RIGA TECHNICAL UNIVERSITY

Faculty of computer science and information technology

Master’s Programme in Business Informatics

1862

RĪGAS TEHNISKĀ UNIVERSITĀTE

LABORATORY WORK No. x

„Spatial Database”

„Advanced data technologies”

Author: Axel DECLERCQ

230AEM015

Checked: Lecturer Ainars Auzins

2023/2024

Table of contents

1. Farm visualization

I decided to recreate a farm scene. I tried to add some contents such as animal and trees. Here is my realization :

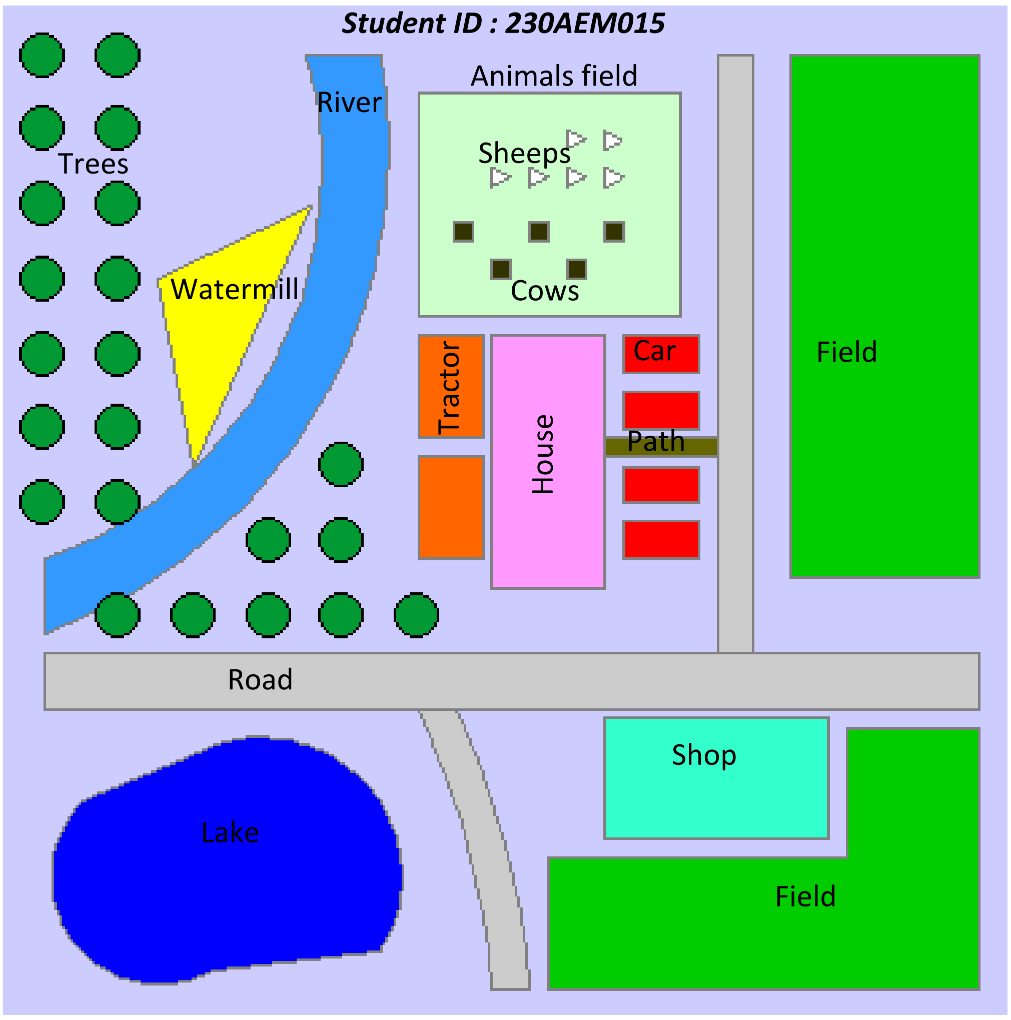


Figure 1. Farm visualization

2. Layers definition

1. **Buildings** : House, Watermill, Shop
2. **Roads** : Road, Path
3. **Nature** : Field, Lake, Trees
4. **Animal** : Cows, Sheep
5. **Vehicles** : Tractor, Car

3. Table creation

Firstly, I created a table for each layer. In each table there is GEO column with type MDSYS.SDO_GEOMETRY that gives information about the geometry shape of the objects. Some table are linked to an "info table" that gives additional informations. The conceptual class diagram is given in the figure 2.

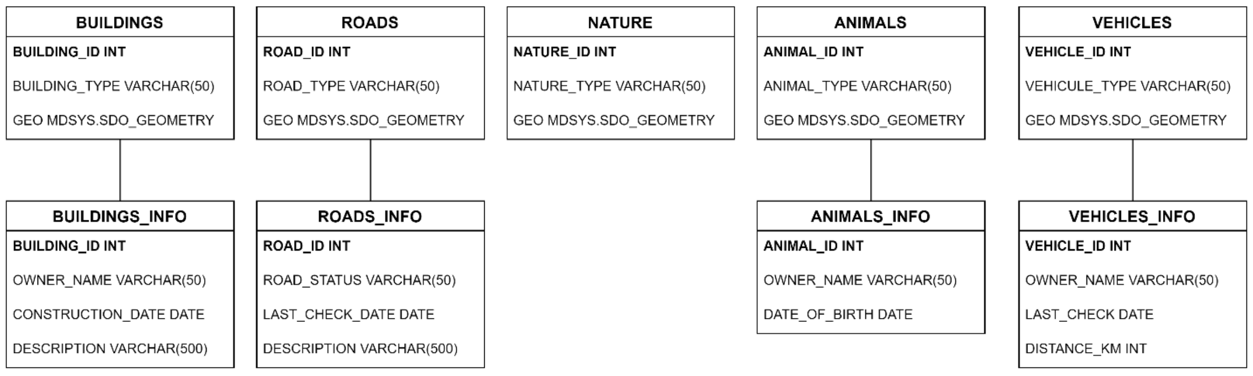
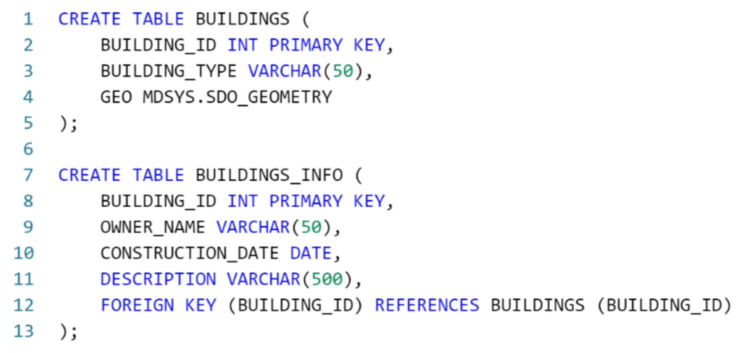


Figure 2. Conceptual class diagram

3.1 Buildings

In the bustling world of my spatial drawing, the "BULDINGS" layer stands tall, featuring iconic structures like Houses, Farms, Windmills, and Shops. Each edifice is a canvas of possibilities, embodying the essence of diverse human activities and lifestyles. To unravel the stories behind these architectural marvels, we delve into the corresponding "BUILDINGS_INFO" table, where details about owners, construction dates, and unique descriptions await.

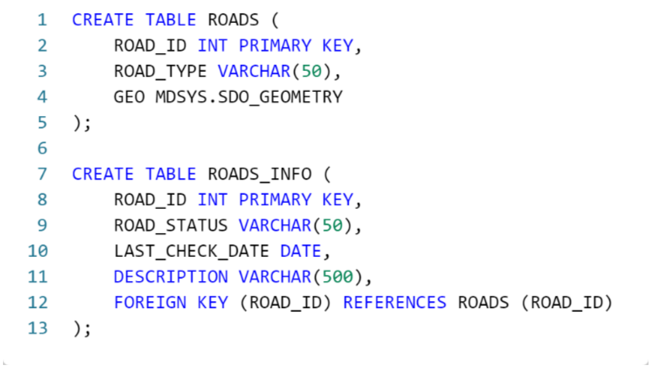


```
CREATE TABLE BUILDINGS (
    BUILDING_ID INT PRIMARY KEY,
    BUILDING_TYPE VARCHAR(50),
    GEO MDSYS.SDO_GEOMETRY
);

CREATE TABLE BUILDINGS_INFO (
    BUILDING_ID INT PRIMARY KEY,
    OWNER_NAME VARCHAR(50),
    CONSTRUCTION_DATE DATE,
    DESCRIPTION VARCHAR(500),
    FOREIGN KEY (BUILDING_ID) REFERENCES BUILDINGS (BUILDING_ID)
);
```

Figure 3.1. Creation of the buildings tables

3.2 Roads

The intricate network of pathways and thoroughfares is captured in the "ROADS" layer, showcasing the dynamic movement within my spatial creation. Here, Roads and Paths intertwine, guiding the flow of life. The accompanying "ROADS_INFO" table unveils essential insights into road statuses, last check dates, and detailed descriptions, adding depth to the narrative of this transport infrastructure.

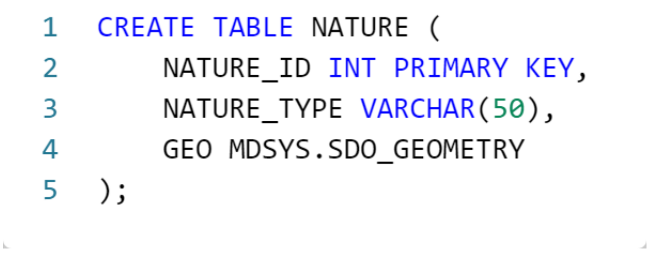


```sql
CREATE TABLE ROADS (
    ROAD_ID INT PRIMARY KEY,
    ROAD_TYPE VARCHAR(50),
    GEO MDSYS.SDO_GEOMETRY
);

CREATE TABLE ROADS_INFO (
    ROAD_ID INT PRIMARY KEY,
    ROAD_STATUS VARCHAR(50),
    LAST_CHECK_DATE DATE,
    DESCRIPTION VARCHAR(500),
    FOREIGN KEY (ROAD_ID) REFERENCES ROADS (ROAD_ID)
);
```

Figure 3.2. Creation of the roads tables

3.3 Nature

Nature takes center stage in the "NATURE" layer, adorned with serene Lakes, lush Trees, and picturesque Hay. Each element reflects the tranquility and beauty of the environment.

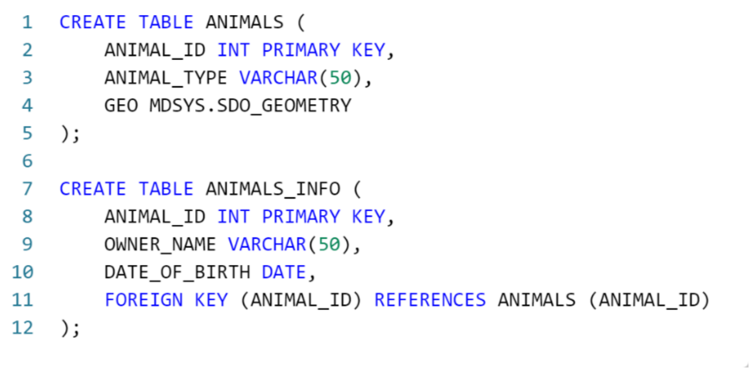


```sql
CREATE TABLE NATURE (
    NATURE_ID INT PRIMARY KEY,
    NATURE_TYPE VARCHAR(50),
    GEO MDSYS.SDO_GEOMETRY
);
```

Figure 3.3. Creation of the NATURE table

3.4 Animals

In the heart of my spatial canvas, the "ANIMAL" layer introduces animated life with the inclusion of Cows and Sheep. These adorable creatures bring vitality to the scene, engaging in their everyday activities. As animals are inherently expressive, their charm speaks for itself without the need for additional information in an accompanying info table.

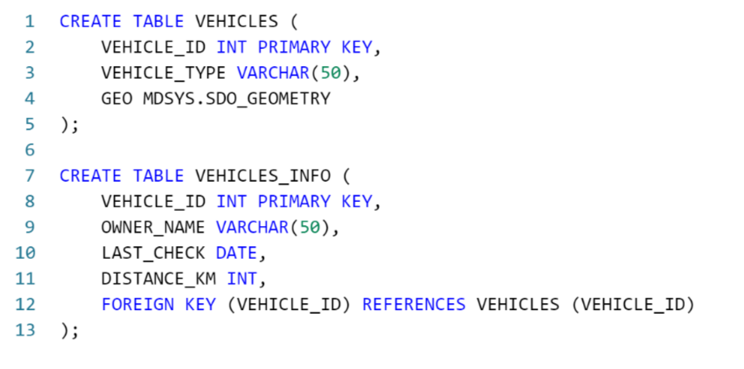


```sql
CREATE TABLE ANIMALS (
    ANIMAL_ID INT PRIMARY KEY,
    ANIMAL_TYPE VARCHAR(50),
    GEO MDSYS.SDO_GEOMETRY
);

CREATE TABLE ANIMALS_INFO (
    ANIMAL_ID INT PRIMARY KEY,
    OWNER_NAME VARCHAR(50),
    DATE_OF_BIRTH DATE,
    FOREIGN KEY (ANIMAL_ID) REFERENCES ANIMALS (ANIMAL_ID)
);
```

Figure 3.4. Creation of the animals tables

3.5 Vehicles

The "VEHICLES" layer injects a sense of motion and utility, featuring essential modes of transport like Tractors and Cars. These dynamic elements symbolize progress and connectivity. To unravel the stories behind these vehicles, the "VEHICLES_INFO" table takes the wheel, providing details about owners, last check dates, and distances covered – a testament to the evolving landscape within my spatial creation.

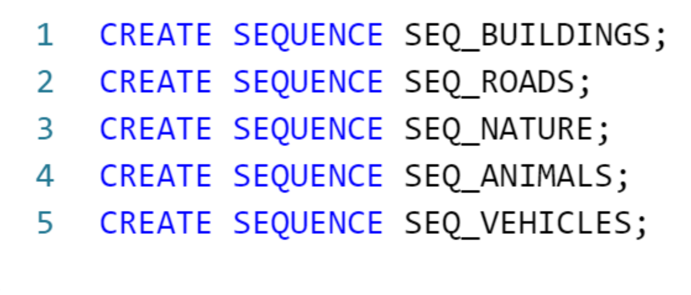


```sql
CREATE TABLE VEHICLES (
    VEHICLE_ID INT PRIMARY KEY,
    VEHICLE_TYPE VARCHAR(50),
    GEO MDSYS.SDO_GEOMETRY
);

CREATE TABLE VEHICLES_INFO (
    VEHICLE_ID INT PRIMARY KEY,
    OWNER_NAME VARCHAR(50),
    LAST_CHECK DATE,
    DISTANCE_KM INT,
    FOREIGN KEY (VEHICLE_ID) REFERENCES VEHICLES (VEHICLE_ID)
);
```

Figure 3.5. Creation of the vehicles tables

3.6 Sequences

As I used ID for object line of my tables, I decided to use sequences to create them.



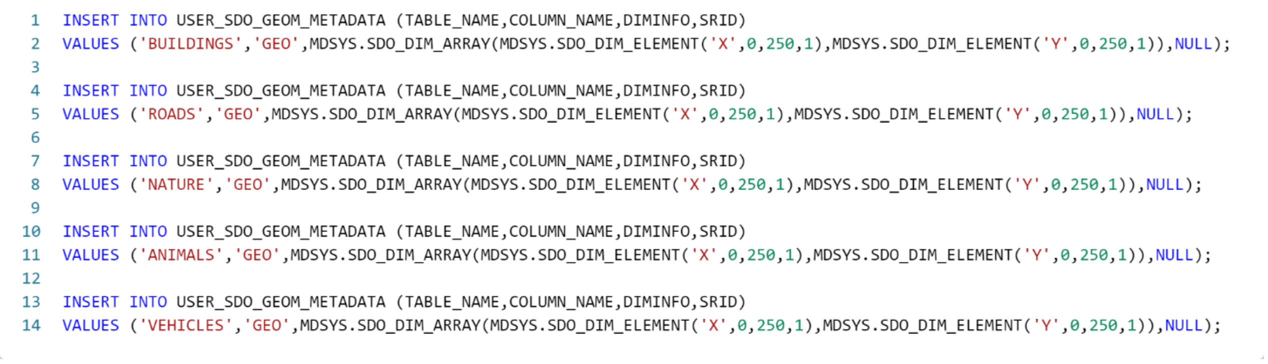
```sql
CREATE SEQUENCE SEQ_BUILDINGS;
CREATE SEQUENCE SEQ_ROADS;
CREATE SEQUENCE SEQ_NATURE;
CREATE SEQUENCE SEQ_ANIMALS;
CREATE SEQUENCE SEQ_VEHICLES;
```

Figure 3.5. Creation of the sequences

4. Metadata

Metadata, in the context of spatial databases, refers to information that describes the spatial data itself. This information enables the understanding and interpretation of the geospatial characteristics of the objects stored in the database. In an Oracle Spatial database, metadata related to spatial objects is typically stored in the Oracle data dictionary. Specifically, it is stored in system tables that are part of the MDSYS schema, which is the system schema for Oracle Spatial. In this project, metadata will be store in USER_SDO_GEOM_METADATA. This table contains information about geographic objects stored in the database, such as the table name, geometric column, coordinate system, geometry type, etc.

The information stored in USER_SDO_GEOM_METADATA for each layer sets spatial boundaries for each object in each table. Specifically, as it is a 2D visualization task, I have chosen to work within a data range from 0 to 250 for both x and y coordinates. These boundaries help control and standardize the placement of spatial objects in my database, ensuring consistency and accurate interpretation of spatial data. These constraints are defined by the dimensions and resolution specified in DIMINFO during the insertion of metadata.



```sql
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME,COLUMN_NAME,DIMINFO,SRID)
VALUES ('BUILDINGS','GEO',MDSYS.SDO_DIM_ARRAY(MDSYS.SDO_DIM_ELEMENT('X',0,250,1),MDSYS.SDO_DIM_ELEMENT('Y',0,250,1)),NULL);

INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME,COLUMN_NAME,DIMINFO,SRID)
VALUES ('ROADS','GEO',MDSYS.SDO_DIM_ARRAY(MDSYS.SDO_DIM_ELEMENT('X',0,250,1),MDSYS.SDO_DIM_ELEMENT('Y',0,250,1)),NULL);

INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME,COLUMN_NAME,DIMINFO,SRID)
VALUES ('NATURE','GEO',MDSYS.SDO_DIM_ARRAY(MDSYS.SDO_DIM_ELEMENT('X',0,250,1),MDSYS.SDO_DIM_ELEMENT('Y',0,250,1)),NULL);

INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME,COLUMN_NAME,DIMINFO,SRID)
VALUES ('ANIMALS','GEO',MDSYS.SDO_DIM_ARRAY(MDSYS.SDO_DIM_ELEMENT('X',0,250,1),MDSYS.SDO_DIM_ELEMENT('Y',0,250,1)),NULL);

INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME,COLUMN_NAME,DIMINFO,SRID)
VALUES ('VEHICLES','GEO',MDSYS.SDO_DIM_ARRAY(MDSYS.SDO_DIM_ELEMENT('X',0,250,1),MDSYS.SDO_DIM_ELEMENT('Y',0,250,1)),NULL);
```

Figure 4. Creation of the metadata

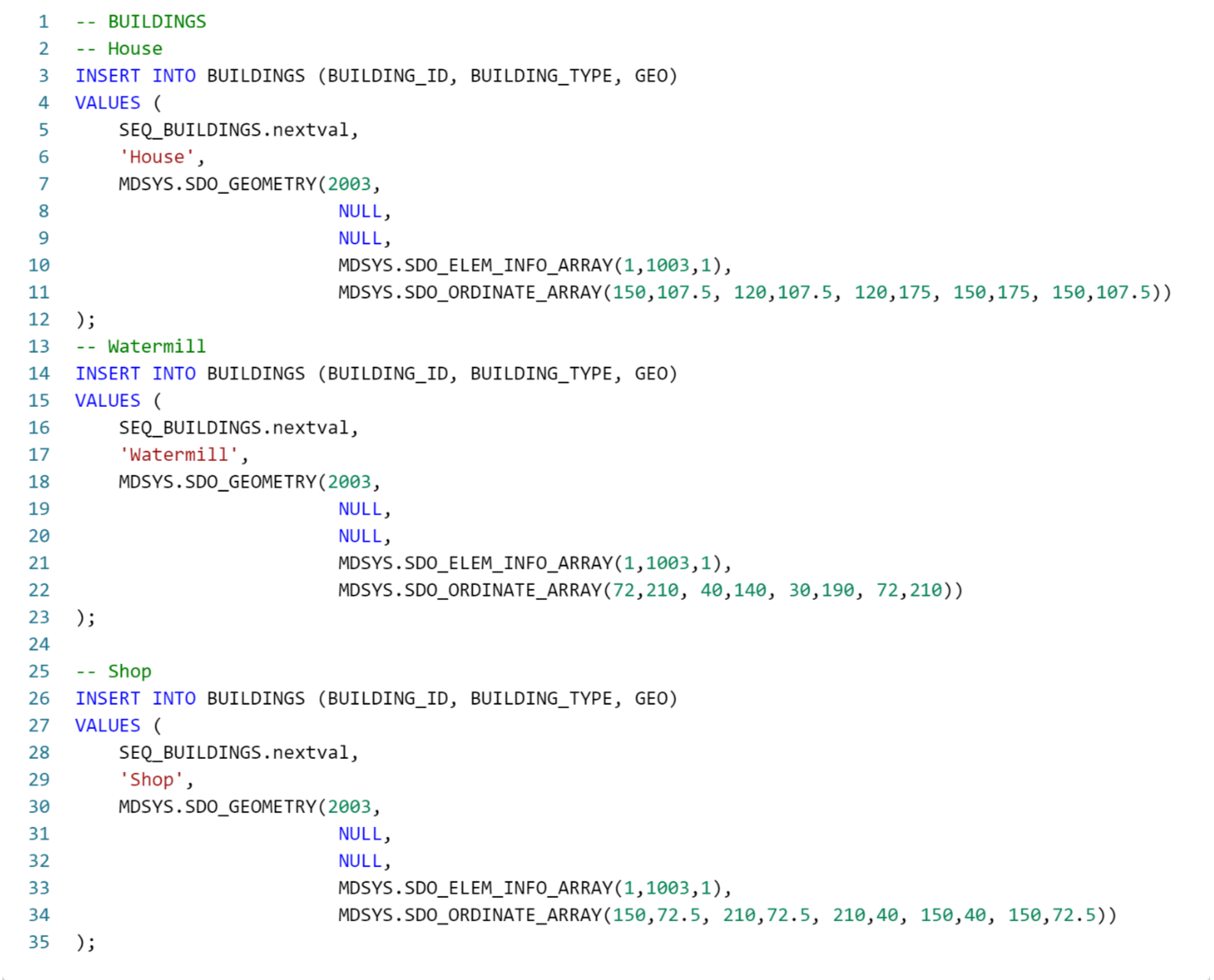
Since my data is independent of any specific spatial reference system and I do not need to explicitly associate an SRID, leaving the field as NULL is a valid option.

5. Values insertion

The meta I defined previously set a square limitation of a shape 250 x 250. All the objects must be in this square.

BUILDINGS:

This layer contains one house in a shape of a rectangle, one watermill in shape of triangle that is aside the river and a shope in a shape of rectangle.

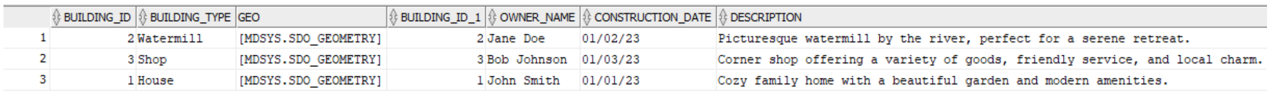


```sql
-- BUILDINGS
-- House
INSERT INTO BUILDINGS (BUILDING_ID, BUILDING_TYPE, GEO)
VALUES (
    SEQ_BUILDINGS.nextval,
    'House',
    MDSYS.SDO_GEOMETRY(2003,
                        NULL,
                        NULL,
                        MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),
                        MDSYS.SDO_ORDINATE_ARRAY(150,107.5, 120,107.5, 120,175, 150,175, 150,107.5))
);
-- Watermill
INSERT INTO BUILDINGS (BUILDING_ID, BUILDING_TYPE, GEO)
VALUES (
    SEQ_BUILDINGS.nextval,
    'Watermill',
    MDSYS.SDO_GEOMETRY(2003,
                        NULL,
                        NULL,
                        MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),
                        MDSYS.SDO_ORDINATE_ARRAY(72,210, 40,140, 30,190, 72,210))
);

-- Shop
INSERT INTO BUILDINGS (BUILDING_ID, BUILDING_TYPE, GEO)
VALUES (
    SEQ_BUILDINGS.nextval,
    'Shop',
    MDSYS.SDO_GEOMETRY(2003,
                        NULL,
                        NULL,
                        MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),
                        MDSYS.SDO_ORDINATE_ARRAY(150,72.5, 210,72.5, 210,40, 150,40, 150,72.5))
);
```

Figure 5.1.1. Building values insertion

```sql
--BUILDINGS INFO
-- House Info
INSERT INTO BUILDINGS_INFO (BUILDING_ID, OWNER_NAME, CONSTRUCTION_DATE, DESCRIPTION)
VALUES (
    (SELECT BUILDING_ID FROM BUILDINGS WHERE BUILDING_TYPE = 'House'),
    'John Smith',
    TO_DATE('2023-01-01', 'YYYY-MM-DD'),
    'Cozy family home with a beautiful garden and modern amenities.'
);
-- Watermill Info
INSERT INTO BUILDINGS_INFO (BUILDING_ID, OWNER_NAME, CONSTRUCTION_DATE, DESCRIPTION)
VALUES (
    (SELECT BUILDING_ID FROM BUILDINGS WHERE BUILDING_TYPE = 'Watermill'),
    'Jane Doe',
    TO_DATE('2023-02-01', 'YYYY-MM-DD'),
    'Picturesque watermill by the river, perfect for a serene retreat.'
);
-- Shop Info
INSERT INTO BUILDINGS_INFO (BUILDING_ID, OWNER_NAME, CONSTRUCTION_DATE, DESCRIPTION)
VALUES (
    (SELECT BUILDING_ID FROM BUILDINGS WHERE BUILDING_TYPE = 'Shop'),
    'Bob Johnson', -- Random owner name
    TO_DATE('2023-03-01', 'YYYY-MM-DD'),
    'Corner shop offering a variety of goods, friendly service, and local charm.'
);
```

Figure 5.1.2. Building_info values insertion

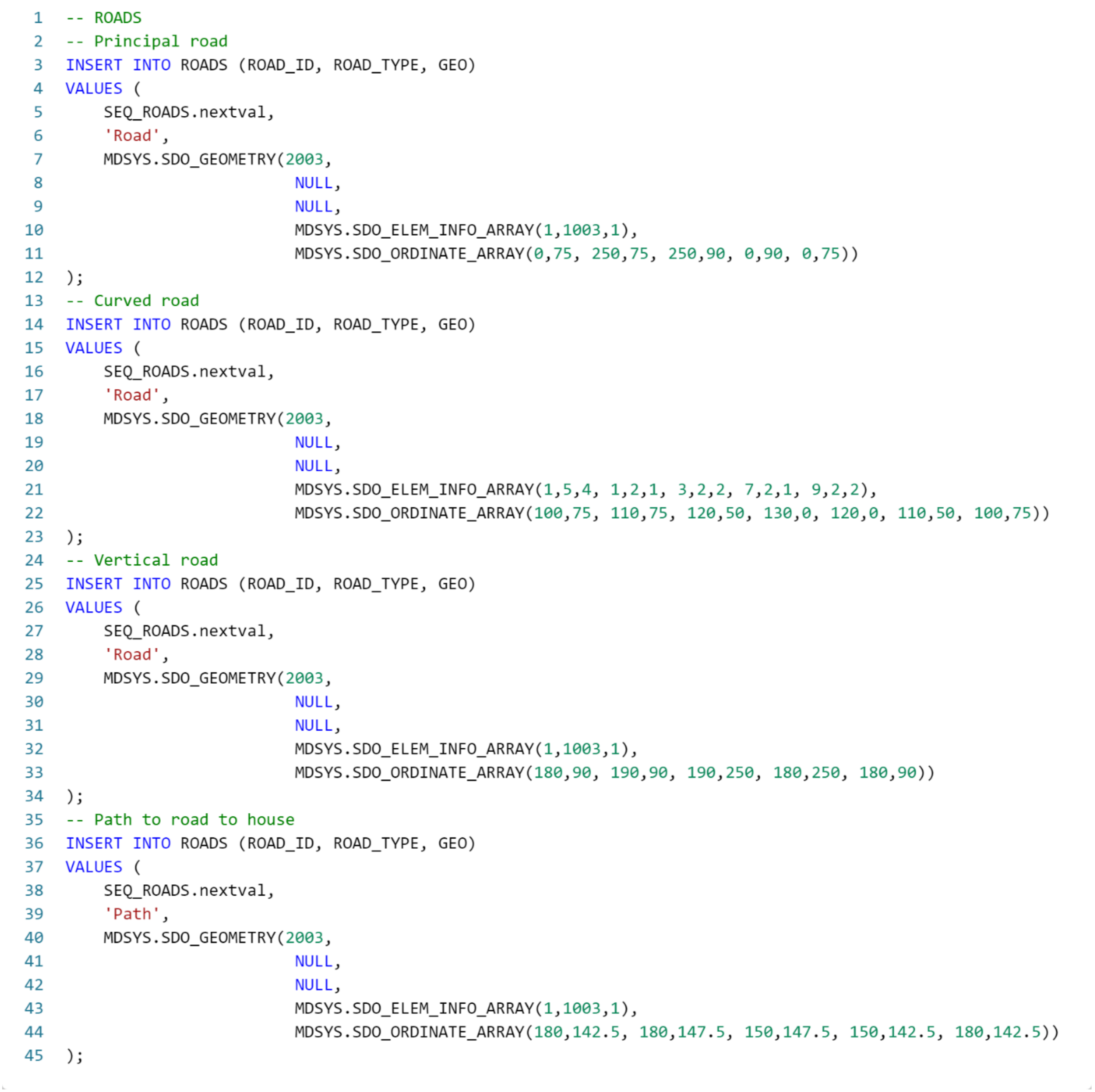


	BUILDING_ID	BUILDING_TYPE	GEO	BUILDING_ID_1	OWNER_NAME	CONSTRUCTION_DATE	DESCRIPTION
1	2	Watermill	[MDSYS.SDO_GEOMETRY]	2	Jane Doe	01/02/23	Picturesque watermill by the river, perfect for a serene retreat.
2	3	Shop	[MDSYS.SDO_GEOMETRY]	3	Bob Johnson	01/03/23	Corner shop offering a variety of goods, friendly service, and local charm.
3	1	House	[MDSYS.SDO_GEOMETRY]	1	John Smith	01/01/23	Cozy family home with a beautiful garden and modern amenities.

Figure 5.1.3 Buildings insertion validation

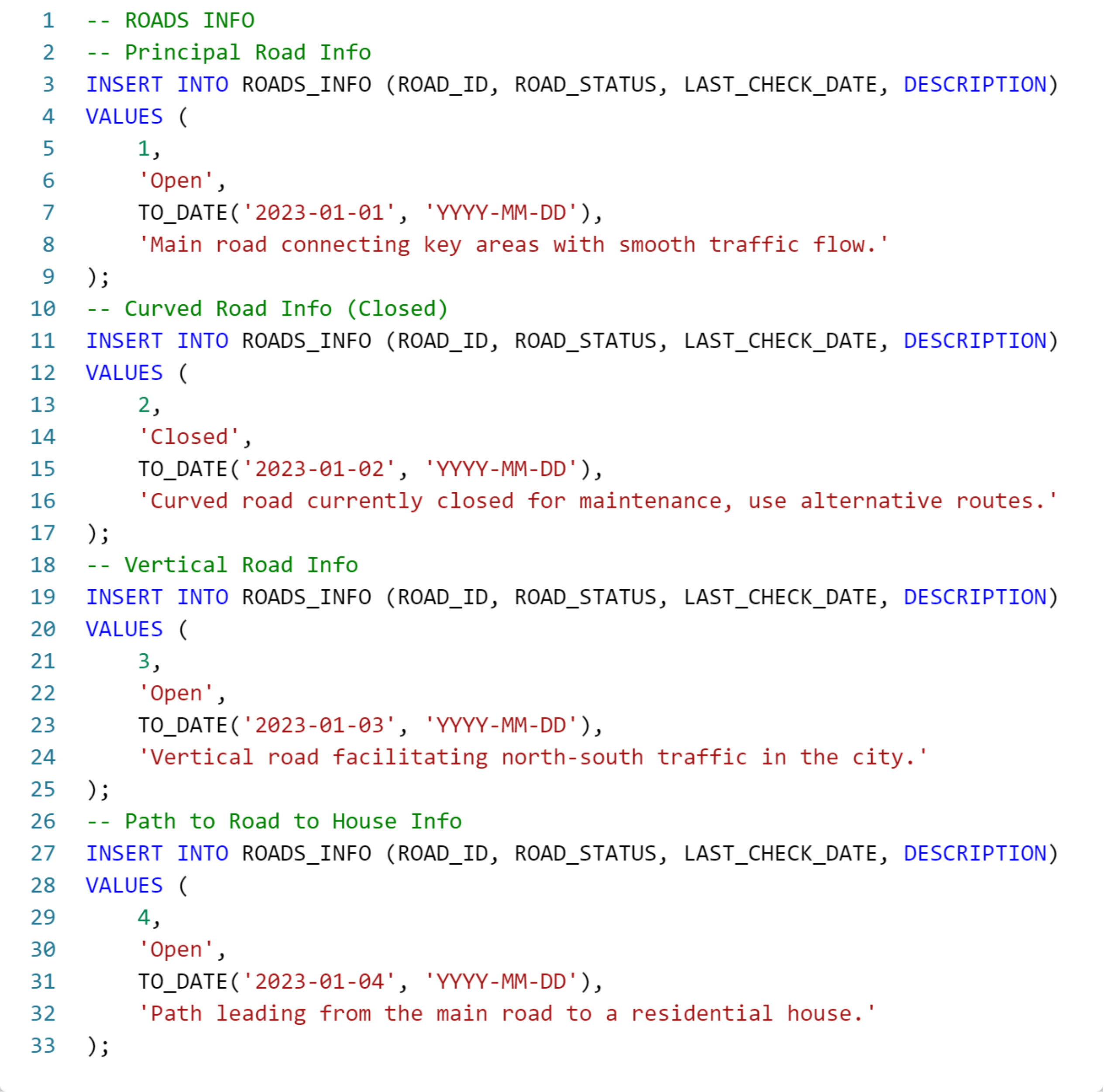
ROADS:

This layer contains the principal road that cross the limitation square horizontally, a vertical road to the top, a curved road to the bottom and a path to the house.



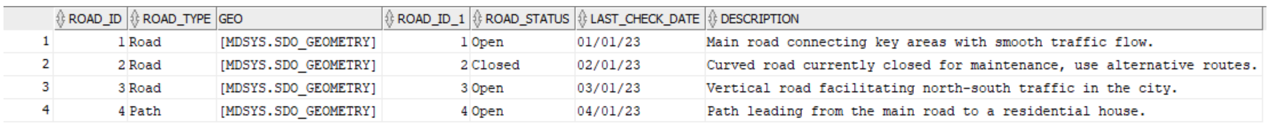
```sql
-- ROADS
-- Principal road
INSERT INTO ROADS (ROAD_ID, ROAD_TYPE, GEO)
VALUES (
    SEQ_ROADS.nextval,
    'Road',
    MDSYS.SDO_GEOMETRY(2003,
                        NULL,
                        NULL,
                        MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),
                        MDSYS.SDO_ORDINATE_ARRAY(0,75, 250,75, 250,90, 0,90, 0,75))
);
-- Curved road
INSERT INTO ROADS (ROAD_ID, ROAD_TYPE, GEO)
VALUES (
    SEQ_ROADS.nextval,
    'Road',
    MDSYS.SDO_GEOMETRY(2003,
                        NULL,
                        NULL,
                        MDSYS.SDO_ELEM_INFO_ARRAY(1,5,4, 1,2,1, 3,2,2, 7,2,1, 9,2,2),
                        MDSYS.SDO_ORDINATE_ARRAY(100,75, 110,75, 120,50, 130,0, 120,0, 110,50, 100,75))
);
-- Vertical road
INSERT INTO ROADS (ROAD_ID, ROAD_TYPE, GEO)
VALUES (
    SEQ_ROADS.nextval,
    'Road',
    MDSYS.SDO_GEOMETRY(2003,
                        NULL,
                        NULL,
                        MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),
                        MDSYS.SDO_ORDINATE_ARRAY(180,90, 190,90, 190,250, 180,250, 180,90))
);
-- Path to road to house
INSERT INTO ROADS (ROAD_ID, ROAD_TYPE, GEO)
VALUES (
    SEQ_ROADS.nextval,
    'Path',
    MDSYS.SDO_GEOMETRY(2003,
                        NULL,
                        NULL,
                        MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),
                        MDSYS.SDO_ORDINATE_ARRAY(180,142.5, 180,147.5, 150,147.5, 150,142.5, 180,142.5))
);
```

Figure 5.2.1. Road values insertion



```sql
-- ROADS INFO
-- Principal Road Info
INSERT INTO ROADS_INFO (ROAD_ID, ROAD_STATUS, LAST_CHECK_DATE, DESCRIPTION)
VALUES (
    1,
    'Open',
    TO_DATE('2023-01-01', 'YYYY-MM-DD'),
    'Main road connecting key areas with smooth traffic flow.'
);
-- Curved Road Info (Closed)
INSERT INTO ROADS_INFO (ROAD_ID, ROAD_STATUS, LAST_CHECK_DATE, DESCRIPTION)
VALUES (
    2,
    'Closed',
    TO_DATE('2023-01-02', 'YYYY-MM-DD'),
    'Curved road currently closed for maintenance, use alternative routes.'
);
-- Vertical Road Info
INSERT INTO ROADS_INFO (ROAD_ID, ROAD_STATUS, LAST_CHECK_DATE, DESCRIPTION)
VALUES (
    3,
    'Open',
    TO_DATE('2023-01-03', 'YYYY-MM-DD'),
    'Vertical road facilitating north-south traffic in the city.'
);
-- Path to Road to House Info
INSERT INTO ROADS_INFO (ROAD_ID, ROAD_STATUS, LAST_CHECK_DATE, DESCRIPTION)
VALUES (
    4,
    'Open',
    TO_DATE('2023-01-04', 'YYYY-MM-DD'),
    'Path leading from the main road to a residential house.'
);
```

Figure 5.2.2. Roads_info values insertion

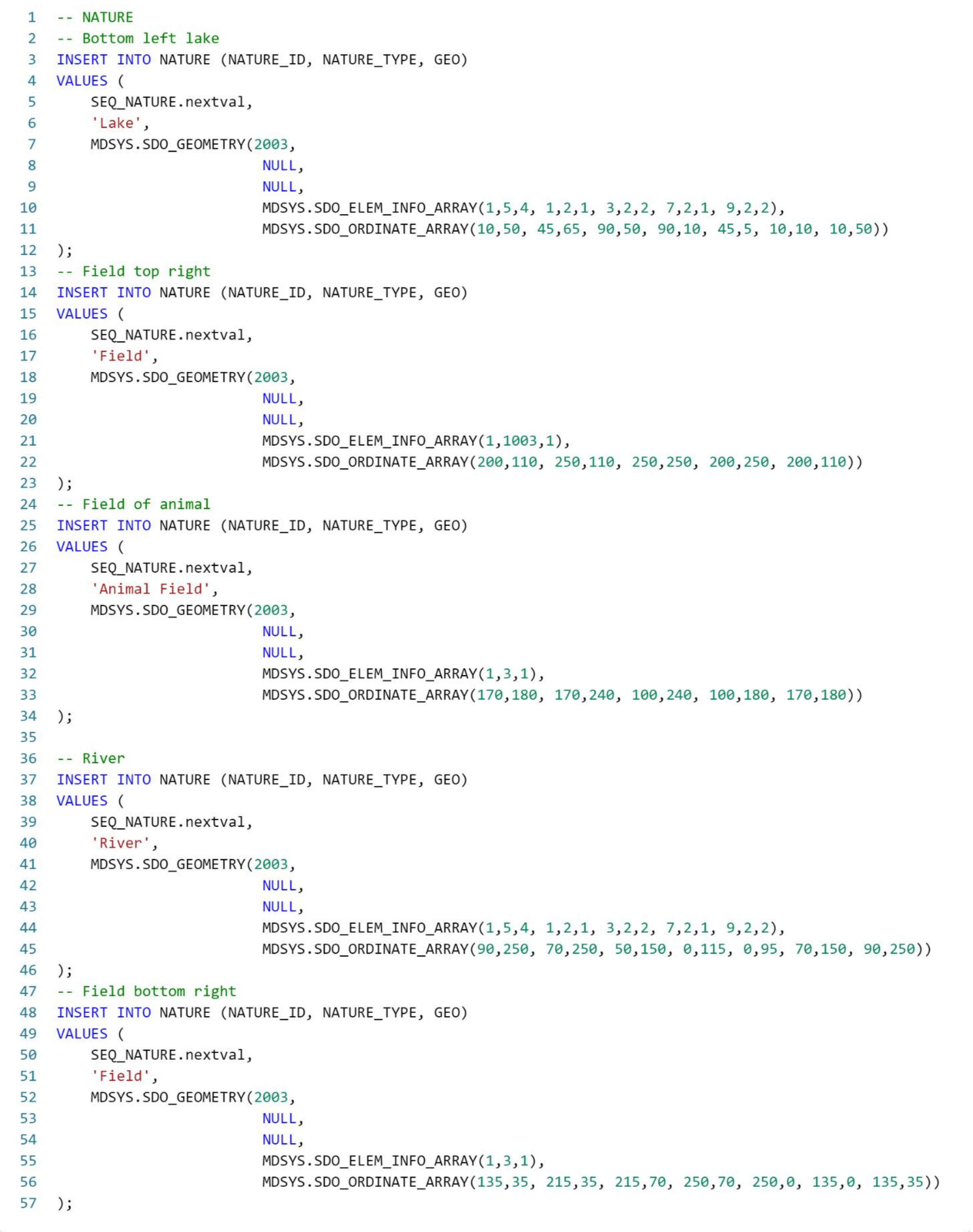


	ROAD_ID	ROAD_TYPE	GEO	ROAD_ID_1	ROAD_STATUS	LAST_CHECK_DATE	DESCRIPTION
1	1	Road	[MDSYS.SDO_GEOMETRY]	1	Open	01/01/23	Main road connecting key areas with smooth traffic flow.
2	2	Road	[MDSYS.SDO_GEOMETRY]	2	Closed	02/01/23	Curved road currently closed for maintenance, use alternative routes.
3	3	Road	[MDSYS.SDO_GEOMETRY]	3	Open	03/01/23	Vertical road facilitating north-south traffic in the city.
4	4	Path	[MDSYS.SDO_GEOMETRY]	4	Open	04/01/23	Path leading from the main road to a residential house.

Figure 5.2.3. Roads insertion validation

NATURE:

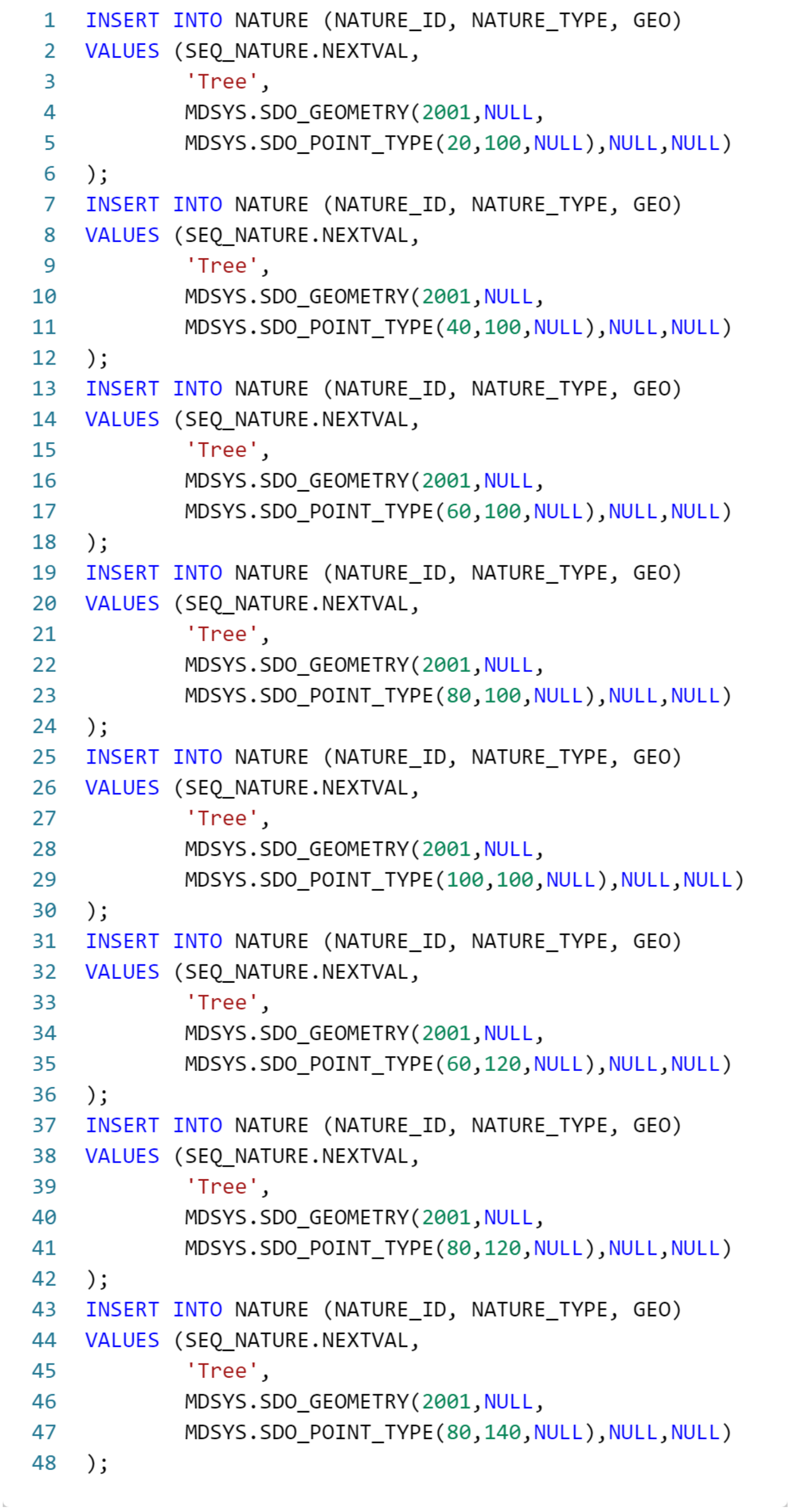
This layer contains one lake in shape of an oval, 2 fields one of which has a special shape, a river that is curved and some trees. I firstly created my trees aside to principal road. Then I realized that I still have some space to build a forest on the left.



```sql
-- NATURE
-- Bottom left lake
INSERT INTO NATURE (NATURE_ID, NATURE_TYPE, GEO)
VALUES (
    SEQ_NATURE.nextval,
    'Lake',
    MDSYS.SDO_GEOMETRY(2003,
                        NULL,
                        NULL,
                        MDSYS.SDO_ELEM_INFO_ARRAY(1,5,4, 1,2,1, 3,2,2, 7,2,1, 9,2,2),
                        MDSYS.SDO_ORDINATE_ARRAY(10,50, 45,65, 90,50, 90,10, 45,5, 10,10, 10,50))
);
-- Field top right
INSERT INTO NATURE (NATURE_ID, NATURE_TYPE, GEO)
VALUES (
    SEQ_NATURE.nextval,
    'Field',
    MDSYS.SDO_GEOMETRY(2003,
                        NULL,
                        NULL,
                        MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),
                        MDSYS.SDO_ORDINATE_ARRAY(200,110, 250,110, 250,250, 200,250, 200,110))
);
-- Field of animal
INSERT INTO NATURE (NATURE_ID, NATURE_TYPE, GEO)
VALUES (
    SEQ_NATURE.nextval,
    'Animal Field',
    MDSYS.SDO_GEOMETRY(2003,
                        NULL,
                        NULL,
                        MDSYS.SDO_ELEM_INFO_ARRAY(1,3,1),
                        MDSYS.SDO_ORDINATE_ARRAY(170,180, 170,240, 100,240, 100,180, 170,180))
);

-- River
INSERT INTO NATURE (NATURE_ID, NATURE_TYPE, GEO)
VALUES (
    SEQ_NATURE.nextval,
    'River',
    MDSYS.SDO_GEOMETRY(2003,
                        NULL,
                        NULL,
                        MDSYS.SDO_ELEM_INFO_ARRAY(1,5,4, 1,2,1, 3,2,2, 7,2,1, 9,2,2),
                        MDSYS.SDO_ORDINATE_ARRAY(90,250, 70,250, 50,150, 0,115, 0,95, 70,150, 90,250))
);
-- Field bottom right
INSERT INTO NATURE (NATURE_ID, NATURE_TYPE, GEO)
VALUES (
    SEQ_NATURE.nextval,
    'Field',
    MDSYS.SDO_GEOMETRY(2003,
                        NULL,
                        NULL,
                        MDSYS.SDO_ELEM_INFO_ARRAY(1,3,1),
                        MDSYS.SDO_ORDINATE_ARRAY(135,35, 215,35, 215,70, 250,70, 250,0, 135,0, 135,35))
);
```

Figure 5.3.1 Nature values insertion

To create the trees, I used two methods. The first was to hardcode the coordinates of all the trees along the principal road.



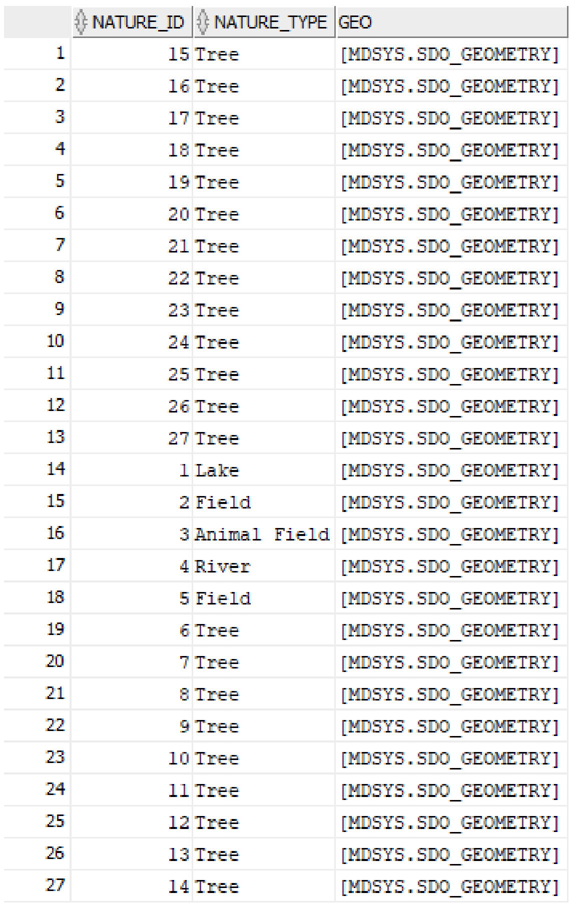
```
INSERT INTO NATURE (NATURE_ID, NATURE_TYPE, GEO)
VALUES (SEQ_NATURE.NEXTVAL,
        'Tree',
        MDSYS.SDO_GEOMETRY(2001,NULL,
        MDSYS.SDO_POINT_TYPE(20,100,NULL),NULL,NULL)
);
INSERT INTO NATURE (NATURE_ID, NATURE_TYPE, GEO)
VALUES (SEQ_NATURE.NEXTVAL,
        'Tree',
        MDSYS.SDO_GEOMETRY(2001,NULL,
        MDSYS.SDO_POINT_TYPE(40,100,NULL),NULL,NULL)
);
INSERT INTO NATURE (NATURE_ID, NATURE_TYPE, GEO)
VALUES (SEQ_NATURE.NEXTVAL,
        'Tree',
        MDSYS.SDO_GEOMETRY(2001,NULL,
        MDSYS.SDO_POINT_TYPE(60,100,NULL),NULL,NULL)
);
INSERT INTO NATURE (NATURE_ID, NATURE_TYPE, GEO)
VALUES (SEQ_NATURE.NEXTVAL,
        'Tree',
        MDSYS.SDO_GEOMETRY(2001,NULL,
        MDSYS.SDO_POINT_TYPE(80,100,NULL),NULL,NULL)
);
INSERT INTO NATURE (NATURE_ID, NATURE_TYPE, GEO)
VALUES (SEQ_NATURE.NEXTVAL,
        'Tree',
        MDSYS.SDO_GEOMETRY(2001,NULL,
        MDSYS.SDO_POINT_TYPE(100,100,NULL),NULL,NULL)
);
INSERT INTO NATURE (NATURE_ID, NATURE_TYPE, GEO)
VALUES (SEQ_NATURE.NEXTVAL,
        'Tree',
        MDSYS.SDO_GEOMETRY(2001,NULL,
        MDSYS.SDO_POINT_TYPE(60,120,NULL),NULL,NULL)
);
INSERT INTO NATURE (NATURE_ID, NATURE_TYPE, GEO)
VALUES (SEQ_NATURE.NEXTVAL,
        'Tree',
        MDSYS.SDO_GEOMETRY(2001,NULL,
        MDSYS.SDO_POINT_TYPE(80,120,NULL),NULL,NULL)
);
INSERT INTO NATURE (NATURE_ID, NATURE_TYPE, GEO)
VALUES (SEQ_NATURE.NEXTVAL,
        'Tree',
        MDSYS.SDO_GEOMETRY(2001,NULL,
        MDSYS.SDO_POINT_TYPE(80,140,NULL),NULL,NULL)
);
```

Figure 5.3.2. Trees values insertion (hardcode)

Then, to build my forest, I used a **procedure**, as all the trees are lined.

```
-- Forest on the left
CREATE OR REPLACE PROCEDURE ADD_TREES_ALONG_AXIS_X(
    p_start_y NUMBER,
    p_end_y NUMBER,
    p_spacing NUMBER
)
IS
    v_current_y NUMBER := p_start_y;
BEGIN
    -- Add trees along x=0
    WHILE v_current_y <= p_end_y
    LOOP
        INSERT INTO NATURE (NATURE_ID, NATURE_TYPE, GEO)
        VALUES (SEQ_NATURE.NEXTVAL,
                'Tree',
                MDSYS.SDO_GEOMETRY(2001, NULL,
                                   MDSYS.SDO_POINT_TYPE(0, v_current_y, NULL),
                                   NULL, NULL)
                );

        v_current_y := v_current_y + p_spacing;
    END LOOP;

    -- Re initialize y value
    v_current_y := p_start_y;

    -- add trees along x=20
    WHILE v_current_y <= p_end_y
    LOOP
        INSERT INTO NATURE (NATURE_ID, NATURE_TYPE, GEO)
        VALUES (SEQ_NATURE.NEXTVAL,
                'Tree',
                MDSYS.SDO_GEOMETRY(2001, NULL,
                                   MDSYS.SDO_POINT_TYPE(20, v_current_y, NULL),
                                   NULL, NULL)
                );

        v_current_y := v_current_y + p_spacing;
    END LOOP;
END ADD_TREES_ALONG_AXIS_X;
/
BEGIN
    ADD_TREES_ALONG_AXIS_X(130, 250, 20);
END;
/
```

Figure 5.3.3. Tree values insertion (procedure)

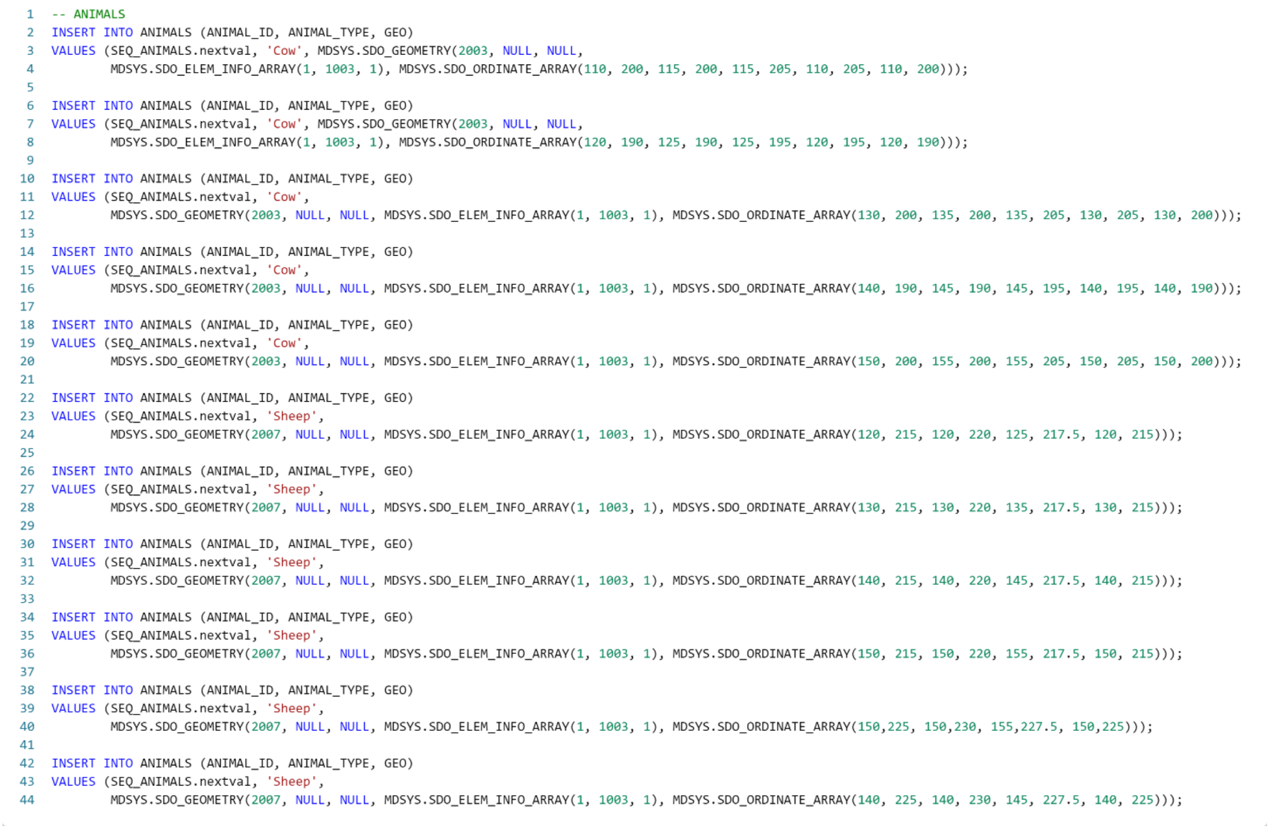


	NATURE_ID	NATURE_TYPE	GEO
1	15	Tree	[MDSYS.SDO_GEOMETRY]
2	16	Tree	[MDSYS.SDO_GEOMETRY]
3	17	Tree	[MDSYS.SDO_GEOMETRY]
4	18	Tree	[MDSYS.SDO_GEOMETRY]
5	19	Tree	[MDSYS.SDO_GEOMETRY]
6	20	Tree	[MDSYS.SDO_GEOMETRY]
7	21	Tree	[MDSYS.SDO_GEOMETRY]
8	22	Tree	[MDSYS.SDO_GEOMETRY]
9	23	Tree	[MDSYS.SDO_GEOMETRY]
10	24	Tree	[MDSYS.SDO_GEOMETRY]
11	25	Tree	[MDSYS.SDO_GEOMETRY]
12	26	Tree	[MDSYS.SDO_GEOMETRY]
13	27	Tree	[MDSYS.SDO_GEOMETRY]
14	1	Lake	[MDSYS.SDO_GEOMETRY]
15	2	Field	[MDSYS.SDO_GEOMETRY]
16	3	Animal Field	[MDSYS.SDO_GEOMETRY]
17	4	River	[MDSYS.SDO_GEOMETRY]
18	5	Field	[MDSYS.SDO_GEOMETRY]
19	6	Tree	[MDSYS.SDO_GEOMETRY]
20	7	Tree	[MDSYS.SDO_GEOMETRY]
21	8	Tree	[MDSYS.SDO_GEOMETRY]
22	9	Tree	[MDSYS.SDO_GEOMETRY]
23	10	Tree	[MDSYS.SDO_GEOMETRY]
24	11	Tree	[MDSYS.SDO_GEOMETRY]
25	12	Tree	[MDSYS.SDO_GEOMETRY]
26	13	Tree	[MDSYS.SDO_GEOMETRY]
27	14	Tree	[MDSYS.SDO_GEOMETRY]

Figure 5.3.4. Nature insertion validation

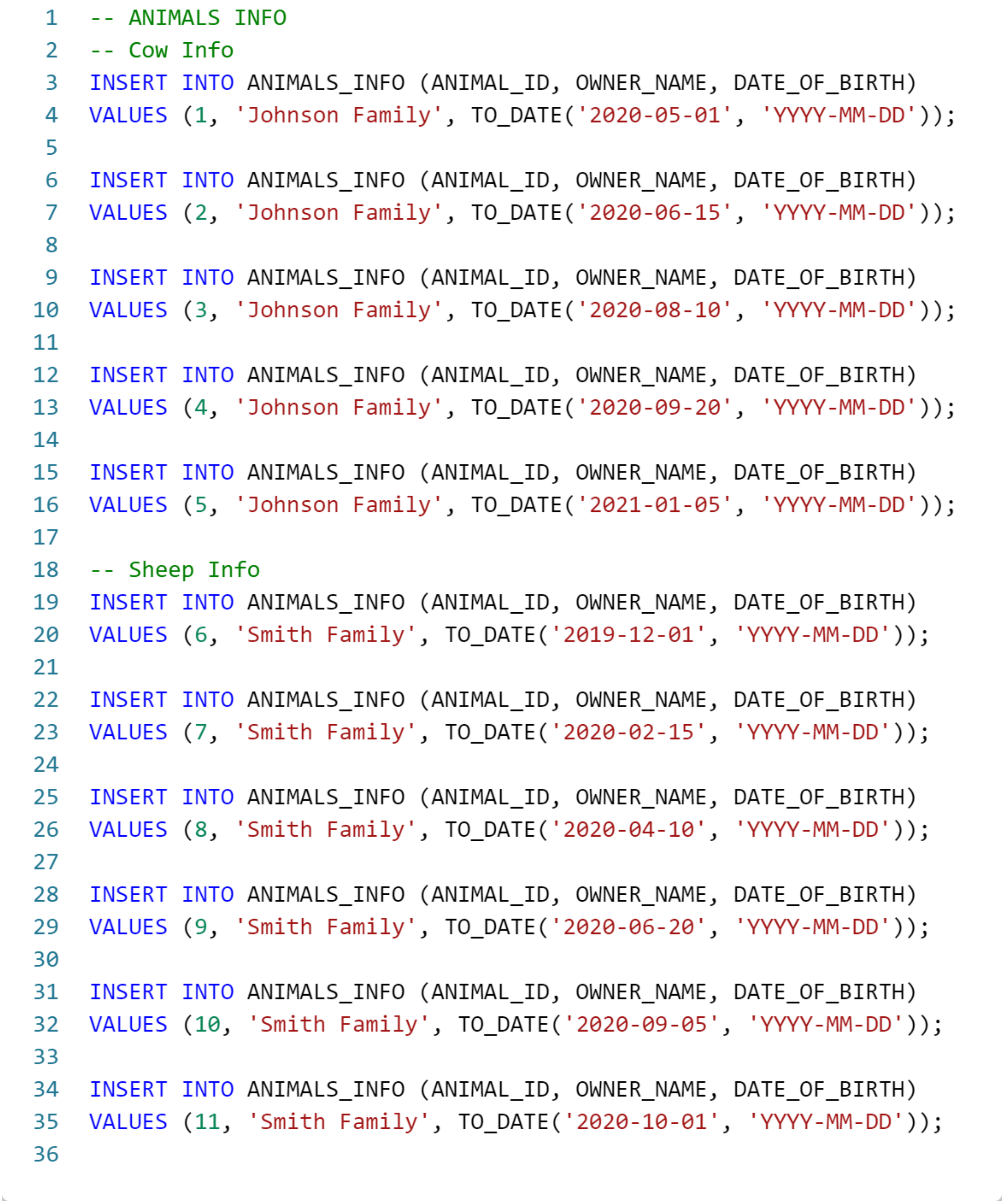
ANIMALS:

This layer contains some cows in a shape of a square and some sheeps in a shape of a triangle. They are both in the animal field previously implemented.



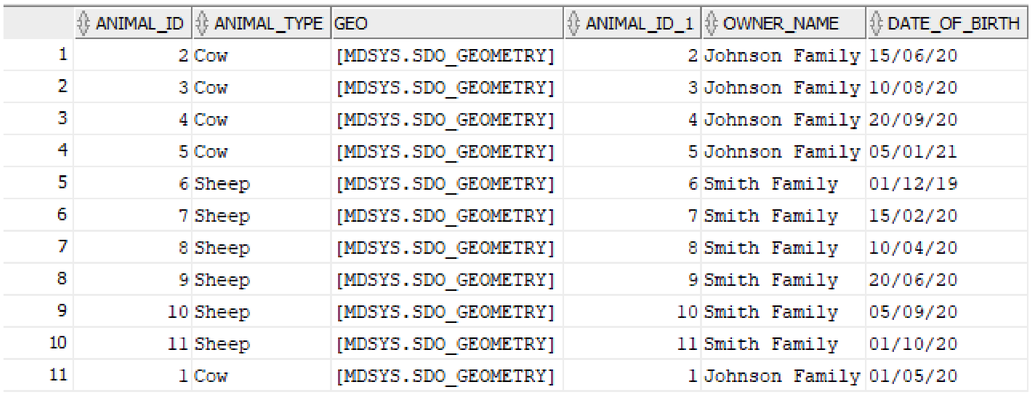
```sql
-- ANIMALS
INSERT INTO ANIMALS (ANIMAL_ID, ANIMAL_TYPE, GEO)
VALUES (SEQ_ANIMALS.nextval, 'Cow', MDSYS.SDO_GEOMETRY(2003, NULL, NULL,
        MDSYS.SDO_ELEM_INFO_ARRAY(1, 1003, 1), MDSYS.SDO_ORDINATE_ARRAY(110, 200, 115, 200, 115, 205, 110, 205, 110, 200)));

INSERT INTO ANIMALS (ANIMAL_ID, ANIMAL_TYPE, GEO)
VALUES (SEQ_ANIMALS.nextval, 'Cow', MDSYS.SDO_GEOMETRY(2003, NULL, NULL,
        MDSYS.SDO_ELEM_INFO_ARRAY(1, 1003, 1), MDSYS.SDO_ORDINATE_ARRAY(120, 190, 125, 190, 125, 195, 120, 195, 120, 190)));

INSERT INTO ANIMALS (ANIMAL_ID, ANIMAL_TYPE, GEO)
VALUES (SEQ_ANIMALS.nextval, 'Cow',
        MDSYS.SDO_GEOMETRY(2003, NULL, NULL, MDSYS.SDO_ELEM_INFO_ARRAY(1, 1003, 1), MDSYS.SDO_ORDINATE_ARRAY(130, 200, 135, 200, 135, 205, 130, 205, 130, 200)));

INSERT INTO ANIMALS (ANIMAL_ID, ANIMAL_TYPE, GEO)
VALUES (SEQ_ANIMALS.nextval, 'Cow',
        MDSYS.SDO_GEOMETRY(2003, NULL, NULL, MDSYS.SDO_ELEM_INFO_ARRAY(1, 1003, 1), MDSYS.SDO_ORDINATE_ARRAY(140, 190, 145, 190, 145, 195, 140, 195, 140, 190)));

INSERT INTO ANIMALS (ANIMAL_ID, ANIMAL_TYPE, GEO)
VALUES (SEQ_ANIMALS.nextval, 'Cow',
        MDSYS.SDO_GEOMETRY(2003, NULL, NULL, MDSYS.SDO_ELEM_INFO_ARRAY(1, 1003, 1), MDSYS.SDO_ORDINATE_ARRAY(150, 200, 155, 200, 155, 205, 150, 205, 150, 200)));

INSERT INTO ANIMALS (ANIMAL_ID, ANIMAL_TYPE, GEO)
VALUES (SEQ_ANIMALS.nextval, 'Sheep',
        MDSYS.SDO_GEOMETRY(2007, NULL, NULL, MDSYS.SDO_ELEM_INFO_ARRAY(1, 1003, 1), MDSYS.SDO_ORDINATE_ARRAY(120, 215, 120, 220, 125, 217.5, 120, 215)));

INSERT INTO ANIMALS (ANIMAL_ID, ANIMAL_TYPE, GEO)
VALUES (SEQ_ANIMALS.nextval, 'Sheep',
        MDSYS.SDO_GEOMETRY(2007, NULL, NULL, MDSYS.SDO_ELEM_INFO_ARRAY(1, 1003, 1), MDSYS.SDO_ORDINATE_ARRAY(130, 215, 130, 220, 135, 217.5, 130, 215)));

INSERT INTO ANIMALS (ANIMAL_ID, ANIMAL_TYPE, GEO)
VALUES (SEQ_ANIMALS.nextval, 'Sheep',
        MDSYS.SDO_GEOMETRY(2007, NULL, NULL, MDSYS.SDO_ELEM_INFO_ARRAY(1, 1003, 1), MDSYS.SDO_ORDINATE_ARRAY(140, 215, 140, 220, 145, 217.5, 140, 215)));

INSERT INTO ANIMALS (ANIMAL_ID, ANIMAL_TYPE, GEO)
VALUES (SEQ_ANIMALS.nextval, 'Sheep',
        MDSYS.SDO_GEOMETRY(2007, NULL, NULL, MDSYS.SDO_ELEM_INFO_ARRAY(1, 1003, 1), MDSYS.SDO_ORDINATE_ARRAY(150, 215, 150, 220, 155, 217.5, 150, 215)));

INSERT INTO ANIMALS (ANIMAL_ID, ANIMAL_TYPE, GEO)
VALUES (SEQ_ANIMALS.nextval, 'Sheep',
        MDSYS.SDO_GEOMETRY(2007, NULL, NULL, MDSYS.SDO_ELEM_INFO_ARRAY(1, 1003, 1), MDSYS.SDO_ORDINATE_ARRAY(150,225, 150,230, 155,227.5, 150,225)));

INSERT INTO ANIMALS (ANIMAL_ID, ANIMAL_TYPE, GEO)
VALUES (SEQ_ANIMALS.nextval, 'Sheep',
        MDSYS.SDO_GEOMETRY(2007, NULL, NULL, MDSYS.SDO_ELEM_INFO_ARRAY(1, 1003, 1), MDSYS.SDO_ORDINATE_ARRAY(140, 225, 140, 230, 145, 227.5, 140, 225)));
```

Figure 5.4.1 Animal values insertion



```sql
-- ANIMALS INFO
-- Cow Info
INSERT INTO ANIMALS_INFO (ANIMAL_ID, OWNER_NAME, DATE_OF_BIRTH)
VALUES (1, 'Johnson Family', TO_DATE('2020-05-01', 'YYYY-MM-DD'));

INSERT INTO ANIMALS_INFO (ANIMAL_ID, OWNER_NAME, DATE_OF_BIRTH)
VALUES (2, 'Johnson Family', TO_DATE('2020-06-15', 'YYYY-MM-DD'));

INSERT INTO ANIMALS_INFO (ANIMAL_ID, OWNER_NAME, DATE_OF_BIRTH)
VALUES (3, 'Johnson Family', TO_DATE('2020-08-10', 'YYYY-MM-DD'));

INSERT INTO ANIMALS_INFO (ANIMAL_ID, OWNER_NAME, DATE_OF_BIRTH)
VALUES (4, 'Johnson Family', TO_DATE('2020-09-20', 'YYYY-MM-DD'));

INSERT INTO ANIMALS_INFO (ANIMAL_ID, OWNER_NAME, DATE_OF_BIRTH)
VALUES (5, 'Johnson Family', TO_DATE('2021-01-05', 'YYYY-MM-DD'));

-- Sheep Info
INSERT INTO ANIMALS_INFO (ANIMAL_ID, OWNER_NAME, DATE_OF_BIRTH)
VALUES (6, 'Smith Family', TO_DATE('2019-12-01', 'YYYY-MM-DD'));

INSERT INTO ANIMALS_INFO (ANIMAL_ID, OWNER_NAME, DATE_OF_BIRTH)
VALUES (7, 'Smith Family', TO_DATE('2020-02-15', 'YYYY-MM-DD'));

INSERT INTO ANIMALS_INFO (ANIMAL_ID, OWNER_NAME, DATE_OF_BIRTH)
VALUES (8, 'Smith Family', TO_DATE('2020-04-10', 'YYYY-MM-DD'));

INSERT INTO ANIMALS_INFO (ANIMAL_ID, OWNER_NAME, DATE_OF_BIRTH)
VALUES (9, 'Smith Family', TO_DATE('2020-06-20', 'YYYY-MM-DD'));

INSERT INTO ANIMALS_INFO (ANIMAL_ID, OWNER_NAME, DATE_OF_BIRTH)
VALUES (10, 'Smith Family', TO_DATE('2020-09-05', 'YYYY-MM-DD'));

INSERT INTO ANIMALS_INFO (ANIMAL_ID, OWNER_NAME, DATE_OF_BIRTH)
VALUES (11, 'Smith Family', TO_DATE('2020-10-01', 'YYYY-MM-DD'));

```

Figure 5.4.2. Animals_info values insertion

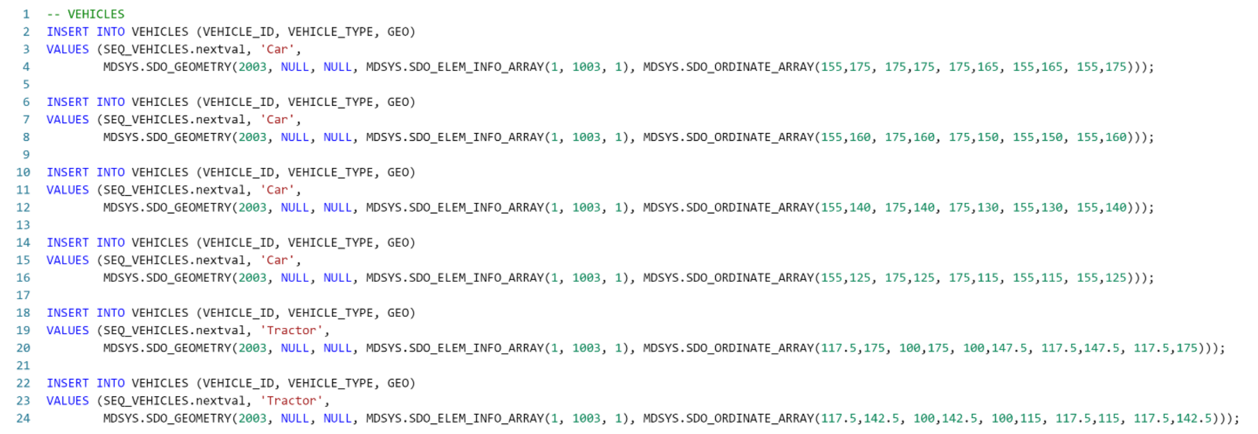


	ANIMAL_ID	ANIMAL_TYPE	GEO	ANIMAL_ID_1	OWNER_NAME	DATE_OF_BIRTH
1	2	Cow	[MDSYS.SDO_GEOMETRY]	2	Johnson Family	15/06/20
2	3	Cow	[MDSYS.SDO_GEOMETRY]	3	Johnson Family	10/08/20
3	4	Cow	[MDSYS.SDO_GEOMETRY]	4	Johnson Family	20/09/20
4	5	Cow	[MDSYS.SDO_GEOMETRY]	5	Johnson Family	05/01/21
5	6	Sheep	[MDSYS.SDO_GEOMETRY]	6	Smith Family	01/12/19
6	7	Sheep	[MDSYS.SDO_GEOMETRY]	7	Smith Family	15/02/20
7	8	Sheep	[MDSYS.SDO_GEOMETRY]	8	Smith Family	10/04/20
8	9	Sheep	[MDSYS.SDO_GEOMETRY]	9	Smith Family	20/06/20
9	10	Sheep	[MDSYS.SDO_GEOMETRY]	10	Smith Family	05/09/20
10	11	Sheep	[MDSYS.SDO_GEOMETRY]	11	Smith Family	01/10/20
11	1	Cow	[MDSYS.SDO_GEOMETRY]	1	Johnson Family	01/05/20

Figure 5.4.3 Animals insertion validation

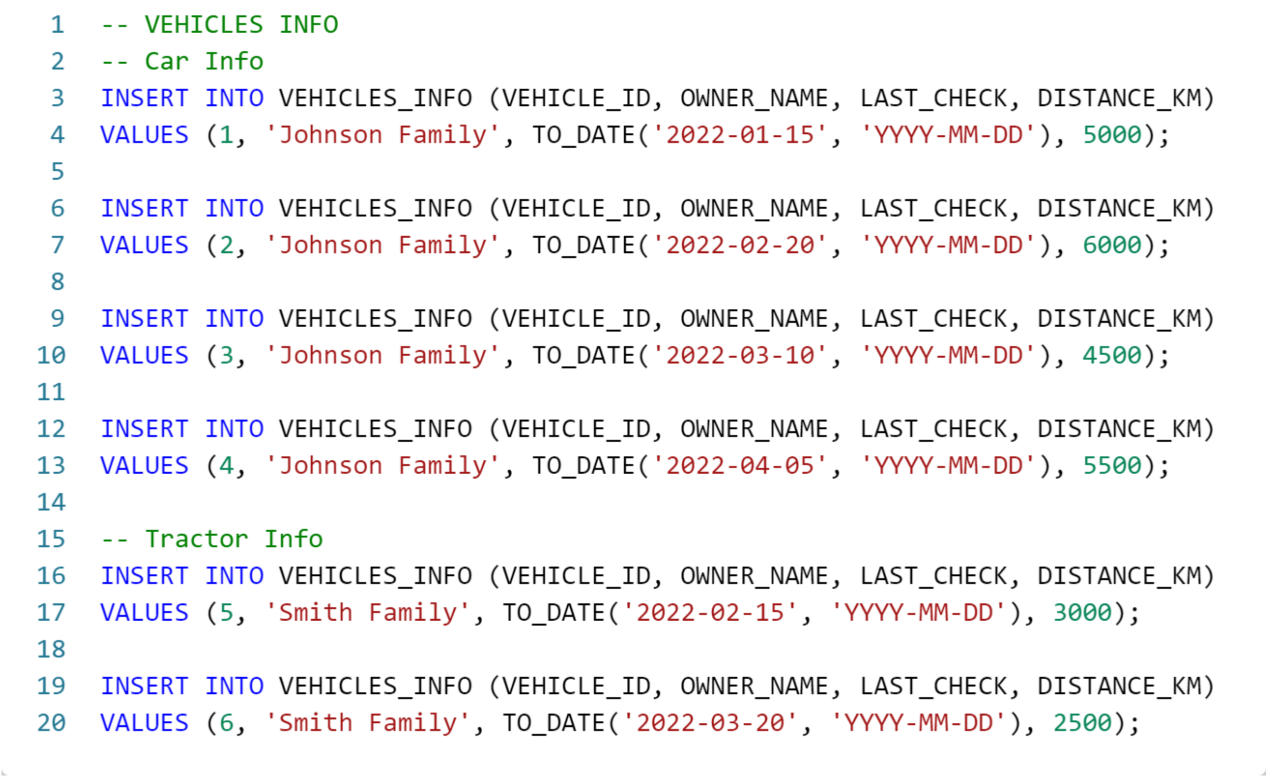
VEHICLES:

This layer contains cars and tractor in a shape of a rectangle that are parked close to the house.



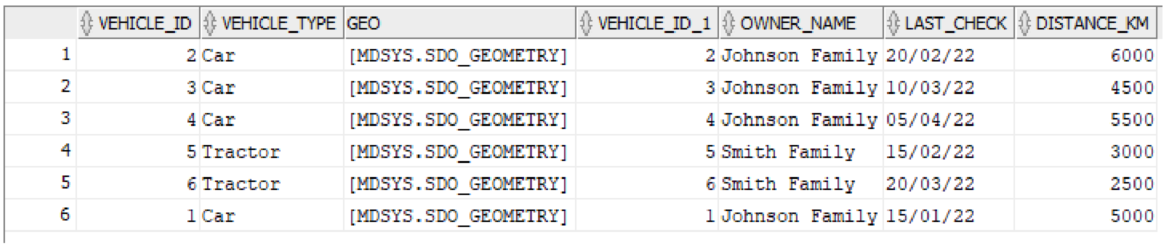
```sql
-- VEHICLES
INSERT INTO VEHICLES (VEHICLE_ID, VEHICLE_TYPE, GEO)
VALUES (SEQ_VEHICLES.nextval, 'Car',
        MDSYS.SDO_GEOMETRY(2003, NULL, NULL, MDSYS.SDO_ELEM_INFO_ARRAY(1, 1003, 1), MDSYS.SDO_ORDINATE_ARRAY(155,175, 175,175, 175,165, 155,165, 155,175)));

INSERT INTO VEHICLES (VEHICLE_ID, VEHICLE_TYPE, GEO)
VALUES (SEQ_VEHICLES.nextval, 'Car',
        MDSYS.SDO_GEOMETRY(2003, NULL, NULL, MDSYS.SDO_ELEM_INFO_ARRAY(1, 1003, 1), MDSYS.SDO_ORDINATE_ARRAY(155,160, 175,160, 175,150, 155,150, 155,160)));

INSERT INTO VEHICLES (VEHICLE_ID, VEHICLE_TYPE, GEO)
VALUES (SEQ_VEHICLES.nextval, 'Car',
        MDSYS.SDO_GEOMETRY(2003, NULL, NULL, MDSYS.SDO_ELEM_INFO_ARRAY(1, 1003, 1), MDSYS.SDO_ORDINATE_ARRAY(155,140, 175,140, 175,130, 155,130, 155,140)));

INSERT INTO VEHICLES (VEHICLE_ID, VEHICLE_TYPE, GEO)
VALUES (SEQ_VEHICLES.nextval, 'Car',
        MDSYS.SDO_GEOMETRY(2003, NULL, NULL, MDSYS.SDO_ELEM_INFO_ARRAY(1, 1003, 1), MDSYS.SDO_ORDINATE_ARRAY(155,125, 175,125, 175,115, 155,115, 155,125)));

INSERT INTO VEHICLES (VEHICLE_ID, VEHICLE_TYPE, GEO)
VALUES (SEQ_VEHICLES.nextval, 'Tractor',
        MDSYS.SDO_GEOMETRY(2003, NULL, NULL, MDSYS.SDO_ELEM_INFO_ARRAY(1, 1003, 1), MDSYS.SDO_ORDINATE_ARRAY(117.5,175, 100,175, 100,147.5, 117.5,147.5, 117.5,175)));

INSERT INTO VEHICLES (VEHICLE_ID, VEHICLE_TYPE, GEO)
VALUES (SEQ_VEHICLES.nextval, 'Tractor',
        MDSYS.SDO_GEOMETRY(2003, NULL, NULL, MDSYS.SDO_ELEM_INFO_ARRAY(1, 1003, 1), MDSYS.SDO_ORDINATE_ARRAY(117.5,142.5, 100,142.5, 100,115, 117.5,115, 117.5,142.5)));
```

Figure 5.5.1 Vehicle values insertion



```sql
-- VEHICLES INFO
-- Car Info
INSERT INTO VEHICLES_INFO (VEHICLE_ID, OWNER_NAME, LAST_CHECK, DISTANCE_KM)
VALUES (1, 'Johnson Family', TO_DATE('2022-01-15', 'YYYY-MM-DD'), 5000);

INSERT INTO VEHICLES_INFO (VEHICLE_ID, OWNER_NAME, LAST_CHECK, DISTANCE_KM)
VALUES (2, 'Johnson Family', TO_DATE('2022-02-20', 'YYYY-MM-DD'), 6000);

INSERT INTO VEHICLES_INFO (VEHICLE_ID, OWNER_NAME, LAST_CHECK, DISTANCE_KM)
VALUES (3, 'Johnson Family', TO_DATE('2022-03-10', 'YYYY-MM-DD'), 4500);

INSERT INTO VEHICLES_INFO (VEHICLE_ID, OWNER_NAME, LAST_CHECK, DISTANCE_KM)
VALUES (4, 'Johnson Family', TO_DATE('2022-04-05', 'YYYY-MM-DD'), 5500);

-- Tractor Info
INSERT INTO VEHICLES_INFO (VEHICLE_ID, OWNER_NAME, LAST_CHECK, DISTANCE_KM)
VALUES (5, 'Smith Family', TO_DATE('2022-02-15', 'YYYY-MM-DD'), 3000);

INSERT INTO VEHICLES_INFO (VEHICLE_ID, OWNER_NAME, LAST_CHECK, DISTANCE_KM)
VALUES (6, 'Smith Family', TO_DATE('2022-03-20', 'YYYY-MM-DD'), 2500);
```

Figure 5.5.2. Vehicles_info values insertion

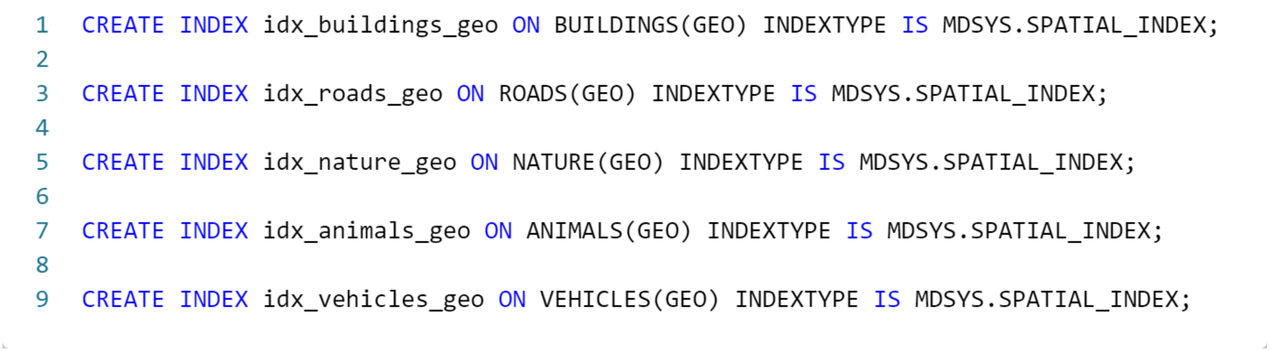


	VEHICLE_ID	VEHICLE_TYPE	GEO	VEHICLE_ID_1	OWNER_NAME	LAST_CHECK	DISTANCE_KM
1	2	Car	[MDSYS.SDO_GEOMETRY]	2	Johnson Family	20/02/22	6000
2	3	Car	[MDSYS.SDO_GEOMETRY]	3	Johnson Family	10/03/22	4500
3	4	Car	[MDSYS.SDO_GEOMETRY]	4	Johnson Family	05/04/22	5500
4	5	Tractor	[MDSYS.SDO_GEOMETRY]	5	Smith Family	15/02/22	3000
5	6	Tractor	[MDSYS.SDO_GEOMETRY]	6	Smith Family	20/03/22	2500
6	1	Car	[MDSYS.SDO_GEOMETRY]	1	Johnson Family	15/01/22	5000

Figure 5.5.3 Vehicles insertion validation

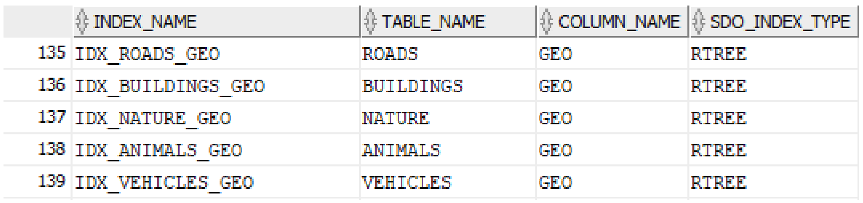
6. Spatial indexes

Spatial indexes are used in databases to speed up spatial queries, which involve geographic or geospatial data. These data may include information about physical locations, geometric shapes (points, lines, polygons), distances, and other space-related features.



```sql
CREATE INDEX idx_buildings_geo ON BUILDINGS(GEO) INDEXTYPE IS MDSYS.SPATIAL_INDEX;

CREATE INDEX idx_roads_geo ON ROADS(GEO) INDEXTYPE IS MDSYS.SPATIAL_INDEX;

CREATE INDEX idx_nature_geo ON NATURE(GEO) INDEXTYPE IS MDSYS.SPATIAL_INDEX;

CREATE INDEX idx_animals_geo ON ANIMALS(GEO) INDEXTYPE IS MDSYS.SPATIAL_INDEX;

CREATE INDEX idx_vehicles_geo ON VEHICLES(GEO) INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

Figure 6.1. Spatial indexes creation



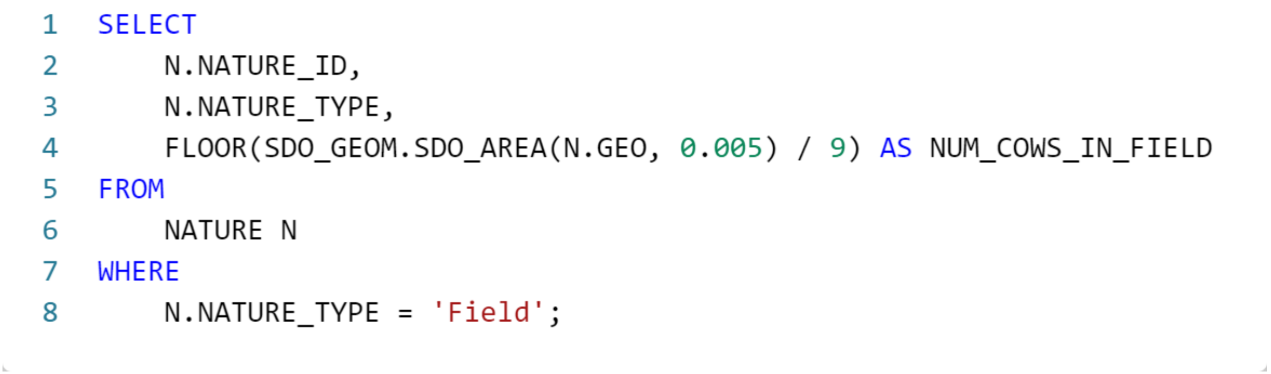
	INDEX_NAME	TABLE_NAME	COLUMN_NAME	SDO_INDEX_TYPE
135	IDX_ROADS_GEO	ROADS	GEO	RTREE
136	IDX_BUILDINGS_GEO	BUILDINGS	GEO	RTREE
137	IDX_NATURE_GEO	NATURE	GEO	RTREE
138	IDX_ANIMALS_GEO	ANIMALS	GEO	RTREE
139	IDX_VEHICLES_GEO	VEHICLES	GEO	RTREE

Figure 6.2. Spatial indexes insertion validation

7. Spatial queries

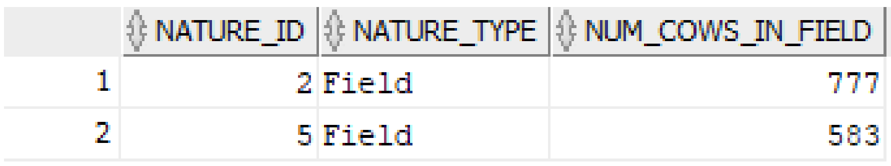
a. The number of cows per field

Since a cow is reprensented by a 3 x 3 square, we have the following query.



```sql
SELECT
    N.NATURE_ID,
    N.NATURE_TYPE,
    FLOOR(SDO_GEOM.SDO_AREA(N.GEO, 0.005) / 9) AS NUM_COWS_IN_FIELD
FROM
    NATURE N
WHERE
    N.NATURE_TYPE = 'Field';
```

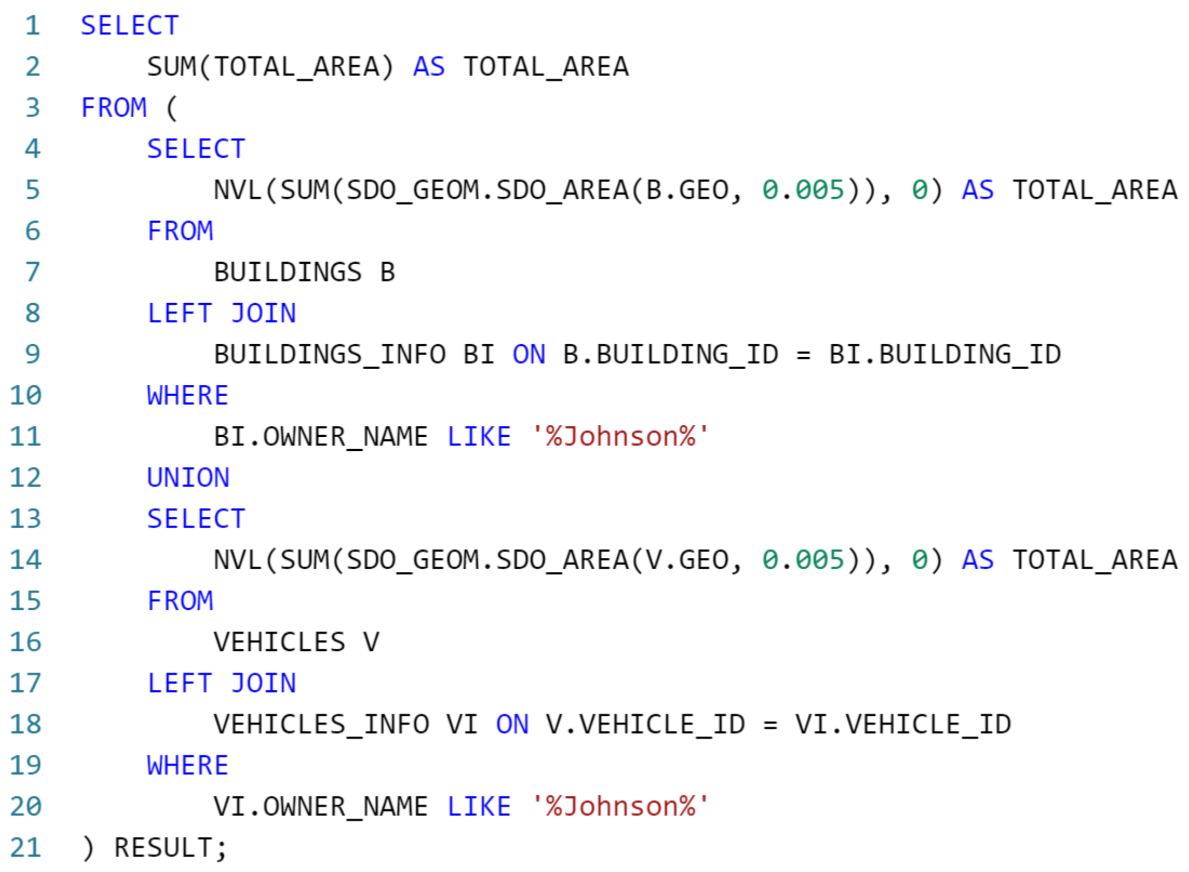
Figure 7.1.1. Query to get the number of cows per field



	NATURE_ID	NATURE_TYPE	NUM_COWS_IN_FIELD
1	2	Field	777
2	5	Field	583

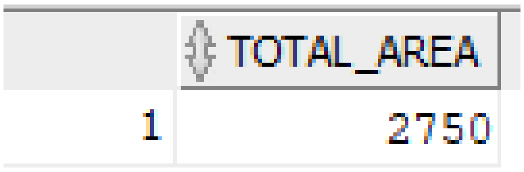
Figure 7.1.2 Result of query a.

b. Total aera of Johnson family (building and vehicles)



```
SELECT
    SUM(TOTAL_AREA) AS TOTAL_AREA
FROM (
    SELECT
        NVL(SUM(SDO_GEOM.SDO_AREA(B.GEO, 0.005)), 0) AS TOTAL_AREA
    FROM
        BUILDINGS B
    LEFT JOIN
        BUILDINGS_INFO BI ON B.BUILDING_ID = BI.BUILDING_ID
    WHERE
        BI.OWNER_NAME LIKE '%Johnson%'
    UNION
    SELECT
        NVL(SUM(SDO_GEOM.SDO_AREA(V.GEO, 0.005)), 0) AS TOTAL_AREA
    FROM
        VEHICLES V
    LEFT JOIN
        VEHICLES_INFO VI ON V.VEHICLE_ID = VI.VEHICLE_ID
    WHERE
        VI.OWNER_NAME LIKE '%Johnson%'
) RESULT;
```

Figure 7.2.1. Query to get the total aera of the Johnson family



	TOTAL_AREA
1	2750

Figure 7.2.2. Result of query b.

c. Zombie invasion in the city

This query allows us to know which object will be invaded buy the zombie that are present in the the red square.

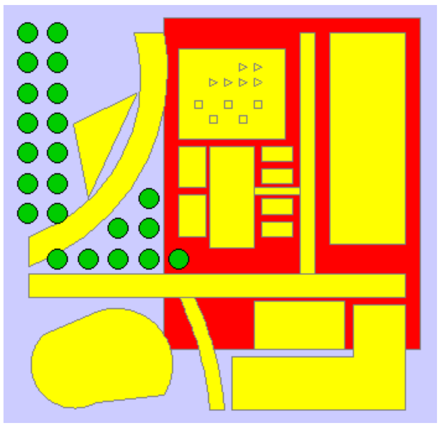
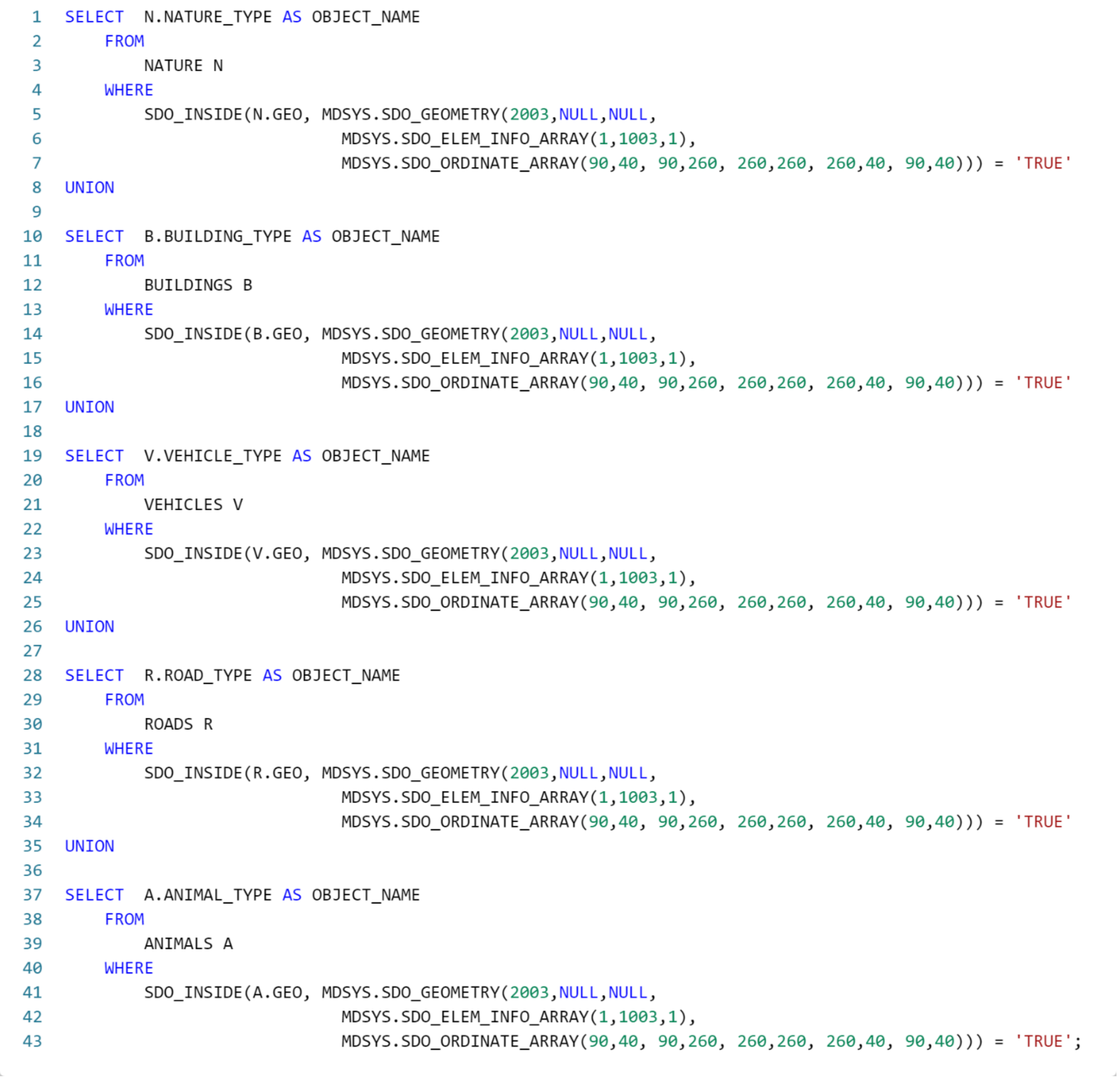


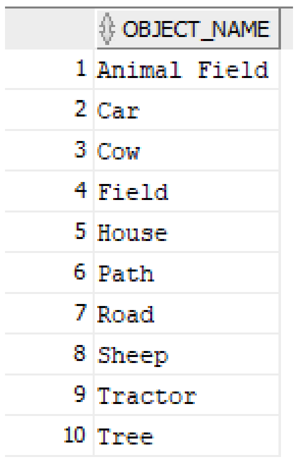
Figure 7.3.1. Visualization of the zombie invasion

This query looks for all the object that are totally inside in the red square. It is an intersection problem.



```sql
SELECT  N.NATURE_TYPE AS OBJECT_NAME
    FROM
        NATURE N
    WHERE
        SDO_INSIDE(N.GEO, MDSYS.SDO_GEOMETRY(2003,NULL,NULL,
                                MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),
                                MDSYS.SDO_ORDINATE_ARRAY(90,40, 90,260, 260,260, 260,40, 90,40))) = 'TRUE'
UNION

SELECT  B.BUILDING_TYPE AS OBJECT_NAME
    FROM
        BUILDINGS B
    WHERE
        SDO_INSIDE(B.GEO, MDSYS.SDO_GEOMETRY(2003,NULL,NULL,
                                MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),
                                MDSYS.SDO_ORDINATE_ARRAY(90,40, 90,260, 260,260, 260,40, 90,40))) = 'TRUE'
UNION

SELECT  V.VEHICLE_TYPE AS OBJECT_NAME
    FROM
        VEHICLES V
    WHERE
        SDO_INSIDE(V.GEO, MDSYS.SDO_GEOMETRY(2003,NULL,NULL,
                                MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),
                                MDSYS.SDO_ORDINATE_ARRAY(90,40, 90,260, 260,260, 260,40, 90,40))) = 'TRUE'
UNION

SELECT  R.ROAD_TYPE AS OBJECT_NAME
    FROM
        ROADS R
    WHERE
        SDO_INSIDE(R.GEO, MDSYS.SDO_GEOMETRY(2003,NULL,NULL,
                                MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),
                                MDSYS.SDO_ORDINATE_ARRAY(90,40, 90,260, 260,260, 260,40, 90,40))) = 'TRUE'
UNION

SELECT  A.ANIMAL_TYPE AS OBJECT_NAME
    FROM
        ANIMALS A
    WHERE
        SDO_INSIDE(A.GEO, MDSYS.SDO_GEOMETRY(2003,NULL,NULL,
                                MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),
                                MDSYS.SDO_ORDINATE_ARRAY(90,40, 90,260, 260,260, 260,40, 90,40))) = 'TRUE';
```

Figure 7.3.2. Query for zombie invasion

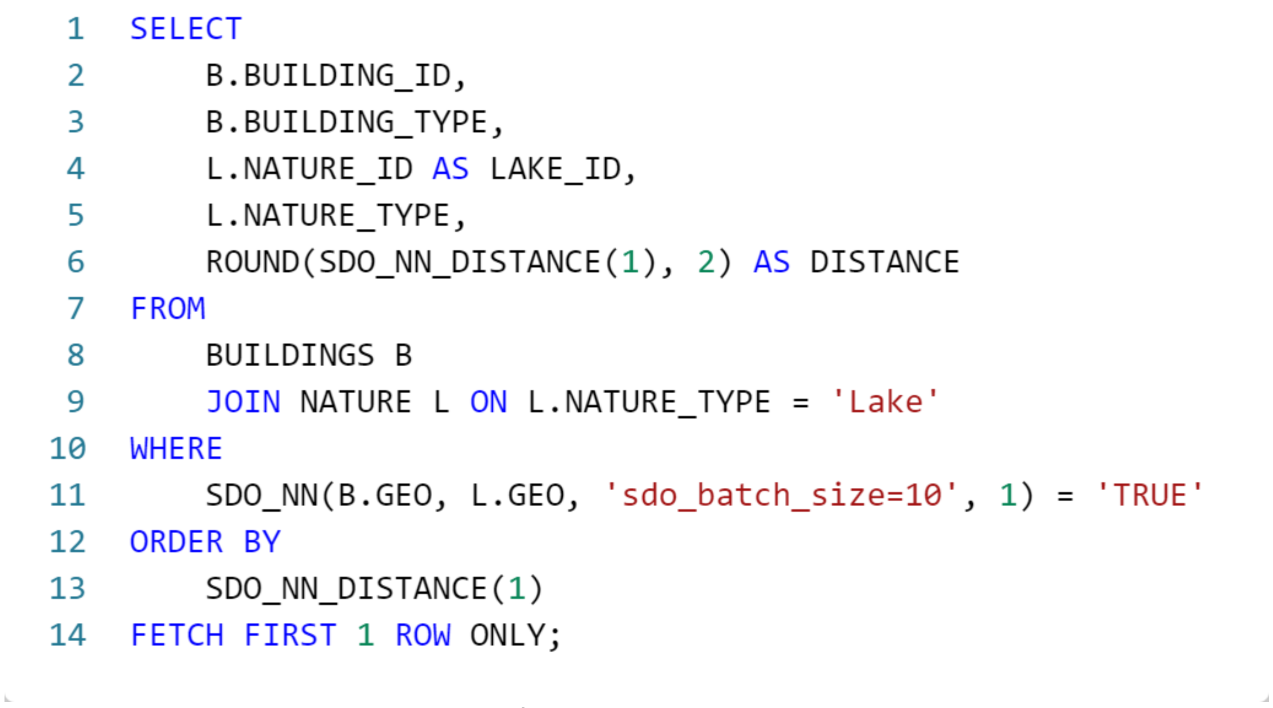


	OBJECT_NAME
1	Animal Field
2	Car
3	Cow
4	Field
5	House
6	Path
7	Road
8	Sheep
9	Tractor
10	Tree

Figure 7.3.3. Result for the query c.

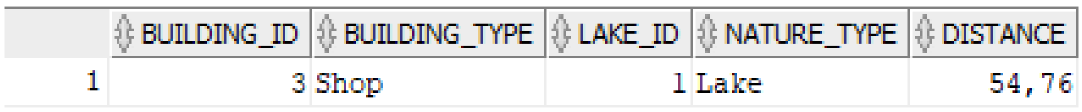
d. Which building is the nearest to the lake ?

The idea of this query is the calculate the distance of each building from the lake, order these result and keep the one with the smallest distance.



```sql
SELECT
    B.BUILDING_ID,
    B.BUILDING_TYPE,
    L.NATURE_ID AS LAKE_ID,
    L.NATURE_TYPE,
    ROUND(SDO_NN_DISTANCE(1), 2) AS DISTANCE
FROM
    BUILDINGS B
    JOIN NATURE L ON L.NATURE_TYPE = 'Lake'
WHERE
    SDO_NN(B.GEO, L.GEO, 'sdo_batch_size=10', 1) = 'TRUE'
ORDER BY
    SDO_NN_DISTANCE(1)
FETCH FIRST 1 ROW ONLY;
```

Figure 7.4.1. Query to find the building that is the nearest to the lake

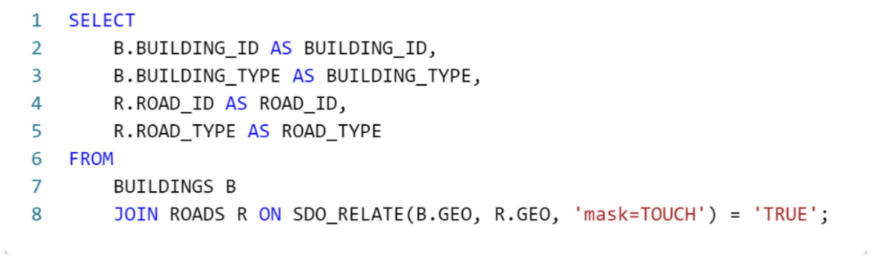


	BUILDING_ID	BUILDING_TYPE	LAKE_ID	NATURE_TYPE	DISTANCE
1	3	Shop	1	Lake	54,76

Figure 7.4.2. Results of the query d.

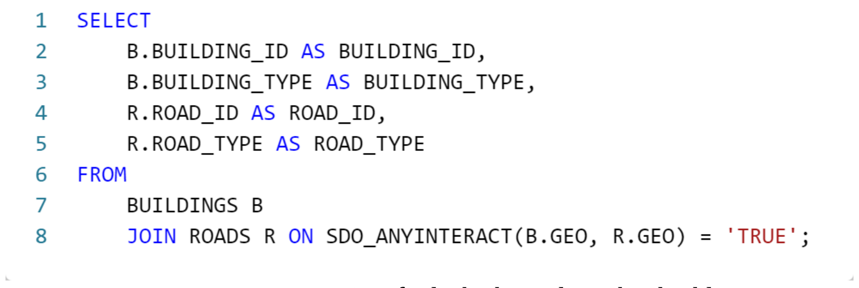
e. Find the roads that lead to a building

The idea is to find which road is in contact with a building. I found two ways to write the query. The first one use SDO_RELATE and the second one use SDO_ANYINTERACT.



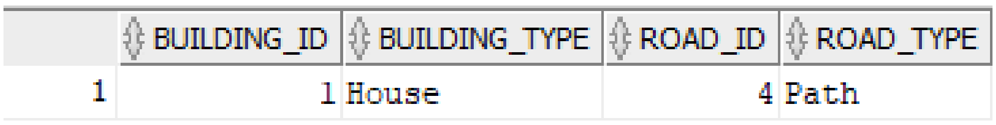
```sql
SELECT
    B.BUILDING_ID AS BUILDING_ID,
    B.BUILDING_TYPE AS BUILDING_TYPE,
    R.ROAD_ID AS ROAD_ID,
    R.ROAD_TYPE AS ROAD_TYPE
FROM
    BUILDINGS B
    JOIN ROADS R ON SDO_RELATE(B.GEO, R.GEO, 'mask=TOUCH') = 'TRUE';
```

Figure 7.5.1. Query 1 to find which road touch a building



```sql
SELECT
    B.BUILDING_ID AS BUILDING_ID,
    B.BUILDING_TYPE AS BUILDING_TYPE,
    R.ROAD_ID AS ROAD_ID,
    R.ROAD_TYPE AS ROAD_TYPE
FROM
    BUILDINGS B
    JOIN ROADS R ON SDO_ANYINTERACT(B.GEO, R.GEO) = 'TRUE';
```

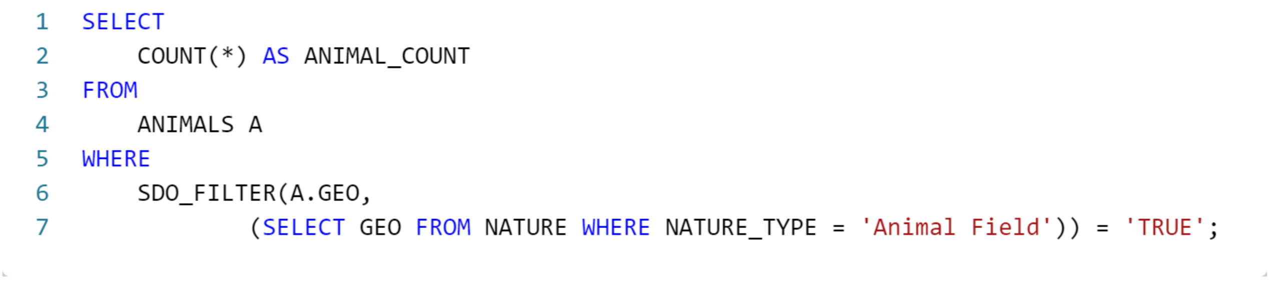
Figure 7.5.2. Query 2 to find which road touch a building



	BUILDING_ID	BUILDING_TYPE	ROAD_ID	ROAD_TYPE
1	1	House	4	Path

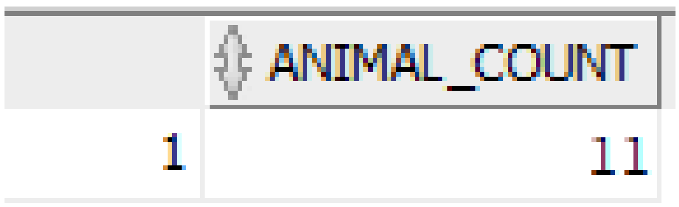
Figure 7.5.3. Result for query e. 1 and 2

f. Count how many animals are in the field



```sql
SELECT
    COUNT(*) AS ANIMAL_COUNT
FROM
    ANIMALS A
WHERE
    SDO_FILTER(A.GEO,
            (SELECT GEO FROM NATURE WHERE NATURE_TYPE = 'Animal Field')) = 'TRUE';
```

Figure 7.6.1. Query to count how many animals are in the field



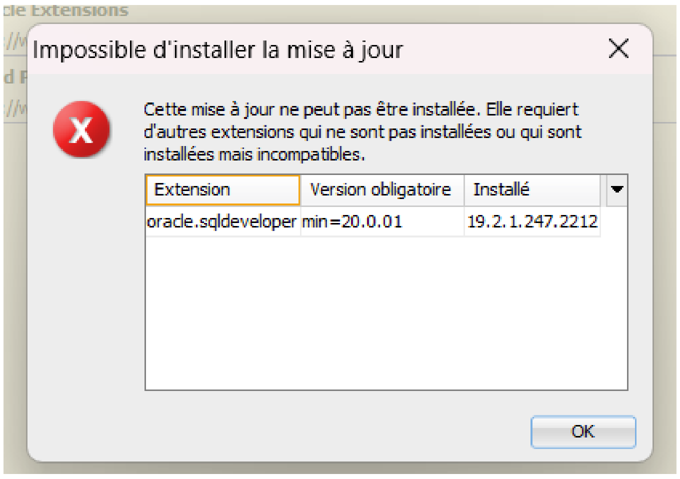
	ANIMAL_COUNT
1	11

Figure 7.6.2. Result of the query f.

8. Visualization

In this paragraph, I explain what tool I used for visualization and why.

As I am working with SQL Developer 19.2, I was not able to install GeoRaptor as it was recommended. On the site of GeoRaptor it was written that this extension is available for SQL Developer 18+ but I actually encountered this issue :



Impossible d'installer la mise à jour

Cette mise à jour ne peut pas être installée. Elle requiert d'autres extensions qui ne sont pas installées ou qui sont installées mais incompatibles.

Extension	Version obligatoire	Installé
oracle.sqldeveloper	min=20.0.01	19.2.1.247.2212

OK

Figure 8.1. Error when trying to install GeoRaptor

It is written that the minimal mandatory version is 20.

Then, I made the choice to visualize all my work with the the tool "map view" available from the installation of SQL Developer. This was not the most optimized tool that I could use but it still work.

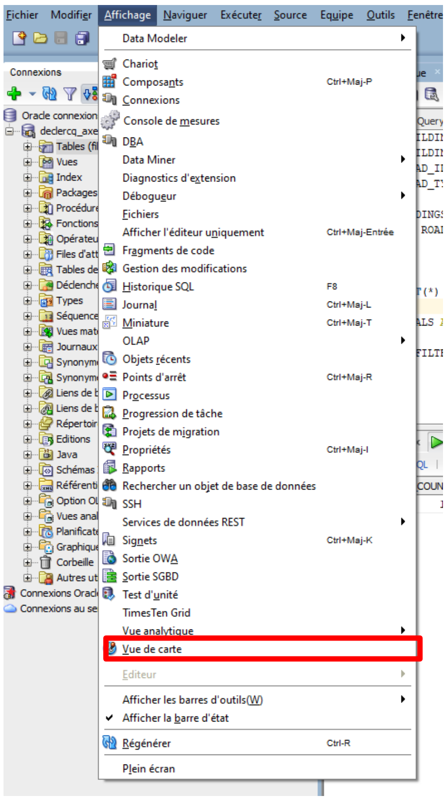
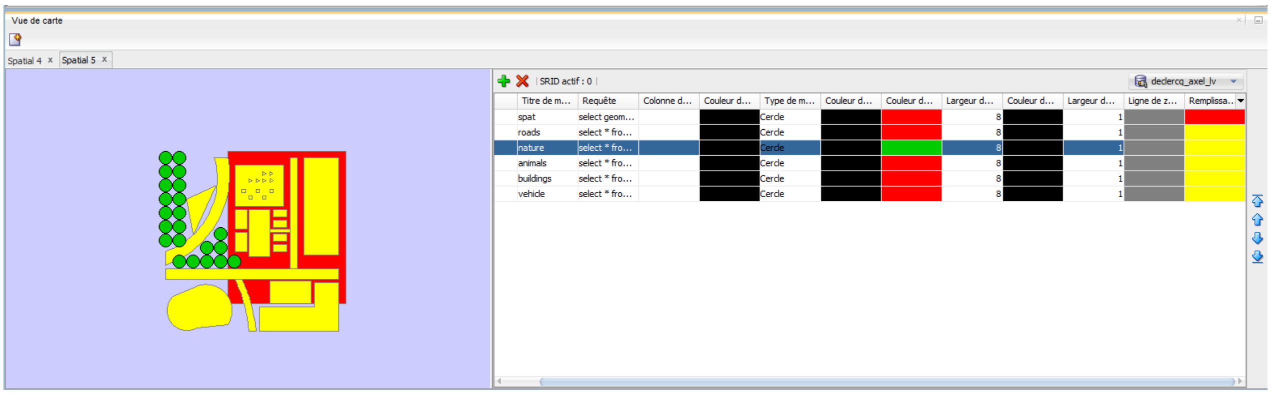


Figure 7.2. Find map view in SQL Developer

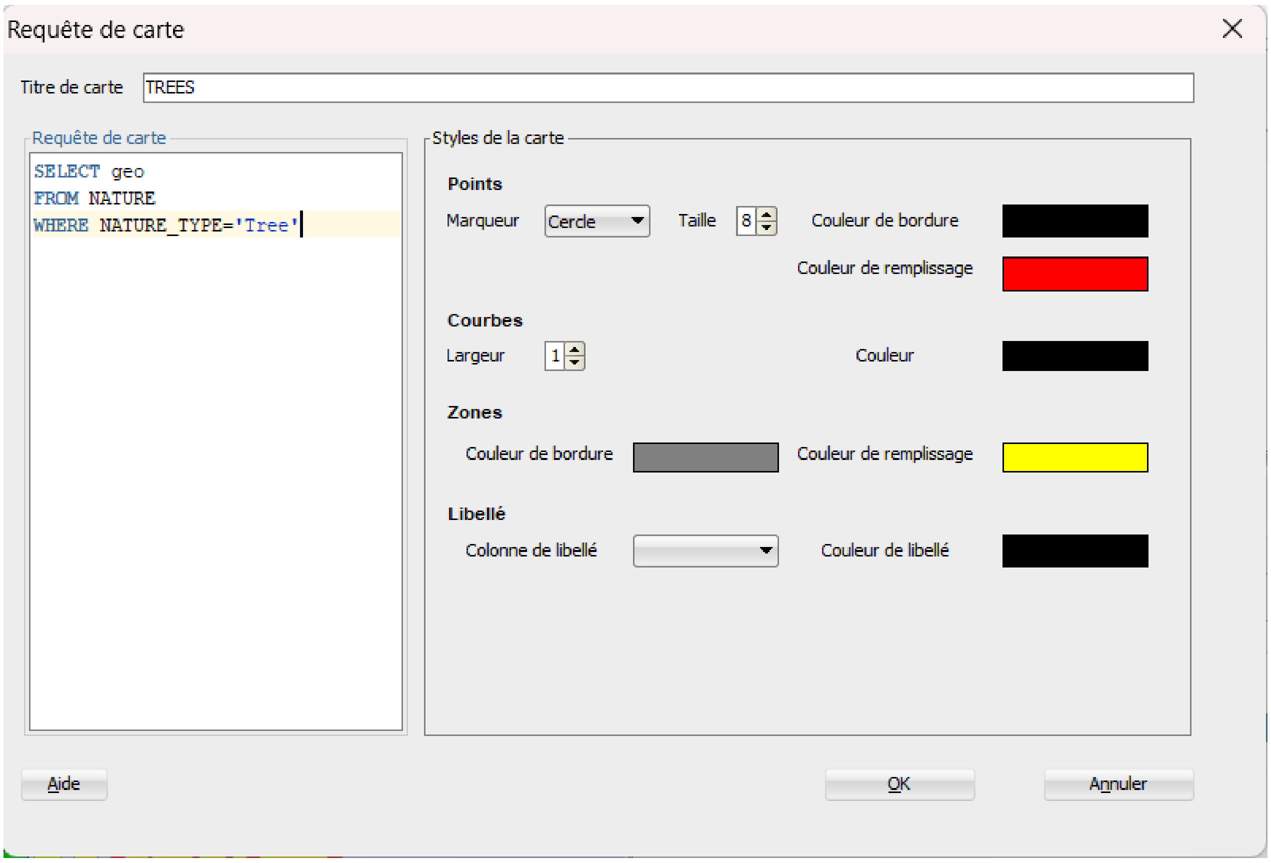
Then it displays a window like in the figure 7.3.



	Titre de m...	Requête	Colonne d...	Couleur d...	Type de m...	Couleur d...	Couleur d...	Largeur d...	Couleur d...	Largeur d...	Ligne de z...	Remplissa...
	spat	select geom...		⬛	Cercle	⬛	🟥	8	⬛	1	▨	🟥
	roads	select * fro...		⬛	Cercle	⬛	🟥	8	⬛	1	▨	🟨
	nature	select * fro...		⬛	Cercle	⬛	🟩	8	⬛	1	▨	🟨
	animals	select * fro...		⬛	Cercle	⬛	🟥	8	⬛	1	▨	🟨
	buildings	select * fro...		⬛	Cercle	⬛	🟥	8	⬛	1	▨	🟨
	vehicle	select * fro...		⬛	Cercle	⬛	🟥	8	⬛	1	▨	🟨

Figure 7.3. Map view window

To add a new object on the map, we have to click on the green cross and then the window on figure 7.4 is displayed.



Titre de carte: TREES

Requête de carte

```
SELECT geo
FROM NATURE
WHERE NATURE_TYPE='Tree'
```

Styles de la carte

Points
Marqueur: Cercle
Taille: 8
Couleur de bordure: ⬛
Couleur de remplissage: 🟥

Courbes
Largeur: 1
Couleur: ⬛

Zones
Couleur de bordure: ▨
Couleur de remplissage: 🟨

Libellé
Colonne de libellé:
Couleur de libellé: ⬛

Figure 7.4. Insert geometry in the map

For each object, “map view” section asks for a specific query including geometrical coordinates. Then, we can select some paramaters such as the shape of the points or the color of the shape.

9. Conclusion

Let’s summarize all the steps I did to realize this work and talk about conclusions I can make.

Firstly, I thought about how I could create my city. I wanted to create a farm scene that is close to the city I lived in France. I drew it quickly on a paper to visualize how it could look like and then I modelled the class I need. I realized my conceptual class diagram with draw.io. Once the classes I need and the city I want to create were clear, I implemented the table and the sequence. I set the metadata to fix a spatial space where I can work. Then, I inserted geometrical values in my tables. This step was long and tedious. Indeed, it requires orienting oneself within a Cartesian coordinate system. I did some little calculus such as slope coefficent, dilation, rotation, to build a city that is well organized. Finally, I was able to visualize the result. I inserted all the informational data and started to work on my queries. For the queries, I tried to find some interesting question that I could answer not in a simple way.

My conclusions regarding this work are the following. Firstly, to work with “map view” could be hard as when there are too many objects, it starts to bug and to slow the SQL Developer application. Then, it was really interesting to be free to create my own city and to visualize my databases on a coordinate system. I realized how useful it can be, especially for those who work with GIS data for example. As I navigated through this endeavor, it became evident that the power of spatial data extends far beyond mere visualization, offering valuable insights and practical applications in various fields.

If you are interesting in copy pasting my code, you can find it on my personal GitHub here.

I hope you find this work interesting. On my side, I enjoyed working on it a lot, as it was both creative and rich in knowledge.