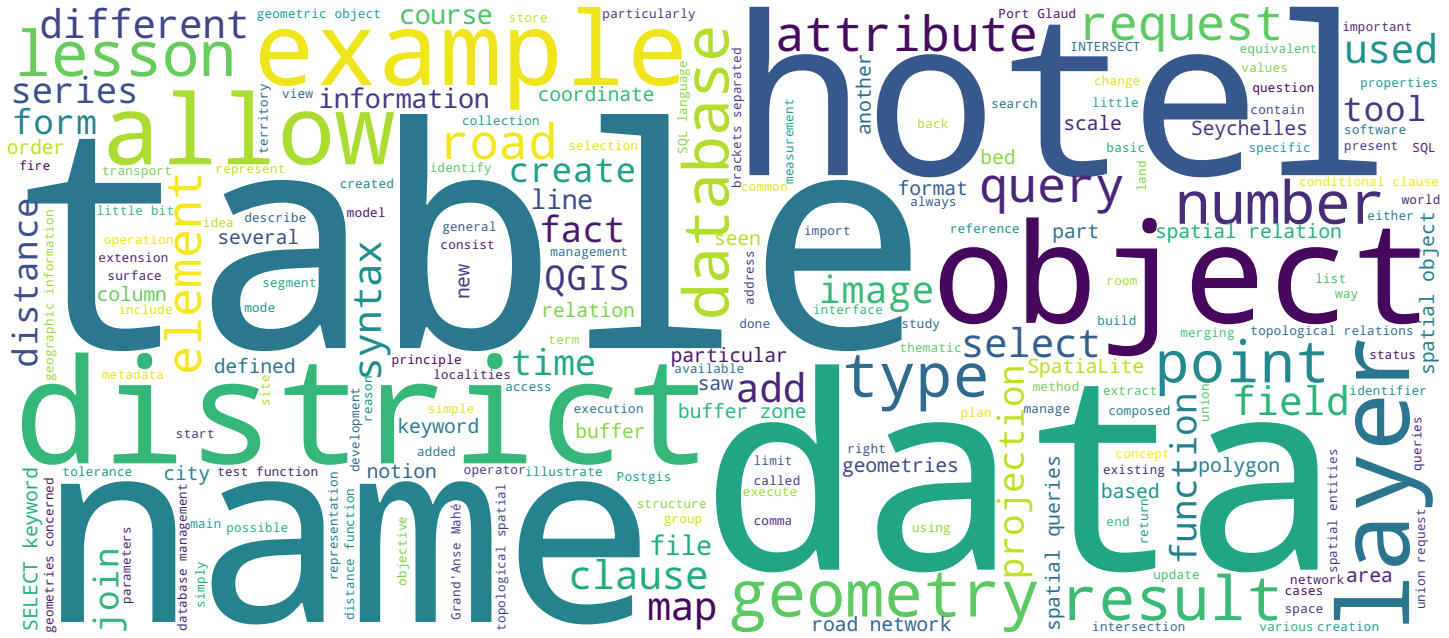
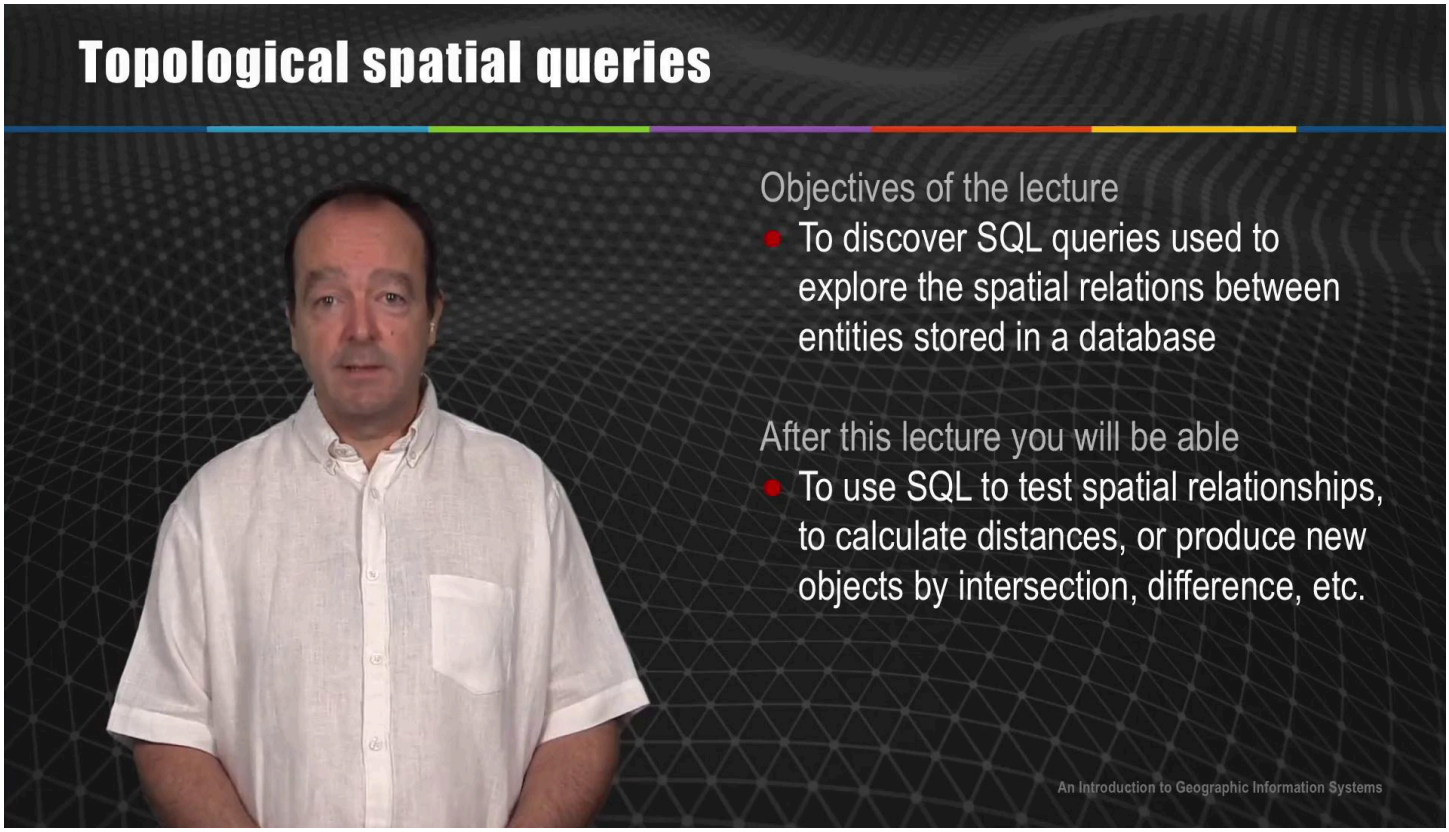
Topological spatial queries

An Introduction to Geographic Information Systems

Stéphane Joost, Marc Soutter, Fernand Kouamé, Amadou Sall





In this lesson, we will address the question of topological spatial queries, so something that completes the geometric spatial queries that we saw in the previous lesson. These topological spatial queries are therefore interested in extracting information which characterize not the spatial objects themselves but rather the spatial relations and the relations between spatial objects. The objective of this lesson is to explore the syntax field of SQL queries which allow to extract information characterizing spatial relations between objects hosted in a database. At the end of this lesson you should be able to write queries which will allow you to test spatial relations between objects, to calculate characteristics such as the distance between objects or to make new objects by intersection or difference or other operations involving existing spatial objects.

Notes

Summary

0m 23s

Topological spatial queries

http://www.gaia-gis.it/gaia-sins/spatialite-sql-4.2.0.html

SQLite

http://postgis.net/docs/index.html

PostgreSQL

PostGIS

Spatial PostgreSQL

Just as in the case of geometric spatial queries, topological spatial queries are numerous.

Notes

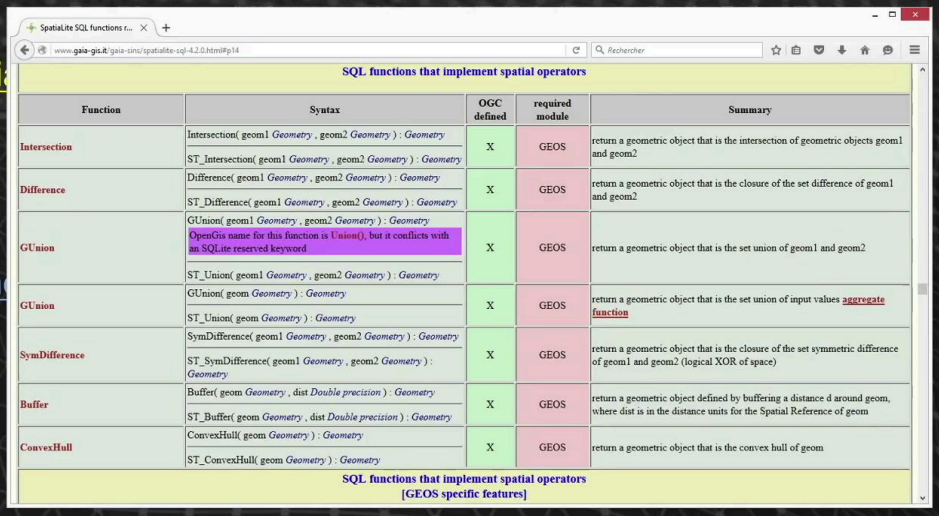
Summary

1m 22s

Topological spatial queries

http://www.gaia-gis.it/gai

http://postgis.net/docs/in



SQL functions that implement spatial operators

Function	Syntax	OGC defined	required module	Summary
Intersection	Intersection(geom1 *Geometry* , geom2 *Geometry*) : *Geometry* ST_Intersection(geom1 *Geometry* , geom2 *Geometry*) : *Geometry*	X	GEOS	return a geometric object that is the intersection of geometric objects geom1 and geom2
Difference	Difference(geom1 *Geometry* , geom2 *Geometry*) : *Geometry* ST_Difference(geom1 *Geometry* , geom2 *Geometry*) : *Geometry*	X	GEOS	return a geometric object that is the closure of the set difference of geom1 and geom2
GUnion	GUnion(geom1 *Geometry* , geom2 *Geometry*) : *Geometry* OpenGis name for this function is Union(), but it conflicts with an SQLite reserved keyword ST_Union(geom1 *Geometry* , geom2 *Geometry*) : *Geometry*	X	GEOS	return a geometric object that is the set union of geom1 and geom2
GUnion	GUnion(geom *Geometry*) : *Geometry* ST_Union(geom *Geometry*) : *Geometry*	X	GEOS	return a geometric object that is the set union of input values aggregate function
SymDifference	SymDifference(geom1 *Geometry* , geom2 *Geometry*) : *Geometry* ST_SymDifference(geom1 *Geometry* , geom2 *Geometry*) : *Geometry*	X	GEOS	return a geometric object that is the closure of the set symmetric difference of geom1 and geom2 (logical XOR of space)
Buffer	Buffer(geom *Geometry* , dist *Double precision*) : *Geometry* ST_Buffer(geom *Geometry* , dist *Double precision*) : *Geometry*	X	GEOS	return a geometric object defined by buffering a distance d around geom, where dist is in the distance units for the Spatial Reference of geom
ConvexHull	ConvexHull(geom *Geometry*) : *Geometry* ST_ConvexHull(geom *Geometry*) : *Geometry*	X	GEOS	return a geometric object that is the convex hull of geom

SQL functions that implement spatial operators
[GEOS specific features]

There are many keywords that can be used.

Notes

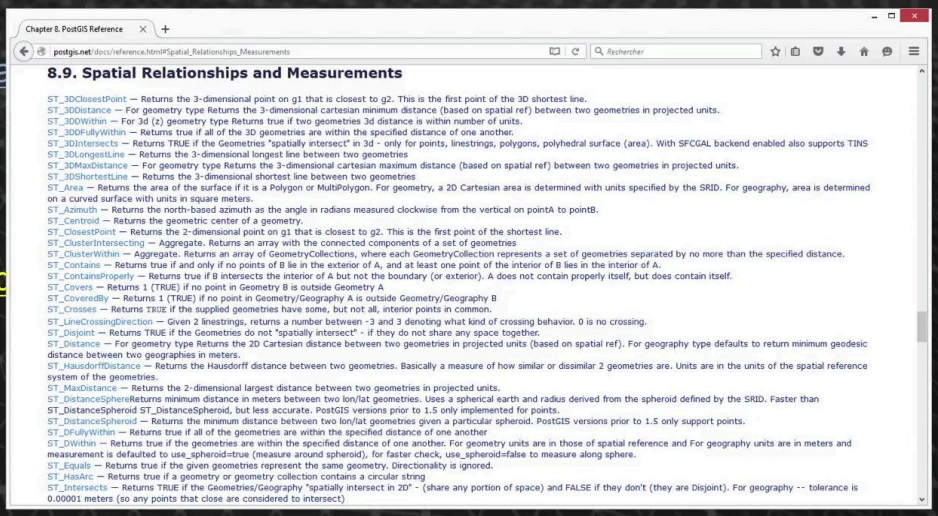
Summary

1m 30s

Topological spatial queries

http://www.gaia-gis.it/gaia

http://postgis.net/docs/inc

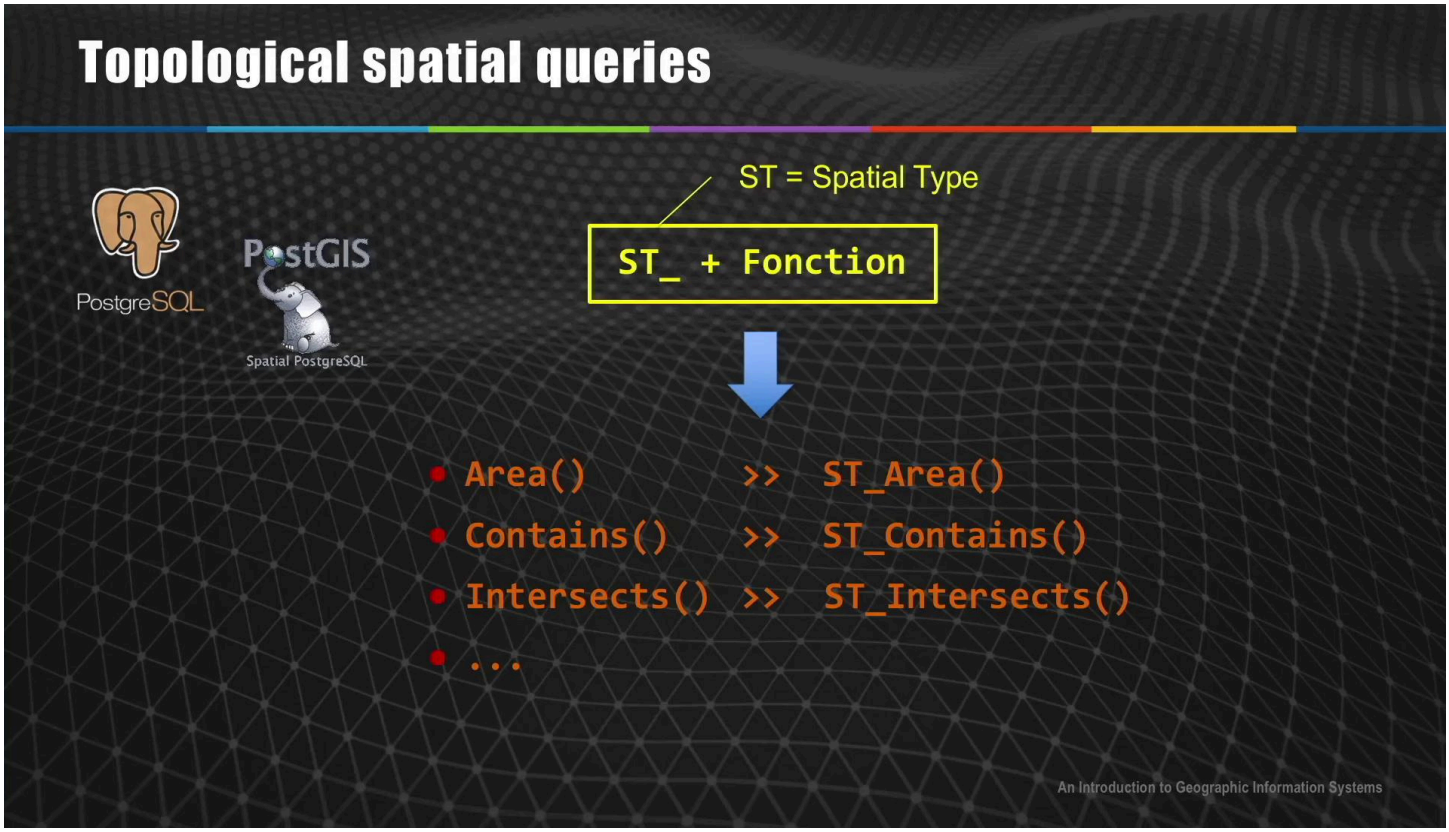


We will present only a very small part of it so this is the reason why I encourage you to take a look on the sites that document the available functions in SpatiaLite and in Postgis.

Notes

Summary

1m 38s



Again, the syntax is pretty much the same in both cases with, in the case of Postgis, the addition of the letters S and T underlined before the function.

Notes

Summary

1m 50s

Notion of topological query

Spatial relations

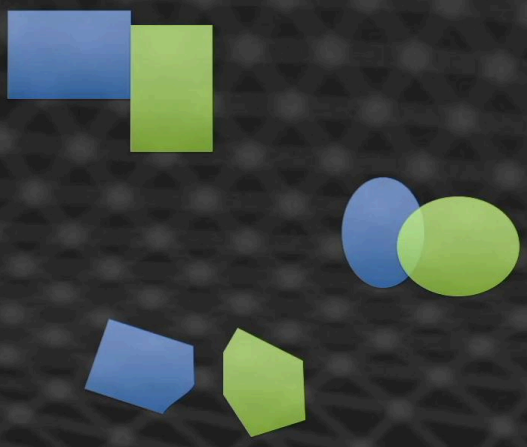
- Any property shared by two spatial entities defines a spatial dependency or spatial relation

➡ Distances, …

Topological relations

- Qualitative spatial relations or properties, independent from any measure, invariant under continuous deformation

➡ Adjacency, Connectivity, Inclusion, Intersection

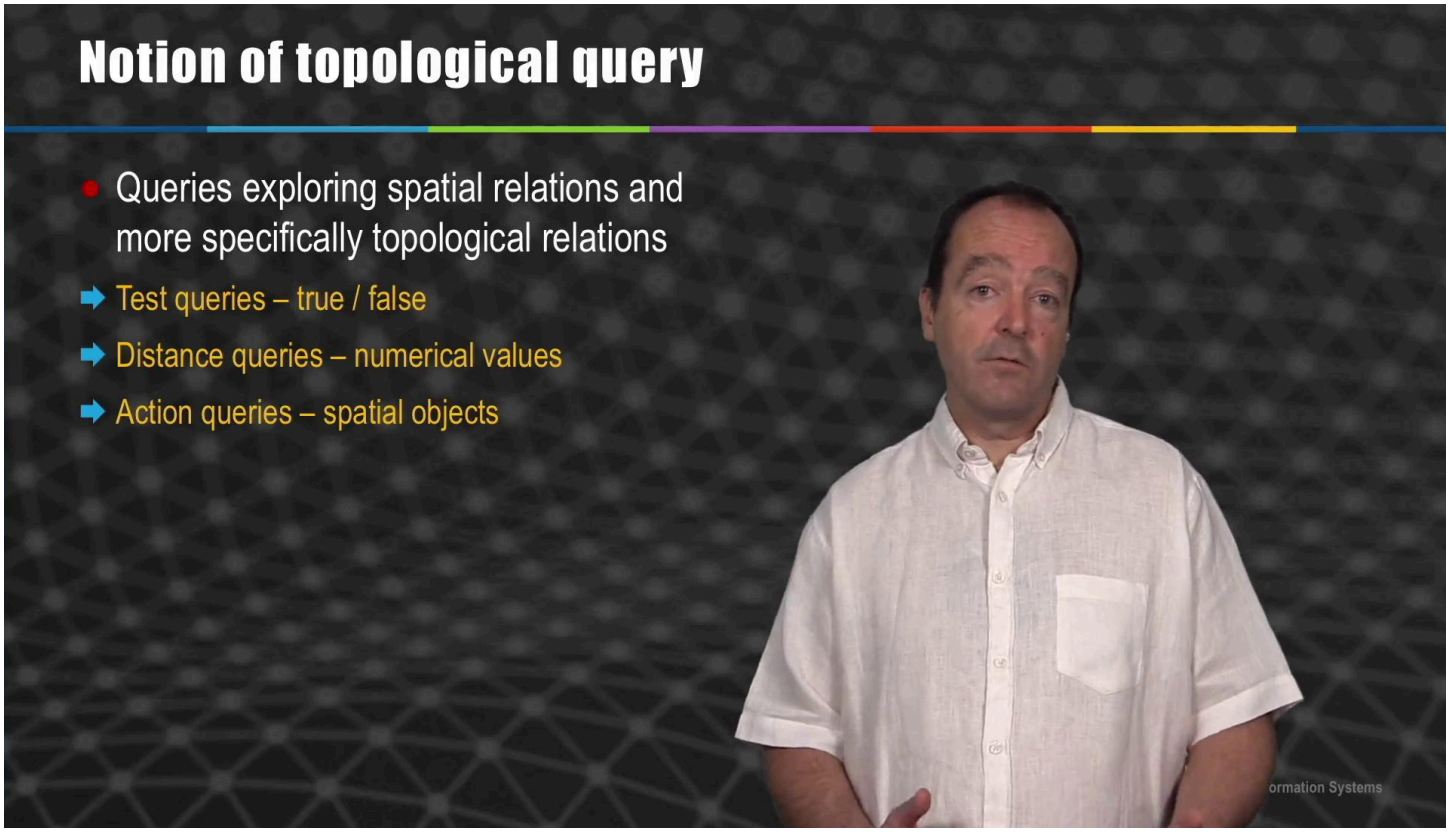


In this lesson, we will first go back to the notion of topology and topological query, notion which had been presented at the beginning of the course. Then we will present the test functions before addressing the distance function and ending by spatial operators. We had seen by presenting the notions of topologies that a relation, a spatial dependence is defined by a property shared by two spatial entities. As for example the simplest one of them, the distance that separates two points. We also saw that the topological relations are a sub category of spatial relation, of a qualitative nature, independent of the measure and characterized by the fact of being invariant under continuous deformation.

Notes

Summary

1m 57s



The main forms of topological relations are the adjacency, the connectivity, the inclusion and the intersection. The requests which are on the spatial relations between objects and more particularly on topological relations can be subdivided roughly into 3 categories. First, the test requests which return "true" or "false" values.

Notes

Summary

2m 55s

Test functions

Testing spatial relations

- Functions used to test topological relations between two spatial entities, on the basis if their geometries
- Syntax

```
SELECT FUNCTION (geometry1, geometry2) FROM name_table
FUNCTIONS:
Equals(), Disjoint(), Touches(), Within(), Overlaps()
Crosses(), Intersects(), Contains(), Relates()
```

An Introduction to Geographic Information Systems

Distance queries which return a numeric value and action requests which return spatial objects. Test requests or test function allow to test spatial relations or more precisely the topological relations between two spatial entities on the basis of their geometry. The syntax is always the SELECT keyword, the test function with the two geometries concerned in brackets, separated by a comma, the FROM clause and the table. The main functions that can be used are listed here in particular, we can test the equality between two geometries, the fact that they are distant from one another, that they touch each other, that one is contained into the other or that it contains the other, that they overlap, that they cross each other, etc.

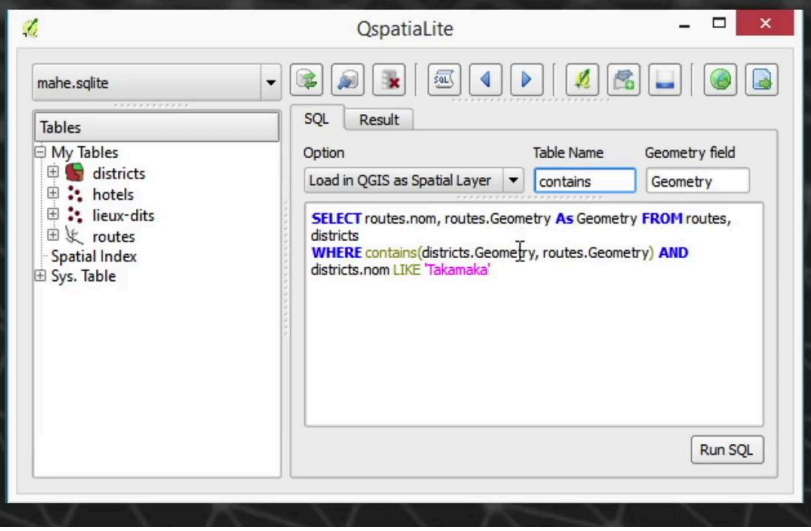
Notes

Summary

3m 21s

Test functions

Testing spatial relations



In this example we will select the hotels the name of the hotels which are located within the Takamaka districts. To do this, the syntax is SELECT, the name of the hotel, the FROM clause with the two hotels and districts tables and a conditional clause in which we specify the fact that the geometry of the hotels must be inside the geometry of the districts and that the district name should resemble Takamaka. By executing this query we get the 6 hotels which are in this district. We can now, in a second example, replace the hotels by the roads and search for the roads which intersect the district of Takamaka. So we see that we have a series of 21 roads which form an intersection with that district. To store the result of this query as a new layer we have to add in the request a "select the geometry attribute" column and define with the geometry name, which is the field that we will use to make the layer in QGIS, give a name to this layer, in this case INTERSECT, and execute this request. We make a second request to select this time the roads that touch the district of Takamaka. We can see that this request does not give any result, so there is no object which properly touches the district and a third request to identify the roads which are contained in the district.

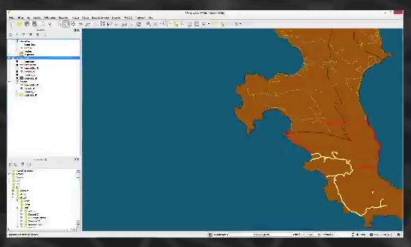
Notes

Summary

4m 16s

Test functions

Testing spatial relations



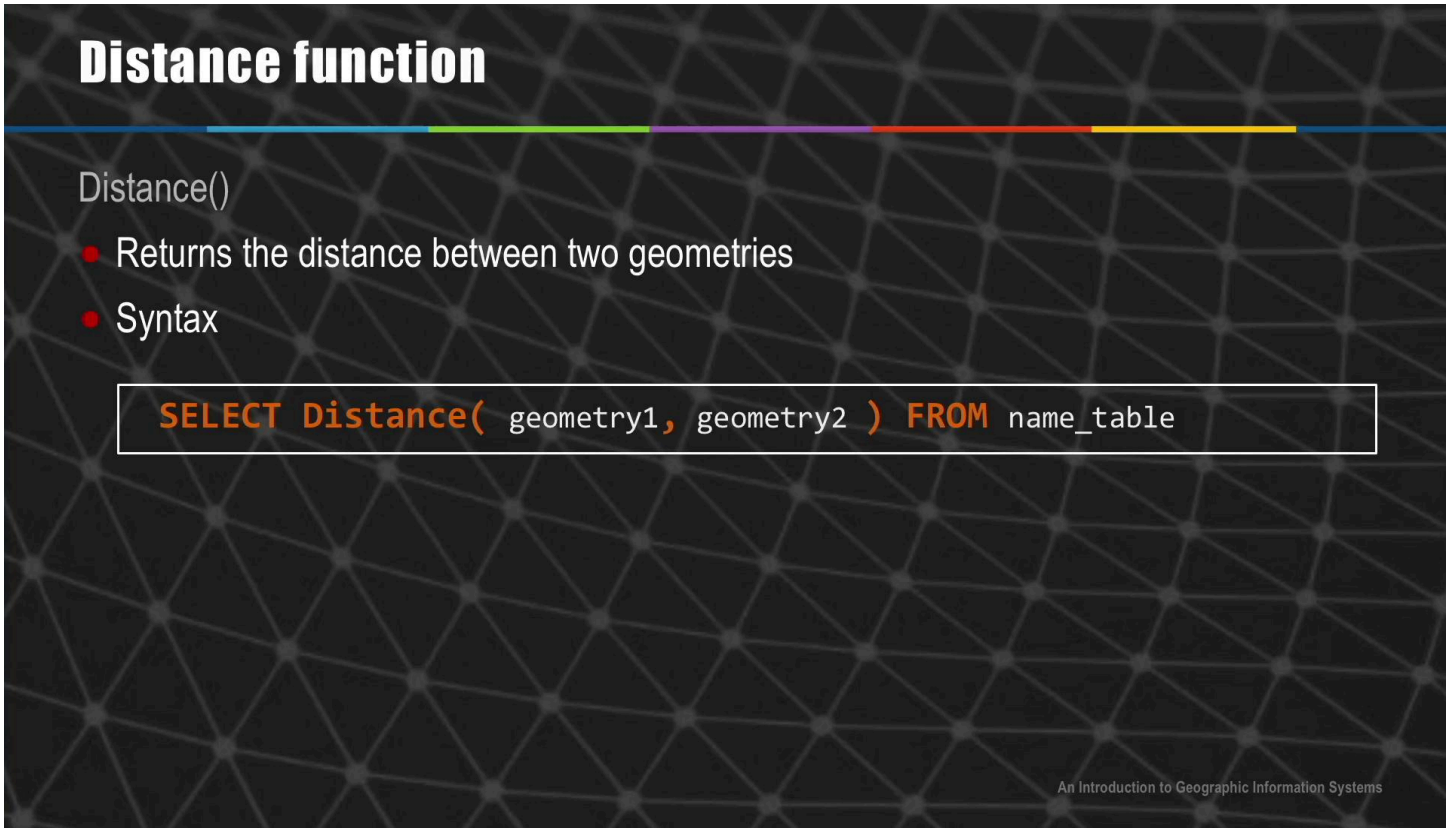
An Introduction to Geographic Information Systems

We also export the result of that query as a layer in QGIS and if we now look at these two layers, the "intersection" layer in red and the "content" layer in yellow, it is clear that the intersection layer includes more objects than the layer of what is strictly contained in the district.

Notes

Summary

6m 18s



The distance function simply allows to calculate the minimum distance separating two geometries, generally expressed in meters, well it depends on the projection system used. The syntax is of the same nature as in the previous cases, the SELECT keyword, the distance function, the two geometries concerned in brackets, separated by a comma, the FROM clause, and the name of the table.

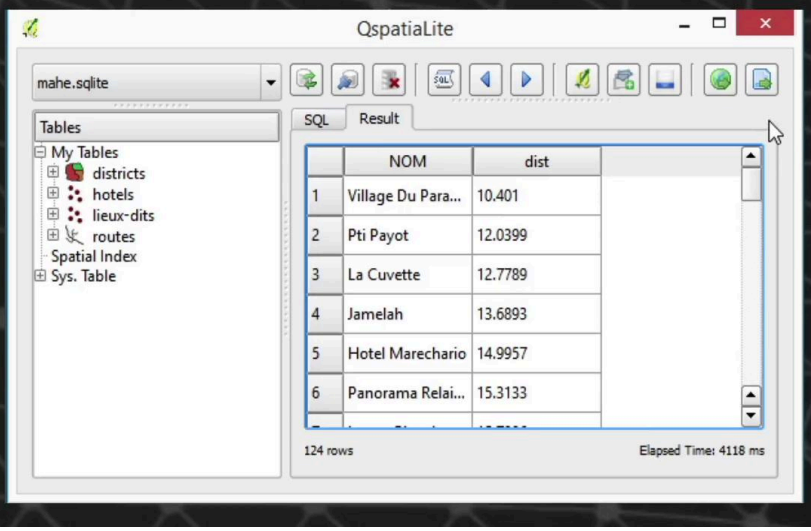
Notes

Summary

6m 44s

Distance function

Distance()

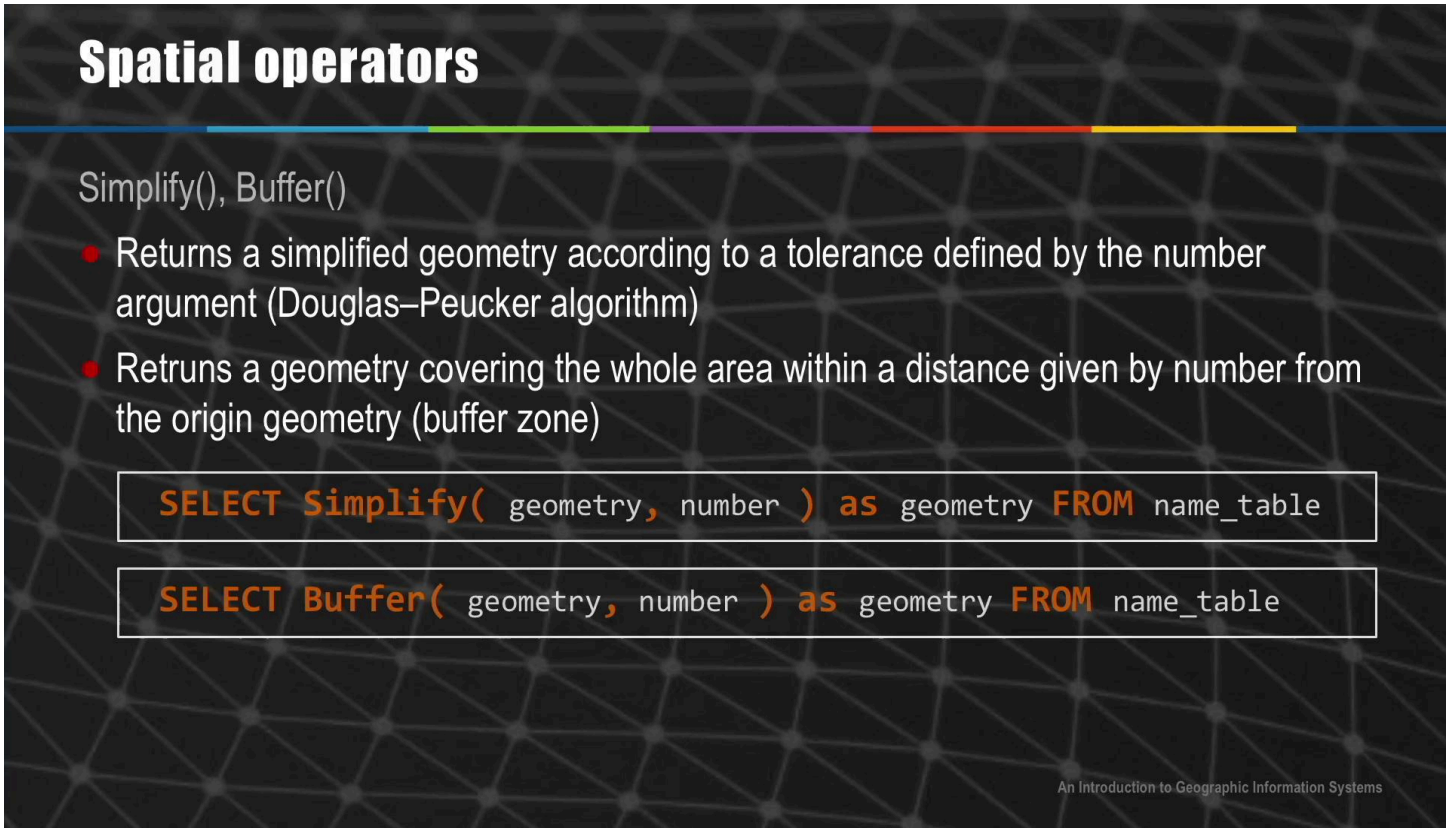


In this example, we will build a query to classify the hotels according to their distance from the road network. In the syntax, we will start with the SELECT keyword, the name of the hotels and a second factor which will be the distance between the hotel geometries and the geometry of the roads and we will take the minimum value of all the distances separating a hotel from all the roads. The field "calculate the minimum distance" is given an alias, DIST, in the FROM clause the two tables concerned, so hotels and roads, 183 00:07:49,807 --> 00:07:52,526 a grouping clause, the name of the hotels and then a ranking in increasing order of distances.

Notes

Summary

7m 10s

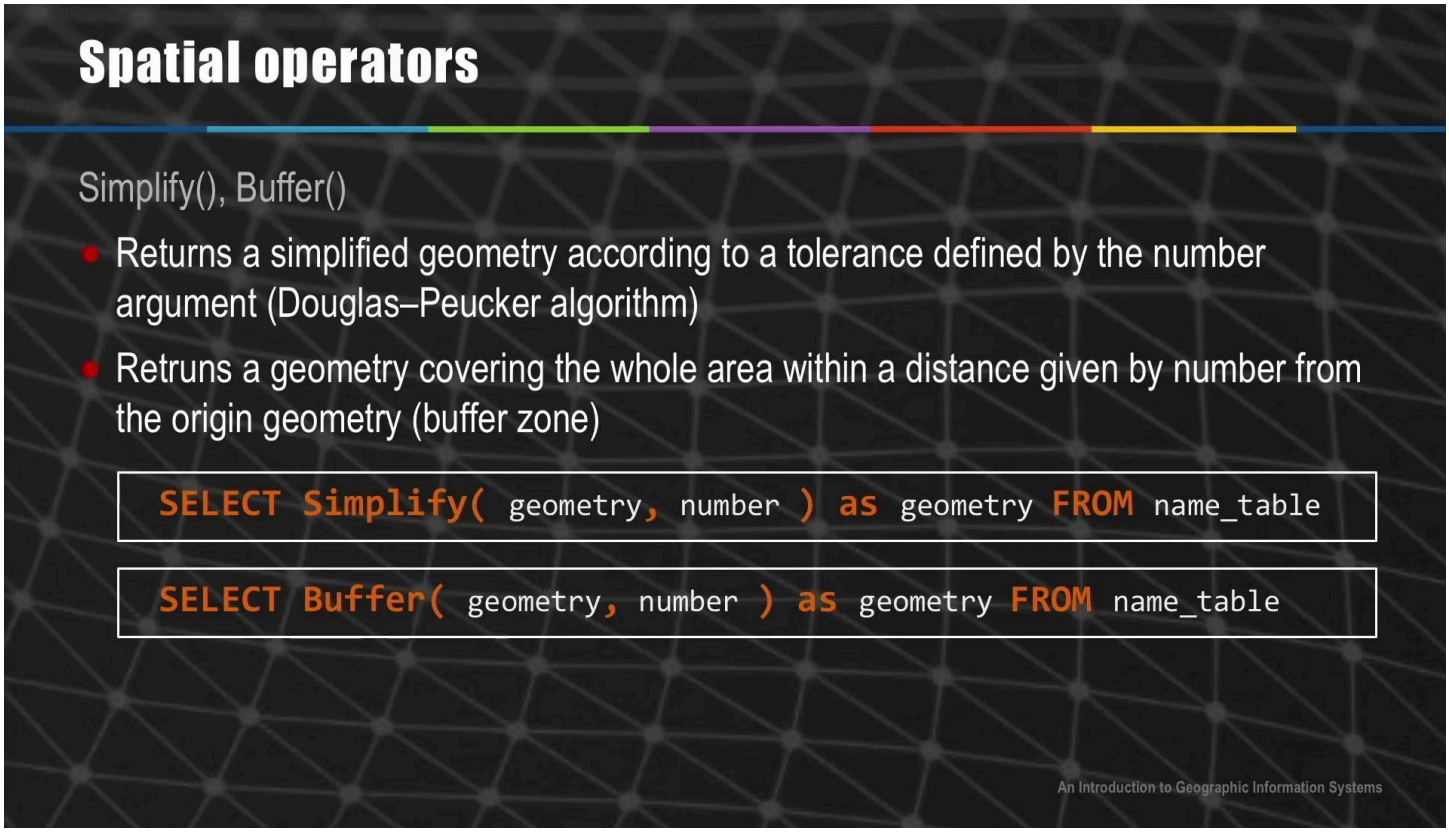


By executing this query we find the series of 124 hotels with the distance which separates them from the road network, between 10 and 600m. A series of spatial operators to finish, so operators that return new geometric objects created from existing geometric objects. The first of these operators is the union query which allows to merge two geometries into a single object. The intersection request which returns the common part of two geometries. The difference request which subtracts a geometry of another. The syntax, as usual is composed by the SELECT keyword Followed by the function, the two geometries concerned in brackets, separated by a comma, the AS GEOMETRY alias to be able to reuse this result, say that it is a geometry and the table from where the data come. As in the case of the length of the polylines, the union request, the UNION keyword, is part of a series of keywords reserved in SpatiaLite, that is why we use GUnion instead and as always in Postgis the equivalent will be ST underlined UNION. Two other important space operators, SIMPLIFY and BUFFER.

Notes

Summary

8m 01s



The first, SIMPLIFY, refers to a simplified geometry, taking into account a tolerance which is defined by the numerical argument given in the function, so it is a simplification which is based on a Douglas-Peucker algorithm, which allows to reduce the number of points of which a geometry is composed, which can be important in the display performances of the geometry on the screen. The BUFFER function returns a geometry which covers an entire area located at a distance inferior or equal to the number given as an attribute of an original geometry. It is the concept of buffer zone. At the syntax level still the SELECT keyword, the function, the geometry, comma, the numeric variable, a tolerance in the case of SIMPLIFY, a distance in the case of BUFFER, AS GEOMETRY because we create geometries and the original table.

Notes

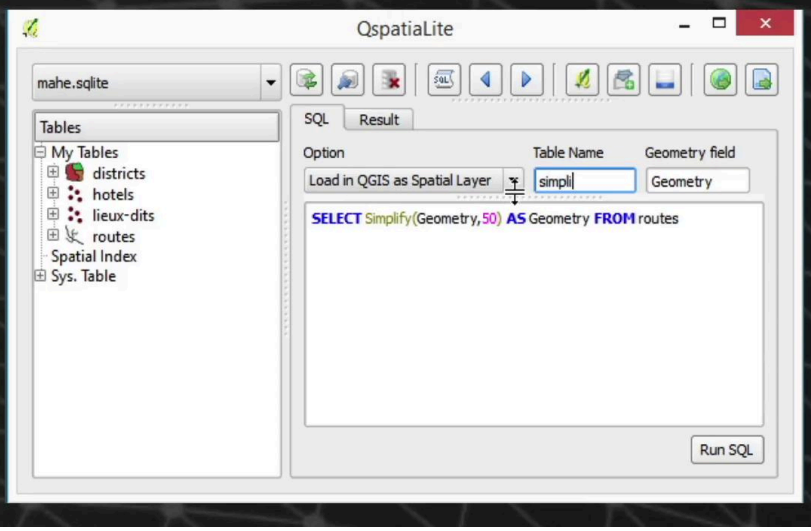
Summary

9m 39s

Spatial operators

Union(), Intersect(), Difference()

Simplify(), Buffer()

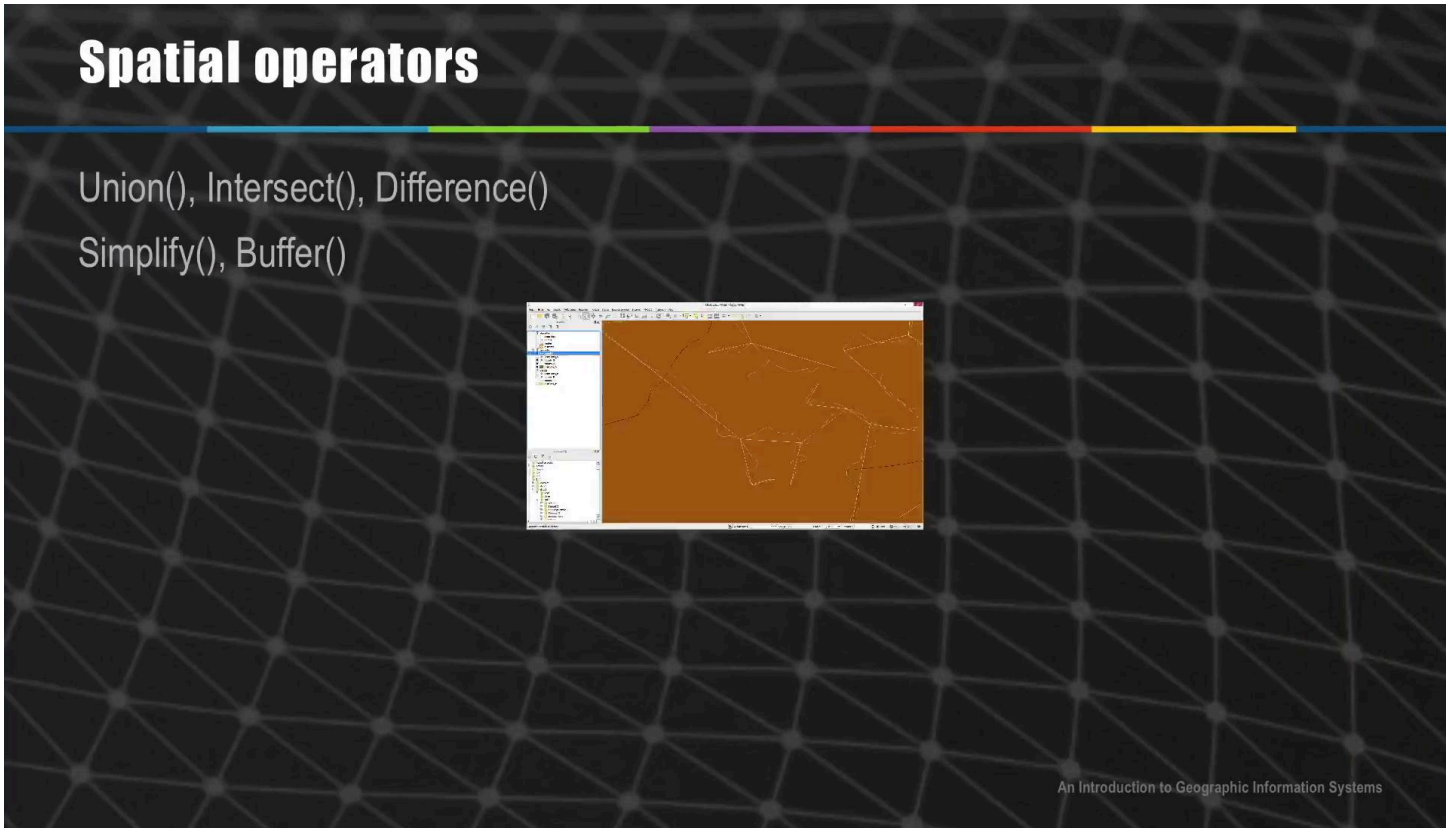


In this first example we will illustrate the use of the SIMPLIFY function by simplifying the road network of the Seychelles with a tolerance of 50. We see that the request produces geometric objects indeed. They have to be defined as geometries to be able to recover them in QGIS as usual. A name is given to the table, SIMPLIFY, the geometry being of course the GEOMETRY field.

Notes

Summary

10m 43s



This simplified layer appears here in white and if we go a little bit closer we see that the detail of the road network has been erased in favor of a simpler structure.

Notes

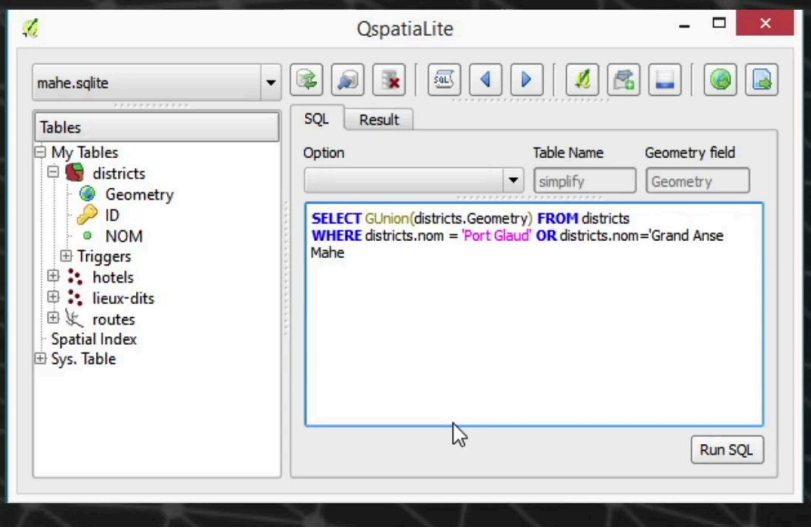
Summary

11m 15s

Spatial operators

Union(), Intersect(), Difference()

Simplify(), Buffer()



Second example with the union. We will establish a union request between two districts, the district of Port Glaud and the district of Grand'Anse Mahé. We transfer the union request on the geometry field of the district table and in the conditional clause which clarifies things we say that we want the name of districts chosen for the union to be either Port Glaud or Grand'Anse Mahé.

Notes

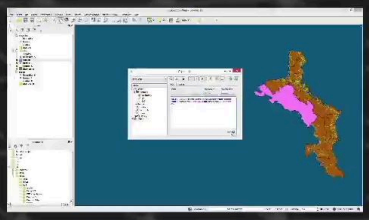
Summary

11m 30s

Spatial operators

Union(), Intersect(), Difference()

Simplify(), Buffer()



An Introduction to Geographic Information Systems

The execution of this request causes the merging of the two entities.

Notes

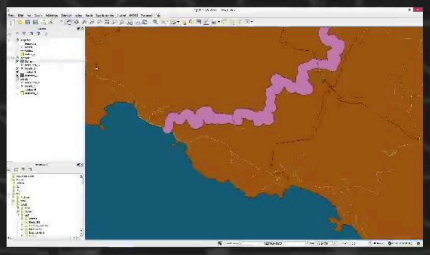
Summary

12m 10s

Spatial operators

Union(), Intersect(), Difference()

Simplify(), Buffer()



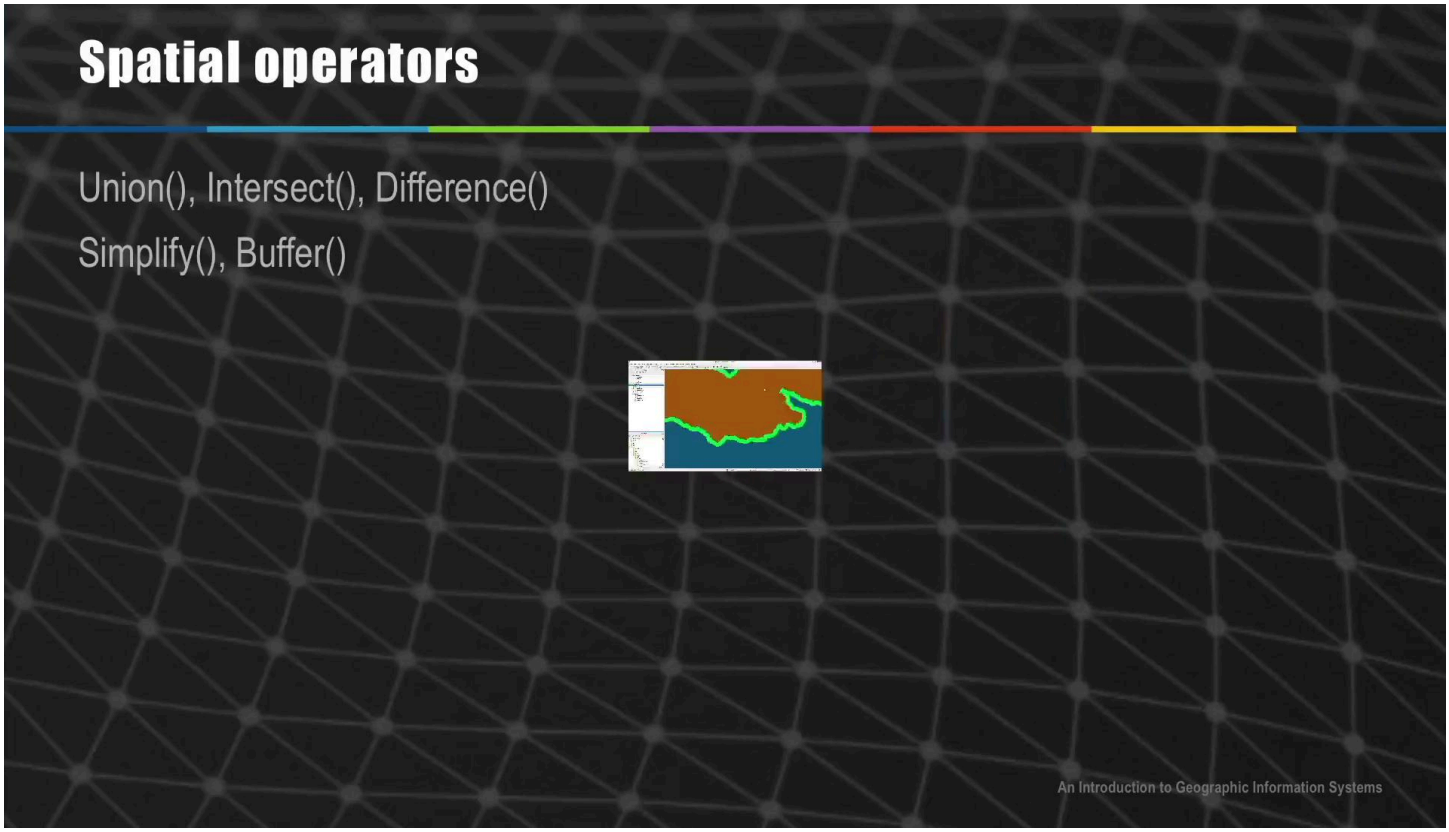
An Introduction to Geographic Information Systems

Third case, we will build a buffer zone, a buffer, around the Sans Soucis road which is located in the center-western part of the Seychelles. So we define a buffer with a width of 100m around that road and in the conditional clause we will simply say that we want all the segments of the road which look like Sans Soucis road. The execution of this request provides geometric objects. With the AS GEOMETRY and by defining a table name, we add this request as a layer in the QGIS interface and we see that at the time I execute the request the buffer zone is displayed on the map, here in pink, and if we look a little closer we see that the different sections of the road will give rise to a succession of buffer zone pieces which fit with each other.

Notes

Summary

12m 17s



As a final example, we will create a buffer zone around the two districts of Port Glaud and Grand'Anse Mahé by taking up the merging request of these elements from before and this time we will add the creation of a buffer to this merging. If we move the district layer above this buffer and that we look little closer we see that we have created a 100m block around these two merged districts.

Notes

Summary

13m 26s