# Pantallas web

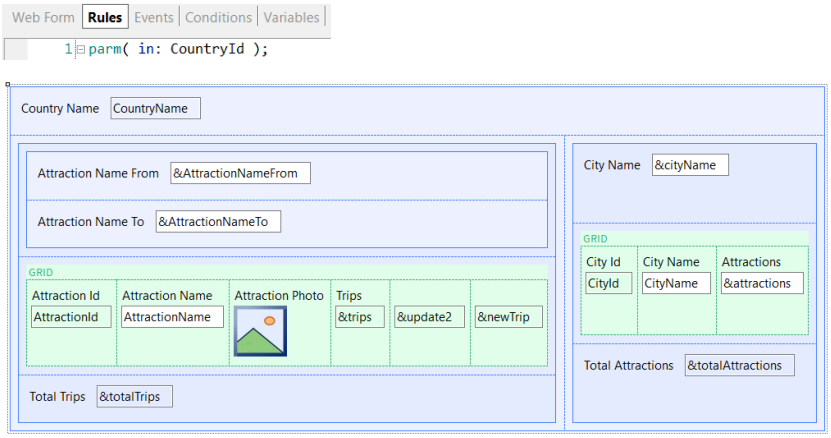
## Tablas base y navegaciones en Web panel con múltiples grid

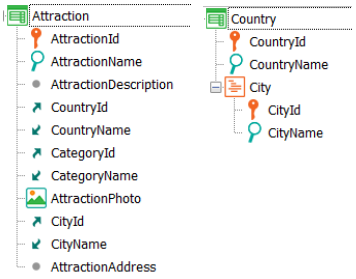
GeneXus™

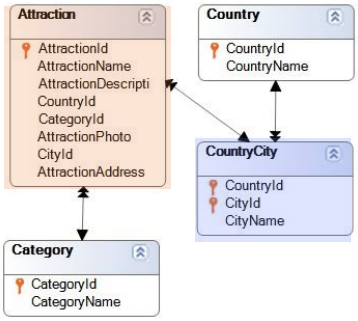
## Web Panel with SEVERAL Grids

Ahora, ¿qué pasa cuando un web panel tiene más de un grid?
Evidentemente ya no podremos hablar de tabla base del web panel, sino 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
```

Y la determinación de las navegaciones dependerá de si los grids son paralelos o anidados. Empecemos por estudiar el caso de los grids paralelos.

Cada grid determinará su navegación de manera independiente del otro. Así, puede suceder que para un grid se encuentre tabla base y para otro no.

En este ejemplo ambos grids tendrán tabla base, porque, claramente, vemos atributos en cada uno y con eso ya nos alcanza para saber que habrá una navegación implícita en cada grid.

La pregunta es: ¿cómo se determina la tabla base de cada uno?

Antes de contestarla, observemos que este ejemplo solo se diferencia del anterior en que hemos agregado el grid de la derecha y una variable para filtrar los datos de ese grid y otra para mostrar un total.

Como sabemos, además del evento Refresh genérico de todo el panel, cuando hay más de un grid el evento Load genérico desaparece, y ahora tenemos un evento Refresh y un evento Load específicos de cada grid. Cada evento Load se disparará una única vez o N dependiendo de si GeneXus encuentra o no tabla base para ese grid.

No es difícil intuir que la tabla base del primer grid será Attraction y la del segundo CountryCity, y que en ambos casos se va a filtrar por CountryId,

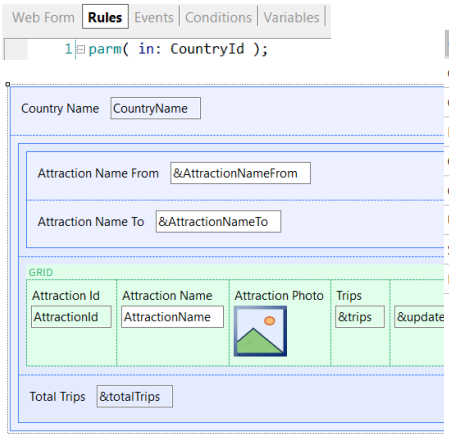
atributo recibido por parámetro, que, como siempre, no va a participar en absoluto en la determinación de las tablas base, pero sí luego de que éstas estén definidas.

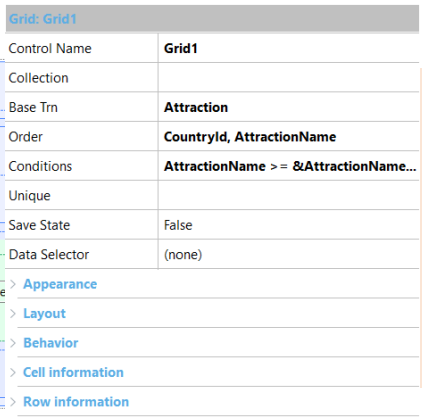
Sin embargo, podríamos pensar que como ambas tablas están relacionadas en la base de datos (veamos que de hecho CountryCity forma parte de la tabla extendida de Attraction, por lo que por cada atracción que se cargue en este grid, habrá un registro de esta tabla asociado, o, mirándolo al revés, por cada ciudad que se cargue en este otro, habrá N atracciones relacionadas)... decíamos, podríamos pensar que esa relación tendrá un impacto en lo que se cargue en los grids, pero no. GeneXus no establecerá ninguna relación implícita entre ellos.

En el primer grid se cargarán todas las atracciones del país recibido por parámetro y en el segundo todas las ciudades de ese país.

Habiendo despejado esta posible confusión, veamos ahora sí, cómo GeneXus determina la tabla 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
```

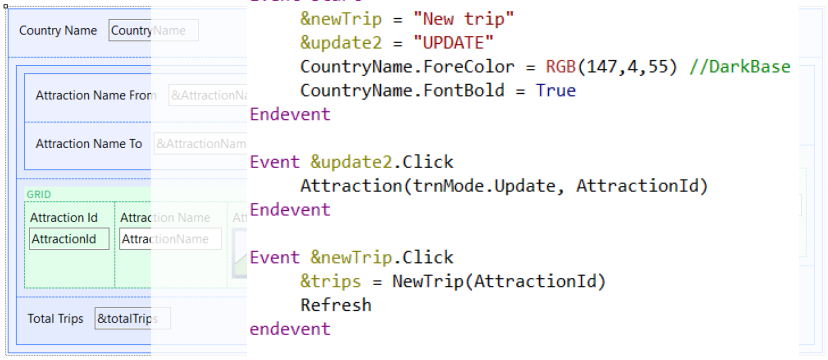
Toma el primero en orden.

Considera los atributos del grid (visibles u ocultos), las mismas propiedades del grid que vimos para el caso de un solo grid (la Base Transaction, obviamente, y las propiedades Order, Conditions, Unique, Data Selector). Y, a diferencia de los casos anteriores, aquí **no va a considerar** los atributos "sueltos" de **todos** los eventos, sino únicamente del **evento Load del grid**. Esto significa que si en el evento Refresh, por ejemplo, hubiese un atributo suelto, éste no 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
```

```
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
```

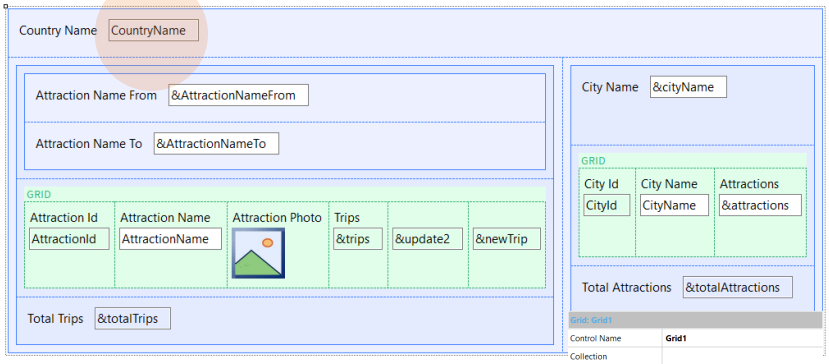
- Attributes in the **grid** (visible or hidden)
- Grid **Base Trn** property
- Grid **Order** property
- Grid **Conditions** property
- Grid **Unique** property
- Grid **Data Selector** property
- Attributes in the grid's **Load event** (without context, ie: For each command and inline aggregate formula)

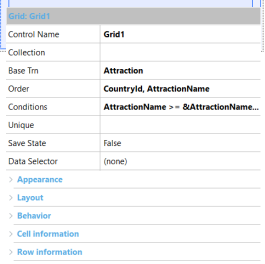
Lo mismo en cualquier otro evento. Como estos otros.

## 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

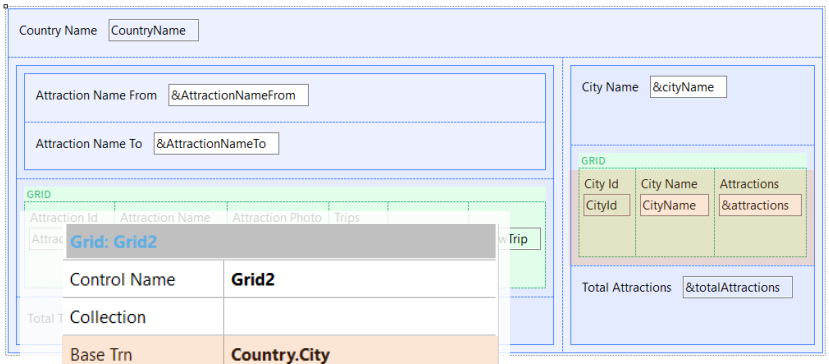
Pero además, para el caso del primer grid y **solo para él**, de haber atributos en la **parte fija**, como es el caso, estos atributos **también se considerarán** para la determinación de su tabla base. **Y solo para ella**.
Para ninguno de los otros grids los atributos de la parte fija van a participar.

Así, en nuestro caso para determinar la tabla base del Grid1 se consideran todos estos atributos del grid, y los que hubiera "sueltos" en este evento Load. No hay ninguno. Y además, claro, las propiedades mencionadas del grid.
Queda claro **por qué habrá tabla base** y es Attraction. Si alguno de estos atributos no estuvieran en la tabla extendida de Attraction, se nos lo advertiría en el listado de navegación.

# With several Grids: parallels

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

```
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) |

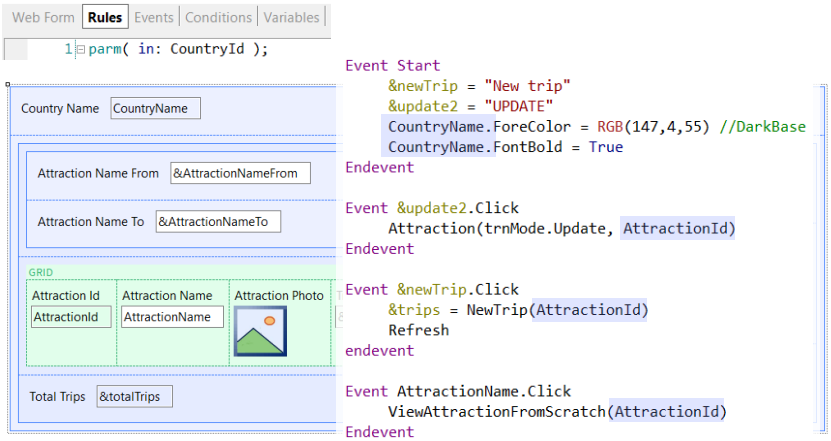
- Attributes in the **grid** (visible or hidden)
- Grid **Base Trn** property
- Grid **Order** property
- Grid **Conditions** property
- Grid **Unique** property
- Grid **Data Selector** property
- Attributes in the grid's **Load event** (without context, ie: For each command and inline aggregate formula)

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

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

Por otro lado, para determinar la tabla base del Grid2 se van a considerar, entonces, todos estos atributos del grid y los que hubiera "sueltos" en el Load. En este caso tampoco hay ninguno. Además, se van a considerar las propiedades del Grid. Vemos claramente por qué la tabla base es CountryCity. En este caso el atributo CountryName de la parte fija no participa.

## With several Grids: parallels



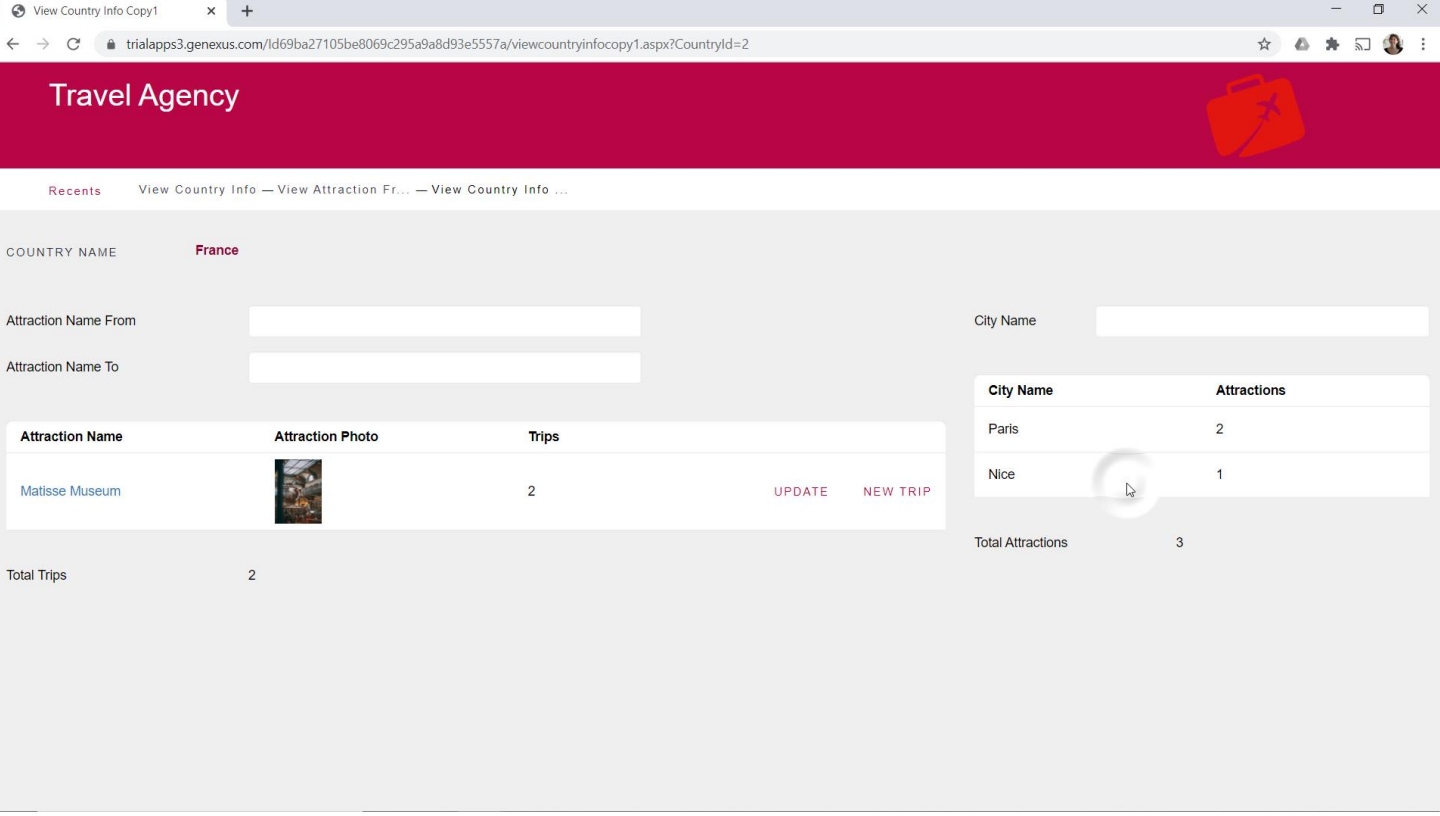

- Attributes in the **grid** (visible or hidden)
- Grid **Base Trn** property
- Grid **Order** property
- Grid **Conditions** property
- Grid **Unique** property
- Grid **Data Selector** property
- Attributes in the grid's **Load event** (without context, ie: For each command and inline aggregate formula)

```
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
```

¿Y qué sucede con los atributos que están apareciendo en estos otros eventos? Solo deben pertenecer a la tabla extendida de alguna de las tablas bases de los grids. De lo contrario, se nos lo informará en el listado de navegación.

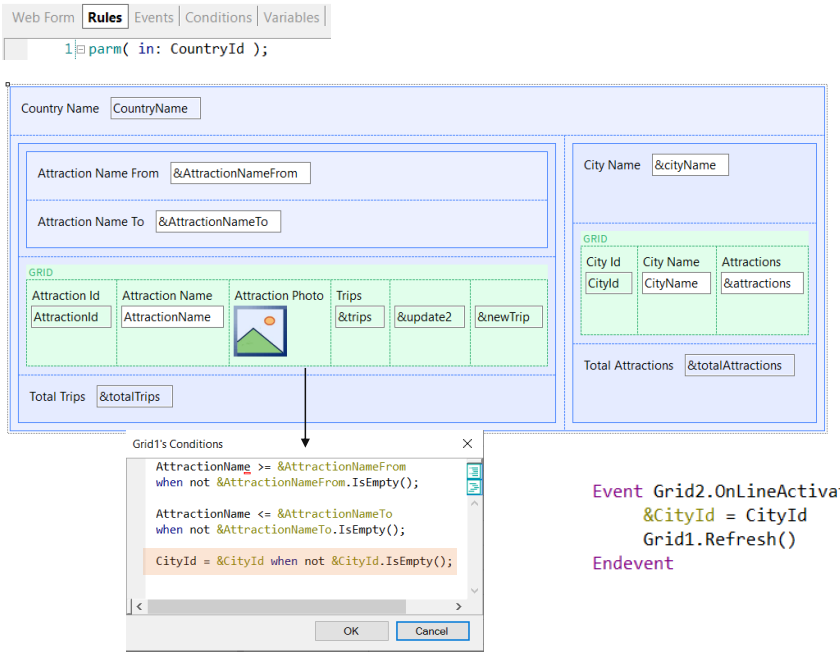
En este caso tenemos a CountryName y a AttractionId.




Dijimos que para grids paralelos las navegaciones **no se relacionan automáticamente**.

Si quisiéramos, por ejemplo, que cuando el usuario hace clic en una línea de la grilla que muestra las ciudades, en la grilla que muestra las atracciones turísticas se muestren solo las de **esa ciudad**, como estamos viendo, ¿cómo haríamos?

## With several Grids: parallels




```
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 |

Hay varias maneras de conseguirlo. Mostraremos la que implementamos aquí. Hicimos un Save as de nuestro panel, y al grid de las ciudades le prendimos la propiedad AllowSelection, para permitir la selección de una línea haciendo clic en cualquier parte de ella. Podemos hacer que aparezca con otro color o no.

Además, programamos el evento OnlineActivate del grid, para que cuando el usuario elige una línea, se dispare y podamos asignarle a una variable el identificador de la ciudad de la línea elegida.

A continuación, enviamos a refrescar el grid de las atracciones, ya que le colocamos una condición más: que se carguen solo las atracciones cuya ciudad coincida con la de la variable &CityId, siempre y cuando ésta no esté vacía.

Con esto conseguimos el comportamiento que mostramos en ejecución.

View Country Info_related

trialapps3.genexus.com/Id69ba27105be8069c295a9a8d93e5557a/viewcountryinfo_related.aspx?CountryId=2

# Travel Agency

Recents WWAttractions From... — View Attraction Fr... — View Country Info_...

COUNTRY NAME **France**

City Name **Paris**

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

City Name **Nice**

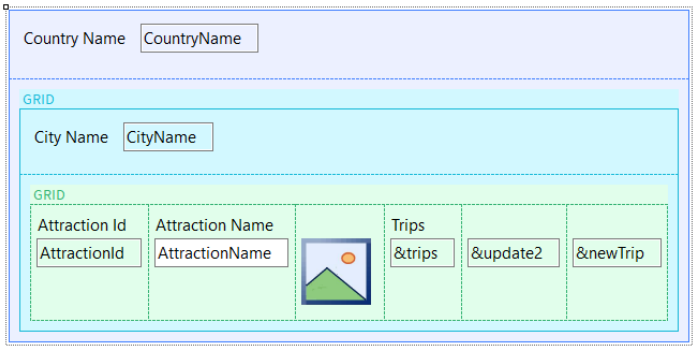
| Attraction Name | | Trips | | |
|---|---|---|---|---|
| Matisse Museum |  | 2 | UPDATE | NEW TRIP |

Pero la otra alternativa es directamente mostrar para cada ciudad del país recibido por parámetro, sus atracciones. Esto es, utilizar grids anidados, como hicimos aquí.

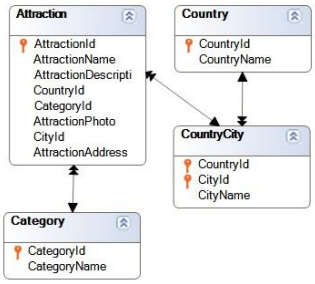
## With several Grids: nested

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 tener grids anidados es equivalente a tener For eachs anidados, por lo que la forma de determinar sus tablas bases y las navegaciones resultantes será análoga.

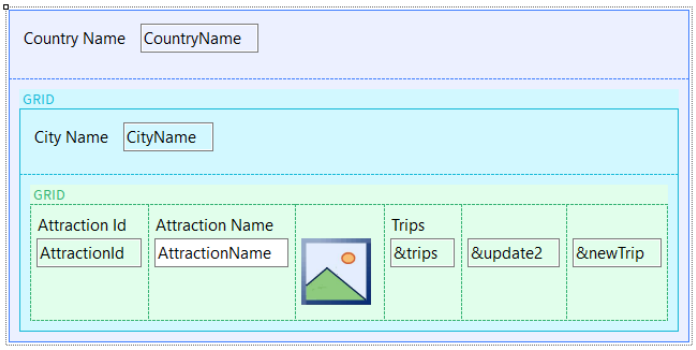
Si programáramos este objeto como listado, tendríamos el For each externo navegando la tabla CountryCity, que tendrá un filtro implícito por CountryId, por lo que recorrerá todas las ciudades del país, y para cada una imprimirá su nombre y antes de pasar a la siguiente, ejecutará el For each interno, que recorrerá la tabla Attraction, filtrando implícitamente por país y ciudad, e imprimiendo cada atracción de ese país y ciudad.

Esta misma navegación es la que conseguiremos en nuestro web panel. Pero como siempre, tenemos dos posibilidades: implementar cada grid con o sin tabla base.

# 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
```

**Free Style Grid: Grid1**

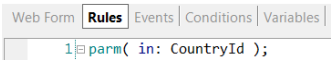
| Control Name | **Grid1** |
|---|---|
| Collection | |
| Rendering Mode | Responsive |
| Save State | False |
| Base Trn | **Country.City** |
| Order | |
| Conditions | |
| Unique | |

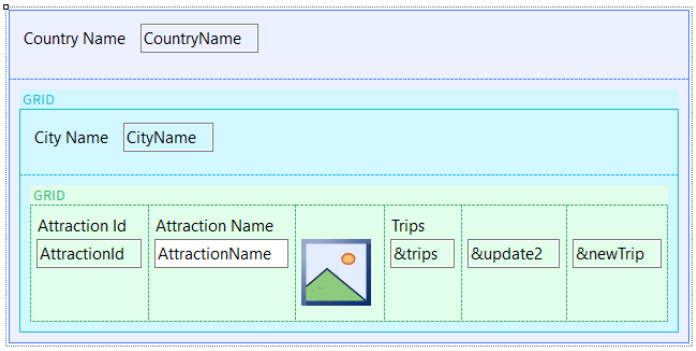
**Grid: Grid2**

| Control Name | **Grid2** |
|---|---|
| Collection | |
| Base Trn | **Attraction** |
| Order | **AttractionName** |
| Conditions | |
| Unique | |
| Data Selector | (none) |

Si implementamos ambos grids con tabla base, que es la forma en la que trabajamos menos, estableceremos transacción base Country.City para el primer grid y Attraction para el 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

En cualquier caso, primero va a producirse el evento Refresh del Grid1, el externo, y a continuación, dependiendo de si el grid tiene o no tabla base, una o N veces el evento Load de ese grid.

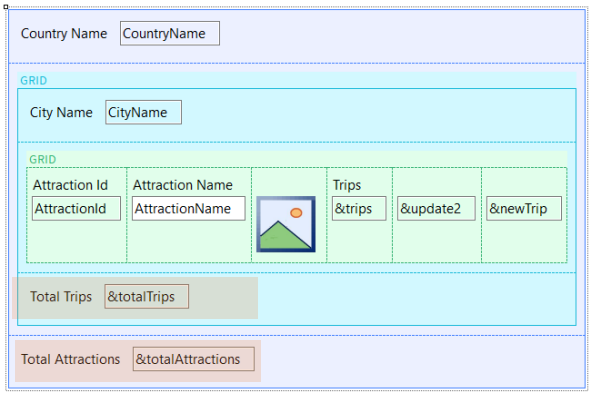
En nuestro caso, como Francia tiene dos ciudades ingresadas, París y Niza, sabemos que el primer Load del Grid externo va a cargar Paris, e inmediatamente, antes de ejecutar otra vez el Load para cargar esta vez Niza, va a producirse el evento Refresh del grid anidado. E inmediatamente su evento Load, una vez o N, otra vez, dependiendo de si tiene o no tabla base. En nuestro caso tiene, así que se disparará un Load para cargar la Torre Eiffel y otro para cargar el museo Louvre.

Una vez que terminó de cargar el grid anidado, ahora sí, pasará a cargar la siguiente ciudad, Niza. Y hará lo mismo, disparará una vez el evento Refresh del grid anidado, para pasar a disparar N veces el evento Load para cargar las nuevas atracciones, las de Niza, que en nuestro ejemplo es una sola.

# With several Grids: nested

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

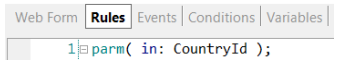
```
1 parm( in: CountryId );
```

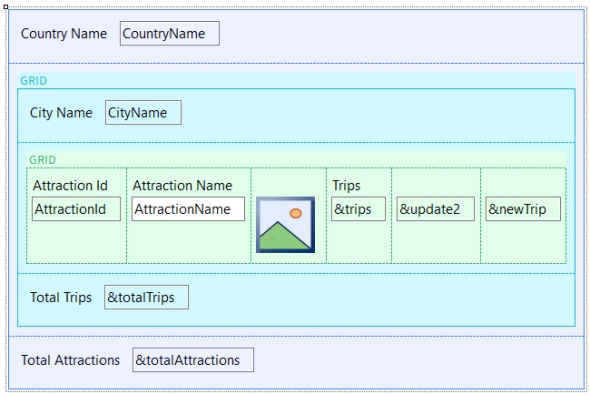



```
Event Start
     &newTrip = "New trip"
     &update2 = "UPDATE"
     CountryName.ForeColor = RGB(147,4,55) //DarkBase
     CountryName.FontBold = True
     CityName.FontBold = True
Endevent

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
```

Aquí agregamos a la pantalla las variables para totalizar que teníamos antes, así el ejemplo nos queda idéntico y con la necesidad de programar todos los eventos del 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
```

Si traducimos esto a un pseudo-código GeneXus, nos quedaría más o menos así.

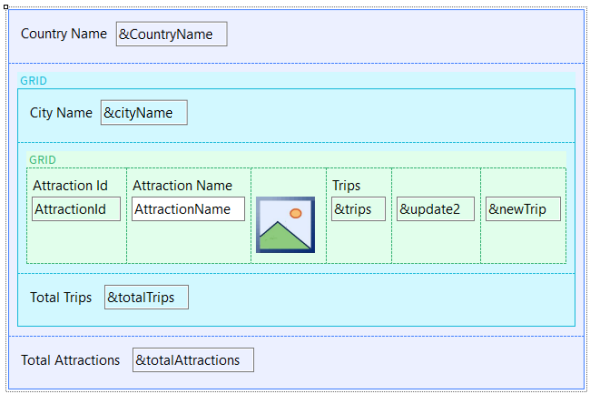
Primero se ejecuta el Refresh del grid externo. Allí ponemos en cero la variable que va a contar el número de atracciones que se cargarán en total.
Luego, por tratarse de un grid con tabla base CountryCity, GeneXus colocará el For each implícito que navegará esa tabla, filtrando por el valor de CountryId recibido en el parámetro. Para cada registro encontrado, se ejecutará el evento Load de ese grid, que contará las atracciones de esa ciudad y se las sumará a la variable que totalizará. A continuación se carga la ciudad en el grid, a partir del comando Load que GeneXus coloca.

Inmediatamente se ejecuta el Refresh del grid anidado, que deja en cero el valor de la suma de trips en los que se encuentran las atracciones que van a cargarse a continuación. Y por tener tabla base, GeneXus escribe otro For each implícito para navegar esa tabla base, Attraction, al que le agrega todas las cláusulas correspondientes de acuerdo a lo que el desarrollador explicitó en las propiedades del grid. En nuestro caso, solo habíamos colocado transacción base y cláusula order. Además agrega las condiciones implícitas que tienen que ver, justamente, conque este grid está anidado a otro y existe relación entre las tablas. Por eso va a recorrer solamente los registros de la tabla de atracciones que coincidan con el país y ciudad del registro que se cargó en el for each externo. Y pada cada uno, va a ejecutar el Load de este grid anidado. Y luego va a cargar la línea en el grid.

# With several Grids: nested

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

```
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

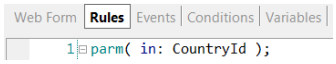
```
Event Grid2.Load
     &trips = Count(TripDate)
     &totalTrips = &totalTrips + &trips
Endevent
```

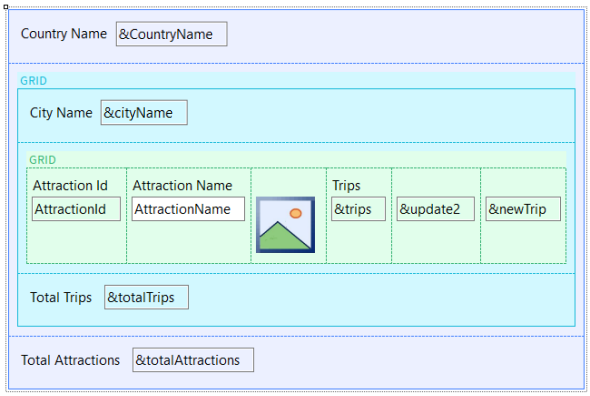
Load

endfor

Por supuesto, si el primer grid fuera sin tabla base, entonces el For each implícito desaparece, así como el comando Load.

## With several Grids: nested



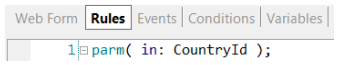

```
Event Grid1.Refresh
     &totalAttractions = 0
endevent

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
```

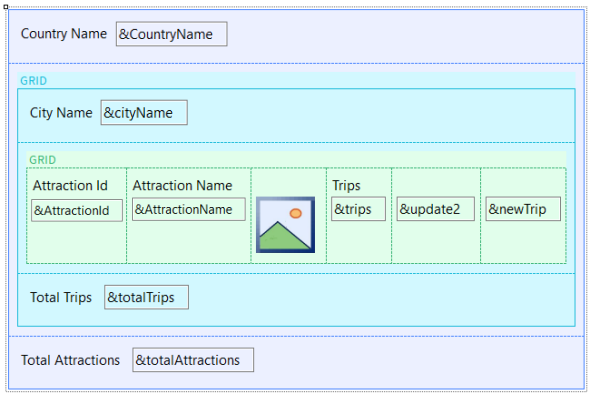
Y tendremos que escribirlos explícitamente en el evento Load del grid.

En este caso, al encontrar el comando Load dentro del evento Load del grid, por estar los grids anidados, GeneXus disparará inmediatamente el evento Refresh y Load del grid anidado. Y allí dependerá, otra vez, de si el grid anidado tiene o no tabla base, para que GeneXus coloque o no un for each implícito y su Load.

Solo que en este caso, tendremos que explicitar el where de ciudades, que antes eran implícito, como entenderemos enseguida.

## With several Grids: nested



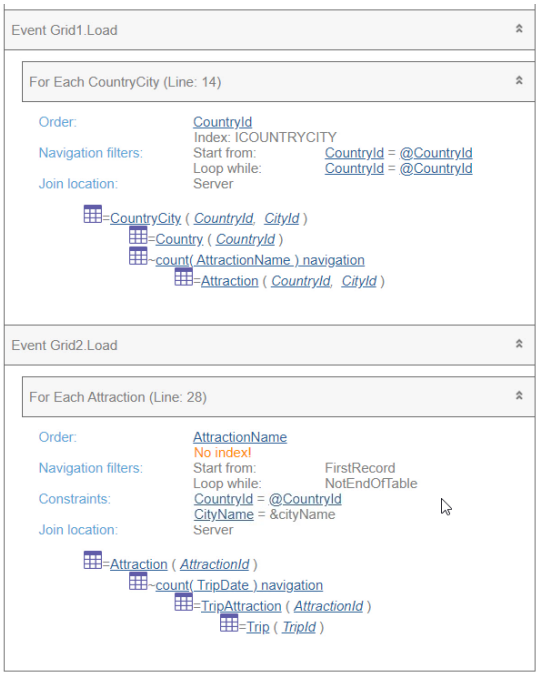

```
Event Grid1.Refresh
     &totalAttractions = 0
endevent

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
```

Si quisiéramos que el grid anidado no tenga tabla base, entonces claramente sustituiremos los atributos por variables en el grid y el for each lo tendremos que programar explícitamente en el evento Load, así como el comando Load.

Justamente, por no tener tablas base y quedar completamente en manos del desarrollador la lógica para la carga de los grids, GeneXus no puede establecer el join automático entre los For eachs, razón por la que debemos explicitar los filtros. Colocamos solo el filtro por CityName, porque el filtro por CountryId ya se va a hacer debido al parámetro.

## 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
```

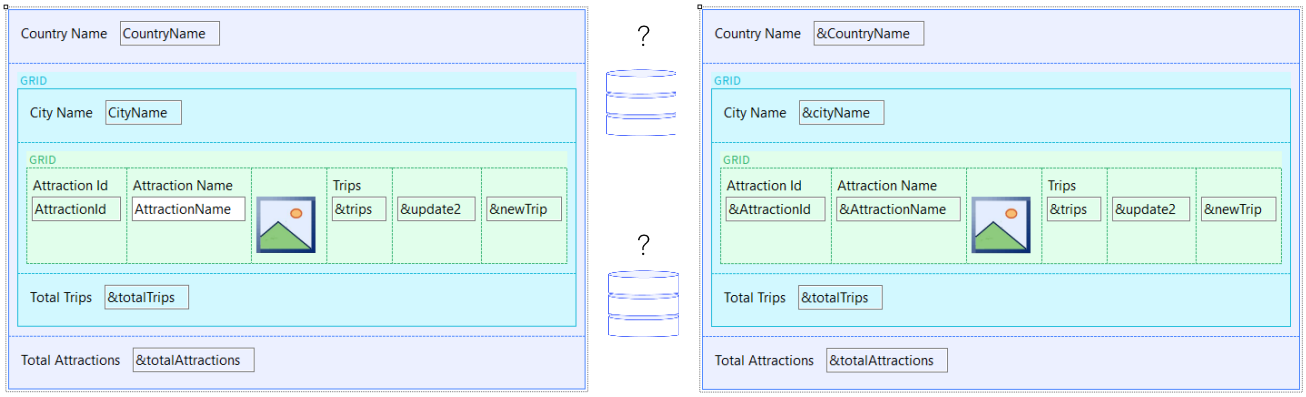



Aquí lo tenemos implementado en GeneXus.

Si observamos su listado de navegación vemos que claramente no eligió tabla base para ninguno de los dos grids. Y si ejecutamos... no vemos ninguna diferencia con el web panel que tenía tablas bases para ambos grids.

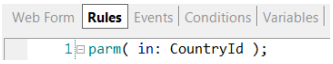
# With several Grids: nested

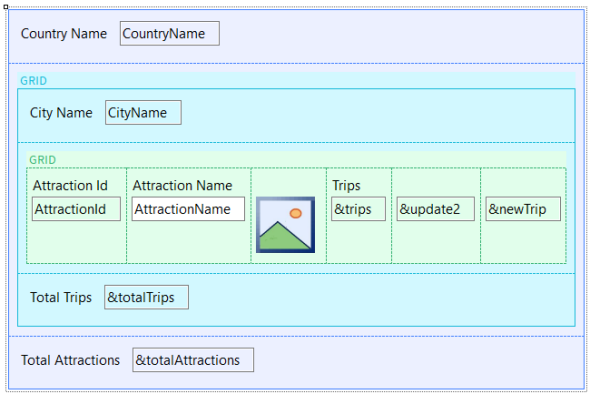
**Base Tables**




Pero para llegar a estas navegaciones primero hubo que determinar las tablas base, análogamente a como sucede con los For eachs.

# With several Grids: nested




1st GRID

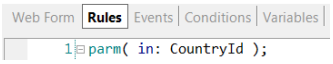
- Attributes in the **grid** (visible or hidden)
- Grid **Base Trn** property
- Grid **Order** property
- Grid **Conditions** property
- Grid **Unique** property
- Grid **Data Selector** property
- Attributes in the grid's **Load event** (without context, ie: For each command and inline aggregate formula)

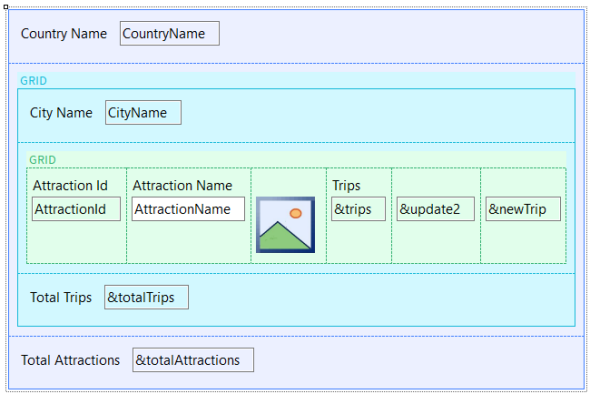
+

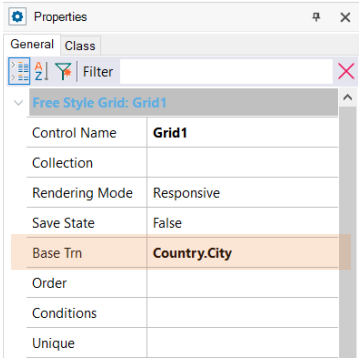
- Fixed-part attributes

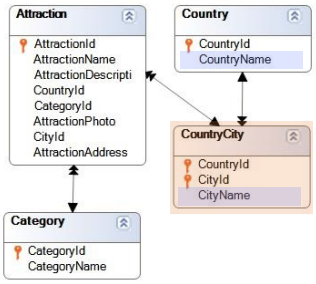
Para determinar la tabla base del primer grid de la pantalla, vale exactamente lo mismo que vimos para grids paralelos.
Es decir, GeneXus toma en cuenta los atributos del propio grid más los de la parte fija de la pantalla. Además, por supuesto, los que aparezcan en las propiedades del grid (Base transaction, Order, etc) y los del evento Load del grid. No los del Refresh del propio grid ni de ningún otro evento.

## With several Grids: nested



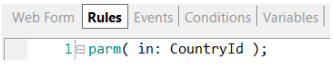



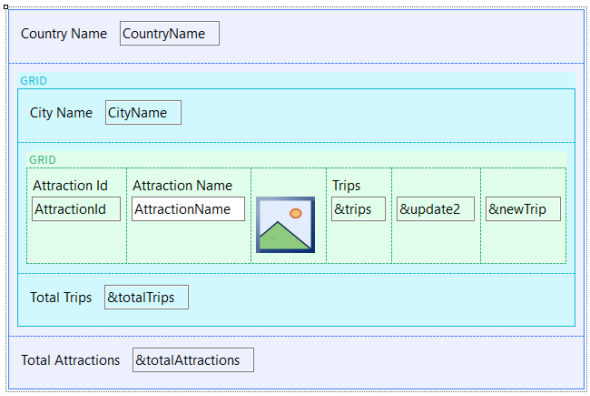



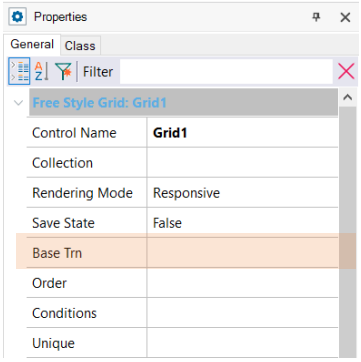



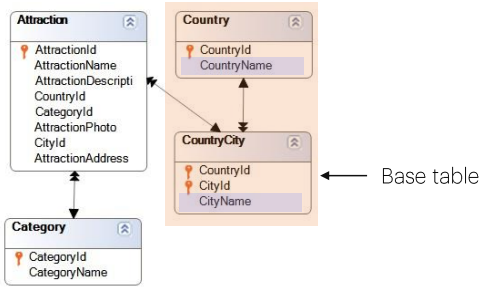

Si hay transacción base especificada, la suya será la tabla base y todos esos atributos que mencionamos deberán pertenecer a su tabla extendida.

## With several Grids: nested



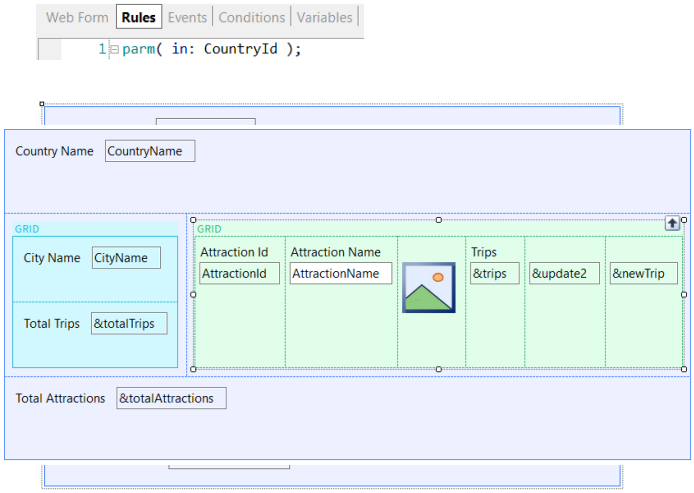




Si no hay transacción base especificada, entonces GeneXus encuentra la **mínima tabla extendida** que contenga a todos los atributos mencionados y elige a su tabla base como la tabla base del grid.

# With several Grids: nested



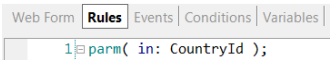

Nested GRID

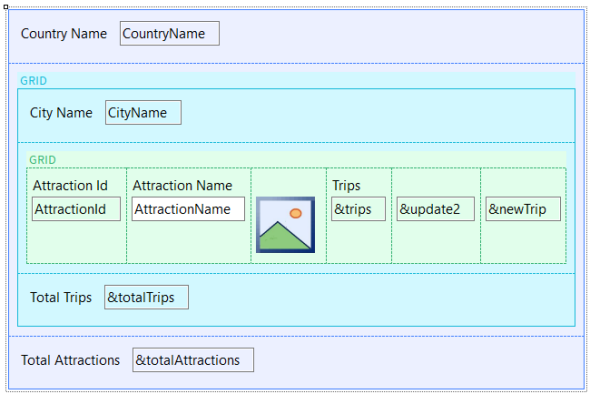
- Attributes in the **grid** (visible or hidden)
- Grid **Base Trn** property
- Grid **Order** property
- Grid **Conditions** property
- Grid **Unique** property
- Grid **Data Selector** property
- Attributes in the grid's **Load event** (without context, ie: For each command and inline aggregate formula)

La tabla base del grid anidado se determina como si el grid fuera paralelo y no anidado, pero con una salvedad, que es la misma que para determinar la tabla base de un for each anidado.

Si el grid anidado NO TIENE ESPECIFICADA TRANSACCIÓN BASE, entonces GeneXus debe determinarla por sí mismo, y aquí es donde el hecho de que este grid esté anidado a otro puede determinar una tabla distinta a la que se determinaría si el grid fuera paralelo.

# With several Grids: nested



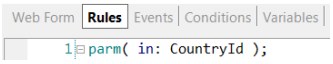



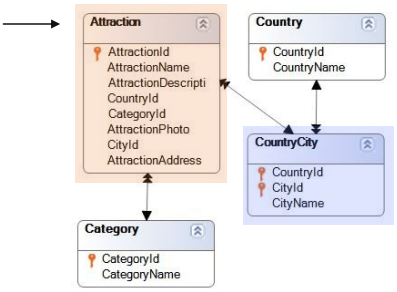

Nested GRID

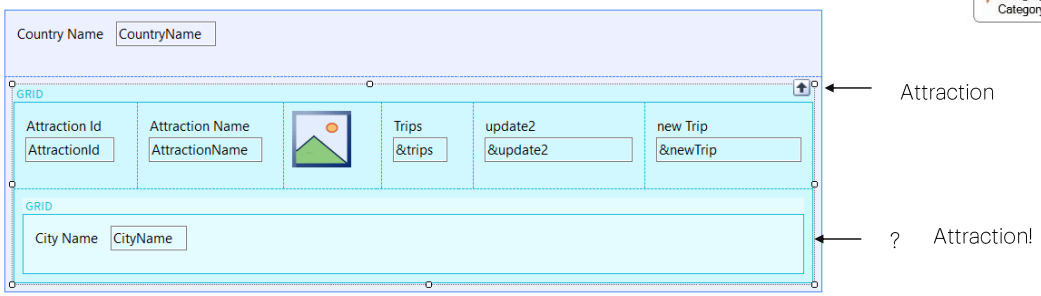
- Attributes in the **grid** (visible or hidden)
- Grid **Base Trn** property
- Grid **Order** property
- Grid **Conditions** property
- Grid **Unique** property
- Grid **Data Selector** property
- Attributes in the grid's **Load event** (without context, ie: For each command and inline aggregate formula)

Son casos poco frecuentes, pero es bueno estar advertido.

## With several Grids: nested




Por ejemplo, si los grids estuviesen invertidos, y el externo navegara la tabla Attraction y en el interno no hubiera especificada transacción base y estuviera solo el atributo CityName implicado, si el grid fuera paralelo, claramente GeneXus determinaría como su tabla base la de ciudades. Sin embargo, en este caso, por estar anidado a un grid que tiene una tabla extendida que incluye a los atributos del segundo grid, entonces elegirá para este segundo grid la misma tabla base que la del primero, implementando, así, un corte de control.

## Base tables: ready!

## And its navigations!

Con esto, terminamos de estudiar cómo se determinan las tablas base y las navegaciones para todos los casos de web panels.

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