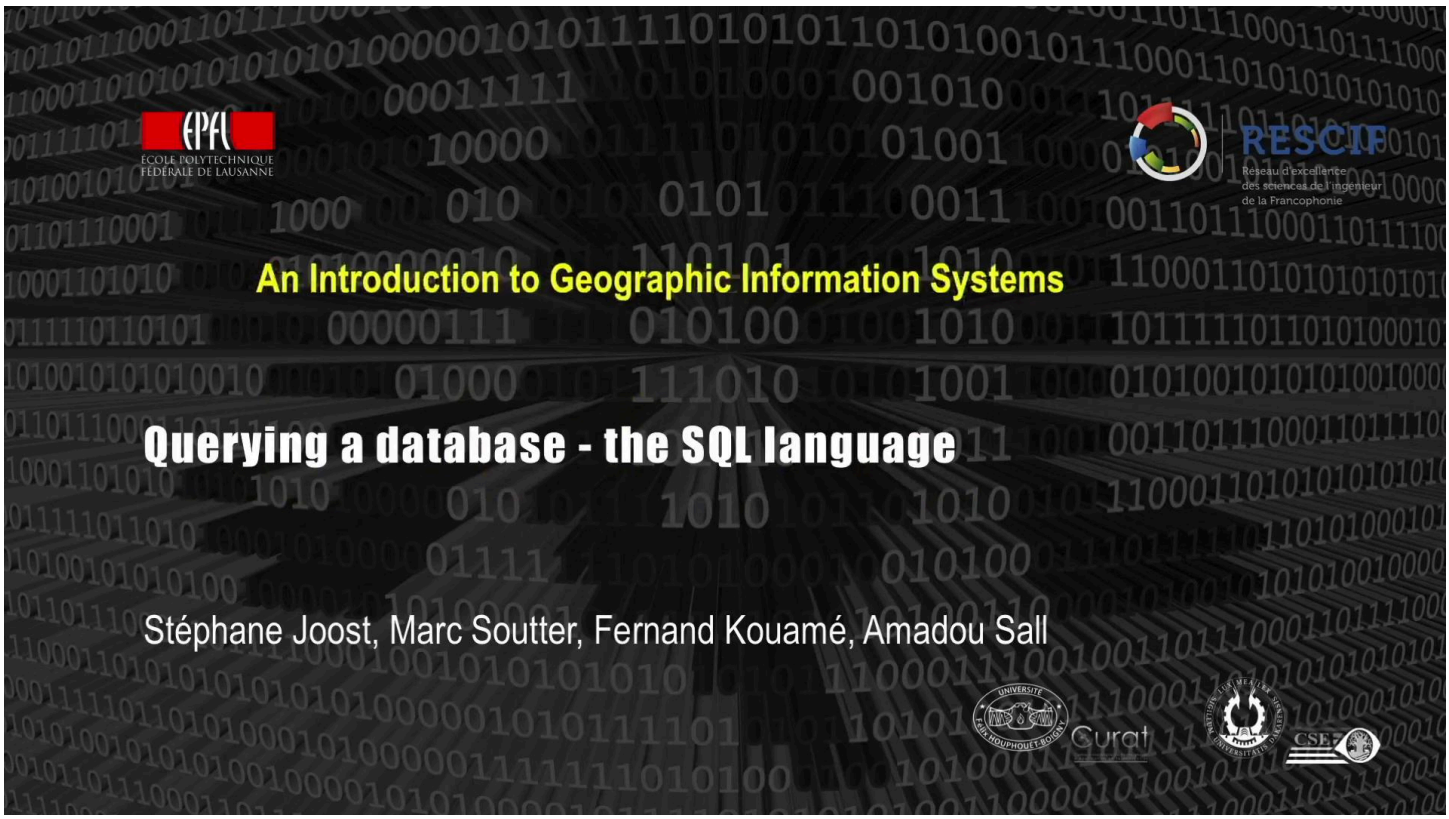
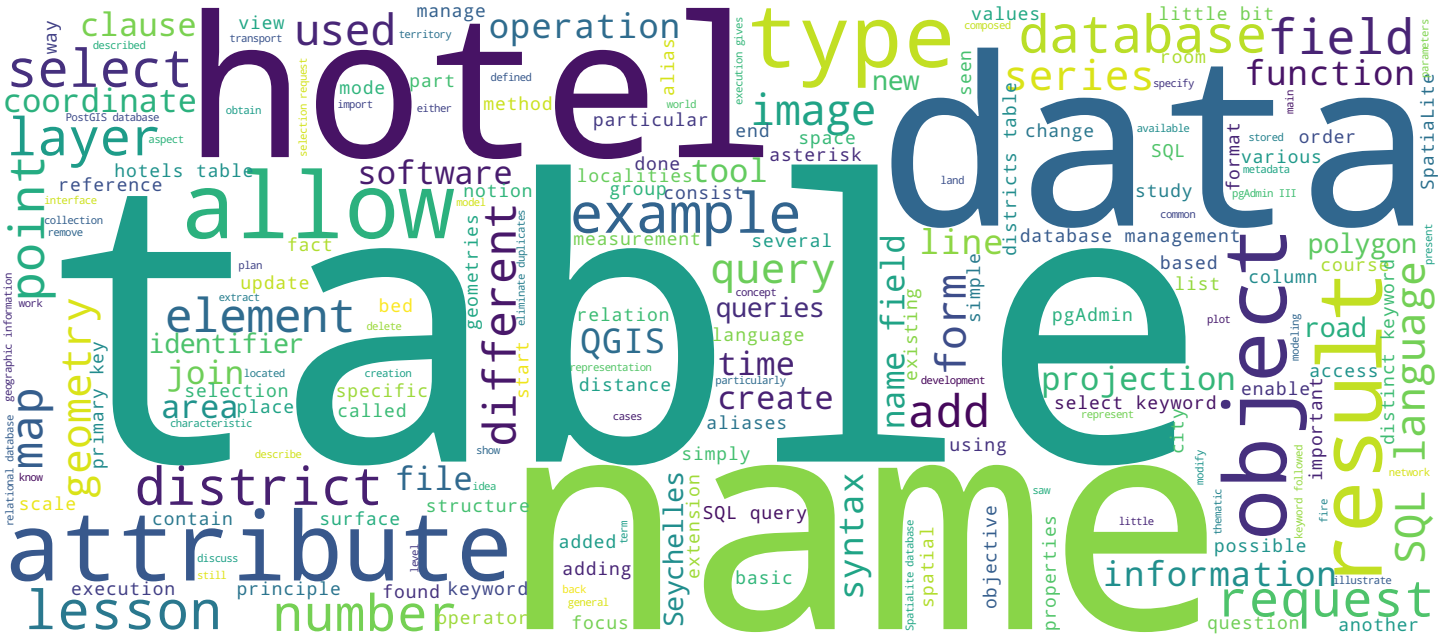
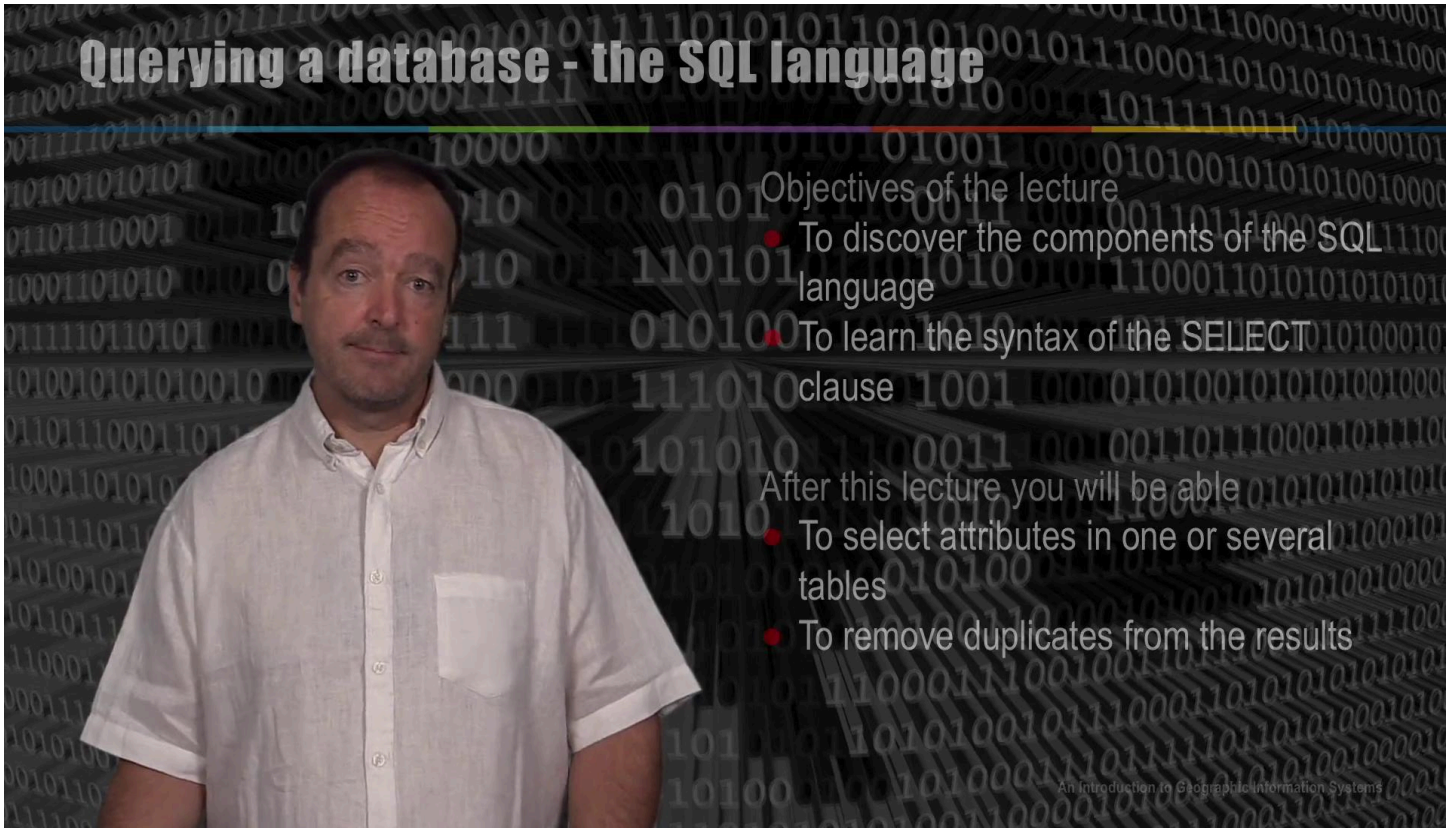




Search MOOC

Video

EPFL

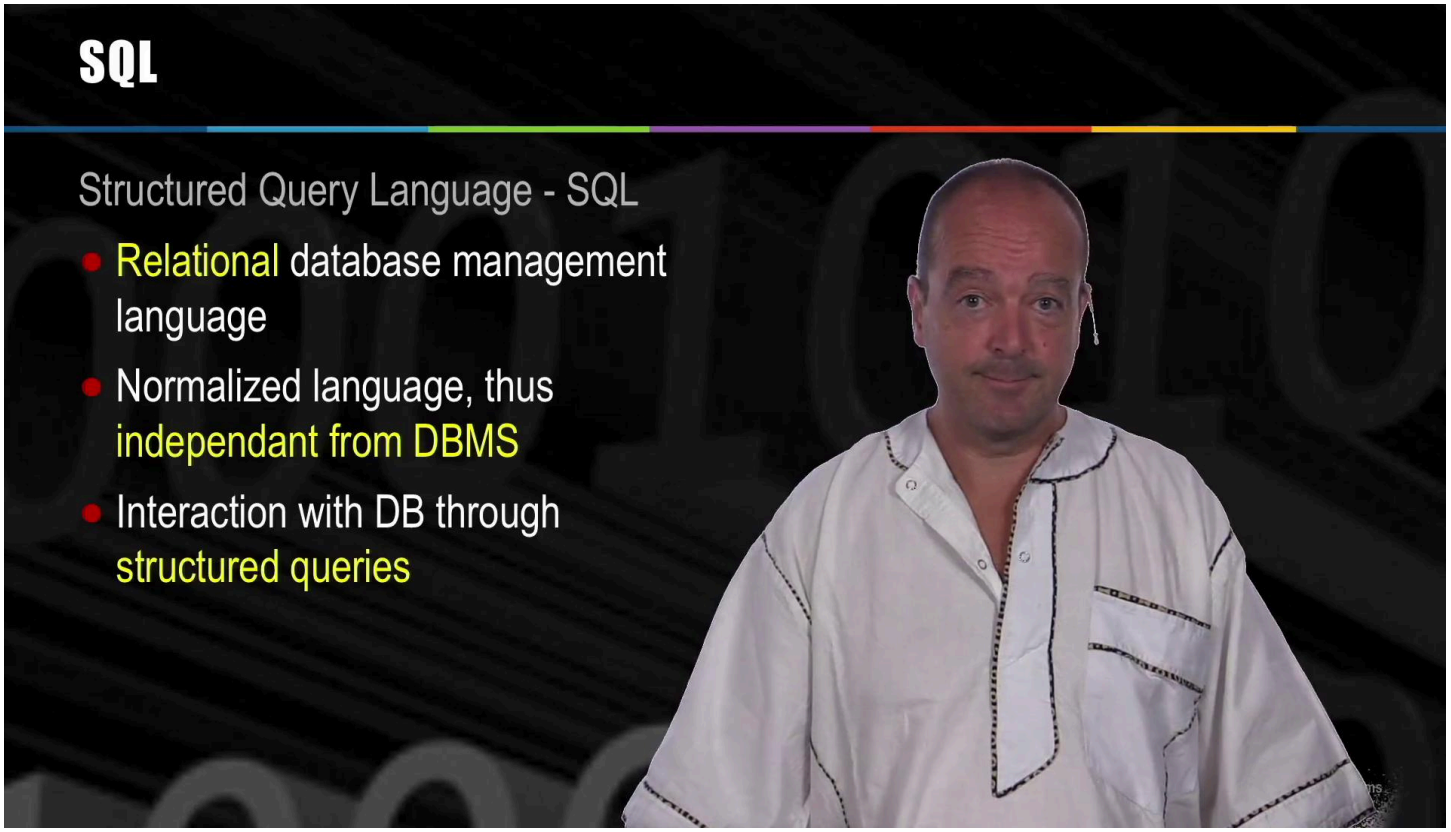


Welcome to this lesson that will focus on querying databases by requests, and the SQL language that is used to write these queries. In this lesson, we will discuss the basics of the SQL language, language that will be the subject of the next 5 or 6 lessons. The objective of this first lesson is to discover the most fundamental principles of the SQL language and in particular the most basic function or clause, which is the "select" clause and its different uses. At the end of this lesson, you should be able to select attributes in a table and... to eliminate duplicates that could be found in this selection.

Notes

Summary

0m 23s

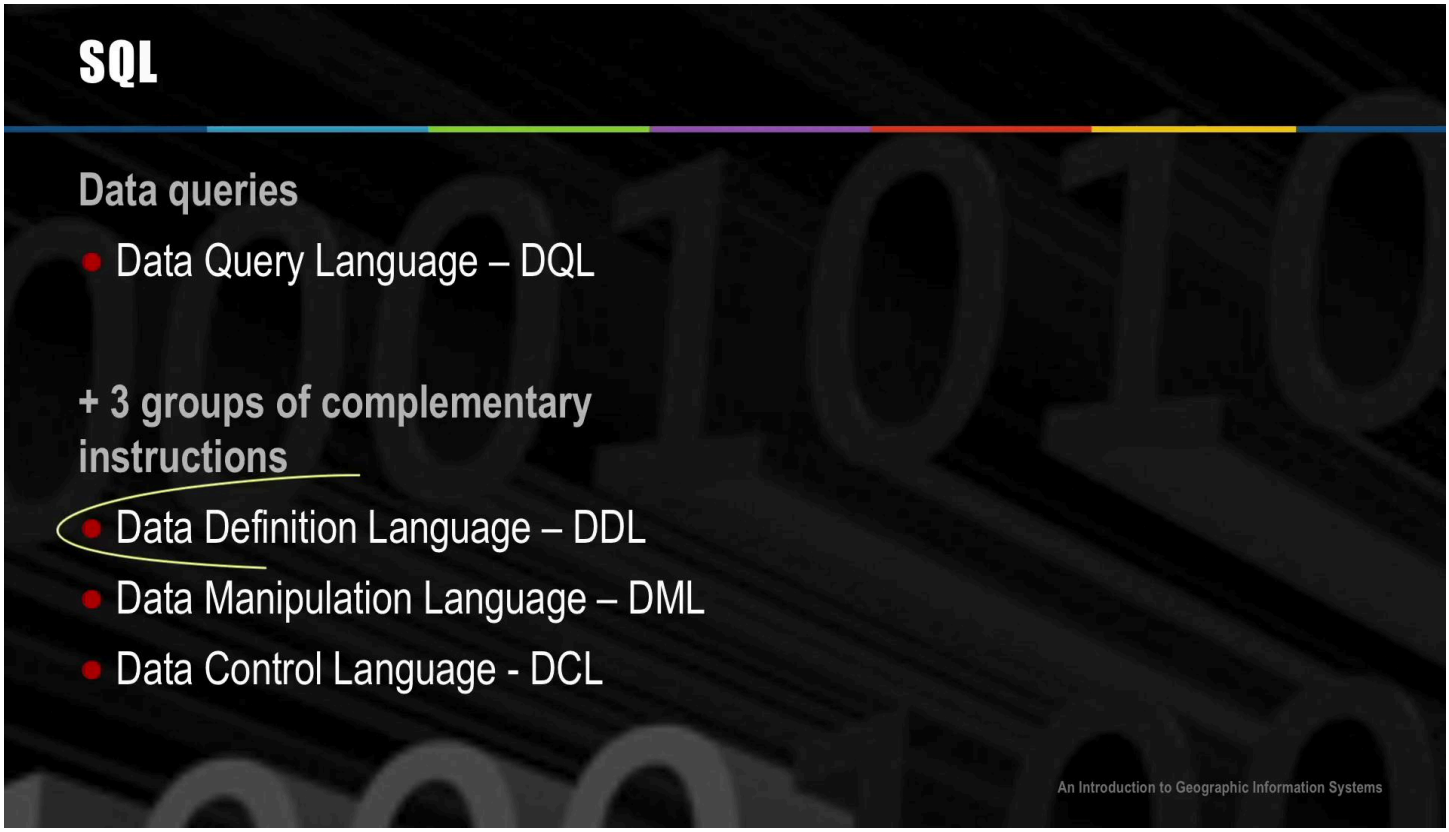


We will see successively the principles of the SQL language then the simple selection of an attribute in a table. The principle of the distinct selection that allows to eliminate duplicates of a query. Then, the selection from 2 tables and finally the use of aliases for the attributes and tables. The SQL or Structured Query Language is a language for relational database management specifically. It is a language that has been standardized so which is in theory independent of the database management system used even if it is true that each database management system has some specialties regarding the syntax used. It is a language that interacts with databases in the form of a structured request as its name suggests, The SQL is composed of 4 groups of additional instructions.

Notes

Summary

1m 11s



First, the data query language which enables to extract data from a database.

Notes

Summary

2m 21s

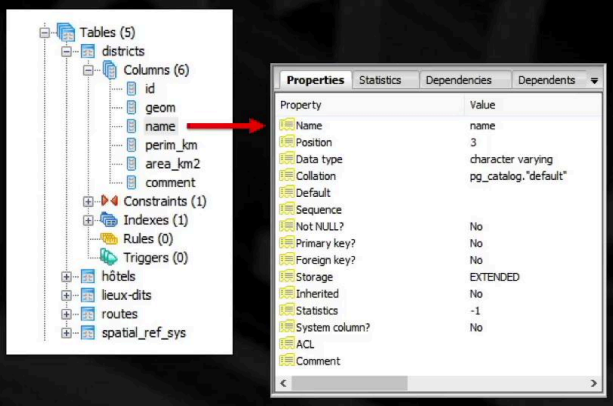
SQL

Data queries

- Data Query Language – DQL

+ 3 groups of complementary instructions

- Data Definition Language – DDL
- Data Manipulation Language – DML
- Data Control Language - DCL



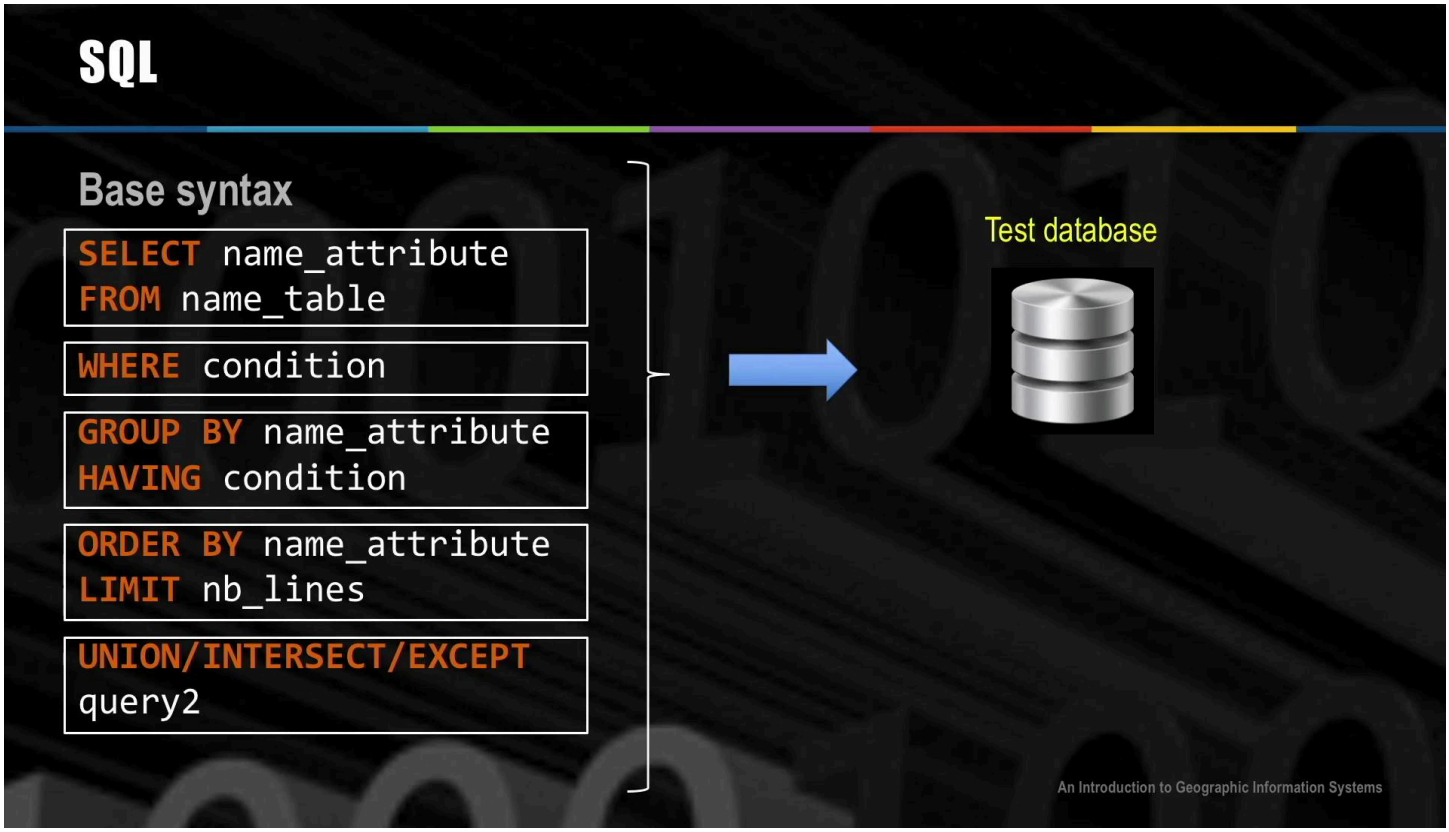
Define or modify the structure of the database

The data definition language which allows to modify or define the structure of a database.

Notes

Summary

2m 31s



The DML manipulation language that allows to insert, to update, to delete data and finally the control language that manages the users' rights and accesses. In the field of data querying, the basic syntax includes selection clauses with the "select" and "from" keywords, the conditional filter clauses with the "where" keyword, aggregation clauses with "group by" or "having" keywords; sorting clauses with "order by" or "limit" keywords and finally merger clauses with the "union", "intersect", "except" keywords. The use of these different keywords in queries will be illustrated later in this lesson and in the following lessons.

Notes

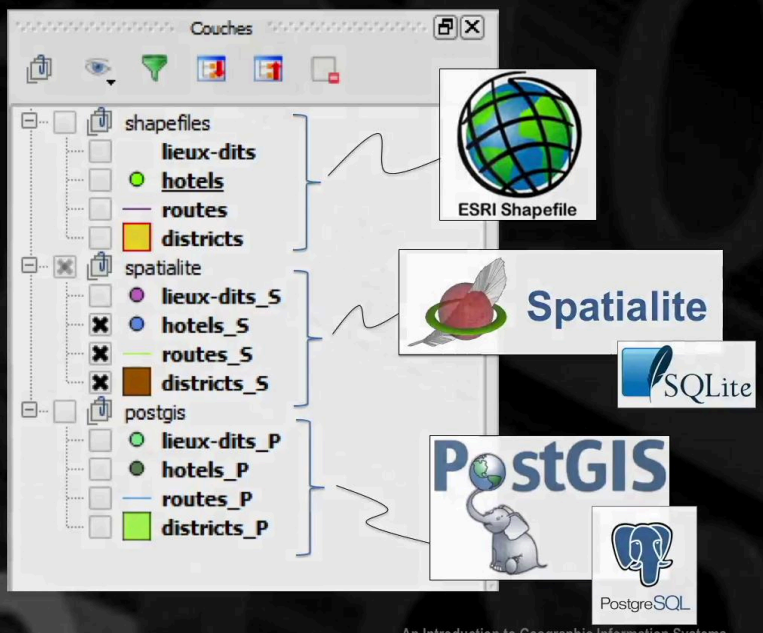
Summary

2m 35s

SQL – Test database

Seychelles, central islands

- 4 data layers
  - Districts (polygons)
  - Hotels (points)
  - Named places (points)
  - Roads (lines)
- In 3 formats
  - Shapefile
  - Spatialite
  - Postgis



On the basis of data that were grouped together in a series of test databases, data on the islands close to the Seychelles which actually comprise 4 layers of data: the districts in the form of polygons, the hotels, the localities that are points and the roads that are lines. These data are stored in 3 different formats which will illustrate three types of approaches, different management of requests. The ESRI Shapefile format, a SpatiaLite database and a PostGIS database.

Notes

Summary

3m 26s

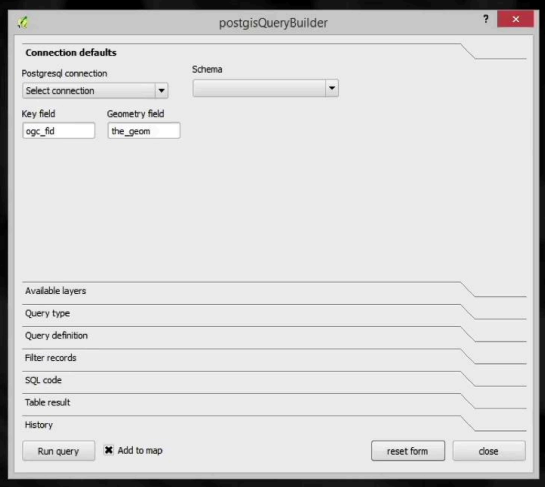
SQL – Query tools

QGIS layers in general
- Query builder (layer properties)
- Queries on attribute tables
- Spatial query extension

Spatialite and PostGis
- Database management extension

Specific for Spatialite
- Sqlite Studio (outside QGIS)
- Qspatialite extension

Specific for PostGis
- pgAdmin III (outside QGIS)
- postGisQueryBuilder extension
- Processing / PostGis SQL query editor extensions



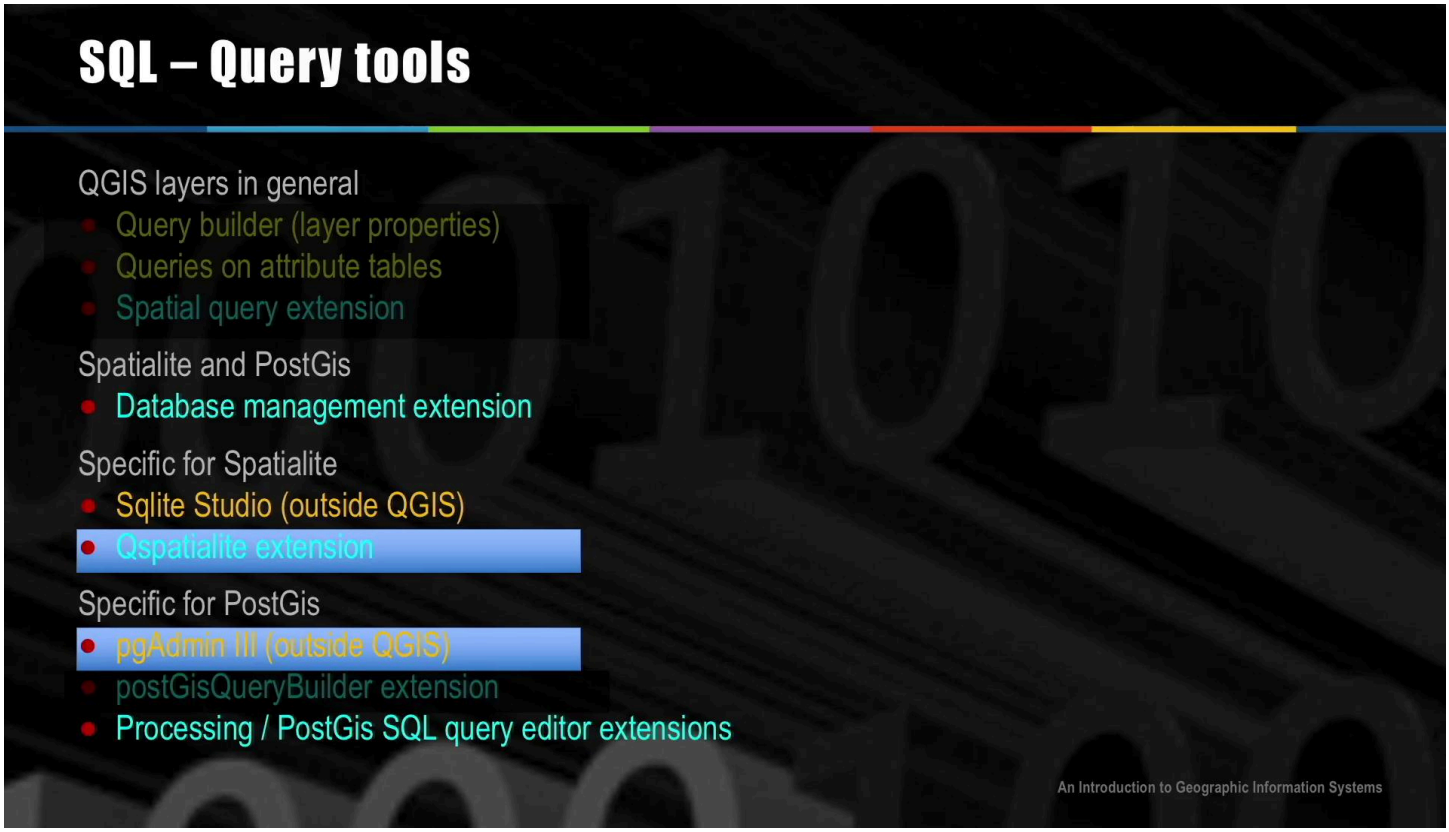
An Introduction to Geographic Information Systems

We can see here that these different layers of data have a certain number of attributes. For the districts, the geometry, the identifier and the name, for hotels again the geometry, the identifier the name and a series of complementary attributes (number of rooms, number of beds, status) and the district in which the hotels are located. For the localities, geometry, identifier and name as well as for roads with the places where they are, the type of road and the type of surface. We see that there is a small key around the identifier which shows that in all these cases, the identifier acts as the primary key. There are many query tools and we will focus on those that can be used in connection with the QGIS software. There is first of all in the software itself a series of 3 query tools which... apply generally to all types of layers. But these 3 solutions do not implement explicitly the SQL language unfortunately, so in this case it is not very interesting. We have a database management extension that allows to process Spatialite and PostGIS databases and probably others, and then a series of solutions specific to Spatialite and specific to PostGIS. Some of them use softwares like the SQLite Studio pgAdmin III.

Notes

Summary

3m 58s



In the PostGIS solutions PostGisQueryBuilder is also unattractive because it does not explicitly implement the SQL language. For the purpose of this course, we will work with the SpatiaLite extension for QGIS which allows to write and execute SQL queries on a Spatialite database in QGIS. And we will also use pgAdmin III which is a software which is part of PostGreSQL and which has the advantage of proposing a graphical query construction tool.

Notes

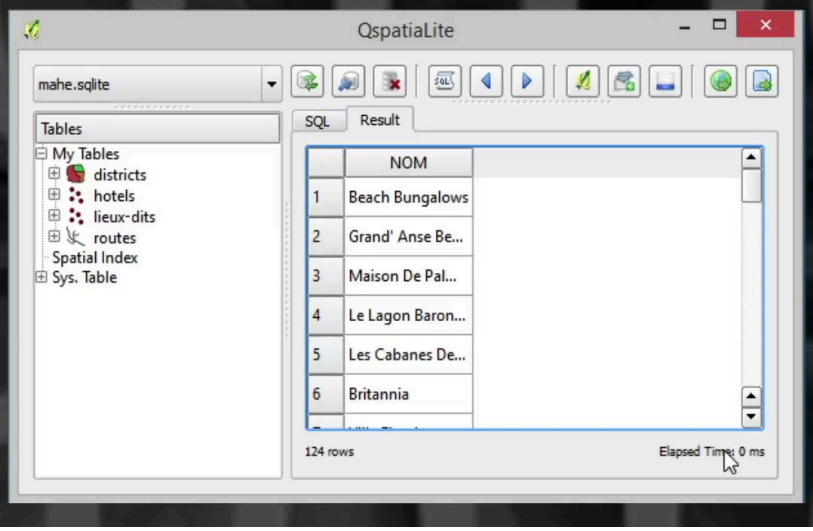
Summary

5m 33s

Simple selection

Select an attribute

```
SELECT name_attribute
FROM name_table
```



The selection request includes the "select" keywords followed by the name of the searched attribute and the "from" keyword which allows to specify in which table is found the attribute in question. In QGIS, with the Spatialite extension, we can see that we can write an SQL query. In the window provided, we will write a query to select the list of hotels by their names. So the attribute we are looking for is the "name" attribute and the table, the "hotels" table. The execution of the query gives the list of hotel names in the Seychelles.

Notes

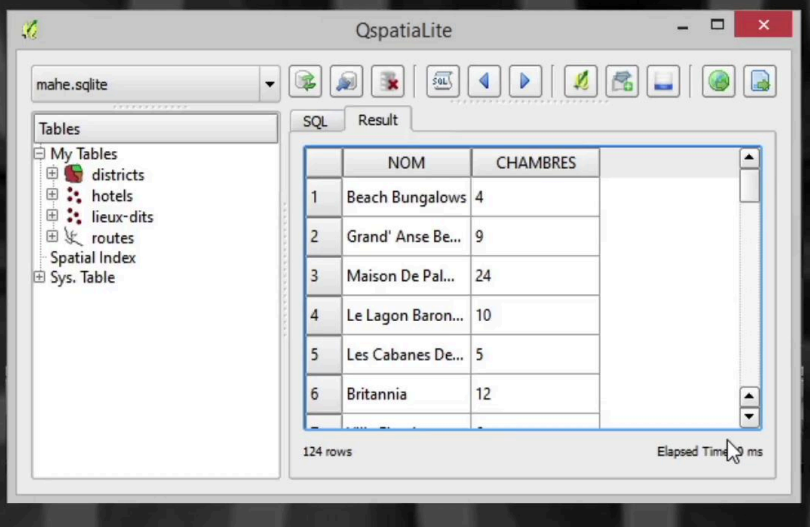
Summary

6m 20s

Simple selection

Select two attributes

```
SELECT name_attribute_1, name_attribute_2
FROM name_table
```

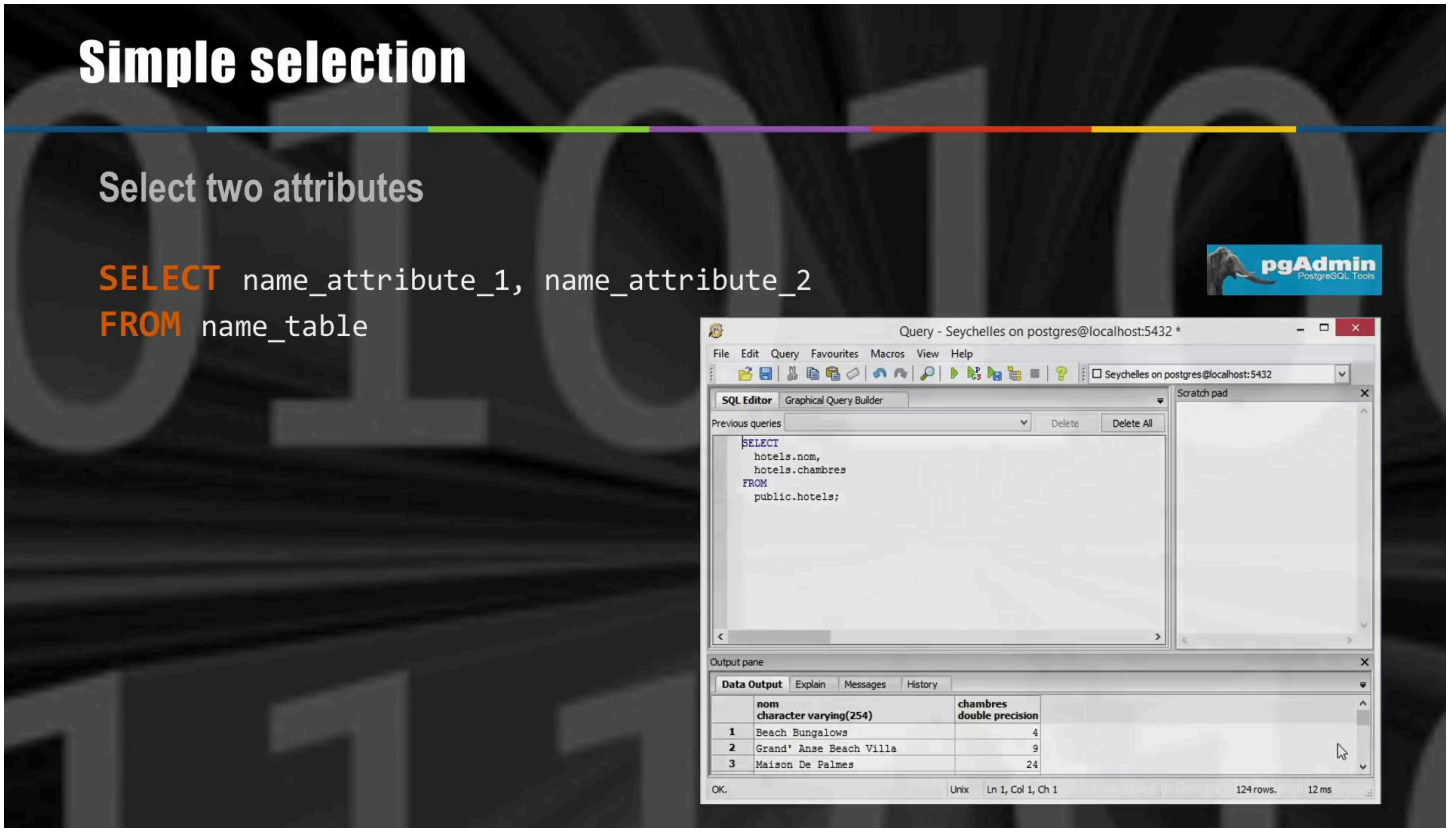


The same operation can be performed graphically in pgAdmin III by adding the hotel table in the construction field of the query by selecting the "name" field. We see that this results in the creation of an SQL query which syntax is a little bit different. This syntax allows to remove any ambiguity in the case where we have several tables with attributes of the same names. On the other hand, it is not obligatory when the doubt is not allowed. The selection request of 2 attributes includes the "select" keyword followed by the name of the attributes separated by a comma then the "from" keyword and the name of the table. In our example, we add the number of rooms as the desired attribute and the execution of the request shows the number of rooms per...

Notes

Summary

7m 01s



depending on the name of the hotel. In pgAdmin, it is sufficient to select the additional field in the graphic constructor which updates the SQL query and its execution gives the expected result.

Notes

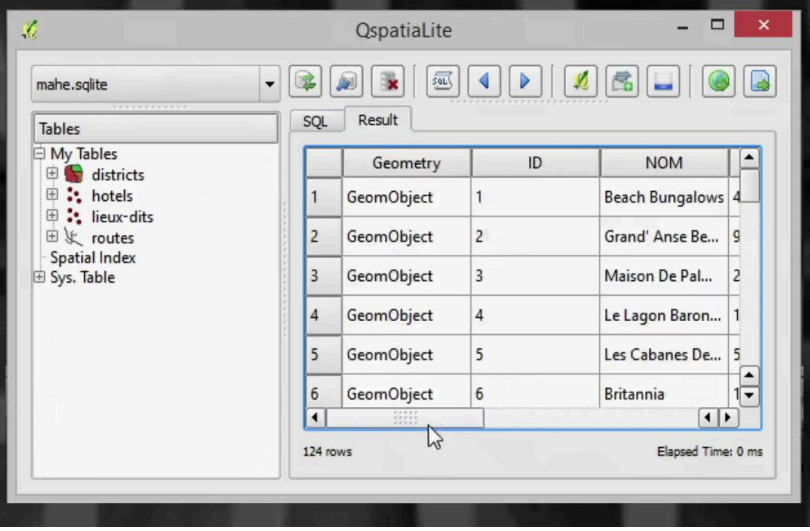
Summary

7m 51s

Simple selection

Select all attributes

```
SELECT *
FROM name_table
```



The syntax that allows to select the set of attributes of a table is constituted by the "select" keyword followed by an asterisk then the "from" keyword and the name of the table. We see in this example, that if we replace the names of the fields searched by an asterisk, we indeed get a table that contains all the fields of the layer.

Notes

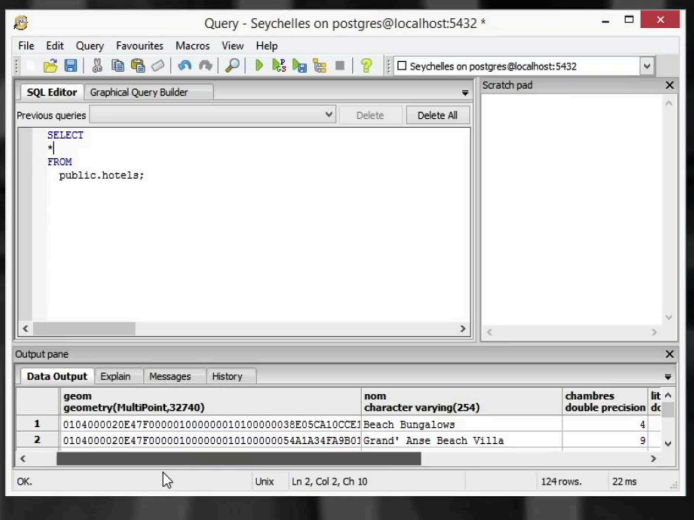
Summary

8m 08s

Simple selection

Select all attributes

```
SELECT *
FROM name_table
```



Similarly, in the graphic constructor of pgAdmin... so we see that there is no asterisk so we can select the set of fields manually or in the SQL window simply replace the fields with an asterisk and the result is still the same.

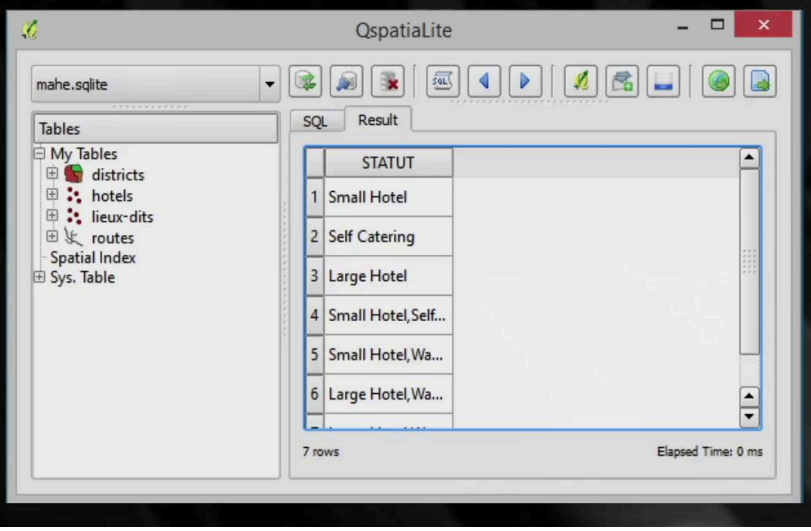
Notes

Summary

8m 32s

Distinct selection

```
SELECT DISTINCT name_attribute
FROM name_table
```



We can note that there are 124 hotels selected. The "distinct" keyword placed after the "select" keyword helps eliminate duplicates which could be found in the result of a request. We see that if a request is made on the hotel status, we get a table in which we find many times the same value since many hotels have the same status - small hotel, big hotel etc. The "distinct" keyword added to this request allows to to filter this result and to only get the list of possible values of the "status" attribute.

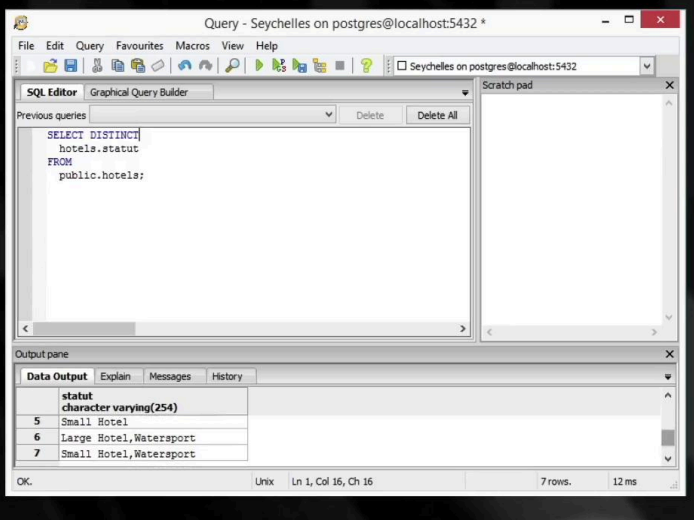
Notes

Summary

8m 57s

Distinct selection

```
SELECT DISTINCT name_attribute
FROM name_table
```



The operation is the same in pgAdmin. And if we keep only the "status" field, we update the request and we add the "distinct" keyword and we see that by executing the request, the result obtained is the same.

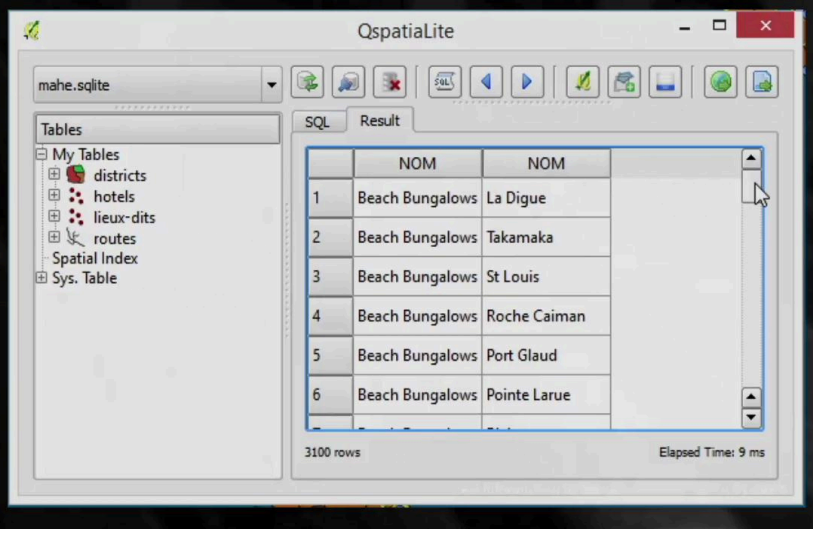
Notes

Summary

9m 59s

Selection from two tables

```
SELECT name_table1.name_attribute1, name_table2.name_attribute2
FROM name_table1, name_table2
```



One of the main interests of the SQL language is to associate multiple tables within a single query to obtain a result which combines data coming from 2 different tables. This more explicit syntax implies describing an attribute by the name of the table from which it comes followed by the name of the attribute, the two elements being separated by a point. In our example here, we will select the "name" field of the "hotels" table so described by hotels.name and in the "districts" table, also the "name" field which shows clearly the value of having this more specific syntax. So these 2 fields taken from the 2 "hotels" and "districts" tables. We see that in the result we find the systematic association of names of the 2 fields and we see that for 124 hotels and 25 districts, that gives us 3,100 results.

Notes

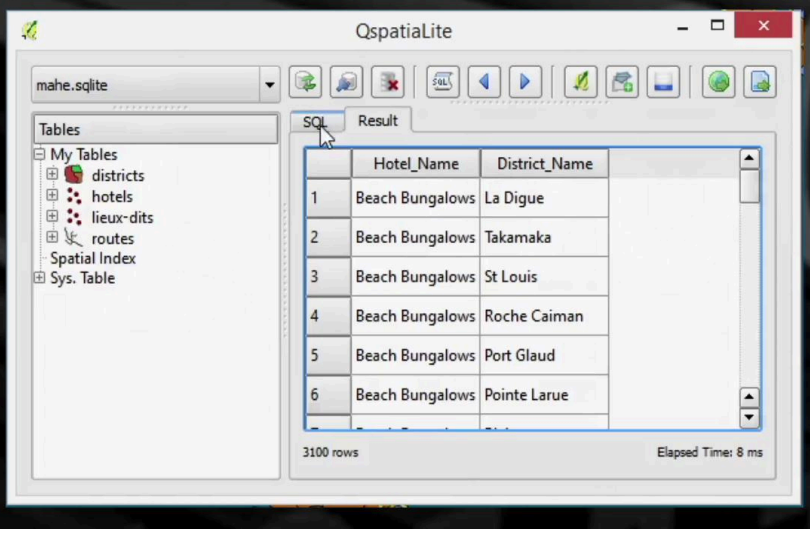
Summary

10m 37s

Alias

Alias of the column name

```
SELECT name_attribute AS name_alias
FROM name_table
```



In pgAdmin, things happen in the same way. We select the "name" field in the "hotels" table. The "districts" table is added, we select the "name" field and in the SQL query tab, we see that the request was written in the right way and the execution gives the same result with 3,100 possibilities. It is sometimes useful to be able to replace the original attribute name by a more readable or comprehensible alias, an operation which is carried out using the "as" keyword, attribute name "as" alias name. In our previous example, we replace here the "name" field of the hotel by Hotel_Name and the "name" field of the district by Districts_Name which also eliminates any ambiguity in the result table where we see that the headers of the columns bear the names that allow us to know what we are dealing with.

Notes

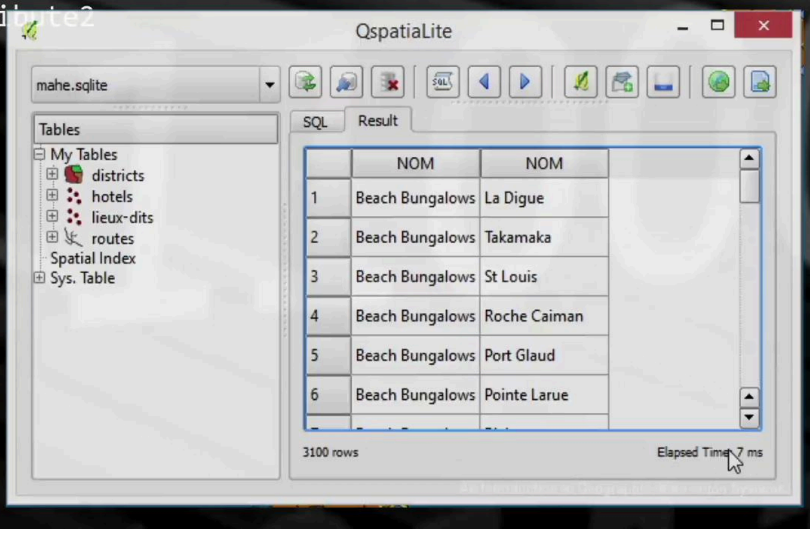
Summary

11m 47s

Alias

Alias of the table name

```
SELECT name_alias1.name_attribute1,
       name_alias2.name_attribute2
FROM name_table1 name_alias1,
     name_table2 name_alias2
```



Same thing in pgAdmin, The alias is specified in the the criteria, the characteristics of the results of the query and we see that by switching to SQL mode, the request has... the syntax of the query and the result are what is expected. Second possibility of using the concept of alias, this time to change the name of the tables. Which is often interesting to simplify the syntax of the queries a little bit. In the case of tables, the aliases do not use the "as" keyword but are simply created by adding a name to the table name with a space between the 2. In our example here... we delete the aliases from the columns, the attributes and we create aliases for tables, H for the "hotels" table And D for the "districts" table, which enables to replace the definition of the attributes sought the names of the tables by their aliases H for "hotels" and D for "districts". And we see that the execution of this request always gives the same result.

Notes

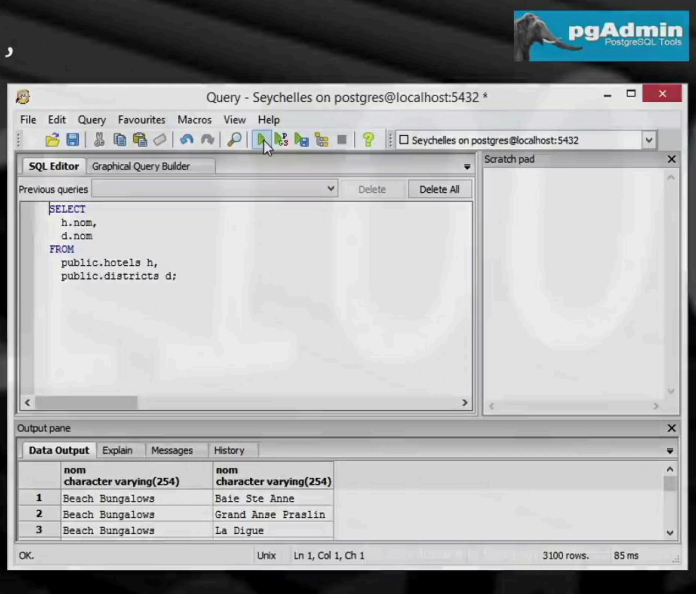
Summary

13m 06s

Alias

Alias of the table name

```
SELECT name_alias1.name_attribute1,
       name_alias2.name_attribute2
FROM name_table1 name_alias1,
     name_table2 name_alias2
```



Same operation in pgAdmin, where we remove the alias from the attributes. Right click on the table to create an alias of the table, H for the "hotels" table, D for the "districts" table as before and when we go back to the SQL editor we see that the syntax was adapted correctly and the execution gives the right result.

Notes

Summary

14m 33s