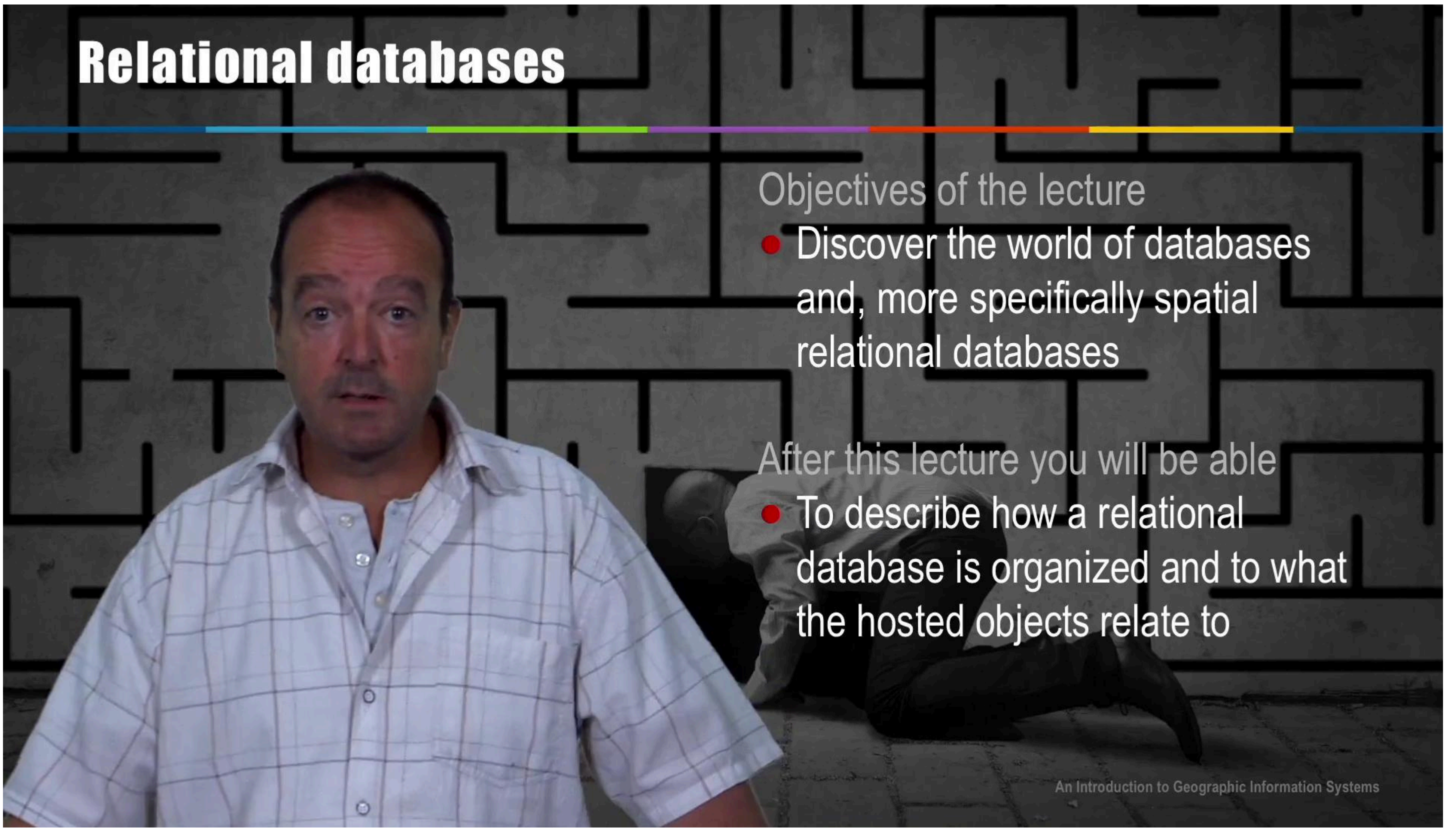
Search MOOC

Video

EPFL

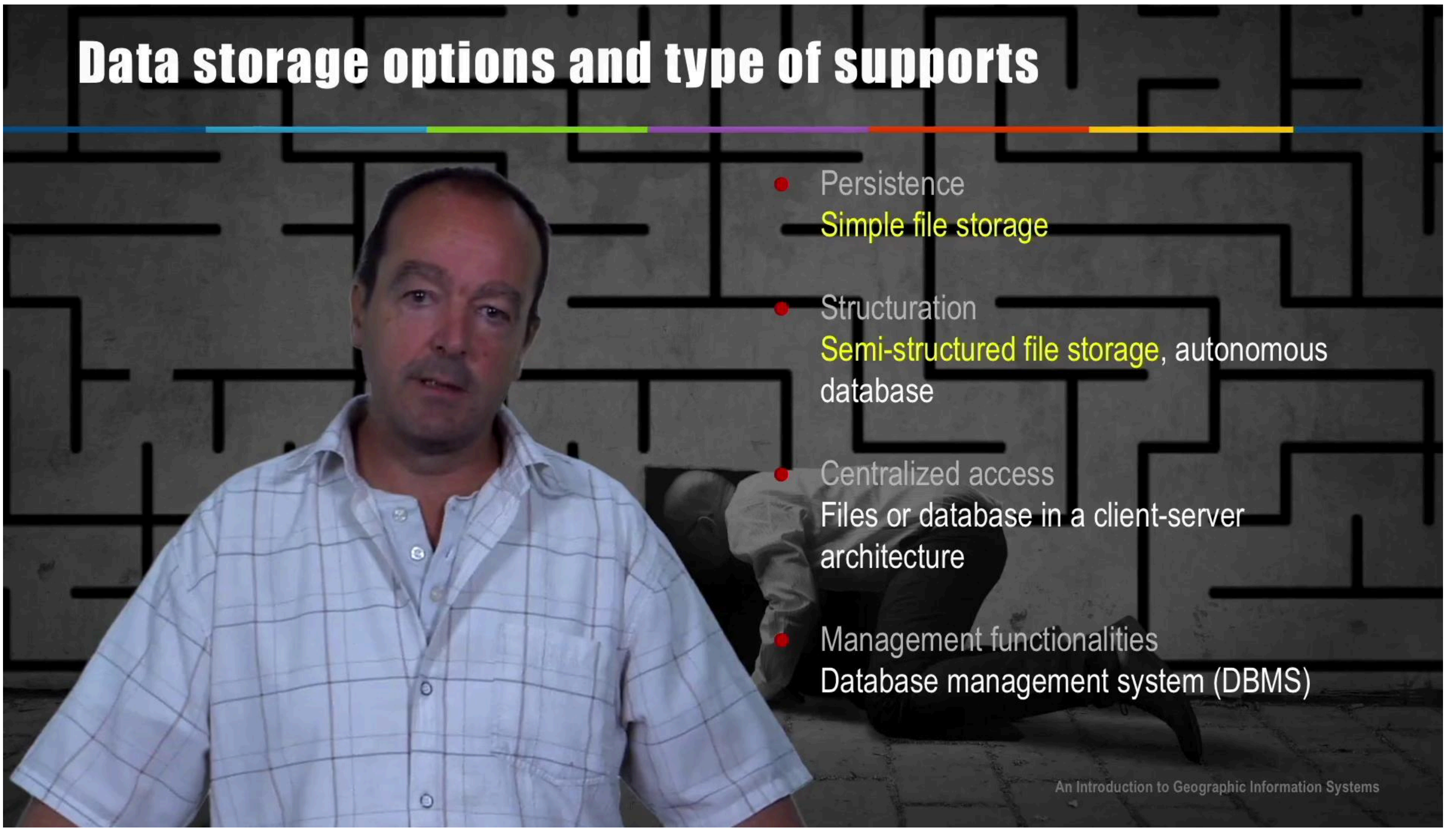


Welcome to this course which will focus on relational databases which are the most widely used form of database and which we find in absolutely all the domains of daily life.

Notes

Summary

0m 24s

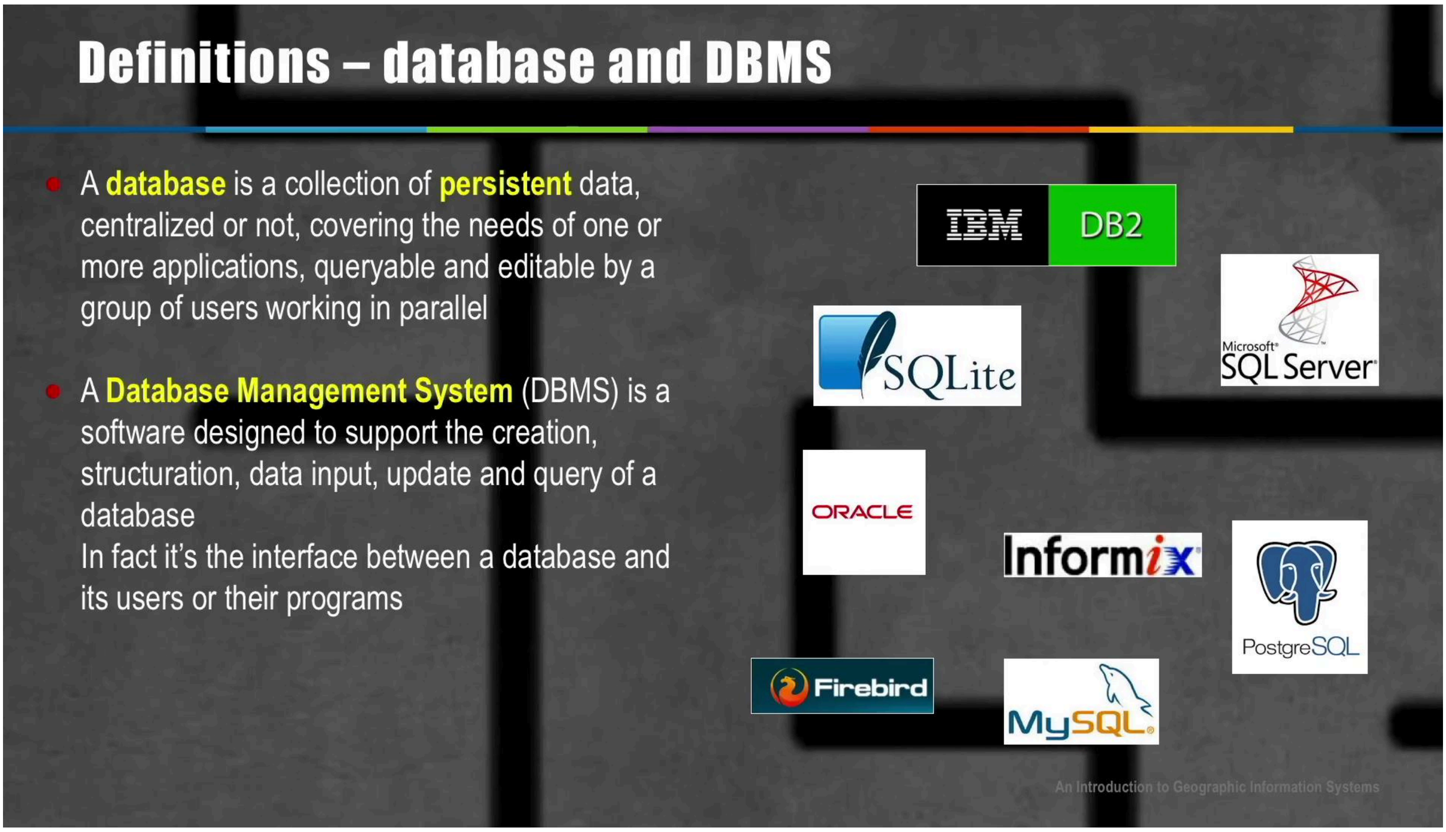


The objectives of this lesson consist in discovering the databases in particular relational databases and how the spatial dimension is integrated into these databases. At the end of this lesson you should be able to describe the concepts that form the relational databases as well as the various types of objects they contain. In the previous lesson we saw several forms of data storage as simple files or semi-structured files. Today, we will review the data storage possibilities namely the databases which allow a more structured storage and most of the time a centralized storage in a client-server architecture.

Notes

Summary

0m 36s

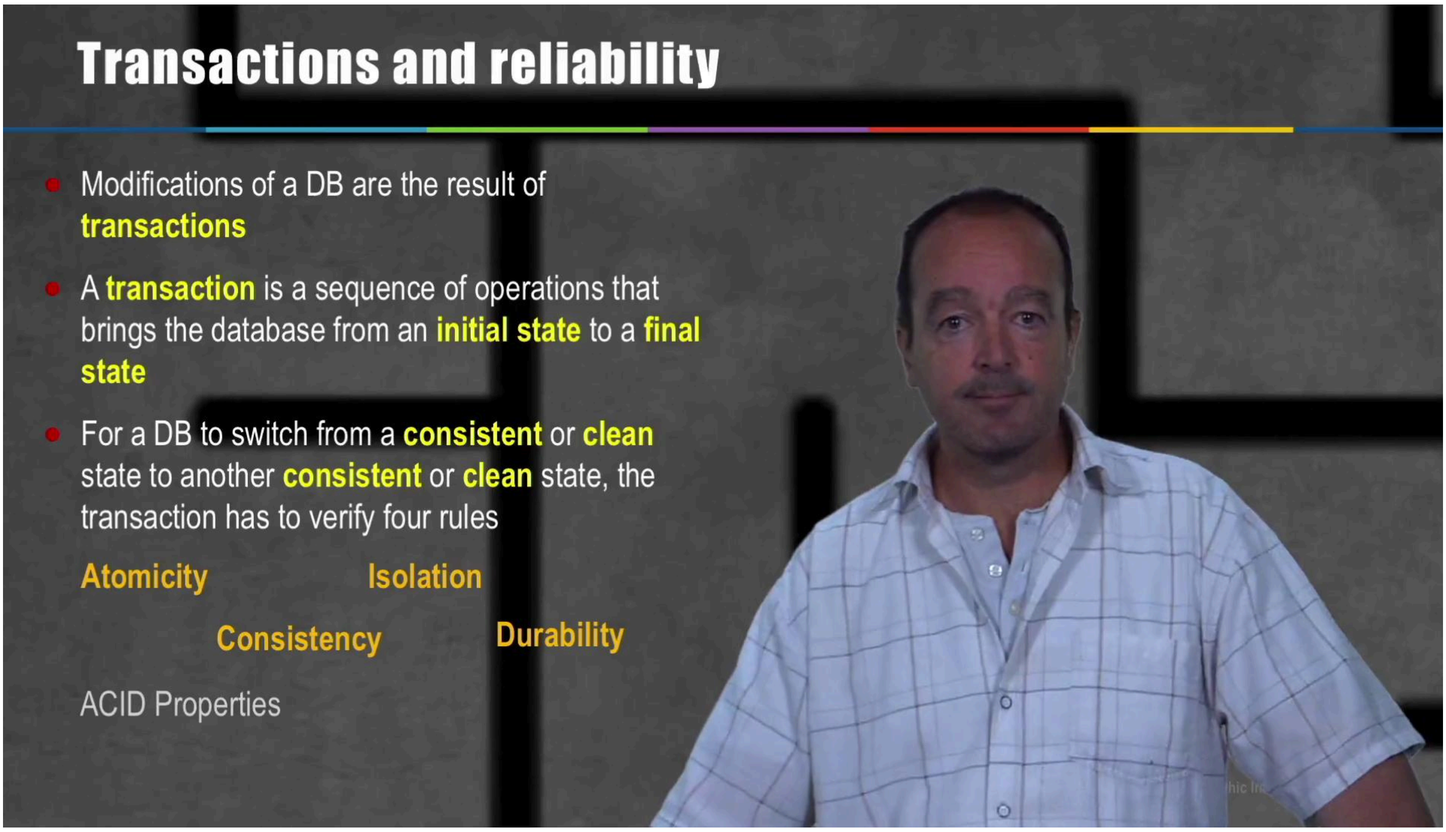


In this lesson, we will address the question of defining a database the reliability notions of these databases. We will give a brief historical overview of the evolution of the different types of database. We will focus on the relational model which we will see in detail. We will then discuss how the spatial component is managed in these databases. And finally, a first brief overview of DBMS softwares, Database Management System softwares, that we will also see on several occasions in the following lessons. A database can be defined as a collection of persistent data possibly centralized used by several applications and by several groups of users which can optionally work in parallel. We can see that with this very general definition the data hosting files that we saw in the previous lesson can also be considered as databases as they offer a certain persistence to the information storage. Database management systems are softwares which allow us to create, structure, document and consult the databases. It is actually the interface between the database and the users or the applications that use databases. As shown in this image there is a very large number of them both in the commercial field as in the open-source software field.

Notes

Summary

1m 22s

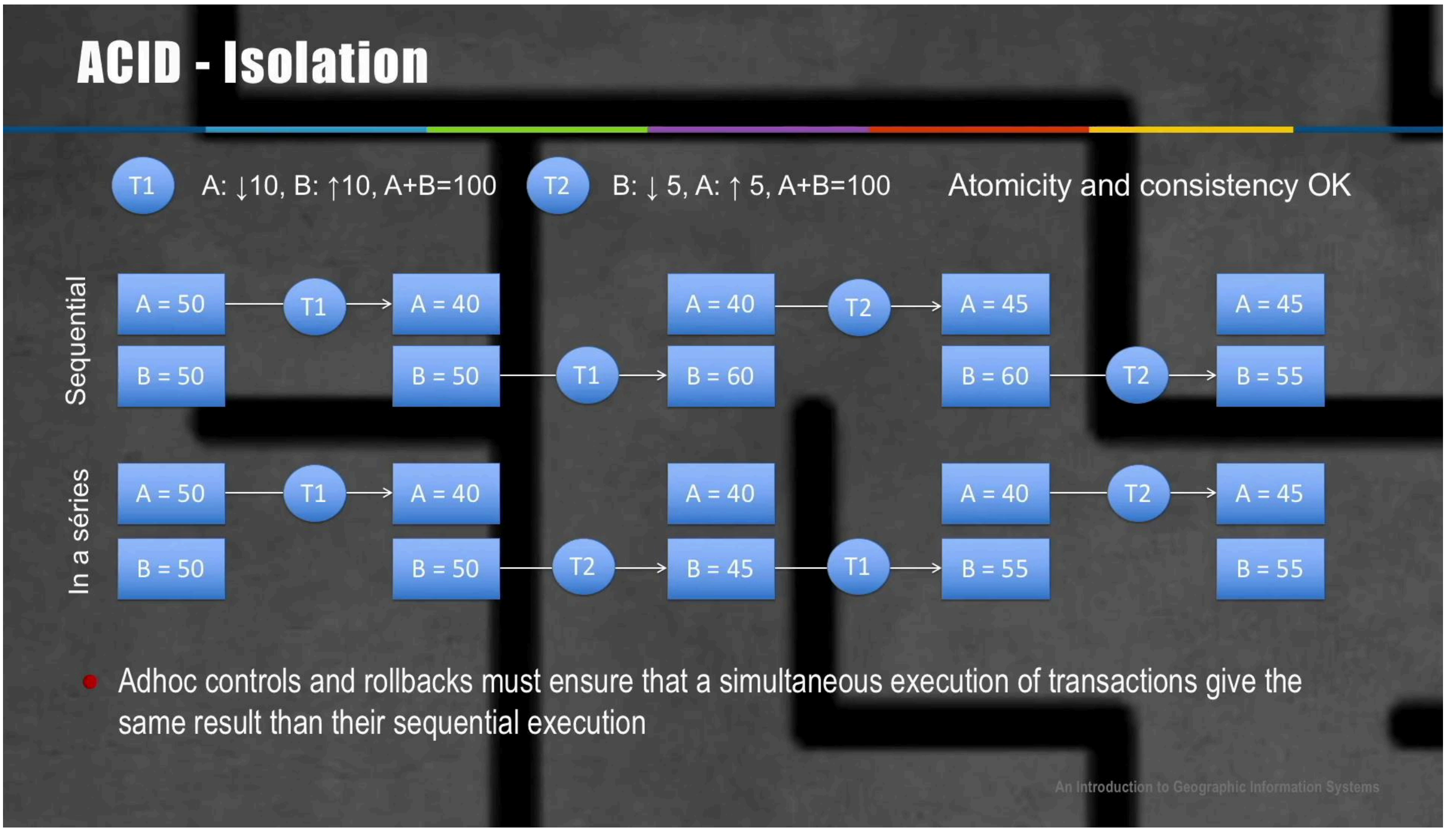


The changes in a database are the result of transactions. Les transactions being a sequence of operations which make the database go from an initial state to a final state. For a database to go from an integral and coherent initial state to an integral and coherent final state, the transactions must meet a number of criteria which are the atomicity, the coherence, the isolation and the durability, criteria that are often called "ACID" and that we will see a little more in detail now.

Notes

Summary

3m 03s

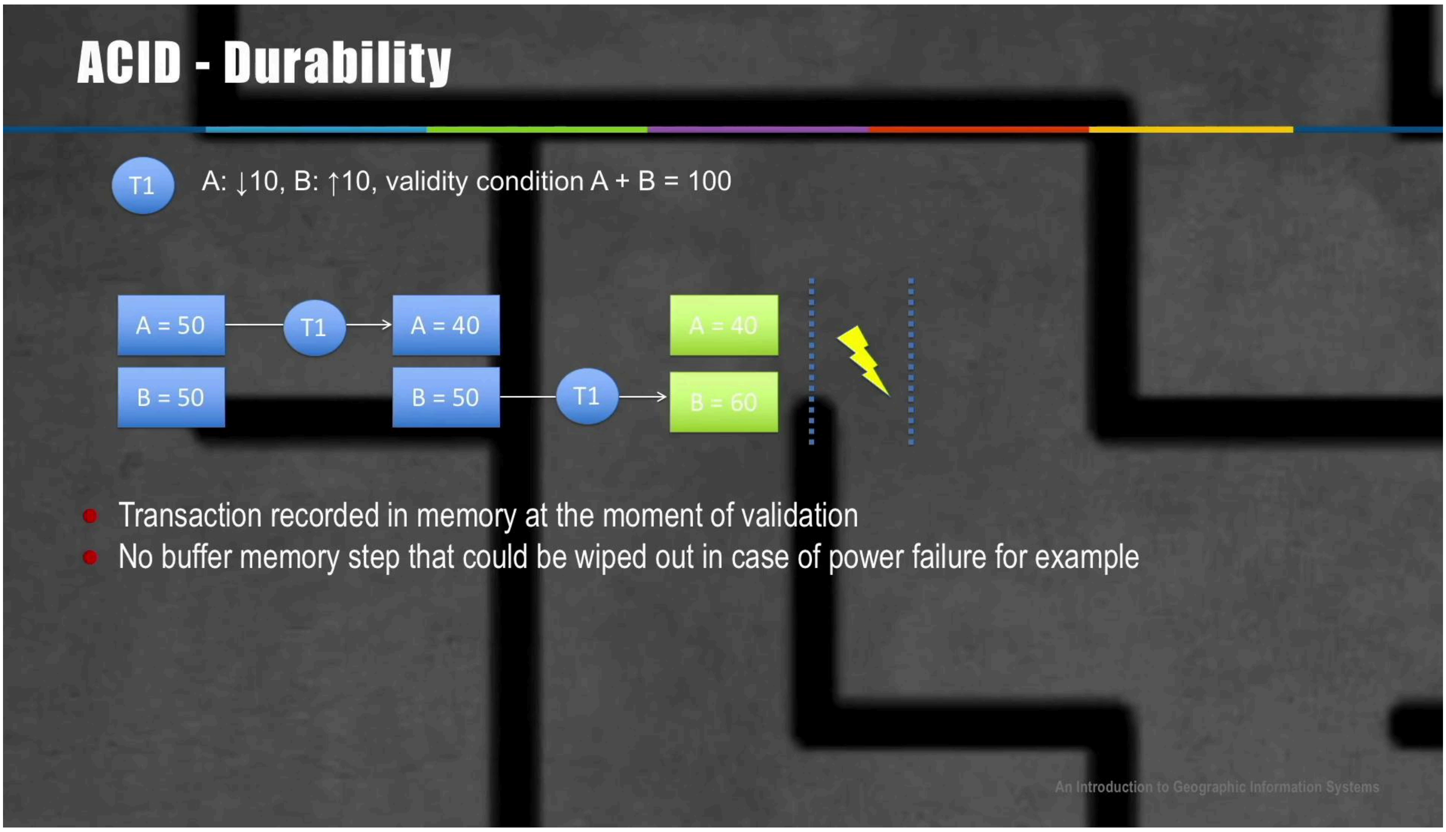


The principle of atomicity stipulates that the transaction is done completely or not at all. In the case here of a transaction which comprises two operations, the first one consisting of deducting 10 from a set A and the second one of adding 10 to a set B, with a validity condition that A + B = 100 we see that the operation is carried out in two stages to get to a result which is then stored in the database by an operation called a commit a validation operation. One of these operations can sometimes fail and in that case the principle of atomicity wants that the first operation of the transaction be reversible and canceled, meaning that you can return to the initial state of the database. Databases should verify coherency principles. If we have here the example of a transaction which consists of deducting 5 from a set A and adding 10 to a set B with a validity condition A + B = 100 as earlier, we see clearly that at the end of the transaction the result no longer meets the condition of validity so the validation has to be prevented and the transaction as a whole has to be canceled. The principle of isolation.

Notes

Summary

3m 41s

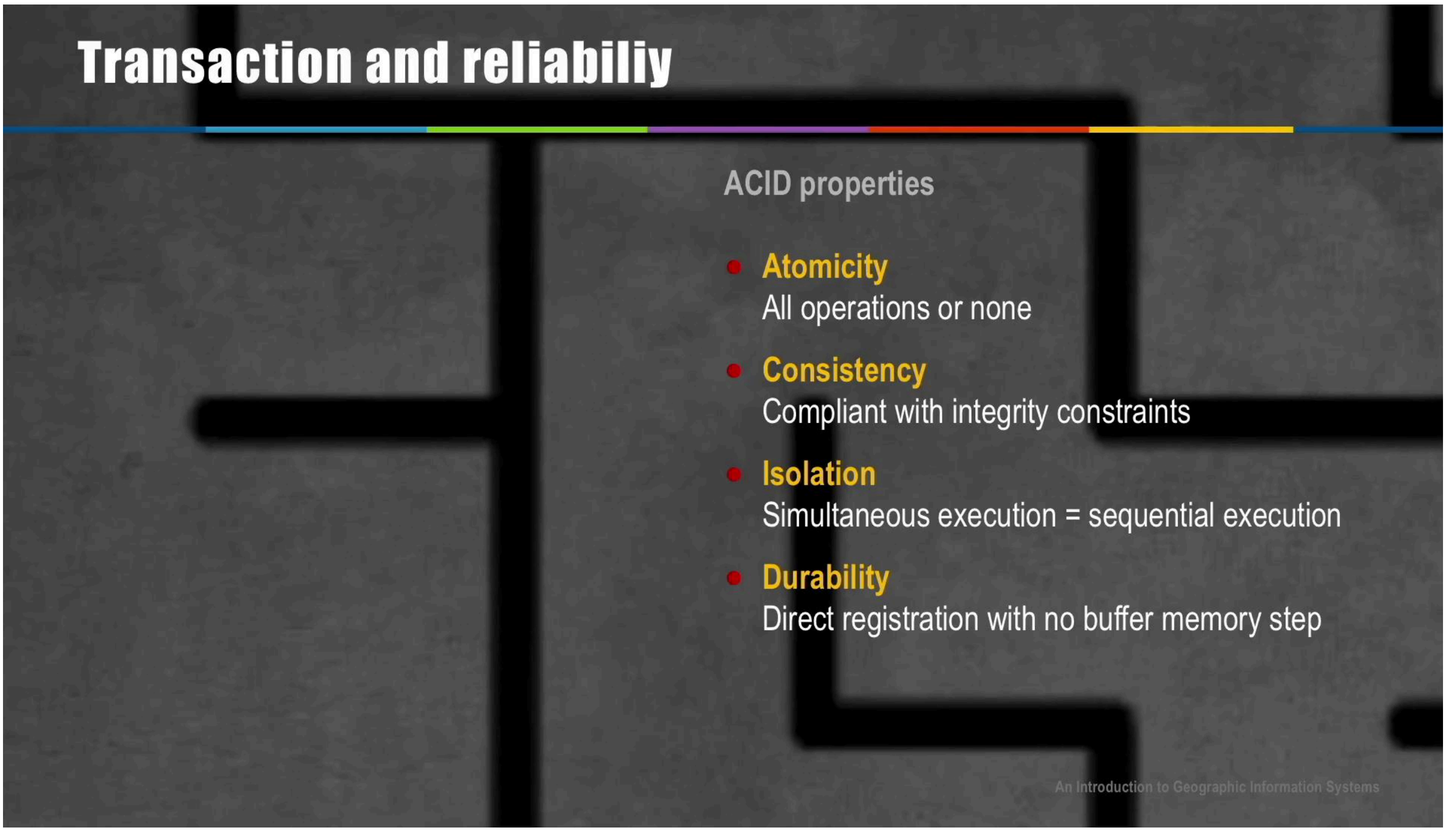


We have here the case of two transactions the first consisting of deducting 10 from a set A and adding 10 to a set B with always the same condition of validity and a second transaction which does more or less the same operation. We deduct 5 from set B and we add 5 to set A. The isolation principle requires that the execution of these transactions one after the other give the same result only if the transactions are executed at the same time that is to say in series. We see that if one of these operations fails the last one for example we have to ensure that the cancellation principle brings us back to the database in a valid state which is the case in the sequential case. But when operations are carried out in series we see that a special procedure needs to be implemented at the level of the database management system to ensure the execution in series in case of a glitch to restore the database in a valid and coherent state. Finally, the durability principle stipulates that the memory recording of the transaction at the time of validation can not be prevented or interrupted by an external event. For example, a power failure an earthquake or whatever. This means technically that it is necessary to avoid going through a buffer memory which could be erased in case of a power failure before the actual validation.

Notes

Summary

5m 01s

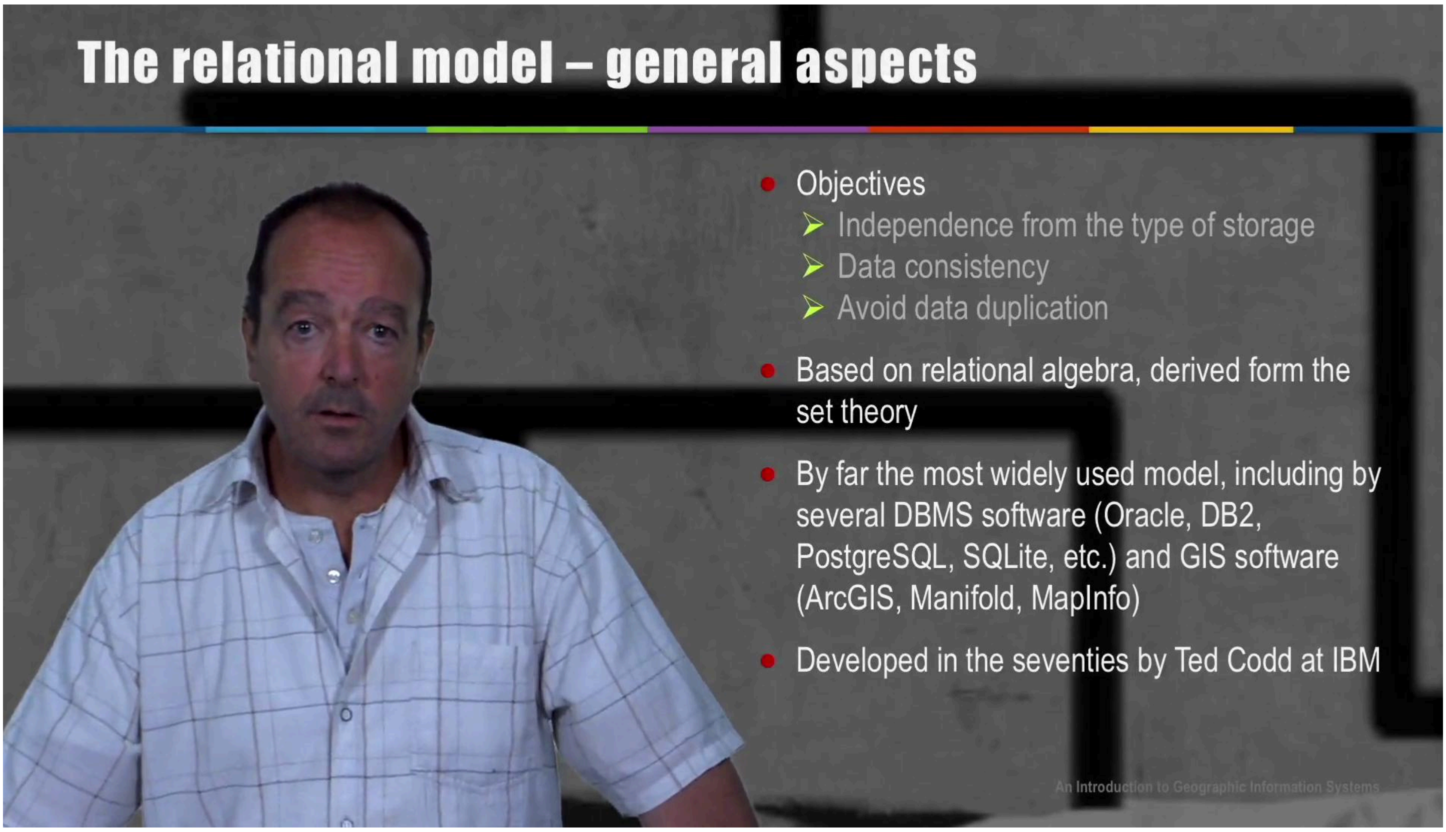


In summary, the ACID principles stipulate for the atomicity that a transaction must be completely performed or not at all, for the coherence that we will verify at the level of the database, the conditions of validity, the isolation principle, that an execution in series gives the same result as a sequential execution of the different operations of a transaction, Aad finally, the principle of durability stipulates that external incidents, power failure or other, should not affect how information is stored and validated in the database.

Notes

Summary

6m 37s

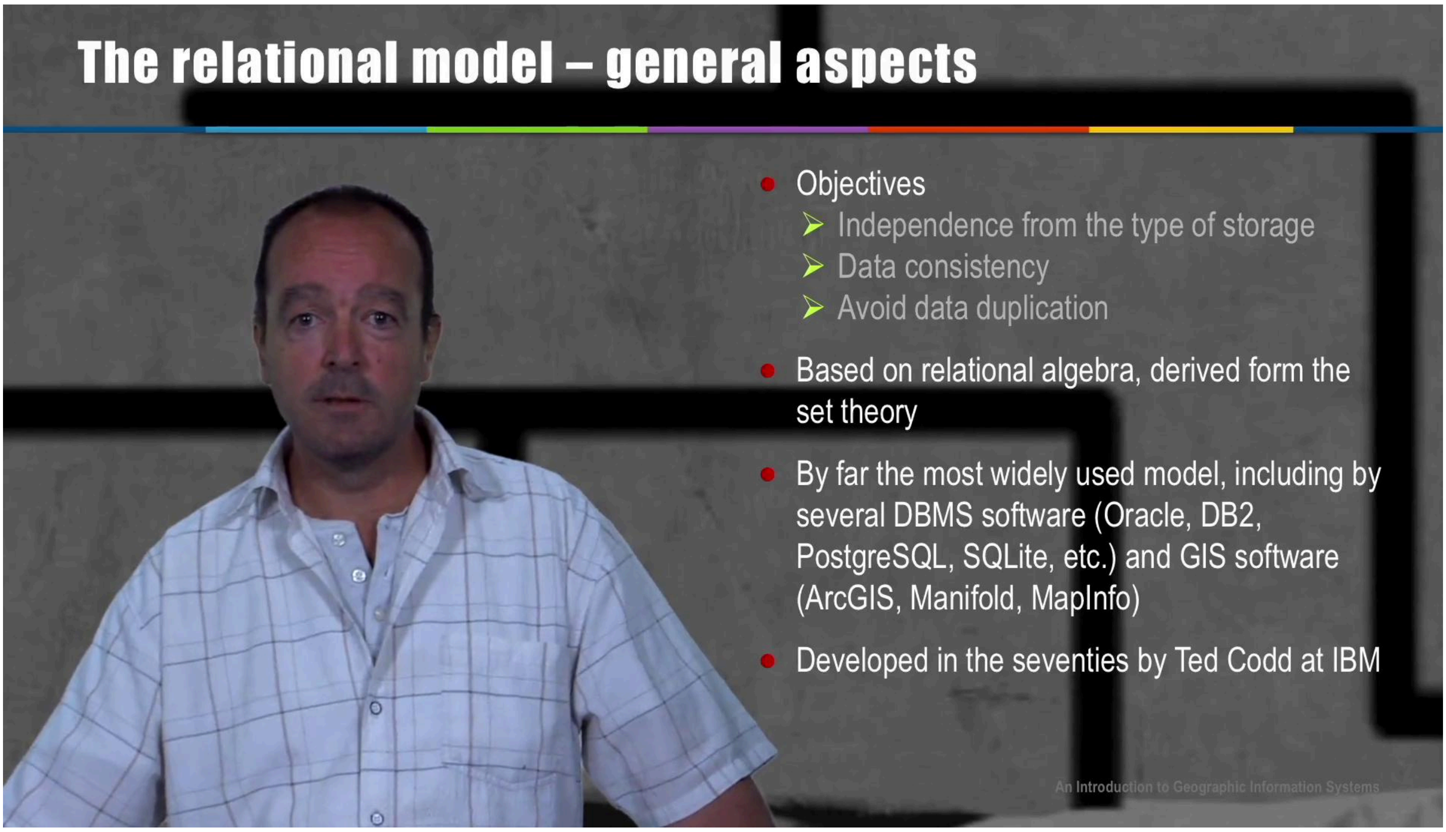


We now move on to the historical typology of databases. From the 50's and 60's storage in the form of files has developed, we saw some examples in the previous lesson, then, from the 60's-70's first there was the development of hierarchical databases. Recordings are associated by relations according to a descending tree. Each element has one and only one parent. In the fields of application, organizational structures, file systems, taxonomic systems, etc. Subsequently the network databases which are a variant of the hierarchical databases with just a multiplicity of possible parents which tree structure is no longer strictly descending. There can be cyclical structures. From the 70's-80's relational databases which propose on the principle of recordings hosted in two-dimensional tables. A relationship being a slab, an attribute in a column or field and the object in the lines. From the 90's the object databases and semi-structured databases in which the data are stored in the form of objects which can have specific and variable structures and of different types. We will speak more precisely in this lesson of relational databases and the object databases theme will be addressed more specifically in a later lesson.

Notes

Summary

7m 24s

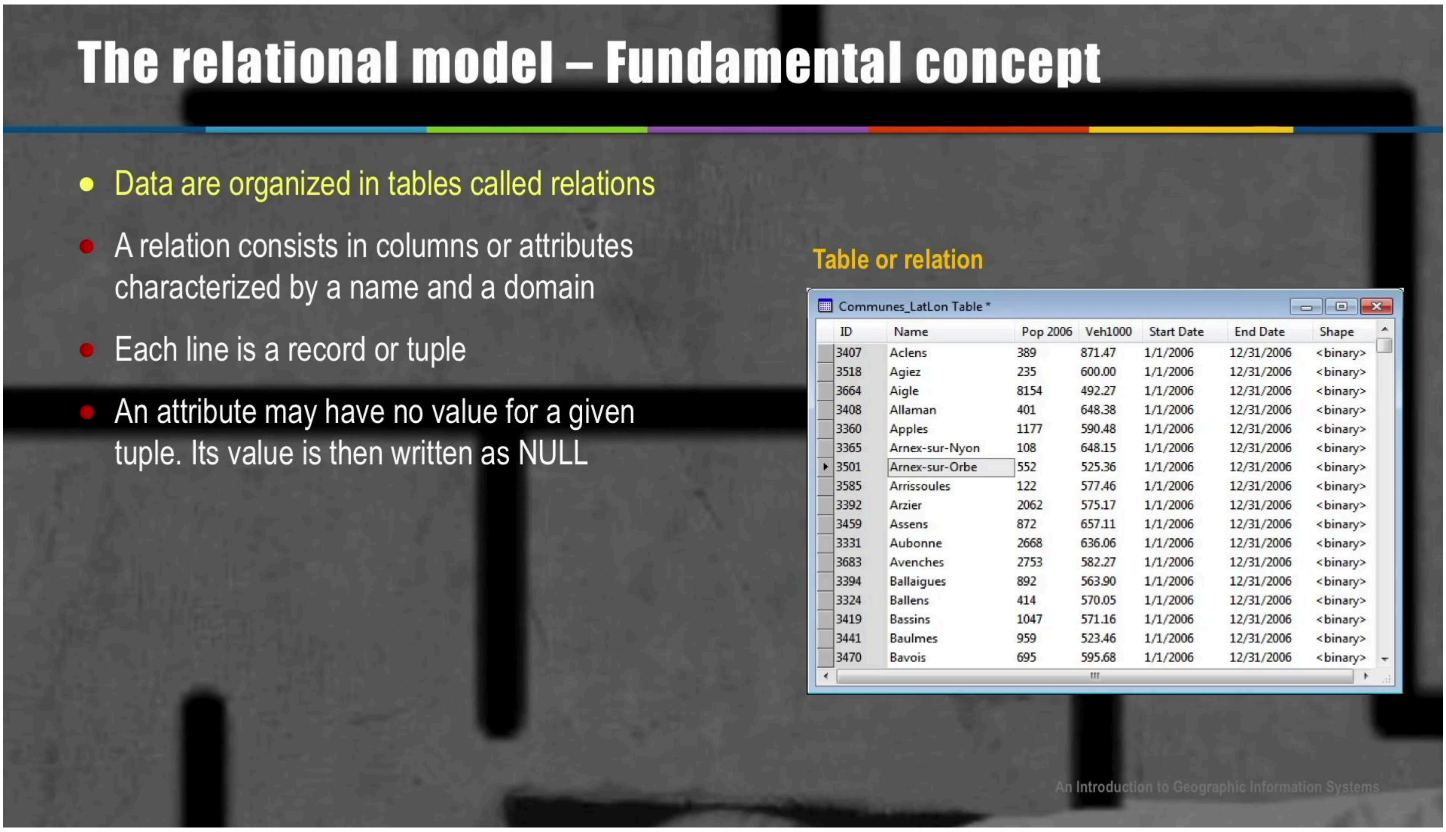


The relational model was developed mainly to meet ACID principles in the transactions that allow to modify a database to ensure that there is good coherence of information, to avoid redundancy of information. It is a model that is based on solid theoretical foundations since this is based on the set theory from which relational algebra is derived. It is a model that was widely used by all the major players in the field of databases. Today, we can say that about 80 to 90% of the databases are built on the relational model. It is something that changes a little bit with the Big Data which uses other types of models. But the relational model remains largely the most important. It is a model that was developed in the 70's at IBM by an engineer named Ted Codd.

Notes

Summary

9m 05s



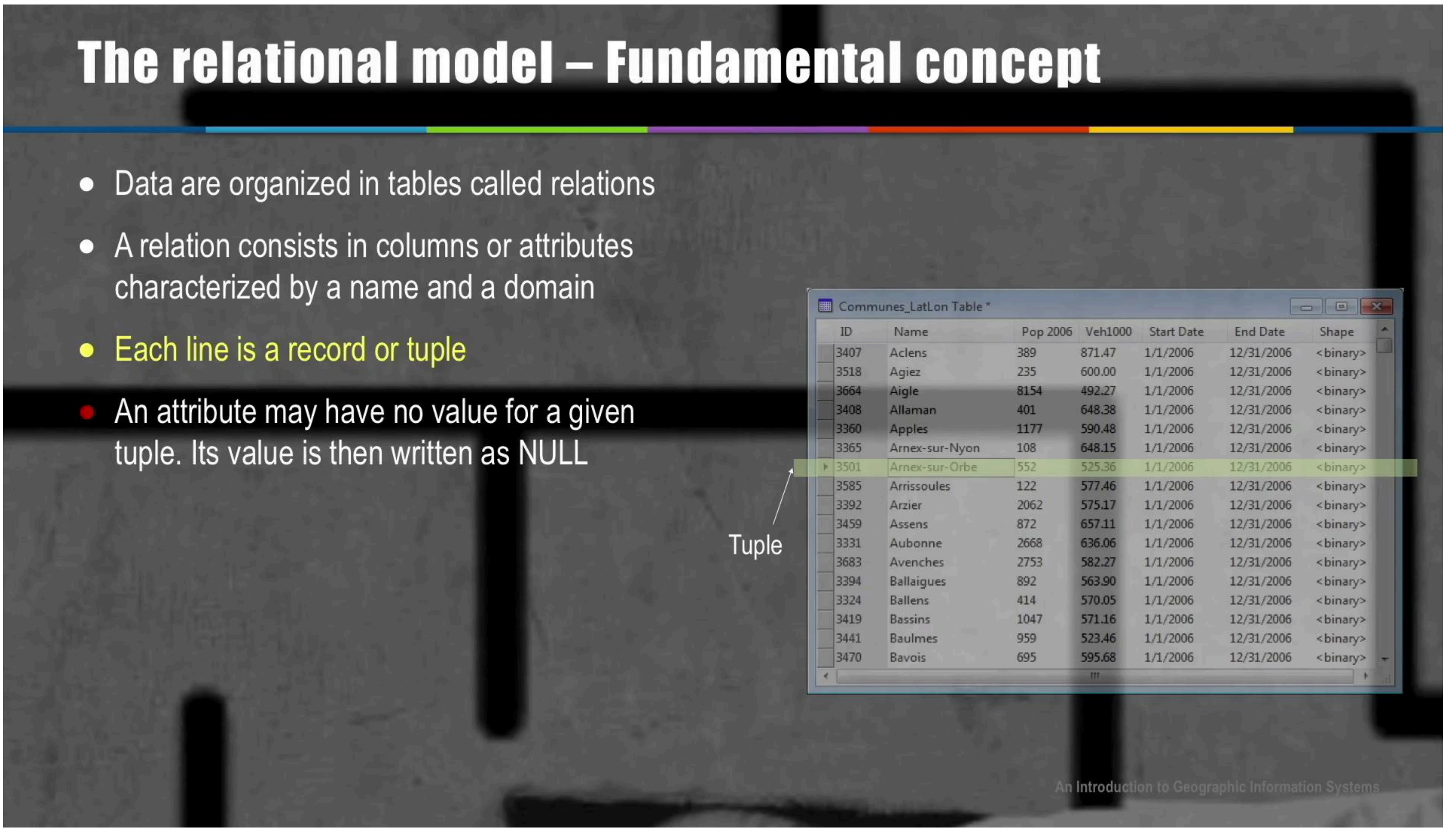
ID	Name	Pop 2006	Veh1000	Start Date	End Date	Shape
3407	Aclens	389	871.47	1/1/2006	12/31/2006	<binary>
3518	Agiez	235	600.00	1/1/2006	12/31/2006	<binary>
3664	Aigle	8154	492.27	1/1/2006	12/31/2006	<binary>
3408	Allaman	401	648.38	1/1/2006	12/31/2006	<binary>
3360	Apples	1177	590.48	1/1/2006	12/31/2006	<binary>
3365	Arnex-sur-Nyon	108	648.15	1/1/2006	12/31/2006	<binary>
3501	Arnex-sur-Orbe	552	525.36	1/1/2006	12/31/2006	<binary>
3585	Arrissoules	122	577.46	1/1/2006	12/31/2006	<binary>
3392	Arzier	2062	575.17	1/1/2006	12/31/2006	<binary>
3459	Assens	872	657.11	1/1/2006	12/31/2006	<binary>
3331	Aubonne	2668	636.06	1/1/2006	12/31/2006	<binary>
3683	Avenches	2753	582.27	1/1/2006	12/31/2006	<binary>
3394	Ballaigues	892	563.90	1/1/2006	12/31/2006	<binary>
3324	Ballens	414	570.05	1/1/2006	12/31/2006	<binary>
3419	Bassins	1047	571.16	1/1/2006	12/31/2006	<binary>
3441	Baulmes	959	523.46	1/1/2006	12/31/2006	<binary>
3470	Bavois	695	595.68	1/1/2006	12/31/2006	<binary>

In the relational model the data are organized in the form of tables also called "relation".

Notes

Summary

10m 03s



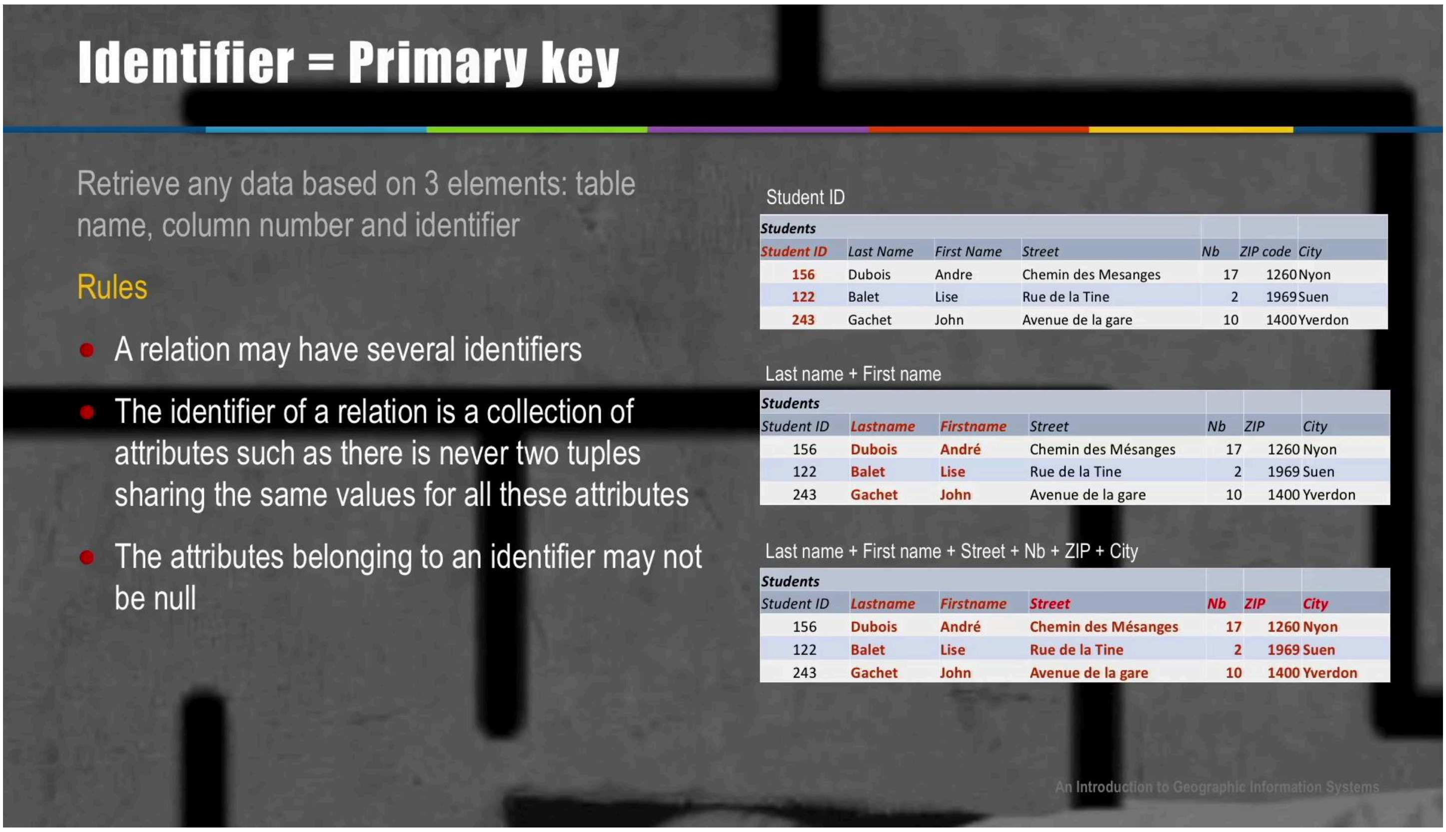
ID	Name	Pop 2006	Veh1000	Start Date	End Date	Shape
3407	Aclens	389	871.47	1/1/2006	12/31/2006	<binary>
3518	Agiez	235	600.00	1/1/2006	12/31/2006	<binary>
3664	Aigle	8154	492.27	1/1/2006	12/31/2006	<binary>
3408	Allaman	401	648.38	1/1/2006	12/31/2006	<binary>
3360	Apples	1177	590.48	1/1/2006	12/31/2006	<binary>
3365	Arnex-sur-Nyon	108	648.15	1/1/2006	12/31/2006	<binary>
3501	Arnex-sur-Orbe	552	525.36	1/1/2006	12/31/2006	<binary>
3585	Arrissoules	122	577.46	1/1/2006	12/31/2006	<binary>
3392	Arzier	2062	575.17	1/1/2006	12/31/2006	<binary>
3459	Assens	872	657.11	1/1/2006	12/31/2006	<binary>
3331	Aubonne	2668	636.06	1/1/2006	12/31/2006	<binary>
3683	Avenches	2753	582.27	1/1/2006	12/31/2006	<binary>
3394	Ballaigues	892	563.90	1/1/2006	12/31/2006	<binary>
3324	Ballens	414	570.05	1/1/2006	12/31/2006	<binary>
3419	Bassins	1047	571.16	1/1/2006	12/31/2006	<binary>
3441	Baulmes	959	523.46	1/1/2006	12/31/2006	<binary>
3470	Bavois	695	595.68	1/1/2006	12/31/2006	<binary>

A relation is made up of columns or attributes characterized by a name and a domain, a domain being a set of values, for example the domain of integer values, the boolean domains, the sport discipline domaine, etc. In a table, each line corresponds to a recording also called "tuple".

Notes

Summary

10m 13s

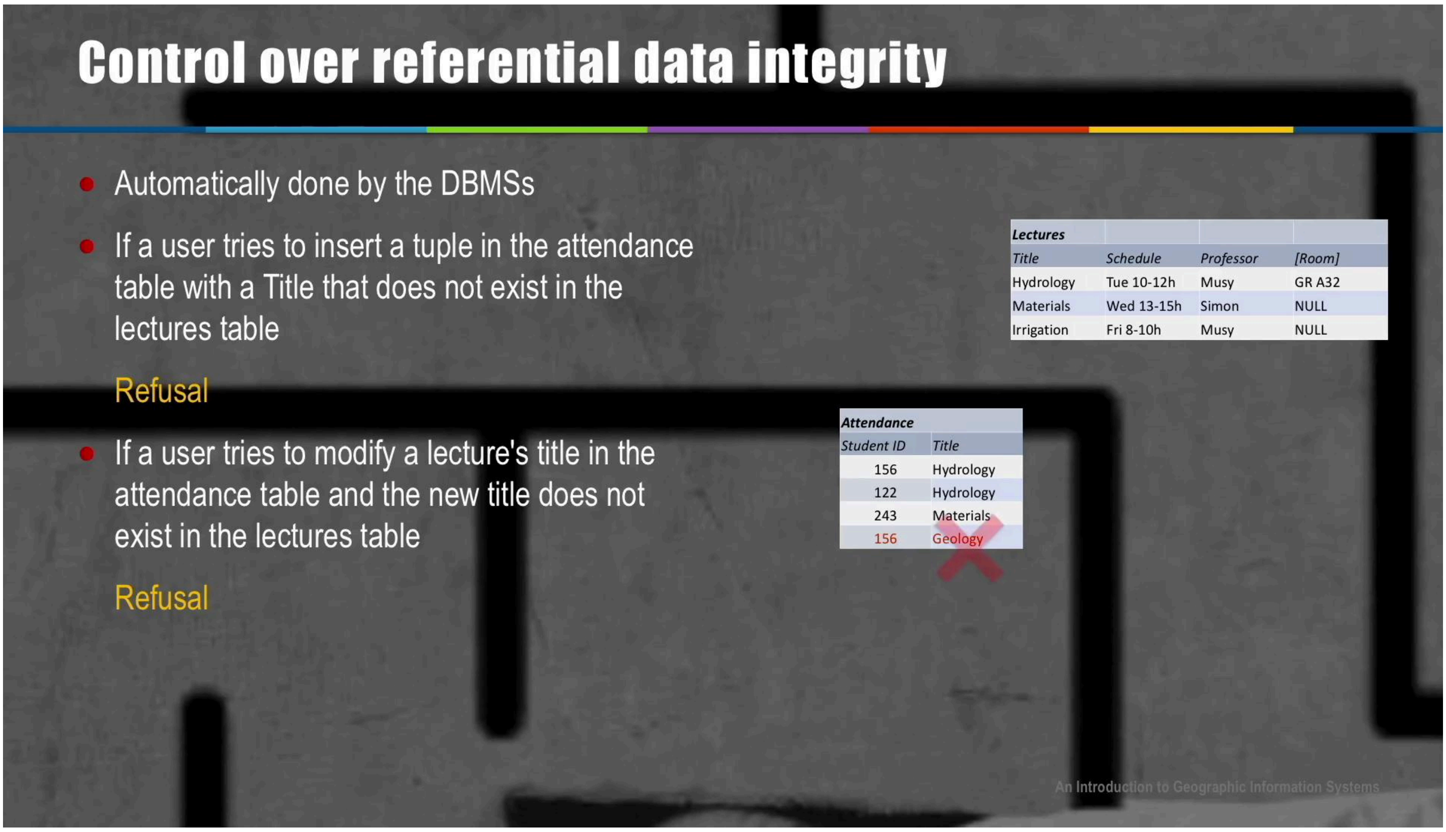


For a tuple it can happen that an attribute is not valued, so it has no value and in this case, we indicate this fact by the value NULL. A set cannot contain the same object twice and it is for this reason that in the relational model we must absolutely ensure that each object is unique. The best way to ensure that this object is unique is to define an identifier specifically for the needs such as in the case of the first table that we see here where we created a student number which allows to unambiguously identify each student. This identifier, sometimes called "primary key" can be manufactured for the needs of the cause but can also be built from existing attributes for example by associating the names and surnames of the students. We clearly see that if we only use the surname and first name we can be faced with homonym problems and have several people who could not be distinguished by this identifier. Hence the need to extend the concept to include the address, the street name the postal number, etc. In fact, the identifier could consist of the set of fields of the table with obviously a number of disadvantages when it comes to indexing objects in order to find them faster in a search or be able to sort things out or things like that. It is for this reason that the use of a specific identifier is the most common case encountered.

Notes

Summary

10m 35s

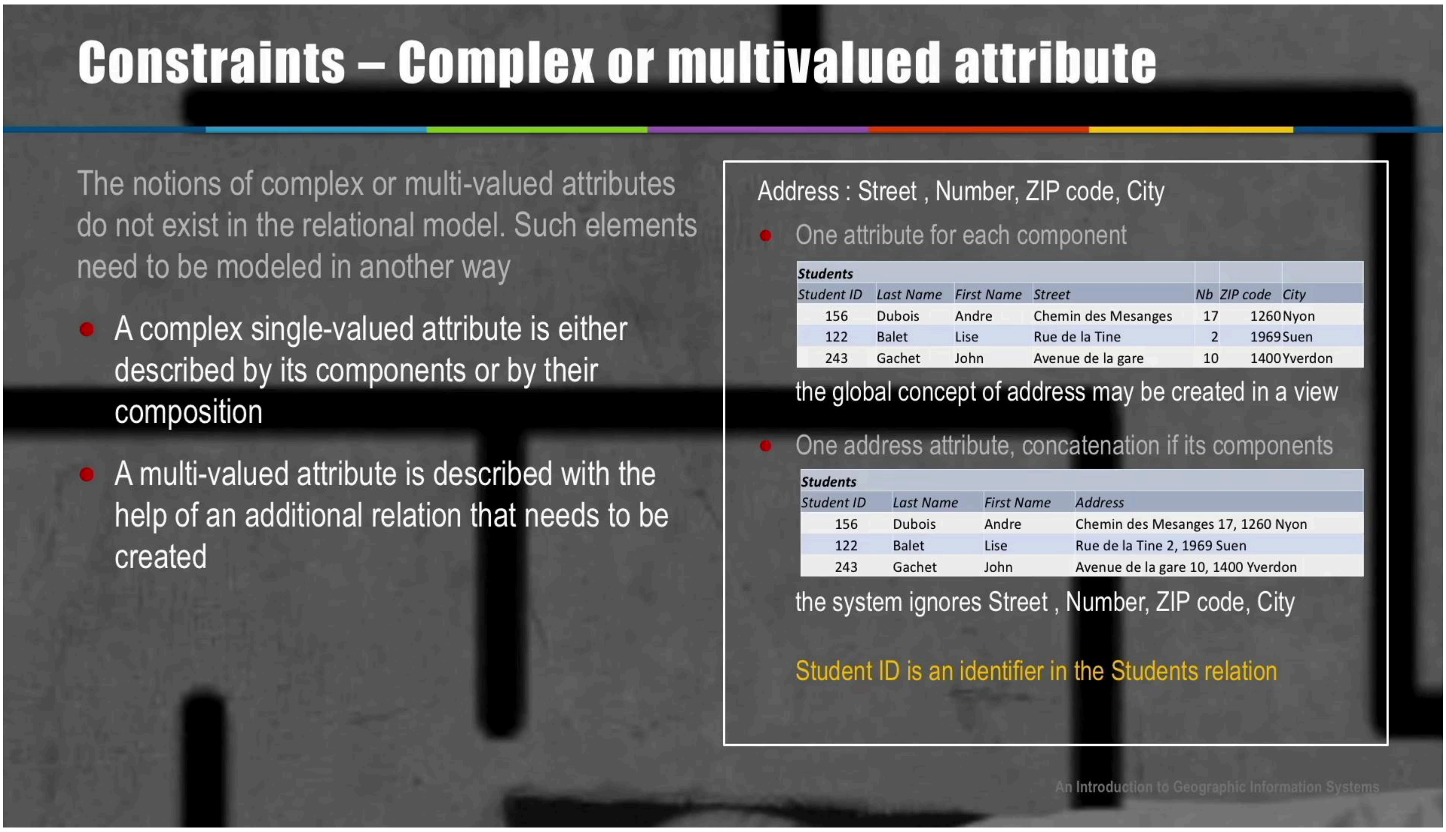


External identifiers, also known as "foreign keys", describe links between different relations, for example, in the case of a course followed by students we have a table of students, a table of courses, a table of teachers and we see that a course is given by a teacher and that this teacher must exist in the teacher's table for the database to be integral. In the course tracking (?) relation which associates a student number with a course title, so the list of courses followed by the students or the list of students following a course, we see that the student number references a student of the student relation and "course title" references a course in the course table. The reference should point to a single object of course and it can happen that this object is NULL if the attribute is optional, for example in the case of the classroom which could not have been assigned yet at the time of documenting the table. The referential integrity of the database is automatically verified by the database management system. For example, if a user wants to ensure a new course of geology in the table of courses followed by students and this course does not exist in the course table, the transaction will be denied.

Notes

Summary

12m 27s

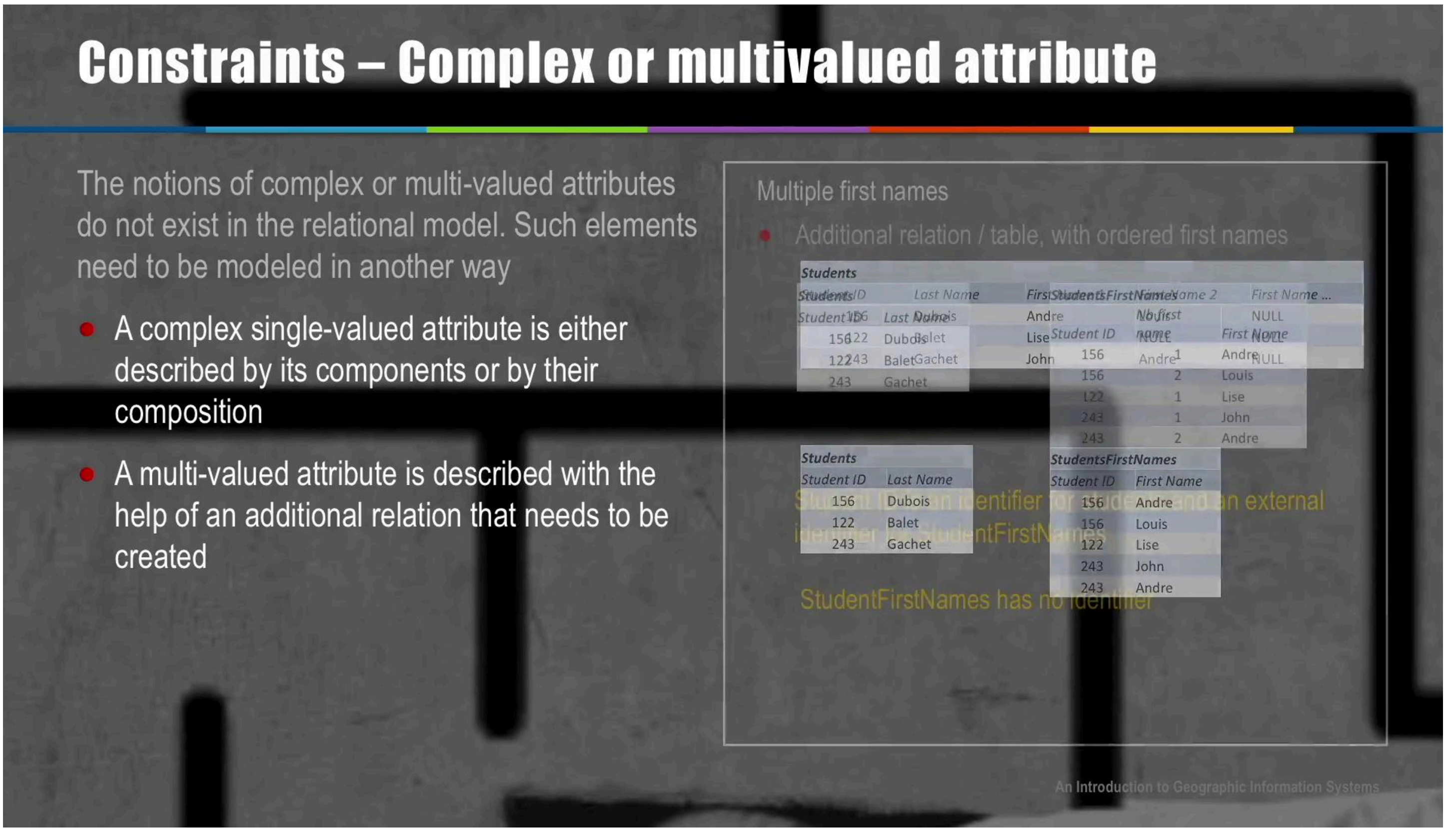


Similarly, if a user wants to change the name of a course and that name does not exist in the reference table, the transaction will also be denied. Similarly, if a user wants to delete a course which is referenced by other relations, the operation can be refused. We can delete references in other relations or alternatively cancel those references. Similarly, if a user wants to change the name of a course in a table and that name is referenced elsewhere we can refuse the operation or update the other table as well. The relational model does not lend itself to a multivalued attribute record, so that have several values or complex attributes, so composed of several elements. It is therefore necessary to model them differently. In the case of a complex monovaluated attribute, such as an address which is composed of a street, a street number, a postal number, a locality, one possibility consists in setting one attribute, one field per component as shown by the table on the right and subsequently to build in the database a view, which is a virtual table containing the aggregation of these different fields to restore the idea of address.

Notes

Summary

13m 53s

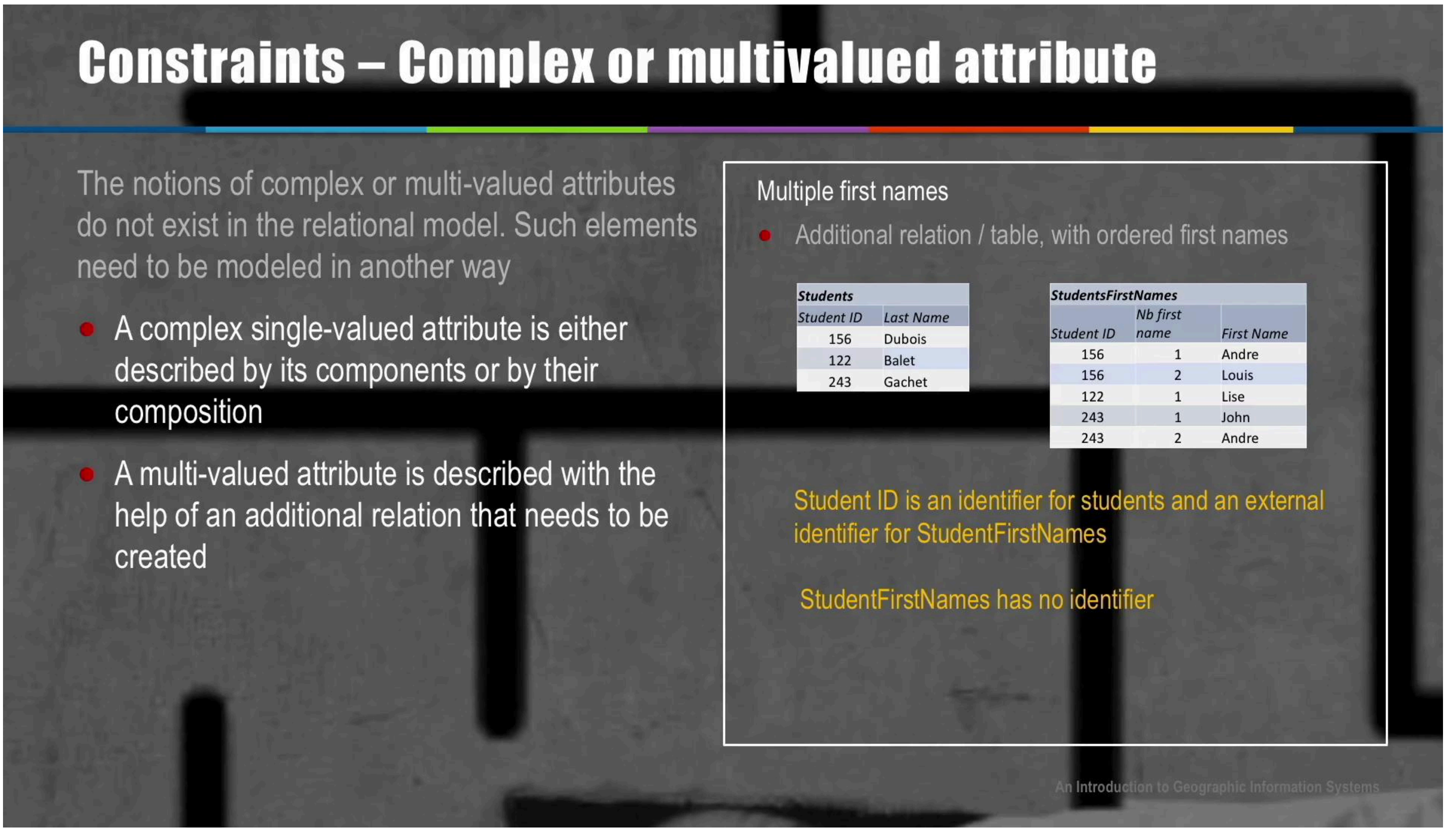


The other possibility consists in creating a "global address" attribute which would be of a character string type in which the entire address, road, number, postal code, town, is saved. We see that in these relations the student number is an identifier of the student relation. For a multivalued attribute, for example the case of multiple first names, one possibility would be to define several attributes for each first name, first name 1, first name 2, etc. This is a bad choice because we do not know how many first names we need to plan, if it is not a first name it can be another type of attribute which could have an innumerable succession of elements so we see that it does not work very well. Especially since this leads to defining many fields which will be filled with null values. This is not something that is very optimal from the database point of view. An alternative solution is to create an additional table in which first names will be hosted in relation to the student identifier to which these first names are related and so we see that the "student number" field, which is a primary key of the student table becomes a foreign key of the "study first name" table.

Notes

Summary

15m 27s

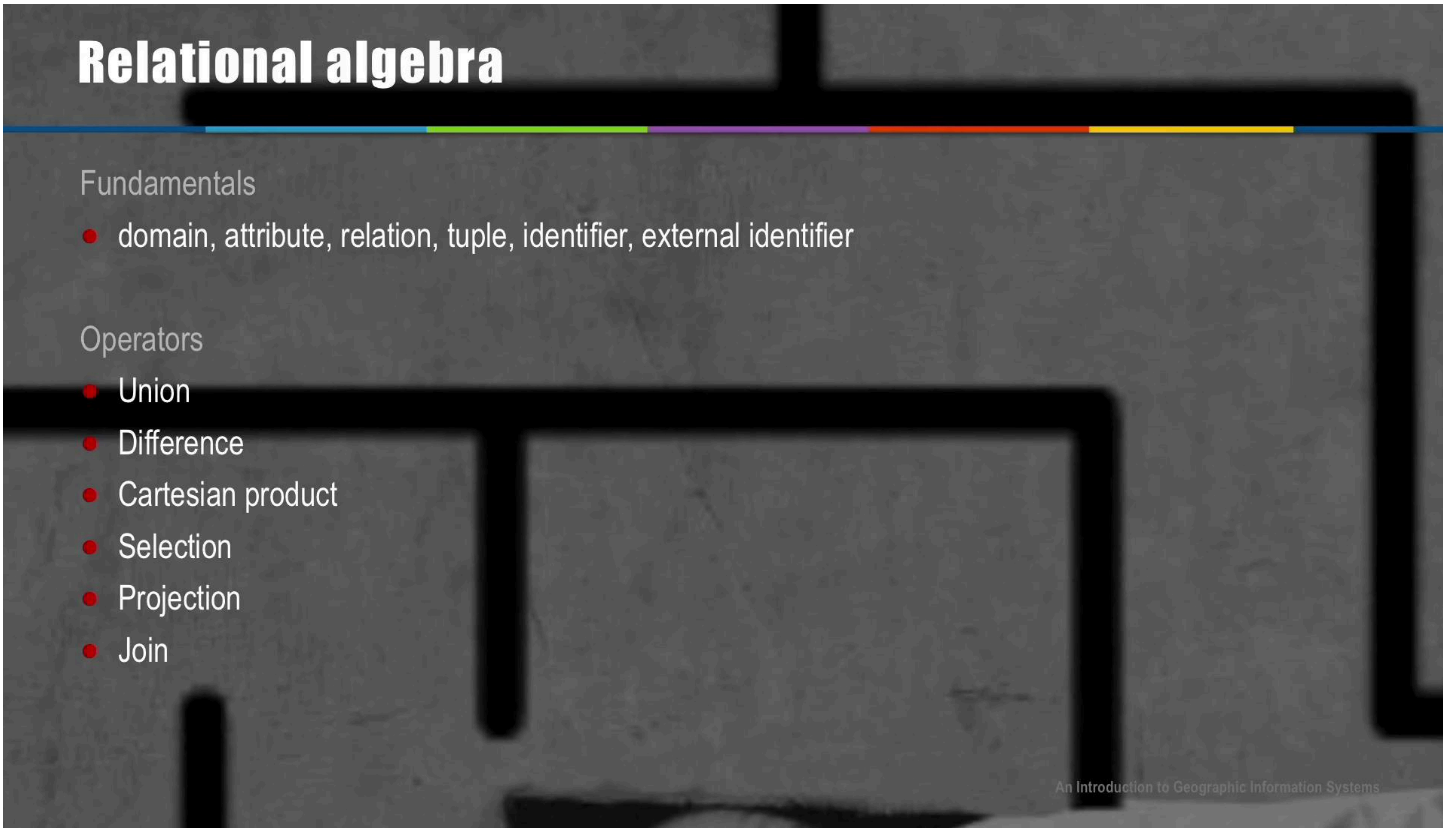


Another possibility, a little more subtle, consists in integrating into the "study first name" table the number of the first name so that we are able to restore the order in which first names appear, first name, middle name, etc. In this case too, the student number is a primary key for the "student" table and a foreign key for the "study first name" table which does not have a primary key.

Notes

Summary

16m 59s

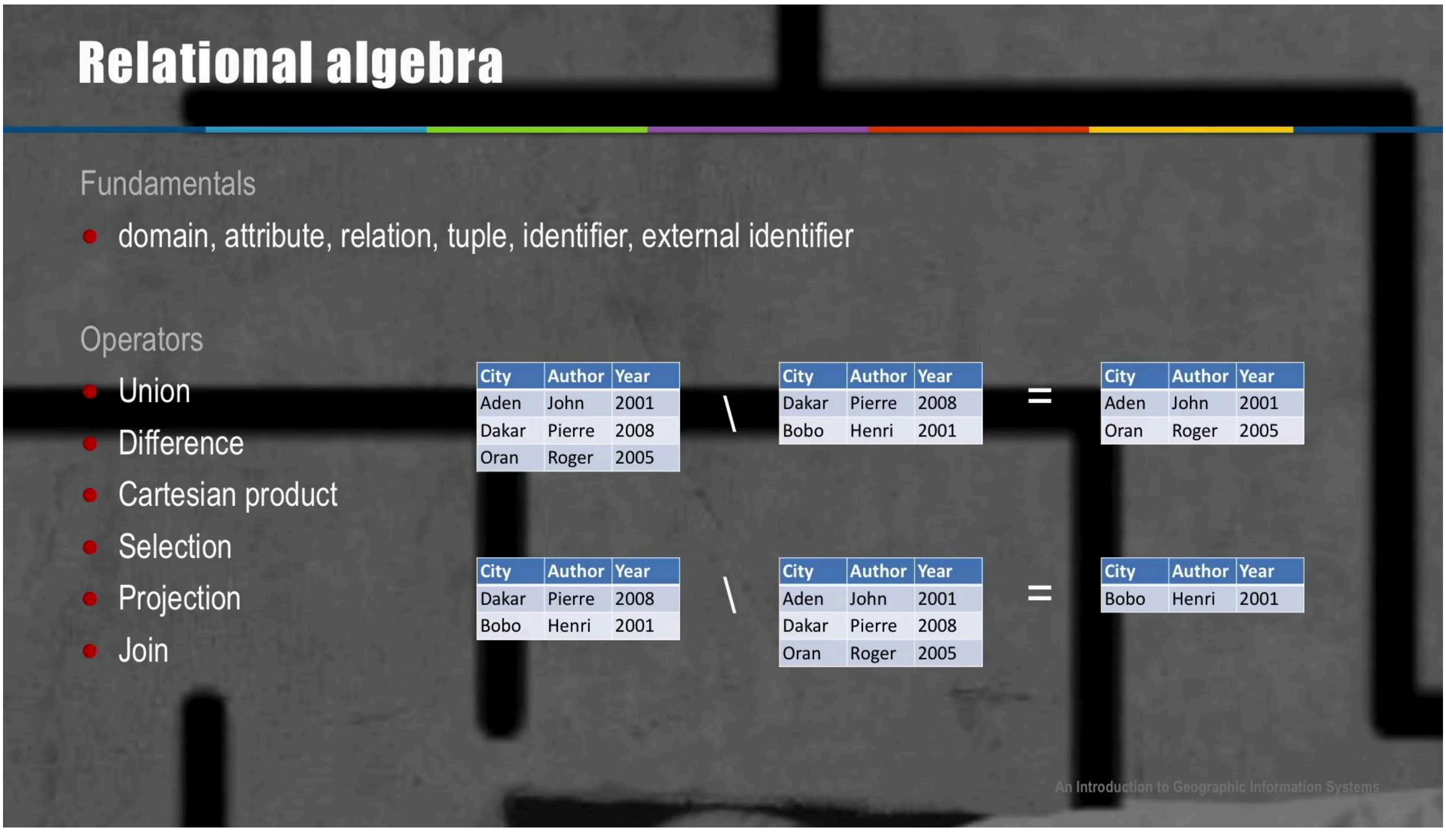


So, we have reviewed the basic notions of the relational model namely the domain of values the attribute, the relation, the tuple the identifier, the external identifier. We still have to discuss the various operators of relational algebra that base the operations which we will subsequently be able to make on these tables, operations which we will discuss in details in the second week of this module with everything that is request, SQL language, etc.

Notes

Summary

17m 28s

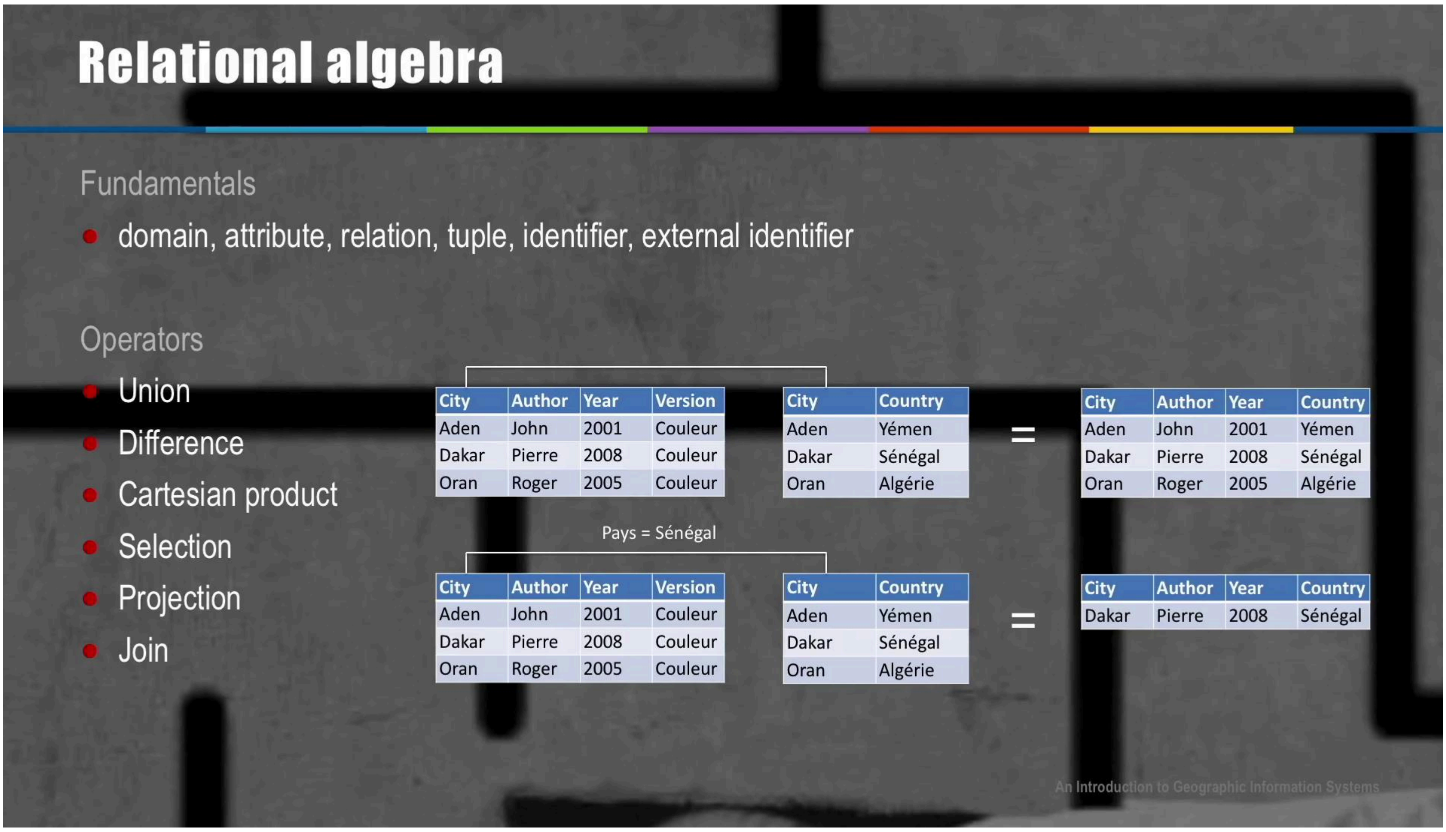


These operators are primarily union operators so we are in a set logic. For the example, we take here a series of photographs which were taken by an author a certain year in a certain city and we see that the union of the two tables consists in aggregating these two tables into one. The difference between two tables is the subtraction of the second o the first so the elimination of the redundant objects of the first. This operation is not commutative, if done in the other direction the result will be different.

Notes

Summary

18m 03s

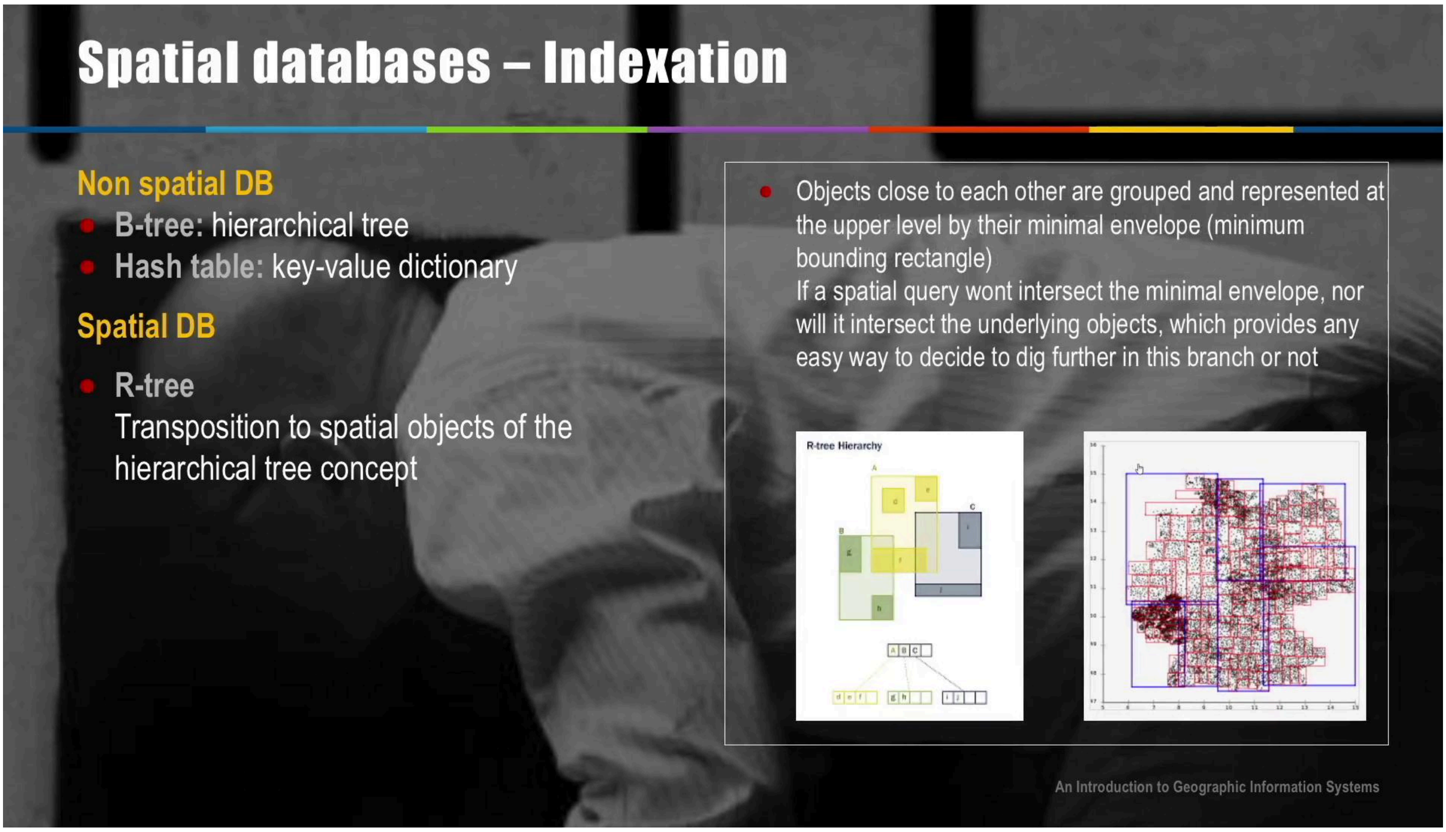


We then have the cross product which consists in associating each element each tuple from the first table to each tuple of the second. So here the colored and black / white versions of the photos which... so we have 2 times 3, 6 pictures. And then the selection operations on a table, a selection which consists in identifying, extracting a certain number of elements. Projection operations which consist in extracting a certain number of attributes for the whole table. Which in this case leads to something that does not make much sense in any case which is contrary to the principles of the set theory since we have identical objects. So it is an operation that would not be valid. And finally the joint operations which consist in associating two tables by means of a field which would be common to them. Here, the table of color photographs with a table that would associate cities and countries. The common field being the city obviously. These joint operations can be assorted by a condition, so we could keep only the joint elements which have Senegal as their country and so only one photograph of the three that were included in the starting joint.

Notes

Summary

18m 58s

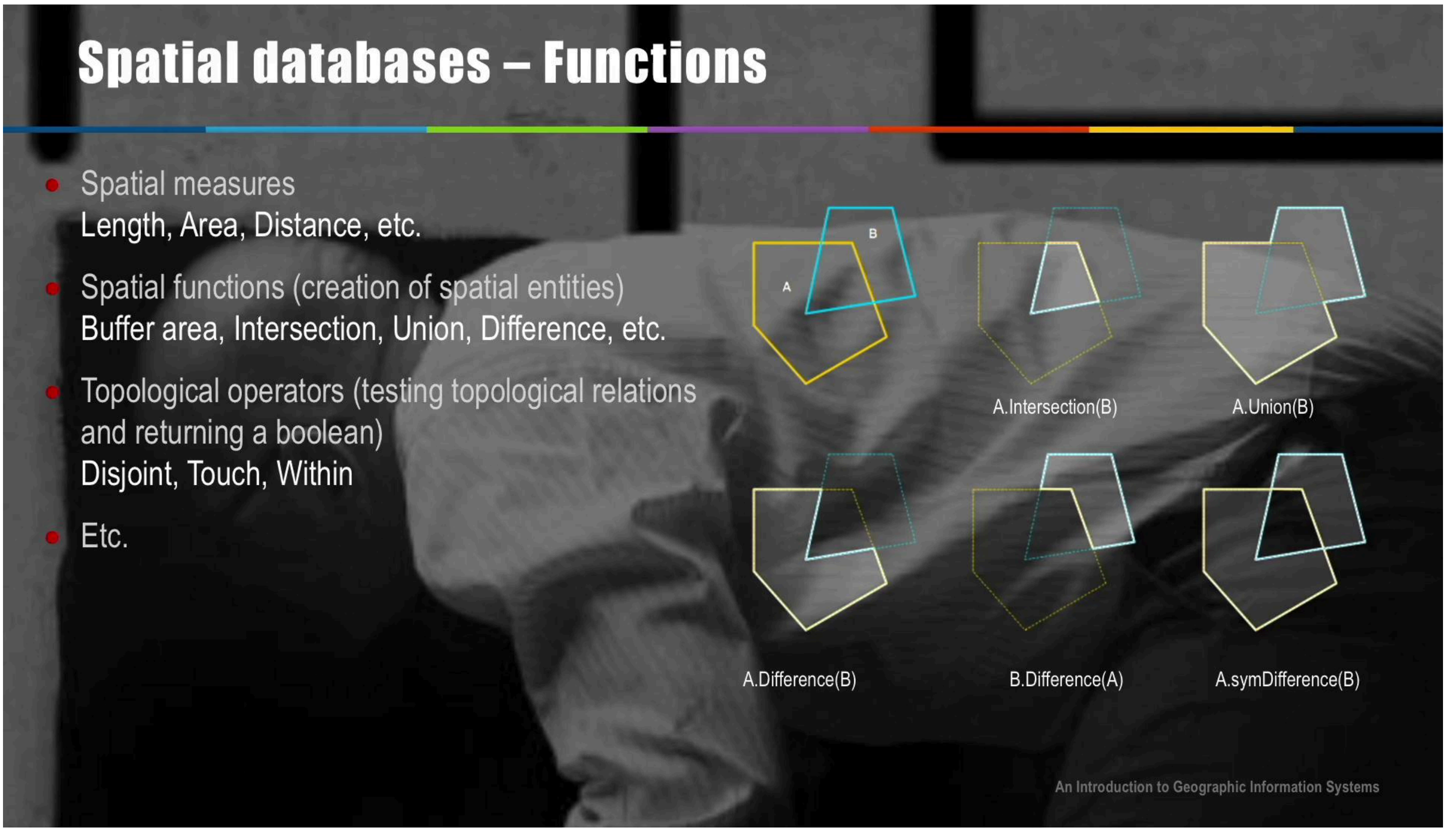


We now come to the specificities of databases that have a spatial component, specificities which are mainly three in number. First, the type of data so the fields concerning these types of data we see that in the top left table, in a traditional database we have a certain number of well-defined data types, varchar for the text, integers for digital, the real also for numbers and dates. In the spatial database we add new data types, point, line, polygon, etc. Finally, the basic geometries which are managed by geographic information systems. The second important area is that of indexing the indexing of spatial objects to be able to subsequently perform requests and quickly find objects and finally a number of functions which allow to carry out specific operations on geometric objects. There are several forms of data indexing. In the field of non-spatial databases we often use a structure called B-tree so a hierarchical tree which makes it easy to find the data. We have in this illustration the example of a tree of the order 5 wherein a knot can have at most 4 keys and 5 children each knot having at least 2 keys and 3 children.

Notes

Summary

20m 44s

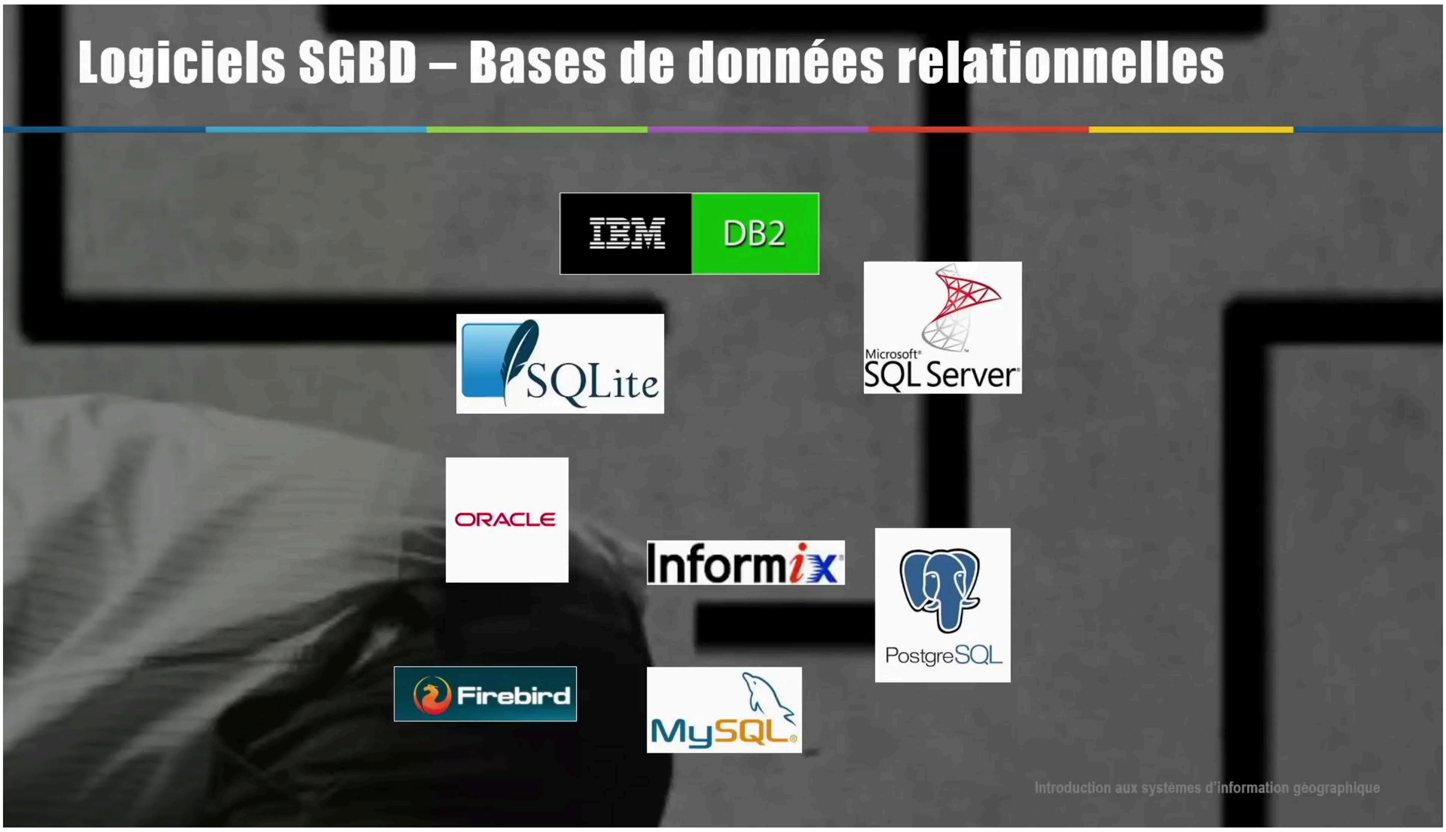


This hierarchical tree idea is found in the spatial database field in the R-tree form in which the closest objects are grouped and represented at the upper scale by a minimal envelope which is the minimum bounding rectangle. And when we then make a request to find the objects if the request does not intersect a minimal envelope we can eliminate all the objects of this envelope and focus on those that remain. In the image here, we see on the left the link between a non-spatial hierarchical approach and a spatial hierarchical approach and then on the right an example of spatial index structuring which concerns post offices in Germany. The Quadtree is another method of spatial indexing which is quite often used in geographical map tiling like Google, etc. It is about a tree structure in which each knot has exactly 4 children so each geographical zone is divided into 4 and again in 4 each time we zoom down. The grid index mode is similar in the sense that it is also a regular tessellation but the subdivision of each entity is not necessarily done in 4 but can be done in 9, 16, etc. Amongst the specifics functions of spatial databases we can distinguish three main families.

Notes

Summary

22m 32s

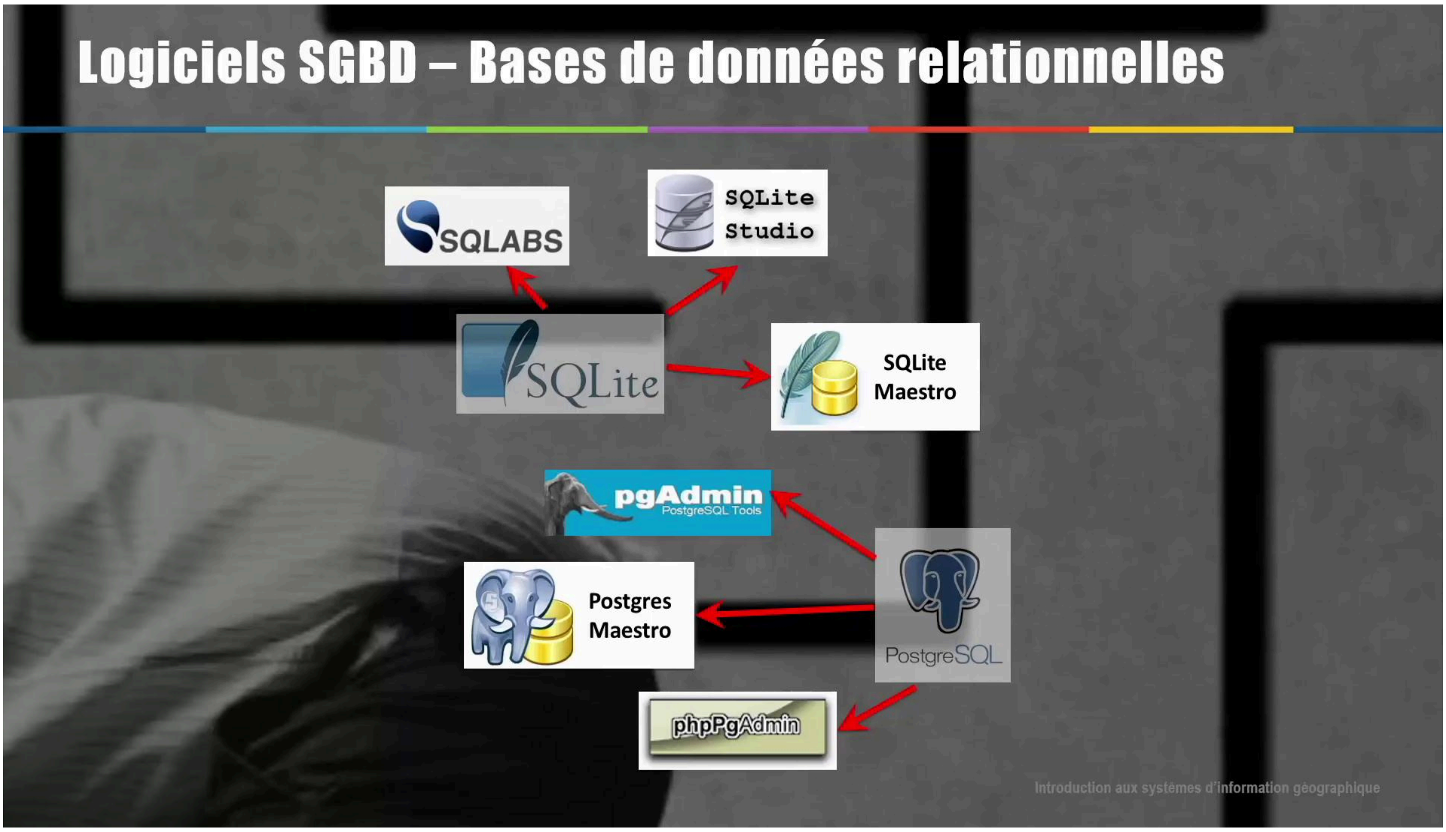


First of all, the family of spatial measurements which gives indications of surface length, distance about geometric objects. Spatial functions that create new entities, for example we can imagine the creation of a buffer zone around a geometric object. The intersection of two objects that creates a new object, the union of two objects, etc. And topological operators which test the veracity of the neighborhood relations. Are two geometries overlaping? Do they touch each other? Are they contained in one another? Or vice versa, etc. All these different types of functions are more and more often implemented in spatial database management systems. As I said at the beginning of this course there is a very large number of database management systems devoted to the relational model not only is there a very large number of databases but these databases are accessible through a series of clients which are either commercial softwares or open-source softwares.

Notes

Summary

24m 28s

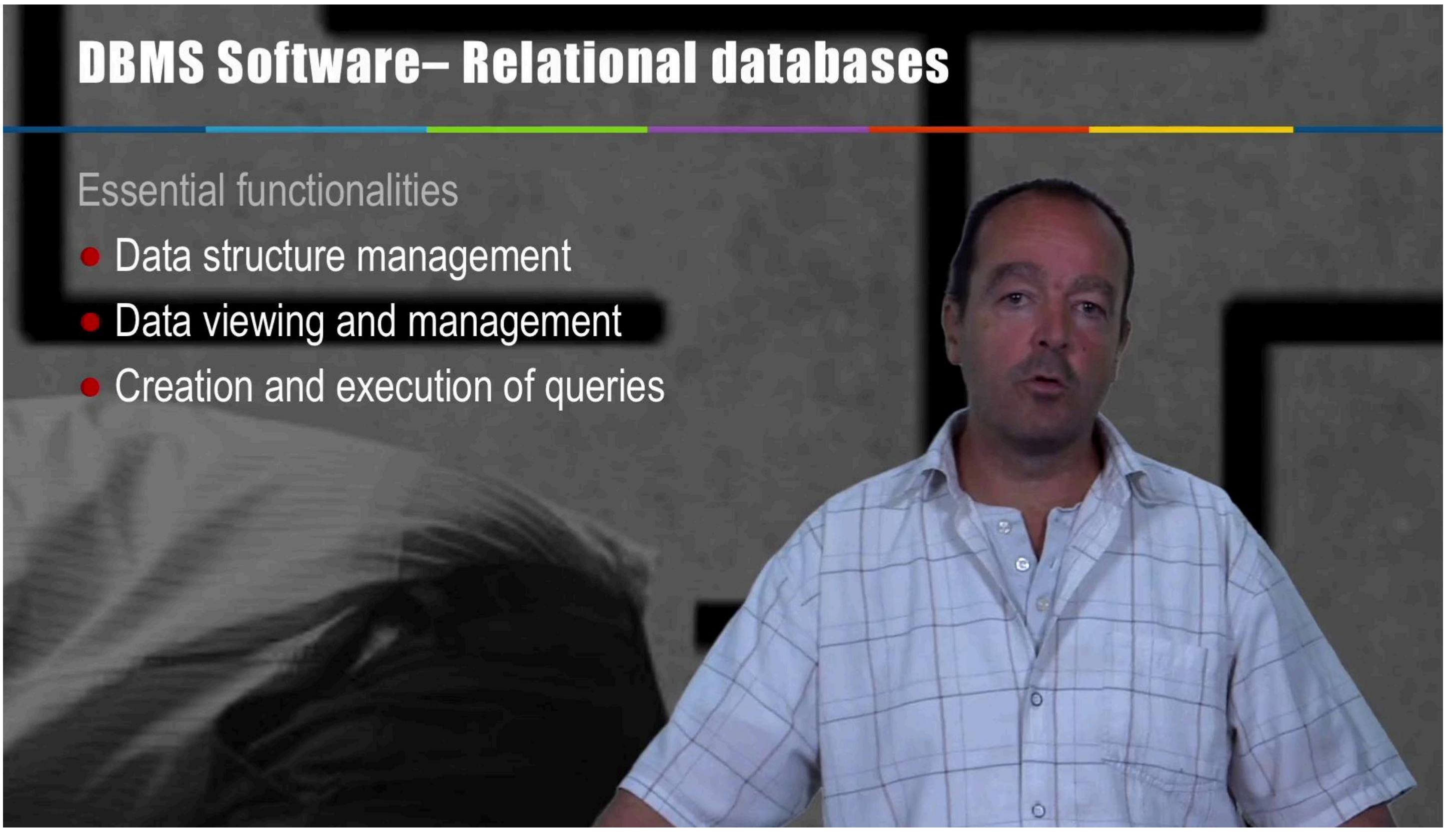


All of these clients provide a user interface and allow to contact these databases and to manipulate them.

Notes

Summary

25m 59s



Amongst the features that we want to find in database management systems, there are actually three types, first, those that allow to manage the structure of the data so to create new fields, new attributes with their area of value and create new indexes, primary keys, external keys, etc., to view data saved in the database, to consult the value tables, possibly to be able to import data from the outside to populate these different tables and finally all the features which are related to the construction and execution of queries to search for objects in a database on the basis of a certain number of criteria. This is something we will see more in details particularly in the lessons of the second week of this module.

Notes

Summary

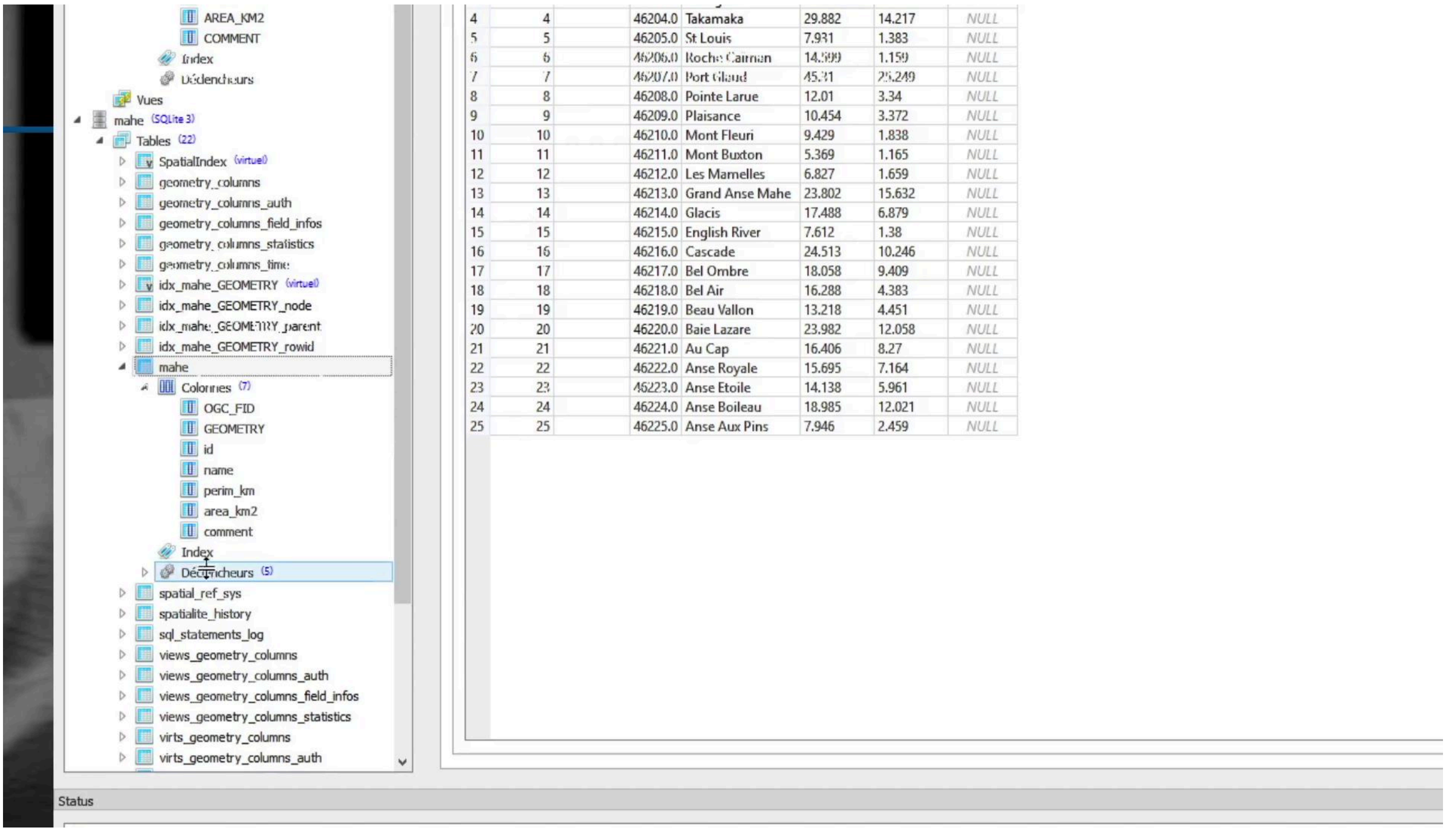
26m 08s

For now, we will take a small example which is that of the Seychelles districts that can be exported in a database format, in this case a SpatiaLite database. So we will call this file "mahe.sqlite" and save it. It is possible to specify some parameters the saved layer will be added... ah, it asks to specify the projection system used, UTM South 40. This new layer can be hidden. And as a second step we will save simply the attributes of the layer, so without the geometry, in the form of an ASCII file a.csv text file, so a text file with values separated by commas. Similarly, the table will be added to the list of available objects.

Notes

Summary

27m 04s

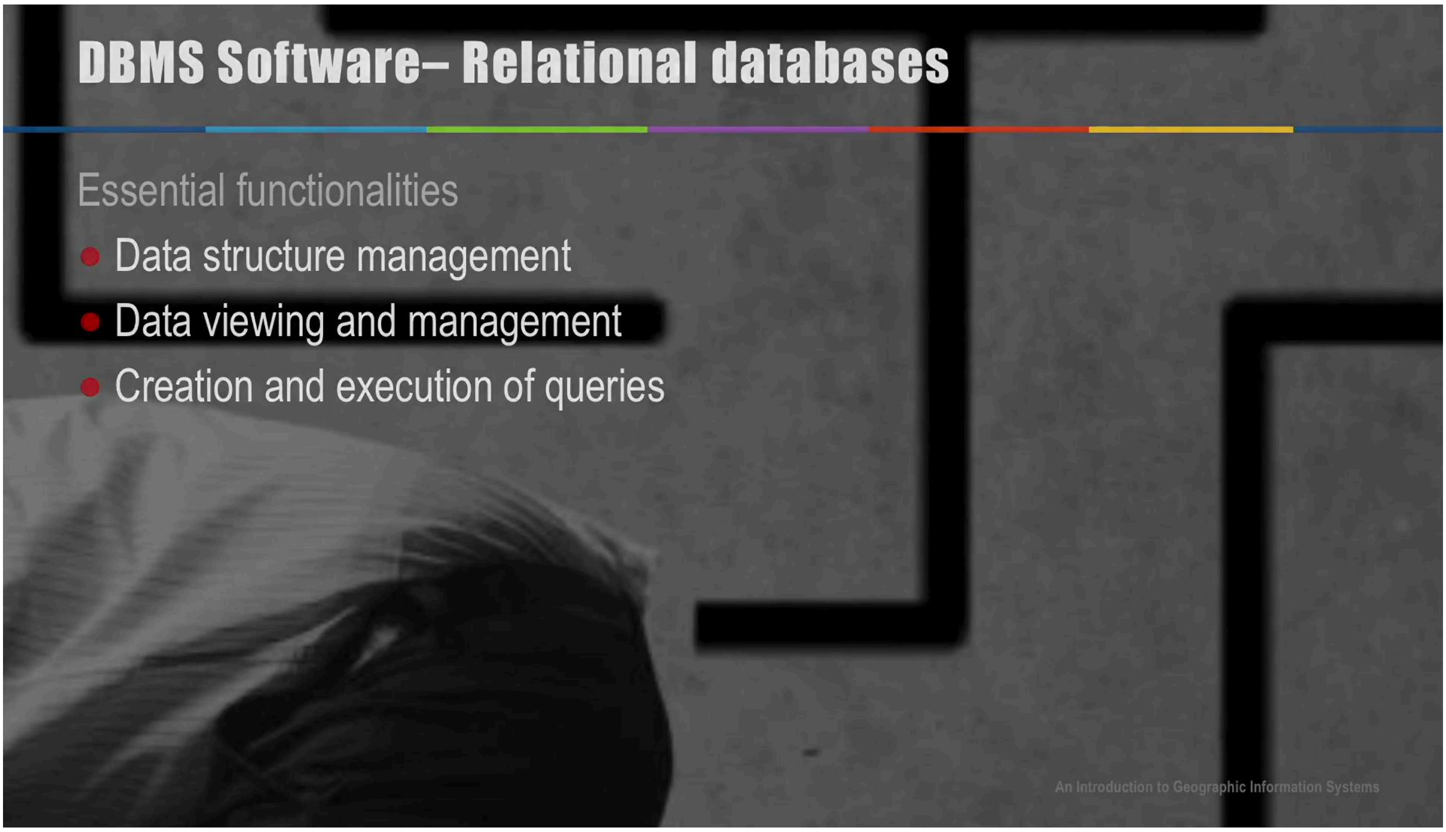


If we go now to a database management software. It is SQLite Studio. We can import. We will start by creating a database in which we will import the text file. So we will call this database "mahe_data". This database once created we will start by connecting to this database and we can see that it can contain tables and views which are in fact query results made a little permanent. In the tables, we will import data in a new table we will create this new table that will be called "district" and import the text file which contains the attributes of these districts which we previously saved. The first line is said to represent column headers and we see that this table now appears in the tree, this district table that contains a number of columns which are the different fields that we had in the QGIS district layer. We can consult the data themselves and modify the data structure if necessary. In a second step, we will add another database which is actually the backed-up QGIS database with the elements of geometry and we see that if we connect to this database we have a much larger number of objects since in fact, to manage the geometric dimension the SQLite database generates a whole bunch of objects which are necessary and which make the thing a little more complex when we approach it like that from afar.

Notes

Summary

28m 11s



But we also find the data table of the districts of Mahé with the same columns, the same values which can be consulted in the form of a table. There is no index yet but triggers which are in fact the rules that the database management system must check during each transaction. So we can define the validation rules case by case. That is it for this introduction to database management system softwares that we will see in more details in other lessons of the course.

Notes

Summary

30m 09s