Web Panels

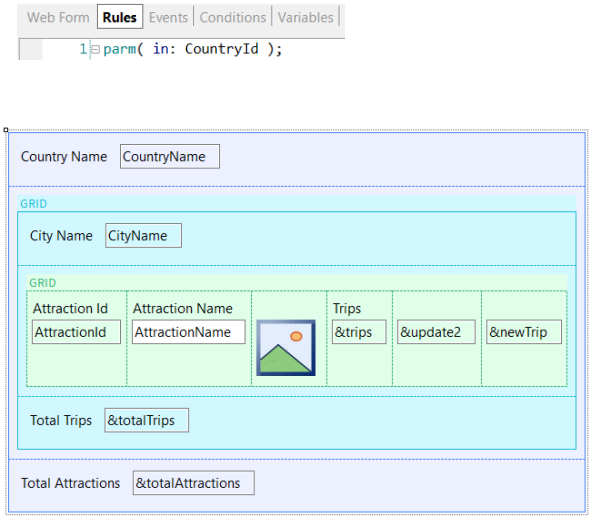
How to Implement Control Breaks in Nested Grids

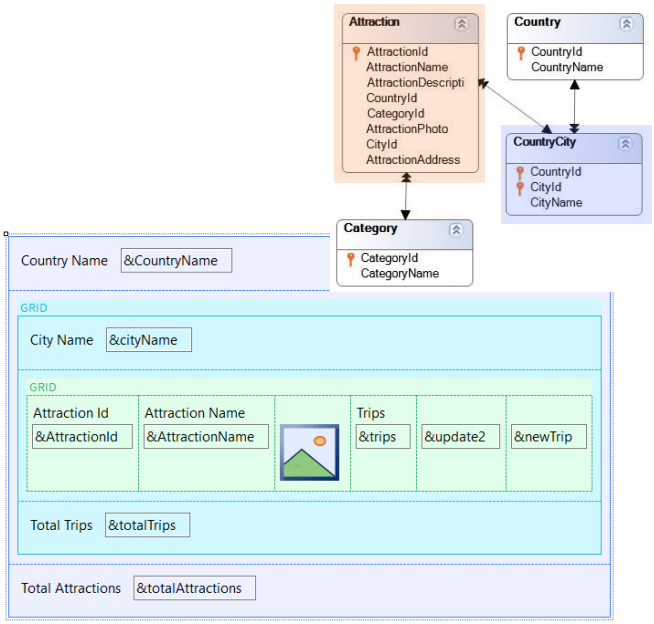
GeneXus™

Web Panel with SEVERAL Grids

In another video we learned how the base tables and the navigation of a web panel with several grids were determined.

With several Grids: nested





In particular, we saw an example of nested grids performing a **join**.

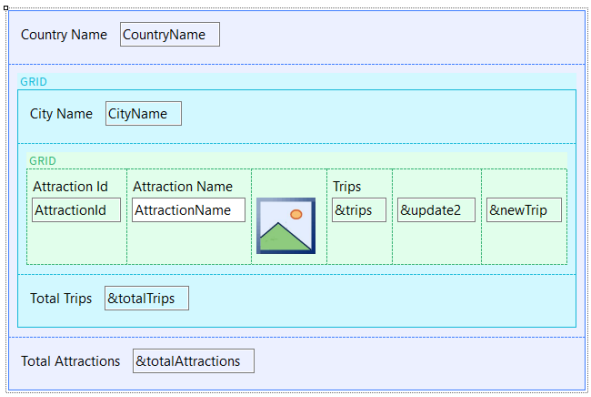
That is, the external grid ran through a table with a 1 to N relationship with the table run through by the internal grid, regardless of whether these grids were implemented with or without a base table.

In both solutions, the web panel received a country identifier in a parameter, and the external grid showed the cities of that country; that is to say, it was going to run through CountryCity; and the internal grid showed the tourist attractions of that city. In other words, it was going to run through Attraction.

With several Grids: nested

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

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



Grid1 and Grid2 **with** Base Tables

Pseudocode

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

For each Country.City

where CountryId = @CountryId

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

Load

```
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

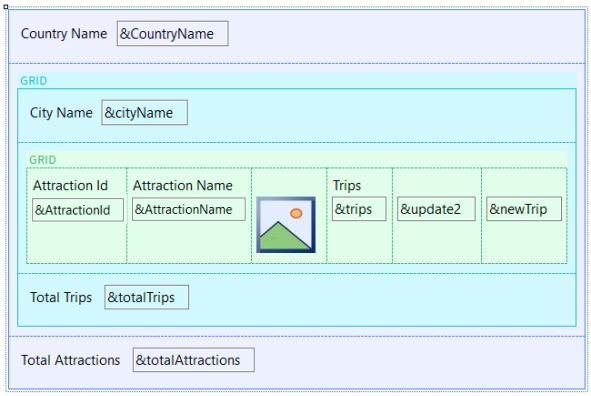
endfor

In the case of grids with a base table, this join was established automatically, without having to do anything. GeneXus detected it and added the filter in its source program.

With several Grids: nested

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

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

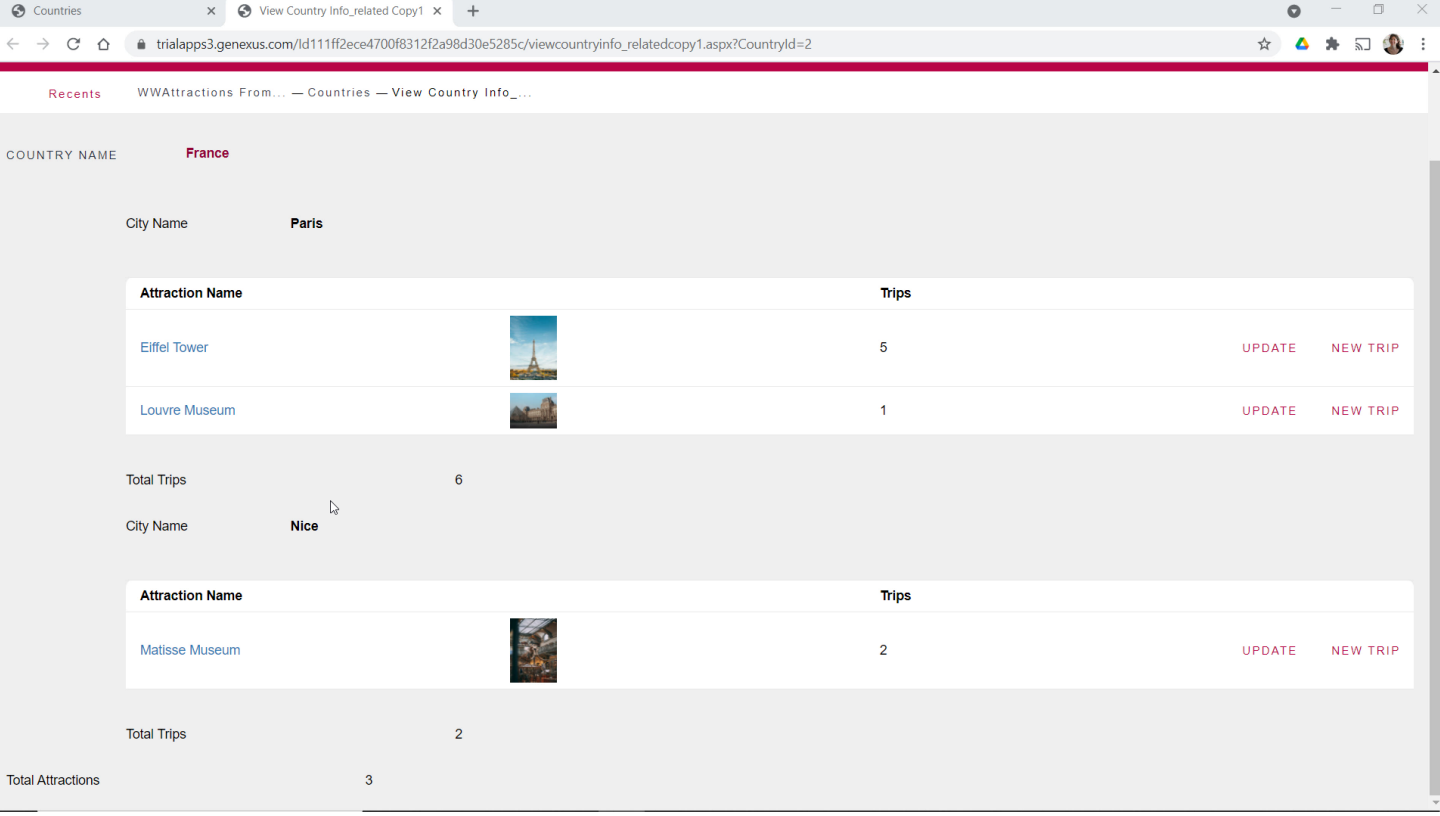


Grid1 and Grid2 **without** Base Tables

Pseudocode

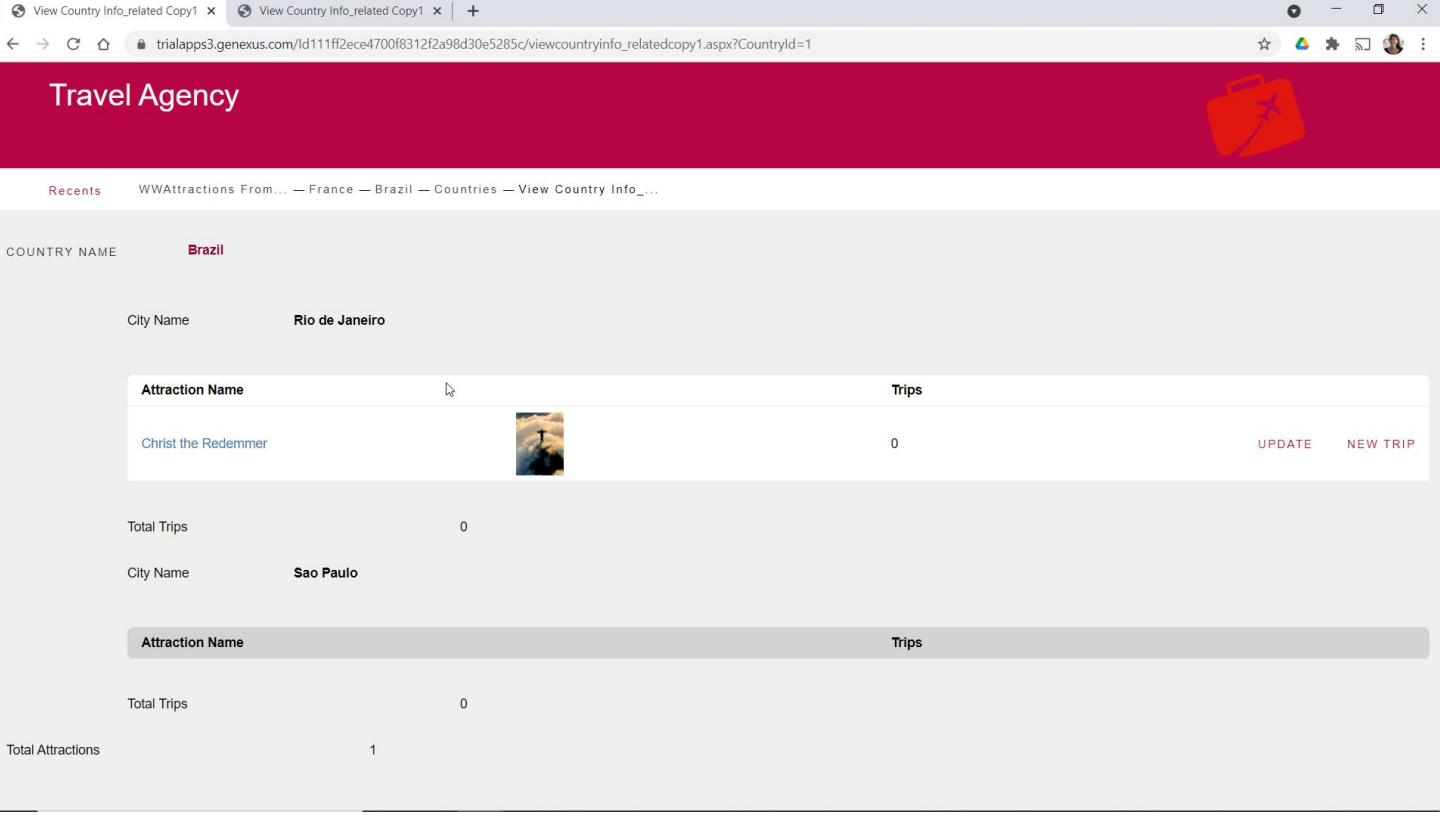
```
Event Grid1.Refresh
     &totalAttractions = 0
endevent

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
     endfor
```

On the other hand, in the case of grids without a base table, we had to specify it in the For each that we implemented to load the nested grid (and we only filtered by city, because the filter by country was already implicit by receiving it in a parameter in the CountryId attribute).



Here is the web panel with both grids **with a base table**. We have added an action in the countries Work With pattern to invoke this web panel.

If we choose France: here we see the attractions of Paris and Nice, which are the two cities we have entered in the system.



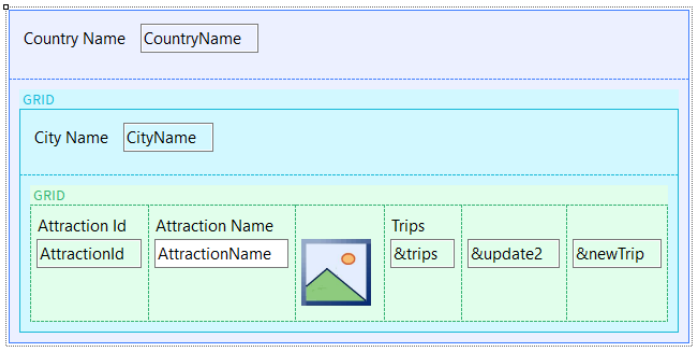
Now, let's see what happens if instead of choosing France we choose Brazil, for example, which also has two cities entered.

We see that for the first one, Rio de Janeiro, a tourist attraction is shown, but for the second one, Sao Paulo, no tourist attractions are shown. This is precisely because it is a **join**.

With several Grids: nested

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

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



Grid1.Refresh()

Grid1 → Load → **Rio de Janeiro**

Grid2.Refresh()

Grid2 → Load — Christ the Redeemer

Grid1 → Load() → **Sao Paulo**

Grid2.Refresh()

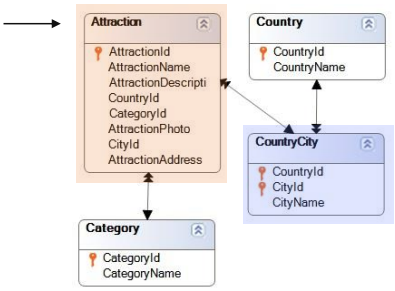
Control Break instead of Join

The Refresh of the external grid will be executed first, and then, once positioned in the first city, Rio de Janeiro, it will be loaded in Grid1; right after that, the Refresh of the nested grid will be executed, and then the attractions of Rio de Janeiro will be loaded, which in this case is only one, Christ the Redeemer.

Then the next city, Sao Paulo, will be loaded and the nested grid will be refreshed. But when running through the table of attractions to load only those of Sao Paulo, none will be found.

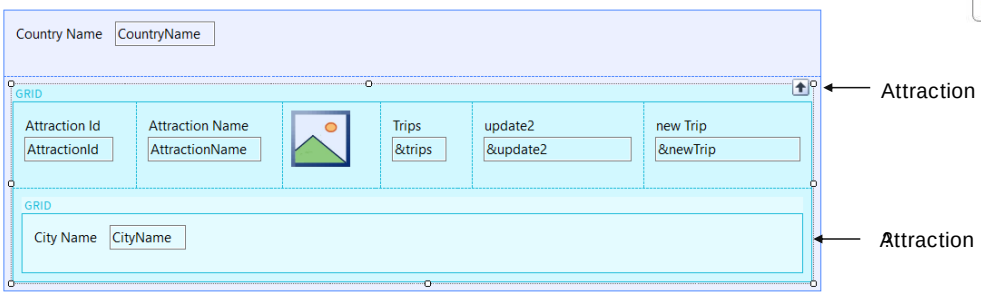
In order to show only cities with attractions, we need to implement a **control break** and not a **join**.

With several Grids: nested



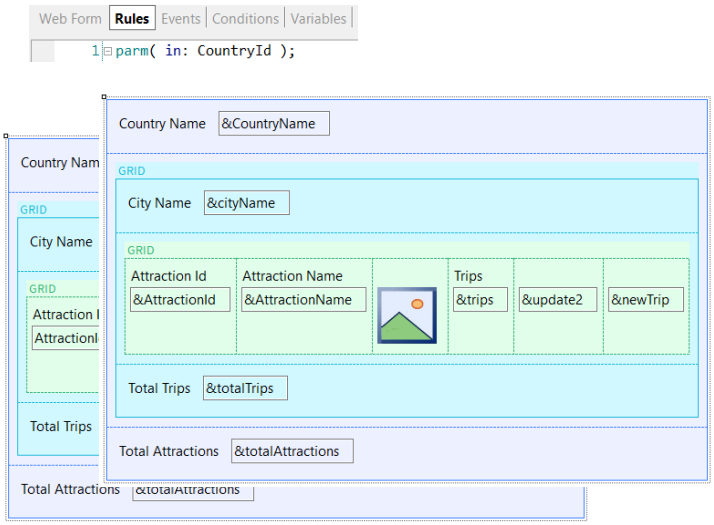
Web Form | **Rules** | Events | Conditions | Variables

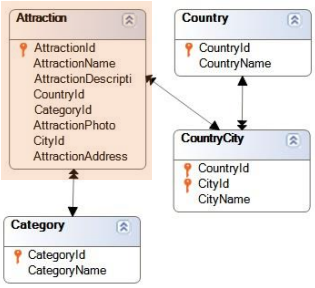
```
1 parm( in: CountryId );
```



We had outlined a case where an unwanted control break would occur instead of a join. But we had only addressed that by not specifying a base transaction for the nested grid. It appeared to be CountryCity, but actually it would be Attraction. We hadn't looked closely at the navigation, and now we will do so with our example.

With several Grids: nested






```
For each Attraction order CountryId, CityId

    print CountryName
    print CityName

    For each Attraction
        print AttractionName, AttractionPhoto, etc.
    endfor

endfor
```

We need, as in the case of a listing, that both For each commands (either implicit, i.e. coming from grids **with** a base table, or explicit, i.e. coming from grids **without** a base table) have the same base table, Attraction. And that the first one makes up the group that is going to make the break by country/city.

To do so, it is enough to modify the **base transaction** and add the **order** to the first grid.

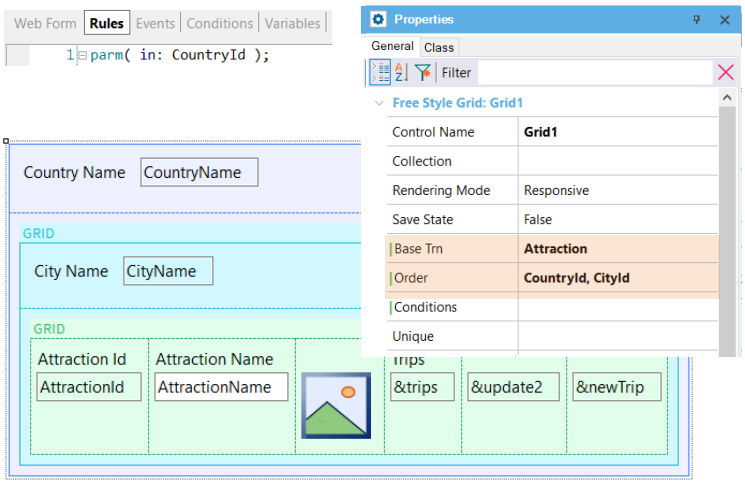
It will depend on whether grids were implemented with or without a base table, to see how to do it.

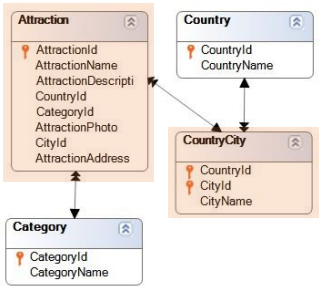
WITH Base Tables

Let's start by the case in which the web panel was implemented with both grids with a base table.

With several Grids: nested

Grid1 and Grid2 with Base Tables

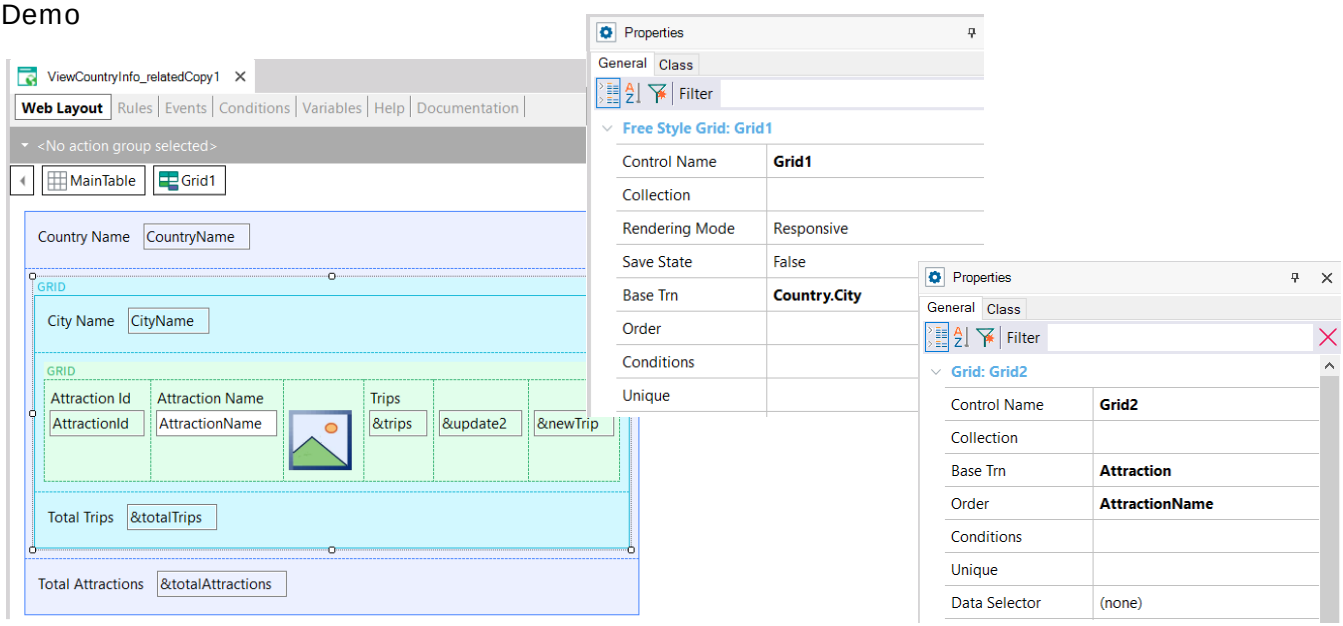





```
For each Attraction order CountryId, CityId

    print CountryName
    print CityName

    For each Attraction
        print AttractionName, AttractionPhoto, etc.
    endfor

endfor
```

In this case, the objective is achieved by changing the properties that we see of the external grid, for these others.

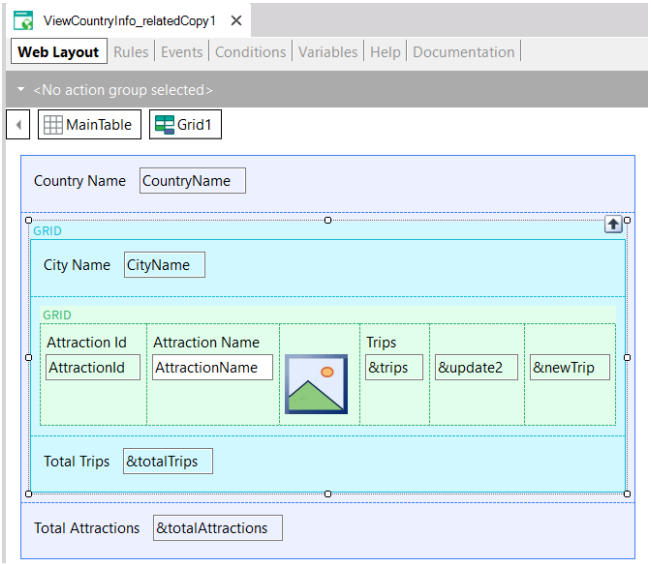
Demo



Before doing it, let's go to GeneXus. We see the properties of both grids: the first one has a Base Trn Country.City, and no order.

The second one has Base Trn Attraction and an order by AttractionName.

Demo

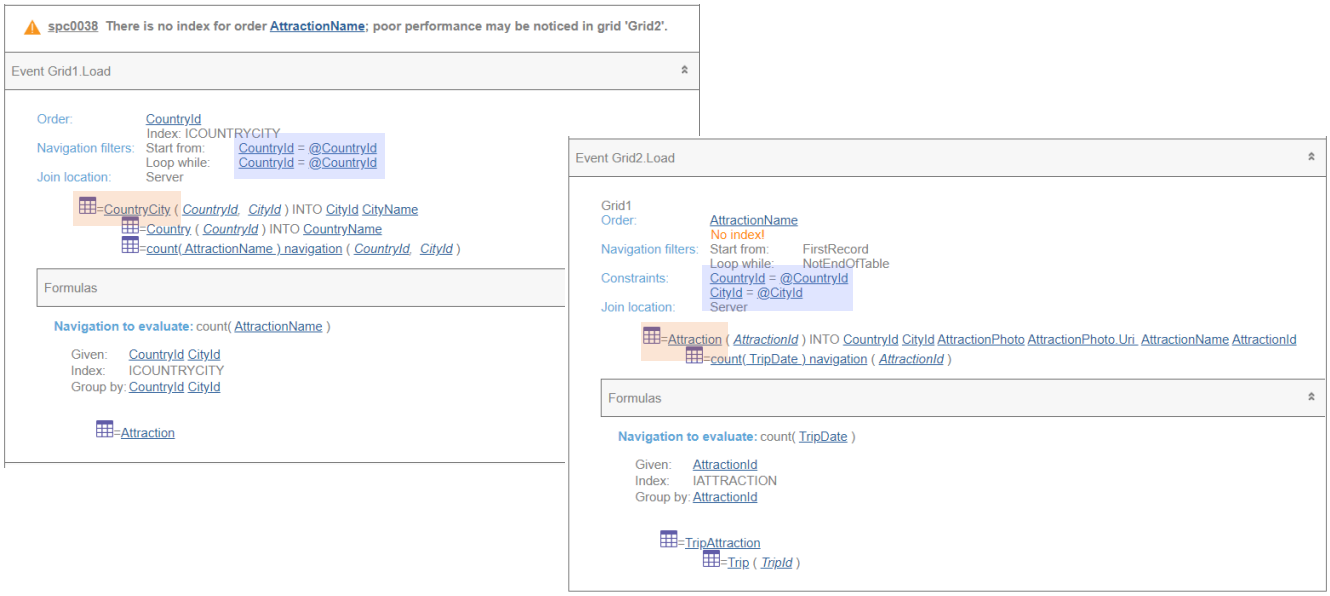



```
Event Grid1.Refresh
    &totalAttractions = 0
endevent

Event Grid2.Refresh
    &totalTrips = 0
Endevent

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

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

Event &update2.Click
    Attraction(trnMode.Update, AttractionId)
Endevent

Event &newTrip.Click
    &trips = NewTrip(AttractionId)
    Refresh
endevent

Event AttractionName.Click
    ViewAttractionFromScratch(AttractionId)
Endevent
```

And in the events we see that the Refresh and Load events have been programmed for each grid, only to initialize and sum or count in variables (which will show the total number of attractions loaded and the total number of trips in which those attractions are included).

We also have user events to call various objects. They don't matter at all here.

Demo



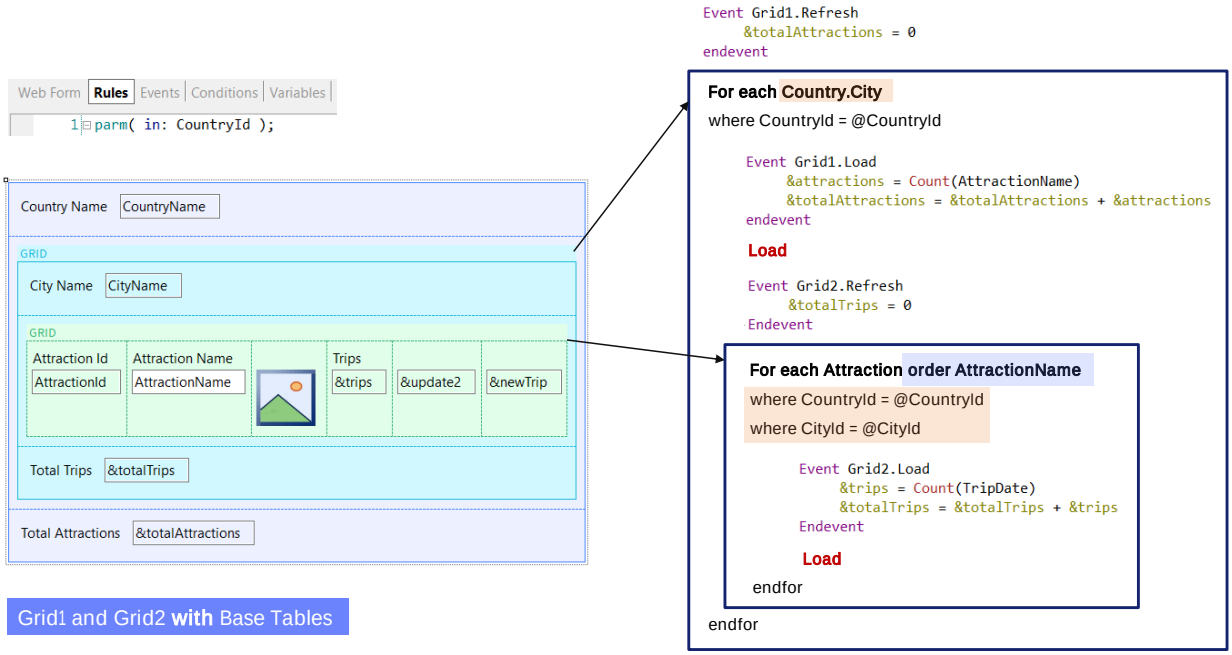
In GeneXus, let's look at the navigation list of the web panel.

We can clearly see the base table of the first grid: CountryCity, which will filter by the country received in a parameter. Next, we go to the nested Load, which has Attraction as a base table, and filters by the country and city of the external grid.

Of course, we see the join

Also, note that it orders the first implicit For each by CountryId, and the second one by AttractionName, for which it reports that there is no index.

With several Grids: nested



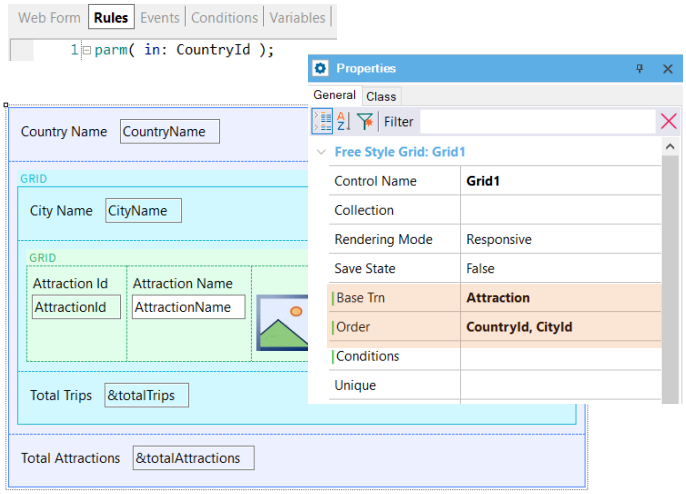
Grid1 and Grid2 **with** Base Tables

The pseudocode of the source that GeneXus will program will be similar to the one displayed. In it, the Base Transaction property of Grid1 was used to program the base Transaction of the implicit For each.

In the second For each, the order clause set was the content of the grid's Order property, which was AttractionName, and that is why we saw those selected indexes.

In short, the Refresh of the external grid will be triggered and then the implicit For each that we are seeing, which together with the internal one will make up a **join**.

With several Grids: nested



Grid1 and Grid2 **with** Base Tables

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

For each Attraction order CountryId, CityId
where CountryId = @CountryId

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

    Load

    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
endfor
```

Now, let's modify the properties of Grid1, the external one.

In doing so, the Base Transaction property of the grid will cause the Base Transaction of the implicit For each to be modified. And the Order property will become the order clause of that For each.

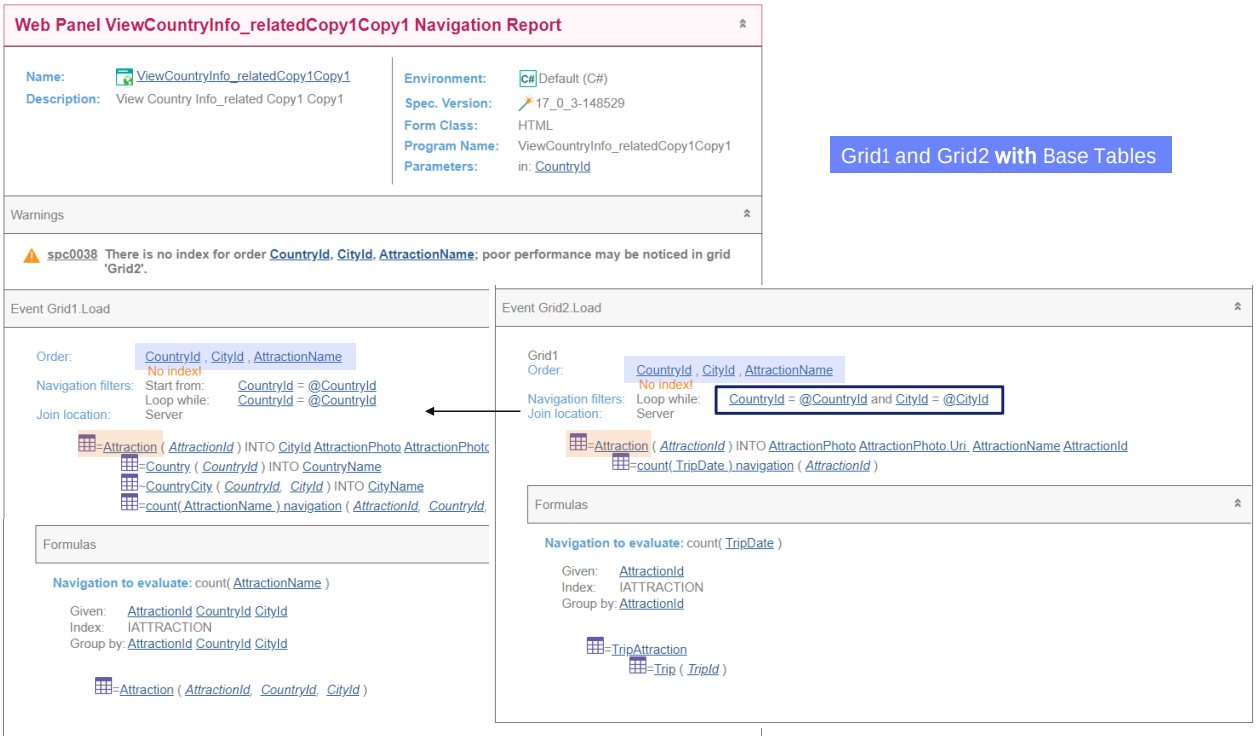
This will be enough, because the Refresh of the external grid will be triggered and then the implicit For each that we are seeing, which together with the internal one will make up a **control break**. Therefore, tourist attractions are going to be grouped by cities of the country (the filter by CountryId is due to the parameter).

Therefore, for **each group** of tourist attractions formed by each city, the Load event will be triggered once. Then the first line will be loaded in the grid, with the CityName of the city of the first group of attractions (and of course, its CountryName, which will be the same for all of them).

Next, the Refresh event of the nested grid will be triggered, after which the implicit for each will be executed, and it will only run through the attractions of the group, i.e., those of <u>that</u> country and city. For each one of these attractions in the group it will trigger the Load event and the Load command to load it into the nested grid.

And so on with all the groups.

Demo

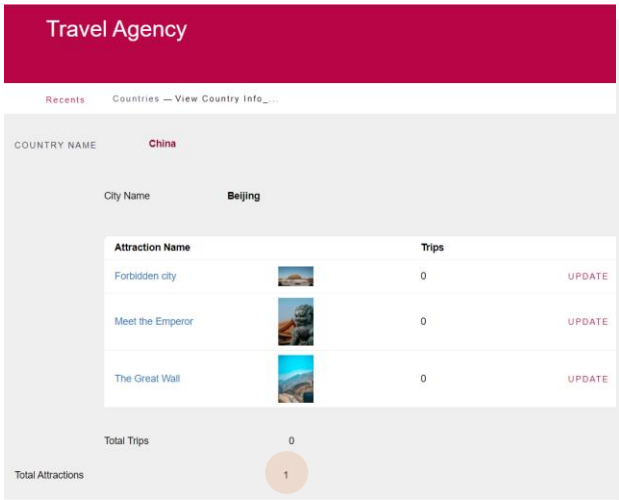


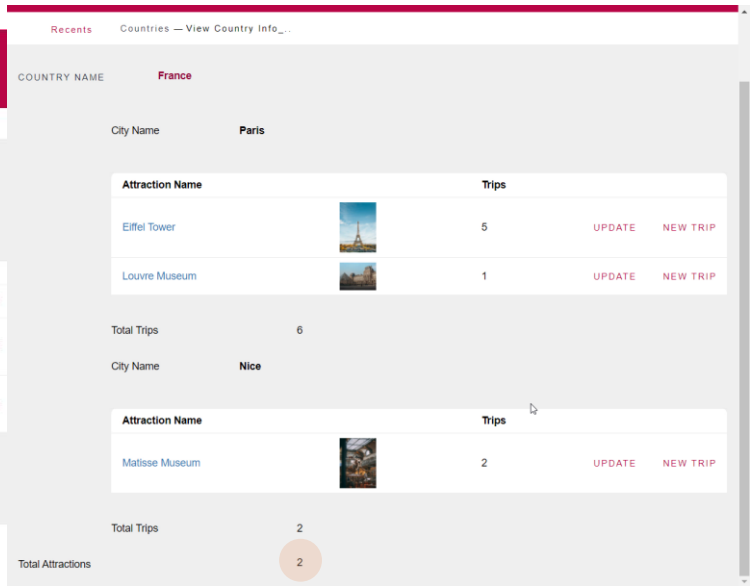
Grid1 and Grid2 **with** Base Tables

If we look at the navigation list, we can clearly see that both grids will navigate the same table, Attraction, using an index made up of the attributes of the Order property of the first grid, plus the order of the nested grid.

And we can clearly see that for the nested grid only the attractions corresponding to the country and city of the first grid will be run through.

Demo





If now we try it at runtime... we see exactly what we wanted.

Take China, for example. Perfect.

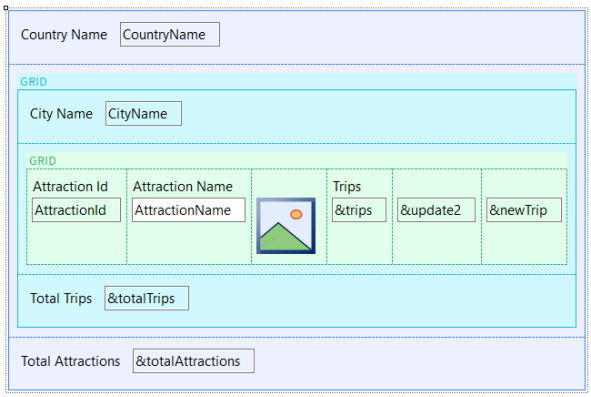
And if we go to France... in this case we don't notice any differences with the case of a join.

But the attractions are not being counted correctly. Why?

With several Grids: nested

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

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



Grid1 and Grid2 **with** Base Tables

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

For each Attraction order CountryId, CityId
where CountryId = @CountryId

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

Load

```
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

endfor

In Grid1 we are using the Count formula to count the attractions corresponding to that country and city. This worked when the base table of the For each was CountryCity, but not now that it is Attraction.

Demo

Web Panel ViewCountryInfo_relatedCopy1Copy1 Navigation Report

Name: ViewCountryInfo_relatedCopy1Copy1
Description: View Country Info_related Copy1 Copy1

Environment: C# Default (C#)
Spec. Version: 17_0_3-148529
Form Class: HTML
Program Name: ViewCountryInfo_relatedCopy1Copy1
Parameters: in: CountryId

Grid1 and Grid2 **with** Base Tables

Warnings

spc0038 **There is no index for order CountryId, CityId, AttractionName; poor performance may be noticed in grid 'Grid2'.**

Event Grid1.Load

Order: CountryId , CityId , AttractionName
No index!
Navigation filters: Start from: CountryId = @CountryId
Loop while: CountryId = @CountryId
Join location: Server

=Attraction (*AttractionId*) INTO CityId AttractionPhoto AttractionPhot
=Country (*CountryId*) INTO CountryName
=CountryCity (*CountryId*, *CityId*) INTO CityName
=count(AttractionName) navigation (*AttractionId*, *CountryId*

Formulas

Navigation to evaluate: count(AttractionName)

Given: AttractionId CountryId CityId
Index: IATTRACTION
Group by: AttractionId CountryId CityId

=Attraction (*AttractionId*, *CountryId*, *CityId*)

Event Grid2.Load

Grid1
Order: CountryId , CityId , AttractionName
No index!
Navigation filters: Loop while: CountryId = @CountryId and CityId = @CityId
Join location: Server

=Attraction (*AttractionId*) INTO AttractionPhoto AttractionPhoto.Uri, AttractionName AttractionId
=count(TripDate) navigation (*AttractionId*)

Formulas

Navigation to evaluate: count(TripDate)

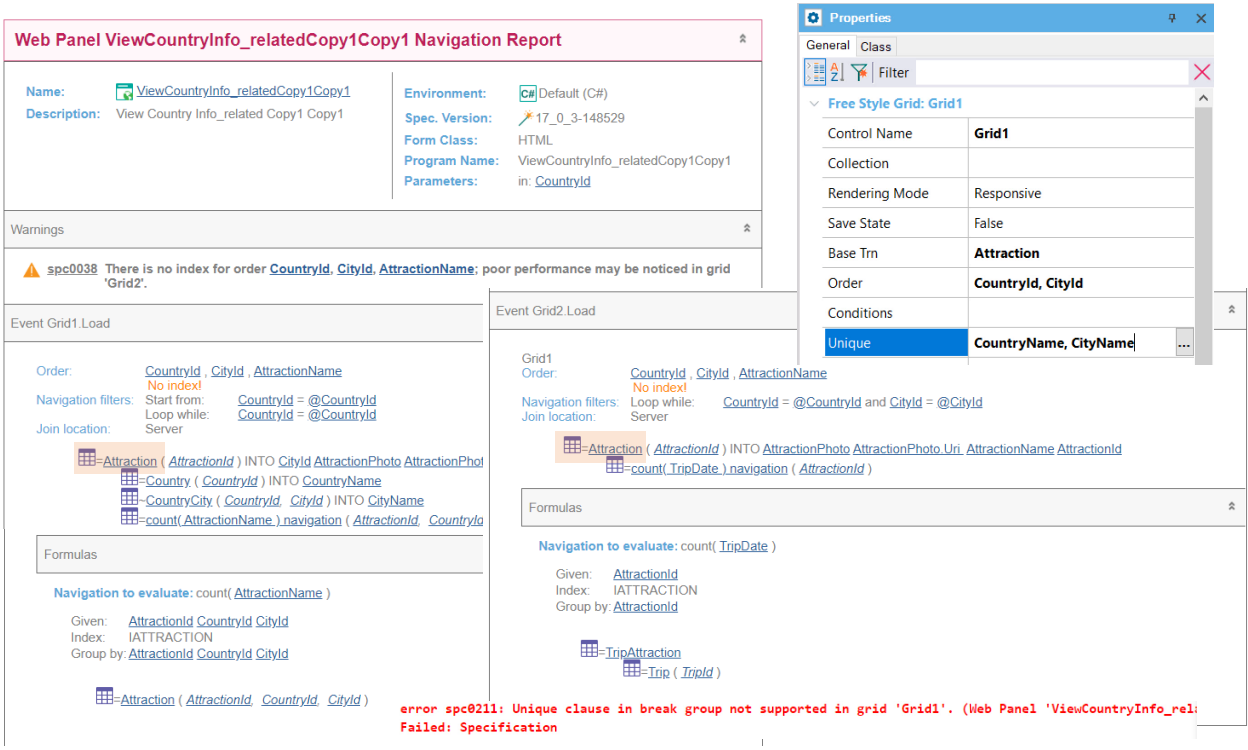
Given: AttractionId
Index: IATTRACTION
Group by: AttractionId

=TripAttraction
=Trip (*TripId*)

The problem with the formula of the first grid is clear in the navigation list.

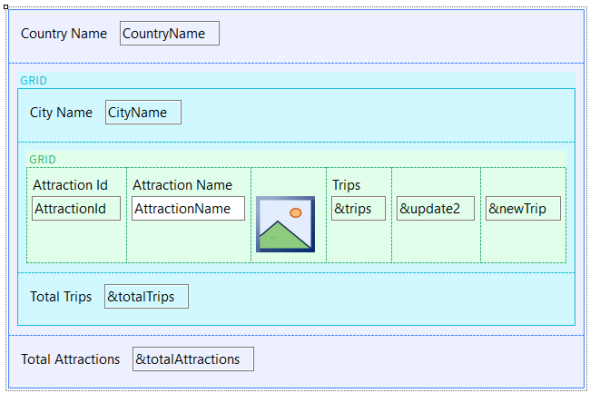
We can't run through a table and make an aggregation on the same table.

Demo



For this to work we would have to use the **unique clause**, which in this case is not useful, because it is not supported for control breaks (we're talking about the control break between grid 1 and grid 2).

With several Grids: nested



Grid1 and Grid2 **with** Base Tables

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

For each Attraction order CountryId, CityId
where CountryId = @CountryId

```
Event Grid1.Load
    &attractions = 0
    for each Attraction
        &attractions = &attractions + 1
    endfor
    &totalAttractions = &totalAttractions + &attractions
endevent
```

Load

```
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

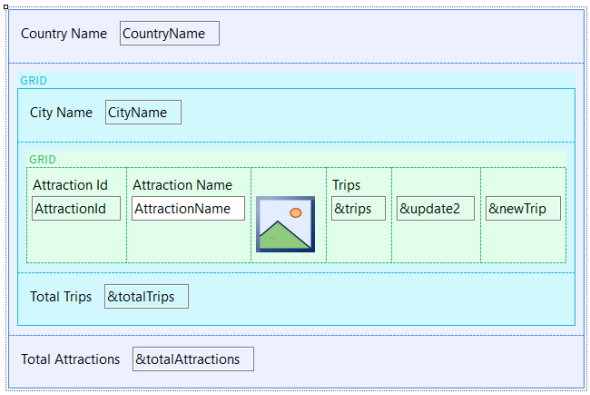
endfor

endfor

Therefore, we could make this calculation with another For each, by implementing another control break nested to the outermost For each.

But this is not what we need. We would have **one** control break **split** in two instances, but the first one would run through all the attractions of the city, and the second one would have no attractions to run through.

With several Grids: nested



Grid1 and Grid2 **with** Base Tables

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

For each Attraction order CountryId, CityId
where CountryId = @CountryId

Load

```
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
    &totalAttractions = &totalAttractions + 1
Endevent
```

Load

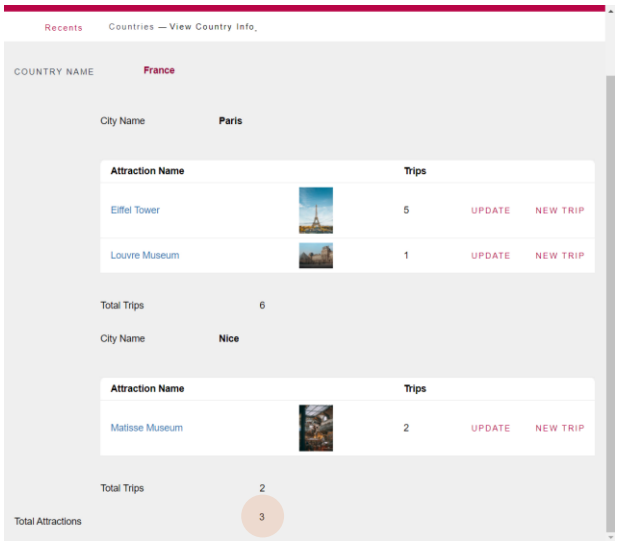
endfor

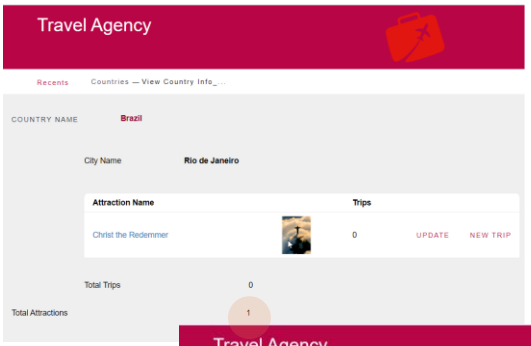
endfor

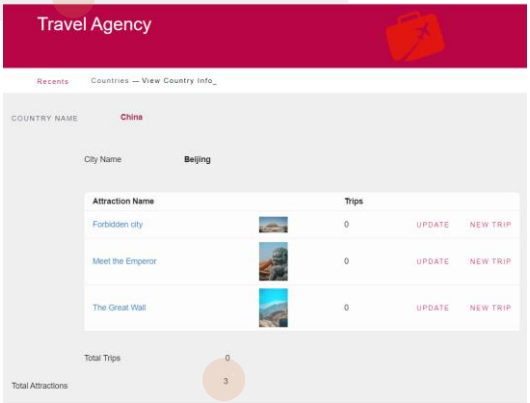
We have a much simpler solution: since the number of attractions will be the sum of all the records loaded in Grid 2, we could have deleted the Load event from Grid 1, and added the sum to the Load of Grid 2, without reinitializing the variable other than in the Refresh...

This makes it much simpler.

Demo







If we run... for example, for France, with its two cities, no difference can be seen with the implementation with a join, because both cities have attractions.

But if we choose Brazil, we do see the difference. Or China, for example.

And now the attractions are being counted correctly.

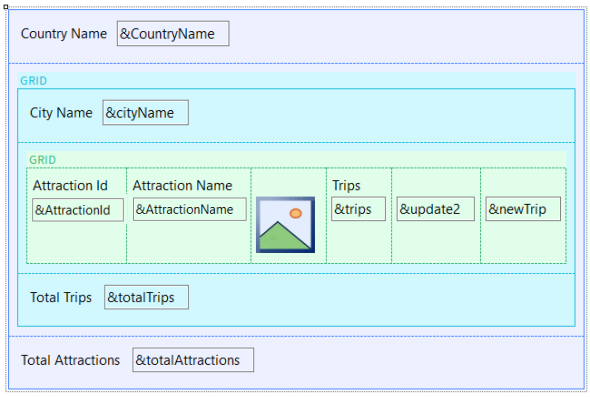
WITHOUT Base Tables

Now let's move on to the implementation case without a base table.

With several Grids: nested

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

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



Grid1 and Grid2 **without** Base Tables

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

Event Grid1.Load
    For each Country.City
        &CountryName = CountryName
        &cityName = CityName
        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
        &totalAttractions = &totalAttractions + 1
        Load
    endfor
Endevent
```

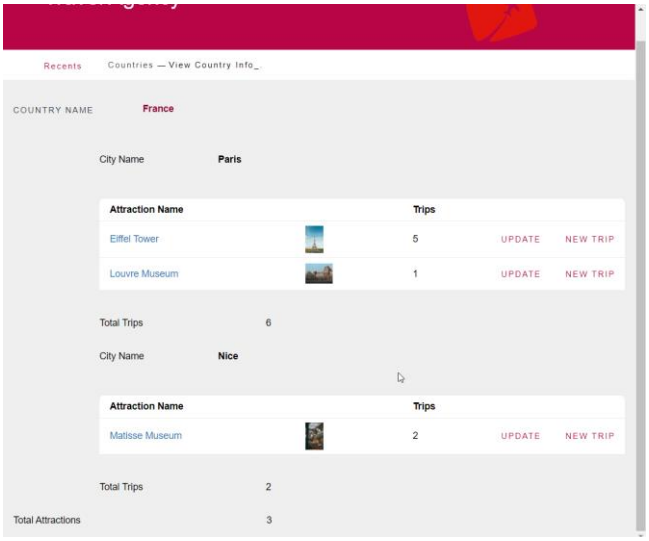
We had this web panel that implemented the same join as in the beginning, but without base tables. Note that in the screen we only have variables and there are no attributes in the properties of any of the grids.

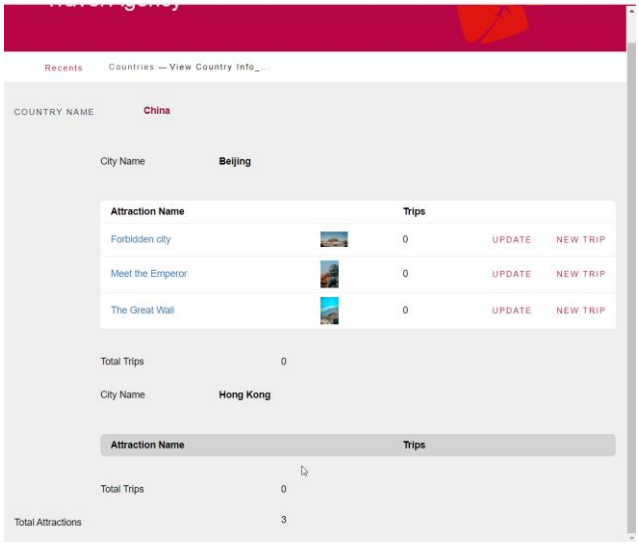
In the events, we explicitly performed the database loading. Let's do a Save As to leave this one as it was, with a join. And implement the control break in another. We take this opportunity to remove the Count of attractions from the first Load, and count in the second one, to make everything simpler.

Now let's modify the Work With Country action to invoke <u>this</u> web panel.

Demo

Grid1 and Grid2 **without** Base Tables



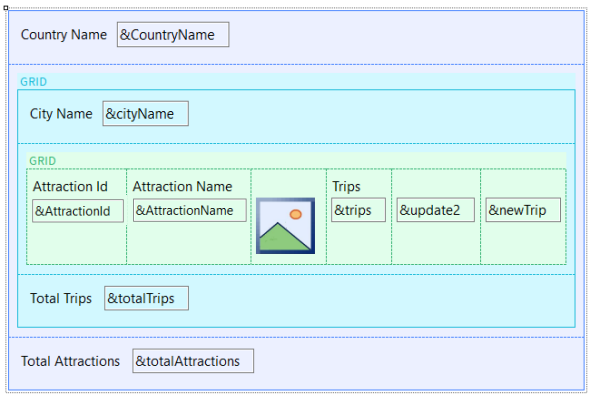


Let's run... and see the attractions of France. And now those of China. The **join** is clearly noticeable and not the **control break**.

With several Grids: nested

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

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



Grid1 and Grid2 **without** Base Tables

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

Event Grid1.Load
     For each Country.City
          &CountryName = CountryName
          &cityName = CityName
          Load
     endfor
endevent
```

Control Break?

Do 'Grid2'

```
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
          &totalAttractions = &totalAttractions + 1
          Load
     endfor
Endevent
```

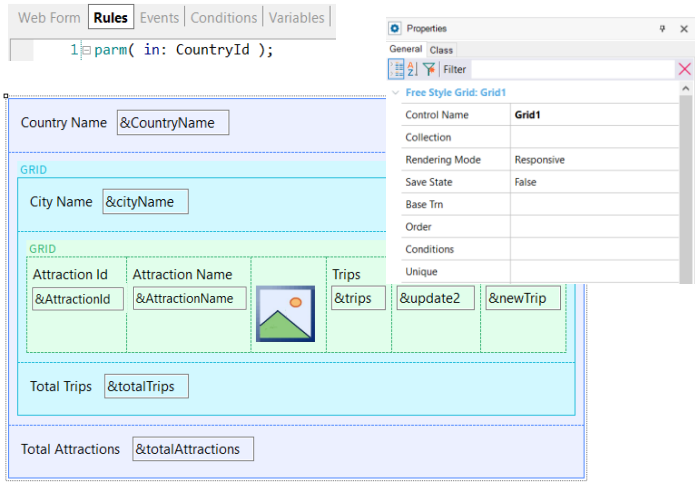
Even though the Load command of the first grid triggers the Refresh event, and immediately after that the Load of the second grid, it doesn't really nest the For each commands. It's as if a subroutine were invoked, as if the For each of the nested grid were executed in isolation.

That's why GeneXus is not finding an automatic join, and we had to explicitly filter the attractions of the city that was loaded in the &cityName variable, which was loaded by the Load event that invoked the Load of the nested grid. We didn't have to also place a filter by CountryId because it is instantiated in the parameter.

Let's keep this in mind, because it will make this case less obvious than it might seem at first.

The question is: How do we make the For each corresponding to grid1 change its base table to Attraction and establish a control break by country, city?

With several Grids: nested



Grid1 and Grid2 **without** Base Tables

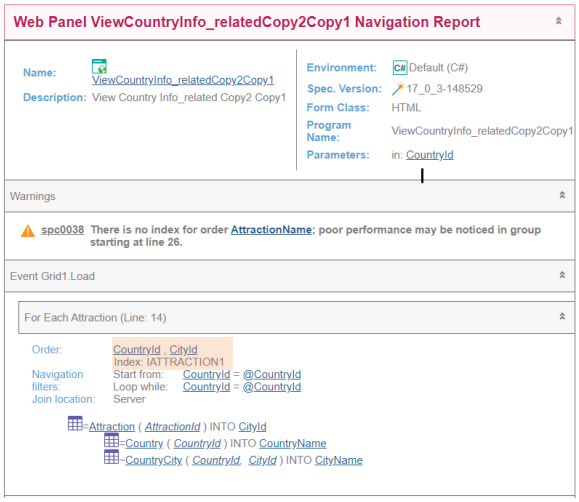
```
Event Grid1.Refresh
     &totalAttractions = 0
endevent

Event Grid1.Load
     For each Attraction order CountryId, CityId
          &CountryName = CountryName
          &cityName = CityName
          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
          &totalAttractions = &totalAttractions + 1
          Load
     endfor
Endevent
```

It won't be by adding Base Trn or Order to the properties of Grid1 (because if we did that, we would transform the implementation into one with a base table), but to the explicit For each of the Load event of Grid1.

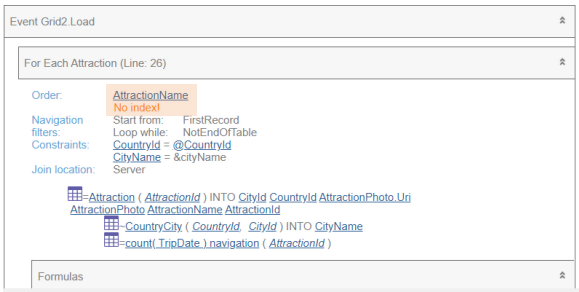
Therefore, it seems obvious that the first thing to do is to modify the first For each so that the Base Transaction is Attraction...

And it would also seem obvious that we should place an order clause to establish the grouping criteria by which we want the control break to be established in relation to the For each of Grid2.

Demo



This is not a Control Break!



However, if we look at the navigation list...

It seems a bit odd, and although every For each apparently does what it should do, it didn't choose the same order for each one, in order to use a single index and make the run through more efficient. Something is not right.

It obviously didn't understand that it will have to make a control break.

View Country Info_related Copy1

trialapps3.genexus.com/Id111ff2ece4700f8312f2a98d30e5285c/viewcountryinfo_relatedcopy1.aspx?CountryId=2

Recents Countries — View Country Info_...

COUNTRY NAME **France**

City Name **Paris**

Attraction Name		Trips		
Eiffel Tower		5	UPDATE	NEW TRIP
Louvre Museum		1	UPDATE	NEW TRIP

Total Trips 6

City Name **Nice**

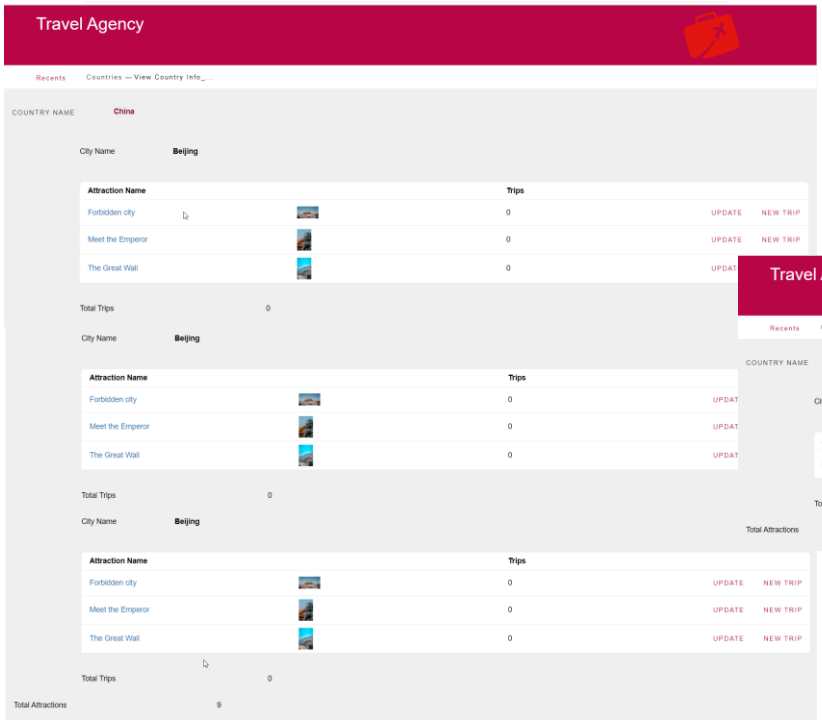
Attraction Name		Trips		
Matisse Museum		2	UPDATE	NEW TRIP

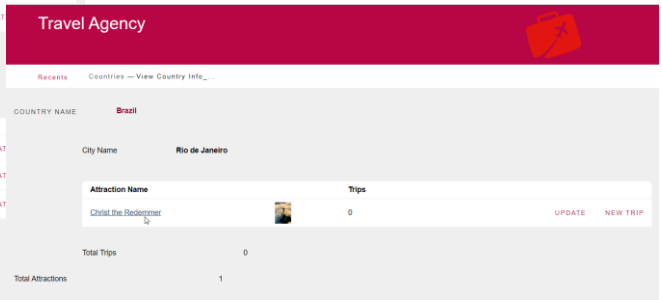
Total Trips 2

Total Attractions 3

And we can confirm this by running it.

Note that for the attractions in France, Paris comes up twice, corresponding to the two Paris attractions that are available.





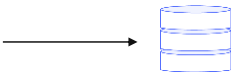
For those in China, Beijing comes up three times, corresponding to the three Beijing attractions.

And for Brazil only once, matching the Rio attractions registered.

What's going on?

With several Grids: nested

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

Event Grid1.Load
     For each Attraction order CountryId, CityId
          &CountryName = CountryName
          &cityName = CityName
          Load

          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
                    &totalAttractions = &totalAttractions + 1
                    Load
               endfor
          Endevent

     endfor
endevent
```



Attraction navigated once!

Obviously, it doesn’t realize that it has to make a control break.

This is because, as we have already mentioned, it doesn't **really** nest the navigations.

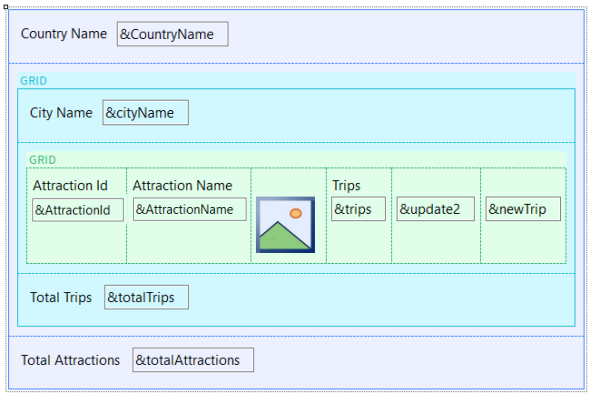
With several Grids: nested

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

Event Grid1.Load
     For each Attraction order CountryId, CityId
          &CountryName = CountryName
          &cityName = CityName
          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
          &totalAttractions = &totalAttractions + 1
          Load
     endfor
Endevent
```

It's as if they were two independent For each commands, only that from one the execution of the other is invoked, but through two separate queries to the database.

With several Grids: nested

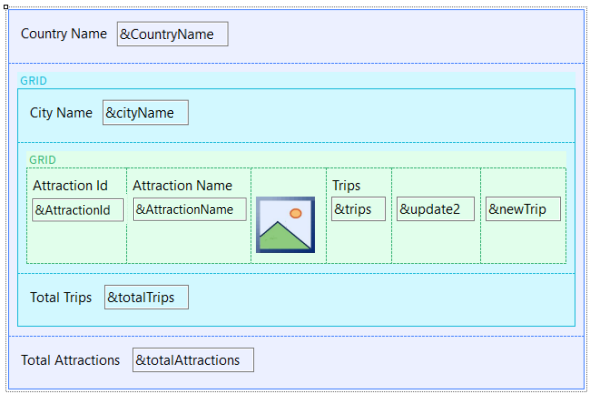


Grid1 and Grid2 **without** Base Tables

Control Break could not be implemented between nested grids

This is the equivalent to saying that we cannot **really** implement a control break between two nested grids without base tables.

With several Grids: nested




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

Event Grid1.Load

    For each Attraction order CountryId, CityId
        unique CountryName, CityName
        &CountryName = CountryName
        &cityName = CityName
        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
        &totalAttractions = &totalAttractions + 1
        Load
    endfor
Endevent
```

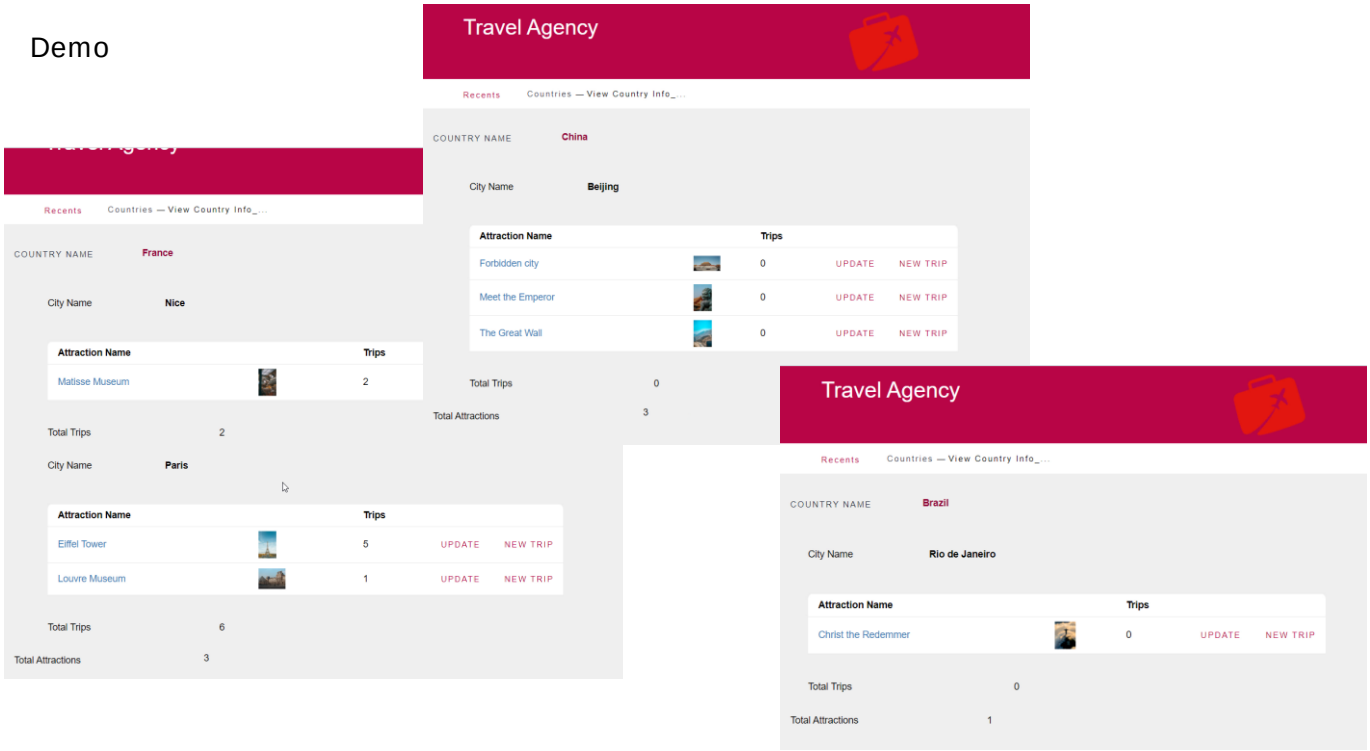
The solution we have for the moment is to use, for the first For each, the unique clause. In other words, if several records in the Attraction table have the same country and city, it will keep only one of them. And for this one, load the variables and execute the Refresh event, and right after that the Load of grid2, which will execute its For each as if it were completely independent of the previous one. And that is precisely why this time it will allow us to use the unique clause.

Demo

```
spc0038 There is no index for order AttractionName; poor performance may be noticed in group
        starting at line 27.

Event Grid1.Load

  For Each Attraction (Line: 14)

    Order:              CountryId , CityId
                        Index: IATTRACTION1
    Unique:             CountryName , CityName
    Navigation          Start from:    CountryId = @CountryId
    filters:            Loop while:    CountryId = @CountryId
    Join location:      Server

        =Attraction ( AttractionId ) INTO CityId
            =Country ( CountryId ) INTO CountryName
            ~CountryCity ( CountryId, CityId ) INTO CityName

Event Grid2.Load

  For Each Attraction (Line: 27)

    Order:              AttractionName
                        No index!
    Navigation          Start from:    FirstRecord
    filters:            Loop while:    NotEndOfTable
    Constraints:        CountryId = @CountryId
                        CityName = &cityName
    Join location:      Server

        =Attraction ( AttractionId ) INTO CityId CountryId AttractionPhoto.Uri
        AttractionPhoto AttractionName AttractionId
            ~CountryCity ( CountryId, CityId ) INTO CityName
            =count( TripDate ) navigation ( AttractionId )
```

If now we look at the navigation list... it seems that it will work this way.

Demo



We run it…

We have succeeded.

WITH or WITHOUT Base Tables?

Grid1 and Grid2 **with** Base Tables

Free Style Grid: Grid1

Control Name	**Grid1**
Collection	
Rendering Mode	Responsive
Save State	False
Base Trn	**Attraction**
Order	**CountryId, CityId**
Conditions	
Unique	

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

Event Grid2.Refresh
     &totalTrips = 0
Endevent

Event Grid2.Load
     &trips = Count(TripDate)
     &totalTrips = &totalTrips + &trips
     &totalAttractions = &totalAttractions + 1
Endevent
```

Grid1 and Grid2 **without** Base Tables

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

 Event Grid1.Load

      For each Attraction order CountryId, CityId
           unique CountryName, CityName
           &CountryName = CountryName
           &cityName = CityName
           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
          &totalAttractions = &totalAttractions + 1
          Load
     endfor
Endevent
```

So, for the moment, we are finding it much easier to implement a control break when grids **have a base table**.
Strictly speaking, it will be a **true control break** only in that case.

In the second case, when grids don't have a base table, we are just simulating it. Actually, there will be two independent queries to the Attraction table and not a single one that solves everything, as it happens in the real control break.

We encourage you to try everything we have seen.

GeneXus™

training.genexus.com
wiki.genexus.com
training.genexus.com/certifications