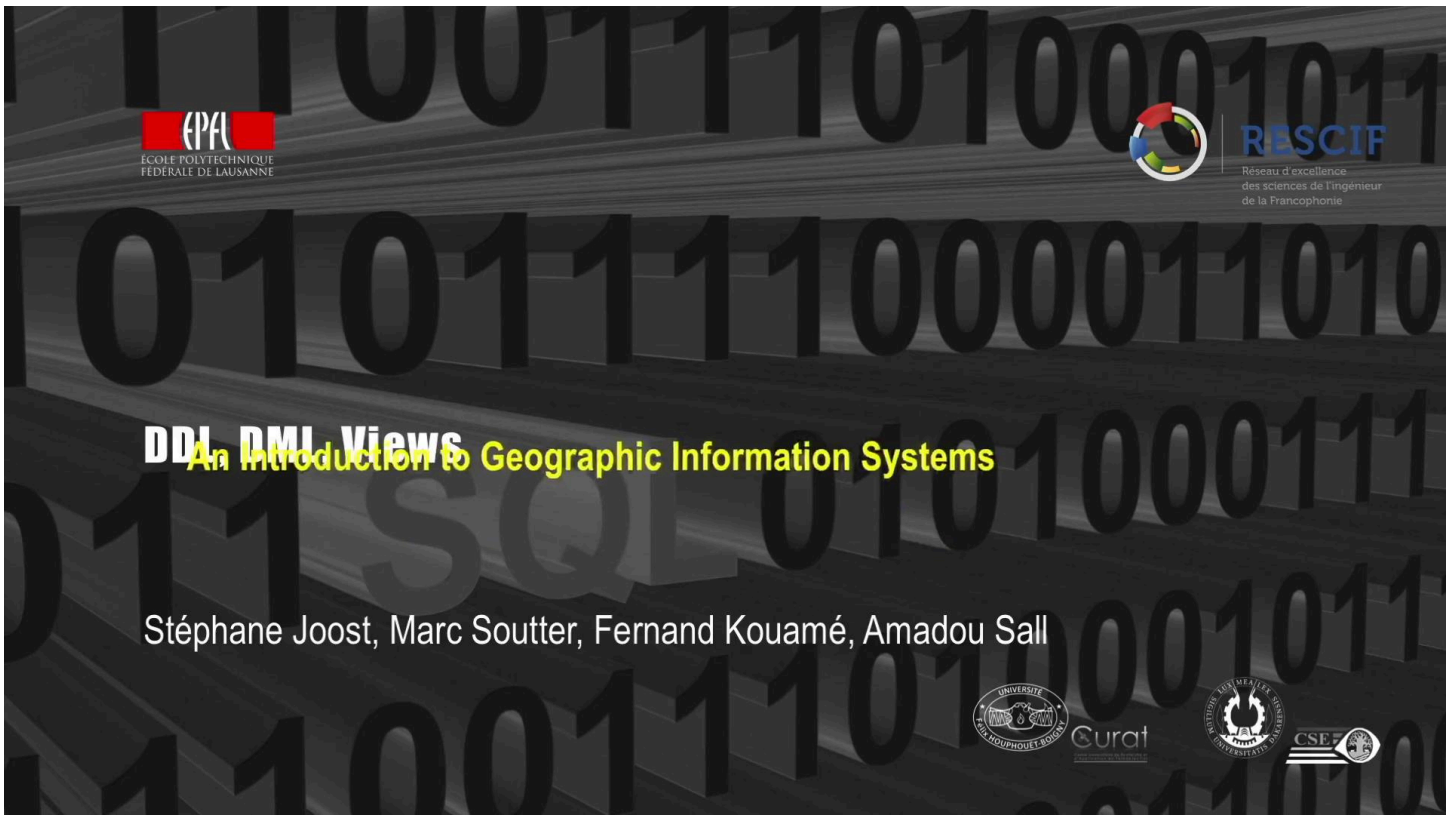
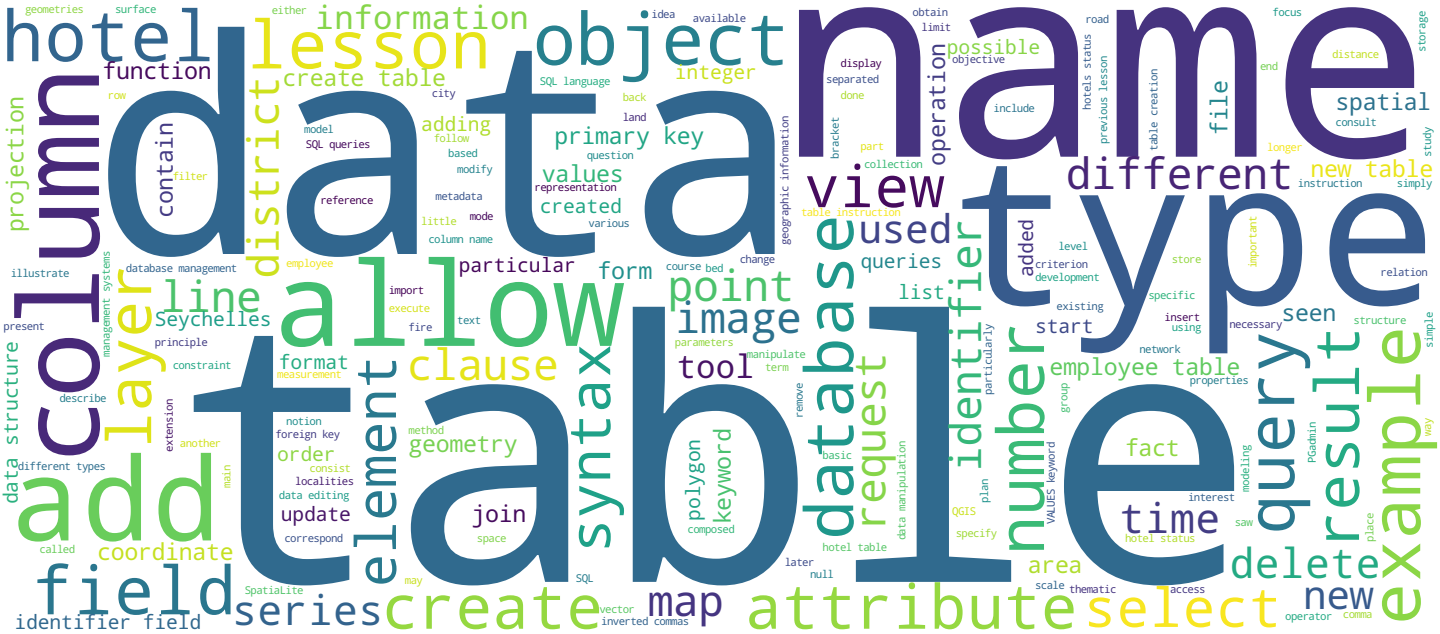
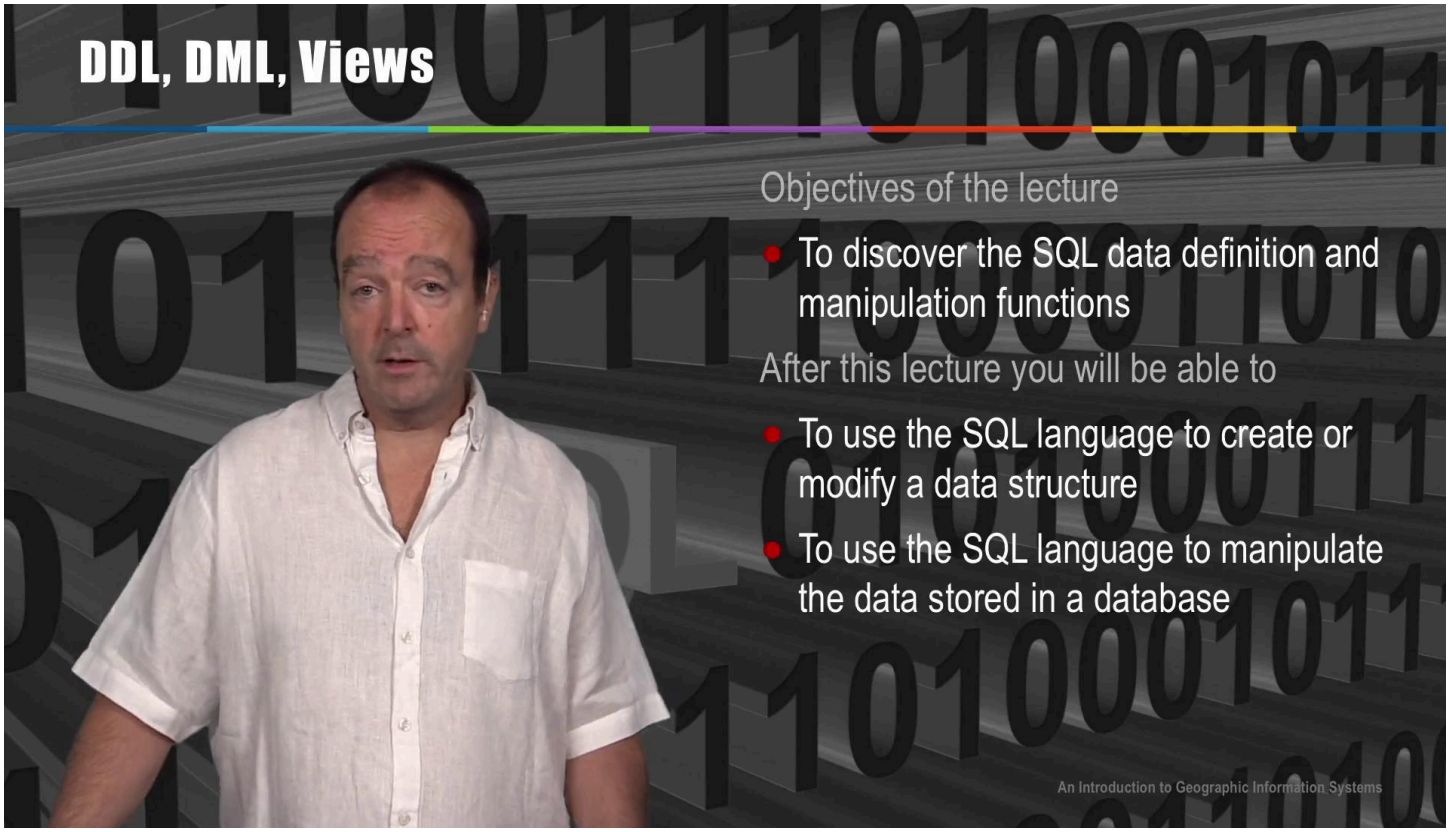




Search MOOC

Video

EPFL

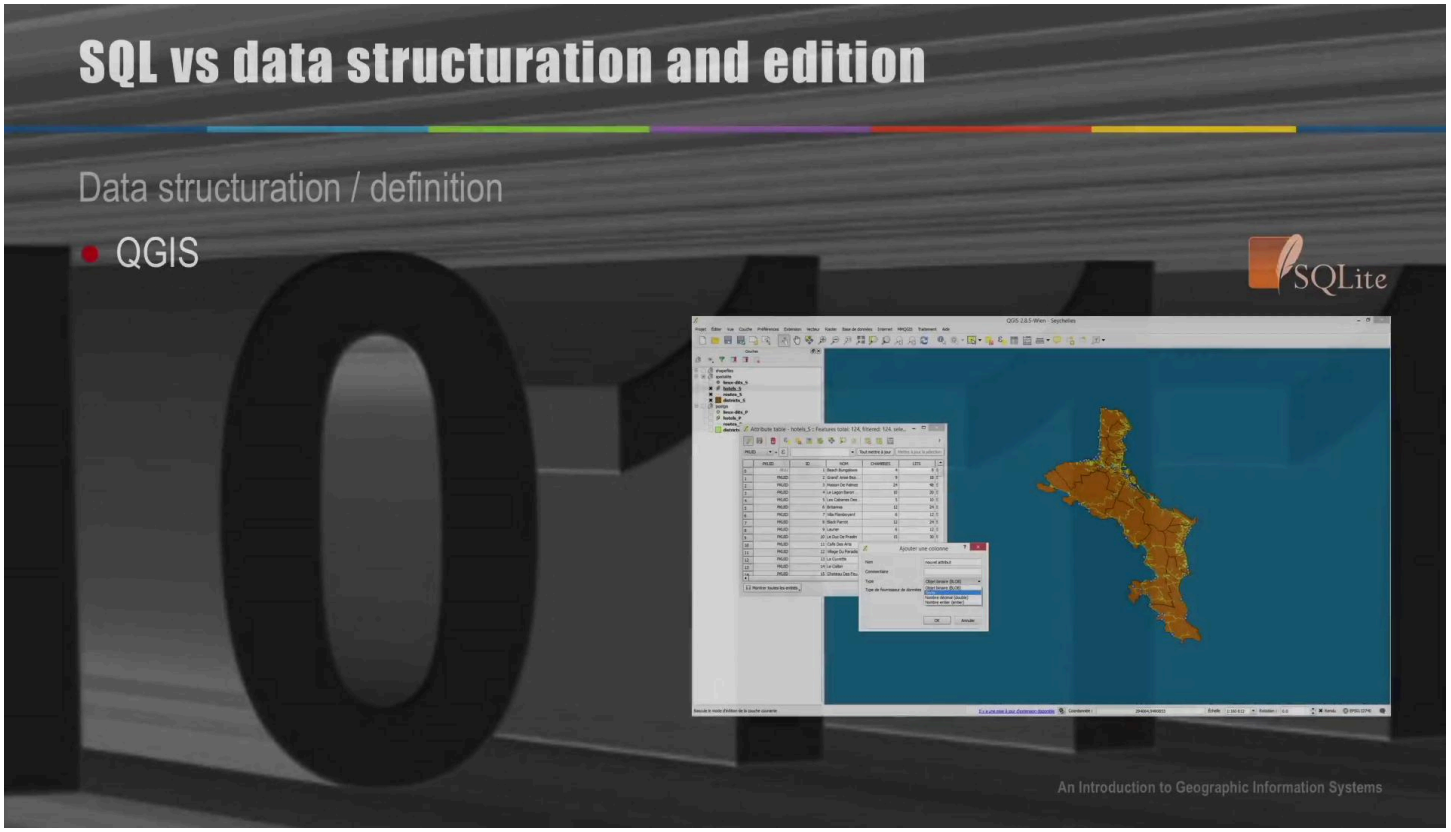


Welcome to this lesson that will focus on the particular elements of the SQL language, the data definition language, DDL, and the data manipulation language, DML. The data structuring tools that allow to create tables with well-defined columns as well as editing tools for attribute data that we saw in the previous lessons, are all elements that ultimately do nothing else than implementing queries of DML or DDL type. We are going to examine these queries during this lesson. The objectives of this lesson are to discover the functions of these data definition and manipulation languages so that we are then able to use SQL queries to create and manipulate data structures and to use SQL queries to manipulate the actual data which are contained in the databases.

Notes

Summary

0m 23s



In this lesson we will start by talking a little bit again about the classic data editing and structuring tools which we have seen until now before addressing more specifically the question of data definition language then that of the data modification language and we will end with a particular aspect which we have not spoken about yet: the views, which are another component of the databases.

Notes

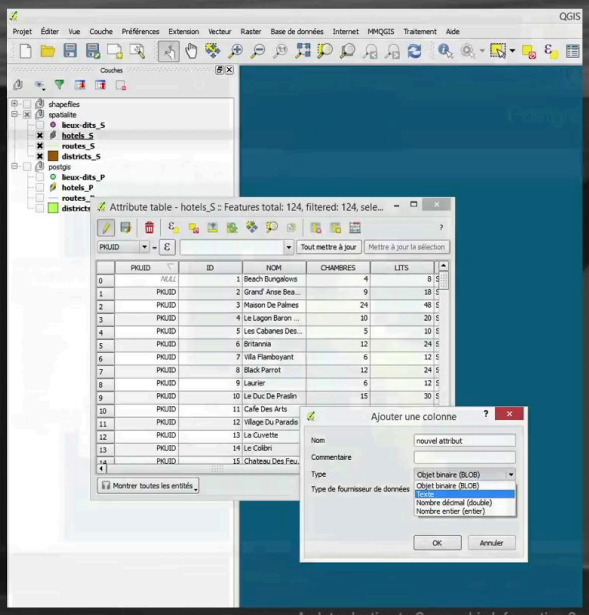
Summary

1m 32s

SQL vs data structuration and edition

Data structuration / definition

- QGIS

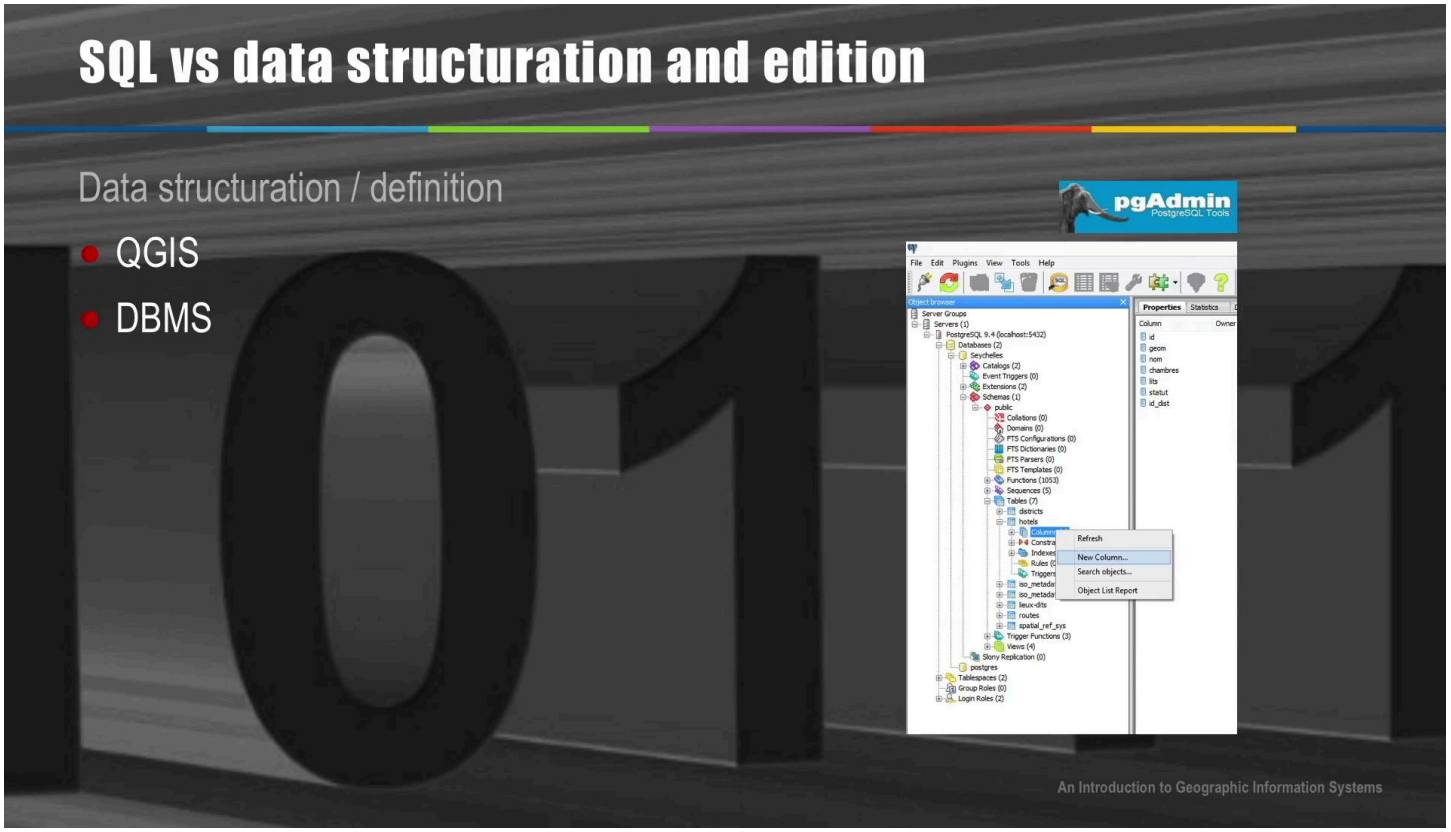


If we go back to the idea of data structuration and definition we remember that the QGIS software proposes elements, tools that allow to create tables directly, to modify these tables by creating fields, columns, of a defined type in the case of a SpatiaLite database as well as in the case of a PostGIS Postgres database.

Notes

Summary

2m 06s

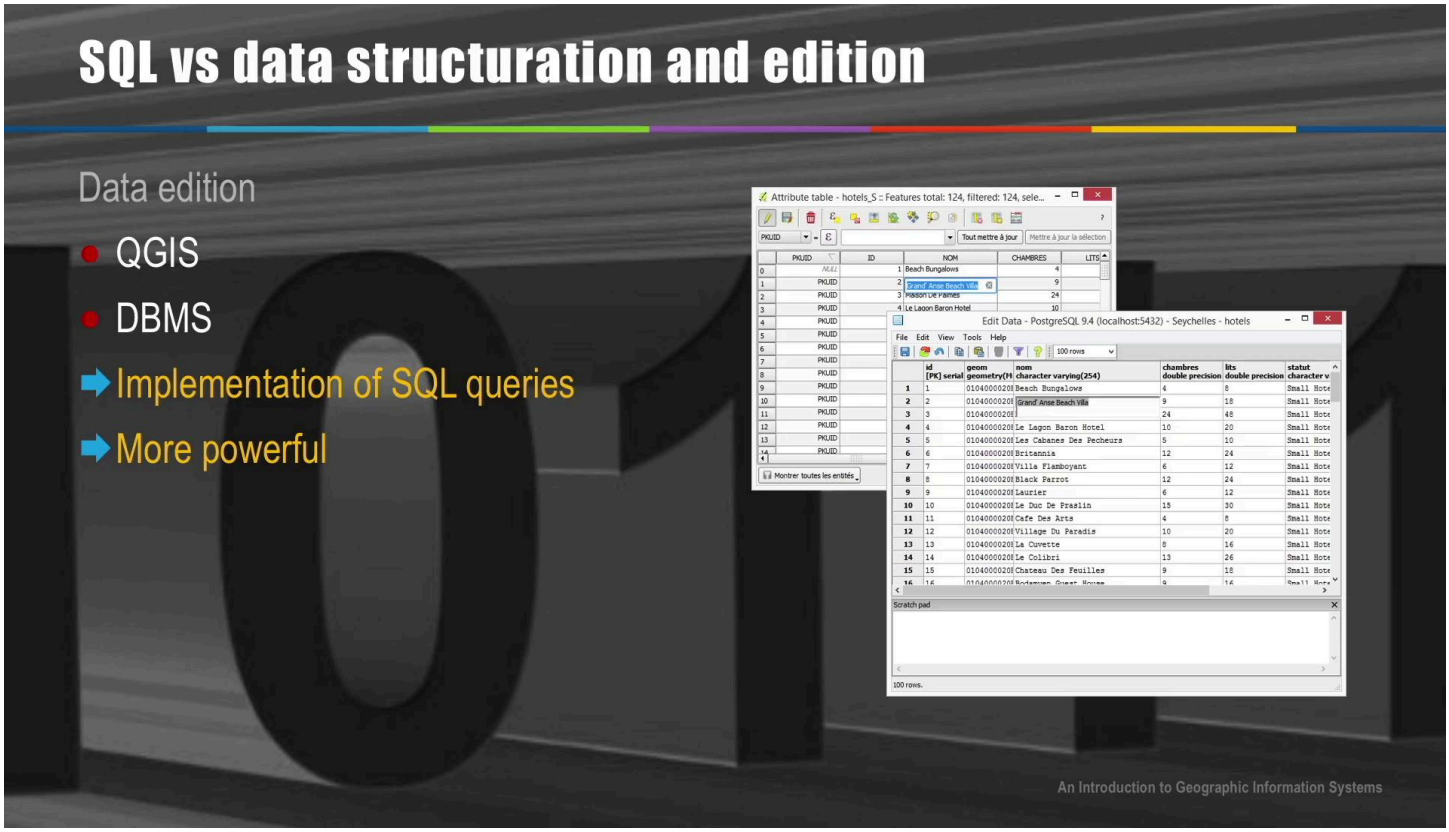


We find the same kind of functionality with the database management systems, just as much for SpatiaLite as for PGadmin, so database management interfaces that allow to manipulate the data structure.

Notes

Summary

2m 33s



Finally, we also discussed the CASE tools which allow to fabricate and manage data structures and eventually transform them into a real physical database. All these various tools we have seen on various occasions in the previous lessons of the course always simply implement SQL queries. It is still simply a casing whose engine consists of SQL queries. In the attribute data editing field we also saw that GIS softwares offer editing tools as well as the database management systems. In this case too the engine of this data editing is composed of SQL queries as we will see later in the lesson.

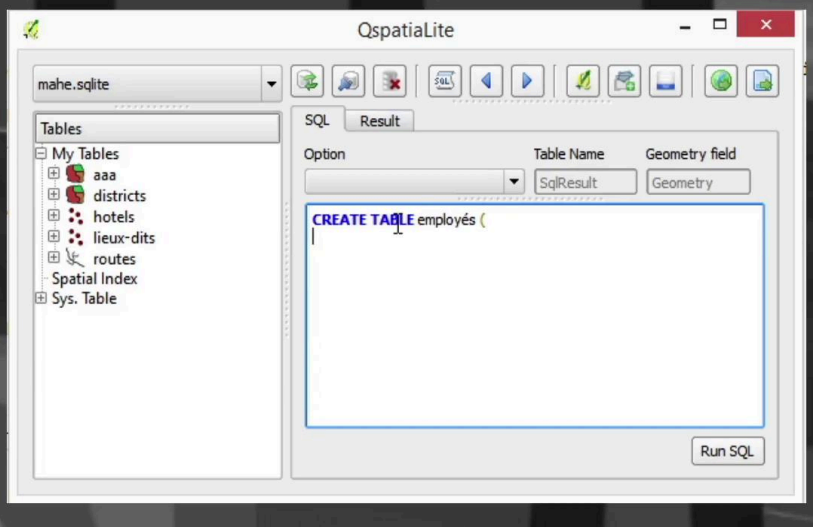
Notes

Summary

2m 50s

Data definition

CREATE TABLE



The advantage of SQL in the data editing field is that it allows to develop approaches that are a little more elaborate than the editing of one attribute at a time. The first and most basic thing we want to do in the data definition field is the creation of tables with the "create table" instruction. This instruction allows to create a table based on the definition of its columns by their names and by the type of data they will contain. So the syntax is of the "create table" type, the name of the table in brackets, the series of columns by their names and types, separated by a comma. The type of column we can create is very variable. We have here on the left a summary table usable in a SQLite database with the interest of having summarized a wide variety of different types in 5 categories which are the integer, the text, the blob for the binary elements such as images, the real values and the numeric elements. On the right side, we have a table of the different types of data that we can define in a postSQL or postGIS database and we see that the list is quite considerable. As an example we will create a table of employees of the Seychelles' hotels with some fields identifying the name, the hotel in which the employee works.

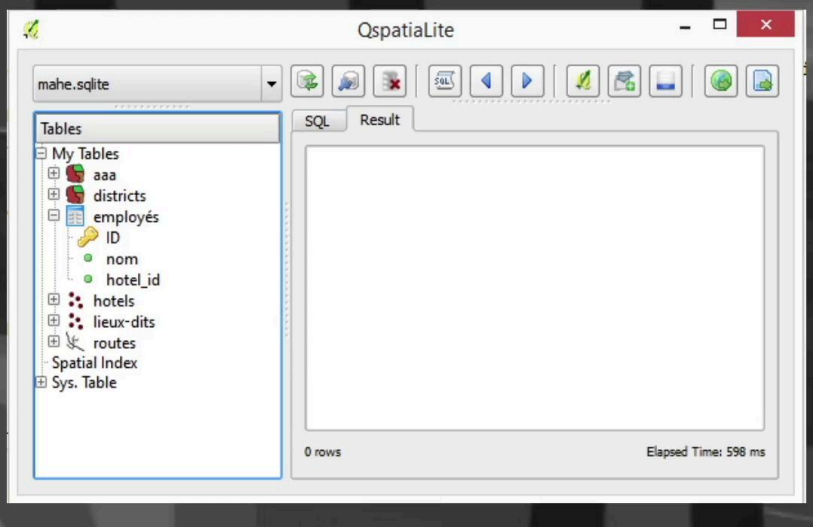
Notes

Summary

3m 41s

Data definition

CREATE TABLE



We will define a primary key and a foreign key. First, the identifier as an integer then the name in text format, the identifier of the hotel, also in integer form, after which we will define the primary key of this table. This primary key will be the identifier of the employee, ID. Then a foreign key to express the fact that the hotel_id field refers to the identifier field of the hotel table which is another table. The request being written in execut this creates this employee table with the three fields with identifier field as primary key.

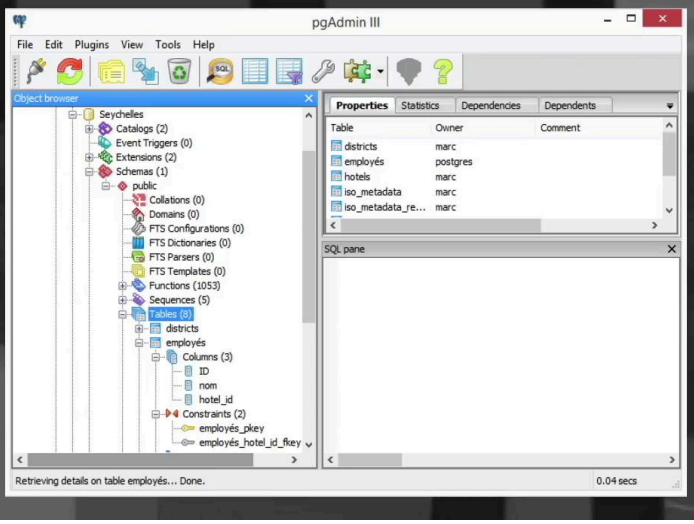
Notes

Summary

5m 30s

Data definition

CREATE TABLE

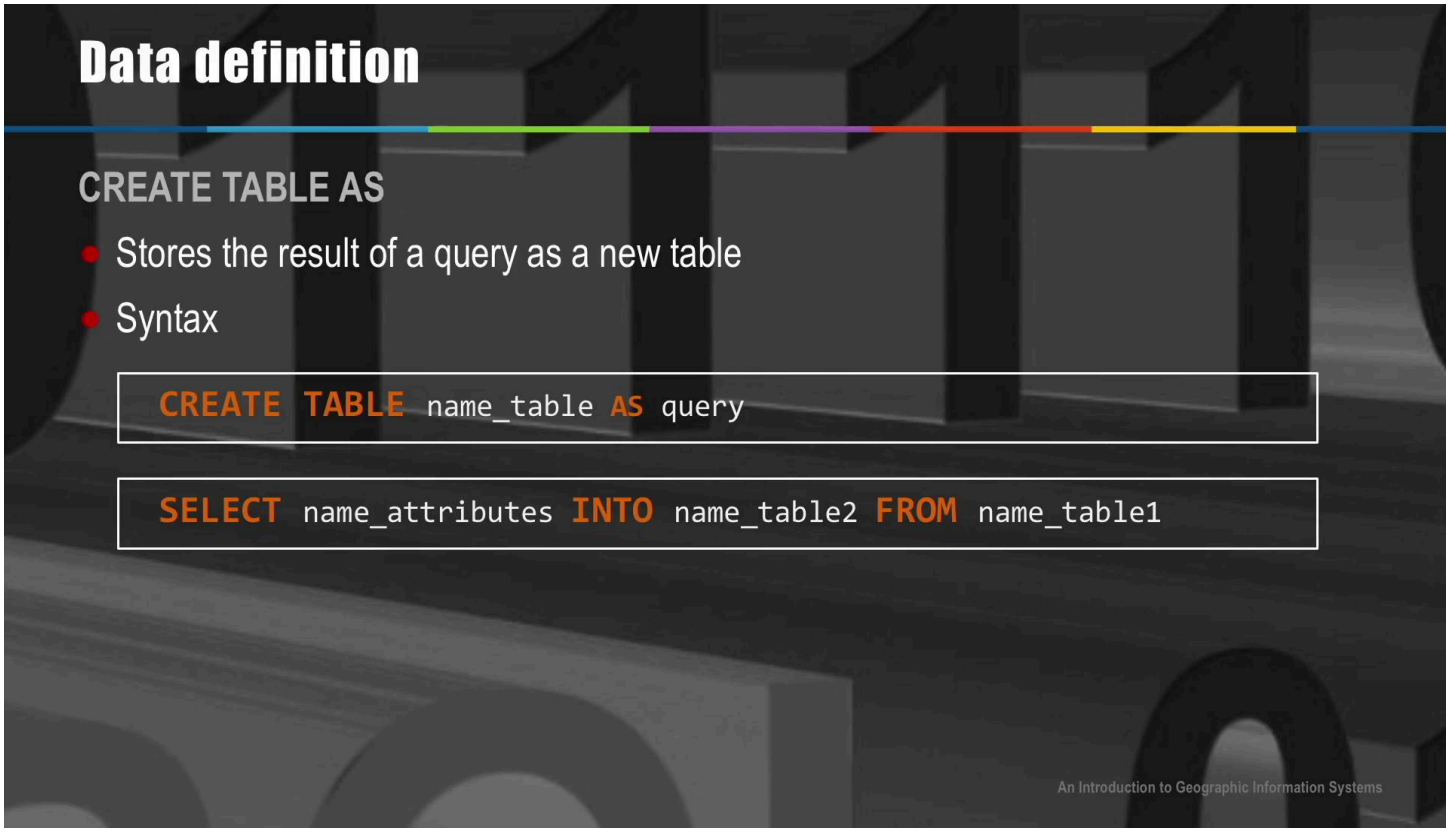


The same type of operation in the case of PGadmin is done graphically through the data manipulation interface so we create a new employee table to which we will add a number of fields, so first the identifier field as an integer. We then add the names of the employees as text then as before we will add an "identifier of the hotel" field to make the join with the table of hotels so it is also an integer here. In the constraints, we will add the primary key, we said that we took the ID field of this new table and then we will add a secondary key that we will define by referencing, by linking the ID hotel field of the employee table with the ID field of the table of hotels. We see that the SQL tab allows to consult the syntax of the query which was created by manipulating these graphic objects. The execution of this request leads to create the employee table with its three columns and then its two constraints in the form of a primary key and a foreign key.

Notes

Summary

6m 28s



A somewhat particular case of table creation with the "create table as" instruction which allows to transform the result of a selection query into a new table. The general syntax is "create table", name of the table, AS, the selection request that we want to store as a new table.

Notes

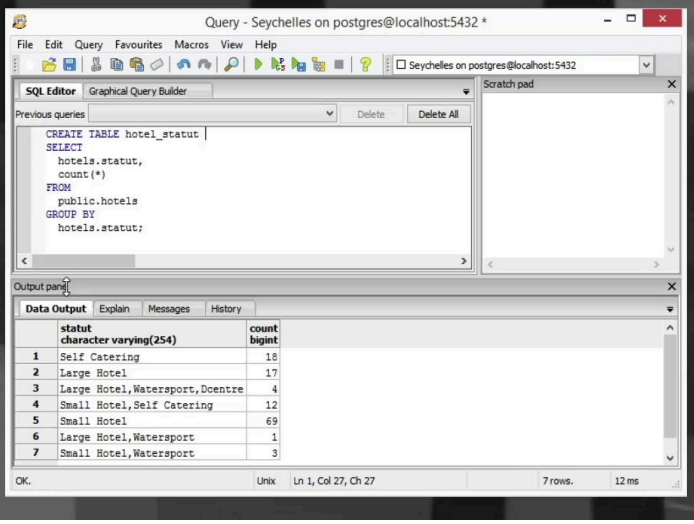
Summary

7m 49s

Data definition

CREATE TABLE AS

SELECT INTO



There is, however, an alternative syntax in the form of a "select into" request where it is a simple request in which the fact of adding the INTO keyword followed by the name of the new table we want to create, leads to the same result namely to store the result of the query in a new table. To illustrate these table creation operations from selection queries we go back to this request which gave the number of hotels in each type of status. We had seven categories, seven types of hotel status with the number of associated elements. We then simply add the clause the "create table" instruction. We will call this table "hotels status" with the AS keyword and then the query that follows. We execute this query and we see that we have created a new table which includes the two "status" fields and number of hotels and when we display the data of this table we see that we find our seven categories with the corresponding number of hotels. The same thing happens in the PGadmin world where we have the same query and similarly we simply add the "create table" clause, the name of the table, "hotels status", AS and the original request of selection behind.

Notes

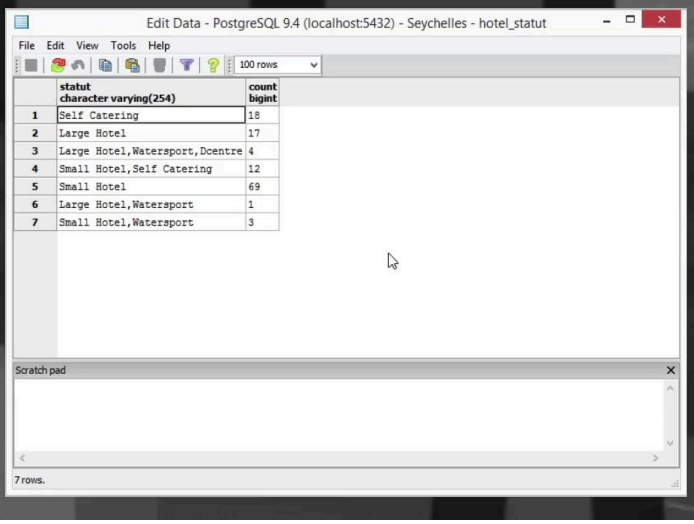
Summary

8m 12s

Data definition

CREATE TABLE AS

SELECT INTO



We can see that when this request is executed and that we go back to the consultation interface of the database, we have created this "hotel status" field which contains these two columns and when we consult the data we find the data we were looking for.

Notes

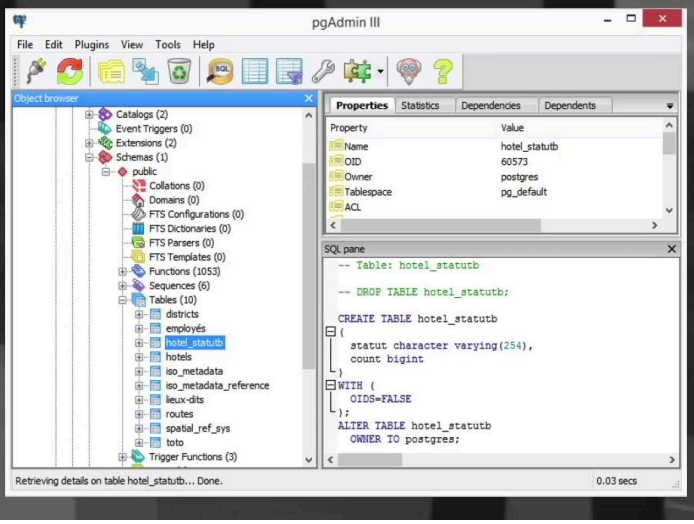
Summary

9m 59s

Data definition

CREATE TABLE AS

SELECT INTO

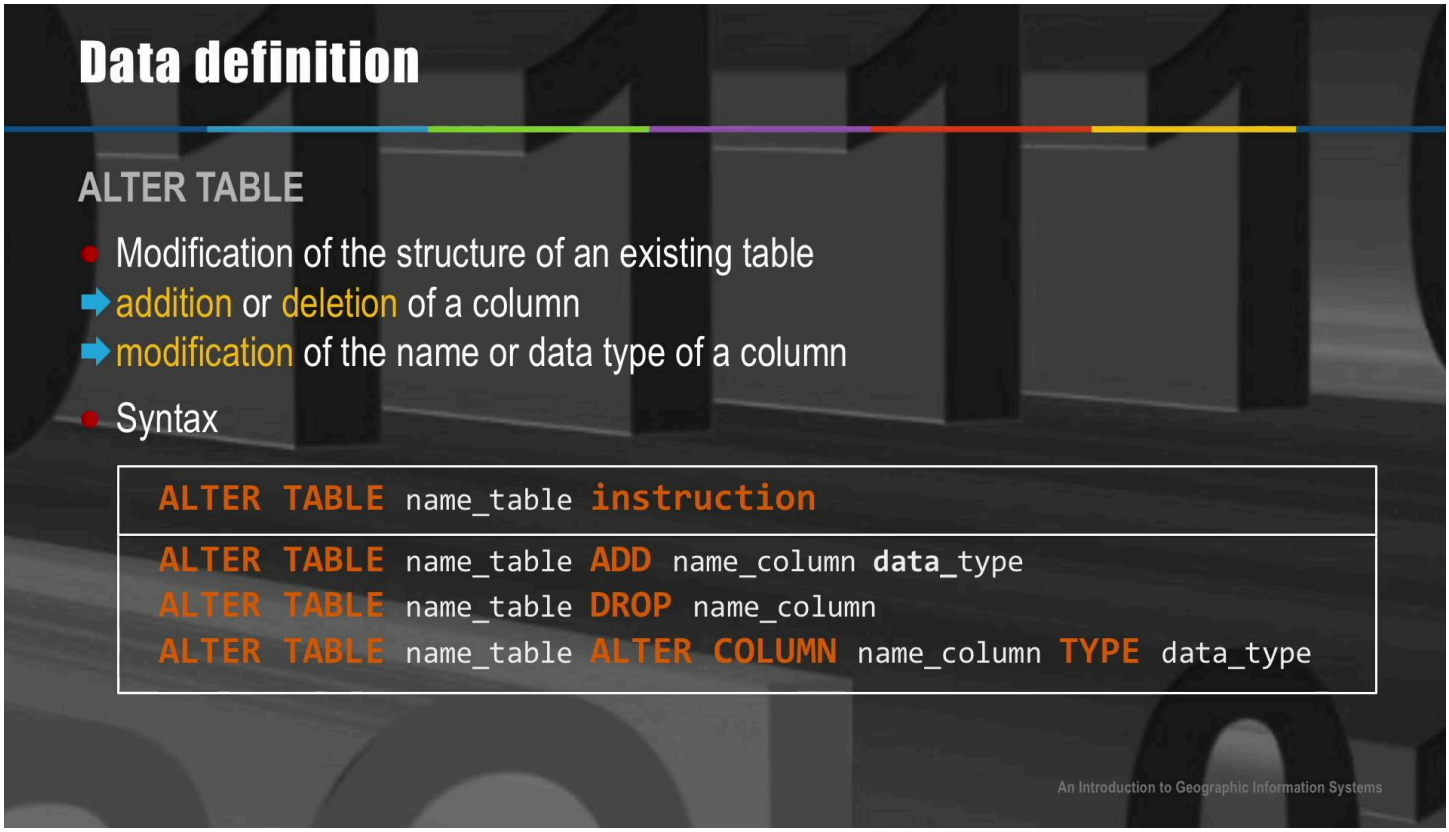


The alternative syntax consists in simply adding the INTO keyword with the name of the new table we want to create in the middle of the selection request to obtain the same result with here a table that we have called "hotels status B" and which contains the same fields and the same values.

Notes

Summary

10m 23s



Second interesting operation in the data definition field when a table has been created, that it exists, we may want to change it modify its structure by adding or deleting a column or by changing the name or type of a column. So we have several types of syntax that are possible with the "alter table" instruction, the name of the table, followed by an instruction which can either be ADD, the column name and the data type, or DROP, the name of the column to delete a column or "alter column" with the name of the column we want to change and the new data type that follows the type keyword.

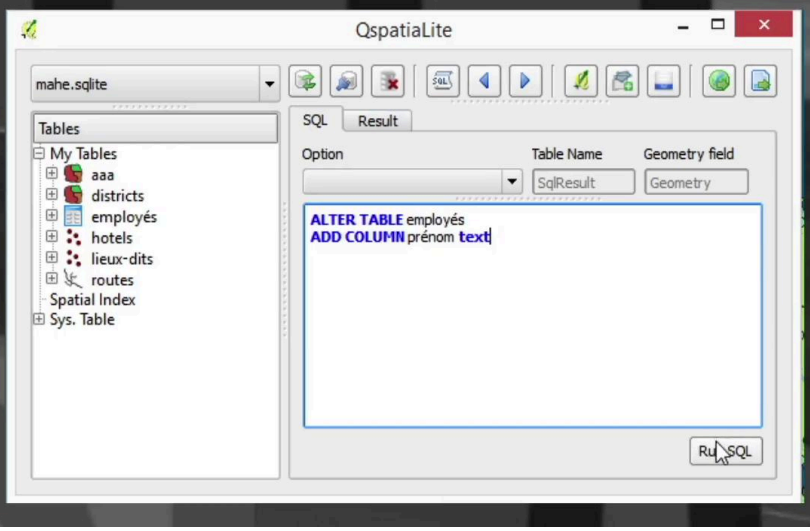
Notes

Summary

10m 51s

Data definition

ALTER TABLE



To illustrate this operation we take the employee table and we will add a column to contain for example the first name in text format.

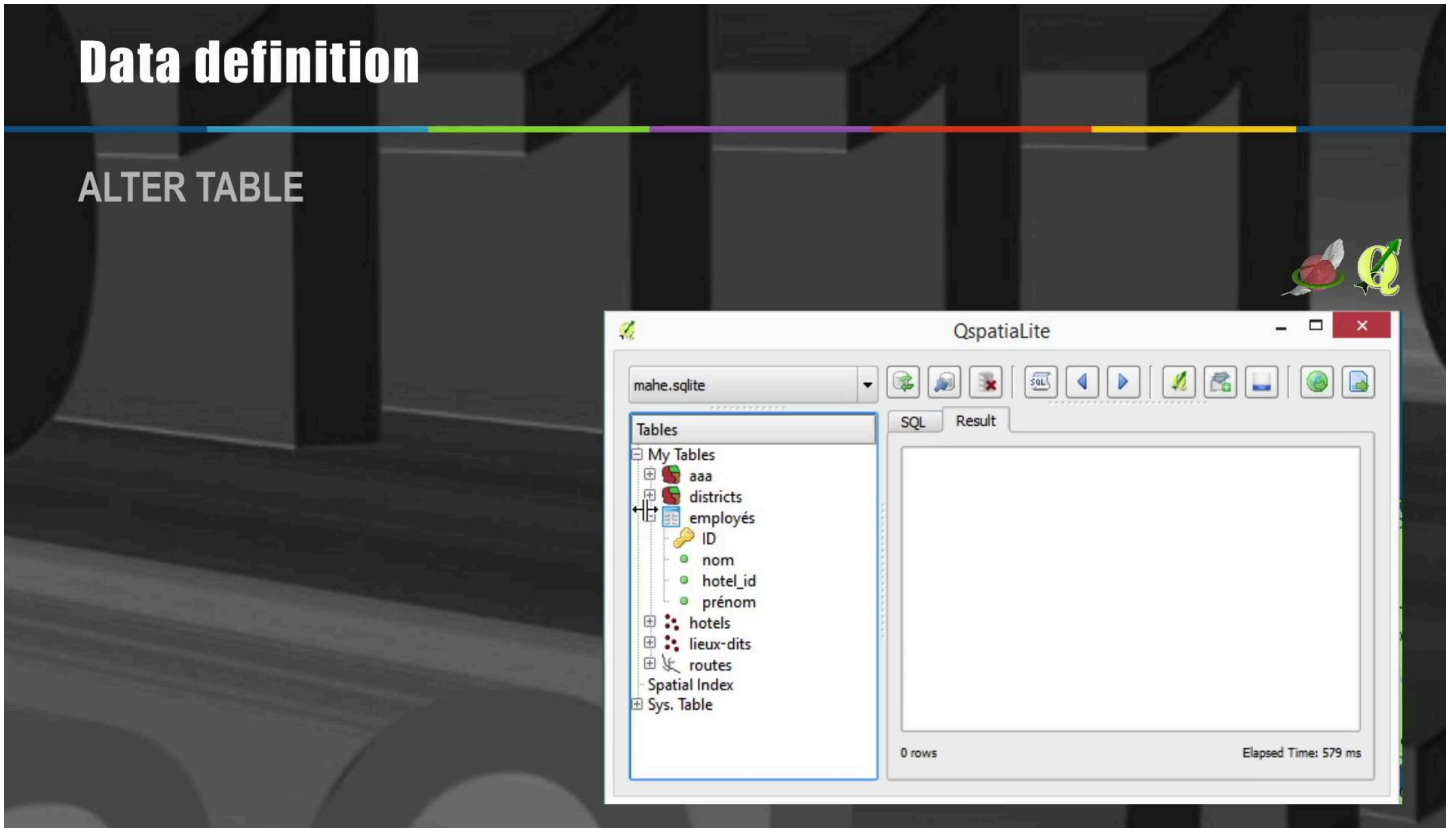
Notes

Summary

11m 31s

Data definition

ALTER TABLE



We execute the query and we see that this new field has been added.

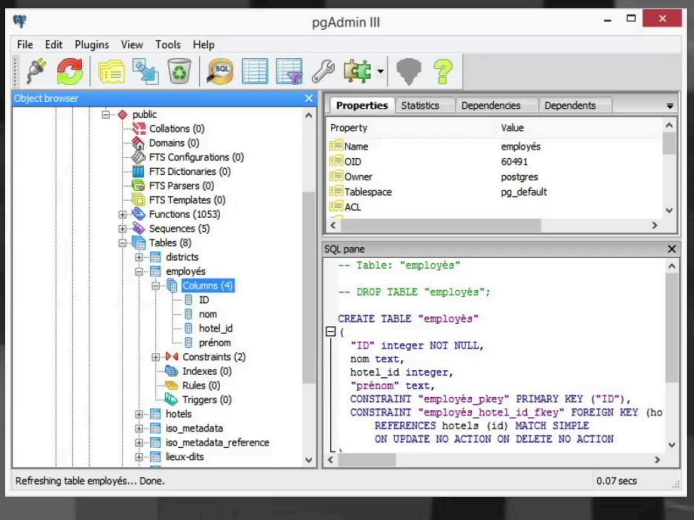
Notes

Summary

11m 47s

Data definition

ALTER TABLE

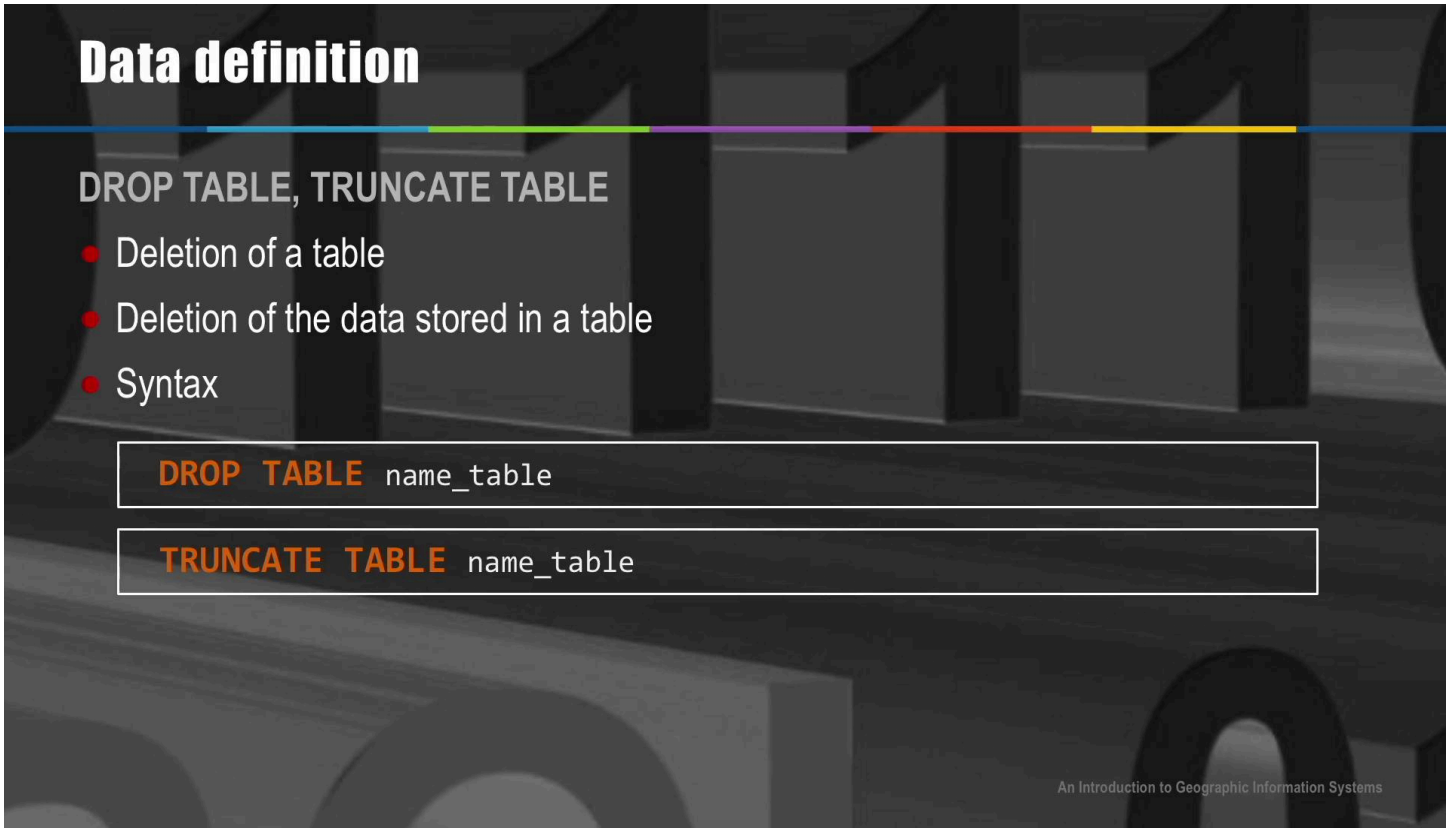


Same thing in PGadmin but this time from the editing interface of the data structure, we add a column, we define its name, its type, its first name and the text type and we see that the SQL tab gives the SQL equivalent of the instruction that is being prepared. We execute this query and we create this new field.

Notes

Summary

11m 52s



Finally, two instructions which allow to delete a table with "drop table" or to delete the content of a table while still preserving its structure with "troncate table". The syntax is simple, the instruction followed by the name of the table on which it is.

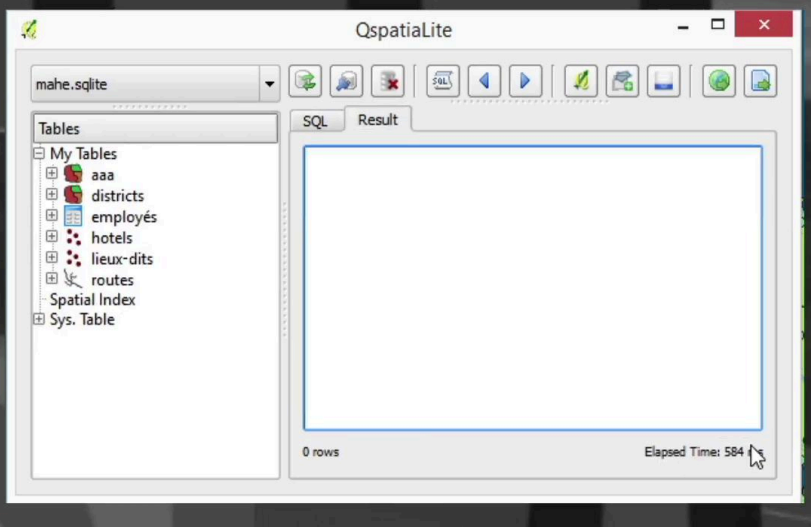
Notes

Summary

12m 13s

Data definition

DROP TABLE, TRUNCATE TABLE



For exemple we can simply take "drop table" to delete the table that we created earlier, the hotel status table.

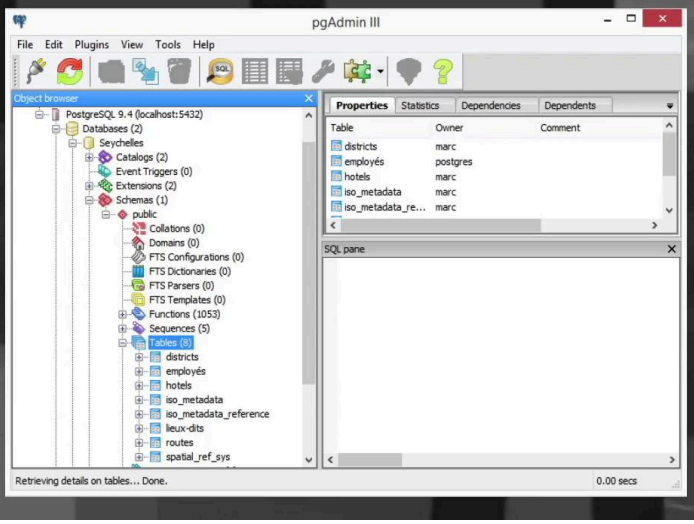
Notes

Summary

12m 32s

Data definition

DROP TABLE, TRUNCATE TABLE

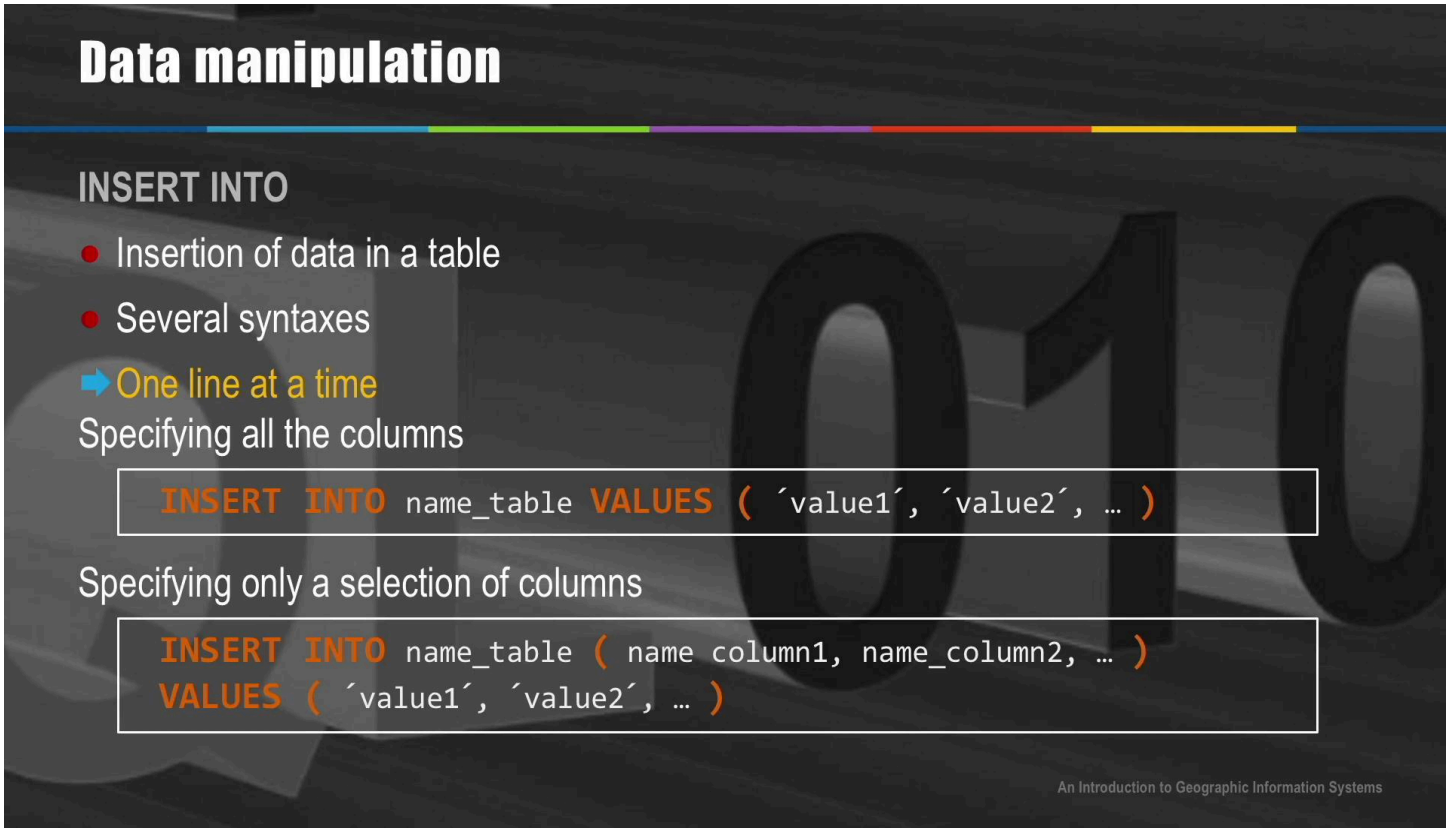


In the SQL domain simply the "troncate" function which allows to notice that the table has been emptied and the "drop" instruction to remove the table itself.

Notes

Summary

12m 46s



In the data manipulation register what we wish to do in the first place, we have created a data structure, we have tables and we would like to add data to these tables with the "insert into" instruction. This statement allows to insert data into a table. It has several syntaxes depending on how we proceed. A first series of syntax concerns the insertion of one line of data at a time and this can be done in two ways, either by giving a value for each of the columns of the table, with a syntax of "insert into" type the name of the table, the VALUES keyword and then the series of values, a value for each column of the table, separated by commas and in brackets. Another possibility, we do not want to document all the fields all the columns in the table and we can specify to which columns the values that we gives as argument correspond with an "insert into" syntax, the name of the table, the series of column names concerned by the values that will follow, in brackets and separated by commas, the VALUES keyword and again as before in brackets, the series of values which corresponds to the order of the columns previously listed.

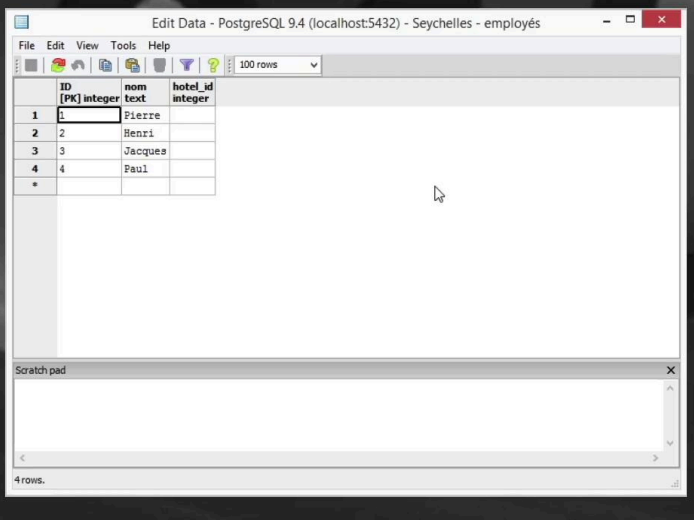
Notes

Summary

13m 19s

Data manipulation

INSERT INTO



Alternativement if we want to insert several lines at a time with a single instruction, so with an "insert into" syntax, the name of the table, the VALUES keyword and a set of value vectors with a value per column of the table so vectors of value in brackets, separated by commas. we have seen in all these instructions that when we add a text field we must indicate the contents in inverted commas, simple inverted commas. In the case of numeric values inverted commas are not required. As an example of the use of these instructions of data insertion we will add data to the employee table. We begin by adding a first element with "Pierre" and the identifier 1. We see that we did not mention all the fields of the table but their order and by default, the missing value was placed as null. We may also want to enter the data in a different order starting with the person's first name and then the identifier. Alternatively, by respecting the order of the fields in the table, we may want to introduce several elements in a row. We see that with all this in our employee table we now have the 4 employees that we added. The main difficulty with this type of addition is the choice of the identifier.

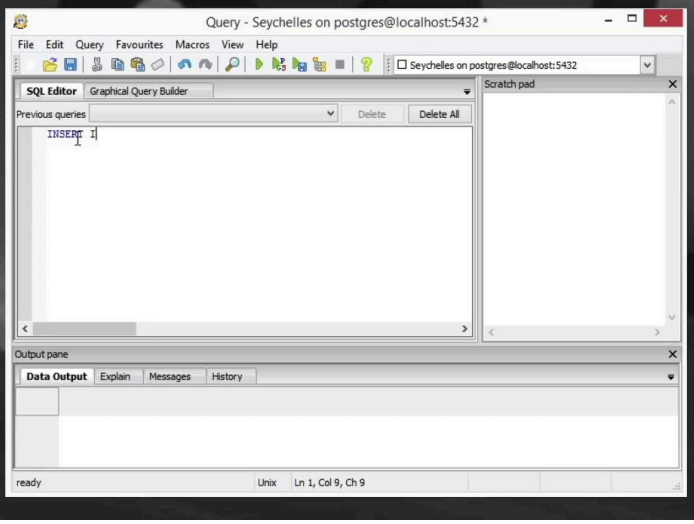
Notes

Summary

14m 44s

Data manipulation

INSERT INTO



If we suddenly have a hundred elements that we would like to add to the table it is a little tedious to define each time the correct identifier. It is necessary to know where we are in the list of these identifiers to ensure that it is a unique value. To be able to perform such operation we need to redefine the identifier field, so we start by deleting the existing field and adding a new column with this new identifier field which will no longer be an integer type but a serial type. We add in the definition of the table this new identifier field as primary key and we can, for good measure, clean the existing table, so we select the existing elements, we delete them and we start again with an empty table. It then becomes possible to write a multiple data insertion query which will add to the employee table, in the name section, all the results of another request which will consist in looking for the names of the hotels taken from the hotel table for which the name of the hotel is for example at Albert, André François, Paul or someone else. This query executed, we come back to the data management interface, we observe that 3 elements have been automatically add and that the identifier has been automatically added as well.

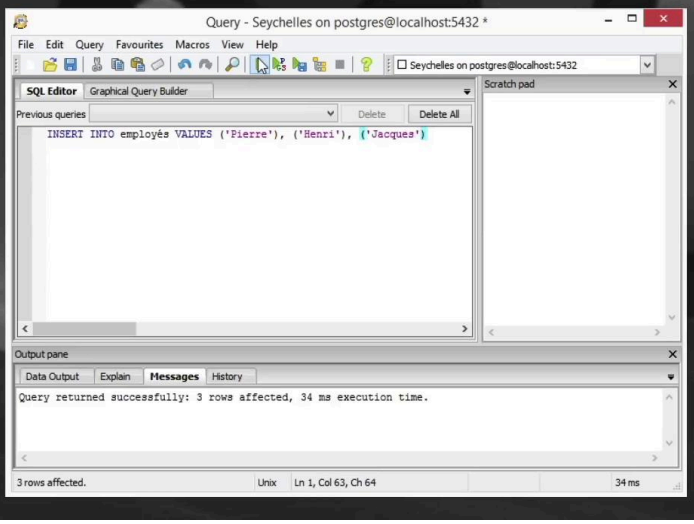
Notes

Summary

16m 38s

Data manipulation

INSERT INTO

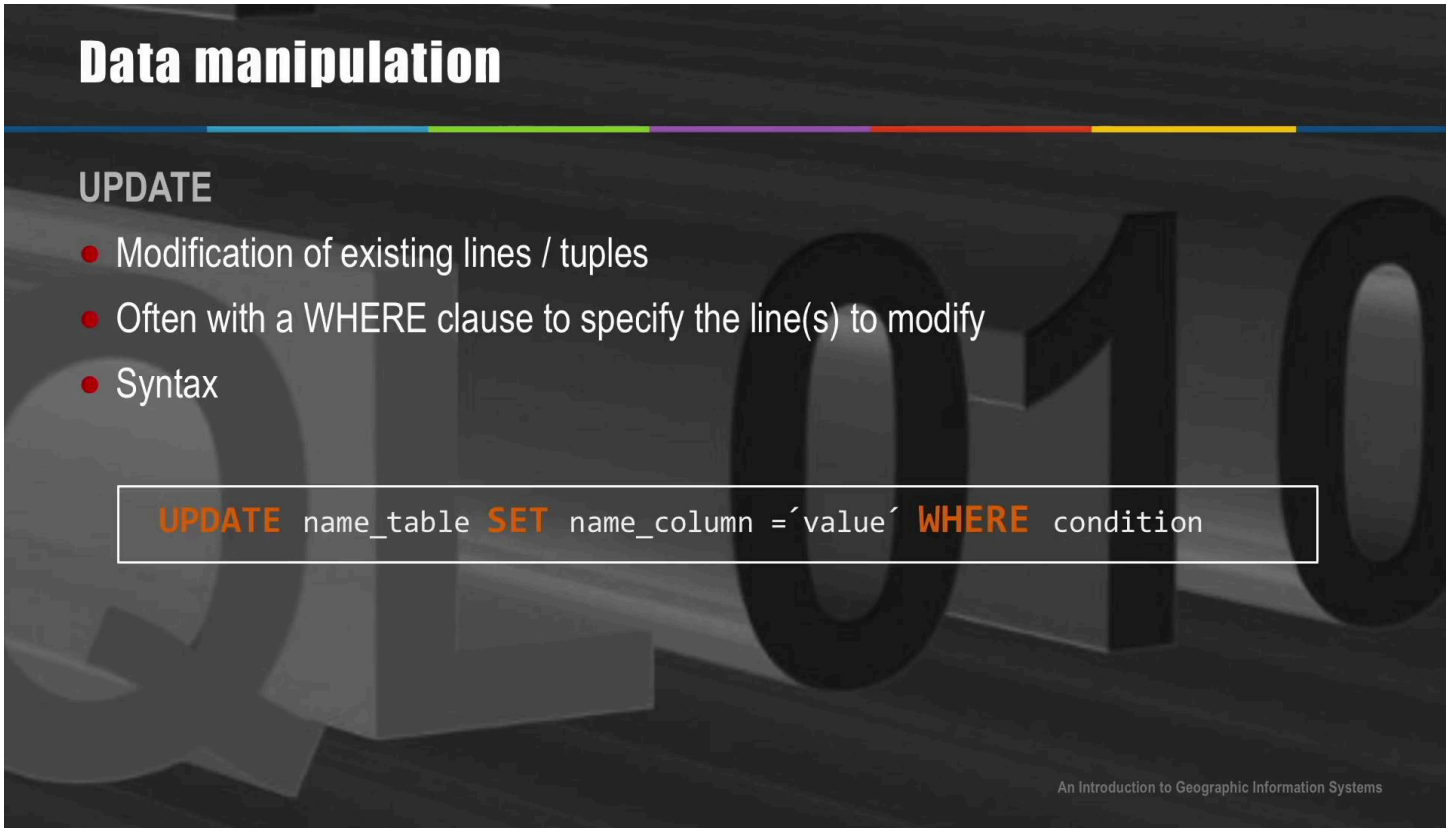


We see that with this new definition of the identifier no longer as an integer but as a value of serial type, it becomes possible to add, to insert employees by simply giving the list of first names, separated by commas.

Notes

Summary

18m 26s

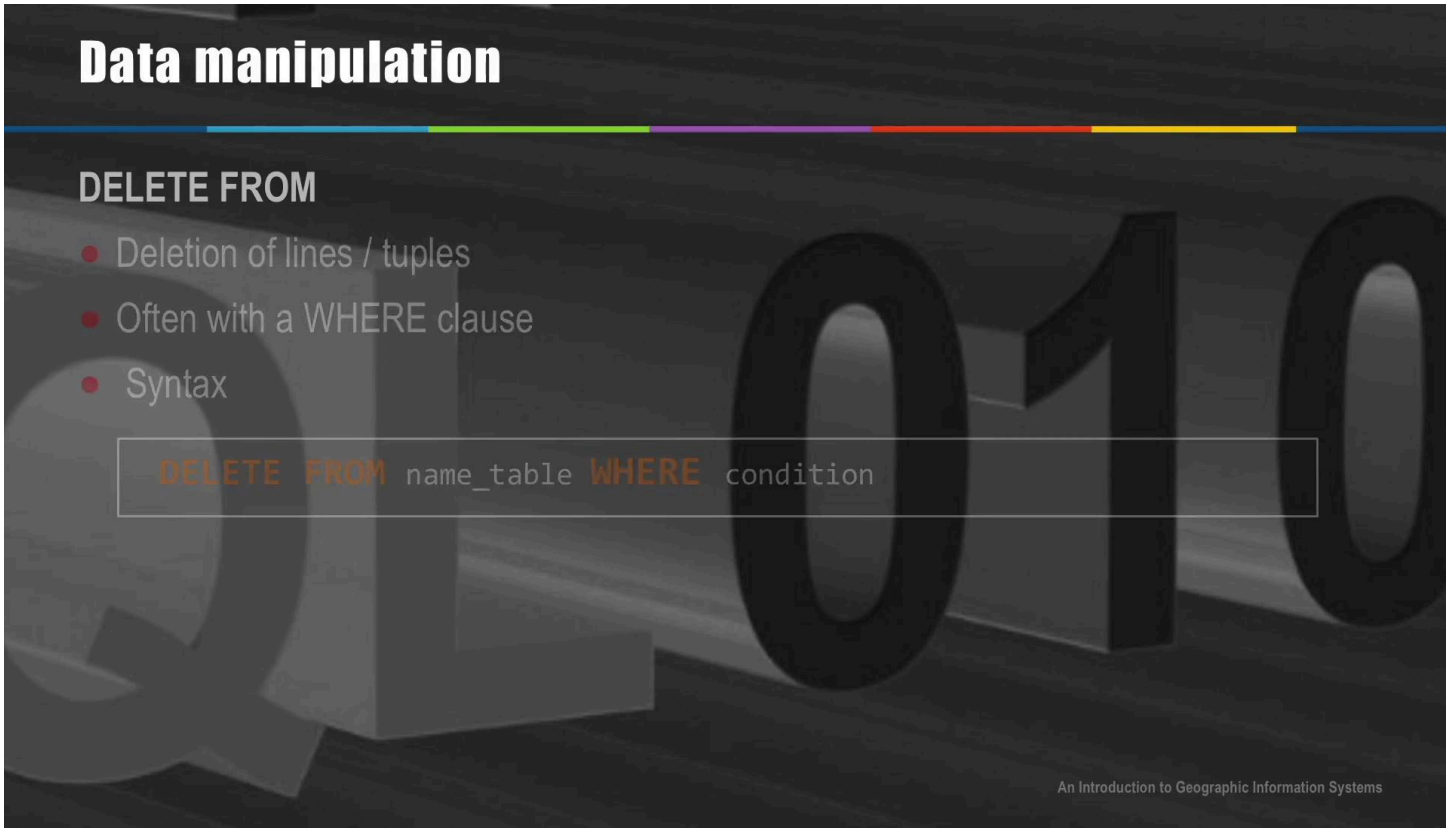


The "update" instruction allows to update the values of existing fields in a table. It is often associated with a WHERE clause to specify on which line the modification should be and possibly to propagate an update over several lines that would satisfy the same criterion. The syntax is therefore "update", the name of the table, "set", the column name and the value to be assigned to this attribute and WHERE, the condition that allows to select and filter the rows of the table on which to update the "column name" attribute with the value given as an argument.

Notes

Summary

18m 46s



In the case of the employee table it is possible to carry this update by simply defining the hotel identifier for the employee Pierre, a setting that has not yet been added. We see, if we display after executing the query, the table, that this identifier is present. The "delete from" instruction, which allows you to delete records or lines from a data table, which is again like the update instruction often associated to a WHERE clause which defines the filter condition of the rows of a table that we will delete. The syntax will be "delete from", the name of the table, WHERE, the condition that the lines must fill that will be deleted.

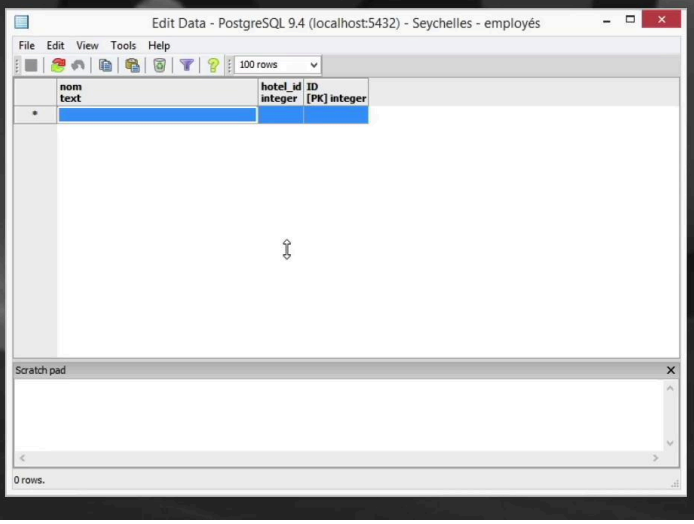
Notes

Summary

19m 30s

Data manipulation

DELETE FROM

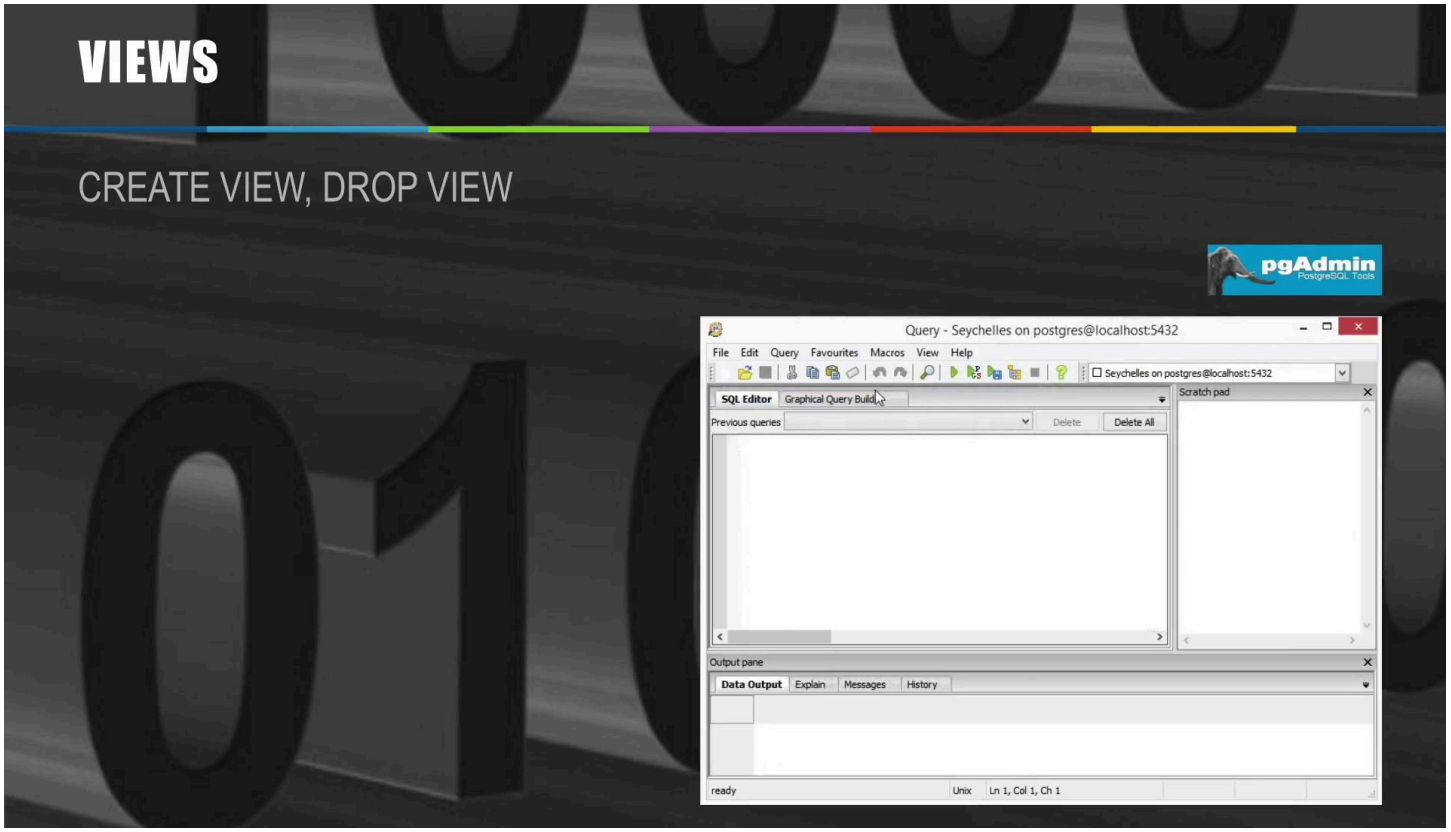


As a simple example we take the table of employees from which we will remove the line that bears the name Pierre. We see that only one line is affected and when we look at the data we see that the information about Peter has disappeared.

Notes

Summary

20m 28s

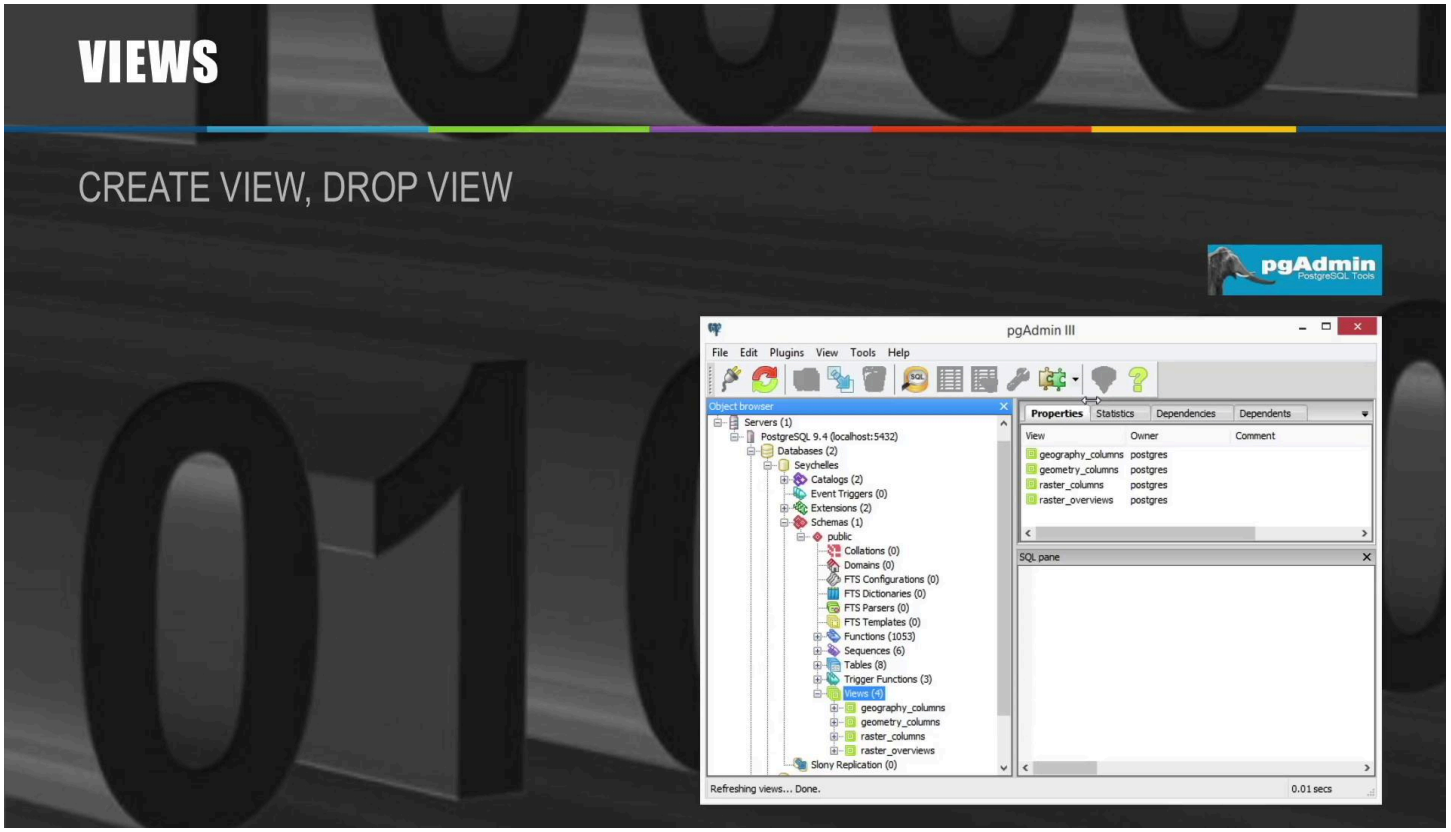


We see that we can also delete lines much more directly using the functions provided for this purpose of the PGadmin interface. It is often interesting to be able to store and reuse the results of a query in particular to construct nested queries for example, with the disadvantage however when we use the table creation functions "create as" and "select into", to create a duplication of data which causes update problems. The interest of the "view" object is that it is a virtual table, so it is a non-editable extract of existing data, so simply a data presentation tool that is a little different. The original data remaining stored in the original tables. There is therefore no data editing in these views. The syntax is simple, "create view", name of the view, AS and the selection query that combines the different data that we want to present in the view and "drop view" to delete this view. To illustrate these creation and view deletion instructions we will create a test view as a simple reflection of the hotel table, so we select all the fields of the hotel table and we create this view. We note that this new view is available in the views collection and if we look at these data we find the hotels of the Seychelles but in a version that is not editable.

Notes

Summary

20m 50s



What is interesting is to see that we can then, when we make other queries, we can find this view in the usable elements to build queries in the same way we could use tables. Deletion, either by the menu element integrated in PGadmin or as an SQL instruction, "drop view", the name of the view and we see that the view has indeed disappeared.

Notes

Summary

22m 51s