, null, null,
  sdo_elem_info_array (1,1003,1),
  sdo_ordinate_array (3,13, 6,13, 6,14, 3,14, 3,13)));

INSERT INTO vehicles (id, name, v_g) VALUES (vehicles_seq.nextval,
'plane1',sdo_geometry (2003, null, null,
  sdo_elem_info_array (1,1003,1),
  sdo_ordinate_array (2,3, 2,2, 5,2, 5,1, 6,1, 6,2, 9,2, 9,3, 6,3,
6,5, 5,5, 5,3,2,3)));
```



Vehicles graphical display

Indexes creation

In order to be displayed correctly, layouts have to be indexed, therefore indexes are created, they describe the way layers are displayed relatively to each other.

SQL code

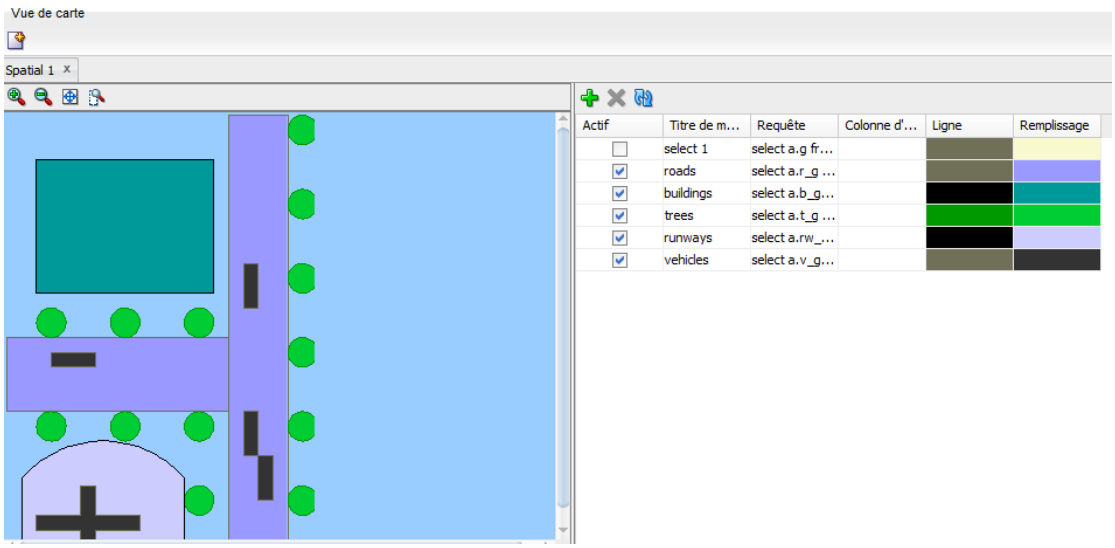
```
create index IND_buildings on buildings(b_g) indextype is
MDSYS.SPATIAL_INDEX;
create index IND_trees on trees(t_g) indextype is
MDSYS.SPATIAL_INDEX;
create index IND_runways on runways(rw_g) indextype is
MDSYS.SPATIAL_INDEX;
create index IND_vehicles on vehicles(v_g) indextype is
MDSYS.SPATIAL_INDEX;
create index IND_roads on roads(r_g) indextype is
MDSYS.SPATIAL_INDEX;
```

Result

There is two ways to display the results: the map view thanks to SQLDeveloper and the GeoRaptor display.

SQLDeveloper map view

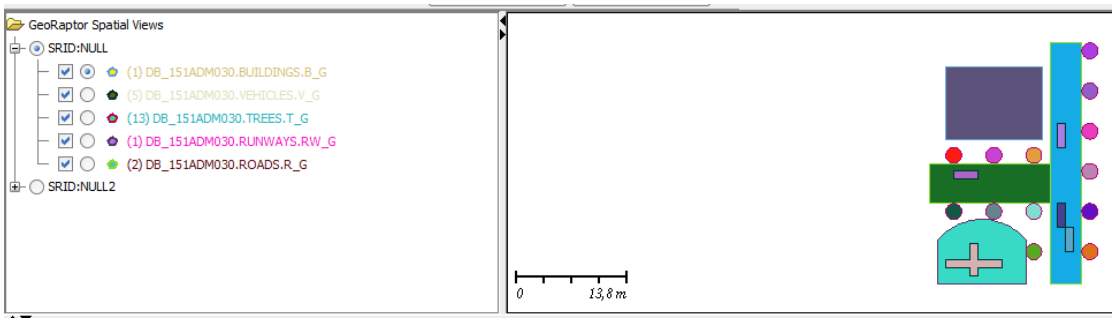
To achive this result, the querries details are:



Map view according to SQLDeveloper

true	roads	select a.r_g from roads a	java.awt.Color[r=112,g=112, b=89]	java.awt.Color[r=153,g=153,b =255]
true	buildings	select a.b_g from buildings a	java.awt.Color[r=0,g=0, b=0]	java.awt.Color[r=0,g=153,b =153]
true	trees	select a.t_g from trees a	java.awt.Color[r=0,g=153,b =0]	java.awt.Color[r=0,g=204,b= 51]
true	runways	select a.rw_g from runways a	java.awt.Color[r=0,g=0, b=0]	java.awt.Color[r=204,g=204, b=255]
true	vehicles	select a.v_g from vehicles a	java.awt.Color[r=112,g=11 2,b=89]	java.awt.Color[r=51,g=51, b=51]

GeoRaptor view



GeoRaptor simultaneous display of all tables

Observations

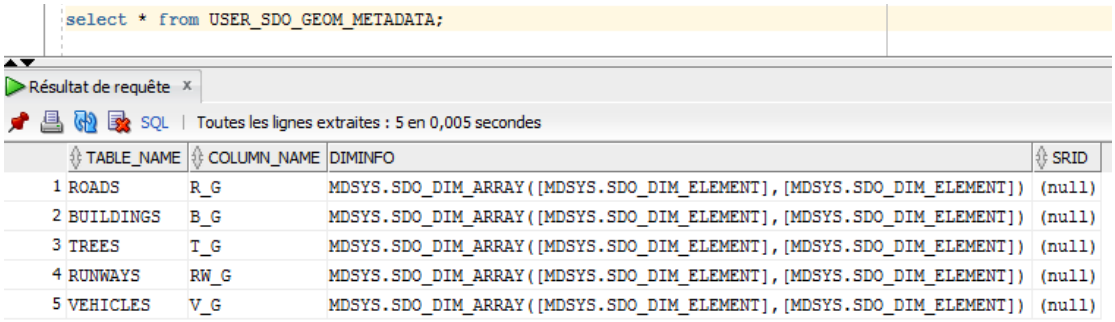
We can see according to these two display that the GeoRaptor tool provides additional informations by providing the scale of the display, it also allows to draw figures.

Nevertheless it seems to be a little problem of compatibility somewhere between GeoRaptor and this version of Java resulting an unexpectingly changing color display of all items in the same layout.

Spacial queries

Inspect metadata

```
select * from USER_SDO_GEOM_METADATA;
```



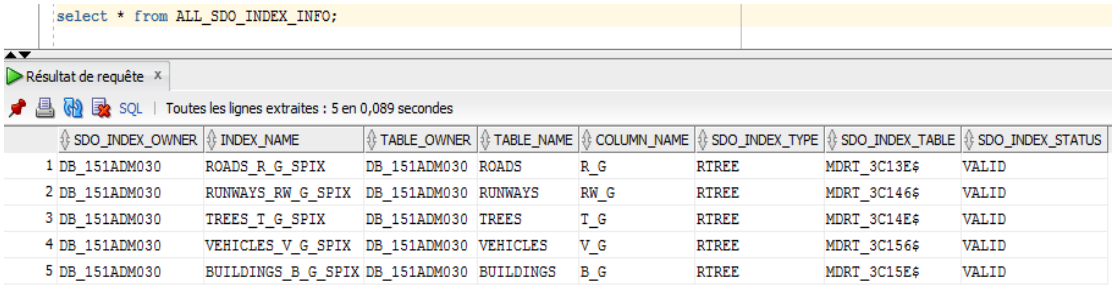
Résultat de requête | SQL | Toutes les lignes extraites : 5 en 0,005 secondes

	TABLE_NAME	COLUMN_NAME	DIMINFO	SRID
1	ROADS	R_G	MDSYS.SDO_DIM_ARRAY([MDSYS.SDO_DIM_ELEMENT],[MDSYS.SDO_DIM_ELEMENT])	(null)
2	BUILDINGS	B_G	MDSYS.SDO_DIM_ARRAY([MDSYS.SDO_DIM_ELEMENT],[MDSYS.SDO_DIM_ELEMENT])	(null)
3	TREES	T_G	MDSYS.SDO_DIM_ARRAY([MDSYS.SDO_DIM_ELEMENT],[MDSYS.SDO_DIM_ELEMENT])	(null)
4	RUNWAYS	RW_G	MDSYS.SDO_DIM_ARRAY([MDSYS.SDO_DIM_ELEMENT],[MDSYS.SDO_DIM_ELEMENT])	(null)
5	VEHICLES	V_G	MDSYS.SDO_DIM_ARRAY([MDSYS.SDO_DIM_ELEMENT],[MDSYS.SDO_DIM_ELEMENT])	(null)

We can see here that all tables have a line in the geometrical meta data and their geomety object is also present.

Inspect indexes

```
select * from ALL_SDO_INDEX_INFO;
```



Résultat de requête | SQL | Toutes les lignes extraites : 5 en 0,089 secondes

	SDO_INDEX_OWNER	INDEX_NAME	TABLE_OWNER	TABLE_NAME	COLUMN_NAME	SDO_INDEX_TYPE	SDO_INDEX_TABLE	SDO_INDEX_STATUS
1	DB_151ADM030	ROADS_R_G_SPIX	DB_151ADM030	ROADS	R_G	RTREE	MDRT_3C13E$	VALID
2	DB_151ADM030	RUNWAYS_RW_G_SPIX	DB_151ADM030	RUNWAYS	RW_G	RTREE	MDRT_3C146$	VALID
3	DB_151ADM030	TREES_T_G_SPIX	DB_151ADM030	TREES	T_G	RTREE	MDRT_3C14E$	VALID
4	DB_151ADM030	VEHICLES_V_G_SPIX	DB_151ADM030	VEHICLES	V_G	RTREE	MDRT_3C156$	VALID
5	DB_151ADM030	BUILDINGS_B_G_SPIX	DB_151ADM030	BUILDINGS	B_G	RTREE	MDRT_3C15E$	VALID

We can see here that all tables have a line in the indexes table, we can also notice that all trees are RTREE, the default type.

Finding which figures are interacting wih the figure named Road1

```
SELECT r.name, v.name
FROM vehicles v, roads r
WHERE r.NAME = 'road1' AND
SDO_FILTER(v.v_g, r.r_g) = 'TRUE';
```

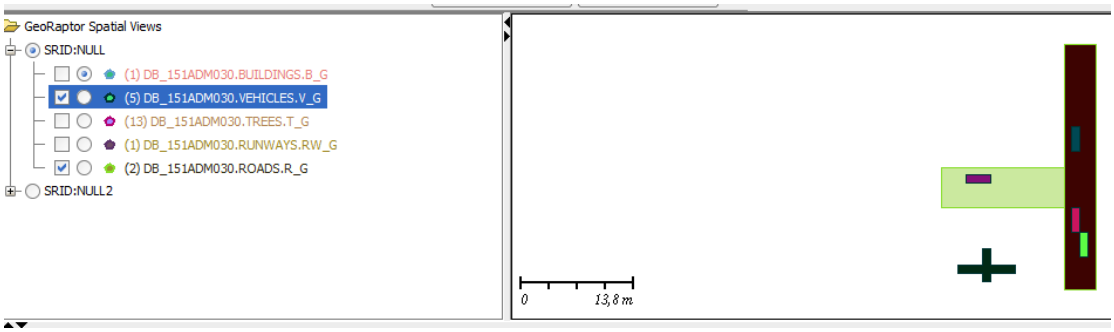
```
SELECT r.name, v.name
FROM vehicles v, roads r
WHERE r.NAME = 'road1' AND
SDO_FILTER(v.v_g, r.r_g) = 'TRUE';
```

Résultat de requête

SQL | Toutes les lignes extraites : 3 en 0,461 secondes

	NAME	NAME_1
1	road1	car3
2	road1	car2
3	road1	car1

We can see that cars number 1, 2 and 3 are interacting with the road1.

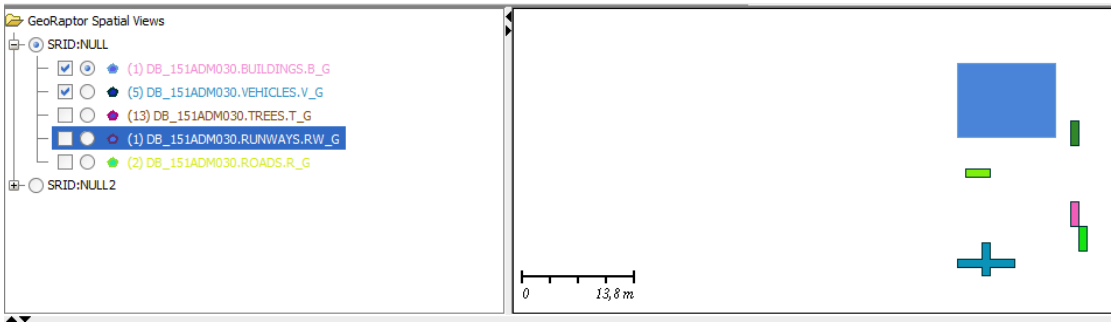


Indeed, they do, road1 is the one on the right and cars 1, 2 and 3 are the left ones too.

Computing distance between two items

Distance between the building and all vehicles

```
SELECT v.NAME, B.NAME, SDO_GEOM.SDO_DISTANCE(v.v_g,b.b_g,0.0005) AS
distance
FROM vehicles V,buildings B;
```



```
SELECT v.NAME, B.NAME, SDO_GEOM.SDO_DISTANCE(v.v_g,b.b_g,0.0005) AS distance
FROM vehicles V,buildings B;
```

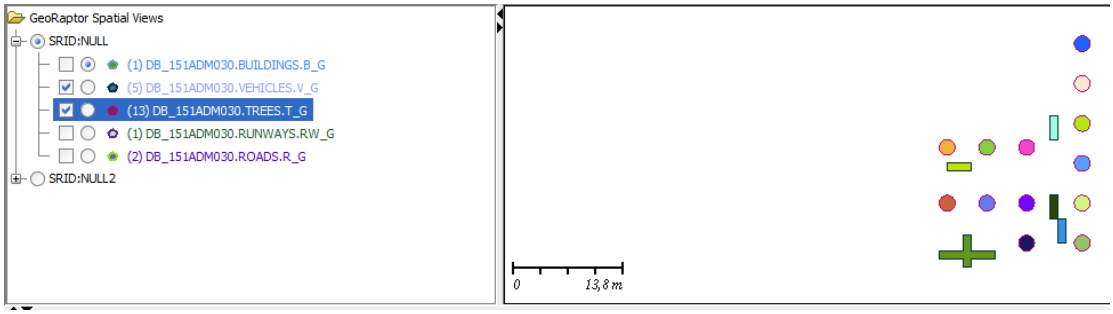
Résultat de requête

SQL | Toutes les lignes extraites : 5 en 0,352 secondes

	NAME	NAME_1	DISTANCE
1	car1	building1	11,4017542509914
2	car2	building1	8,24621125123532
3	car3	building1	2
4	car4	building1	4
5	plane1	building1	13

Distance between the plane and all trees

```
SELECT v.NAME, t.NAME, SDO_GEOM.SDO_DISTANCE(v.v_g,t.t_g,0.0005) AS
distance
FROM vehicles V,trees t
WHERE v.name='plane1';
```



```
SELECT v.NAME, t.NAME, SDO_GEOM.SDO_DISTANCE(v.v_g,t.t_g,0.0005) AS distance
FROM vehicles V,trees t
WHERE v.name='plane1';
```

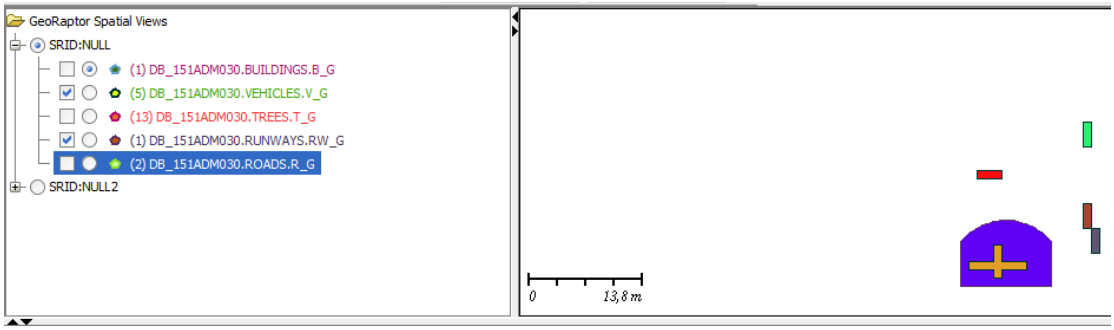
Résultat de requête ×

SQL | Toutes les lignes extraites : 13 en 0,019 secondes

	NAME	NAME_1	DISTANCE
1	plane1	tree1	10,1803398874989
2	plane1	tree2	10,1803398874989
3	plane1	tree3	12,0384048104053
4	plane1	tree4	3,47213595499958
5	plane1	tree5	3,47213595499958
6	plane1	tree6	6,21110255092798
7	plane1	tree7	3,12310562561766
8	plane1	tree8	10,0453610171873
9	plane1	tree9	11,5299640861417
10	plane1	tree10	14,556349186104
11	plane1	tree11	18,4164878389476
12	plane1	tree12	22,6008474424119
13	plane1	tree13	26,7848879788996

Vehicles and runways which has at least one intercation of any type:

```
SELECT v.name,r.name
FROM vehicles v, runways r
WHERE SDO_RELATE(v.v_g, r.rw_g, 'MASK=ANYINTERACT')='TRUE';
```



```
SELECT v.name,r.name
FROM vehicles v, runways r
WHERE SDO_RELATE(v.v_g, r.rw_g, 'MASK=ANYINTERACT')='TRUE';
```

Résultat de requête

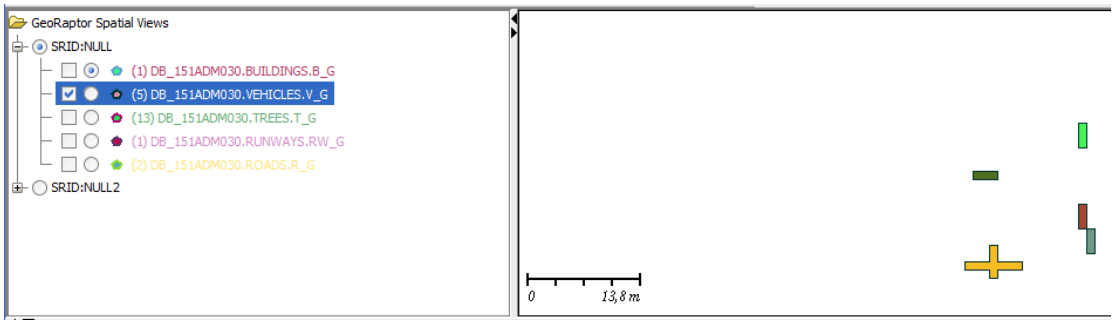
SQL | Toutes les lignes extraites : 1 en 0,227 secondes

	NAME	NAME_1
1	plane1	runway1

Items which are touching each other:

Vehicles which are touching each other

```
SELECT v.name, w.name
FROM vehicles v, vehicles w
WHERE MDSYS.SDO_TOUCH(v.v_g, w.v_g) = 'TRUE';
```




```
SELECT v.name, w.name
FROM vehicles v, vehicles w
WHERE MDSYS.SDO_TOUCH(v.v_g, w.v_g) = 'TRUE';
```

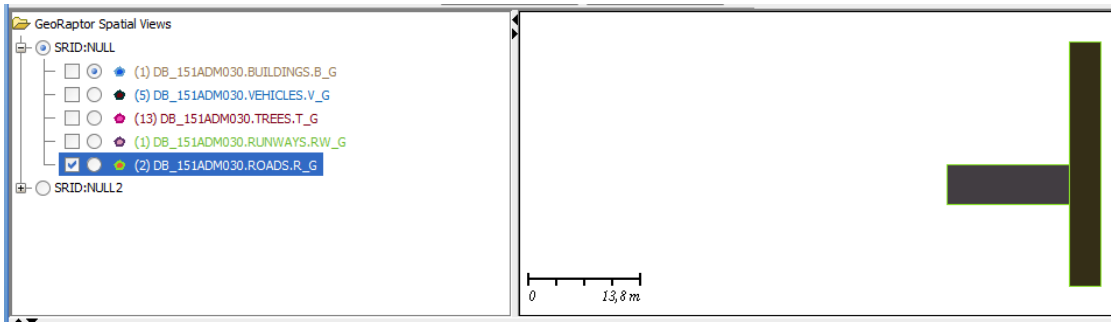
Résultat de requête

SQL | Toutes les lignes extraites : 2 en 0,206 secondes

	NAME	NAME_1
1	car2	car1
2	car1	car2

Roads which are touching each other

```
SELECT r.name, s.name
FROM roads r, roads s
WHERE MDSYS.SDO_TOUCH(r.r_g, s.r_g) = 'TRUE';
```




```
SELECT r.name, s.name
FROM roads r, roads s
WHERE MDSYS.SDO_TOUCH(r.r_g, s.r_g) = 'TRUE';
```

Résultat de requête

SQL | Toutes les lignes extraites : 2 en 0,271 secondes

	NAME	NAME_1
1	road2	road1
2	road1	road2

Links

1st page picture:

http://www.localtrips.net/