Telas web

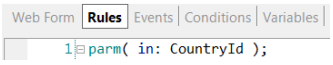
Tabelas base e navegações em Web panel com múltiplos grids

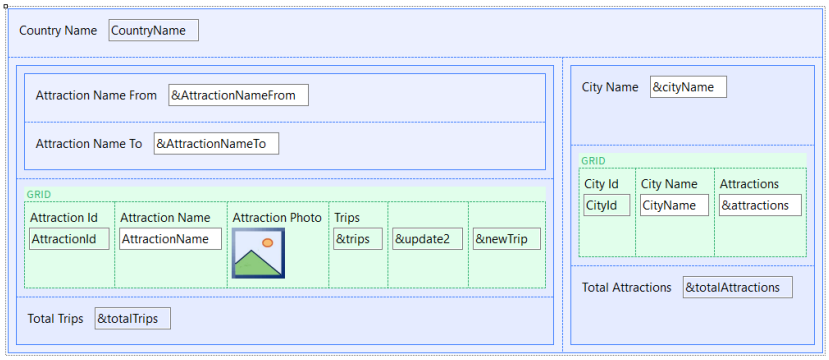
GeneXus™

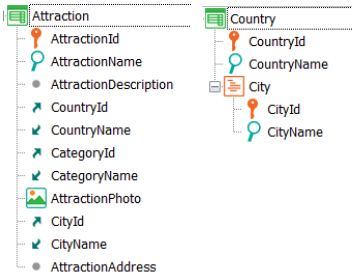
Web Panel with SEVERAL Grids

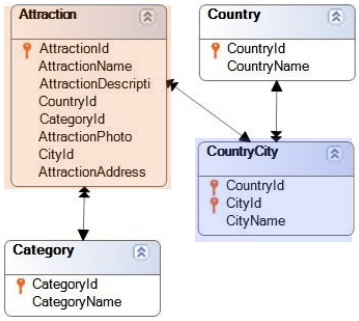
Agora, o que acontece quando um web panel tem mais de um grid? Obviamente não poderemos mais falar de tabela base do web panel, mas de cada grid.

With several Grids: parallels










```
Event Grid1.Refresh
     &totalTrips = 0
Endevent

Event Grid1.Load
     &trips = Count(TripDate)
     &totalTrips = &totalTrips + &trips
Endevent
```

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

Event Grid2.Load
     &attractions = Count(AttractionName)
     &totalAttractions = &totalAttractions + &attractions
endevent
```

E a determinação das navegações dependerá se os grids são paralelos ou aninhados. Vamos começar estudando o caso dos grids paralelos.

Cada grid determinará sua navegação de maneira independente do outro. Assim, pode acontecer que para um grid se encontre tabela base e para outro não.

Neste exemplo, ambos os grids terão tabela base, pois, claramente, vemos atributos em cada um e com isso basta para saber que haverá uma navegação implícita em cada grid.

A pergunta é: como se determina a tabela base de cada um?

Antes de responder, observemos que este exemplo só difere do anterior em que adicionamos o grid da direita e uma variável para filtrar os dados daquele grid e outra para mostrar um total.

Como sabemos, além do evento Refresh genérico de todo o panel, quando houver mais de um grid, o evento Load genérico desaparece, e agora temos um evento Refresh e um evento Load específicos de cada grid. Cada evento Load será disparado apenas uma vez ou N dependendo de GeneXus encontrar ou não tabela base para aquele grid.

Não é difícil intuir que a tabela base do primeiro grid será Attraction e a do segundo CountryCity, e que em ambos os casos será filtrado por CountryId, atributo recebido por parâmetro, que, como sempre, não

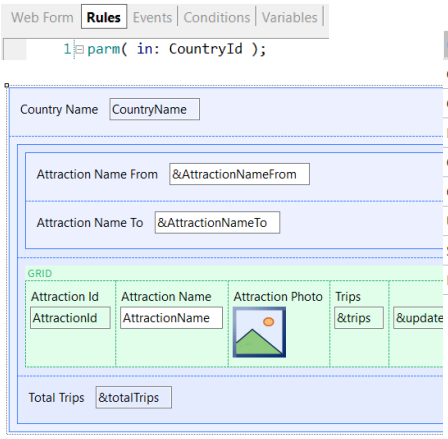
participará em absoluto na determinação das tabelas base, mas sim depois que estas estejam definidas.

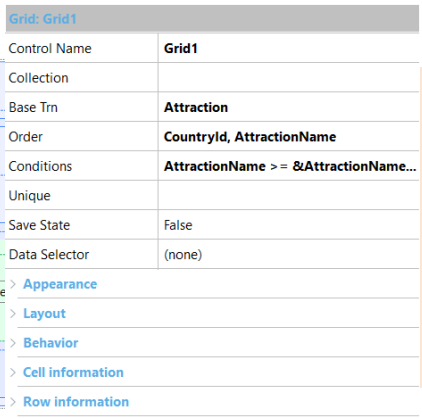
Porém, poderíamos pensar que como ambas as tabelas estão relacionadas na base de dados (vejamos que, de fato, CountryCity faz parte da tabela estendida de Attraction, portanto, para cada atração que for carregada neste grid, haverá um registro desta tabela associado, ou, olhando ao contrário, para cada cidade que for carregada neste outro, haverá N atrações relacionadas)... dissemos, poderíamos pensar que essa relação terá um impacto no que se carregue nos grids, mas não. GeneXus **não estabelecerá nenhuma relação implícita entre eles**.

No primeiro grid serão carregadas todas as atrações do país recebido por parâmetro e no segundo todas as cidades desse país.

Esclarecida esta possível confusão, vejamos agora como GeneXus determina a tabela base de cada grid.

With several Grids: parallels





- 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)

```
Event Grid1.Refresh
     &totalTrips = 0
Endevent

Event Grid1.Load
     &trips = Count(TripDate)
     &totalTrips = &totalTrips + &trips
Endevent
```

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

Event Grid2.Load
     &attractions = Count(AttractionName)
     &totalAttractions = &totalAttractions + &attractions
endevent
```

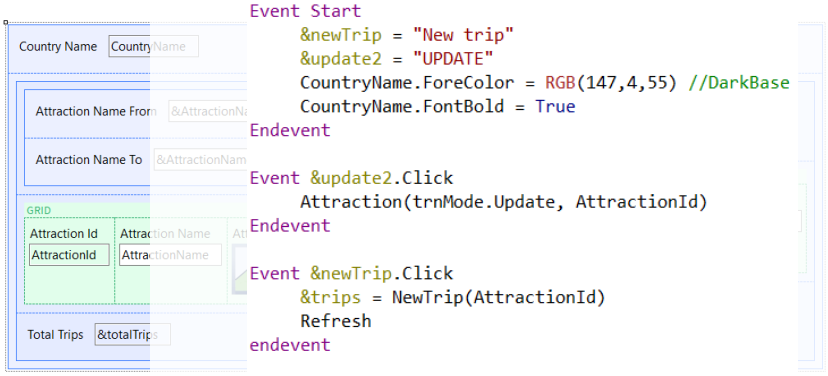
Pega o primeiro em ordem.

Considera os atributos do grid (visíveis ou ocultos), as mesmas propriedades do grid que vimos para o caso de um único grid (a Base Transaction, obviamente, e as propriedades Order, Conditions, Unique, Data Selector). E, ao contrário dos casos anteriores, aqui **não vai considerar** os atributos "soltos" de **todos** os eventos, mas apenas **do evento Load do grid**. Isto significa que se no evento Refresh, por exemplo, houvesse um atributo solto, este não participará.

With several Grids: parallels

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
Endevent

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

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

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

- 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)

```
Event Grid1.Refresh
     &totalTrips = 0
Endevent

Event Grid1.Load
     &trips = Count(TripDate)
     &totalTrips = &totalTrips + &trips
Endevent
```

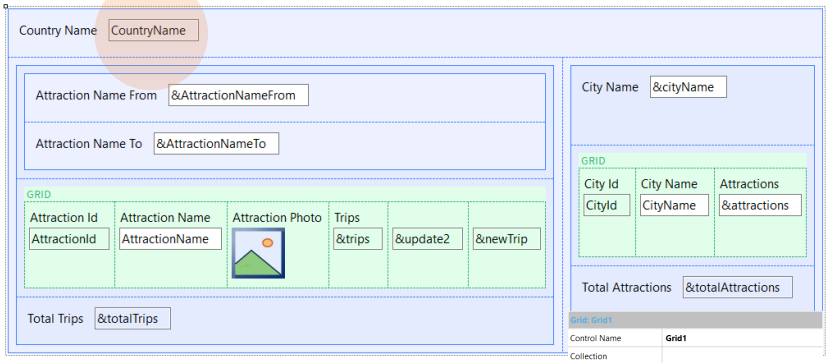
```
Event Grid2.Load
     &attractions = Count(AttractionName)
     &totalAttractions = &totalAttractions + &attractions
endevent
```

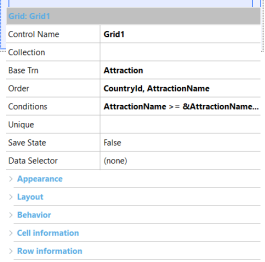
O mesmo em qualquer outro evento. Como estes outros.

With several Grids: parallels

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

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






```
Event Grid1.Refresh
     &totalTrips = 0
Endevent

Event Grid1.Load
     &trips = Count(TripDate)
     &totalTrips = &totalTrips + &trips
Endevent
```

- 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)

+ fixed-part attributes

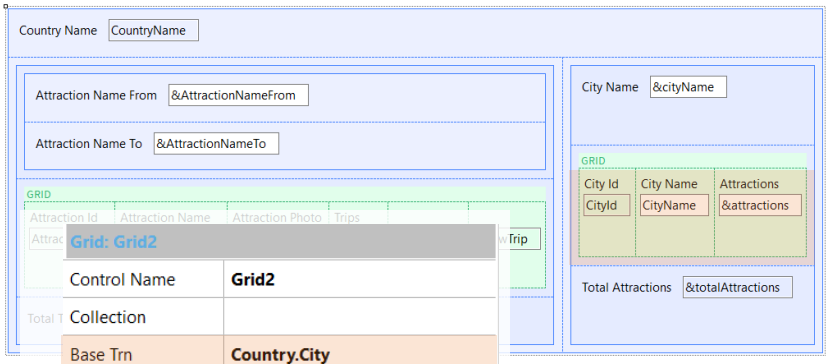
Mas também, para o caso do primeiro grid e **somente para ele**, se houver atributos na **parte fixa**, como é o caso, estes atributos **também serão considerados** para a determinação de sua tabela base. **E apenas para ele**. Para nenhum dos outros grids os atributos da parte fixa participarão.

Assim, no nosso caso, para determinar a tabela base do Grid1, são considerados todos estes atributos do grid, e aqueles que houverem "soltos" neste evento Load. Não há nenhum. E também, claro, as propriedades mencionadas do grid.
Fica claro **por que haverá tabela base** e é Attraction. Se algum destes atributos não estivesse na tabela estendida de Attraction, seríamos avisados na lista de navegação.

With several Grids: parallels

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

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



Grid: Grid2	
Control Name	**Grid2**
Collection	
Base Trn	**Country.City**
Order	
Conditions	**CityName like &cityName whe...**
Unique	
Save State	False
Data Selector	(none)

- 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)

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

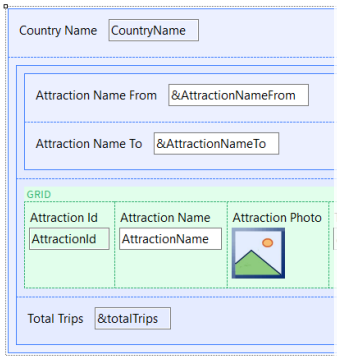
```
Event Grid2.Load
    &attractions = Count(AttractionName)
    &totalAttractions = &totalAttractions + &attractions
endevent
```

Por outro lado, para determinar a tabela base do Grid2, então, serão considerados todos estes atributos do grid e aqueles que houverem "soltos" no Load. Neste caso, também não há nenhum. Além disso, serão consideradas as propriedades do Grid. Vemos claramente por que a tabela base é CountryCity. Neste caso, o atributo CountryName da parte fixa não participa.

With several Grids: parallels

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
Endevent

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

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

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

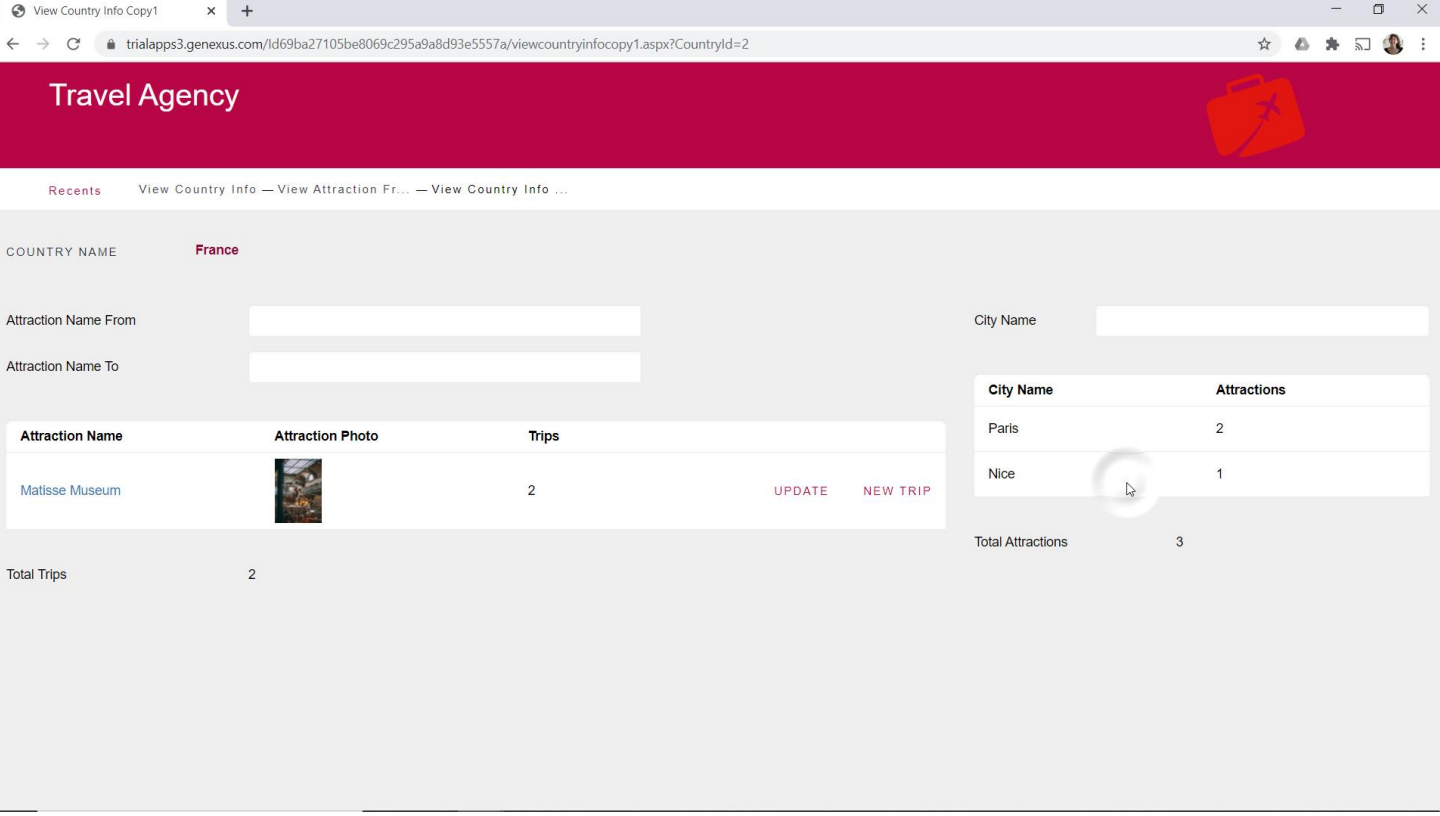
- Attributes in the **grid** (visible or hidden)
- Grid **Base Trn** property
- Grid **Order** property
- Grid **Conditions** property
- Grid **Unique** property
- Grid **Data Selector** property
- Attributes in the grid's **Load event** (without context, ie: For each command and inline aggregate formula)

```
Event Grid1.Refresh
     &totalTrips = 0
Endevent

Event Grid1.Load
     &trips = Count(TripDate)
     &totalTrips = &totalTrips + &trips
Endevent
```

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

Event Grid2.Load
     &attractions = Count(AttractionName)
     &totalAttractions = &totalAttractions + &attractions
endevent
```

E o que acontece com os atributos que estão aparecendo nestes outros eventos? Apenas devem pertencer à tabela estendida de alguma das tabelas base dos grids. Caso contrário, nos informará na lista de navegação.

Neste caso, temos CountryName e AttractionId.



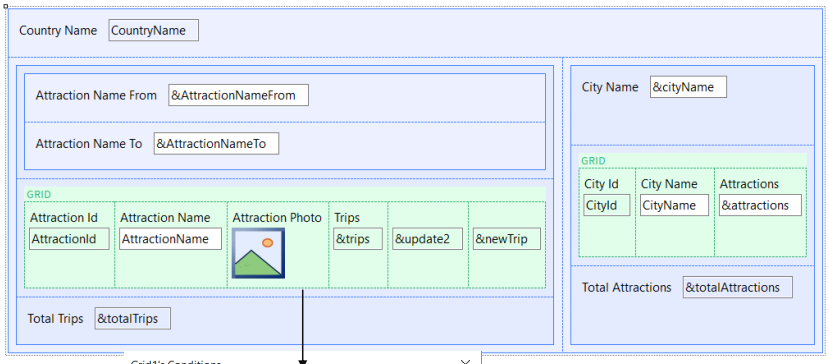
Dissemos que para grids paralelos as navegações **não se relacionam automaticamente**.

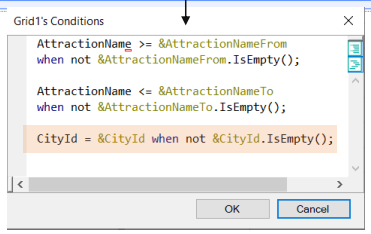
Se quiséssemos, por exemplo, que quando o usuário clique em uma linha do grid que mostra as cidades, no grid que mostra as atrações turísticas, sejam mostradas apenas aquelas **dessa cidade**, como estamos vendo, como faríamos?

With several Grids: parallels

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

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






```
Event Grid2.OnLineActivate
      &CityId = CityId
      Grid1.Refresh()
Endevent
```

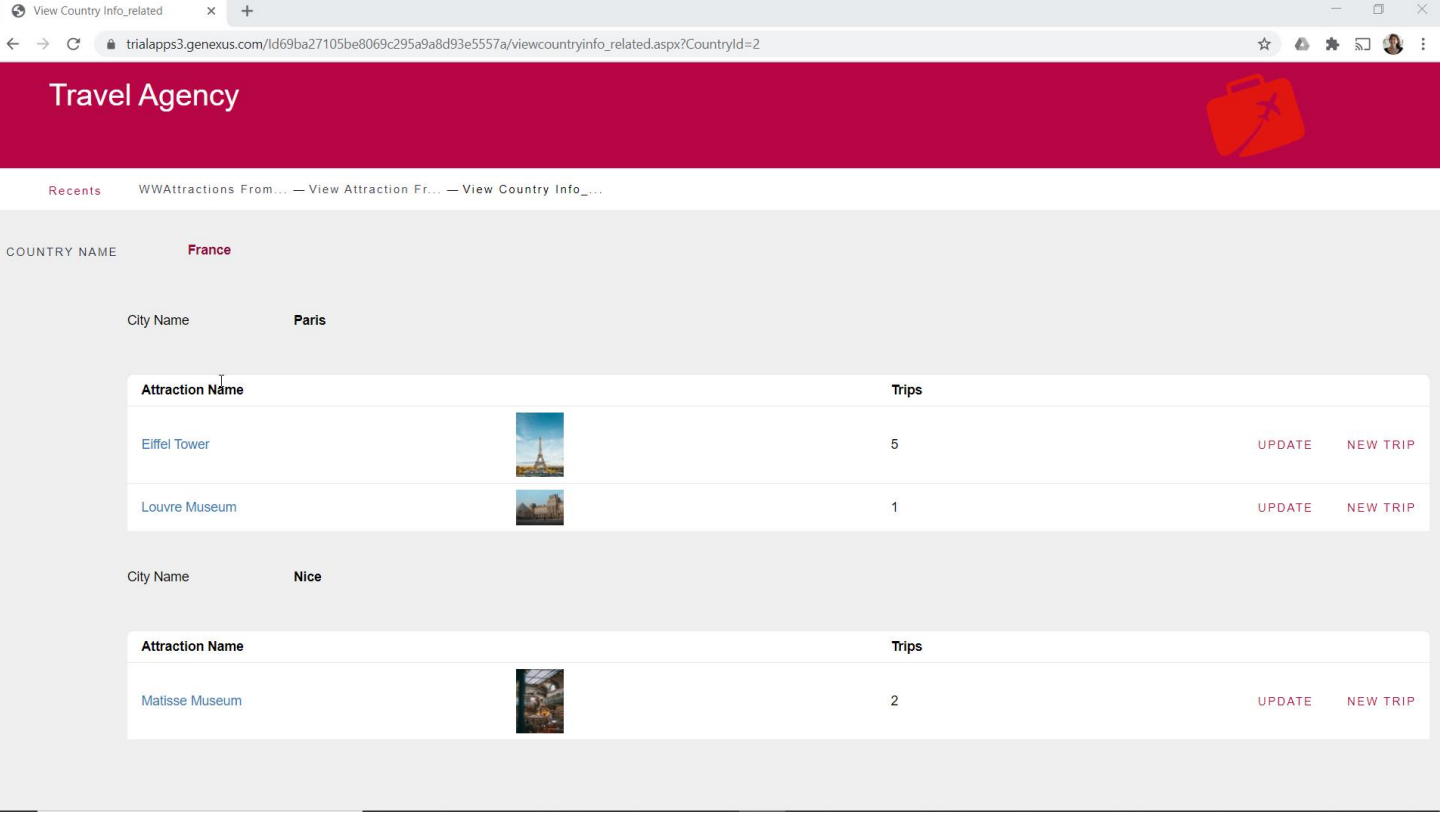
Grid: Grid2	
Control Name	**Grid2**
Collection	
Base Trn	**Country.City**
Order	
Conditions	**CityName like &cityName w...**
Unique	
Save State	False
Data Selector	(none)
Appearance	
Layout	
Behavior	
Sortable	True
Allow Drop	False
Allow Drag	False
Notify Context Ch	False
Allow Collapsing	False
Allow Selection	**True**
Allow Hovering	True

Existem várias maneiras de fazer isso. Mostraremos a que implementamos aqui. Fizemos um Save as de nosso panel e para o grid das cidades definimos a propriedade AllowSelection, para permitir a seleção de uma linha clicando em qualquer parte dela. Podemos fazer com que apareça com outra cor ou não.

Além disso, programamos o evento **OnlineActivate** do grid, para que quando o usuário escolhe uma linha, seja disparado e possamos atribuir a uma variável o identificador da cidade da linha escolhida.

A seguir, enviamos para atualizar o grid das atrações, pois colocamos mais uma condição nele: que sejam carregadas apenas as atrações cuja cidade coincida com a da variável &CityId, desde que esta não esteja vazia.

Com isto obtemos o comportamento que mostramos em execução.

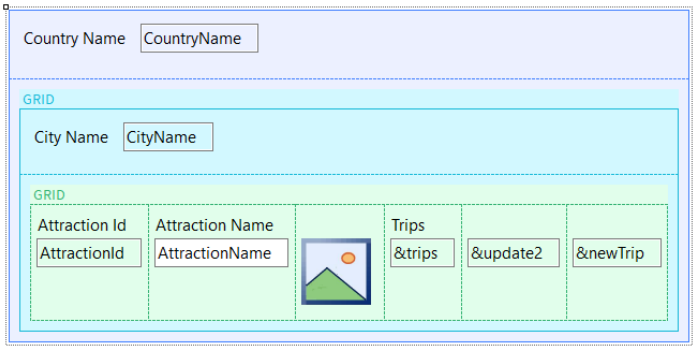


Mas a outra alternativa é mostrar diretamente para cada cidade do país recebido por parâmetro, suas atrações. Ou seja, utilizar grids aninhados, como fizemos aqui.

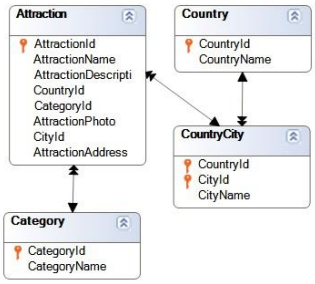
With several Grids: nested

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

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




```
For each Country.City

        print CityPB
        &trips = 0

        For each Attraction
                &trips = Count()
                print AttractionPB
        endfor

endfor
```

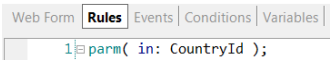


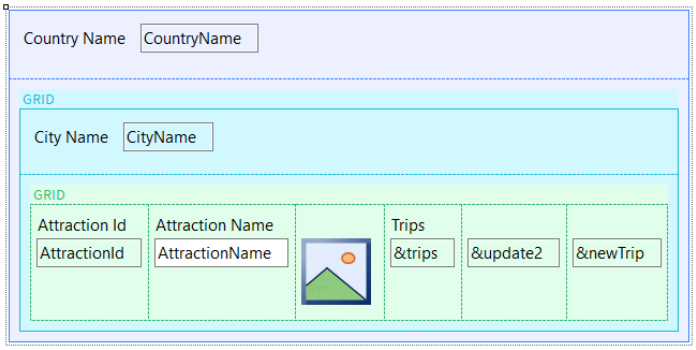
Sabemos que ter grids aninhados é equivalente a ter For eachs aninhados, portanto, a forma de determinar suas tabelas base e as navegações resultantes será análoga.

Se programássemos este objeto como lista, teríamos o For each externo navegando na tabela CountryCity, que terá um filtro implícito por CountryId, portanto percorrerá todas as cidades do país, e para cada uma imprimirá seu nome e antes de passar para a próxima, executará o For each interno, que percorrerá a tabela Attraction, filtrando implicitamente por país e cidade, e imprimindo cada atração desse país e cidade.

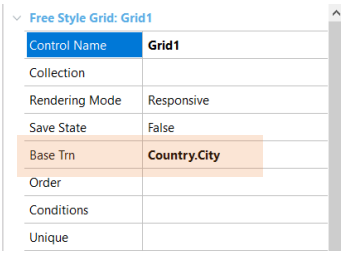
Esta mesma navegação é a que conseguiremos em nosso web panel. Mas como sempre, temos duas possibilidades: implementar cada grid com ou sem tabela base.

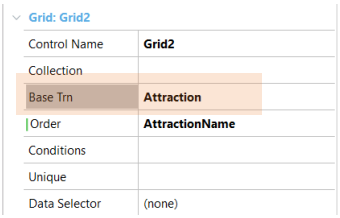
With several Grids: nested





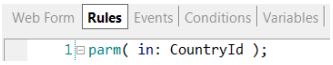

```
For each Country.City

        print CityPB
        &trips = 0

        For each Attraction
              &trips = Count()
              print AttractionPB
        endfor

endfor
```

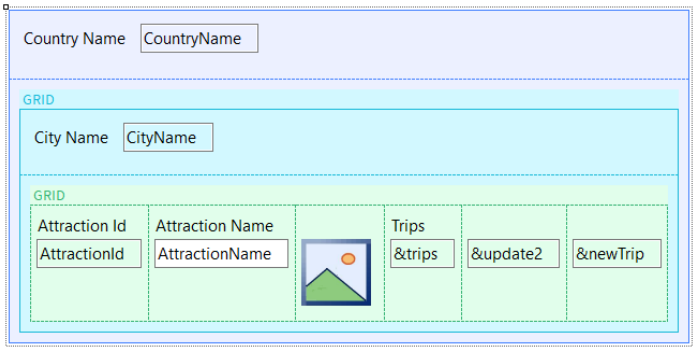




Se implementamos ambos os grids com tabela base, que é a forma na qual trabalhamos menos, estabeleceremos transação base Country.City para o primeiro grid e Attraction para o segundo.

With several Grids: nested





Grid1.Refresh()

Grid1.Load() → **Paris**

Grid2.Refresh()

Grid2.Load() — Eiffel Tower

Grid2.Load() — Louvre Museum

Grid1.Load() → **Nice**

Grid2.Refresh()

Grid2.Load() — Matisse Museum

Em qualquer caso, primeiro ocorrerá o evento Refresh do Grid1, o externo, e depois, dependendo se o grid possui ou não tabela base, uma ou N vezes o evento Load desse grid.

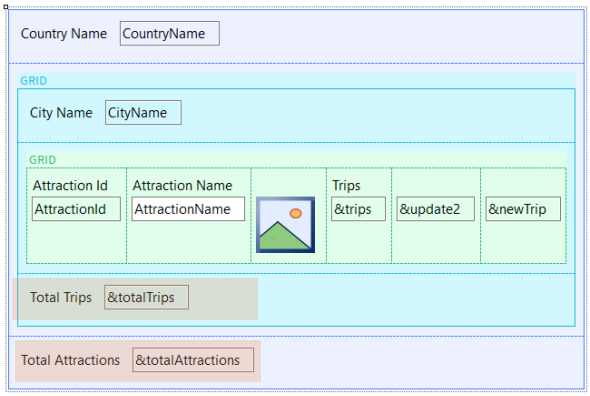
No nosso caso, como França tem duas cidades inseridas, Paris e Nice, sabemos que o primeiro Load do Grid externo carregará Paris, e imediatamente, antes de executar novamente o Load para carregar Nice desta vez, ocorrerá o evento Refresh do grid aninhado. E imediatamente seu evento Load, uma vez ou N, novamente, dependendo se tem ou não tabela base. No nosso caso tem, então será disparado um Load para carregar a Torre Eiffel e outro para carregar o museu do Louvre.

Uma vez que terminou de carregar o grid aninhado, agora sim, passará a carregar a próxima cidade, Nice. E fará o mesmo, disparará uma vez o evento Refresh do grid aninhado, para passar a disparar N vezes o evento Load para carregar as novas atrações, as de Nice, que em nosso exemplo é apenas uma.

With several Grids: nested

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

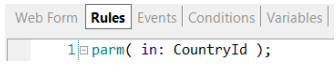
```
1 parm( in: CountryId );
```

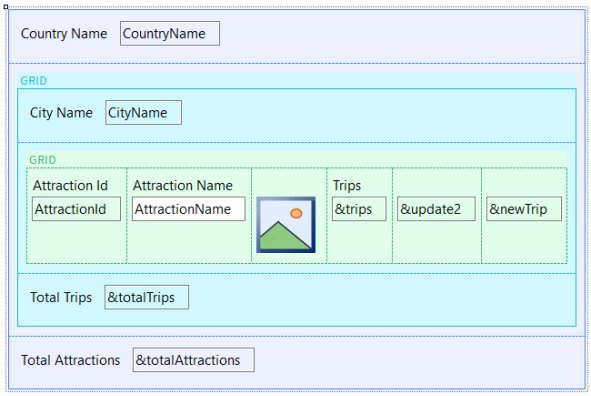



```
Event Start
     &newTrip = "New trip"
     &update2 = "UPDATE"
     CountryName.ForeColor = RGB(147,4,55) //DarkBase
     CountryName.FontBold = True
     CityName.FontBold = True
Endevent

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
```

Aqui adicionamos à tela as variáveis para totalizar que tínhamos antes, assim o exemplo fica idêntico e com a necessidade de programar todos os eventos do sistema.

With several Grids: nested






```
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
```

Se traduzirmos isto para um pseudocódigo GeneXus, ficaria mais ou menos assim.

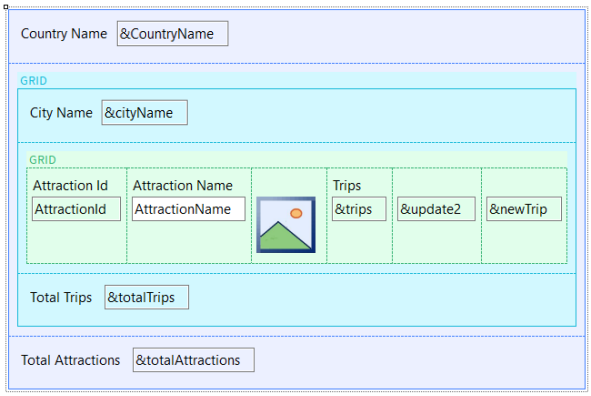
Primeiro, é executado o Refresh do grid externo. Ali configuramos com zero a variável que contará o número de atrações que serão carregadas no total.
Depois, por se tratar de um grid com tabela base CountryCity, GeneXus colocará o For each implícito que navegará nessa tabela, filtrando pelo valor de CountryId recebido no parâmetro. Para cada registro encontrado, será executado o evento Load desse grid, que contará as atrações dessa cidade e as somará à variável que totalizará. A seguir carrega-se a cidade no grid, a partir do comando Load que GeneXus coloca.

Imediatamente é executado o Refresh do grid aninhado, que deixa em zero o valor da soma de trips nas quais se encontram as atrações que serão carregadas a seguir. E por ter tabela base, GeneXus escreve outro For each implícito para navegar nessa tabela base, Attraction, ao qual adiciona todas as cláusulas correspondentes de acordo com o que o desenvolvedor especificou nas propriedades do grid. Em nosso caso, apenas havíamos colocado transação base e cláusula order. Também acrescenta as condições implícitas que têm a ver, justamente, com o fato deste grid estar aninhado a outro e existe relação entre as tabelas. Por isso, somente percorrerá os registros da tabela de atrações que coincidam com o país e cidade do registro que foi carregado no for each externo. E para cada um, executará o Load deste grid aninhado. E então ele carregará a linha no grid.

With several Grids: nested

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

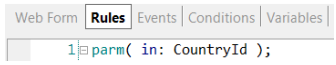
```
Event Grid2.Load
     &trips = Count(TripDate)
     &totalTrips = &totalTrips + &trips
Endevent
```

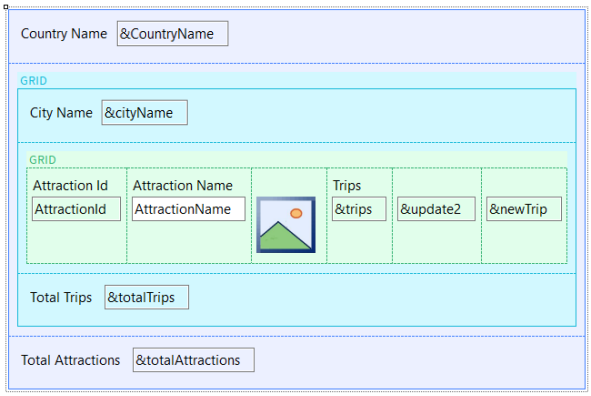
Load

endfor

Obviamente, se o primeiro grid for sem tabela base, então o For each implícito desaparece, assim como o comando Load.

With several Grids: nested






```
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

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

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

          Load

     endfor
```

E teremos que escrevê-los explicitamente no evento Load do grid.

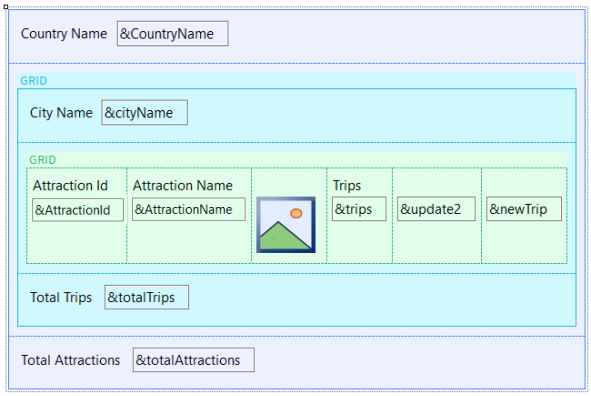
Neste caso, ao encontrar o comando Load dentro do evento Load do grid, por estar os grids aninhados, GeneXus disparará imediatamente os eventos Refresh e Load do grid aninhado. E ali dependerá, novamente, se o grid aninhado tem ou não tabela base, para que GeneXus coloque ou não um for each implícito e seu Load.

Só que neste caso teremos que explicitar o where de cidades, que antes estava implícito, como logo entenderemos.

With several Grids: nested

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

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



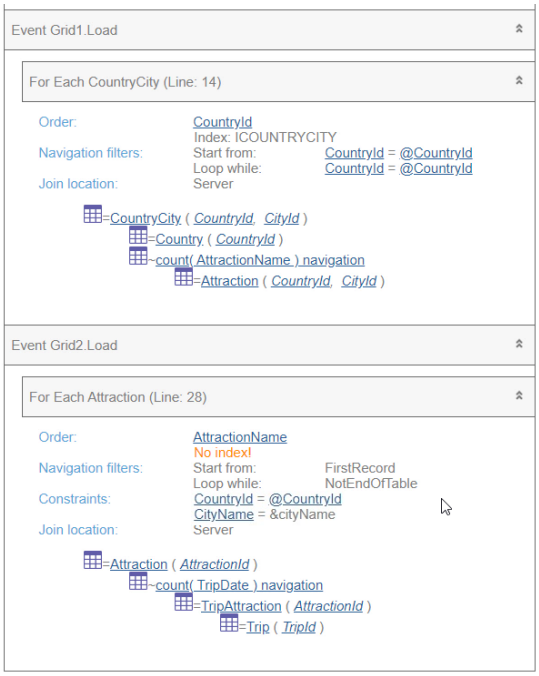

```
Event Grid1.Refresh
     &totalAttractions = 0
endevent

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
```

Se quiséssemos que o grid aninhado não tenha tabela base, então claramente substituiremos os atributos por variáveis no grid e o for each teremos que programar explicitamente no evento Load, bem como o comando Load.

Justamente por não ter tabelas base e ficar totalmente nas mãos do desenvolvedor a lógica para a carga dos grids, GeneXus não pode estabelecer o join automático entre os For eachs, razão pela qual devemos explicitar os filtros. Colocamos apenas o filtro por CityName, pois o filtro por CountryId já será feito devido ao parâmetro.

With several Grids: nested

```
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
```

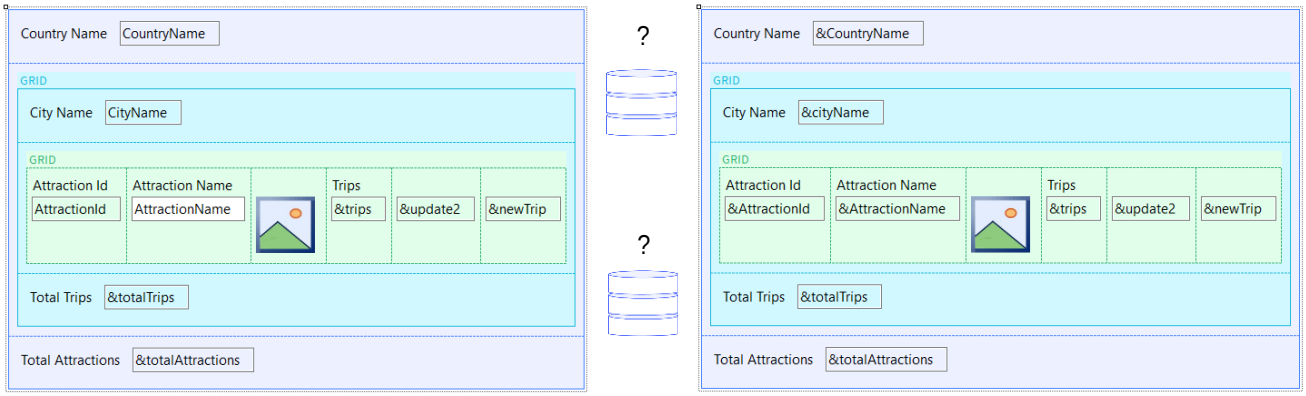


Aqui o temos implementado no GeneXus.

Se observamos sua lista de navegação, vemos que claramente não escolheu tabela base para nenhum dos dois grids. E se executamos... não vemos nenhuma diferença com o web panel que tinha tabelas base para ambos os grids.

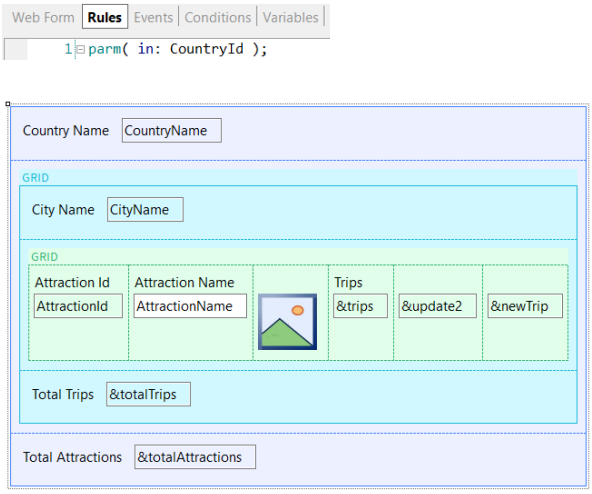
With several Grids: nested

Base Tables



Mas para chegar a estas navegações, foi necessário primeiro determinar as tabelas base, de forma análoga ao que acontece com os For eachs.

With several Grids: nested



1st GRID

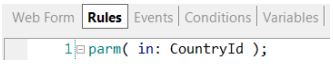
- Attributes in the **grid** (visible or hidden)
- Grid **Base Trn** property
- Grid **Order** property
- Grid **Conditions** property
- Grid **Unique** property
- Grid **Data Selector** property
- Attributes in the grid's **Load event** (without context, ie: For each command and inline aggregate formula)

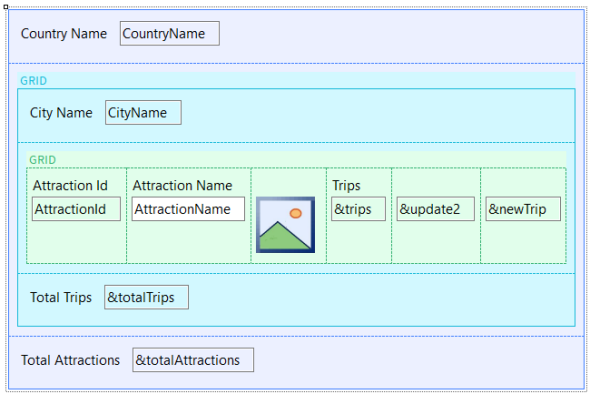
+

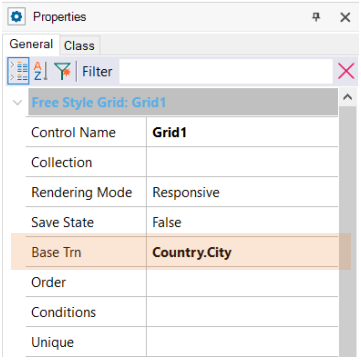
- Fixed-part attributes

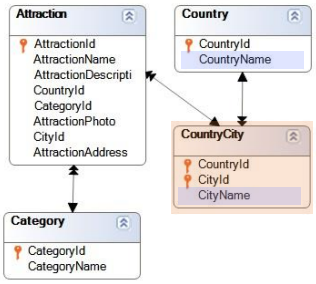
Para determinar a tabela base do primeiro grid da tela, vale exatamente o mesmo que vimos para grids paralelos.
Ou seja, GeneXus leva em consideração os atributos do próprio grid mais os da parte fixa da tela. Além, é claro, daqueles que aparecem nas propriedades do grid (Base transaction, Order, etc) e aqueles do evento Load **do grid**. Não aqueles do Refresh do próprio grid, nem de qualquer outro evento.

With several Grids: nested



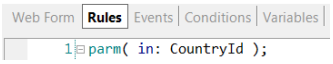


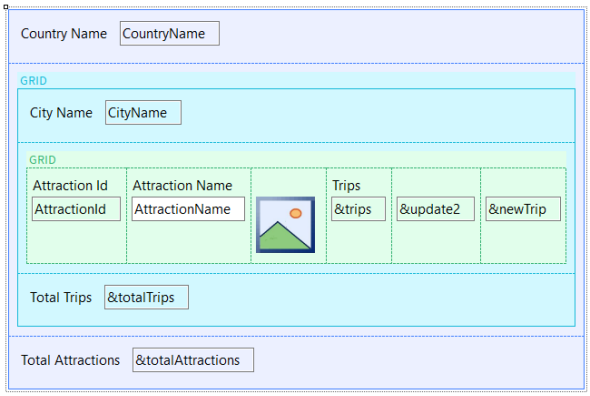


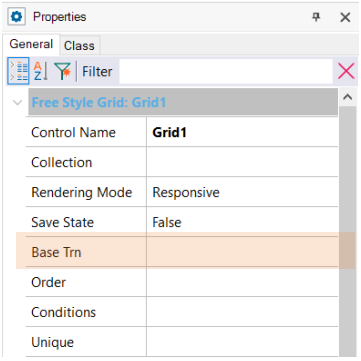


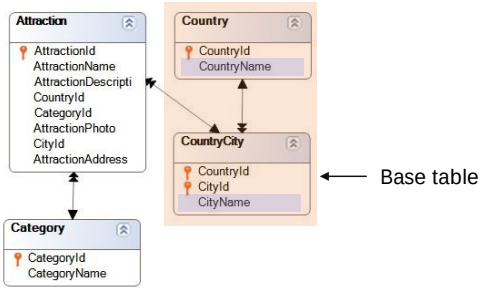
Se existe transação base especificada, a sua será a tabela base e todos esses atributos que mencionamos deverão pertencer à sua tabela estendida.

With several Grids: nested



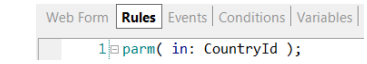


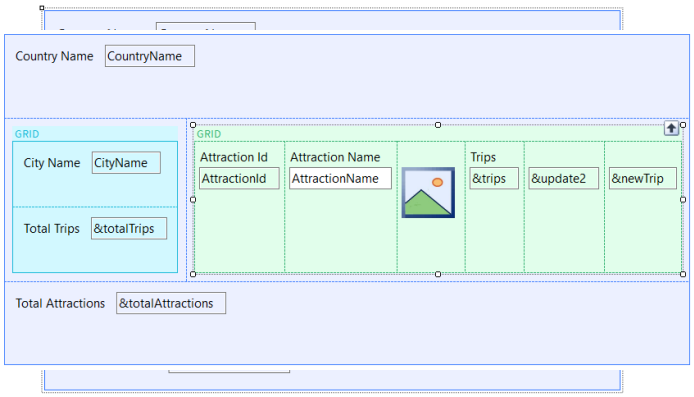




Se não existe transação base especificada, então GeneXus encontra a **mínima tabela estendida** que contenha todos os atributos mencionados e escolhe a sua tabela base como a tabela base do grid.

With several Grids: nested





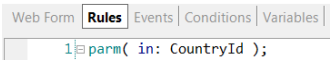
Nested GRID

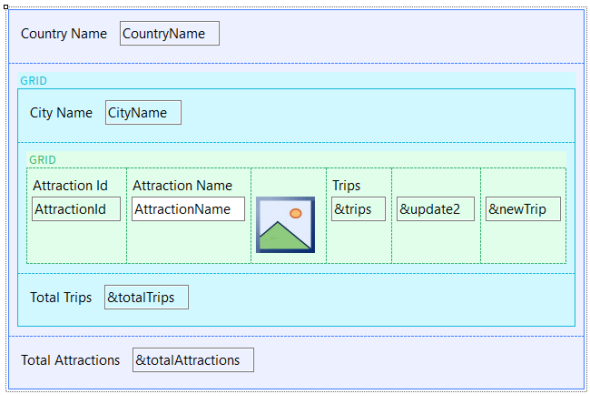
- Attributes in the **grid** (visible or hidden)
- Grid **Base Trn** property
- Grid **Order** property
- Grid **Conditions** property
- Grid **Unique** property
- Grid **Data Selector** property
- Attributes in the grid's **Load event** (without context, ie: For each command and inline aggregate formula)

A tabela base do grid aninhado é determinada como se o grid fosse paralelo e não aninhado, mas com uma ressalva, que é a mesma que para determinar a tabela base de um for each aninhado.

Se o grid aninhado NÃO TEM ESPECIFICADA TRANSAÇÃO BASE, então GeneXus deve determiná-la por si mesmo, e aqui é onde o fato de que este grid esteja aninhado a outro pode determinar uma tabela diferente daquela que seria determinada se o grid fosse paralelo.

With several Grids: nested



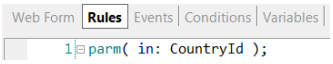


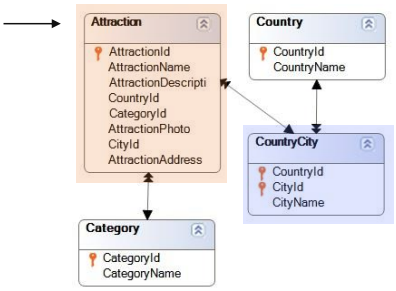
Nested GRID

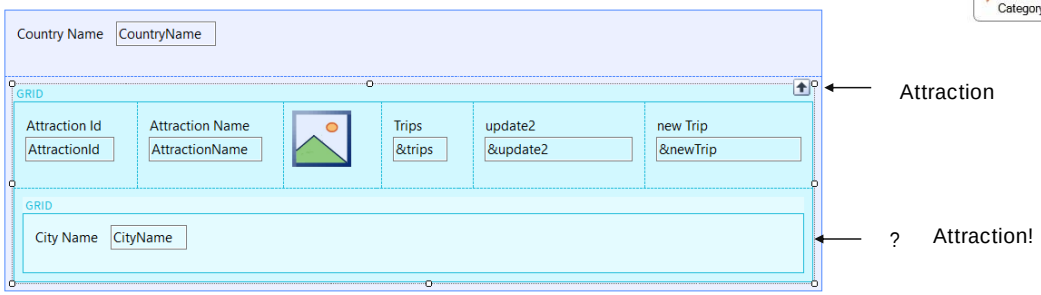
- Attributes in the **grid** (visible or hidden)
- Grid **Base Trn** property
- Grid **Order** property
- Grid **Conditions** property
- Grid **Unique** property
- Grid **Data Selector** property
- Attributes in the grid's **Load event** (without context, ie: For each command and inline aggregate formula)

São casos pouco frequentes, mas é bom estar avisado.

With several Grids: nested







Por exemplo, se os grids estivessem invertidos, e o externo navegasse a tabela Attraction e no interno não tivesse especificada transação base e estivesse apenas o atributo CityName envolvido, se o grid fosse paralelo, claramente GeneXus determinaria como sua tabela base a de cidades. Porém, neste caso, por estar aninhado a um grid que possui uma tabela estendida que inclui os atributos do segundo grid, então escolherá para este segundo grid a mesma tabela base que a do primeiro, implementando assim um corte de controle.

Base tables: ready!

And its navigations!

Com isto, concluímos o estudo de como são determinadas as tabelas base e as navegações para todos os casos de web panels.

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