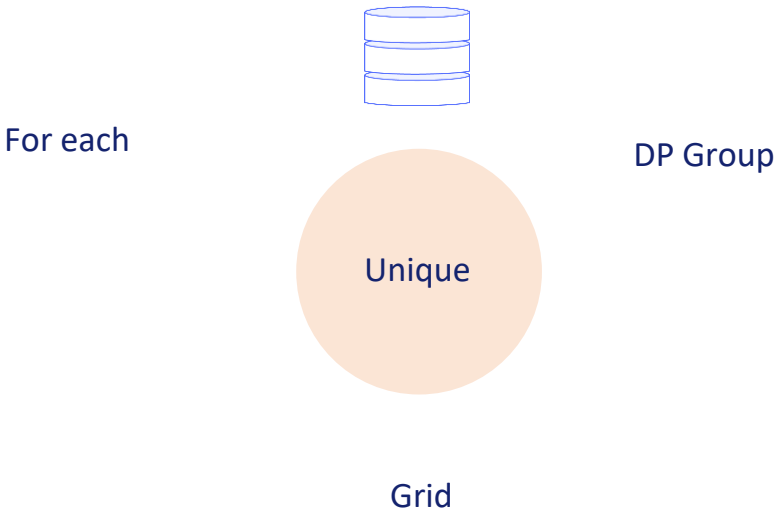
Lógica de consulta a la base de datos con GeneXus

For each: cláusula Unique

GeneXus™



Veremos en algún detalle la cláusula unique aplicada al comando For each pero sabiendo que se puede utilizar análogamente en grupos de data providers y grids con tabla base.

```
For each    BaseTrn1, … , BaseTrnn
        skip expression1 count expression2
        order  att1, att2, … , attn [when condition]
        order att1, att2, … , attn [when condition]
        order none [when condition]
        unique att1, att2, … , attn
        using DataSelector ( parm1, parm2, … , parmn )
        where  condition [when condition]
        where  condition [when condition]
        where  att  IN DataSelector ( parm1, parm2, … , parmn )
        blocking n
                main_code
        when duplicate
                when_duplicate_code
        when none
                when_none_code
endfor
```

Aquí la vemos entre las demás cláusulas del For each.

Navigation group

unique D, E → **Print info //** D, E

A*	B*	C	D	E	F	G	H	I

Cuando para muchos registros se repite el valor de un conjunto de atributos, en este ejemplo, D y E, podemos utilizar la cláusula unique para poder trabajar con uno de todos los registros para los que se da esa repetición (como si fuera un representante del grupo); por ejemplo, imprimiendo los valores de esos atributos (ya que van a ser los mismos para todos los registros del grupo), y luego pasar al siguiente grupo, para hacer lo mismo. Y así sucesivamente hasta agotarlos todos.

Navigation group

unique D, E → **Print info //** D, E

A*	B*	C	D	E	F	G

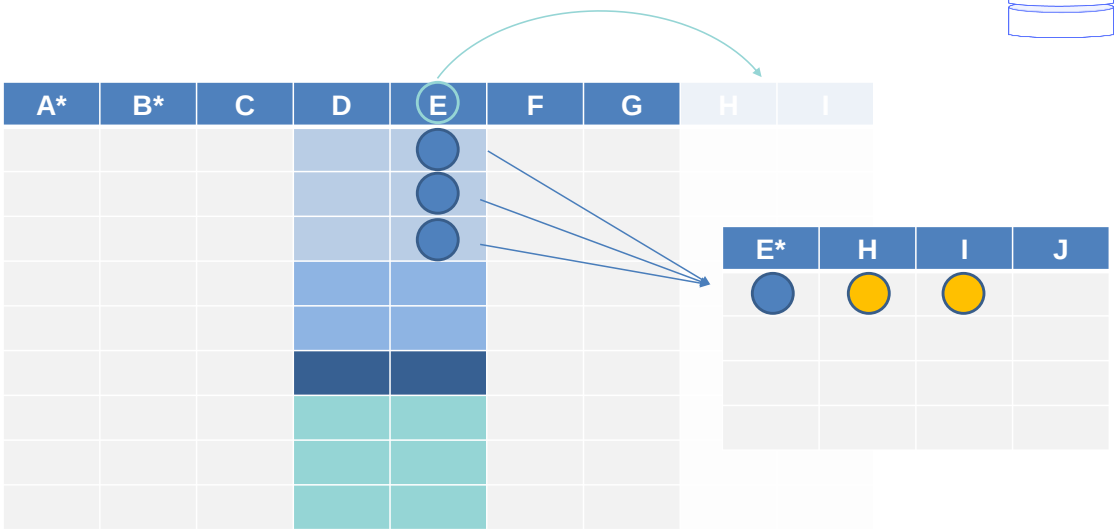
```
For each BaseTrn1, ... , BaseTrnn
    skip expression1 count expression2
    order att1, att2, ... , attn [when condition]
    order att1, att2, ... , attn [when condition]
    order none [when condition]
    unique att1, att2, ... , attn
    using DataSelector ( parm1, parm2, ... , parmn )
    where condition [when condition]
    where condition [when condition]
    where att IN DataSelector ( parm1, parm2, ... , parmn )
    blocking n
        main_code
    when duplicate
        when_duplicate_code
    when none
        when_none_code
endfor
```

Para que esto tenga sentido, dentro del código que se ejecutará para cada grupo solo pueden aparecer atributos cuyo valor sea único para todos y cada uno de los registros del grupo.

Los atributos allí presentes no tienen por qué formar parte de la misma tabla. Pueden estar en la extendida.

Navigation group

unique D, E → **Print info //** D, E



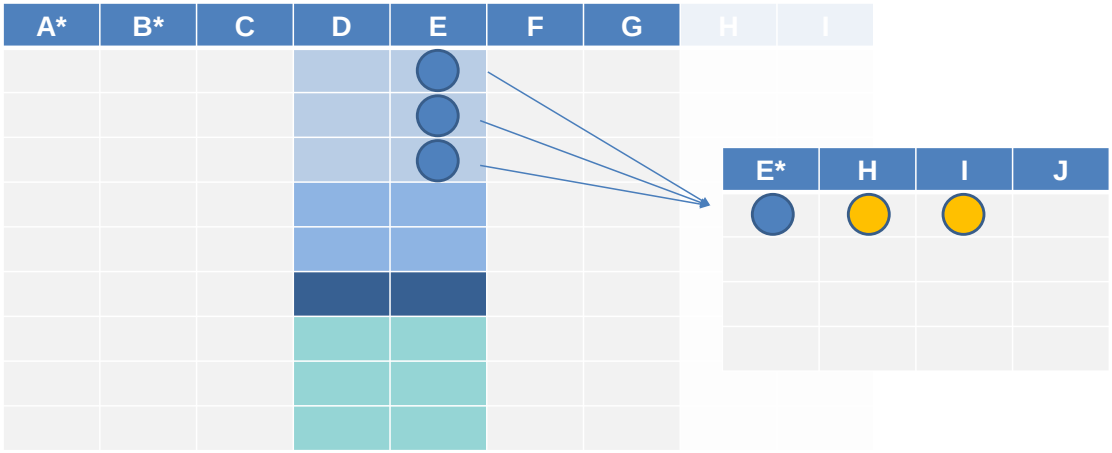
Supongamos que esta es una representación gráfica de la tabla extendida y no una tabla física.

Por ejemplo, supongamos que E es una clave foránea que determina a H y a I. O sea, que existe esta tabla física.

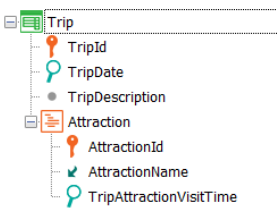
Esto significa que para todos los registros del grupo, como los valores de E son el mismo, los valores de H y de I también lo serán.

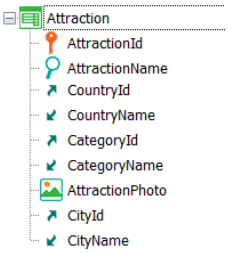
Navigation group

unique D, E → **Print info //** D, E , H, I



Esto hace que también podamos utilizar los atributos H e I dentro del código que se ejecutará para el grupo, dado que también serán únicos para ese conjunto de registros.






```
for each Trip.Attraction
   order AttractionId
      for each Trip.Attraction
      endfor
      print AttractionInfo //AttractionName
endfor


for each Trip.Attraction
   unique AttractionId
      print AttractionInfo //AttractionName
endfor
```

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
2	10	120
2	12	90

AttractionId*	AttractionName	...
1	Louvre Museum	...
2	The Great Wall	...
3	Eiffel Tower	...
4	Christ the Redemmer	...
5	Smithsonian Institute	...
6	Matisse Museum	....
7	Forbidden city	...
10	Rifugio Nuvolau	...
12	Cinque Terre	

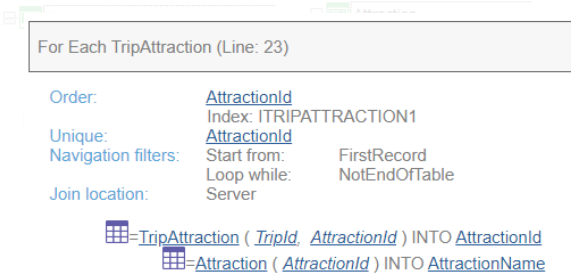
Veamos ejemplos.

Tenemos las excursiones que se pueden realizar a distintas atracciones turísticas que tienen asignado un tiempo de duración de cada visita.

Queremos obtener un listado de las atracciones turísticas que participan de excursiones. Imaginemos que estos son los datos actuales de las tablas. Vamos a querer listar únicamente estas atracciones.

Una primera alternativa que se nos podría ocurrir es implementar un corte de control que navegue TripAttraction y agrupe por AttractionId. De esta manera estaremos seguros de solo listar atracciones que efectivamente están en excursiones, y además, por colocar el comando print luego del for each anidado (aunque lo mismo hubiese sido colocarlo antes), sabemos que estaremos por cada grupo imprimiendo únicamente el AttractionName que se repite.

Esto mismo lo podemos resolver de modo mucho más sencillo utilizando la cláusula Unique. Es su caso de uso más evidente.




```
for each Trip.Attraction
   order AttractionId
      for each Trip.Attraction
      endfor
      print AttractionInfo //AttractionName
endfor
```

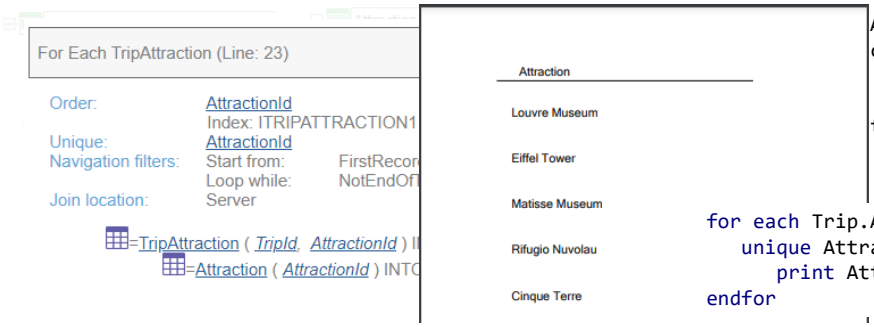
```
for each Trip.Attraction
   unique AttractionId
      print AttractionInfo //AttractionName
endfor
```

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
2	10	120
2	12	90

AttractionId*	AttractionName	...
1	Louvre Museum	...
2	The Great Wall	...
3	Eiffel Tower	...
4	Christ the Redemmer	...
5	Smithsonian Institute	...
6	Matisse Museum	....
7	Forbidden city	...
10	Rifugio Nuvolau	...
12	Cinque Terre	

Si observamos el listado de navegación del For each con Unique vemos que como se tiene un índice por AttractionId en TripAttraction (por ser clave foránea) elegirá ordenar por ese atributo.

Por tanto comienza por el primer grupo donde se repite el valor de AttractionId y se imprime en la salida su AttractionName (que es único para todos los registros del grupo): Louvre Museum.



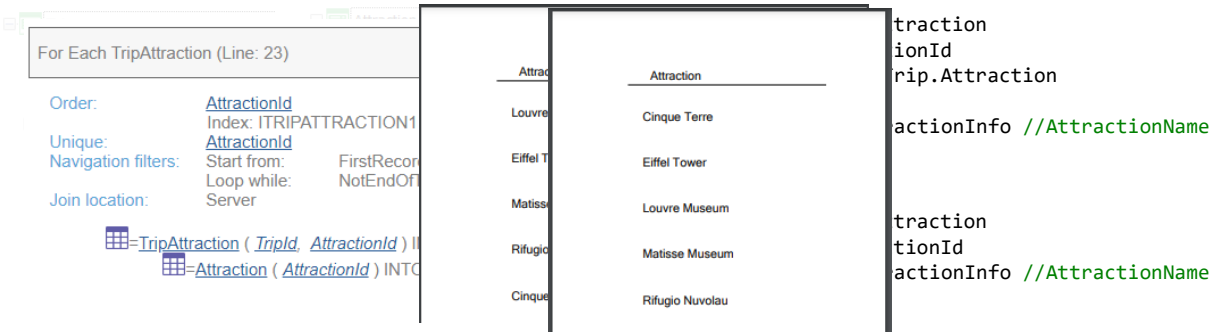

```
Attraction
ctionId
 Trip.Attraction

tractionInfo //AttractionName
```

```
for each Trip.Attraction
   unique AttractionId
      print AttractionInfo //AttractionName
endfor
```

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
2	10	120
2	12	90

AttractionId*	AttractionName	...
1	Louvre Museum	...
2	The Great Wall	...
3	Eiffel Tower	...
4	Christ the Redemmer	...
5	Smithsonian Institute	...
6	Matisse Museum	....
7	Forbidden city	...
10	Rifugio Nuvolau	...
12	Cinque Terre	

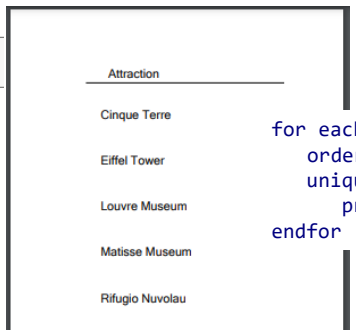
Luego el siguiente grupo es de un solo registro: se imprime Eiffel Tower.
Luego la atracción de id 6, que es Matisse Museum. Luego la 10, y por último la 12.



TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
2	10	120
2	12	90

AttractionId*	AttractionName	...
1	Louvre Museum	...
2	The Great Wall	...
3	Eiffel Tower	...
4	Christ the Redemmer	...
5	Smithsonian Institute	...
6	Matisse Museum	....
7	Forbidden city	...
10	Rifugio Nuvolau	...
12	Cinque Terre	

Si nos interesara que salieran ordenadas esas atracciones por nombre de atracción…




```
for each Trip.Attraction
   order AttractionName
   unique AttractionId
      print AttractionInfo //AttractionName
endfor
```

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
2	10	120
2	12	90

AttractionId*	AttractionName	...
1	Louvre Museum	...
2	The Great Wall	...
3	Eiffel Tower	...
4	Christ the Redemmer	...
5	Smithsonian Institute	...
6	Matisse Museum	....
7	Forbidden city	...
10	Rifugio Nuvolau	...
12	Cinque Terre	

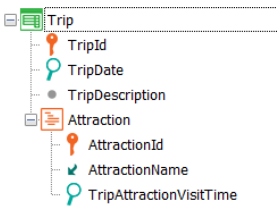
…podríamos agregar la cláusula order.

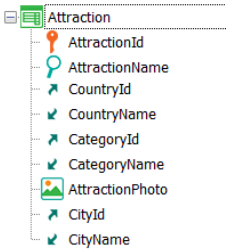
Y así vemos el listado de navegación. El valor único que se busca sigue siendo AttractionId pero se orderará el resultado de la consulta por AttractionName.

¿Podríamos en vez de hacer esto utilizar en la cláusula unique el atributo AttractionName directamente?

Sí, pero observemos dos cosas. Por un lado hacer esto no nos asegura que el listado salga además ordenado por AttractionName. Observemos que el listado de navegación está indicando Order NONE. Es que GeneXus no sabe de la existencia de un índice por AttractionName. Por lo que igual deberíamos ordenar por AttractionName si es lo que queremos.

Pero por otro lado pensemos qué sucedería si tuviéramos en la base de datos dos atracciones con el mismo nombre.






```
for each Trip.Attraction
   order AttractionName
   unique AttractionId
      print AttractionInfo //AttractionName
endfor

for each Trip.Attraction
   order AttractionName
   unique AttractionName
      print AttractionInfo //AttractionName
endfor
```

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
5	8	60
2	10	120
2	12	90

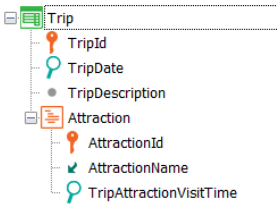
AttractionId*	AttractionName	...
1	Louvre Museum	...
2	The Great Wall	...
3	Eiffel Tower	...
4	Christ the Redemmer	...
5	Smithsonian Institute	...
6	Matisse Museum	....
7	Forbidden city	...
8	Louvre Museum	
10	Rifugio Nuvolau	...
12	Cinque Terre	

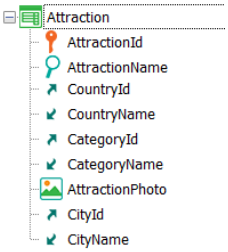
Por ejemplo, la 1 y la 8. Si no tenemos un índice unique por AttractionName esto estará permitido. Y observemos que tenemos a la atracción 1 en 2 trips, y a la 8 en 1.

Entre este For each y este otro la única diferencia es la cláusula unique.

En el primer caso estos dos registros se trabajarán juntos, y se listará Louvre Museum; y este se trabajará por otro lado, en otro grupo, y se listará de vuelta Louvre Museum.

En cambio en el segundo caso los tres registros estarán dentro del mismo grupo, y se listará una sola vez Louvre Museum, aunque corresponda a dos atracciones distintas.



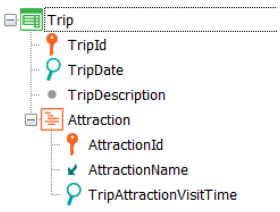


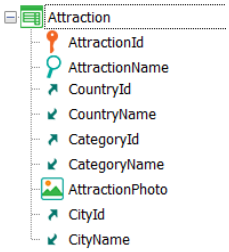

```
for each Trip.Attraction
   order AttractionName
   unique AttractionId
      print AttractionInfo //AttractionName
endfor

for each Trip.Attraction
   order AttractionName
   unique AttractionName
      print AttractionInfo //AttractionName
endfor
```

TripId*	AttractionId*	TripAttractionVisitTime	AttractionName
1	1	180	Louvre Museum
3	1	200	Louvre Museum
1	3	120	Eiffel Tower
3	6	180	Matisse Museum
4	6	240	Matisse Museum
5	8	60	Louvre Museum
2	10	120	Rifugio Nuvolau
2	12	90	Cinque Terre

AttractionId*	AttractionName	...
1	Louvre Museum	...
2	The Great Wall	...
3	Eiffel Tower	...
4	Christ the Redemmer	...
5	Smithsonian Institute	...
6	Matisse Museum	....
7	Forbidden city	...
8	Louvre Museum	
10	Rifugio Nuvolau	...
12	Cinque Terre	

Lo vemos muy claro si imaginamos los datos así, con la tabla extendida como una supertabla.






```
for each Trip.Attraction
   order AttractionName
   unique AttractionId
      print AttractionInfo //AttractionName
endfor

for each Trip.Attraction
   order AttractionName
   unique AttractionName
      print AttractionInfo //AttractionName
endfor
```

TripId*	AttractionId*	TripAttractionVisitTime	AttractionName
2	12	90	Cinque Terre
1	3	120	Eiffel Tower
1	1	180	Louvre Museum
3	1	200	Louvre Museum
5	8	60	Louvre Museum
3	6	180	Matisse Museum
4	6	240	Matisse Museum
2	10	120	Rifugio Nuvolau

AttractionId*	AttractionName	...
1	Louvre Museum	...
2	The Great Wall	...
3	Eiffel Tower	...
4	Christ the Redemmer	...
5	Smithsonian Institute	...
6	Matisse Museum	....
7	Forbidden city	...
8	Louvre Museum	
10	Rifugio Nuvolau	...
12	Cinque Terre	

Si la ordenamos por AttractionName lo vemos con más claridad… Observemos que no hay ningún problema en que el atributo de la cláusula unique esté en la tabla extendida y no en la base.

En definitiva, no será lo mismo pedir por valores únicos para AttractionId que para AttractionName.

Si existe el índice unique entonces sí, el resultado de ambos for eachs será exactamente el mismo, porque no se dará que pueda existir este registro 8.

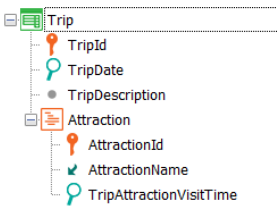
Navigation group

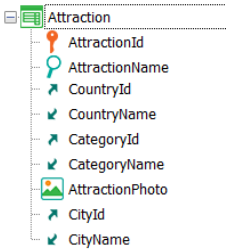
unique D, E → **&qty = Count(C)**
Print info // D, E , &qty

A*	B*	C	D	E	F	G	H	I

Además podemos no solo quedarnos con uno de los registros que se repiten para hacer algo con la info que no varía (como imprimirla), sino que además podemos ejecutar fórmulas de agregación sobre los repetidos, que por ejemplo los cuenten. La fórmula debe navegar, por supuesto, la misma tabla.

Así, se toma el primer grupo y se ejecuta el count sobre sus registros (dará 3 en este caso). Y se imprime D y E, info que es única para ese grupo, y 3.
Luego el siguiente grupo, para el que count dará 2.
Luego el tercero, para el que dará 1. Y por último el cuarto para el que dará 3.



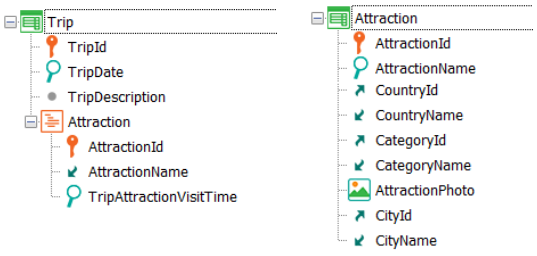



```
for each Trip.Attraction
   unique AttractionId
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //AttractionName, &qty
endfor
```

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
5	8	60
2	10	120
2	12	90

AttractionId*	AttractionName	...
1	Louvre Museum	...
2	The Great Wall	...
3	Eiffel Tower	...
4	Christ the Redemmer	...
5	Smithsonian Institute	...
6	Matisse Museum	....
7	Forbidden city	...
8	Louvre Museum	
10	Rifugio Nuvolau	...
12	Cinque Terre	

En este ejemplo, además de querer quedarnos con AttractionId no repetido para listar su nombre, queremos contar cuántas veces aparece repetido. En definitiva, en cuántos trips está.




```
for each Trip.Attraction
   unique AttractionId
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //AttractionName, &qty
endfor
```

For Each TripAttraction (Line: 23)

Order: AttractionId
Index: ITRIPATTRACTION1
Unique: AttractionId
Navigation filters: Start from: FirstRecord
Loop while: NotEndOfTable
Join location: Server

=TripAttraction (*TripId*, *AttractionId*) INTO AttractionId
=Attraction (*AttractionId*) INTO AttractionName
=count(TripAttractionVisitTime) navigation (*AttractionId*)

Formulas

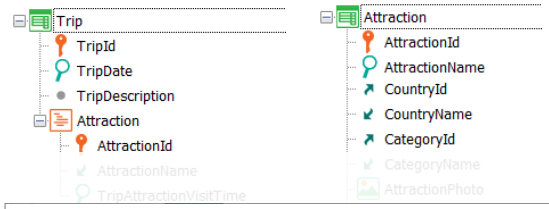
Navigation to evaluate: count(TripAttractionVisitTime)

Given: AttractionId
Index: ITRIPATTRACTION
Group by: AttractionId

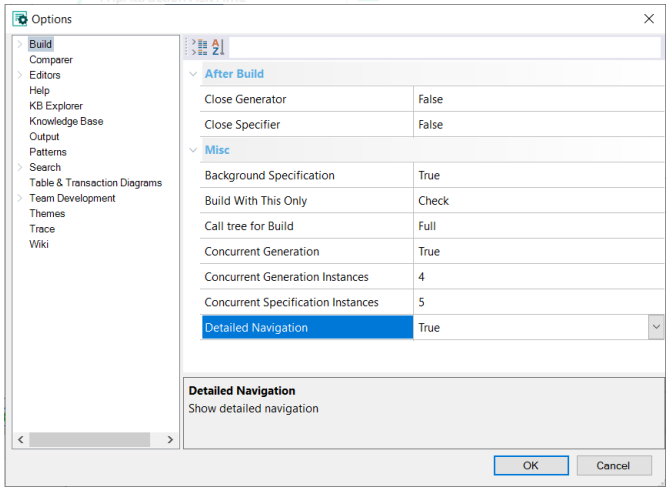
=TripAttraction (*AttractionId*)

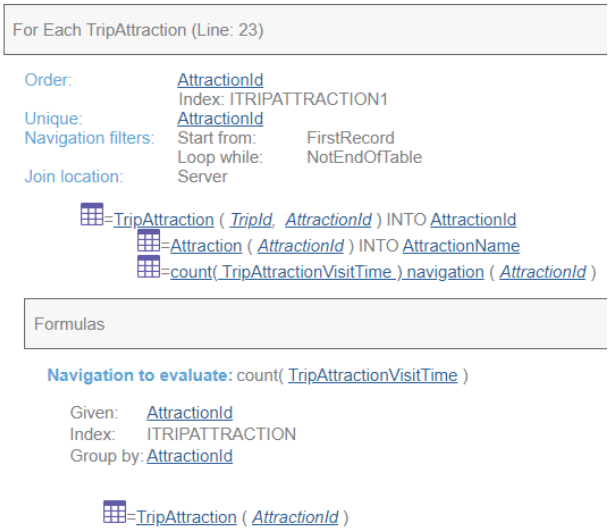
TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
5	8	60
2	10	120
2	12	90

La fórmula Count está utilizando el atributo secundario de la tabla que se quiere navegar, por lo que al determinarse esto, la fórmula Count tendrá un comportamiento especial: va a agrupar por el atributo de unique, tal como vemos en el listado de navegación.




```
for each Trip.Attraction
   unique AttractionId
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //AttractionName, &qty
endfor
```





Recordemos que para que el listado nos muestre la navegación de la fórmula debemos activar la navegación detallada… a través de Tools/options…

Attraction	Trips
Louvre Museum	2
Eiffel Tower	1
Matisse Museum	2
Louvre Museum	1
Rifugio Nuvolau	1
Cinque Terre	1

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
5	8	60
2	10	120
2	12	90

```
for each Trip.Attraction
   unique AttractionId
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //AttractionName, &qty
endfor
```

For Each TripAttraction (Line: 23)

Order: AttractionId
Index: ITRIPATTRACTION1
Unique: AttractionId
Navigation filters: Start from: FirstRecord
Loop while: NotEndOfTable
Join location: Server

=TripAttraction (*TripId*, *AttractionId*) INTO AttractionId
=Attraction (*AttractionId*) INTO AttractionName
=count(TripAttractionVisitTime) navigation (*AttractionId*)

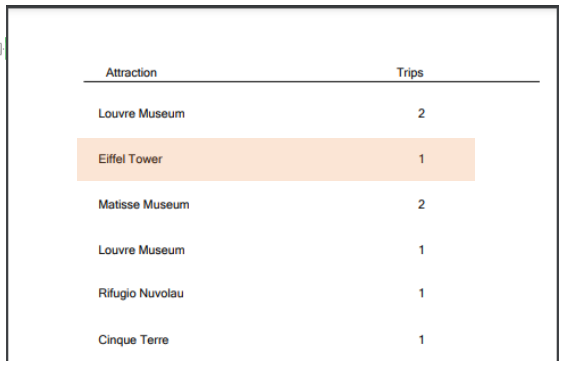
Formulas

Navigation to evaluate: count(TripAttractionVisitTime)

Given: AttractionId
Index: ITRIPATTRACTION
Group by: AttractionId

=TripAttraction (*AttractionId*)

Entonces, para cada grupo de repetidos, se contarán los registros para ese AttractionId dado, el de cada grupo. Así se obtiene el primer grupo con AttractionId repetido, se cuentan sus registros, los que tengan el mismo AttractionId, y se imprime en la salida el nombre de atracción y esa cantidad.



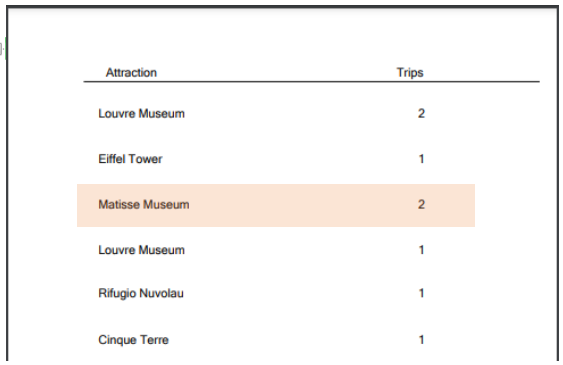
Attraction	Trips
Louvre Museum	2
Eiffel Tower	1
Matisse Museum	2
Louvre Museum	1
Rifugio Nuvolau	1
Cinque Terre	1

```
for each Trip.Attraction
   unique AttractionId
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //AttractionName, &qty
endfor
```

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
5	8	60
2	10	120
2	12	90

AttractionId*	AttractionName	...
1	Louvre Museum	...
2	The Great Wall	...
3	Eiffel Tower	...
4	Christ the Redemmer	...
5	Smithsonian Institute	...
6	Matisse Museum	....
7	Forbidden city	...
8	Louvre Museum	
10	Rifugio Nuvolau	...
12	Cinque Terre	

Luego el próximo grupo, para el que el count da 1.



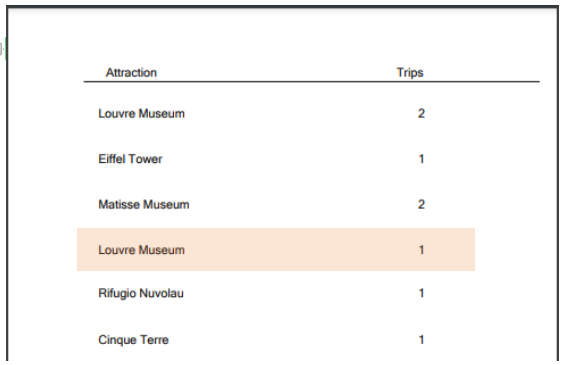
Attraction	Trips
Louvre Museum	2
Eiffel Tower	1
Matisse Museum	2
Louvre Museum	1
Rifugio Nuvolau	1
Cinque Terre	1

```
for each Trip.Attraction
   unique AttractionId
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //AttractionName, &qty
endfor
```

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
5	8	60
2	10	120
2	12	90

AttractionId*	AttractionName	...
1	Louvre Museum	...
2	The Great Wall	...
3	Eiffel Tower	...
4	Christ the Redemmer	...
5	Smithsonian Institute	...
6	Matisse Museum	....
7	Forbidden city	...
8	Louvre Museum	
10	Rifugio Nuvolau	...
12	Cinque Terre	

Luego el siguiente, que da 2.



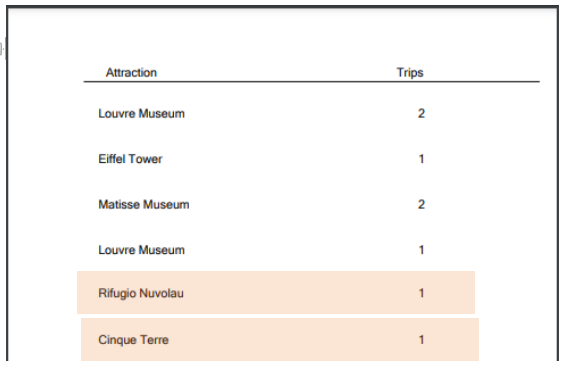
Attraction	Trips
Louvre Museum	2
Eiffel Tower	1
Matisse Museum	2
Louvre Museum	1
Rifugio Nuvolau	1
Cinque Terre	1

```
for each Trip.Attraction
   unique AttractionId
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //AttractionName, &qty
endfor
```

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
5	8	60
2	10	120
2	12	90

AttractionId*	AttractionName	...
1	Louvre Museum	...
2	The Great Wall	...
3	Eiffel Tower	...
4	Christ the Redemmer	...
5	Smithsonian Institute	...
6	Matisse Museum	....
7	Forbidden city	...
8	Louvre Museum	
10	Rifugio Nuvolau	...
12	Cinque Terre	

Luego, el siguiente con el mismo nombre que el primero, da 1.



Attraction	Trips
Louvre Museum	2
Eiffel Tower	1
Matisse Museum	2
Louvre Museum	1
Rifugio Nuvolau	1
Cinque Terre	1

```
for each Trip.Attraction
   unique AttractionId
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //AttractionName, &qty
endfor
```

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
5	8	60
2	10	120
2	12	90

AttractionId*	AttractionName	...
1	Louvre Museum	...
2	The Great Wall	...
3	Eiffel Tower	...
4	Christ the Redemmer	...
5	Smithsonian Institute	...
6	Matisse Museum	....
7	Forbidden city	...
8	Louvre Museum	
10	Rifugio Nuvolau	...
12	Cinque Terre	

El siguiente también da 1, y el último, también.

En cambio, si en lugar de AttractionId…

Attraction	Trips
Cinque Terre	1
Eiffel Tower	1
Louvre Museum	3
Matisse Museum	2
Rifugio Nuvolau	1

```
for each Trip.Attraction
   unique AttractionName
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //AttractionName, &qty
endfor
```

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
5	8	60
2	10	120
2	12	90

AttractionId*	AttractionName	...
1	Louvre Museum	...
2	The Great Wall	...
3	Eiffel Tower	...
4	Christ the Redemmer	...
5	Smithsonian Institute	...
6	Matisse Museum	....
7	Forbidden city	...
8	Louvre Museum	
10	Rifugio Nuvolau	...
12	Cinque Terre	

…utilizamos AttractionName en la cláusula unique, entonces el grupo correspondiente a Louvre Museum contará 3 registros.

Attraction	Trips
Cinque Terre	1
Eiffel Tower	1
Louvre Museum	3
Matisse Museum	2
Rifugio Nuvolau	1

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
5	8	60
2	10	120
2	12	90

```
for each Trip.Attraction
   unique AttractionName
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //AttractionName, &qty
endfor
```

For Each TripAttraction (Line: 23)

Order: NONE
Unique: AttractionName
Navigation filters: Start from: FirstRecord
Loop while: NotEndOfTable
Join location: Server

=TripAttraction (*TripId*, *AttractionId*) INTO AttractionId
=Attraction (*AttractionId*) INTO AttractionName
=count(TripAttractionVisitTime) navigation (*AttractionName*)

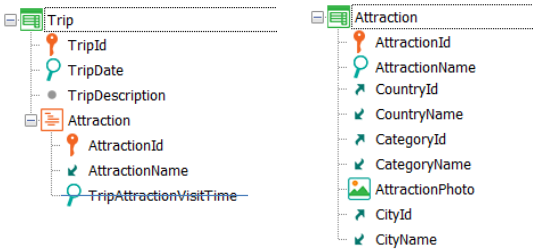
Formulas

Navigation to evaluate: count(TripAttractionVisitTime)

Given: AttractionName
Index: ITRIPATTRACTION
Group by: AttractionName

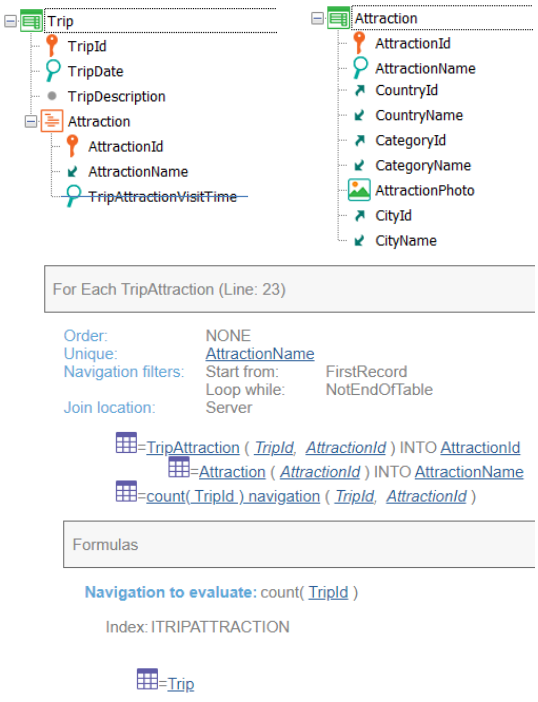
=TripAttraction
=Attraction (*AttractionId*)

En la navegación vemos el Given y el Group by.

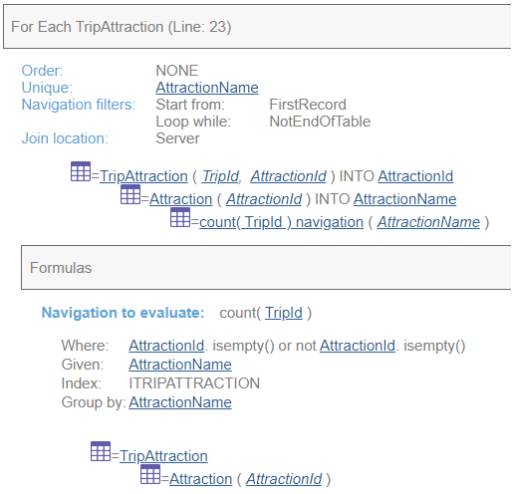



```
for each Trip.Attraction
   unique AttractionName
      &qty = Count(TripAttractionVisitTime)

      print AttractionInfo //AttractionName, &qty
endfor
```

Atendamos a este caso particular. Si en la tabla que queremos navegar no hay ningún atributo secundario, entonces posiblemente debamos hacer algo para que GeneXus entienda que se quiere navegar esa tabla para la fórmula.

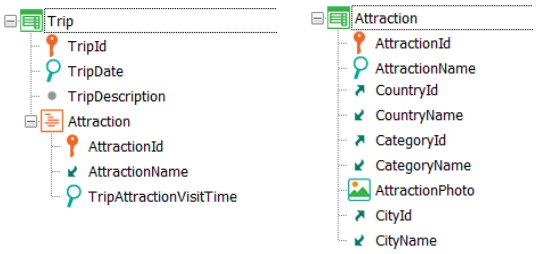



```
for each Trip.Attraction
   unique AttractionName
      &qty = Count(TripId), AttractionId.IsEmpty() or
                                 not AttractionId.IsEmpty())
      print AttractionInfo //AttractionName, &qty
endfor
```



Es decir, si, por ejemplo colocamos para el Count el atributo TripId y en unique dejamos AttractionName, puede que GeneXus no elija la tabla TripAttraction para resolver la fórmula Count, sino Trip, y el resultado no será el deseado.
Veamos que se informa que navegará la tabla Trip y contará todos los trips entonces, porque no hay ninguna condición informada para la fórmula.

Necesitamos que elija navegar TripAttraction para que haga lo que queremos.
Como no tenemos trn base para las fórmulas, podemos usar un truco: agregar una condición que siempre sea true y que contenga un atributo que haga determinar la tabla base que queremos. Por ejemplo, esta condición que utiliza a AttractionId y que siempre será true.
Observemos el listado de navegación indicando lo que queremos. Ahora sí está navegando TripAttraction y además está agrupando por el AttractionName dado en el for each, por tanto, solo contando los tripattractions del mismo AttractionName.




```
for each Trip.Attraction
   unique TripDate, AttractionName
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //TripDate, AttractionName, &qty
endfor
```

01/01/2023 Eiffel Tower 1

TripId*	TripDate	TripDescription
1	01/01/2023	...
2	04/04/2023	...
3	01/01/2023	...
4	05/05/2023	...

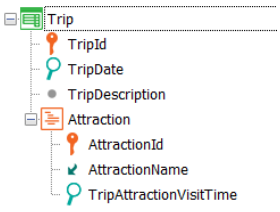
TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	6	180
4	6	240
3	8	60

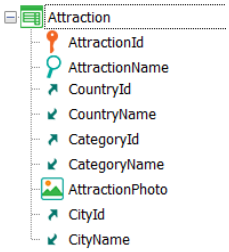
AttractionId*	AttractionName	...
3	Eiffel Tower	...
1	Louvre Museum	...
8	Louvre Museum	....
6	Matisse Museum	....

Vayamos ahora un poco más allá. Sabemos que podemos especificar varios atributos en la cláusula unique, y que no tienen por qué pertenecer a la tabla base del for each, como en este ejemplo.

Queremos contar la cantidad de trips que en una misma fecha incluyen visita a un mismo nombre de atracción. Es decir, para un mismo TripDate y AttractionName, cuántos registros hay en TripAttraction.

Si estos son los datos de las tablas (solo mostramos los registros relevantes), vemos que para Eiffel Tower solo habrá un registro en TripAttraction: el del trip 1, que es en esta fecha.






```
for each Trip.Attraction
   unique TripDate, AttractionName
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //TripDate, AttractionName, &qty
endfor
```

01/01/2023 Eiffel Tower 1

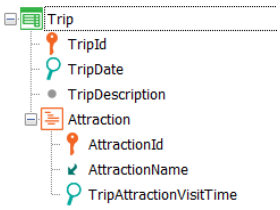
01/01/2023 Louvre Museum 2

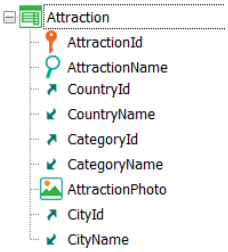
TripId*	TripDate	TripDescription
1	01/01/2023	...
2	04/04/2023	...
3	01/01/2023	...
4	05/05/2023	...

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	6	180
4	6	240
3	8	60

AttractionId*	AttractionName	...
3	Eiffel Tower	...
1	Louvre Museum	...
8	Louvre Museum	....
6	Matisse Museum	....

Al Louvre Museum lo tenemos en estos 3 registros. Si vamos a mirar las fechas, para el trip 1 y para el 3 son la misma, entonces en la salida tendremos…






```
for each Trip.Attraction
   unique TripDate, AttractionName
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //TripDate, AttractionName, &qty
endfor
```

01/01/2023 Eiffel Tower 1

01/01/2023 Louvre Museum 2

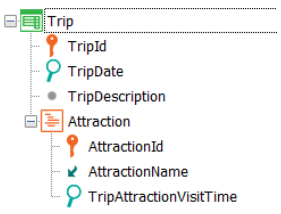
04/04/2023 Louvre Museum 1

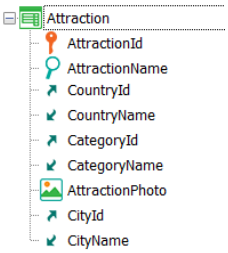
TripId*	TripDate	TripDescription
1	01/01/2023	...
2	04/04/2023	...
3	01/01/2023	...
4	05/05/2023	...

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	6	180
4	6	240
3	8	60

AttractionId*	AttractionName	...
3	Eiffel Tower	...
1	Louvre Museum	...
8	Louvre Museum	....
6	Matisse Museum	....

Y para el 2 es esta otra, así que saldrá en la salida esto.






```
for each Trip.Attraction
   unique TripDate, AttractionName
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //TripDate, AttractionName, &qty
endfor
```

TripId*	TripDate	TripDescription
1	01/01/2023	...
2	04/04/2023	...
3	01/01/2023	...
4	05/05/2023	...

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	6	180
4	6	240
3	8	60

01/01/2023 Eiffel Tower 1

01/01/2023 Louvre Museum 2

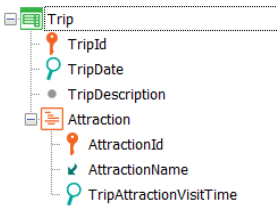
04/04/2023 Louvre Museum 1

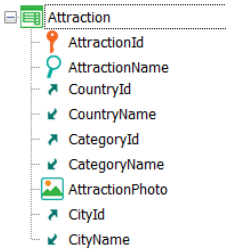
01/01/2023 Matisse Museum 1

05/05/2023 Matisse Museum 1

AttractionId*	AttractionName	...
3	Eiffel Tower	...
1	Louvre Museum	...
8	Louvre Museum	....
6	Matisse Museum	....

Y por último para Matisse Museum: tenemos el trip 3 y el 4, que como tienen fechas distintas, dará lugar a dos prints en la salida.






```
for each Trip.Attraction
   unique TripDate, AttractionName
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //TripDate, AttractionName, &qty
endfor
```

TripId*	TripDate	TripDescription
1	01/01/2023	...
2	04/04/2023	...
3	01/01/2023	...
4	05/05/2023	...

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	6	180
4	6	240
3	8	60

01/01/2023 Eiffel Tower 1

01/01/2023 Louvre Museum 2

04/04/2023 Louvre Museum 1

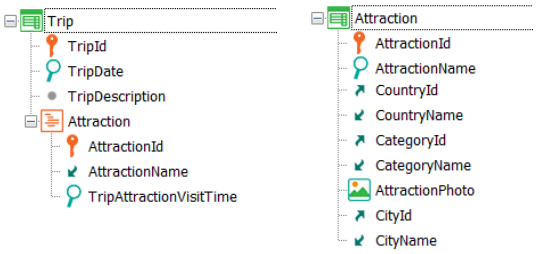
01/01/2023 Matisse Museum 1

05/05/2023 Matisse Museum 1

AttractionId*	AttractionName	...
3	Eiffel Tower	...
1	Louvre Museum	...
8	Louvre Museum	....
6	Matisse Museum	....

En este ejemplo ninguno de los dos atributos de la cláusula unique pertenecen a la tabla base del For each.

También podríamos utilizar fórmulas en la cláusula unique, siempre y cuando sean globales.




```
for each Trip.Attraction
   unique TripDate.Year(), AttractionName
      &tripYear = TripDate.Year()
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //&tripYear, AttractionName, &qty
endfor
```

2023 Eiffel Tower 1

TripId*	TripDate	TripDescription
1	01/01/2023	...
2	04/04/2023 2024	...
3	01/01/2023	...
4	05/05/2023	...

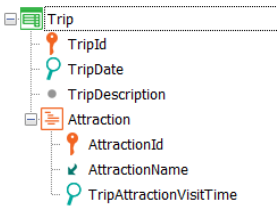
TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	6	180
4	6	240
3	8	60

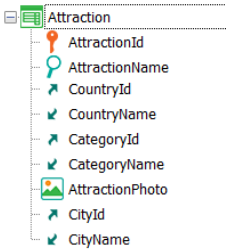
AttractionId*	AttractionName	...
3	Eiffel Tower	...
1	Louvre Museum	...
8	Louvre Museum	....
6	Matisse Museum	....

Por ejemplo, imaginemos que queremos contar los trips por nombre de atracción, pero de acuerdo al año de la excursión. Es decir, para cada nombre de atracción, por año, en cuántos trips está.

Nos veremos tentados a escribir esta cláusula unique, de modo tal que si cambiamos por ejemplo el año del trip 2, por este otro, entonces la salida tendría que ser la siguiente:

AttractionName Eiffel Tower solo está en un trip, así que se lista su año y el count dará 1.






```
for each Trip.Attraction
   unique TripDate.Year(), AttractionName
      &tripYear = TripDate.Year()
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //&tripYear, AttractionName, &qty
endfor
```

2023 Eiffel Tower 1

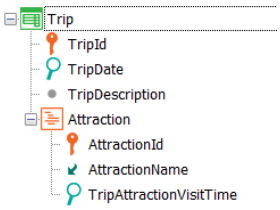
2023 Louvre Museum 2

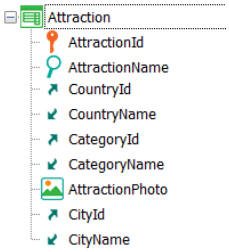
TripId*	TripDate	TripDescription
1	01/01/2023	...
2	04/04/2023 2024	...
3	01/01/2023	...
4	05/05/2023	...

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	6	180
4	6	240
3	8	60

AttractionId*	AttractionName	...
3	Eiffel Tower	...
1	Louvre Museum	...
8	Louvre Museum	....
6	Matisse Museum	....

Luego viene Louvre Museum que está en 3 trips. El año del 1 y del 3 coinciden, por lo que en la salida se verá esto.






```
for each Trip.Attraction
   unique TripDate.Year(), AttractionName
      &tripYear = TripDate.Year()
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //&tripYear, AttractionName, &qty
endfor
```

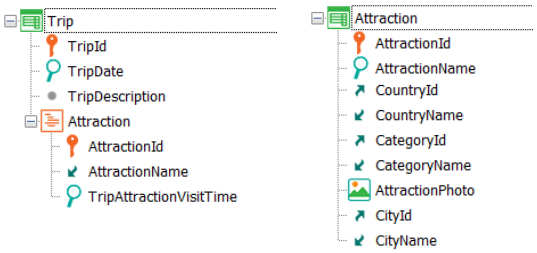
2023	Eiffel Tower	1
2023	Louvre Museum	2
2024	Louvre Museum	1

TripId*	TripDate	TripDescription
1	01/01/2023	...
2	04/04/2023 2024	...
3	01/01/2023	...
4	05/05/2023	...

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	6	180
4	6	240
3	8	60

AttractionId*	AttractionName	...
3	Eiffel Tower	...
1	Louvre Museum	...
8	Louvre Museum	....
6	Matisse Museum	....

Mientras que como para el trip 2 el año es otro, entonces se mostrará en la salida esto.




```
for each Trip.Attraction
   unique TripDate.Year(), AttractionName
      &tripYear = TripDate.Year()
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //&tripYear, AttractionName, &qty
endfor
```

TripId*	TripDate	TripDescription
1	01/01/2023	...
2	04/04/2023 2024	...
3	01/01/2023	...
4	05/05/2023	...

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	6	180
4	6	240
3	8	60

2023 Eiffel Tower 1

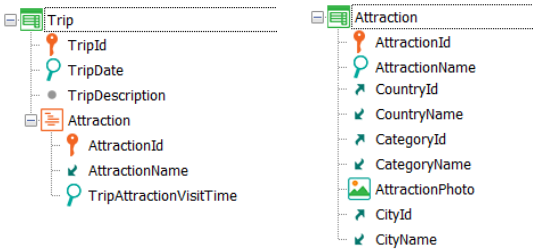
2023 Louvre Museum 2

2024 Louvre Museum 1

2023 Matisse Museum 2

AttractionId*	AttractionName	...
3	Eiffel Tower	...
1	Louvre Museum	...
8	Louvre Museum	....
6	Matisse Museum	....

Luego pasamos a la última AttractionName, que está en dos trips, el 3 y el 4, que son del mismo año, por lo que en la salida veremos.



TripId*	TripDate	TripDescription
1	01/01/2023	...
2	04/04/2023 2024	...
3	01/01/2023	...
4	05/05/2023	...

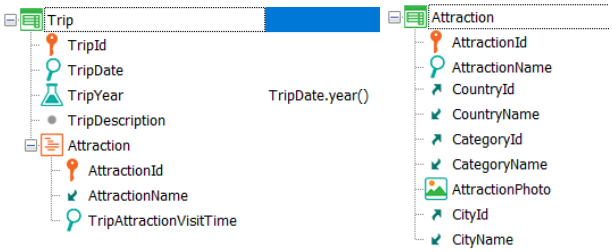
TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	6	180
4	6	240
3	8	60

```
for each Trip.Attraction
   unique TripDate.Year(), AttractionName
      &tripYear = TripDate.Year()
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //&tripYear, AttractionName, &qty
endfor
```

```
For each BaseTrn1, ... , BaseTrnn
    skip expression1 count expression2
    order att1, att2, ... , attn [when condition]
    order att1, att2, ... , attn [when condition]
    order none [when condition]
    unique att1, att2, ... , attn
    using DataSelector ( parm1, parm2, ... , parmn )
    where condition [when condition]
    where condition [when condition]
    where att IN DataSelector ( parm1, parm2, ... , parmn )
    blocking n
        main_code
    when duplicate
        when_duplicate_code
    when none
        when_none_code
endfor
```

El problema es que no se nos permitirá utilizar una expresión en la cláusula unique. Solamente podemos colocar atributos, como queda claro en la sintaxis.

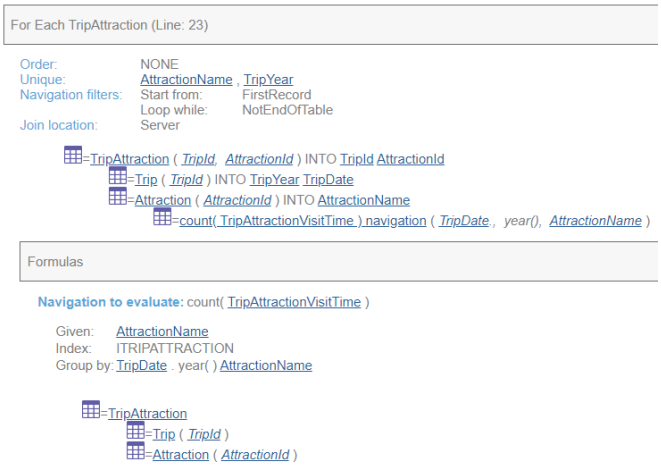
Pero esos atributos bien pueden ser atributos fórmula.



TripId*	TripDate	TripDescription
1	01/01/2023	...
2	04/04/2023 2024	...
3	01/01/2023	...
4	05/05/2023	...

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	6	180
4	6	240
3	8	60

```
for each Trip.Attraction
   unique       TripYear , AttractionName
      &tripYear = TripDate.Year()
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //&tripYear, AttractionName, &qty
endfor
```



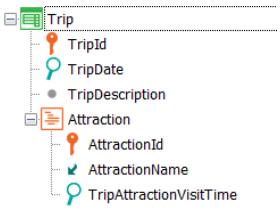
Por lo que en nuestro ejemplo si tuviéramos un atributo TripYear, fórmula, y la utilizamos en la cláusula unique todo funcionará como esperábamos, como nos muestra el listado de navegación.

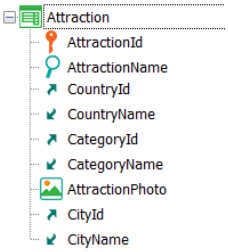
Restrictions

```
For each   BaseTrn1, … , BaseTrnn
    skip expression1 count expression2
    order  att1, att2, … , attn [when condition]
    order att1, att2, … , attn [when condition]
    order none [when condition]
    unique att1, att2, … , attn
    using DataSelector ( parm1, parm2, … , parmn )
    where  condition [when condition]
    where  condition [when condition]
    where  att  IN DataSelector ( parm1, parm2, … , parmn )
    blocking n
            main_code
    when duplicate
            when_duplicate_code
    when none
            when_none_code
endfor
```

Vimos como restricción, entonces, que solo pueden utilizarse atributos (que pueden ser fórmulas) pero no expresiones.

Otra restricción: solo pueden incluirse en el código principal o cuerpo del for each atributos que tengan valores únicos para aquellos de la cláusula unique.






```
for each Trip.Attraction
   unique AttractionId
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //AttractionName, &qty
endfor
```

Louvre Museum 2

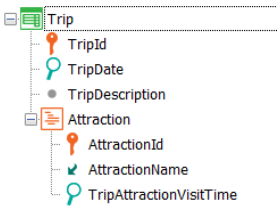
Louvre Museum 1

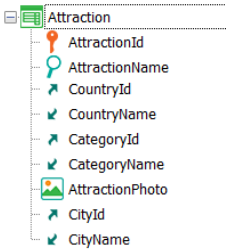
TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	6	180
4	6	240
3	8	60

AttractionId*	AttractionName	...
3	Eiffel Tower	...
1	Louvre Museum	...
8	Louvre Museum	....
6	Matisse Museum	....

En este ejemplo en que estamos pidiendo valores únicos de tripattraction de acuerdo a AttractionId, vemos, por ejemplo, que estos dos registros se procesarán una vez y se mostrará Louvre Museum junto con 2 como resultado del count. Y por otro lado este otro registro se procesará solo, mostrando Louvre Museum y 1 en la salida.

En el printblock pudo colocarse AttractionName porque para cada AttractionId de la cláusula unique es único su valor. También podríamos colocar CountryName, CityName, CategoryName, es decir atributos únicos para AttractionId. Pero si hubiéramos colocado TripAttractionVisitTime o TripDate, nos arrojará un error.






```
for each Trip.Attraction
   unique AttractionName
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //AttractionName, &qty, AttractionId
endfor
```

Louvre Museum 3

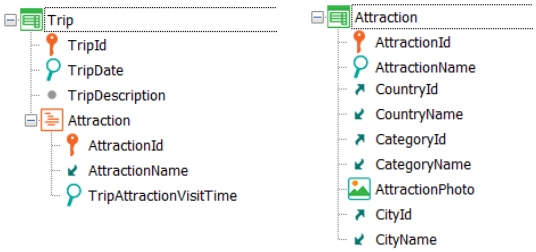
TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	6	180
4	6	240
3	8	60

AttractionId*	AttractionName	...
3	Eiffel Tower	...
1	Louvre Museum	...
8	Louvre Museum	....
6	Matisse Museum	....

Hay casos más sutiles que pueden llevarnos a engaño. Por ejemplo, ¿qué pasaría si en lugar de AttractionId colocáramos AttractionName en la unique?

Con este código no habrá problema, pero mostrará algo diferente que el anterior si hay atracciones con nombre repetido, como en este caso.
Porque agrupará estos 3 registros y mostrará 3 como resultado de la cuenta.

¿Qué pasaría si por distracción colocáramos AttractionId en el printblock?
Tendremos el mismo error que si colocáramos TripId o TripDate.
Es que de un AttractionName no se obtiene un único AttractionId.

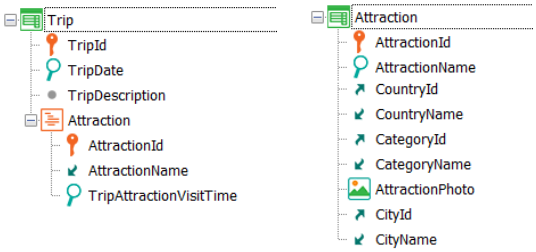



```
for each Trip.Attraction
   order AttractionId
   unique AttractionName
   Where TripDate > &today
      &qty = Count(TripAttractionVisitTime, TripDate>&today)
      print AttractionInfo //AttractionName, &qty
endfor
```

```
For each BaseTrn1, ... , BaseTrnn
    skip expression1 count expression2
    order att1, att2, ... , attn [when condition]
    order att1, att2, ... , attn [when condition]
    order none [when condition]
    unique att1, att2, ... , attn
    using DataSelector ( parm1, parm2, ... , parmn )
    where condition [when condition]
    where condition [when condition]
    where att IN DataSelector ( parm1, parm2, ... , parmn )
    blocking n
            main_code
    when duplicate
            when_duplicate_code
    when none
            when_none_code
endfor
```

La restricción solo aplica al cuerpo del for each, no a las otras cláusulas. En particular, no aplica a los filtros. Es decir, aplica a lo que ocurre luego de que los registros fueron ordenados y filtrados.

Así, podríamos querer por ejemplo listar cada nombre de atracción que está en trips posteriores a la fecha de hoy, junto con al cantidad de trips en los que se encuentra. Y con la consulta ordenada por AttractionId.




```
for each Trip.Attraction
   order AttractionId
   unique AttractionName
   Where TripDate > &today
      &qty = Count(TripAttractionVisitTime, TripDate>&today)
      print AttractionInfo //AttractionName, &qty
endfor
```

&today: 03/03/2023

TripId*	TripDate	TripDescription
1	01/01/2023	...
2	04/04/2023	...
3	01/01/2023	...
4	05/05/2023	...

Louvre Museum 2 Louvre Museum 4

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	8	60
4	8	240

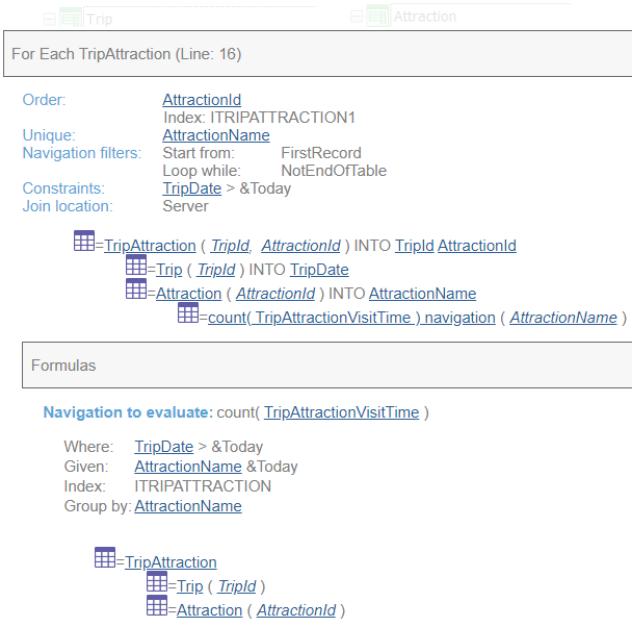
AttractionId*	AttractionName	...
1	Louvre Museum	...
3	Eiffel Tower	...
8	Louvre Museum	....

Si los datos son estos y la fecha de la variable &today es esta, entonces pensemos cuál deberá ser el resultado de la consulta.

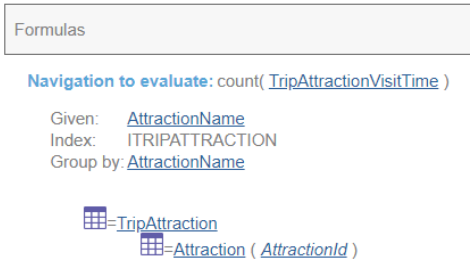
Se ordena la recorrida de la tabla TripAttraction por AttractionId, pero además se consideran una sola vez los registros que comparten el mismo AttractionName. En este caso, para el primer registro serán estos cuatro. Pero de esos, ¿cuántos pasarán el filtro? El primero no lo pasa, el segundo sí, el tercero no, y el cuarto sí.

Si la fórmula Count incluye también la misma condición, entonces en la salida se mostrará Louvre Museum, y 2.
Si en cambio no hubiéramos agregado la misma condición de filtro para la fórmula, saldrá Louvre Museum y 4.

Luego vamos al siguiente registro para procesar en el For each. Es el único que nos queda, cuya fecha no cumple con el filtro, por lo que no se procesa nada más. El resultado final será este (dependiendo de si se agregó o no la condición de filtro a la fórmula).




```
for each Trip.Attraction
   order AttractionId
   unique AttractionName
   Where TripDate > &today
      &qty = Count(TripAttractionVisitTime, ~~TripDate>&today~~)
      print AttractionInfo //AttractionName, &qty
endfor
```



De hecho, si observamos el listado de navegación, vemos claramente que la fórmula se calcula como deseamos. Observemos el Group by y cómo en el Where se muestra el filtro sobre los registros a ser contados.

Si eliminamos la condición del Count, entonces el listado de navegación informará esto otro para la fórmula.

Restrictions

For each *BaseTrn$_1$, … , BaseTrn$_n$*

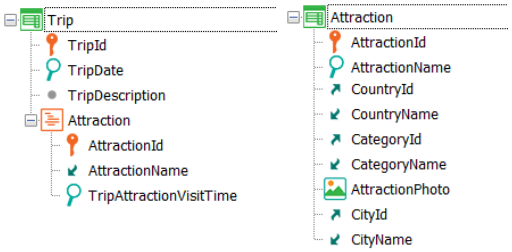
skip *expression$_1$* **count** *expression$_2$*
order *att$_1$, att$_2$, … , att$_n$* [**when** *condition*]
order *att$_1$, att$_2$, … , att$_n$* [**when** *condition*]
order none [**when** *condition*]
unique *att$_1$, att$_2$, … , att$_n$*
using *DataSelector* (*parm$_1$, parm$_2$, … , parm$_n$*)
where *condition* [**when** *condition*]
where *condition* [**when** *condition*]
where *att* **IN** *DataSelector* (*parm$_1$, parm$_2$, … , parm$_n$*)
blocking *n*
main_code
when duplicate
when_duplicate_code
when none
when_none_code

endfor

For each **≠ Base table**

endfor

Por último, mencionemos la última restricción: si utilizamos cláusula unique solo tiene sentido anidarle otro for each si éste no tiene la misma tabla base. Es decir, no podemos utilizar unique en cortes de control, como uno en principio podría pretender.




```
For each Trip.Attraction
   order AttractionName
      print AttractionInfo //AttractionName
      For each Trip.Attraction
         print TripAttractionInfo //TripDate TripAttractionVisitTime
      endfor
endfor
```

Eiffel Tower

01/01/2023 120

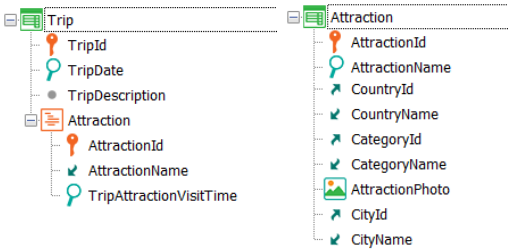
TripId*	TripDate	TripDescription
1	01/01/2023	...
2	04/04/2023	...
3	01/01/2023	...
4	05/05/2023	...

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	8	60
4	8	240

AttractionId*	AttractionName	...
3	Eiffel Tower	...
1	Louvre Museum	...
8	Louvre Museum	....

Así, por ejemplo, si queremos un listado de las atracciones que están en trips, con la duración de la visita a la atracción en cada uno de esos trips, no tendremos otra alternativa que implementarlo como el clásico corte de control.

Recorremos TripAttraction ordenada por AttractionName y para el primer registro de TripAttraction con el primer AttractionName imprimimos el nombre de atracción y luego iteramos por los registros con el mismo AttractionName, aquí este solo. Imprimimos la fecha del trip y el tiempo de visita, y pasamos al siguiente grupo…




```
For each Trip.Attraction
   order AttractionName
      print AttractionInfo //AttractionName
      For each Trip.Attraction
         print TripAttractionInfo //TripDate TripAttractionVisitTime
      endfor
endfor
```

Eiffel Tower

01/01/2023 120

Louvre Museum

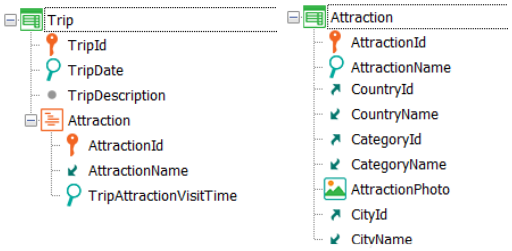
01/01/2023 180
04/04/2023 200
01/01/2023 60
05/05/2023 240

TripId*	TripDate	TripDescription
1	01/01/2023	...
2	04/04/2023	...
3	01/01/2023	...
4	05/05/2023	...

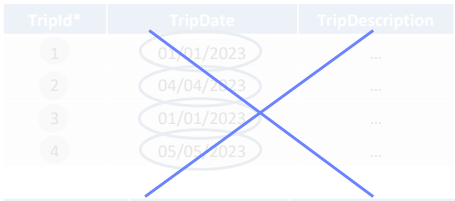
TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	8	60
4	8	240

AttractionId*	AttractionName	...
3	Eiffel Tower	...
1	Louvre Museum	...
8	Louvre Museum	....

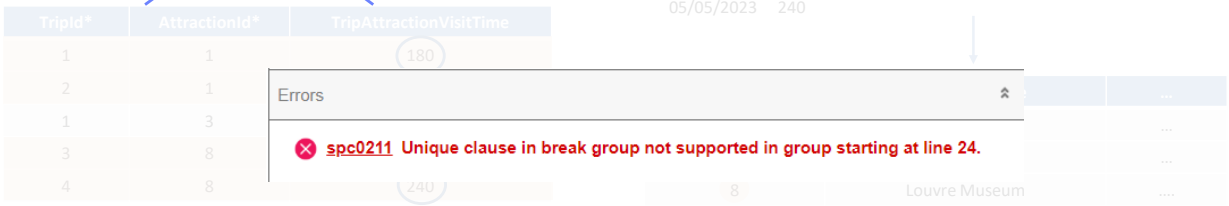
…que es el que corresponde a estos registros de igual nombre en Attraction. Imprimimos el nombre, y otra vez, iteramos con el for each anidado mientras no cambie el AttractionName. Y así tenemos en la salida…



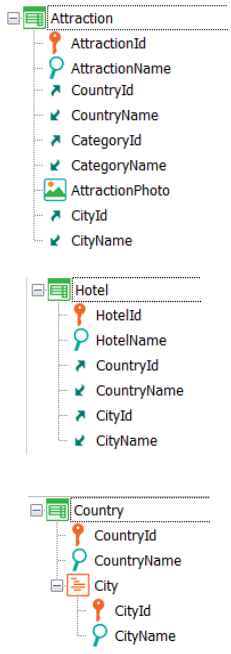

```
For each Trip.Attraction
   order AttractionName
      print AttractionInfo //AttractionName
      For each Trip.Attraction
         print TripAttractionInfo //TripDate TripAttractionVisitTime
      endfor
endfor
```



```
For each Trip.Attraction
   unique AttractionName
      print AttractionInfo //AttractionName
      For each Trip.Attraction
         print TripAttractionInfo //TripDate TripAttractionVisitTime
      endfor
endfor
```



Ahora, si esto mismo quisiéramos implementarlo con la cláusula unique GeneXus no nos lo permitirá. El listado de navegación nos mostrará este error.




```
For each Attraction
   order CountryName
   unique CountryName
      &qty = count(AttractionName)
      print MainInfo //CountryName, &qty
      for each Hotel
         print HotelInfo //HotelName
      endfor
endfor
```

Brazil **1**

France **3**

Oh la la

Liberte

Cinque Terre **1**

Imperio

AttractionId*	AttractionName	CountryId	...
2	Christ	5	...
3	Eiffel Tower	2	...
1	Louvre Museum	2	...
8	Matisse Museum	2	....
4	Cinque Terre	15	...

CountryId*	CountryName
5	Brazil
2	France
15	Italy

HotelId*	HotelName	CountryId	...
1	Oh la la	2	...
3	Imperio	15	....
4	Liberte	2	...

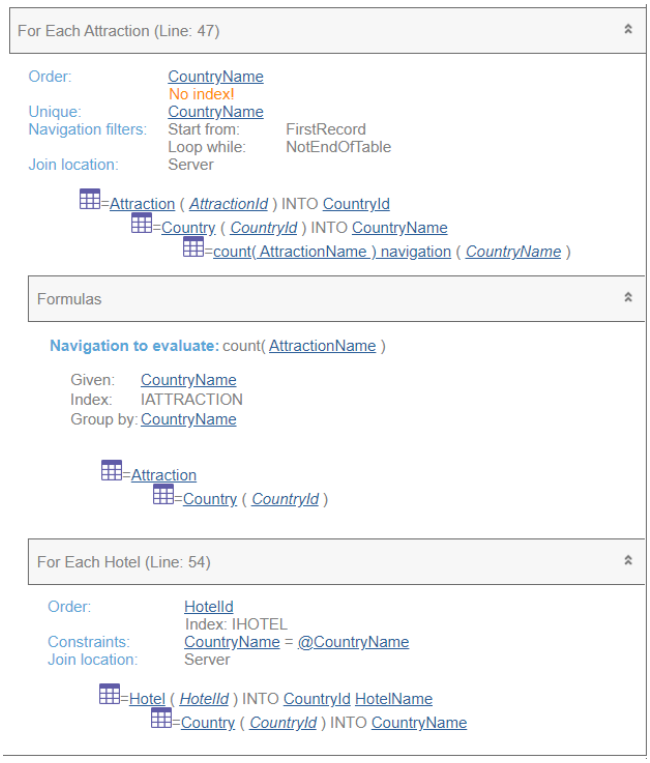
Pero sí podemos anidar un for each que navegue otra tabla.

Por ejemplo veamos este caso en el que estamos recorriendo la tabla de atracciones ordenando por CountryName, de la extendida, y pidiendo procesar una única vez a todos los registros que repitan el valor de CountryName.

Para ellos, queremos contar las atracciones del mismo país, imprimir ese país con el número de atracciones, y navegar la tabla Hotel imprimiendo los nombres de hoteles del mismo país.

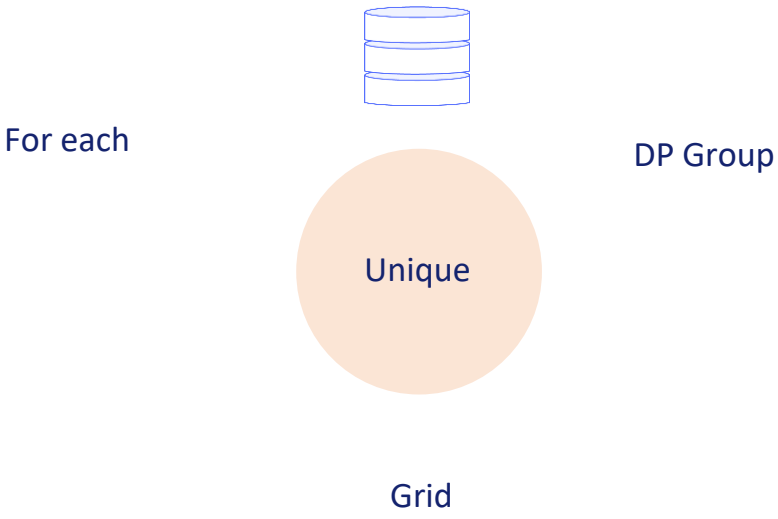
Así, con estos datos, veremos en la salida lo siguiente. Aquí tenemos la tabla Attraction ordenada por CountryName.
Hay solo un registro para el país del primer registro: se lista entonces su país y 1 para la cuenta. Como no hay ningún hotel de Brazil, se pasará al siguiente registro en Attraction, que será el 3, correspondiente a France. Hay 3 registros con el país France, así que en la salida se verá… Y luego se ejecuta el for each anidado, que imprimirá los hoteles de France. Por lo tanto se verá en la salida…

Y por último tendremos…




```
For each Attraction
   order CountryName
   unique CountryName
      &qty = count(AttractionName)
      print MainInfo //CountryName, &qty
      for each Hotel
         print HotelInfo //HotelName
      endfor
endfor
```

Si observamos el listado de navegación nos quedará claro que se está implementando lo que queríamos. Observemos la Constraint del for each anidado por CountryName.



Por último, para terminar, volver a mencionar que si bien nos concentramos en la cláusula unique para el comando For each, su lógica vale para todas las demás formas de consulta.

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