Web Screens

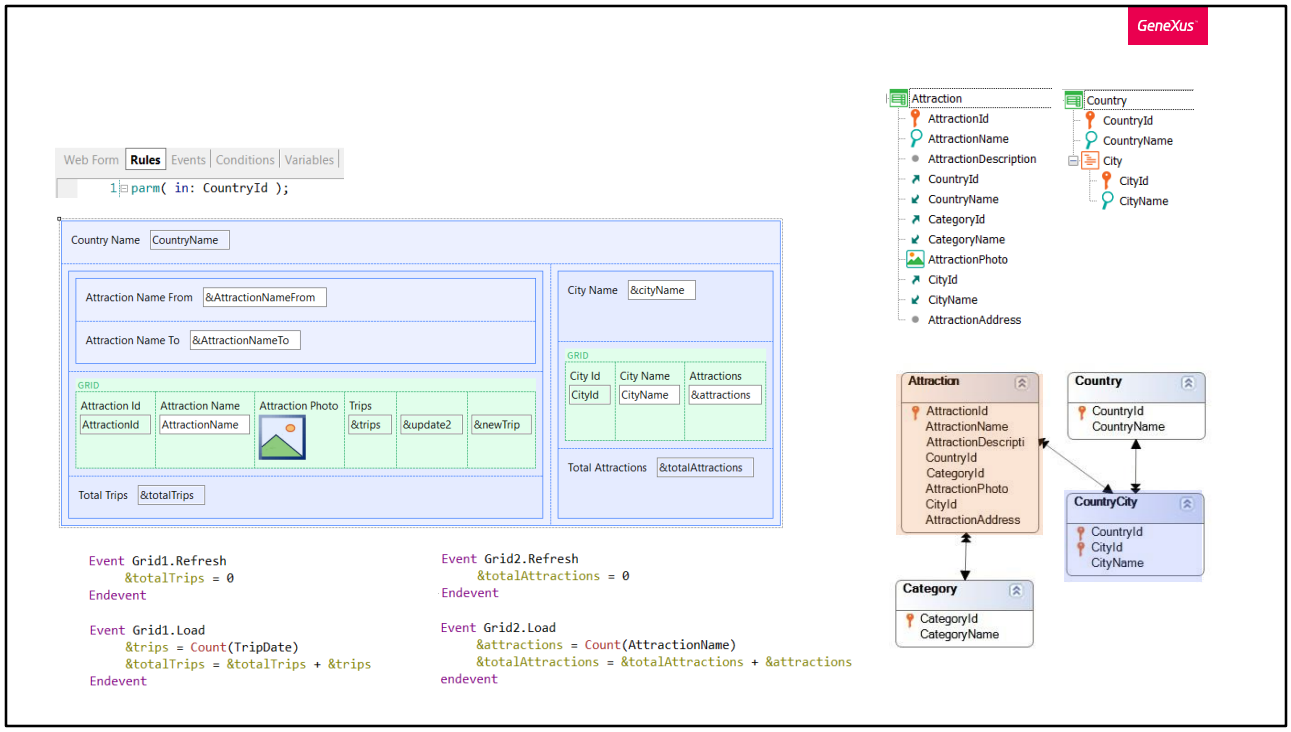
Base tables and navigation in web panel with multiple grids

GeneXus™

GeneXus

Web Panel with SEVERAL Grids

But, what happens when a web panel has more than one grid? In that case, it's not possible to consider the base table of the web panel, but rather that of each grid.



The definition of navigations will depend on whether the grids are parallel or nested.

Consider the case of parallel grids in the first place: each grid will determine its own navigation independently from the other. So, it is possible that a base table exist for one grid and not for another grid.

In this example, both grids will have base table, because, clearly, you can see attributes in each of them, and that is enough to confirm that there will be an implicit navigation in each grid.

And the question is: how do we determine the base table for each of them?

Start by noting that this example differs from the previous one only in the fact that you have added the grid on the right and a variable to filter the data in that grid and another one to show a total.

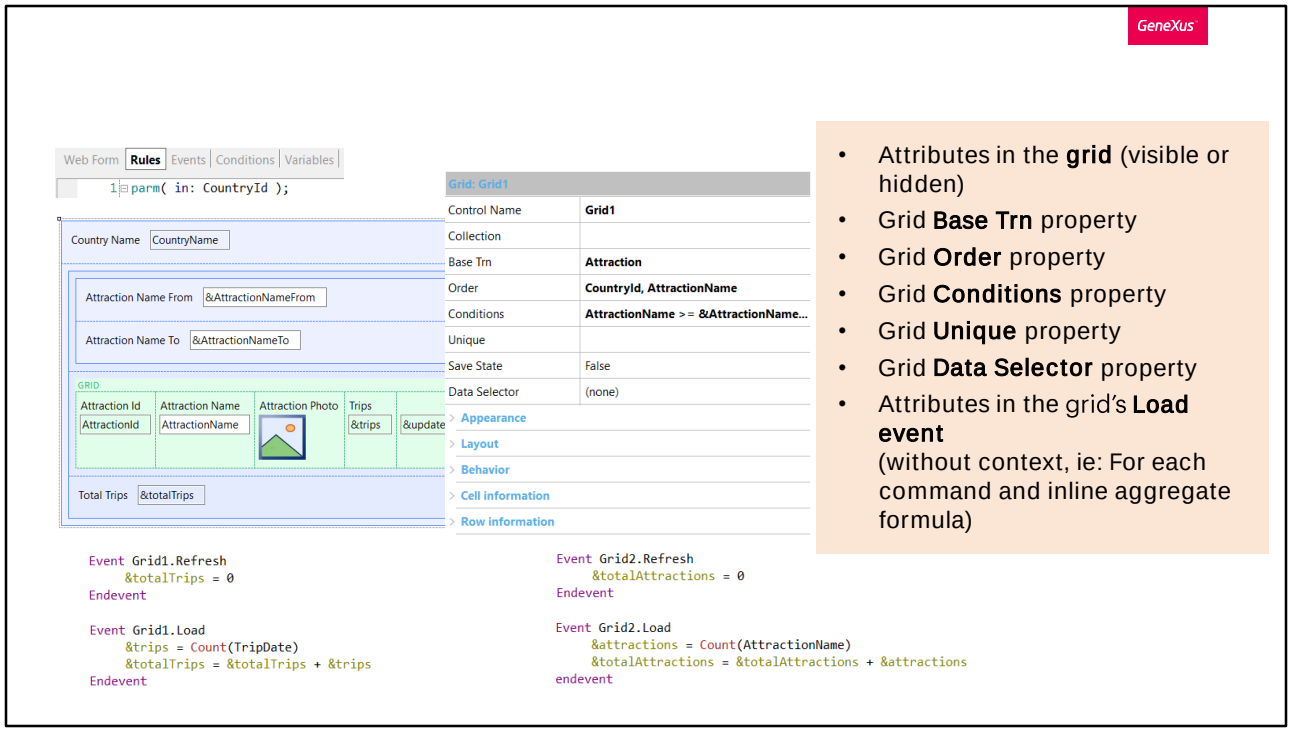
It is a known fact that, in addition to the generic Refresh event of the whole panel, when there's more than one grid, the generic Load event disappears. Now you have specific Refresh event and Load event for each grid. Each Load event will be triggered only once or N times depending on

whether GeneXus finds a base table for that grid or not.

It is easy to infer that the base table of the first grid will be Attraction, and the base table for the second grid will be CountryCity, and in both cases the filtering will be by CountryId, an attribute received by parameter which, as usual, will not take part in the definition of base tables at all. It will be part of what follows such definition.

However, you could think that, since both tables are related in the database (note that, in fact, CountryCity is part of the extended table of Attraction, so, for each attraction loaded on this grid, there will be an associated record in this table... or considering it the other way around, for each city loaded in this other one, there will be N related attractions). As mentioned, you could think that this relation will have an effect on what is loaded into the grids, but it actually will not. GeneXus will not define any implied relation between them.

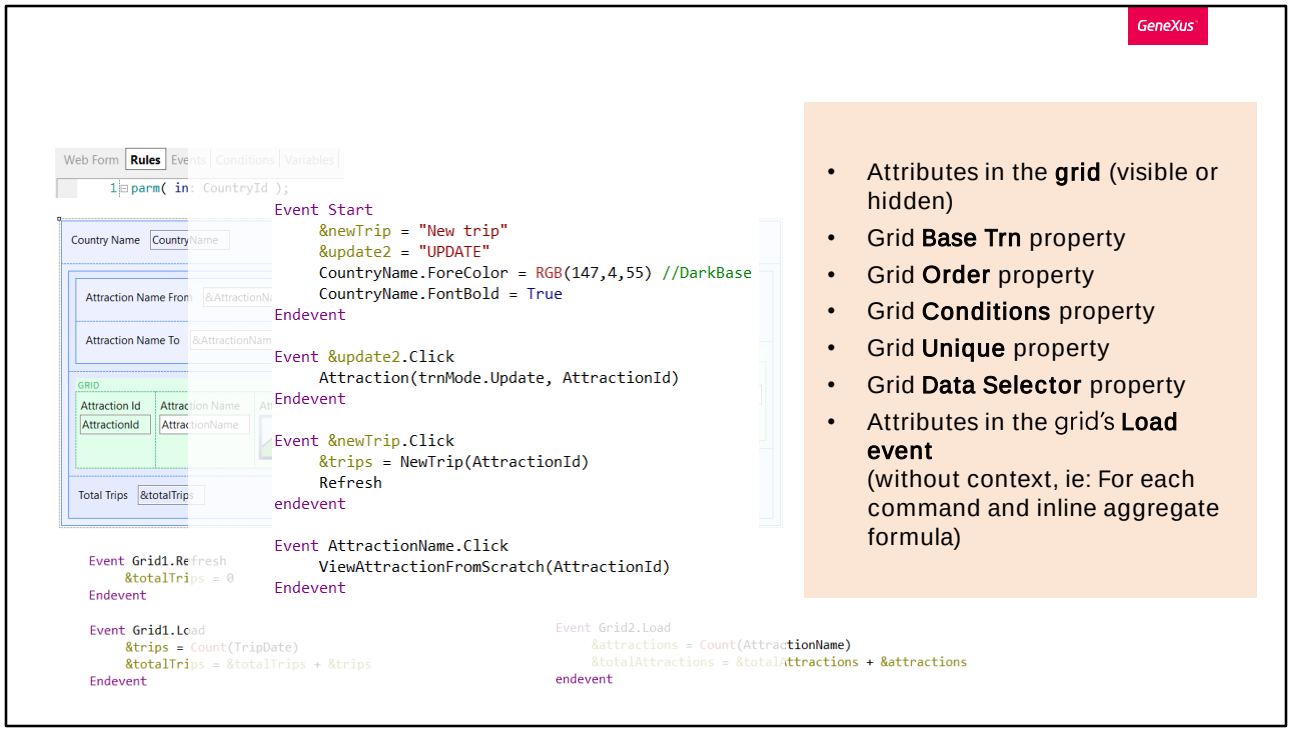
All the attractions in the country received by parameter will be loaded in the first grid, and all the cities in that country will be loaded in the second grid.
With this possible confusion now cleared, focus now on how GeneXus defines the base table for each grid.



- Attributes in the **grid** (visible or hidden)
- Grid **Base Trn** property
- Grid **Order** property
- Grid **Conditions** property
- Grid **Unique** property
- Grid **Data Selector** property
- Attributes in the grid's **Load event**
  (without context, ie: For each command and inline aggregate formula)

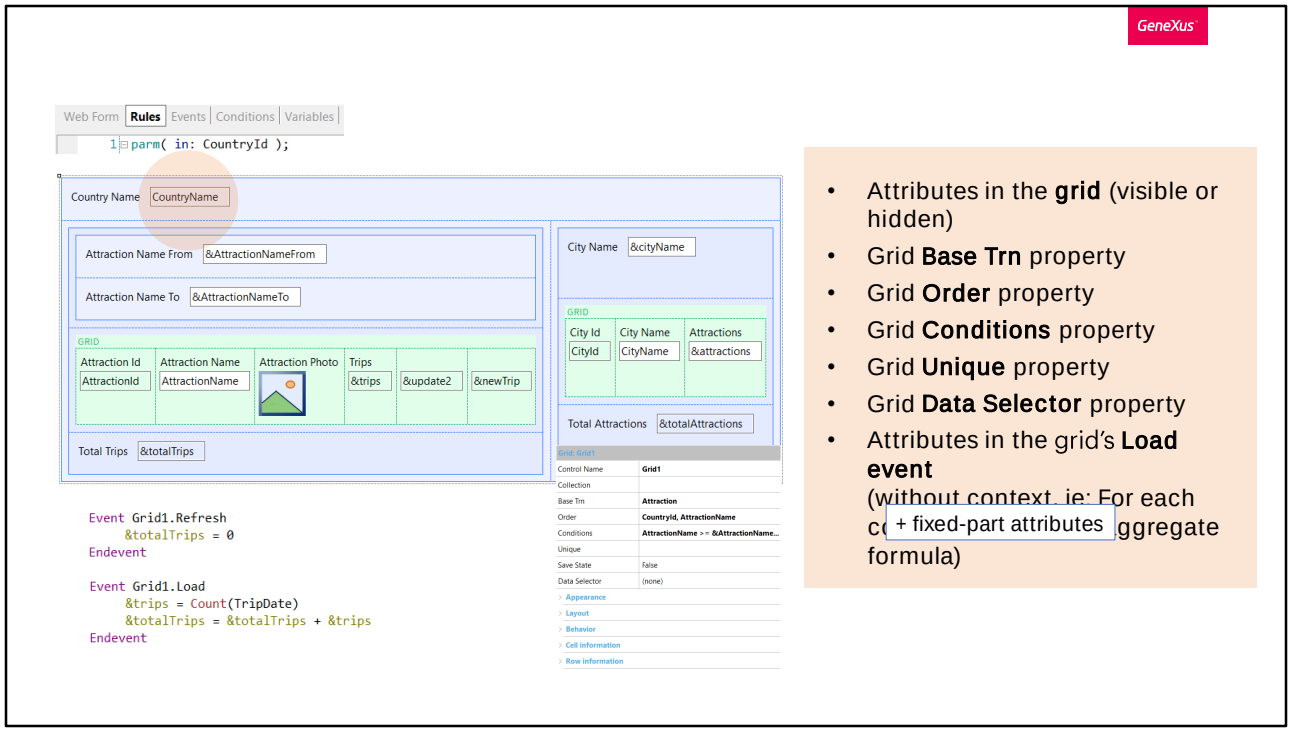
Take the first one in line.

Consider the grid's attributes (visible or hidden), the same grid properties as for the case of a single grid (the Base Transaction, obviously, and properties: Order, Conditions, Unique, and Data Selector). Unlike in previous cases, the "separate" attributes of all events will not be considered here. Only the Load event of the grid will be. So, in the case that a separate attribute existed in the Refresh event, it would not be participating.



- Attributes in the **grid** (visible or hidden)
- Grid **Base Trn** property
- Grid **Order** property
- Grid **Conditions** property
- Grid **Unique** property
- Grid **Data Selector** property
- Attributes in the grid's **Load event**
  (without context, ie: For each command and inline aggregate formula)

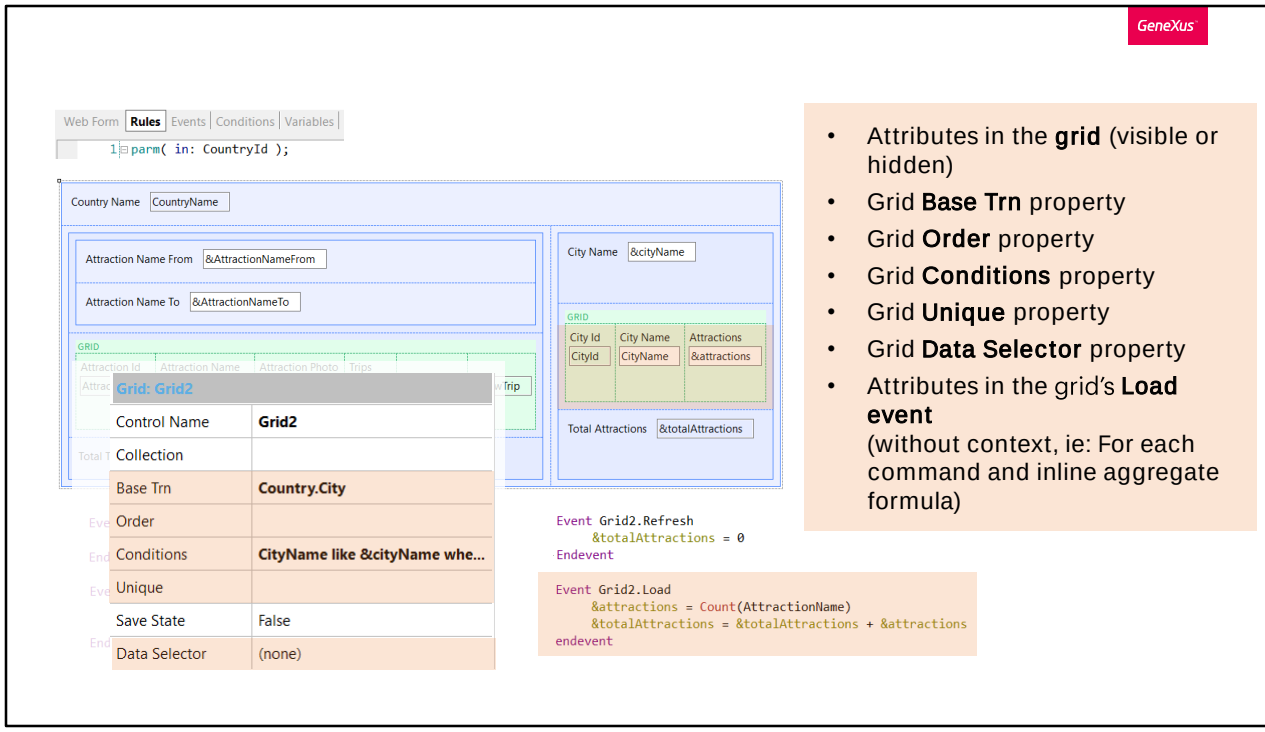
And the same goes for any other event. Like these others.



In addition, for the case of the first grid -and only for it- if attributes exist in the fixed part, as is the case, these attributes will also be considered for determining the base table. And only for it. In the case of all the other grids none of the attributes of the fixed part will participate.

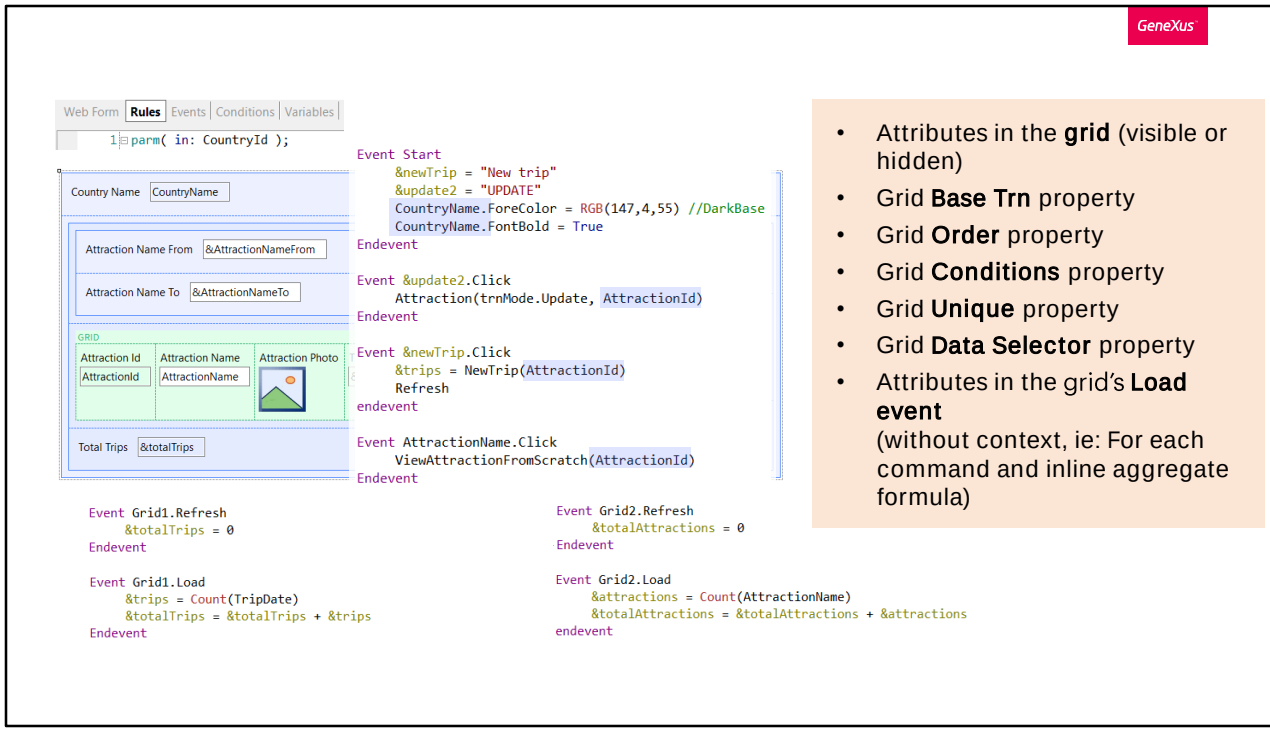
So, in our case, in determining the base table of Grid1 all these grid attributes are considered, in addition to any others "separated" in this Load event. There isn't any. And, of course, the grid properties mentioned.

It is clear why there will be base table, which is Attraction. If any of these attributes were not in Attraction's extended table, then we would be warned in the navigation list.



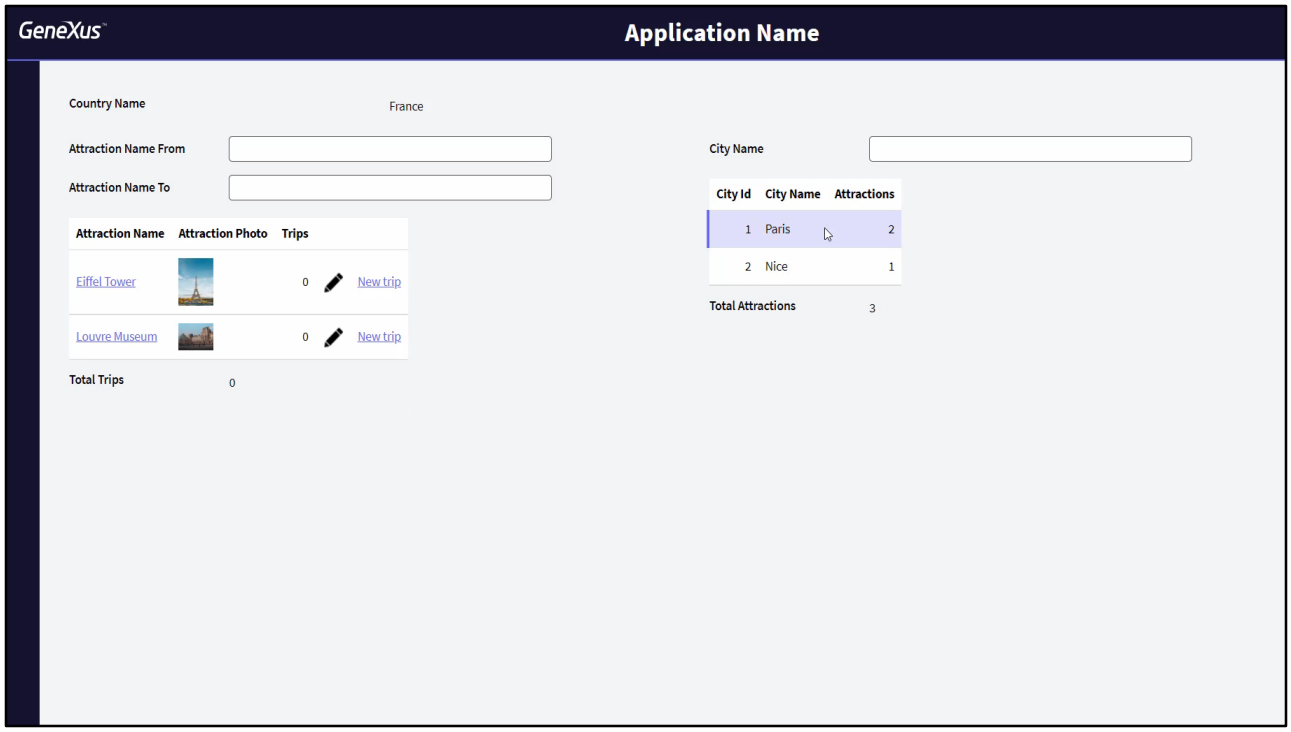
- Attributes in the **grid** (visible or hidden)
- Grid **Base Trn** property
- Grid **Order** property
- Grid **Conditions** property
- Grid **Unique** property
- Grid **Data Selector** property
- Attributes in the grid's **Load event**
  (without context, ie: For each command and inline aggregate formula)

Additionally, in order to determine the base table of Grid2, the ones considered will be all these grid attributes and any others "separated" in the Load.
In this case, there isn't any either. Also the Grid's properties will be considered.
We can clearly see why the base table is CountryCity. In this case, the CountryName attribute of the fixed part doesn't participate.



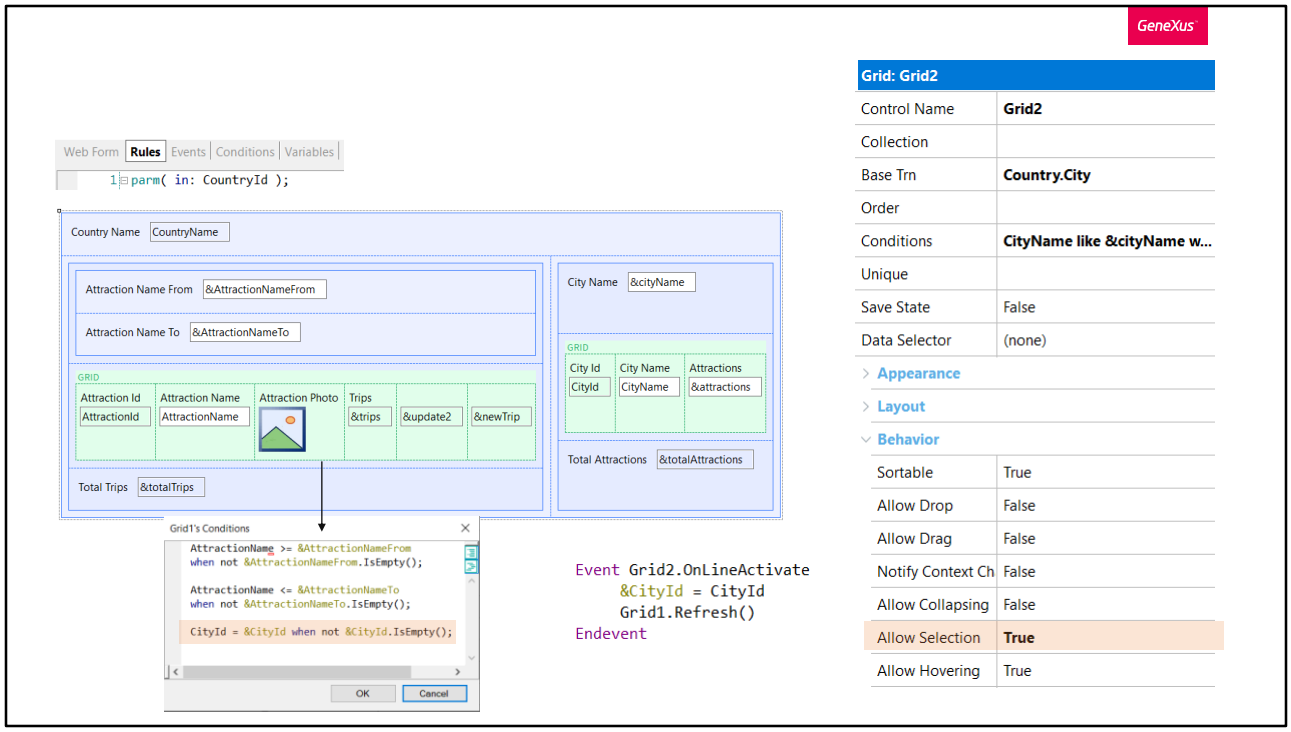
But what happens with the attributes that appear in these other events? They must just belong to the extended table of any of the grid base tables. Otherwise, we will informed in the navigation list.

In this case, we have CountryName and AttractionId.



Like we said, for parallel grids, navigations are not related automatically.

If, for example, when the user clicks on a line of the grid that shows cities, we wanted the grid that shows tourist attractions to only show those in that city, as we can see here, how could we do this?

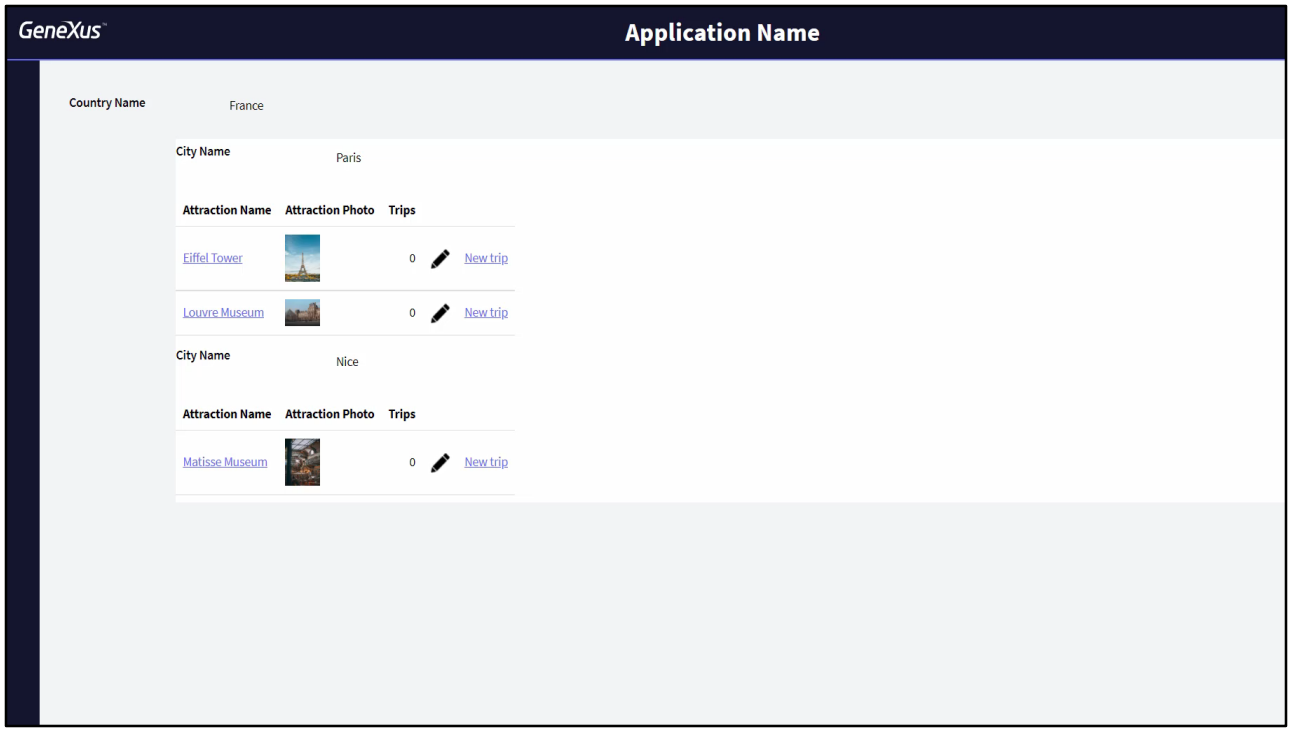


There are several ways to go about it. We will show the one implemented here.
Following a Save of our panel, we set the AllowSelection property for the cities grid to allow the selection of a line through a click on any part of it. We can make it appear with a different color, or not.

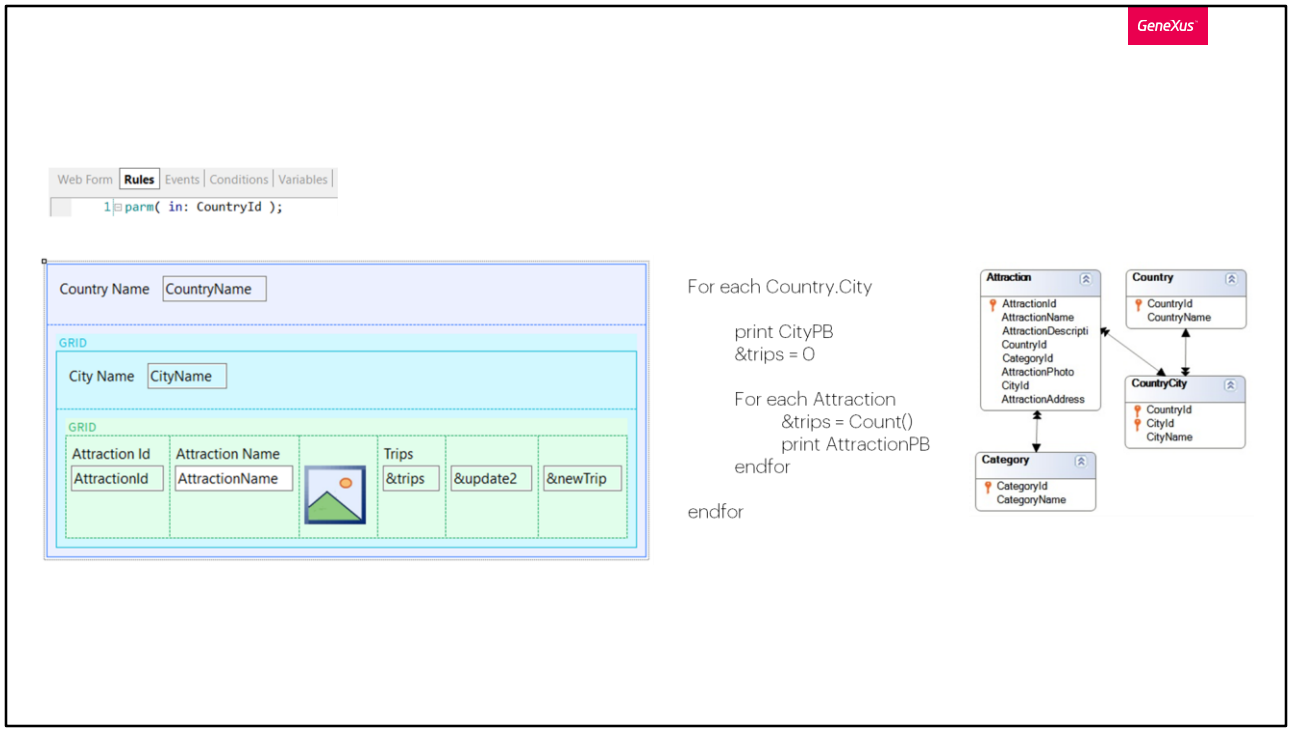
We also program the grid's OnlineActivate event, for a line to be triggered when the user selects it, with the possibility of assigning the city identifier of the line selected to a variable.

Then we refresh the attractions grid, because we added a new condition: that only the attractions whose city matches that of the &CityId variable be loaded, to the extent that the latter is not empty.

This will achieve the behavior shown in runtime.



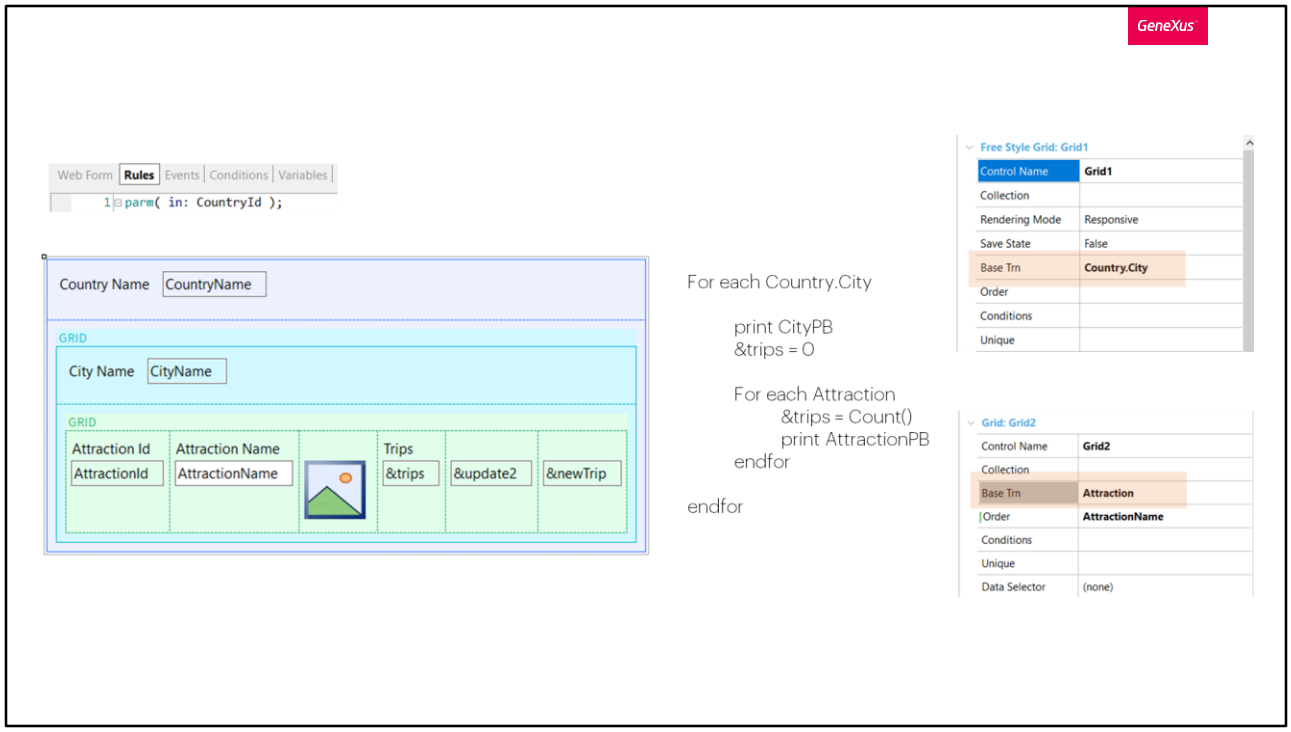
The other alternative is to directly show the attractions for each city in the country received by parameter. This means using nested grids, like we did here.



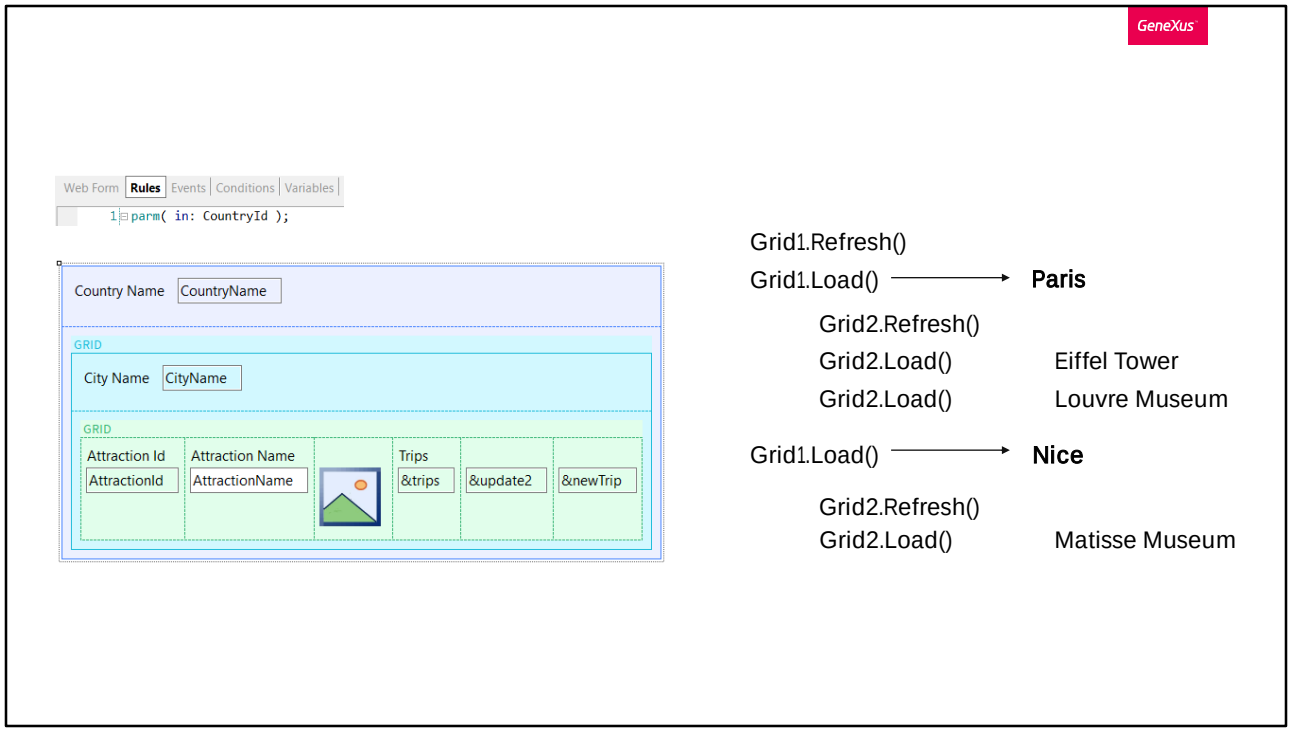
We are well aware that having nested grids is similar to having nested For eachs, so the way to define their base tables and resulting navigations will be analogous.

If you program this object as listed, you would have the external For each navigating the CountryCity table, which will have an implied filter by CountryId, so it will go over all the cities in that country, and for each of them it will print its name; and prior to moving on the next one it will execute the internal For each, which will go over the Attraction table, filtering implicitly by country and city, and printing each attraction in that country and city.

This is the navigation that you will achieve in your web panel, but, like always, there are two options: implementing each grid with, or without base table.



When you implement both grids with base table, which is the way to work less, you will be establishing the Country.City base transaction for the first grid, and Attraction for the second one.



For any case, the Refresh event of Grid1, the external one, will take place first, followed by that grid's Load event N, once or N times depending on whether the grid has its base table or not.

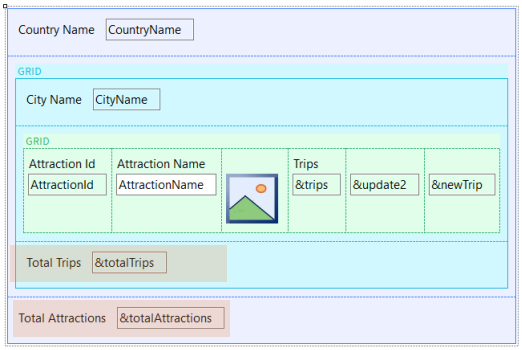
In this case, since two cities (Paris and Nice) have been entered for France, you know that the external grid's first Load will be Paris, and immediately after, and prior to executing the Load again to load Nice, the Refresh event of the nested grid will take place. And its Load event will come immediately after, once or N times, depending on whether it has base table or not. It does have base table in this case, so a Load will be triggered to load the Eiffel Tower and another one to load the Louvre Museum.

Once the load of the nested grid is concluded, it will move on to load the following city, Nice. And it will be the same, triggering the Refresh event of the nested grid once, to then trigger the Load event N times for loading the new attractions, in Nice, which in this example is just one.

GeneXus

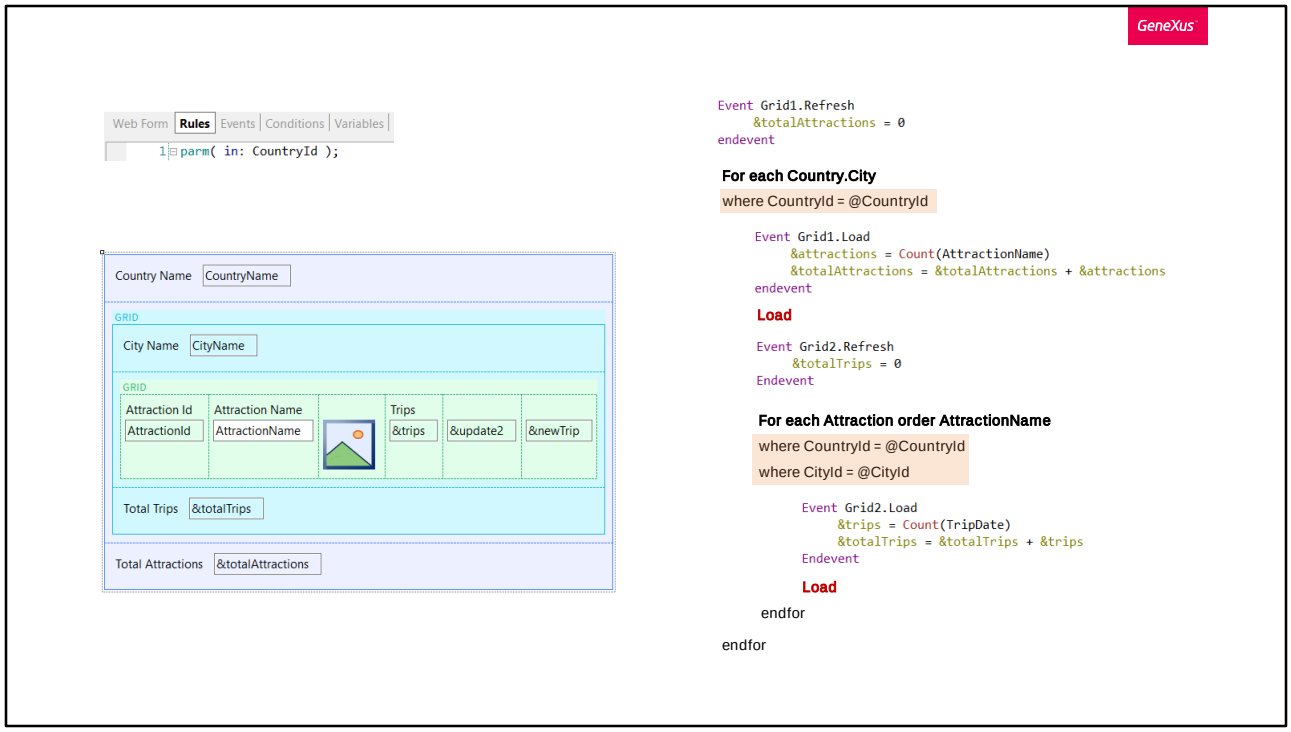
Web Form | Rules | Events | Conditions | Variables

```
parm( in: CountryId );
```




```
Event Start
     &newTrip = "New trip"
     &update2 = "UPDATE"
     CountryName.ForeColor = RGB(147,4,55) //DarkBase
     CountryName.FontBold = True
     CityName.FontBold = True
Endevent

Event Grid1.Refresh
     &totalAttractions = 0
endevent

Event Grid1.Load
     &attractions = Count(AttractionName)
     &totalAttractions = &totalAttractions + &attractions
endevent

Event Grid2.Refresh
     &totalTrips = 0
Endevent

Event Grid2.Load
     &trips = Count(TripDate)
     &totalTrips = &totalTrips + &trips
Endevent
```

Here, the variables are added to the screen to complete the rest you had previously. This will make the example identical, with the need for programming all the events in the system.



Translating this into a GeneXus pseudo-code would result in something like this.

First, the external grid's Refresh is executed. There, we set to zero the variable that will count the total number of attractions that will be loaded. Then, and because this is the case of a grid with the CountryCity base table, GeneXus will establish the implicit For each that will navigate this table, filtering by the value of CountryId received by parameter. For each record found, that grid's Load event will be executed. It will have that city's attractions, which it will add to the variable that will be totalized. Then the city is loaded on the grid, from the Load command that GeneXus includes.

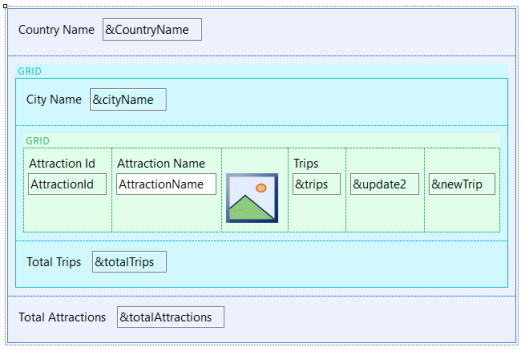
Immediately after, the Refresh of the nested grid is executed, leaving in zero the value of the total sum of trips that include that attractions that are loaded afterwards. And because it has base table, GeneXus writes another implicit For each to navigate that base table –Attraction– to which it adds all the corresponding clauses, according to what the developer made explicit in the grid's properties. In this case, you had only established base transaction and order clause.
Additionally, it adds the implicit conditions that precisely relate to the fact that this grid is nested with another one, and a relation between the table exists. For this reason, it will only go through the records of the attractions

table that match the country and city of the record loaded in the external For each. And for each of them, it will execute the Load of this nested grid. And then it will load the line on the grid.

GeneXus

Web Form | **Rules** | Events | Conditions | Variables

```
1  parm( in: CountryId );
```




```
Event Grid1.Refresh
    &totalAttractions = 0
endevent
```

```
Event Grid1.Load
    &attractions = Count(AttractionName)
    &totalAttractions = &totalAttractions + &attractions
endevent
```

```
Event Grid2.Refresh
    &totalTrips = 0
Endevent
```

For each Attraction order AttractionName

where CountryId = @CountryId

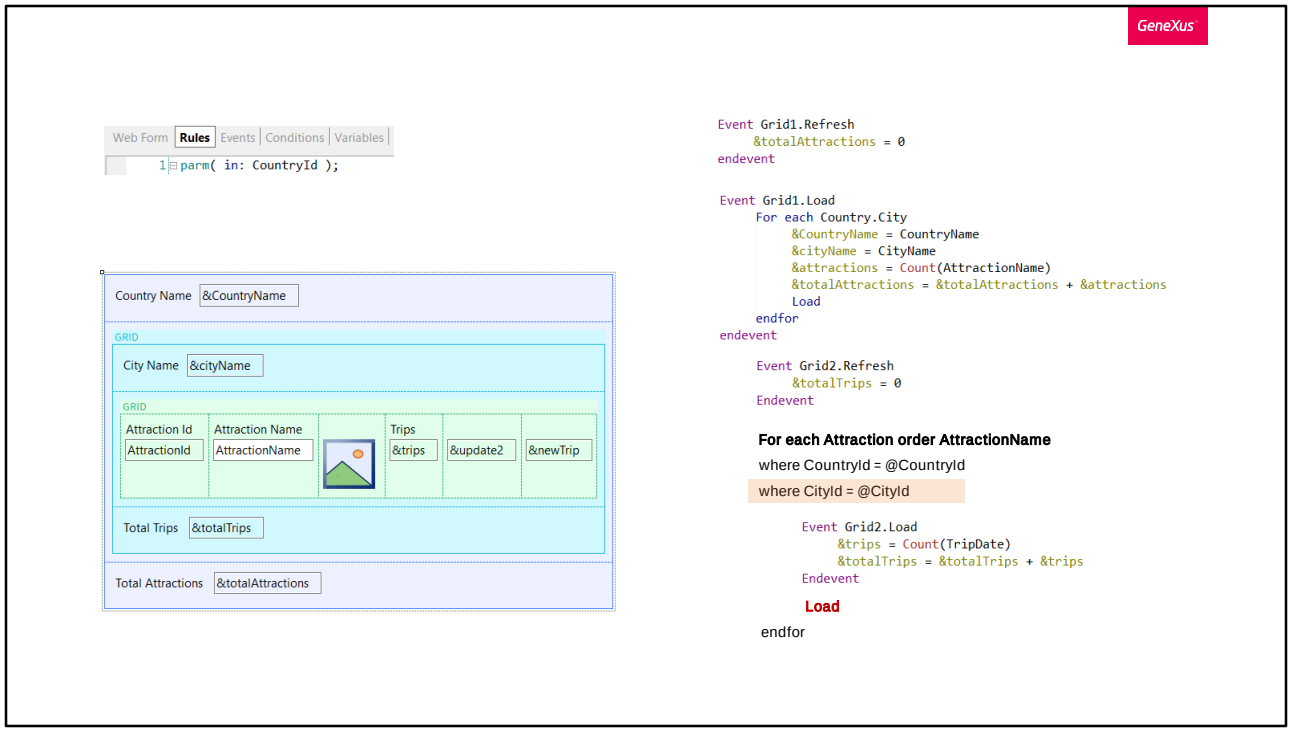
where CityId = @CityId

```
    Event Grid2.Load
        &trips = Count(TripDate)
        &totalTrips = &totalTrips + &trips
    Endevent
```

Load

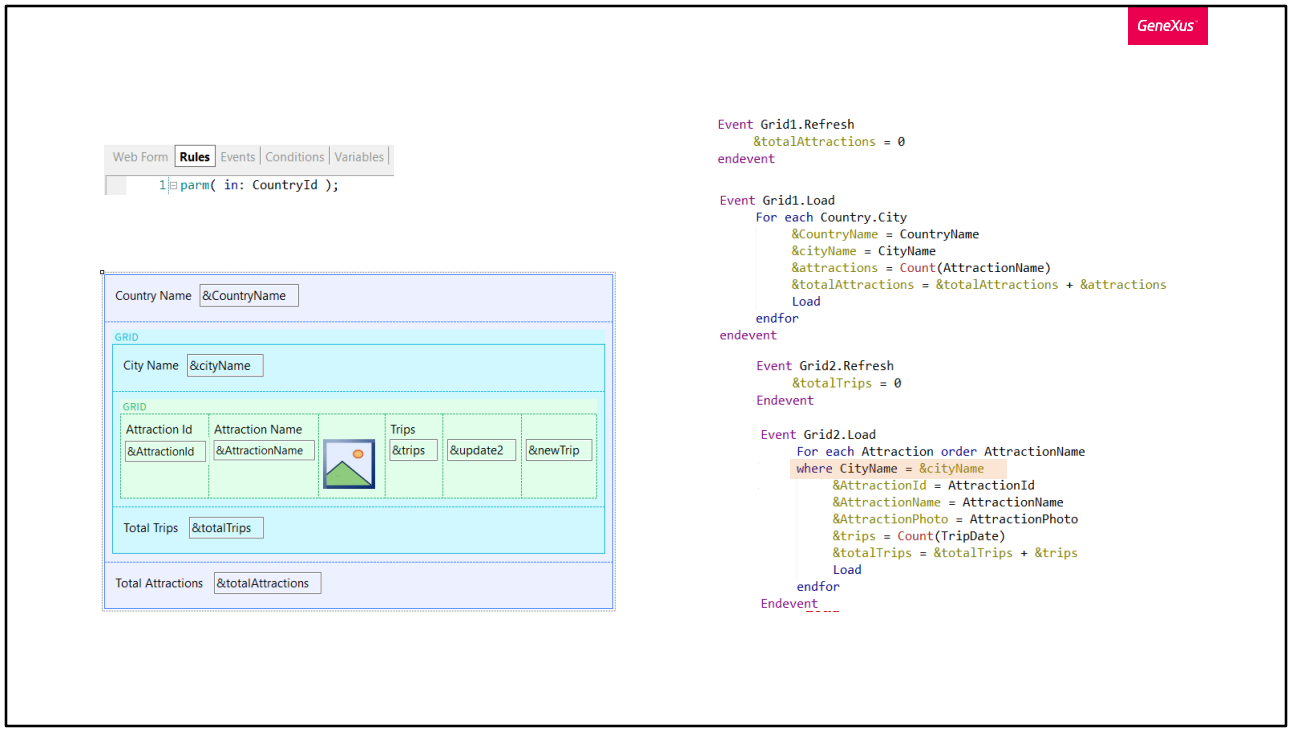
endfor

Of course, if the first grid had no base table, then the implicit For each disappears, as well as the Load command.



And we will have to explicitly write them in the grid's Load event.

In this case, upon finding the Load command inside the grid's Load event because the grids are nested, GeneXus will immediately trigger the Refresh and Load events of the nested grid. And depending, again, on whether the nested grid has base table or not, GeneXus will either establish an implicit For each and its Load or not. Except that in this case, you will have to the explicit the Where of cities, which was implicit before, as you will see next.

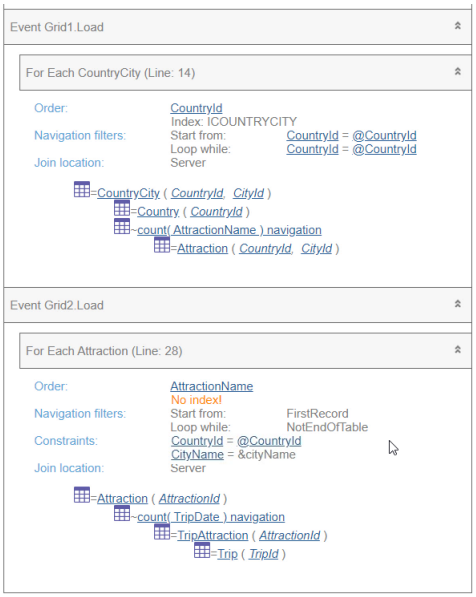


If you wanted that the nested grid not have base table, it is clear that you would substitute attributes by variables in the grid, and you will have to program the For each explicitly in the Load event, as well as in the Load command.

And precisely because there are not base tables, leaving the logic for loading grids fully up to the developer, GeneXus may not establish the automatic join between the For eachs, and for this reason you need to explicit the filters.

You will be establish only the filter by CityName, because the filter by CountryId will be carried out due to the parameter.

```
Event Grid1.Refresh
     &totalAttractions = 0
endevent

Event Grid1.Load
     For each Country.City
          &CountryName = CountryName
          &cityName = CityName
          &attractions = Count(AttractionName)
          &totalAttractions = &totalAttractions + &attractions
          Load
     endfor
endevent

Event Grid2.Refresh
     &totalTrips = 0
Endevent

Event Grid2.Load
     For each Attraction order AttractionName
     where CityName = &cityName
          &AttractionId = AttractionId
          &AttractionName = AttractionName
          &AttractionPhoto = AttractionPhoto
          &trips = Count(TripDate)
          &totalTrips = &totalTrips + &trips
          Load
     endfor
Endevent
```

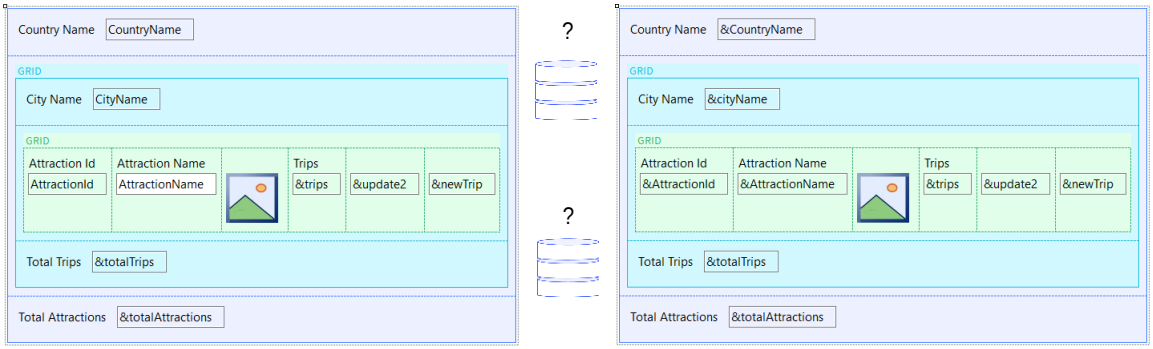


Here, it is implemented in GeneXus.

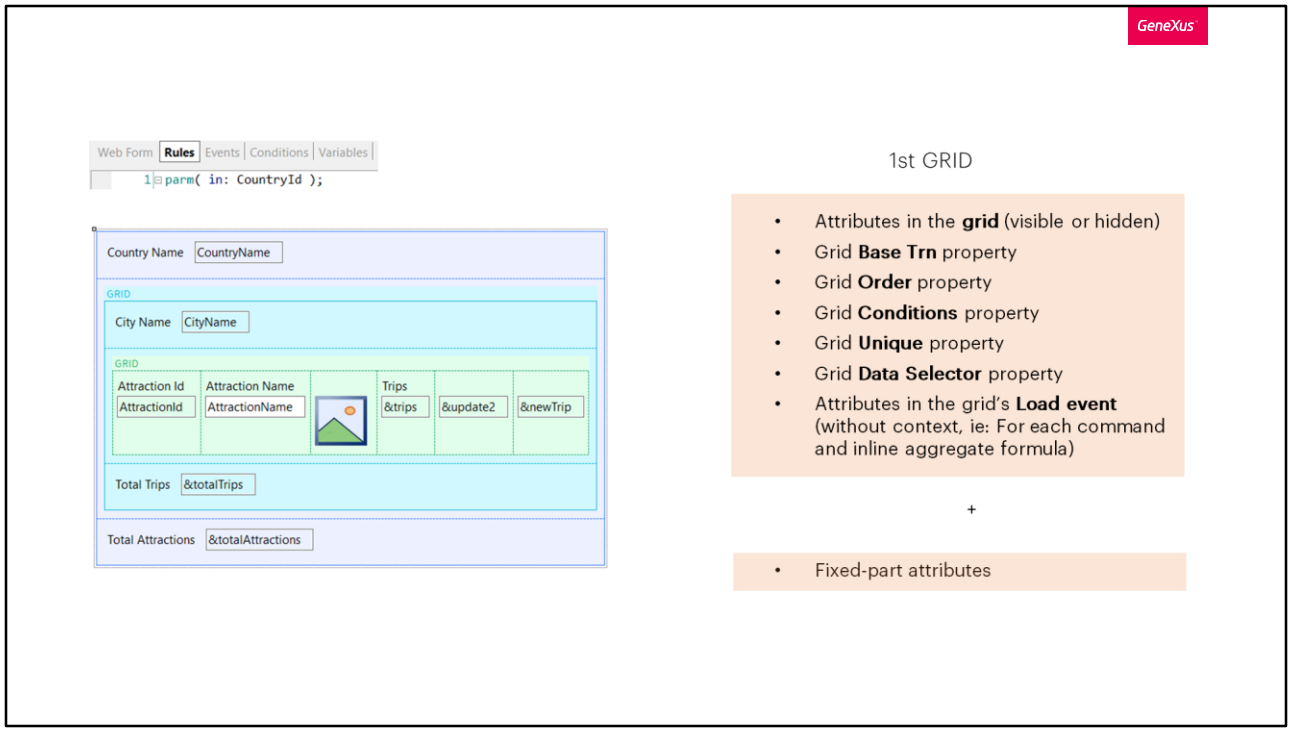
If you take a look at its navigation list, you will clearly see that it did not select base table for any of the two grids.
And if you execute... you will not see any difference with the web panel that had base tables for both grids.

GeneXus

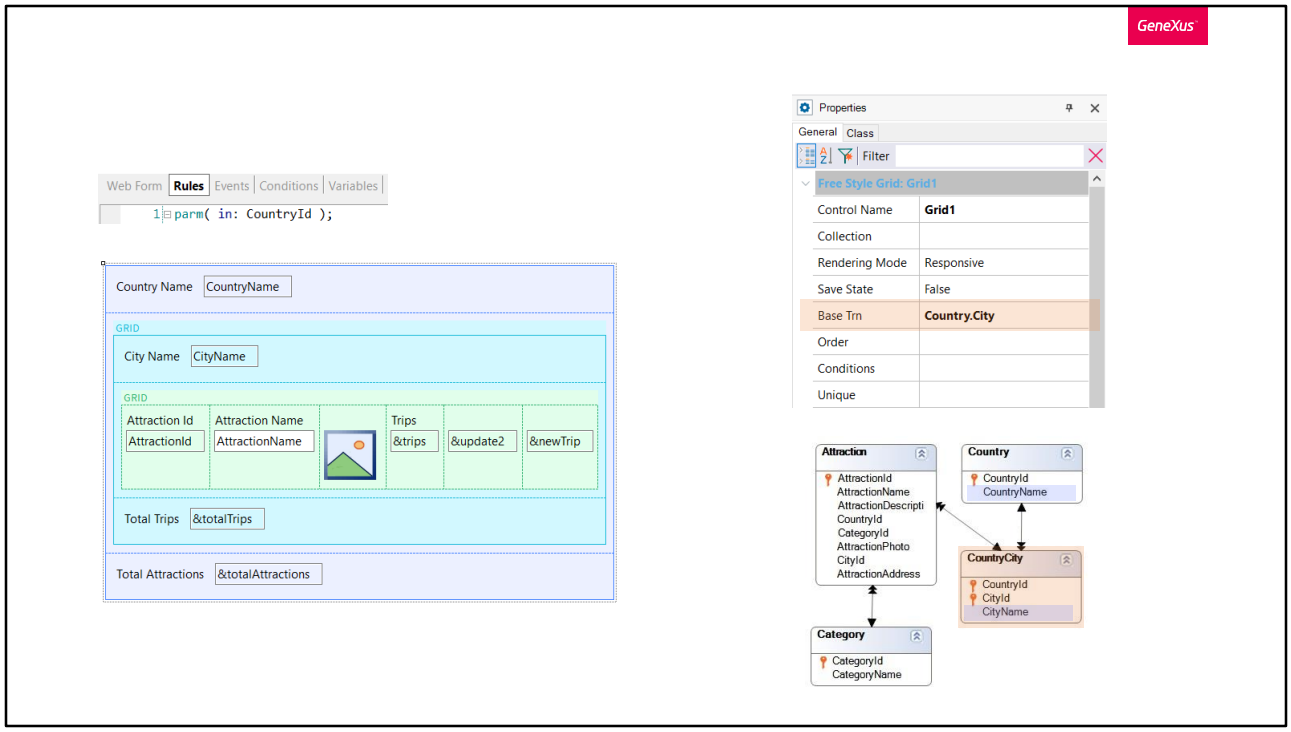
Base Tables



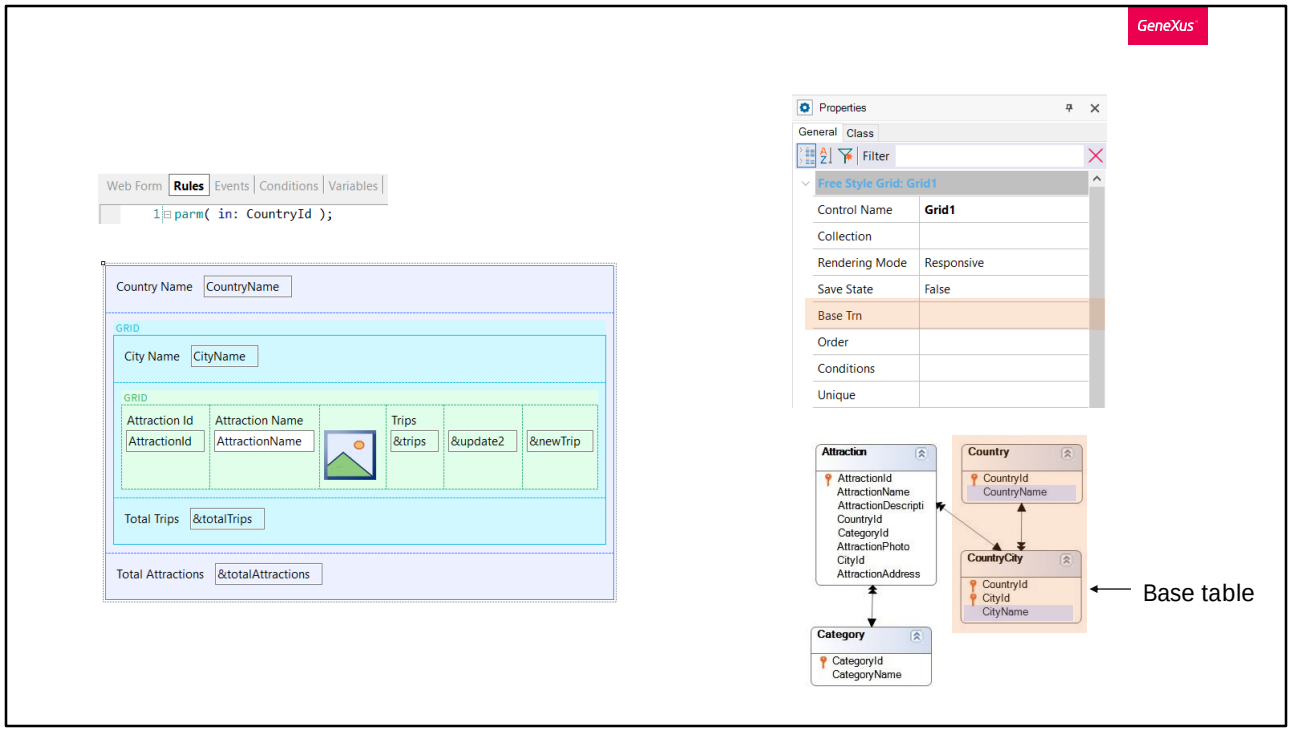
But to reach this navigations, you had to determine the base tables in the first place, just as it occurs with the For eachs.



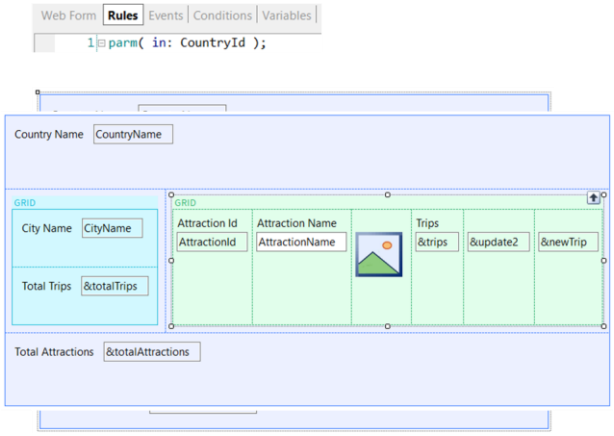
In order to determine the base table of the first grid on screen, the exact same thing that you saw for parallel grids will apply.
That is to say that, GeneXus takes into account the attributes of the grid itself, plus those corresponding to the fixed part of the screen. In addition, of course, to any that may appear in the grid's properties (Base transaction, Order, etc.) and those of the Load event for the grid. But neither those of the Refresh of the grid, nor those of any other event.



When a base transaction has been specified, it will have that base table, and all the attributes mentioned will have to belong to its extended table.



If no base transaction has been specified, then GeneXus will find the minimum extended table containing all the attributes mentioned and will then select its base table as the grid's base table.



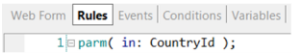
Nested GRID

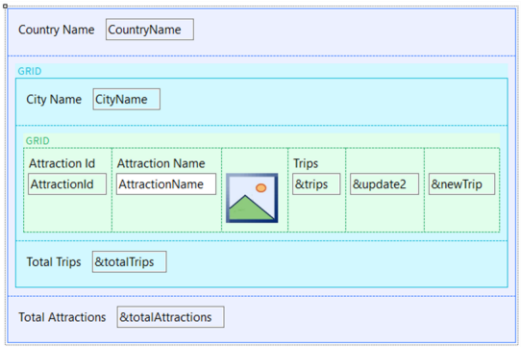
- Attributes in the **grid** (visible or hidden)
- Grid **Base Trn** property
- Grid **Order** property
- Grid **Conditions** property
- Grid **Unique** property
- Grid **Data Selector** property
- Attributes in the grid's **Load event** (without context, ie: For each command and inline aggregate formula)

The base table of the nested grid is determined as if the grid were parallel and not nested, except that it will be the same as for determining the base table of a nested For each.

When the nested grid HAS NO BASE TRANSACTION SPECIFIED, then GeneXus will determine it on its own, and here is where the fact that this grid is nested with another one may determine a table different from the one that would be determined if the grid were parallel.

GeneXus

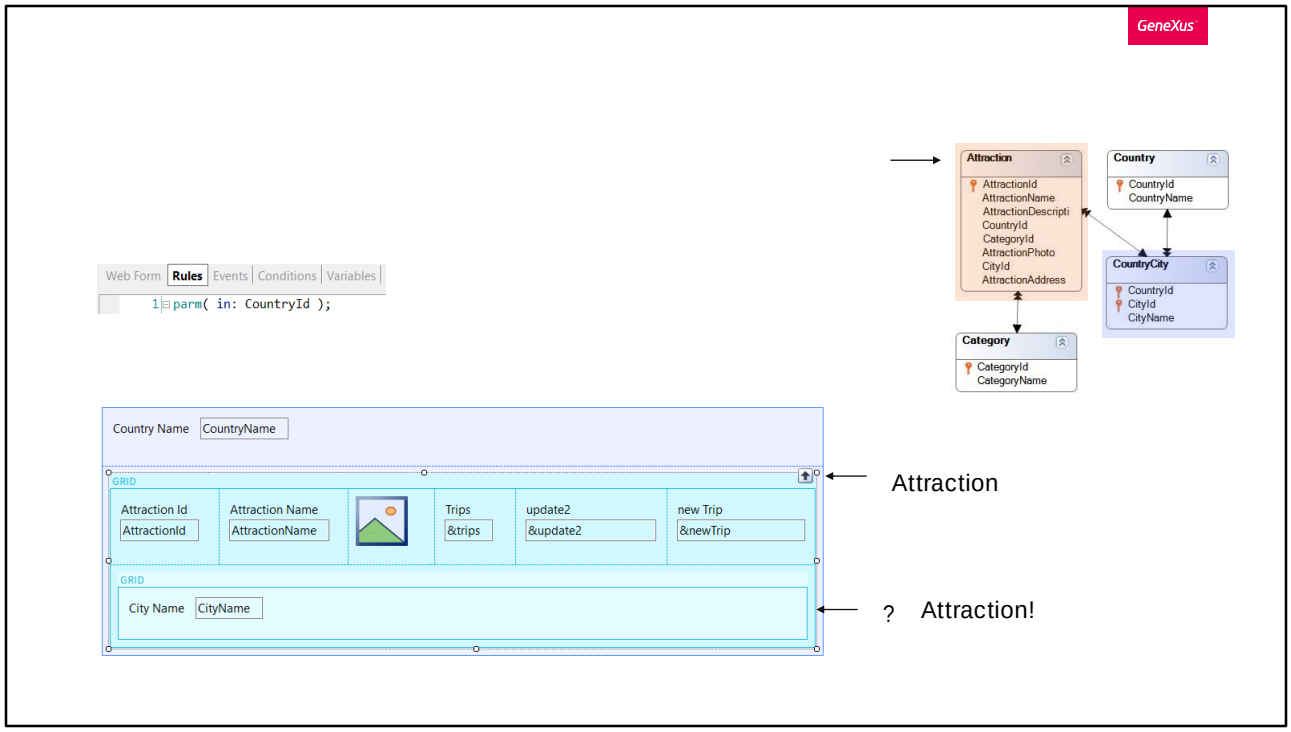




Nested GRID

- Attributes in the **grid** (visible or hidden)
- Grid **Base Trn** property
- Grid **Order** property
- Grid **Conditions** property
- Grid **Unique** property
- Grid **Data Selector** property
- Attributes in the grid's **Load event** (without context, ie: For each command and inline aggregate formula)

These cases are not frequent, but it's good to be aware of them.



For instance, if the grids were inverted and the external grid navigated the Attraction table, with no base transaction specified in the internal grid, with only the CityName attribute implied, and the grid being parallel, then it is clear that GeneXus would determine the cities table as its base table. However, in this case, and because it is nested with a grid that has an extended table that includes the attributes of the second grid, it will select for this second grid the same base table as that of the first grid, thus implementing a control break.

GeneXus

Base tables: ready!

And its navigations!

And this concludes the study of determining base tables and navigations for all cases of web panels.

GeneXus™

training.genexus.com
wiki.genexus.com