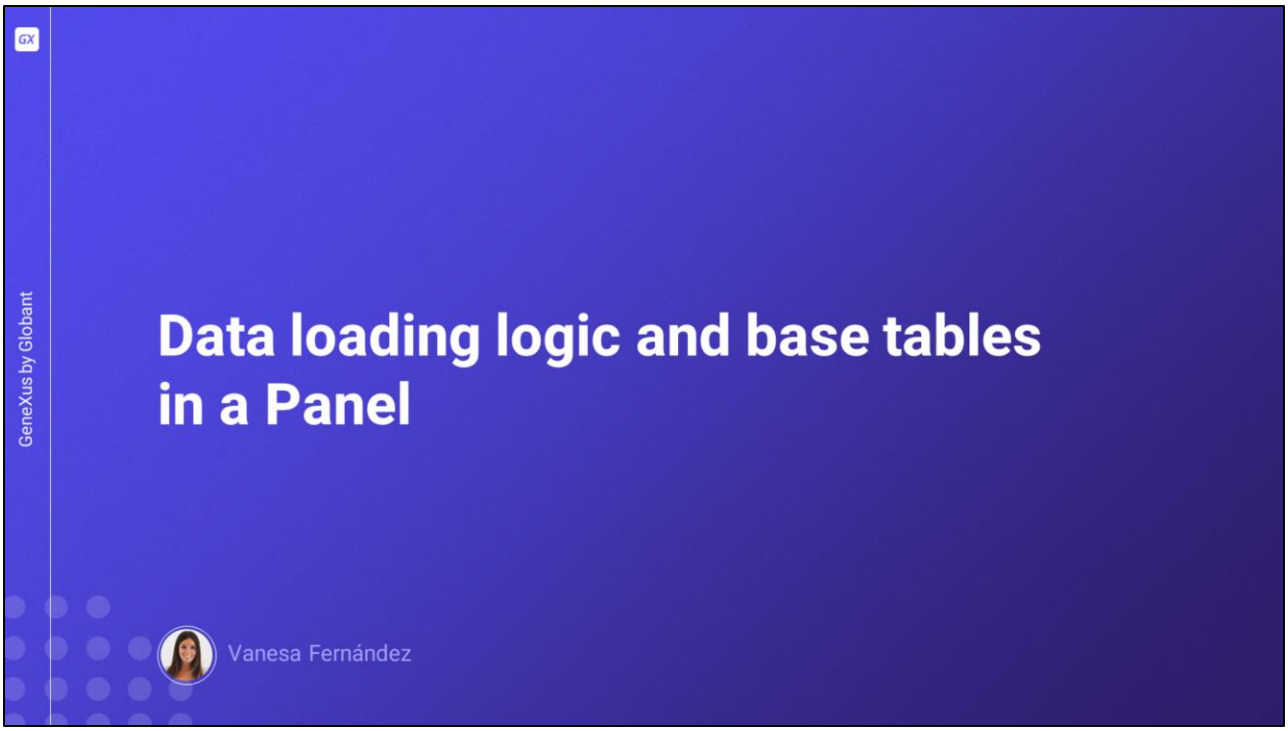
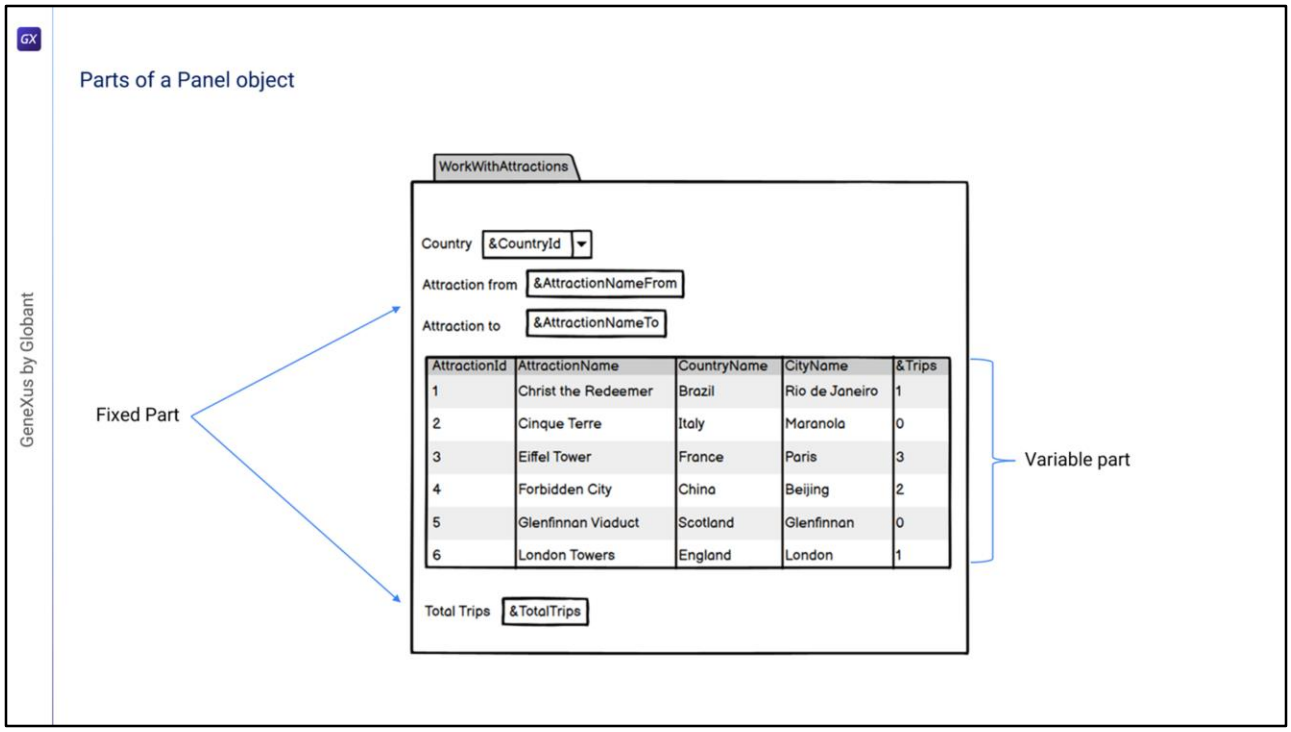


In this video, we will study how the screen of a Panel object is loaded depending on the GUI components included in it, how its base tables are determined, and how we search, sort, and filter the information on screen.



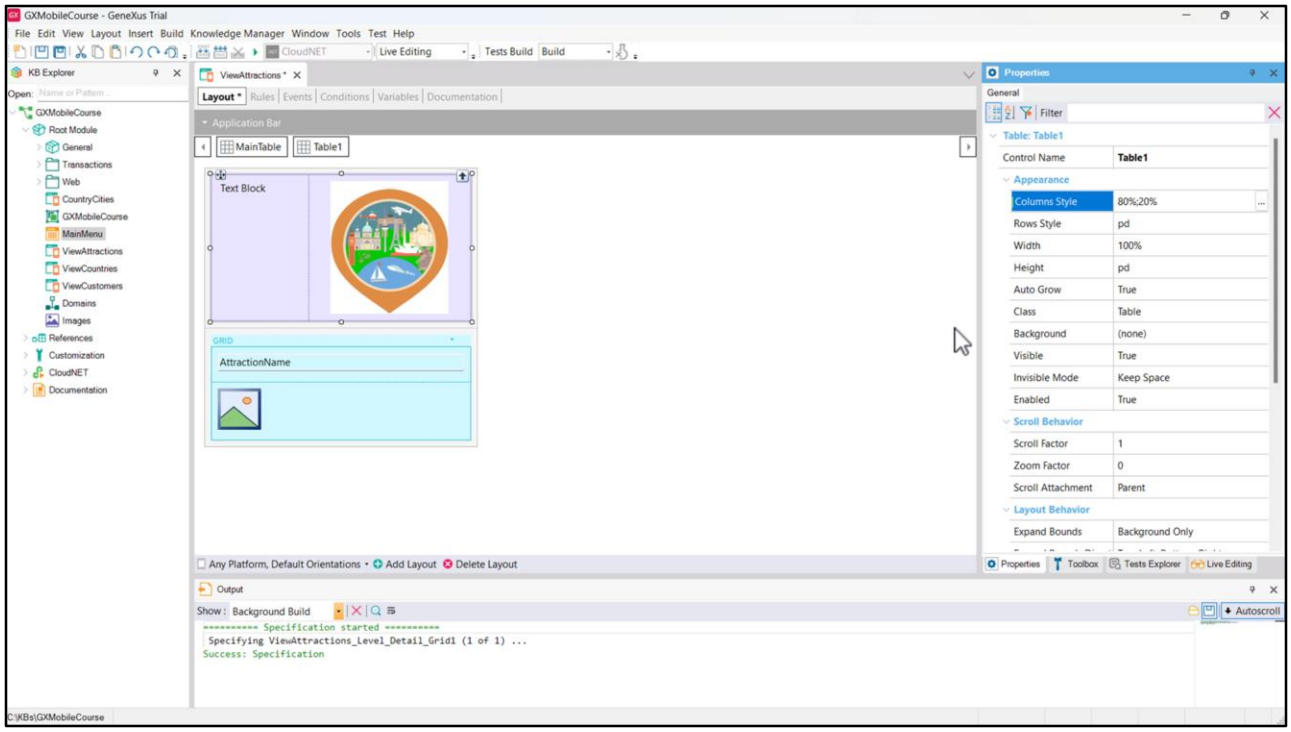
In a Panel object we can identify two distinct parts:

the fixed or flat part contains everything that is in the form and is not contained in any grid. This includes controls such as text blocks, combos, buttons and other elements that we can drag from the toolbar.

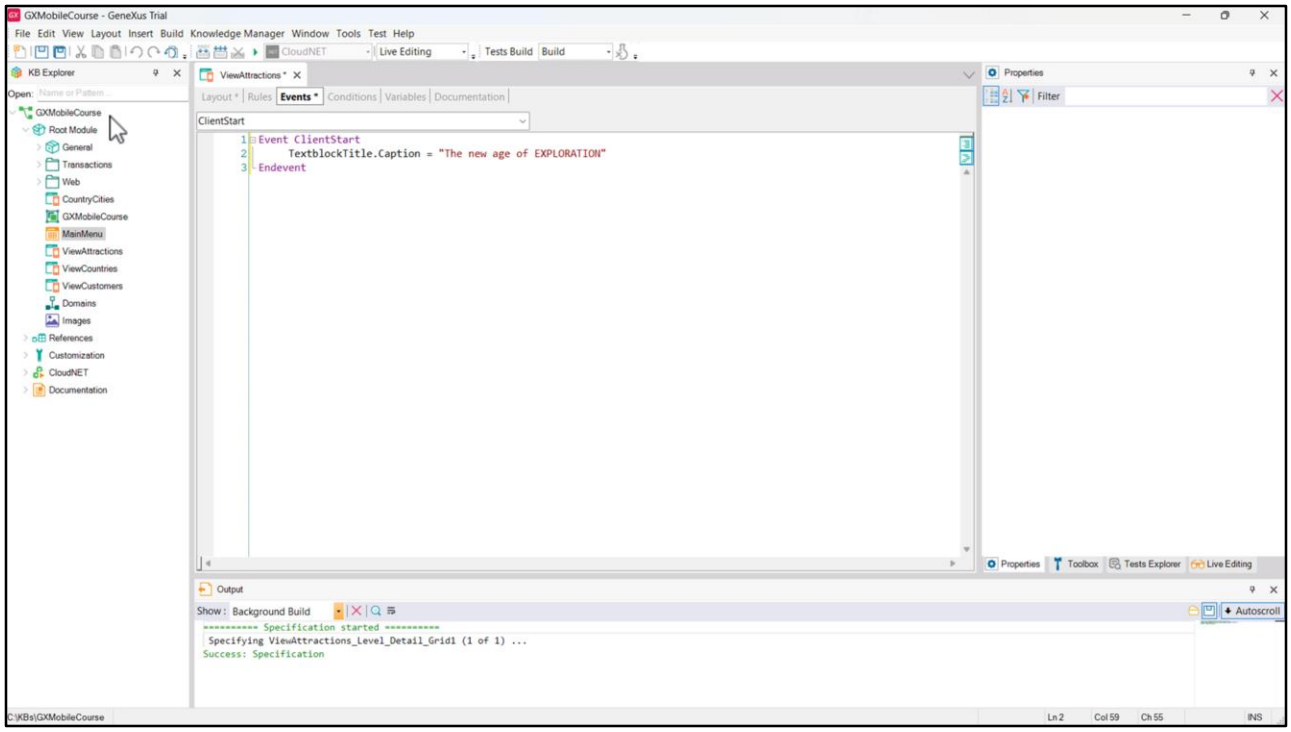
The second part is variable, and is made up of the grid or grids that have been included in the form.

A Panel object will always have a fixed part, and there can be a variable part for each of the grids in the Panel.

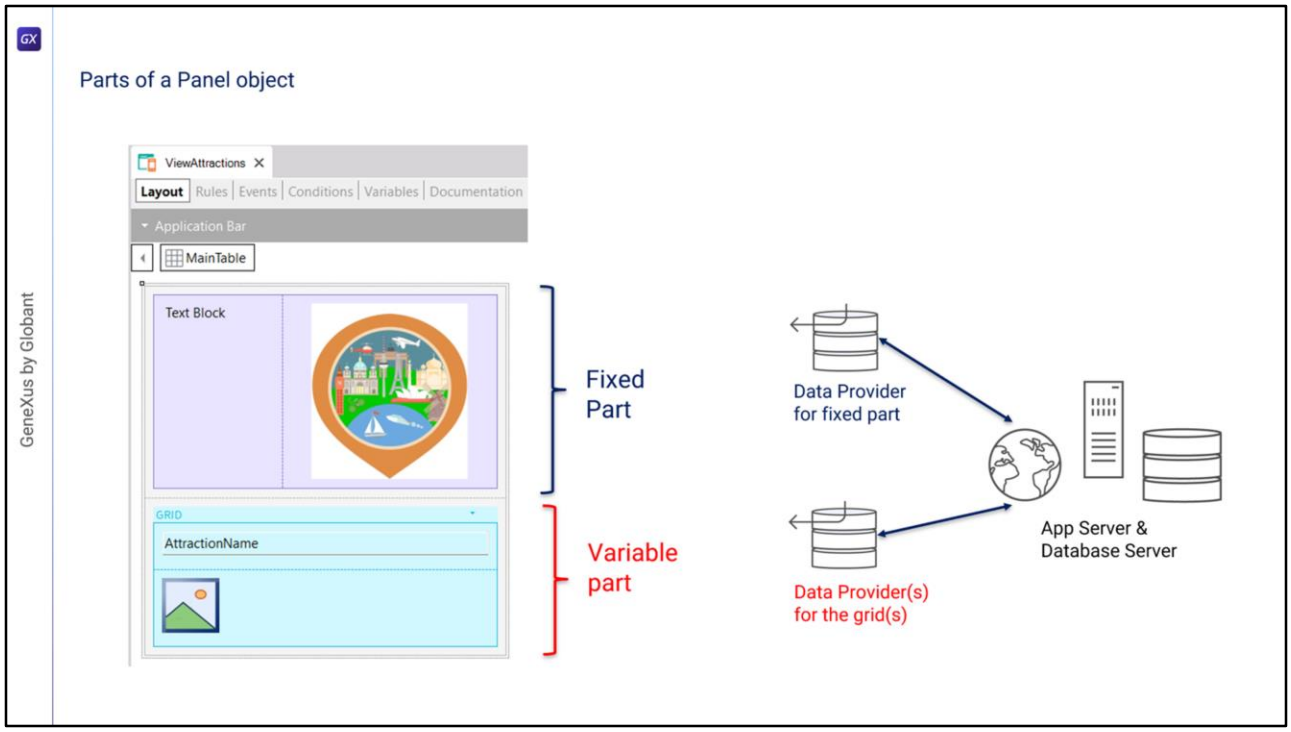
Panel objects differ from Web Panels in that the fixed part is loaded independently of the variable part.



Before moving on, let's go to GeneXus and add some controls to the fixed part of the ViewAttractions Panel we had created. Above and outside the grid, we add a table with a Textblock named TextblockTitle, with centered horizontal alignment and middle vertical alignment, and an image (the same one we are using in the menu, since we are not interested in the screen layout here), with centered horizontal alignment. We configure the Columns Style property of the table so that the first column –containing the Textblock– takes 80% of the total space, and the second column –containing the image– takes the remaining 20%.

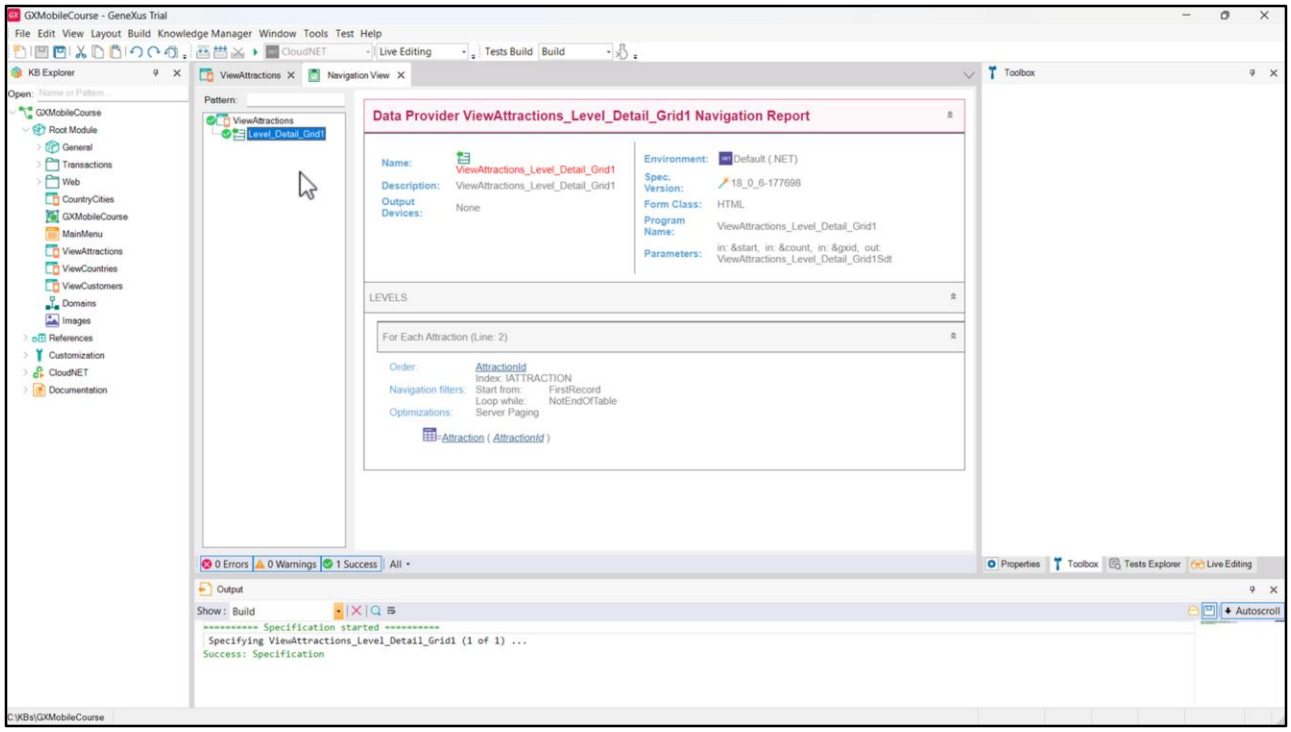


In the ClientStart event, we assign its Caption property the title we want to see on screen: "The new age of EXPLORATION". Next, we save and continue analyzing the Panel data loading.



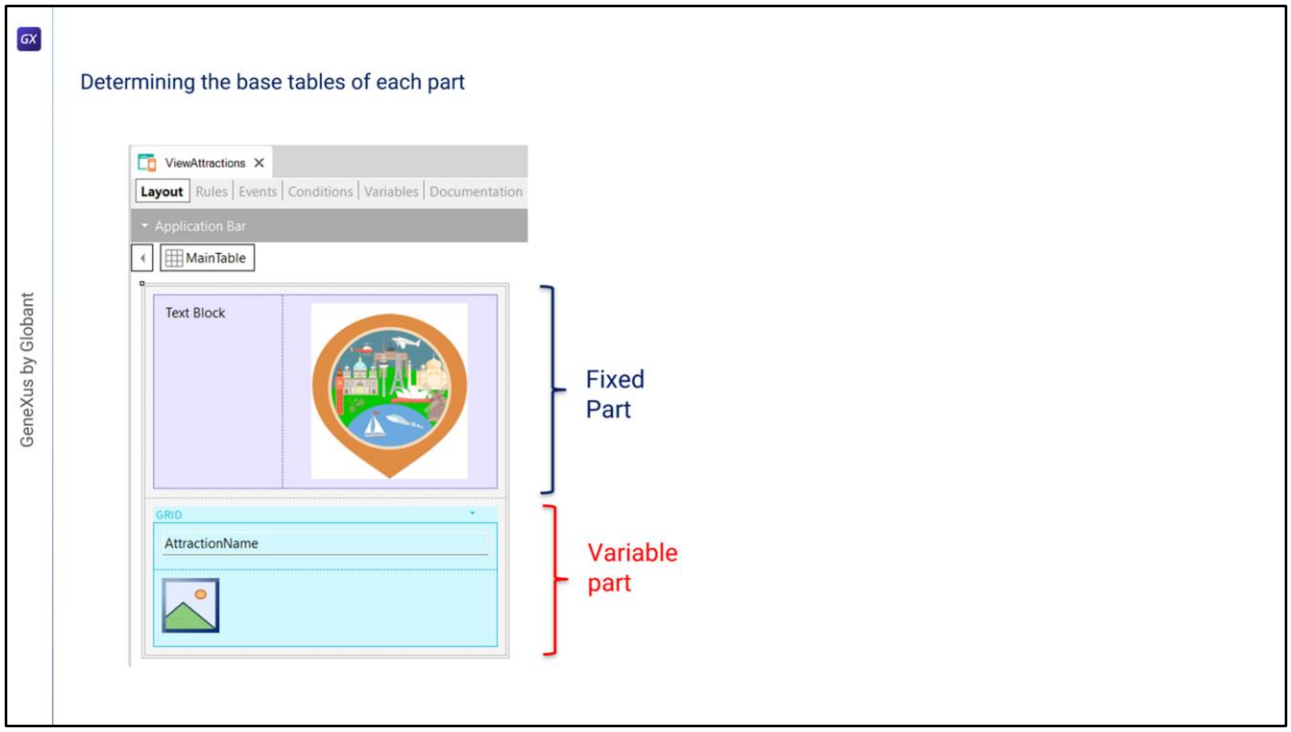
For each part (fixed and grid), GeneXus will automatically generate independent Data Providers that will be published as services on the server and will access the database to obtain the necessary information to load the fixed and variable parts of the Panel.

We will not see these Data Providers in the Knowledge Base because GeneXus will generate and maintain them, but we will be able to see the data accessed by them in the navigation list.



In the navigation list of the ViewAttractions Panel, we can see that below the node of the Panel, there is an entry named Level_Detail_Grid1 corresponding to its variable part, which in this case is made up of Grid1.
If the Panel had more than one grid, a detail node would appear for each grid, since each one will have its own navigation.

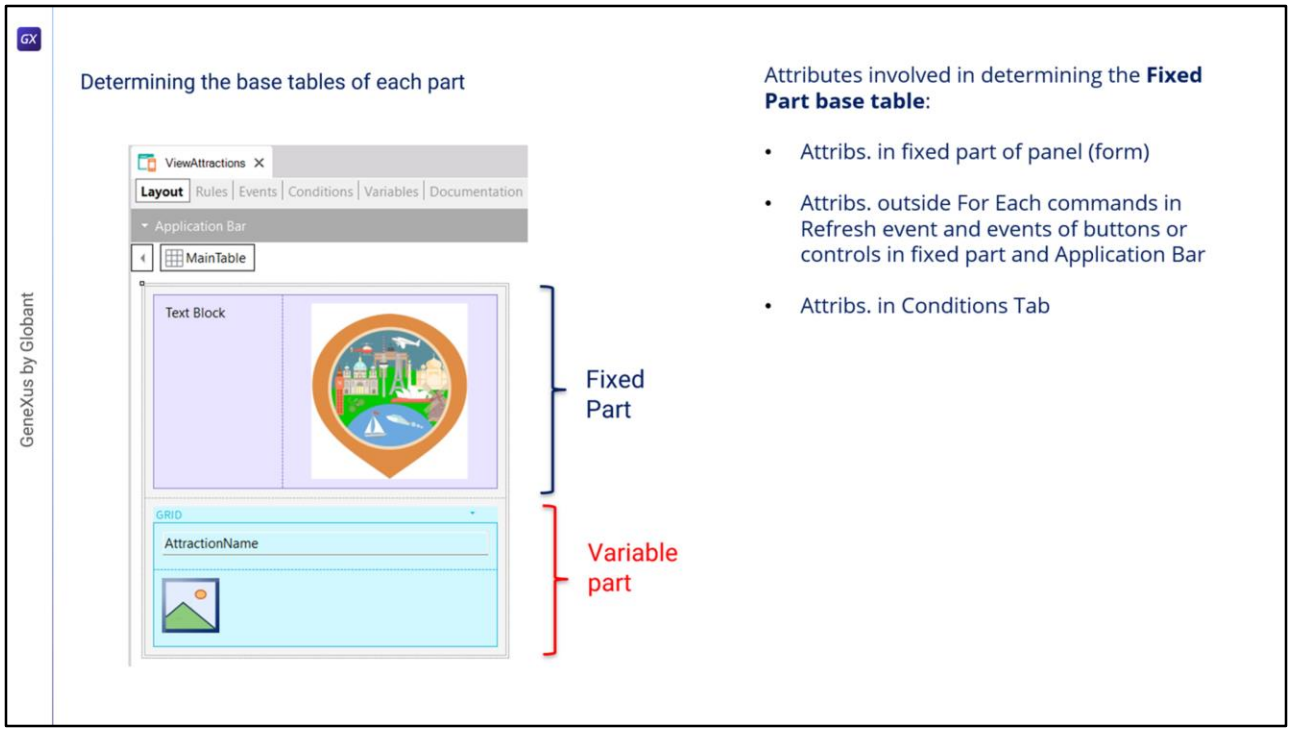
We then select the Level_Detail_Grid1 node, and see that it retrieves the data from the Data Provider ViewAttractions_Level_Detail_Grid1.
This is the Data Provider that was automatically created to load Grid1, published as a service on the server and invoked by the Panel to access the database and retrieve the attractions.
In the Environment section, we can see that the Data Provider was generated in .NET as part of the backend code.
In the parameters section, we see that the Data Provider receives some data from the grid and returns a loaded SDT named View_Attractions_Level_Detail_Grid1, which will contain the data to load the grid.
The information on the list is similar to what we would see with a Web Panel with Attraction base table, since the Panel's grid is running through the Attraction table to show the tourist attractions.



Since the fixed and variable parts of the Panel are loaded separately, the base tables of each part are completely independent of each other. Knowing how GeneXus determines which tables to access and how it runs through them is crucial for our application to work properly.

In the example, we see elements in the fixed part of the panel and a grid, so the base tables of the fixed part and the variable part will be determined separately. It may happen that the fixed part has a base table and the grid doesn't; that the grid has a base table and the fixed part doesn't; that both parts have a base table, or that neither part has a base table.

Remember that here Panels work differently than Web Panels. In a Web Panel with a single grid, if there is a base table, the base table of the Web Panel is unique and there aren't two separate base tables for the fixed part and the grid, as in the case of Panels.

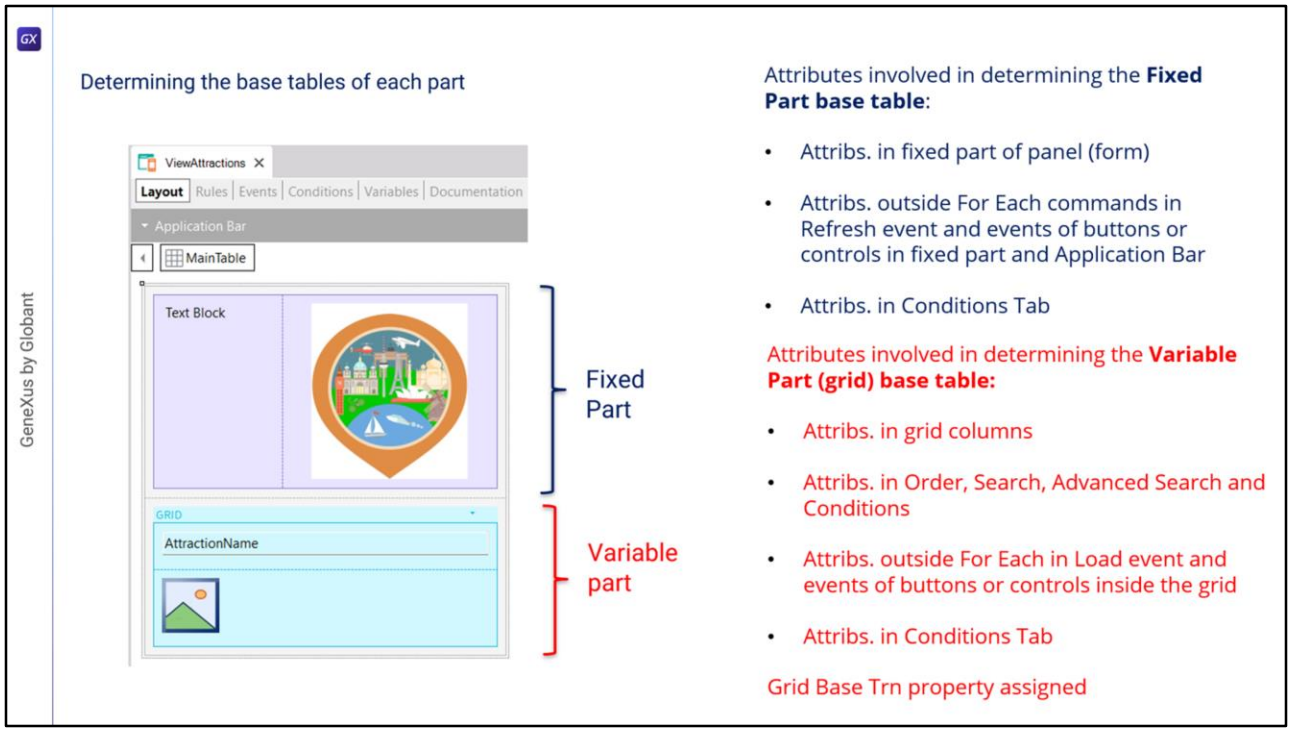


Since in a Panel object the fixed part and the grid determine independent navigations and each part will have its own base table, it is as if there were two parallel For Each commands.

To determine the base table of the fixed part, the attributes that belong to the fixed part of the form and the attributes that belong to the events associated with the fixed part will be taken into account as long as these attributes are outside a For Each command.

These events are the Refresh event and the events associated with buttons or controls of the fixed part, including those of the Application Bar.

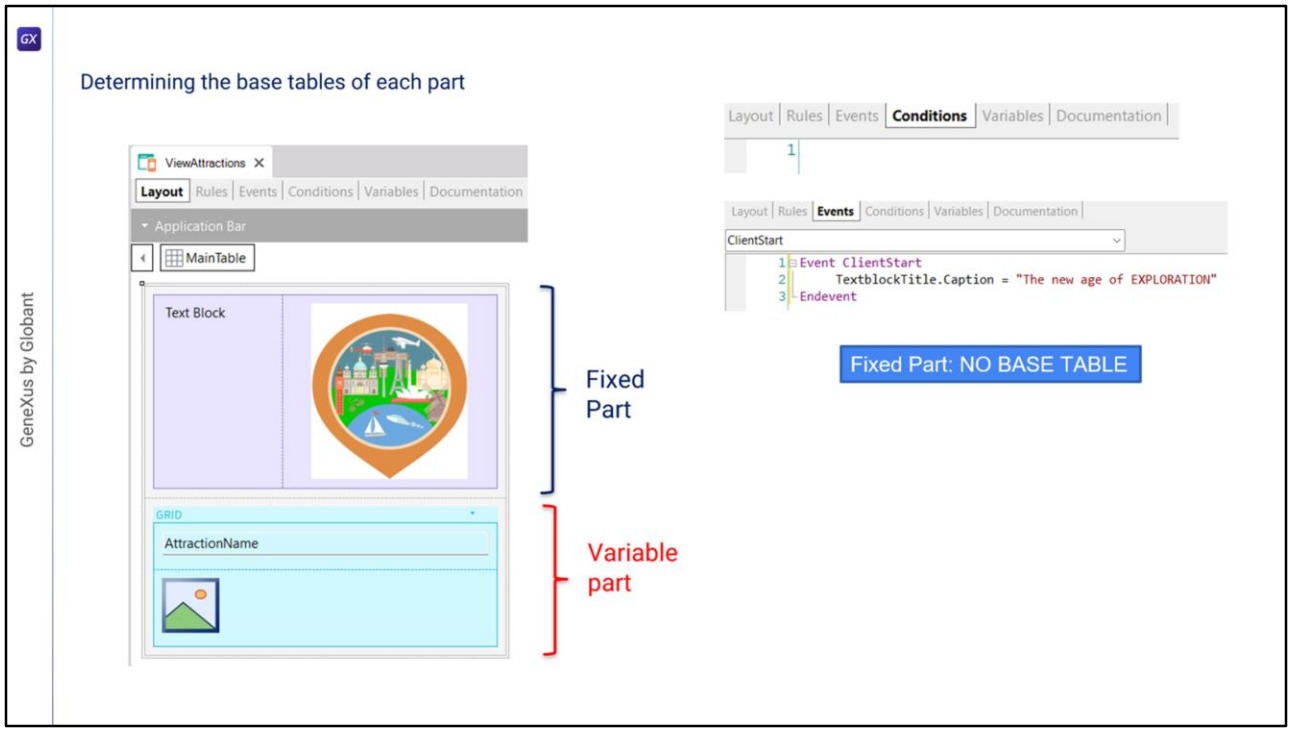
In addition, to determine the base table of the fixed part, the attributes of the Tab Conditions of the Panel object must also be considered.



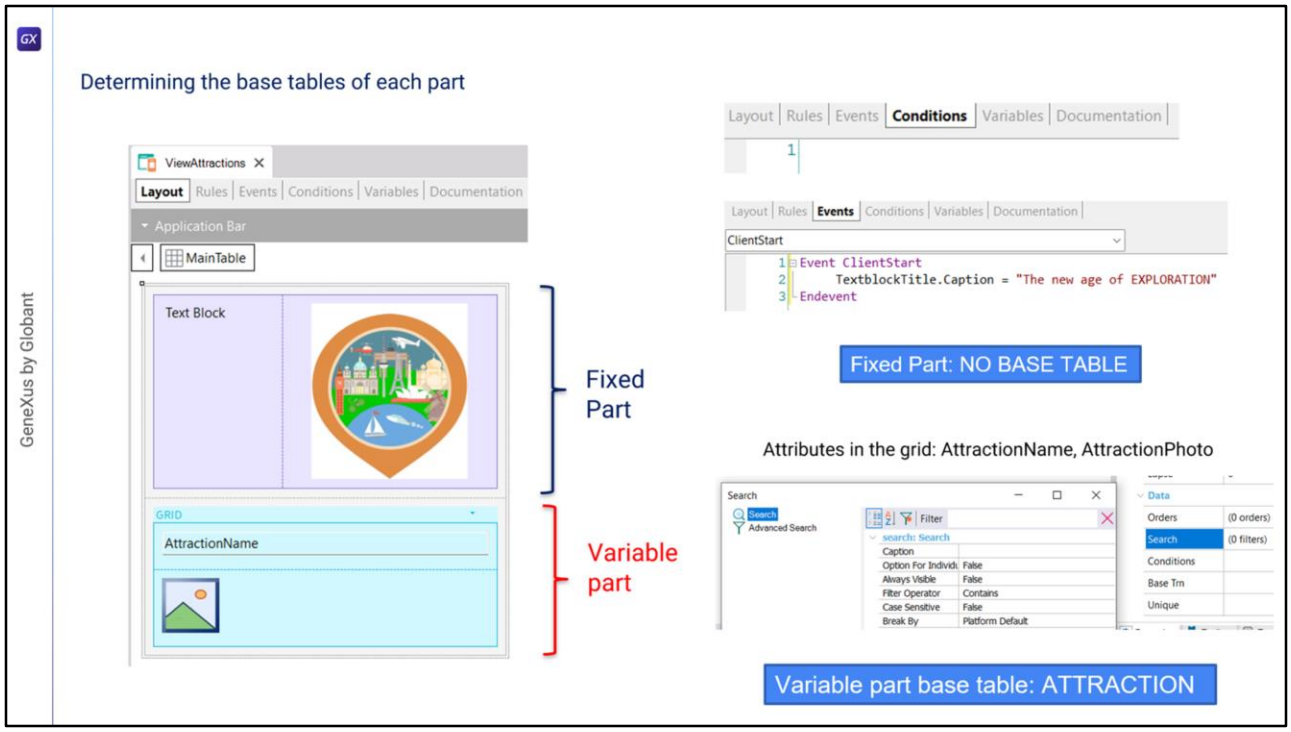
To determine the base table of the variable part, in this case of the grid, the attributes taken into account will be those included in the grid columns, both visible and hidden, the attributes referenced in the Order, Search, Advanced Search and Conditions of the grid, and the attributes belonging to the Load event code as long as they are outside For Each clauses and included in the events of buttons or controls within the grid.

The attributes in the Conditions tab will also be taken into account for determining the base table of all the grids included in the Panel.

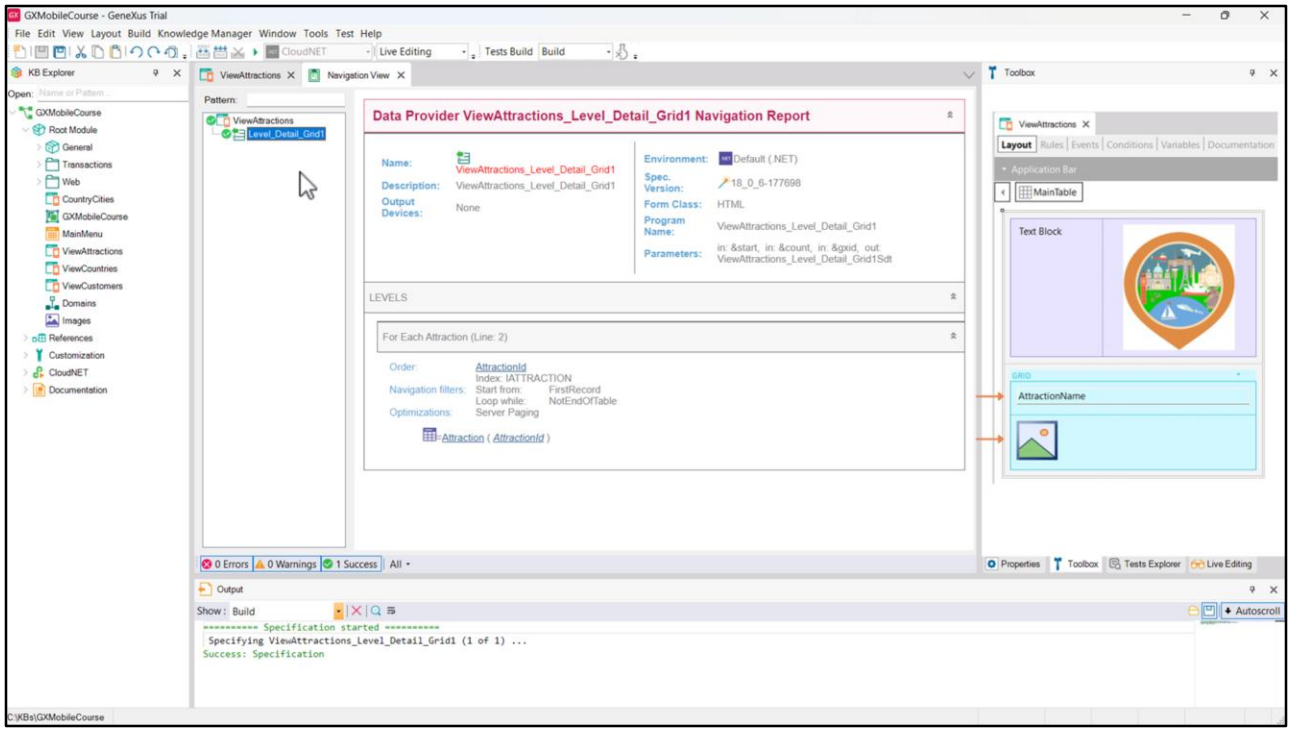
Lastly, the grid will also have a base table if its Base Trn property was assigned with a base transaction. In this case, the attributes that are in the other parts must belong to the extended table of the table associated with the base transaction.



In this example, the fixed part is composed only of a table, a text and an image in the form; that is, there are no attributes. There are no attributes in the Conditions tab of the Panel either. There is no Refresh event or events of controls of the form or buttons in the ApplicationBar; that is to say, the fixed part of this Panel does not have a base table.

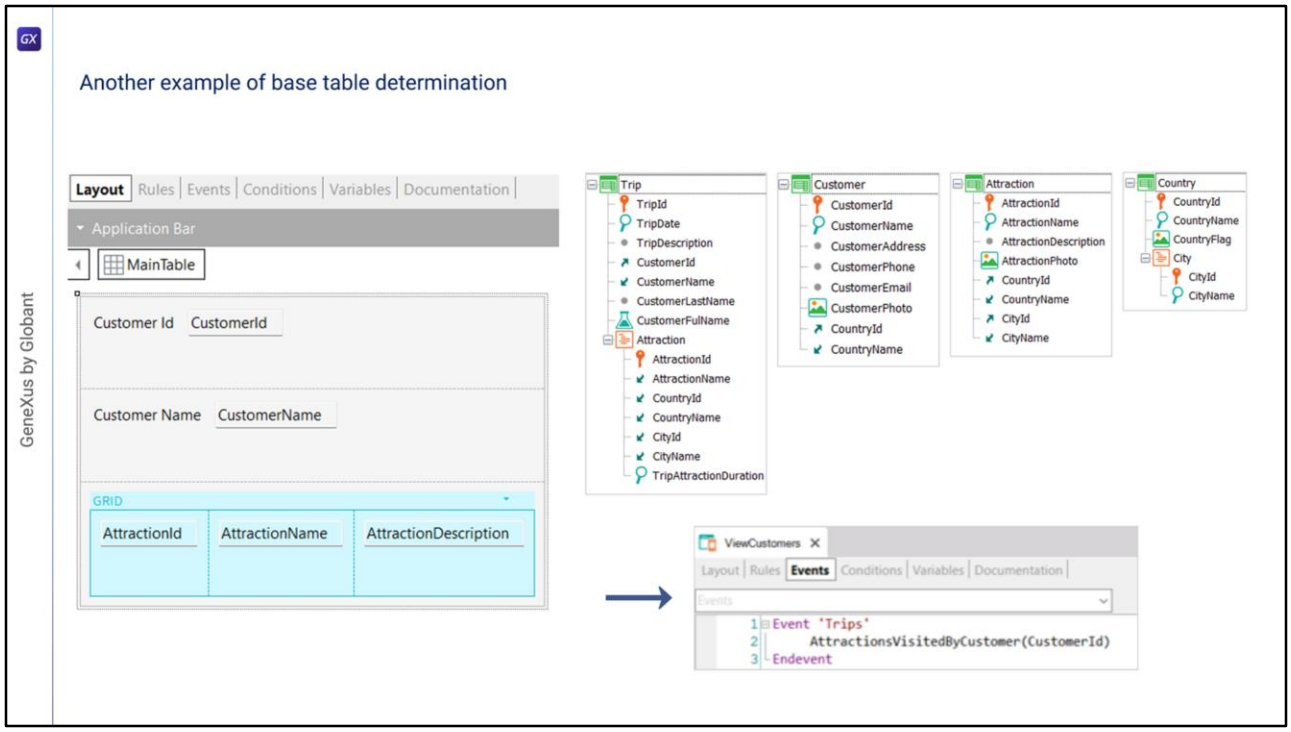


If we analyze the variable part, we see that the grid columns contain the attributes AttractionName and AttractionPhoto. There are no attributes in the Order or Search properties, nor Conditions of the grid, and the value of the Base Trn property is not assigned. In addition, there are no attributes in the Load event or in events of controls inside the grid. Therefore, the variable part of the Panel, composed of the grid, will have the Attraction base table. This means that GeneXus will build a Data Provider as a service on the backend that will run through the Attraction table and retrieve the data that the grid will load.



Let's look at the navigation list of the Panel to check which is the base table of its variable part: it says For Each Attraction, so the base table is indeed ATTRACTION, as we had previously concluded and mentioned.
Below we see that it accesses the Attraction table to retrieve the data of AttractionName and AttractionPhoto.

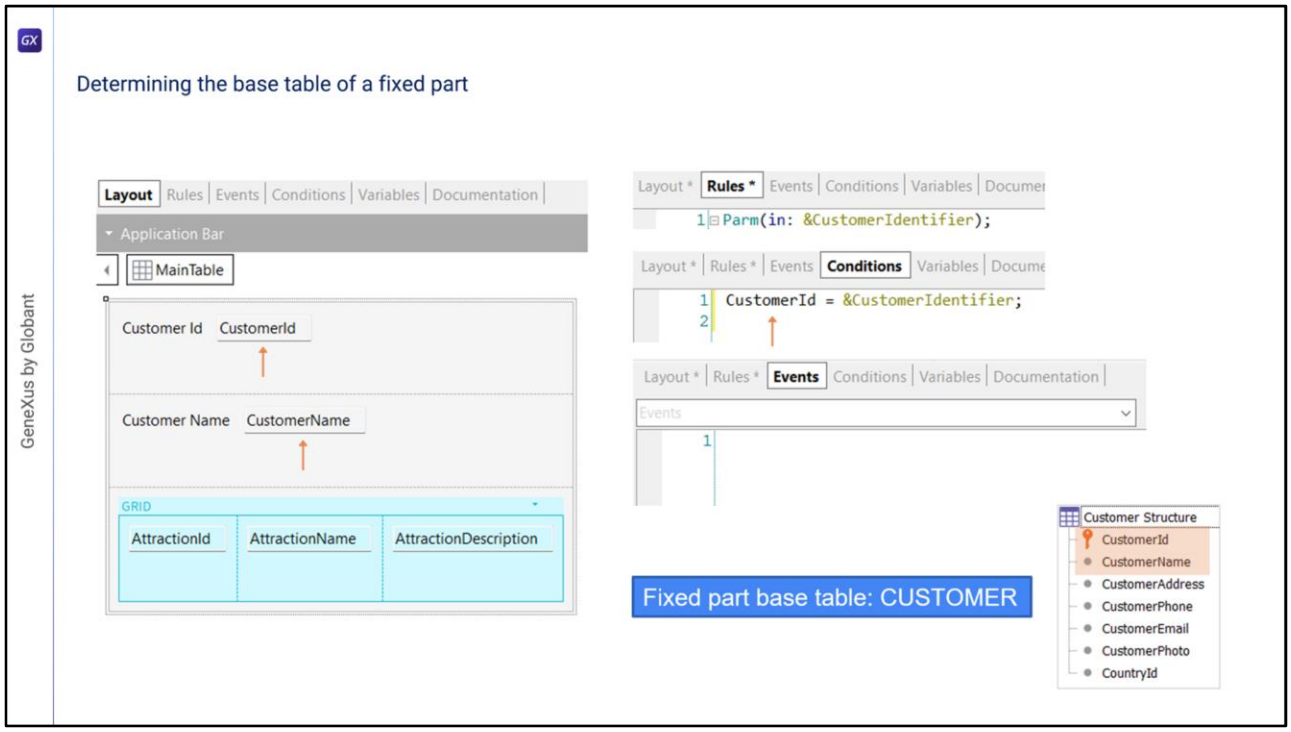
In summary, just because the grid has attributes, in this case in its columns, GeneXus was able to automatically determine which table to run through and create everything necessary to retrieve the required data from the database.



Let's see another example of base table determination. The travel agency wants to have a Panel that shows the attractions visited by a certain customer received by parameter.

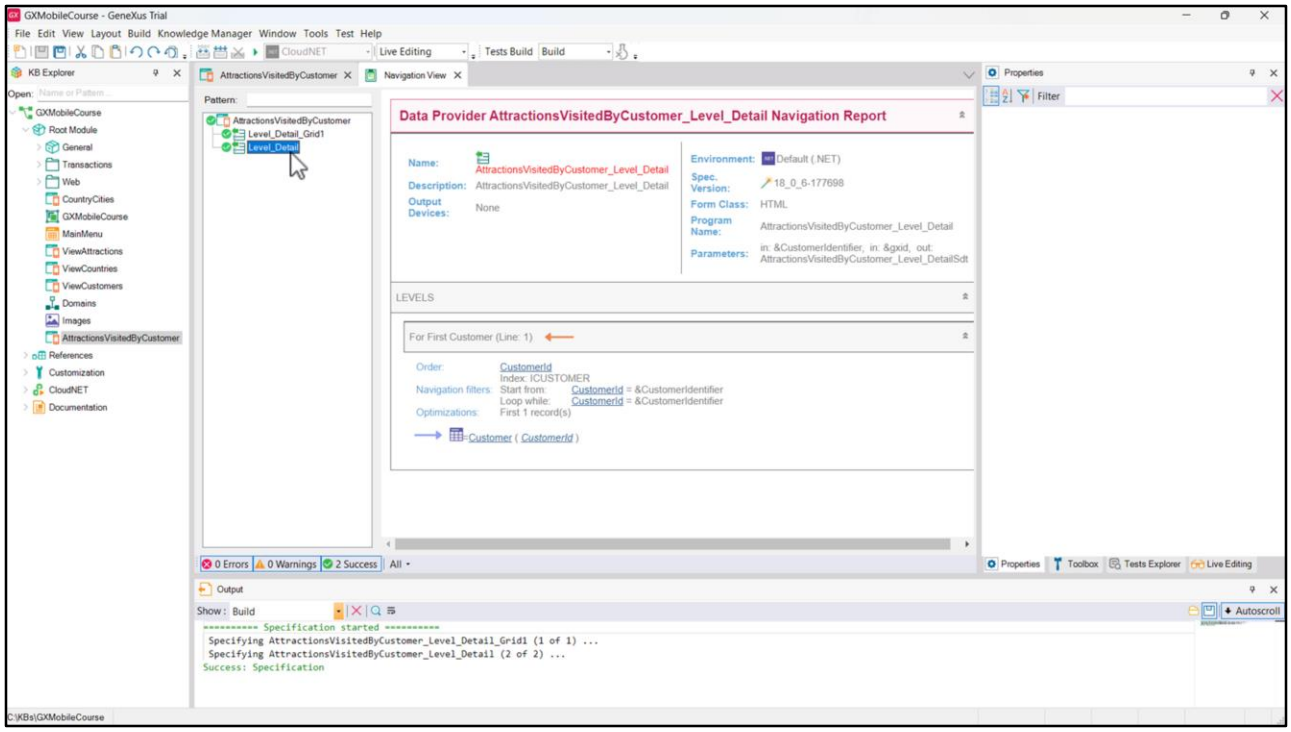
This information is modeled in the Trip transaction, where each trip has a customer (note that CustomerId is a foreign key and the CustomerName attribute is inferred). Each trip also has many attractions that are visited, and are represented by the second level of the Trip transaction. Also shown are the Customer transaction containing the customer data, the Attraction transaction with the attraction data, and the Country transaction with the data of countries and cities.

To test this example, invoke this Panel from another object of your KB, passing it the client identifier as a parameter.

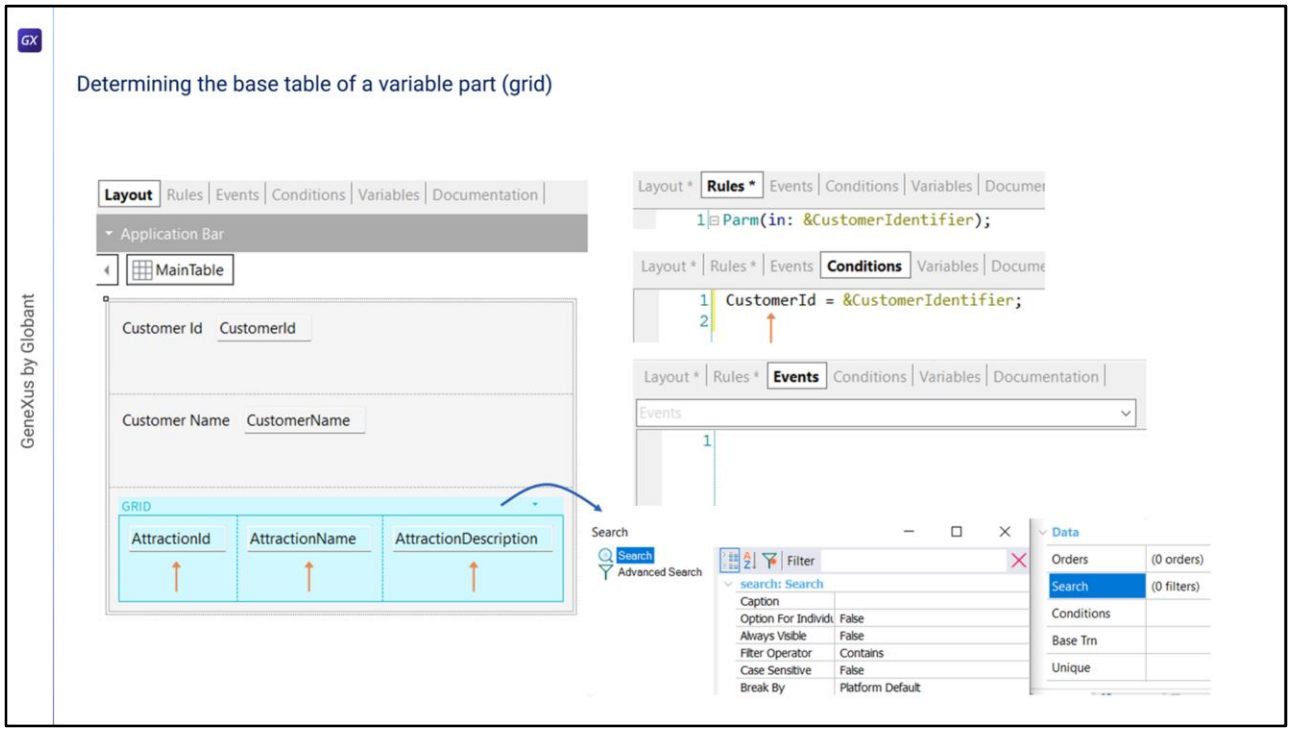


First, let's analyze the fixed part of the Panel. In addition to the attributes included in the form, if we look for attributes in other parts of the Panel, we see that the rules include only the &CustomerIdentifier variable that receives the identifier of the customer to be displayed; in the Conditions tab, we find a filter that ensures that only the data of the customer received by parameter will be displayed.
We have nothing in the events tab.

This means that the only attributes to be analyzed for the fixed part are CustomerId (found in the form and in the Conditions tab) and CustomerName that is found in the form.
The base table is then CUSTOMER as it contains both attributes.

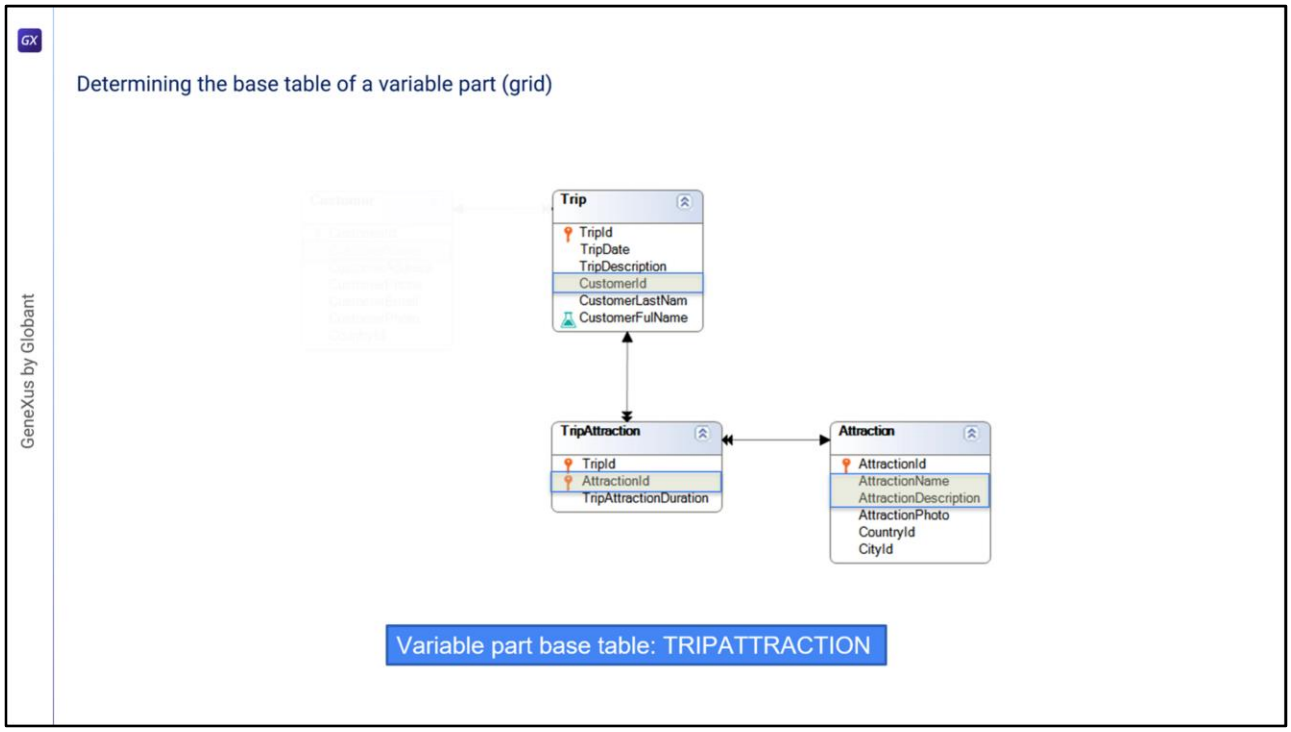


If we look at the navigation list of the Level_Detail node corresponding to the fixed part, we see that the base table is Customer, as we had determined before, and that this table is accessed to retrieve the data of the CustomerId and CustomerName attributes.

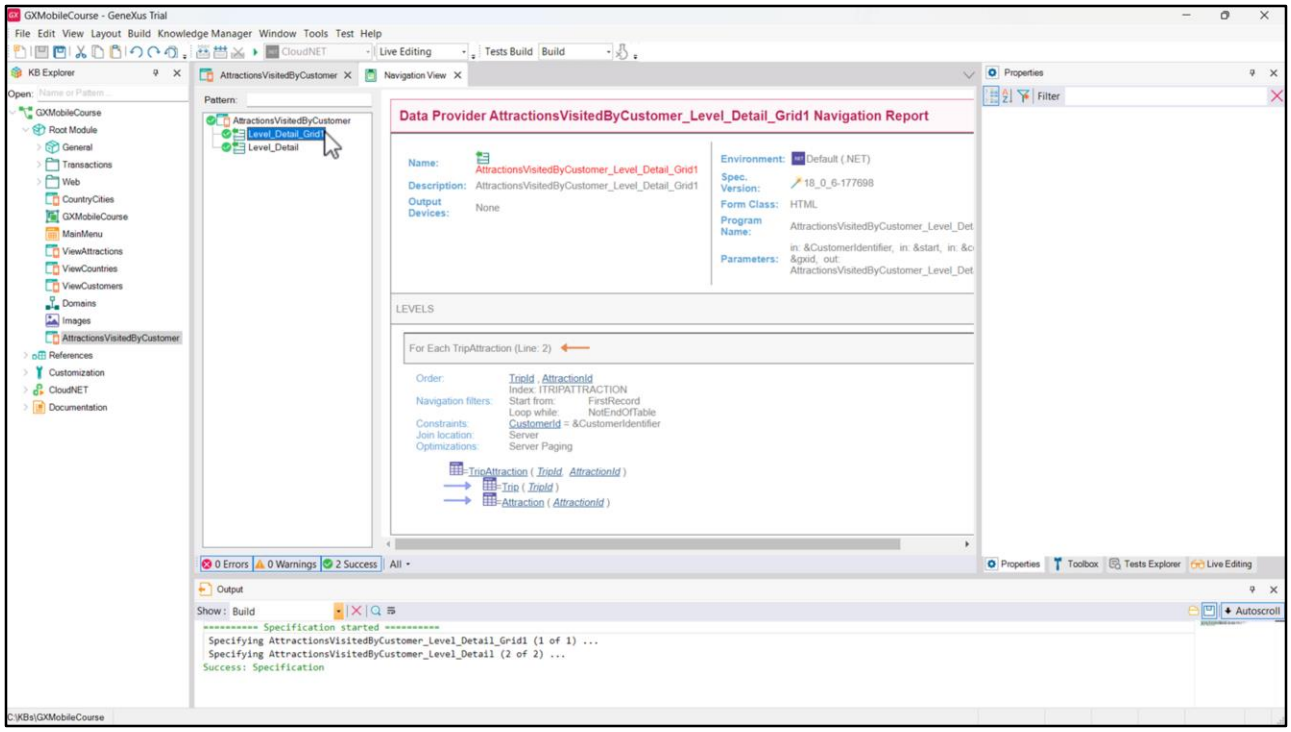


If now we analyze the base table of the variable part, we find the attributes AttractionId, AttractionName, and AttractionDescription as grid columns, and that there are no attributes in any of the grid properties.
We may be tempted to say that the base table of the grid would be ATTRACTION, but let's not forget that the attributes of the Conditions tab must also be considered and there we find the CustomerId attribute as part of a filter.

Therefore, the attributes we find are: AttractionId, AttractionName, AttractionDescription, and CustomerId.



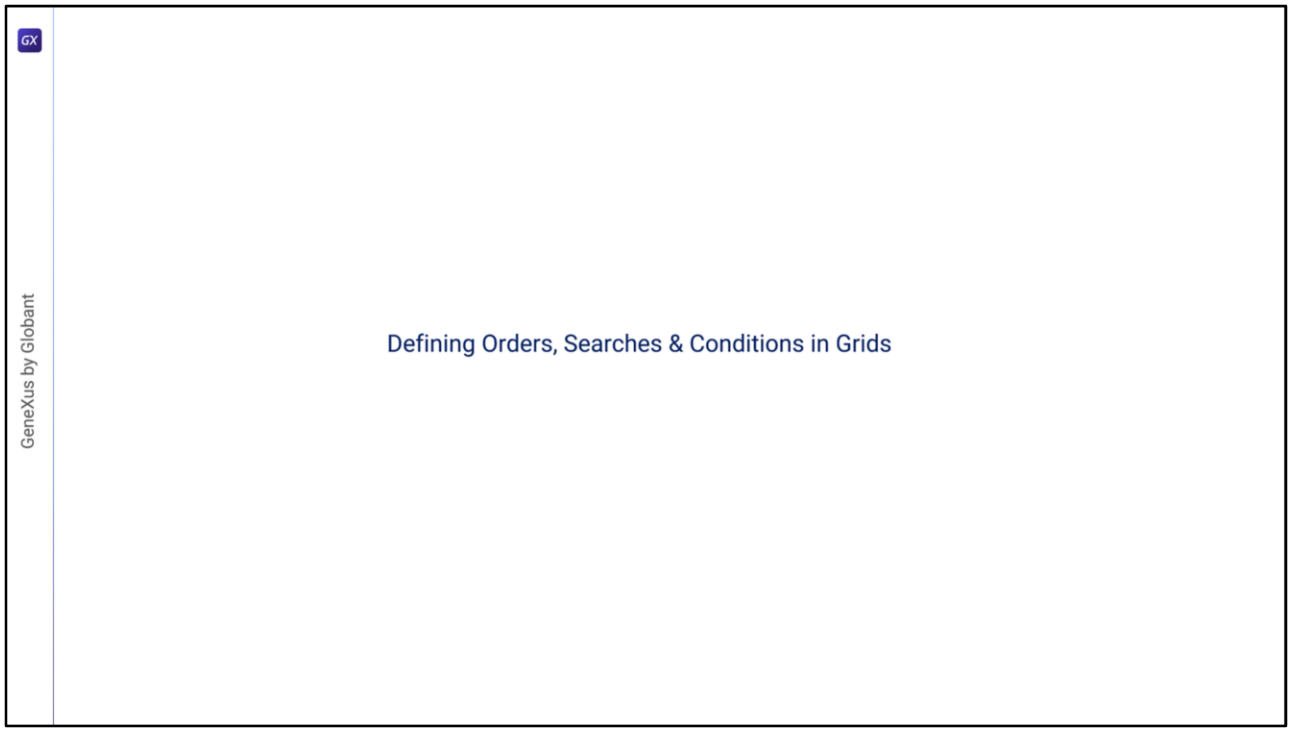
If we analyze the table diagram, we see that the only extended table containing these attributes is the extended table of TripAttraction, so the base table of the variable part of the Panel will be TRIPATTRACTION.



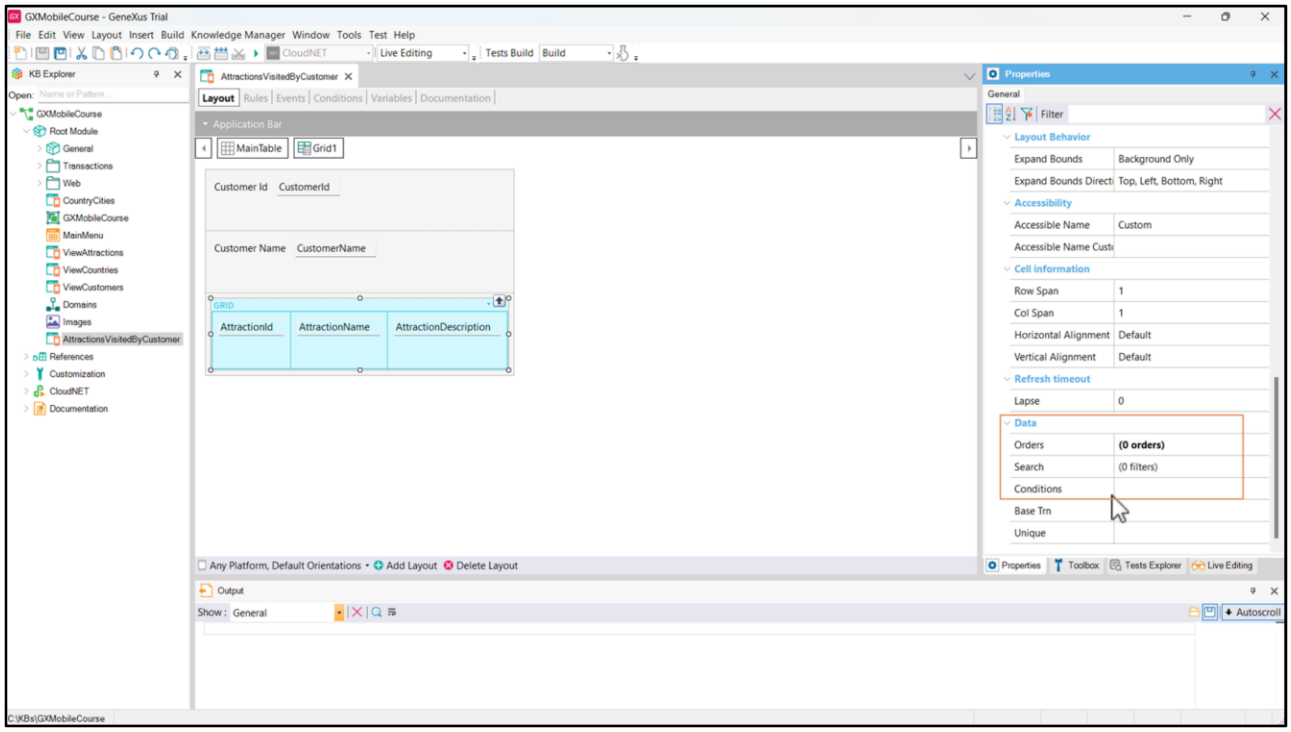
In the navigation list of the Level_Detail_Grid1 node corresponding to the variable part, note that the base table is indeed TRIPATTRACTION, and that it is filtered by the CustomerId value due to the condition included in the Conditions tab.

We also see that the Trip and Attraction tables are accessed to retrieve data from the CustomerId, AttractionName, and AttractionDescription attributes.

We have mentioned before that the code found in the events is also taken into account for determining the base tables of the fixed part and the variable part. In another video we will see the events of a Panel object.

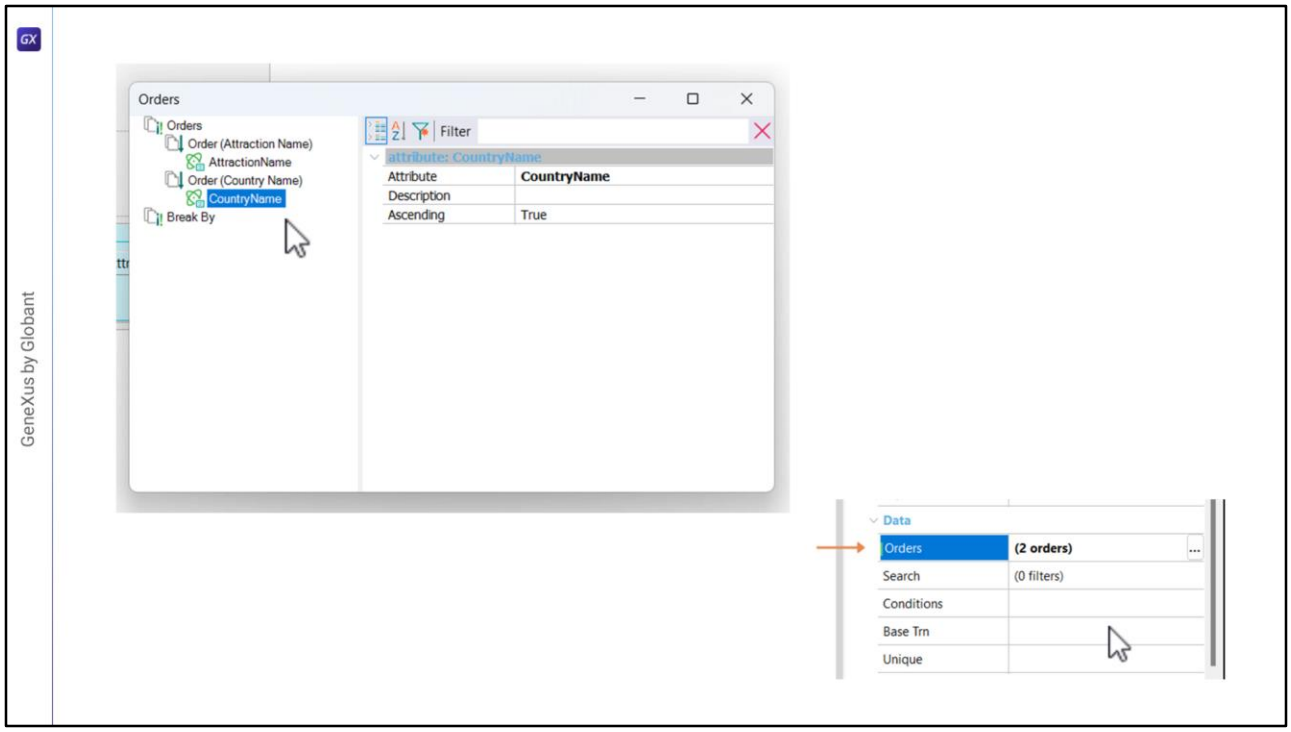


Now let's see how we can sort and group the information, and how to search and set filters and conditions.



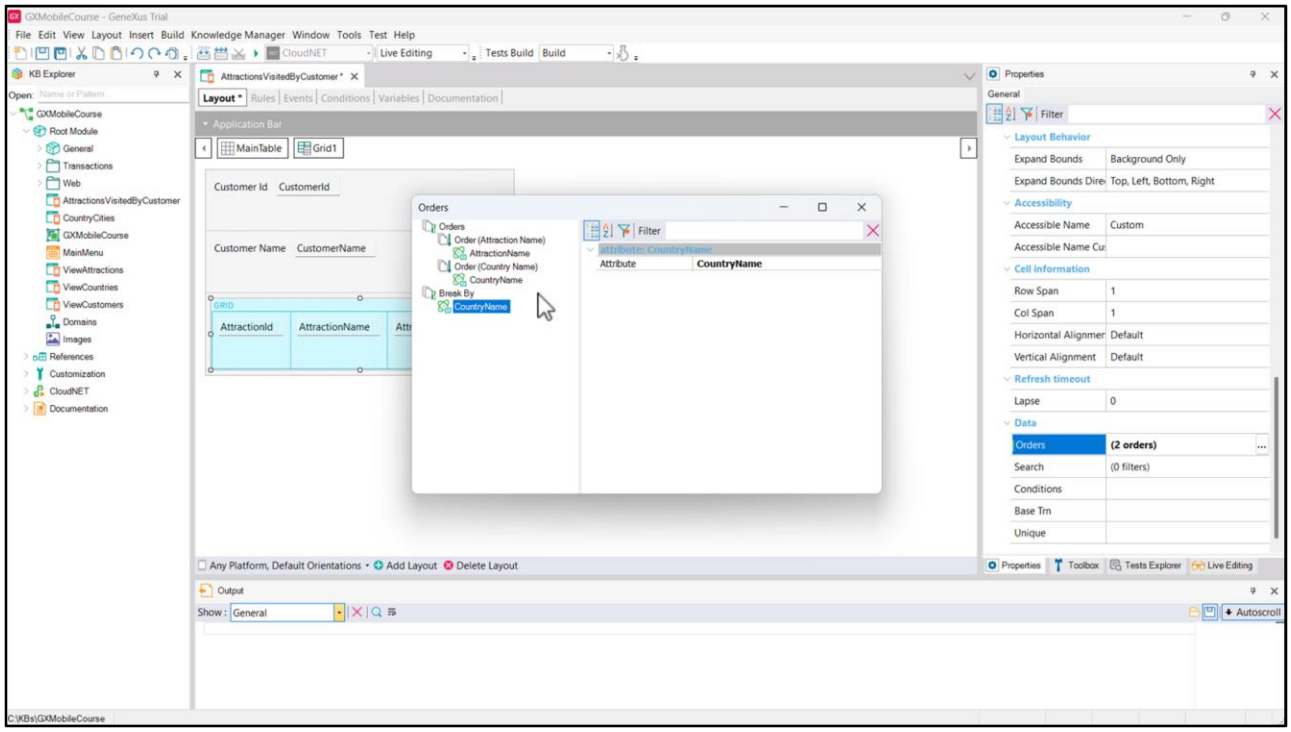
An essential characteristic when displaying information in grids is that it is sorted according to one or more criteria, in order to help the user process the information. We might, for example, want the information in the attractions list to be sorted by name or we might want to sort it by the name of the country in which they are located.

If we go to the properties below the Data group of the grid, we see that no orders, searches, or conditions have been defined for the data.

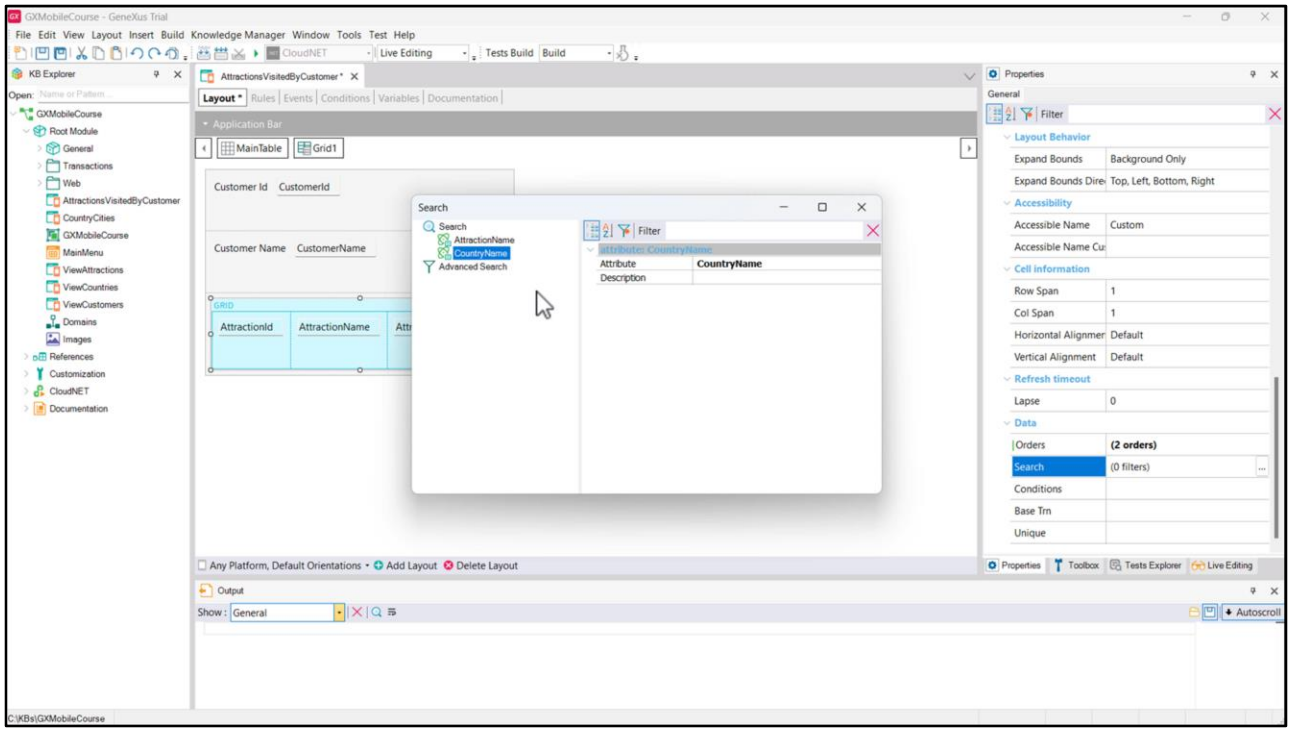


To define our orders, we open the Orders property editor, right-click, select Add, Order. We enter its name, in this first case Attraction Name, right-click on it, and select Add, Attribute; here we select the AttractionName attribute. Let's create, in the same way, the order to sort by country name: right-click, select Add, Order; we name it Country Name... and add the CountryName attribute.

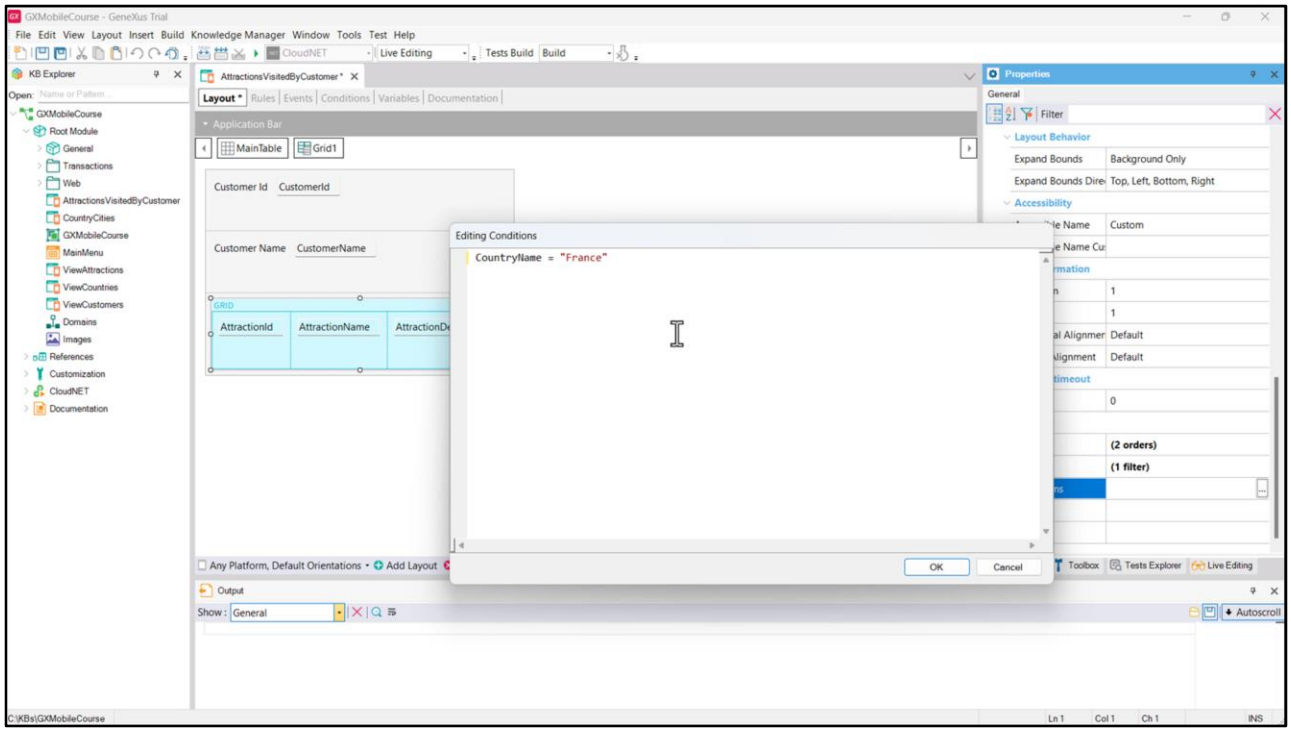
We see that now there are 2 orders defined.



Instead of sorting by country name, the user may also want to have the option to group the attractions according to the country they belong to. We go back to the order definition window, and in the Break By node we right-click, select Add, Attribute, and choose the CountryName attribute to make the control break.



Search is another key feature that every application should provide. For example, we could think about searching for attractions by name or country. To do this, we open the Search property editor, and simply add the two attributes by which we want to search: AttractionName and CountryName.



In addition, sometimes it may be necessary for some records to meet certain conditions to be displayed: for example, we might want the attractions grid to show only attractions in France, so in the Conditions property we type: CountryName = "France".

In this video, we studied the logic used by GeneXus when loading data on a Panel's screen and determining its base tables. Also, we saw how to improve the user experience of our application by providing orders, searches, and conditions for the data to be displayed.