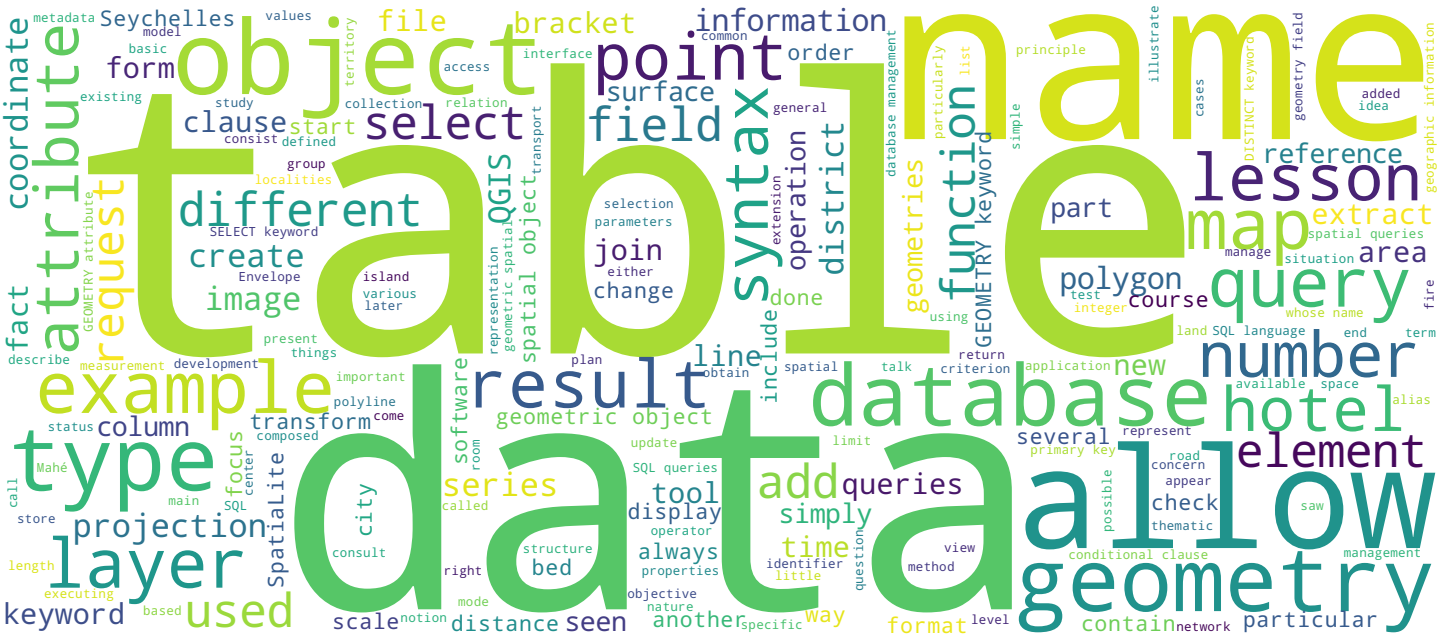
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

RESCIF
Réseau d'excellence des sciences de l'ingénieur de la Francophonie

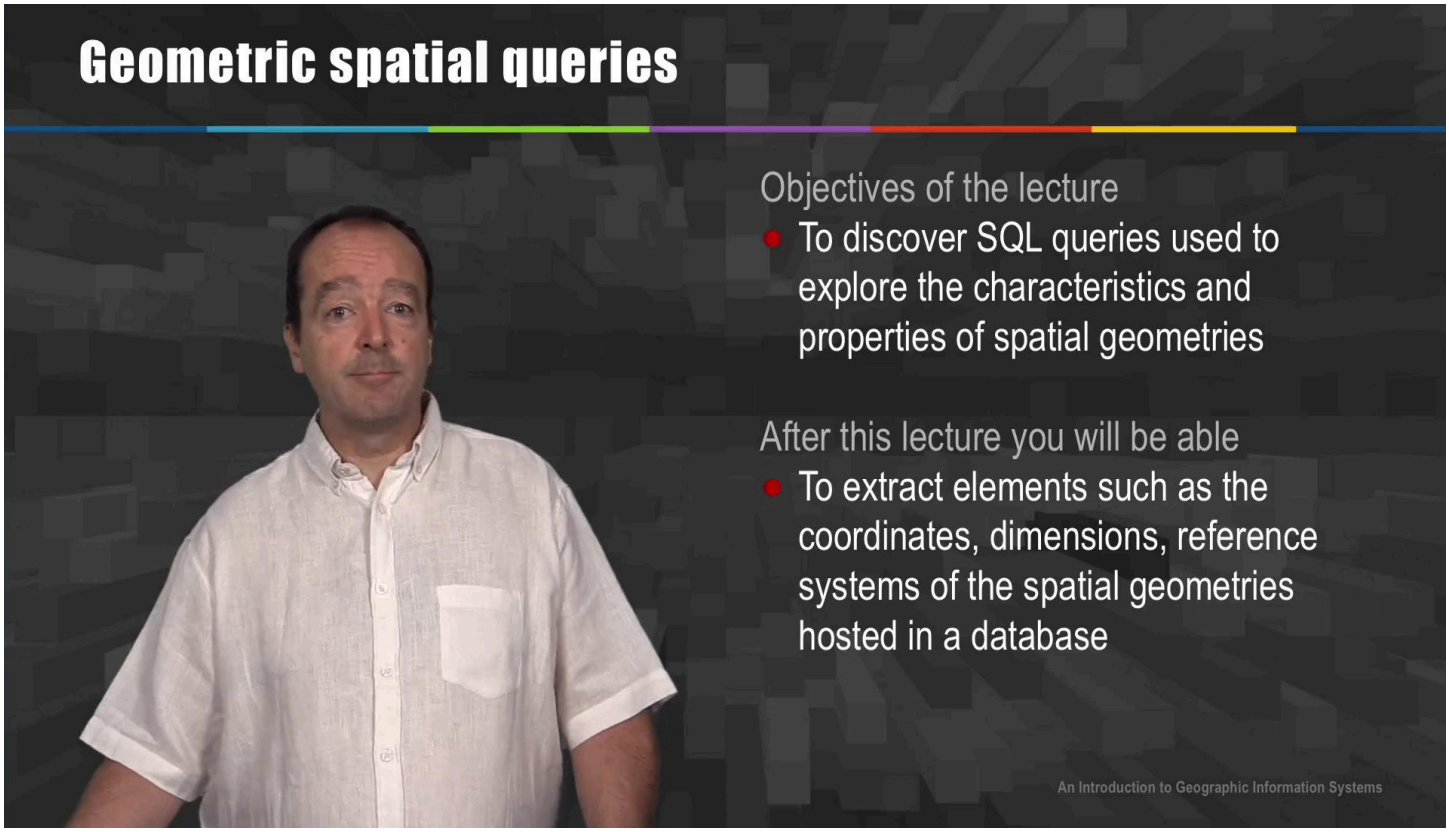
An Introduction to Geographic Information Systems

Geometric spatial queries

Stéphane Joost, Marc Soutter, Fernand Kouamé, Amadou Sall



EPFL

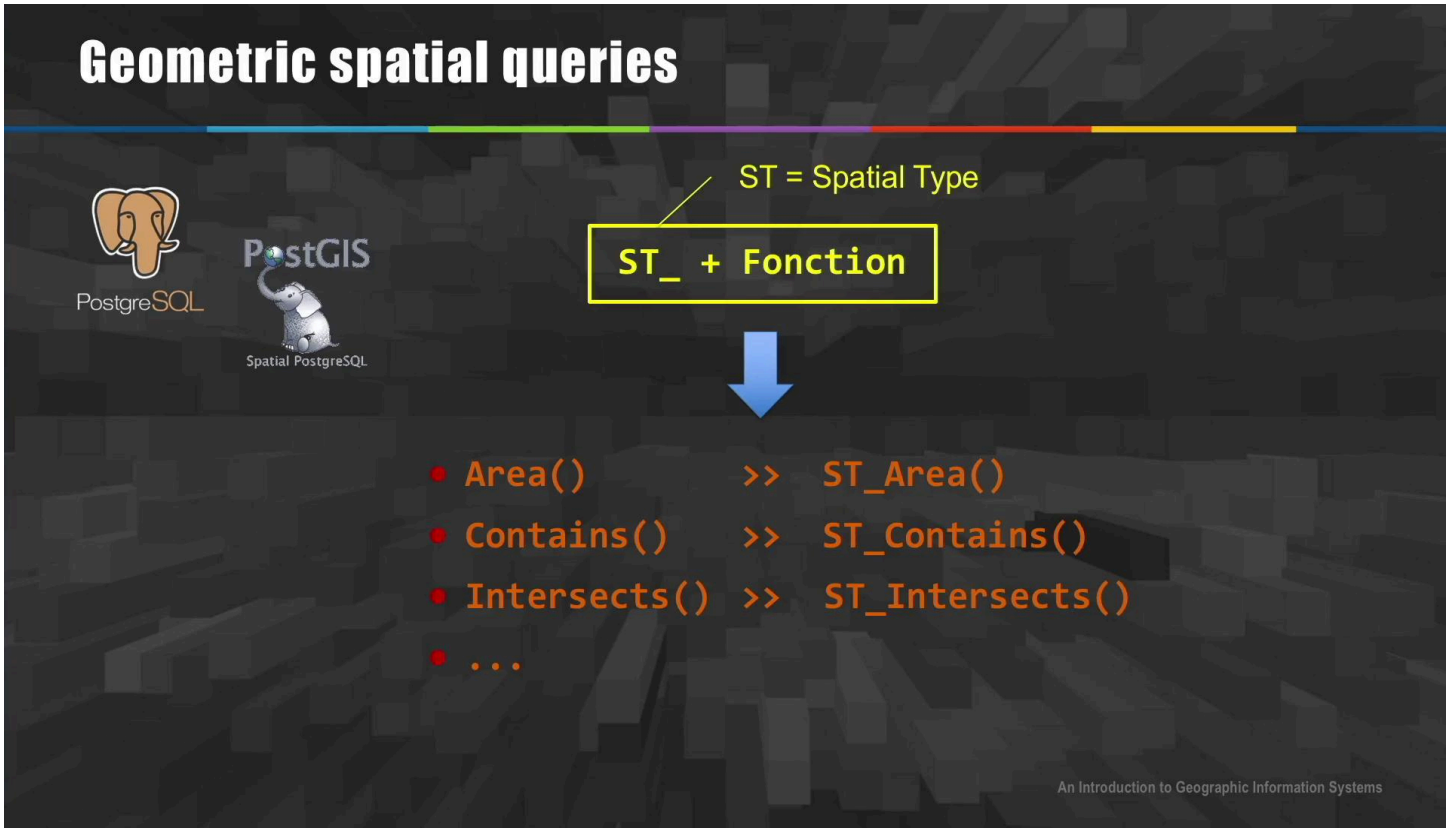


Hello and welcome to this lesson which will focus on geometric spatial queries, a set of queries that is more specifically focused on extracting information on the nature and geometric properties of spatial objects stored in a database.

Notes

Summary

0m 22s



The objective of this lesson is to explore this world of SQL queries that seeks to extract geometric characteristic properties from spatial objects stored in a database, so that you are able to retrieve information such as the coordinates of a point, the perimeter or length of a line, the surface of a polygon from spatial geometries hosted in a database. There is a large number of different geometric spatial queries with a wide variety of keywords that we cannot obviously all review so we will present you some situations, some typical queries which we consider particularly important. It's always a good idea to check out the sites of specialized softwares, in this case the SpatiaLite site and PostGIS site, to get a more complete idea of all the available functionalities in terms of geometric spatial queries. In general, the syntax is pretty much the same in all cases with a particularity however in the PostGIS world which, when the query is on spatial objects, prefaces the keyword with the underlined letters ST, ST for Spatial Type, which allows, in a SQL query, to clearly distinguish everything that is addressing spatial properties and that are specific to the PostGIS world versus PostgreSQL world on which PostGIS is built. But for the rest, the syntax is the same as in SpatiaLite with a few rare exceptions.

Notes

Summary

1m 07s

Geometry and spatial reference

geometry

- The geometry attribute represents the geometry of the spatial object and is used for example in a select request performed to draw objects on a map
- … or to extract the used spatial reference system
- Syntax

```
SELECT geometry FROM name_table
```

```
SELECT SRID(geometry) FROM name_table
```


In this lesson, we will first talk about the queries that allow to select and to display geometries, display in a map, or to consult the spatial reference used, so the ESPG code. We will then see some syntax of SQL queries, of data conversion to transform geometries in text or binary format, queries that allow to extract, to consult the type of geometry with which we are dealing and finally we will see a series of queries that allow to retrieve the specific properties of geometries. The GEOMETRY keyword typically refers to the geometry attribute of a spatial data table, so the geometry of the spatial object and it is used in selection queries to display objects on a map. It can also be used to extract the reference of the projection system used or a lot of other information that we will see later. The syntax is based simply on the SELECT, keyword, GEOMETRY, we select the geometries, FROM, the name of the table to indicate the table from which the geometries are extracted. Then, we can imagine conditional clauses which allow to sort, in the geometries, those that meet certain criteria. We will see an example later. The SRID keyword with the GEOMETRY keyword in brackets to extract the reference of the projection system used, in this case the ESPG code. An example of the application of these elements.

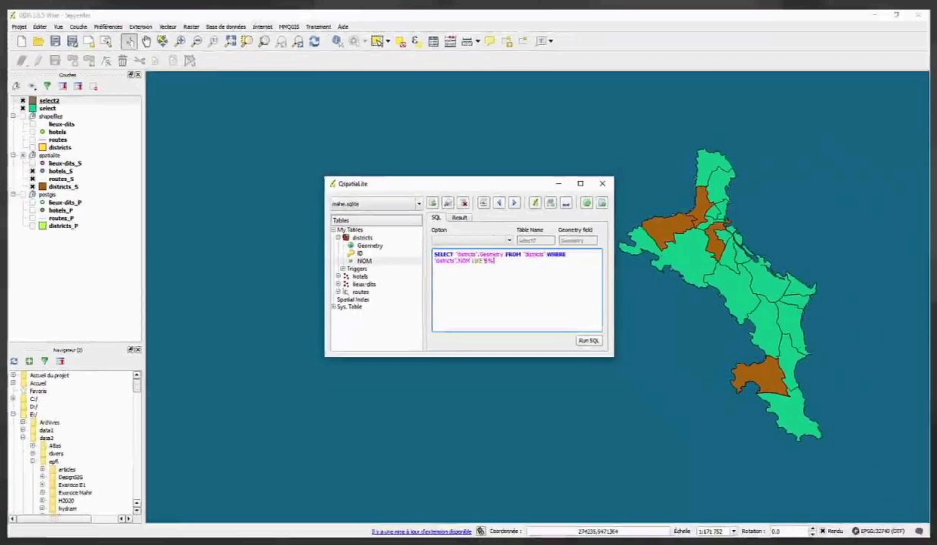
Notes

Summary

2m 26s

Géométrie et référence spatiale

geometry



Introduction aux systèmes d'information géographique

First, when it comes to selecting objects, the easiest way is to use the integrated tool in QGIS with a selection module which uses the SQL language but the SELECT GEOMETRY FROM is hidden and we simply add the criterion in the SQL syntax, here the name that begins with d, and we select all the districts whose name begins with d. Second example, this time in the SpatiaLite module, again with a query in the districts table to select all districts of Mahé. To display the results of the query, they must be loaded as a new layer in QGIS, a layer whose name and field that contains the geometries must be defined. Finally, just as we did with the QGIS query, we can change the SpatiaLite request by introducing a conditional clause to limit the selection to districts whose name begins with d.

Notes

Summary

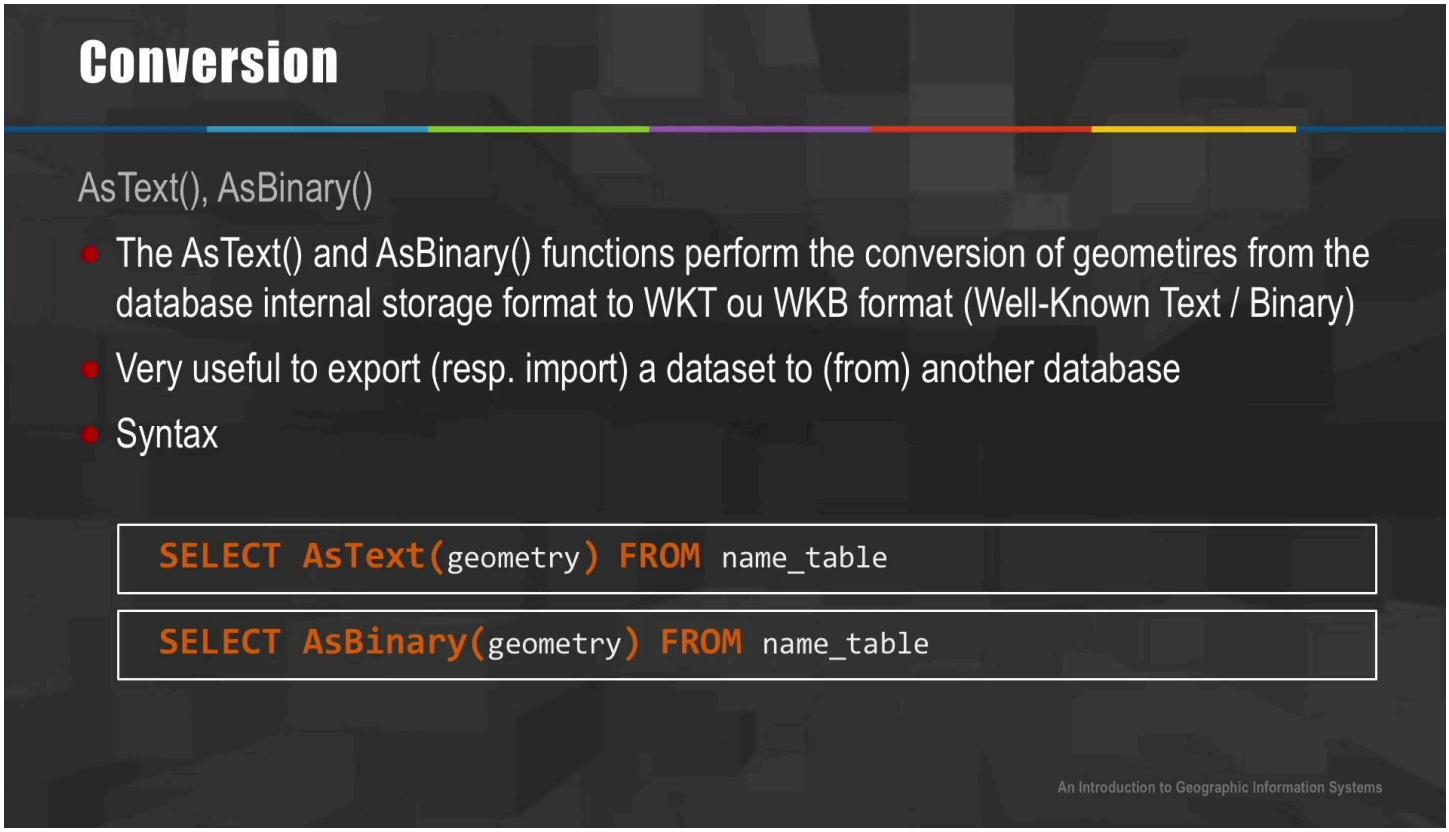
4m 14s

And again, we have to define the result of the query as a new layer to display this selection of districts in the map. To retrieve the projection system of these objects, we eliminate the conditional part of the query and we add the SRID function that applies to geometries. By executing this query, we see that all our objects, or let's say a large part of our objects, are in 32740 which is actually the UTM 40 South system and we see with the DISTINCT keyword that the set of objects have this same reference system.

Notes

Summary

5m 35s



The conversion functions, which also apply to the GEOMETRY attribute, that can transform a geometry into a text format to make it readable or on the contrary, in binary format to store it so that it takes up little space. These two functions are very useful to export / import datasets from one database to another and convert them in another storage format. The syntax is always fairly simple, the SELECT keyword, the AsText function with the geometry in brackets, FROM and the name of the table.

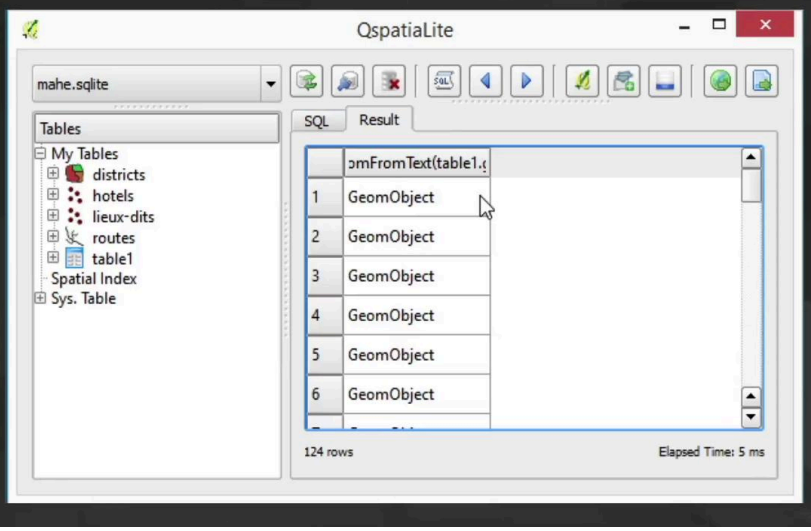
Notes

Summary

6m 22s

Conversion

AsText(), AsBinary()



To illustrate this operation, we take the QGIS SpatiaLite interface and we simply write this request of transformation of the hotel geometries into text and we give an alias to this column, we call it gg, so it comes from the hotel table and we see that our column gg contains a series of multipoints with the coordinates x y. We can create a table from these results, that we will call table 1, and we can add this table in the SpatiaLite database. We can consult the objects of this table. We see that we find our multipoints with a primary key that was added automatically. The equivalent function to transform the geometry into a binary number allows to observe in fact that what we create is a geometric object that we will be able to use for maps. Let's suppose now that we have obtained, imported this table that contains text geometries.

Notes

Summary

7m 00s

Characteristics of geometry

GeometryType(), NPoints(), NRings(), Envelope()

- Information on the type of geometry and other properties: number of points, number of rings, bounding box
- Syntax

```
SELECT GeometryType(geometry) FROM name_table
```

[…]

```
SELECT Envelope(geometry) FROM name_table
```


We can use a GeomFromText function which allows to transform these text geometries into binary geometries, so geometric objects that we will then be able to use to display them in a map. A series of functions that now allows to recover the characteristics of spatial geometry, starting with the type of geometry, the number of points or the number of rings it can contain and its envelope so the bounding box, the rectangle that encompasses the geometry. The syntax is always of SELECT type, the function GeometryType on Envelope and the GEOMETRY keyword in brackets, FROM and the name of the table. So we illustrate this type of queries with the function that allows to extract the type of geometry of a layer, in this case, we will focus on the district layer.

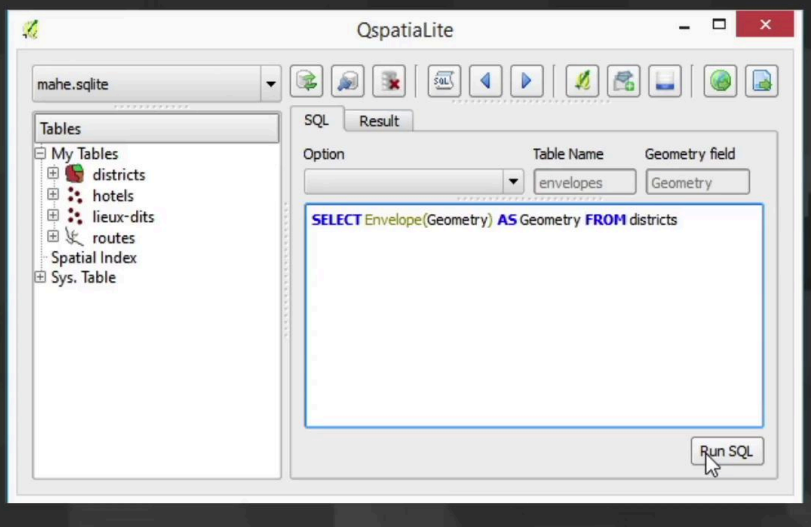
Notes

Summary

8m 46s

Characteristics of geometry

GeometryType(), NPoints(), NRings(), Envelope()

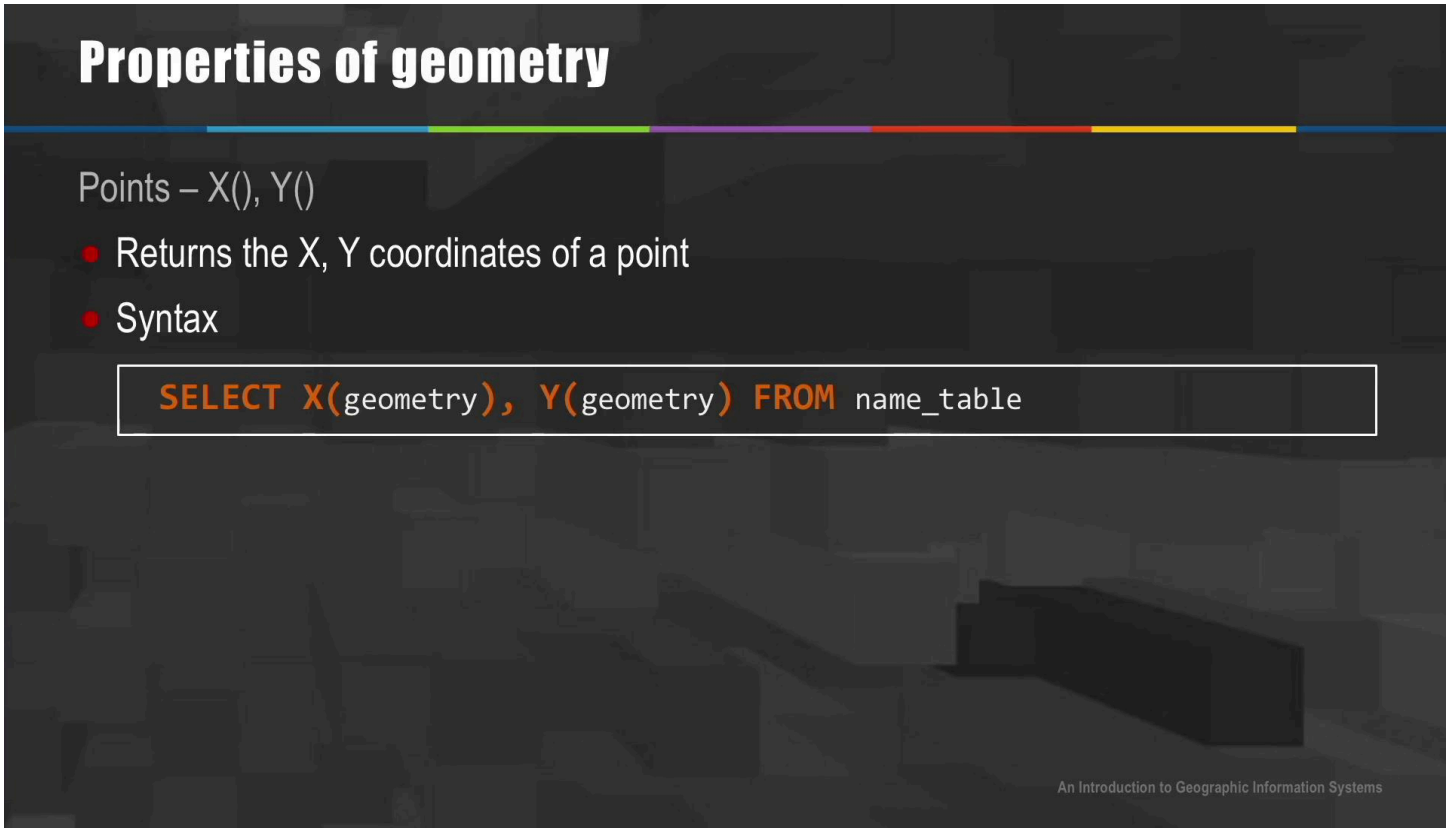


and we see that the SQL syntax here is quite permissive since we can simply mention the GEOMETRY attribute, the name of the table, removing any ambiguity. With the DISTINCT keyword, we can check that the entire district layer is composed of multipolygons. If we now replace this extraction of geometry type by the Envelope function, we see that we create a series of geometric objects and to add these geometric objects to the map, we must make it a GEOMETRY column, define the table name in which we wish to store the thing, define the GEOMETRY field as containing the geometry and execute the query.

Notes

Summary

9m 45s

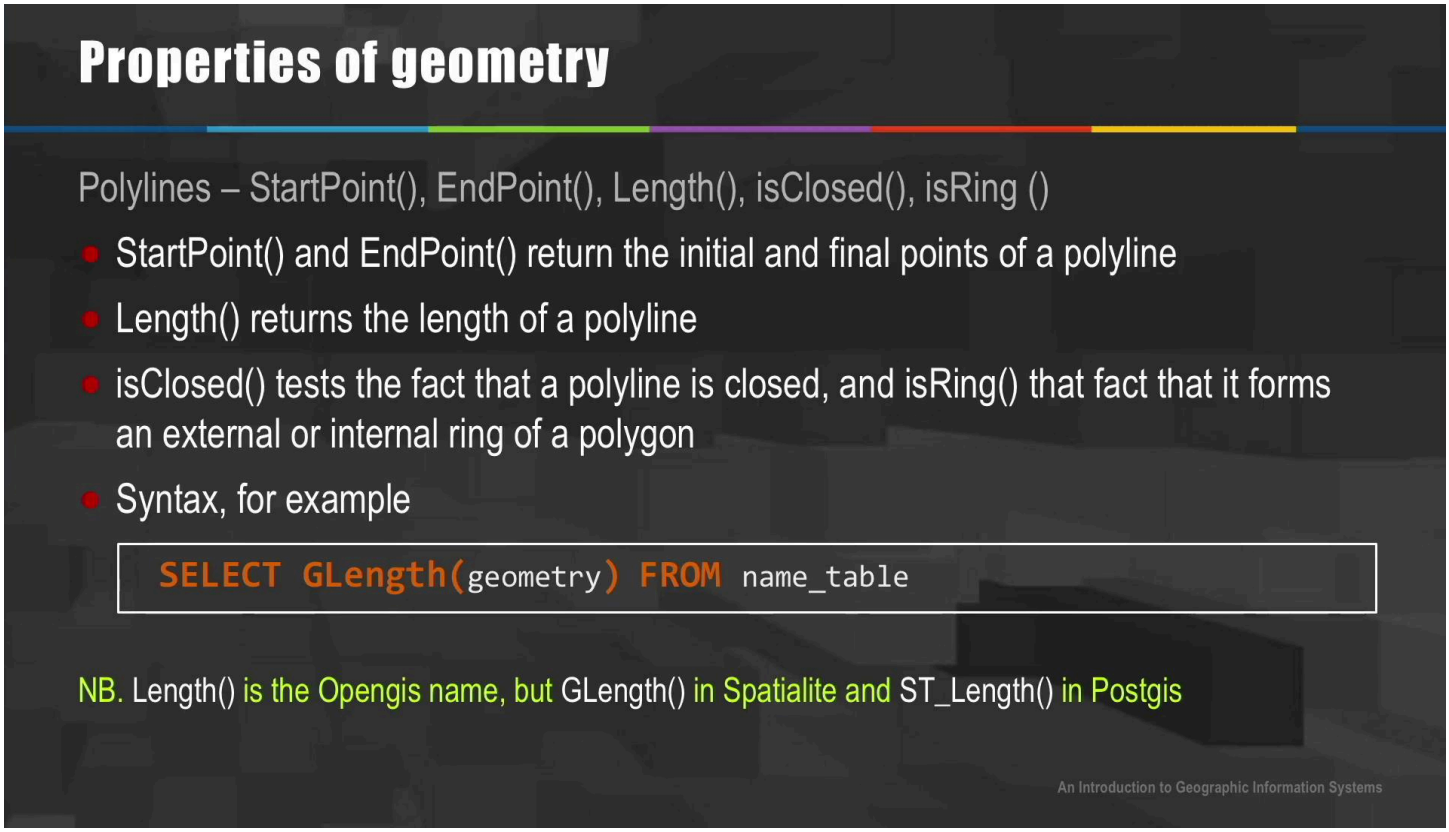


So this query creates a layer called On Envelope and we find the envelopes that include the different districts of the island of Mahé here. Among the many functions that return specific properties of spatial geometries, we will start simply by those returning the coordinates xy of a point with a very simple syntax of SELECT X type in brackets, the geometry, comma Y, the geometry if we want to extract the two coordinates, FROM the name of the table.

Notes

Summary

10m 34s



This syntax is illustrated by selecting the x and y coordinates of the Seychelles' hotels. So here again, simply the GEOMETRY keyword, the name of the table, removing any ambiguity. We see that the result of the Greek colony is in scientific notation and so we can use a CastTo function towards an integer to transform this scientific notation into an integer number to make it more readable. Second series of functions that concern the polylines more with the extraction of the initial and the final points of the polyline, StartPoint, EndPoint, the length, Length and a test to check if the polyline is open or closed and possibly a test to check if it is a ring or not, so if it is part of a polygon. The syntax is of SELECT type, the function in brackets the geometry, FROM the name of the table as usual with a particularity, the fact that the length function that returns the length of the polyline is preceded by a capital G in SpatiaLite for the simple reason that the LENGTH keyword is a reserved word of SpatiaLite and in QGIS, as I said at the beginning, this function is written ST for spatial type, underlined and then the LENGTH keyword.

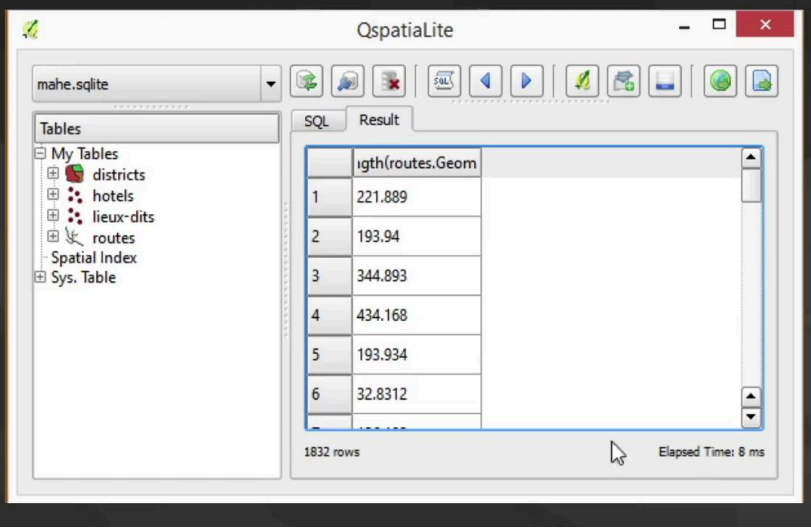
Notes

Summary

11m 14s

Properties of geometry

Polylignes – StartPoint(), EndPoint(), Length(), isClosed(), isRing ()

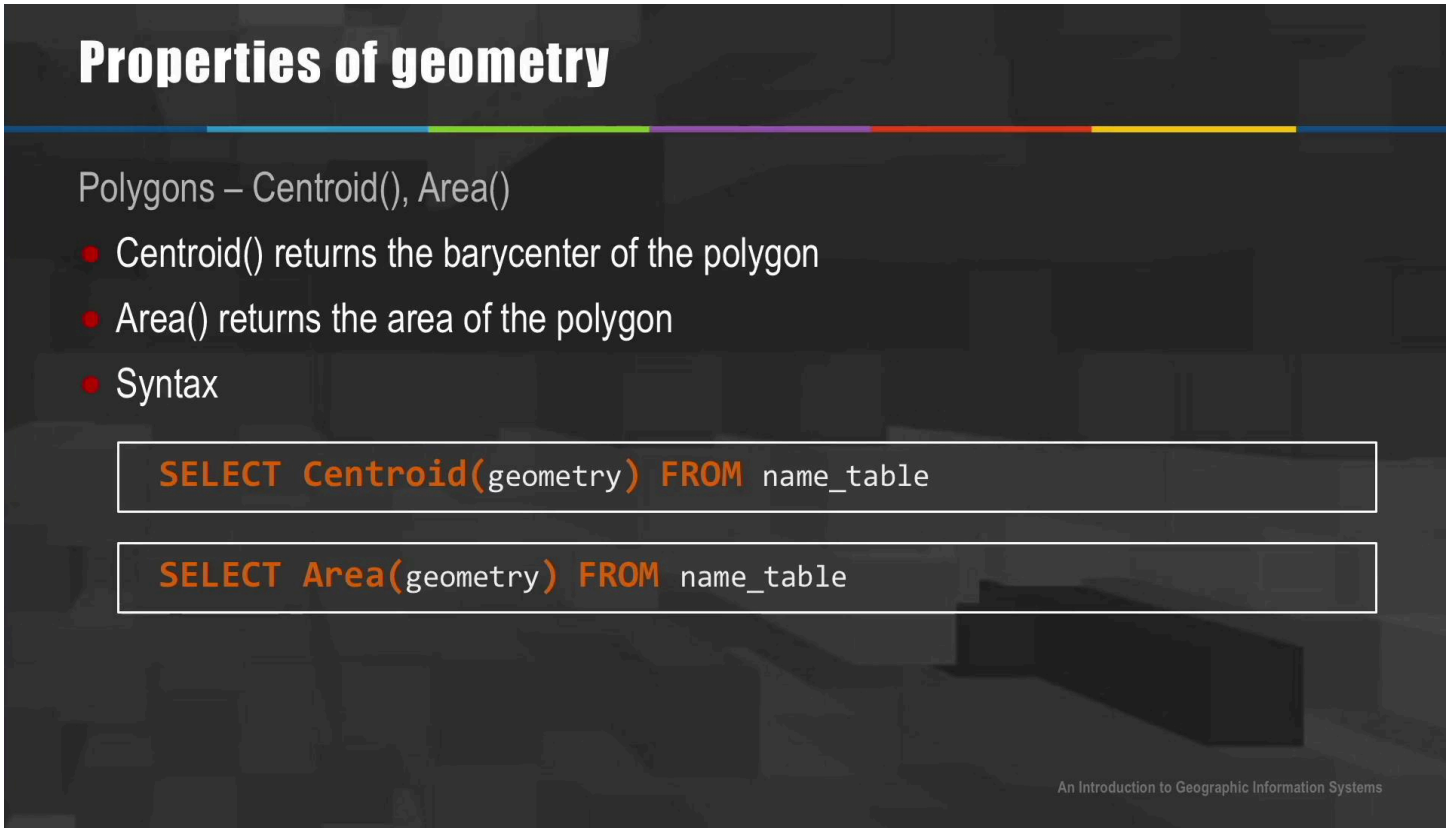


As an example of this type of query, we will extract the length from the road sections of the Seychelles' road map. And we see that by executing this request, we obtain the length in meters of these road sections.

Notes

Summary

12m 53s



Two functions which now concern more specifically the polygons, the Centroid function which returns the polygon barycenter, the center of gravity of the polygon, and the Area function that returns to its surface. The syntax is always the same with the SELECT keyword, the function, the geometry in brackets, the FROM keyword and the name of the table.

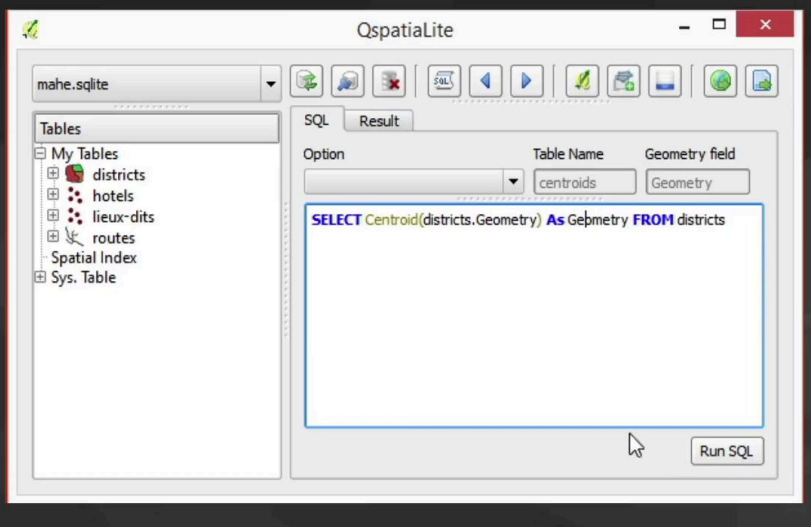
Notes

Summary

13m 14s

Properties of geometry

Polygones – Centroid(), Area()

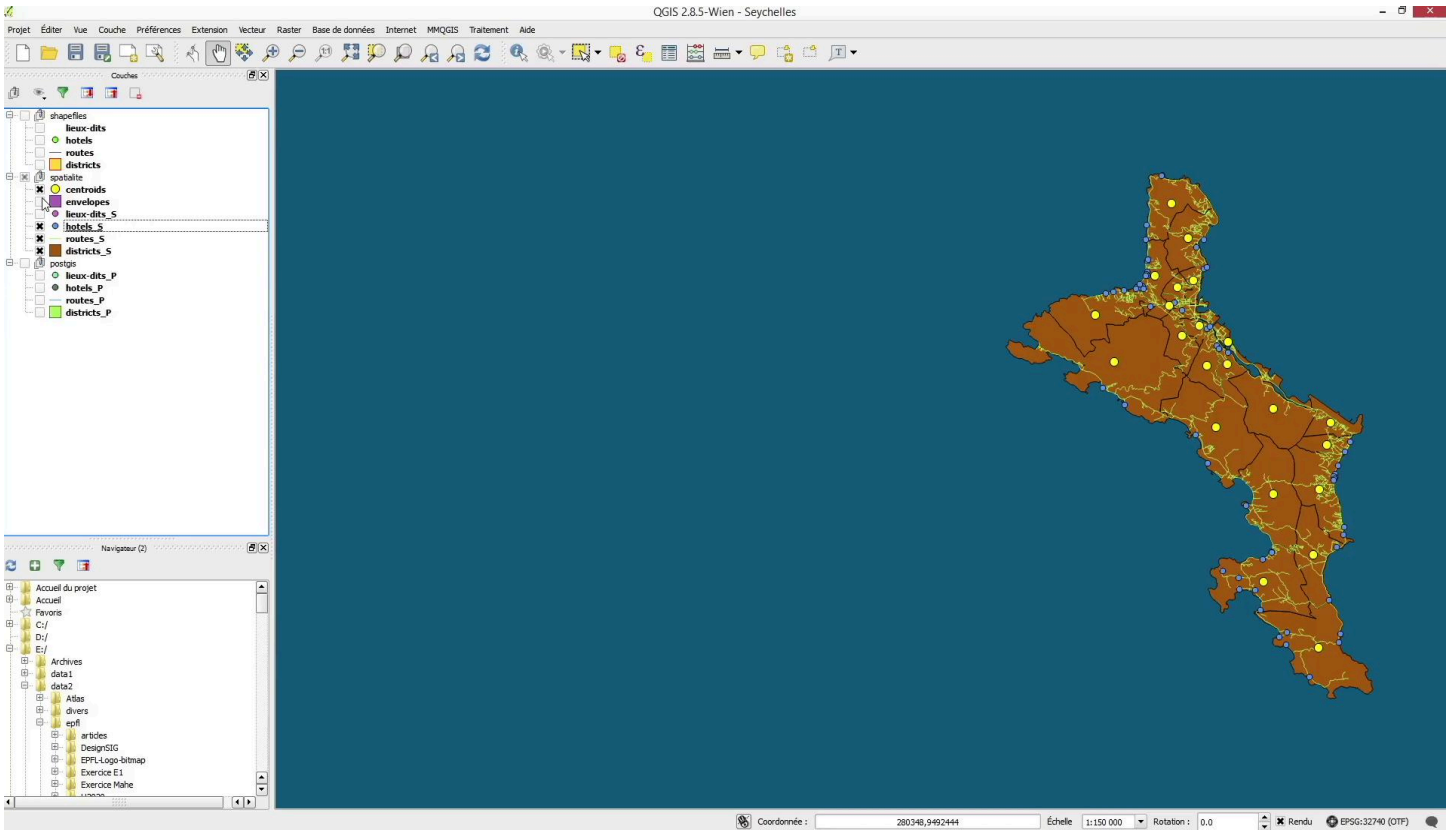


So in our example here, we start by selecting, by extracting the surfaces of Seychelles' districts. That's it. So we have again a notation of scientific type. If we now focus on Centroid, we obrain geometric objects that we may want to represent on the map. So to do this, we must load the result as a layer in QGIS, define the geometry, the geometry field with the GEOMETRY keyword, give a name to the table and make sure that the geometry field is called GEOMETRY.

Notes

Summary

13m 37s



By executing this request, we see that we create a layer called centroid. And if we display these centroid layers, we see that we have yellow dots that appear in the center of the Seychelles' districts.

Notes

Summary

14m 28s