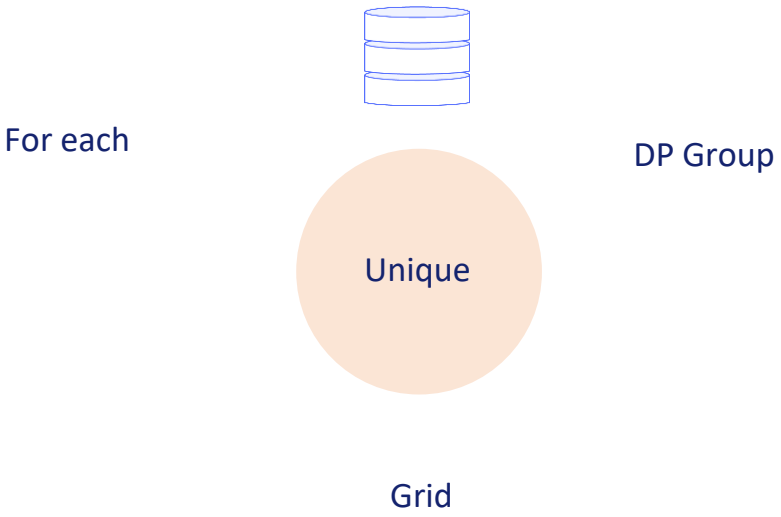
Lógica de consulta à base de dados com GeneXus

For each: cláusula Unique

GeneXus™



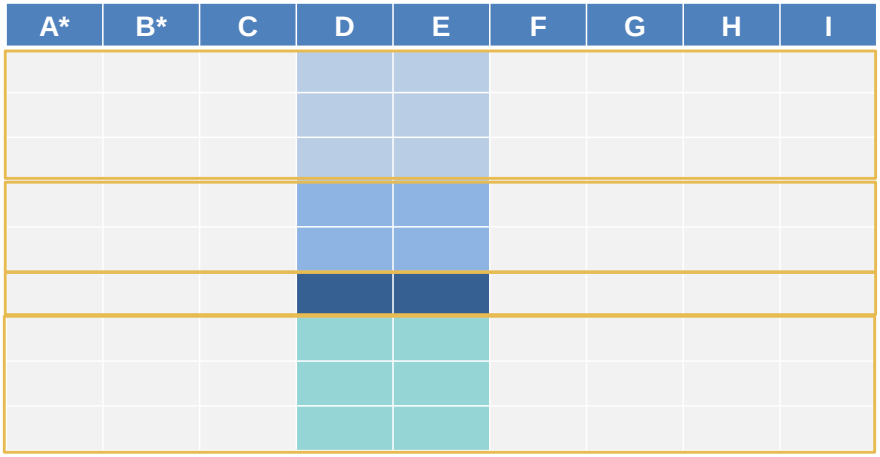
Veremos com algum detalhe a cláusula unique aplicada ao comando For each, mas sabendo que é possível ser utilizada analogamente em grupos de data providers e grids com tabela base.

```
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
```

Aqui a vemos entre as outras cláusulas do For each.

Navigation group

unique D, E → **Print info //** D, E



Quando para muitos registros se repete o valor de um conjunto de atributos, neste exemplo, D e E, podemos utilizar a cláusula unique para poder trabalhar com um de todos os registros para os quais ocorre essa repetição (como se fosse um representante do grupo); por exemplo, imprimindo os valores desses atributos (já que serão os mesmos para todos os registros do grupo), e depois passar para o próximo grupo, para fazer a mesma coisa. E assim sucessivamente até esgotar todos.

Navigation group

unique D, E → **Print info //** D, E

A*	B*	C	D	E	F	G

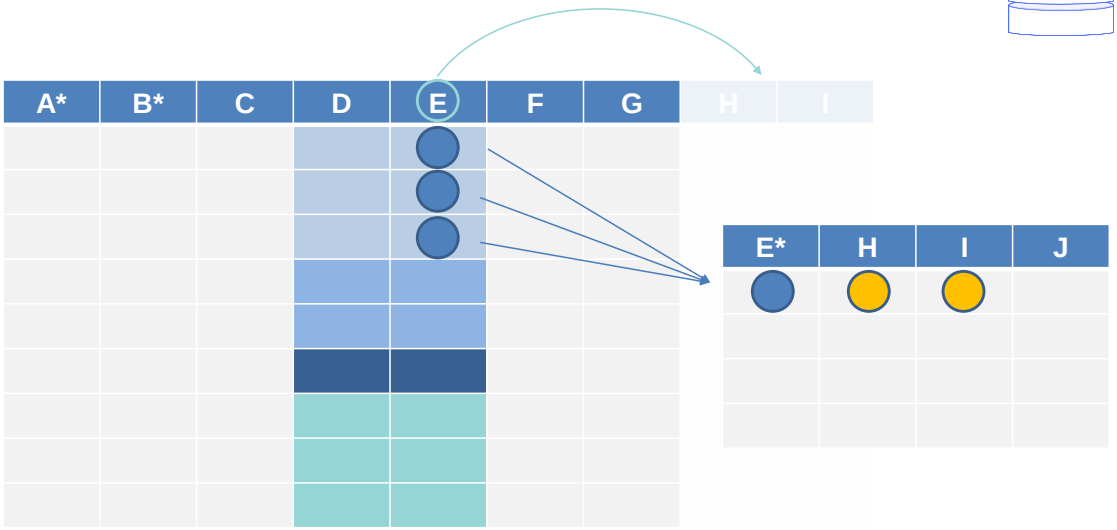
```
For each BaseTrn1, ... , BaseTrnn
    skip expression1 count expression2
    order att1, att2, ... , attn [when condition]
    order att1, att2, ... , attn [when condition]
    order none [when condition]
    unique att1, att2, ... , attn
    using DataSelector ( parm1, parm2, ... , parmn )
    where 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 isto faça sentido, dentro do código que será executado para cada grupo, poderão aparecer apenas atributos cujo valor seja único para todos e cada um dos registros do grupo.

Os atributos ali presentes não precisam fazer parte da mesma tabela. Podem estar na estendida.

Navigation group

unique D, E → **Print info //** D, E



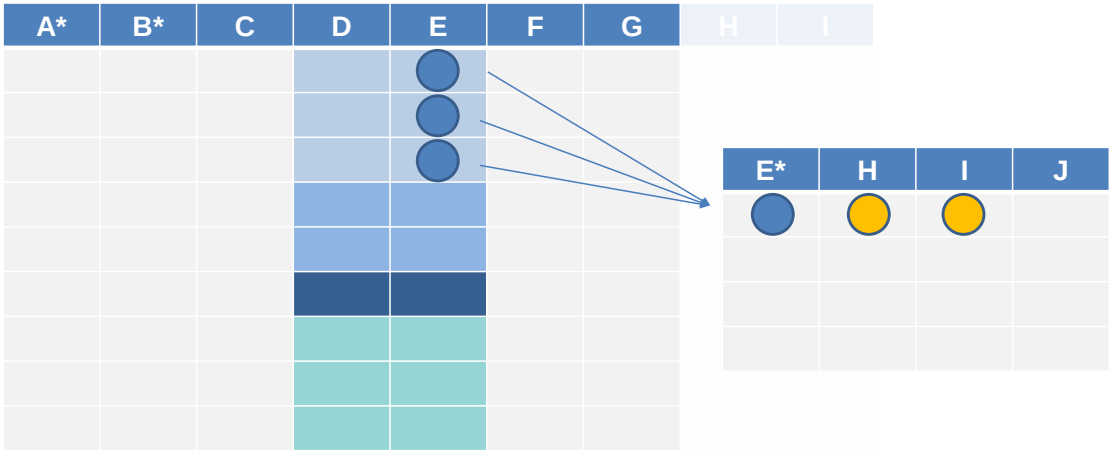
Vamos supor que esta seja uma representação gráfica da tabela estendida e não uma tabela física.

Por exemplo, suponhamos que E seja uma chave estrangeira que determina H e I. Ou seja, que existe esta tabela física.

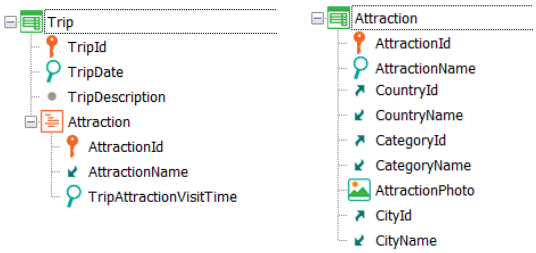
Isto significa que para todos os registros do grupo, como os valores de E são os mesmos, os valores de H e de I também serão os mesmos.

Navigation group

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



Isto significa que também podemos utilizar os atributos H e I dentro do código que será executado para o grupo, pois também serão únicos para esse 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	

Vamos ver exemplos.

Temos as excursões que podem ser feitas para diferentes atrações turísticas que têm atribuído um tempo de duração de cada visita.

Queremos obter uma lista das atrações turísticas que participam de excursões. Vamos imaginar que estes são os dados atuais das tabelas. Vamos querer listar unicamente estas atrações.

Uma primeira alternativa que poderia nos ocorrer é implementar um corte de controle que navegue TripAttraction e agrupe por AttractionId. Desta forma, teremos a certeza de listar apenas atrações que efetivamente estão em excursões e, além disso, por colocar o comando print após o for each aninhado (embora fosse o mesmo colocá-lo antes), sabemos que estaremos para cada grupo imprimindo unicamente o AttractionName que se repete.

O mesmo podemos resolver de modo muito mais simples utilizando a cláusula Unique. É seu caso de uso mais evidente.

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

```
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	

Se observamos a lista de navegação do For each com Unique vemos que como existe um índice por AttractionId em TripAttraction (por ser chave estrangeira) escolherá ordenar por esse atributo.

Portanto, começa pelo primeiro grupo onde se repete o valor de AttractionId e é impresso na saída seu AttractionName (que é único para todos os registros do grupo): Louvre Museum.

For Each TripAttraction (Line: 23)

Order: AttractionId
Index: ITRIPATTRACTION1
Unique: AttractionId
Navigation filters: Start from: FirstRecor
Loop while: NotEndOf
Join location: Server

=TripAttraction (TripId, AttractionId) I
=Attraction (AttractionId) INTO

Attraction

Louvre Museum

Eiffel Tower

Matisse Museum

Rifugio Nuvolau

Cinque Terre

```
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	

Então o próximo grupo é de um único registro: é impressa Eiffel Tower.
Depois a atração de id 6, que é Matisse Museum. Depois a 10 e por último a 12.

For Each TripAttraction (Line: 23)

Order: AttractionId
Index: ITRIPATTRACTION1
Unique: AttractionId
Navigation filters: Start from: FirstRecor
Loop while: NotEndOf
Join location: Server

=TripAttraction (TripId, AttractionId) I
=Attraction (AttractionId) INTO

Attrac
Louvre
Eiffel T
Matiss
Rifugio
Cinque

Attraction
Cinque Terre
Eiffel Tower
Louvre Museum
Matisse Museum
Rifugio Nuvolau

```
traction
ionId
rip.Attraction

actionInfo //AttractionName


traction
tionId
actionInfo //AttractionName
```

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	

Se estivéssemos interessados em que essas atrações saíssem ordenadas por nome de atração…

Trip | Attraction

For Each TripAttraction (Line: 23)

Order: AttractionName
No index!
Unique: AttractionId
Navigation filters: Start from: FirstRecord
Loop while: NotEndOfTable
Join location: Server

=TripAttraction (*TripId*, *AttractionId*) INTO AttractionId
=Attraction (*AttractionId*) INTO AttractionName

Attraction

Cinque Terre
Eiffel Tower
Louvre Museum
Matisse Museum
Rifugio Nuvolau

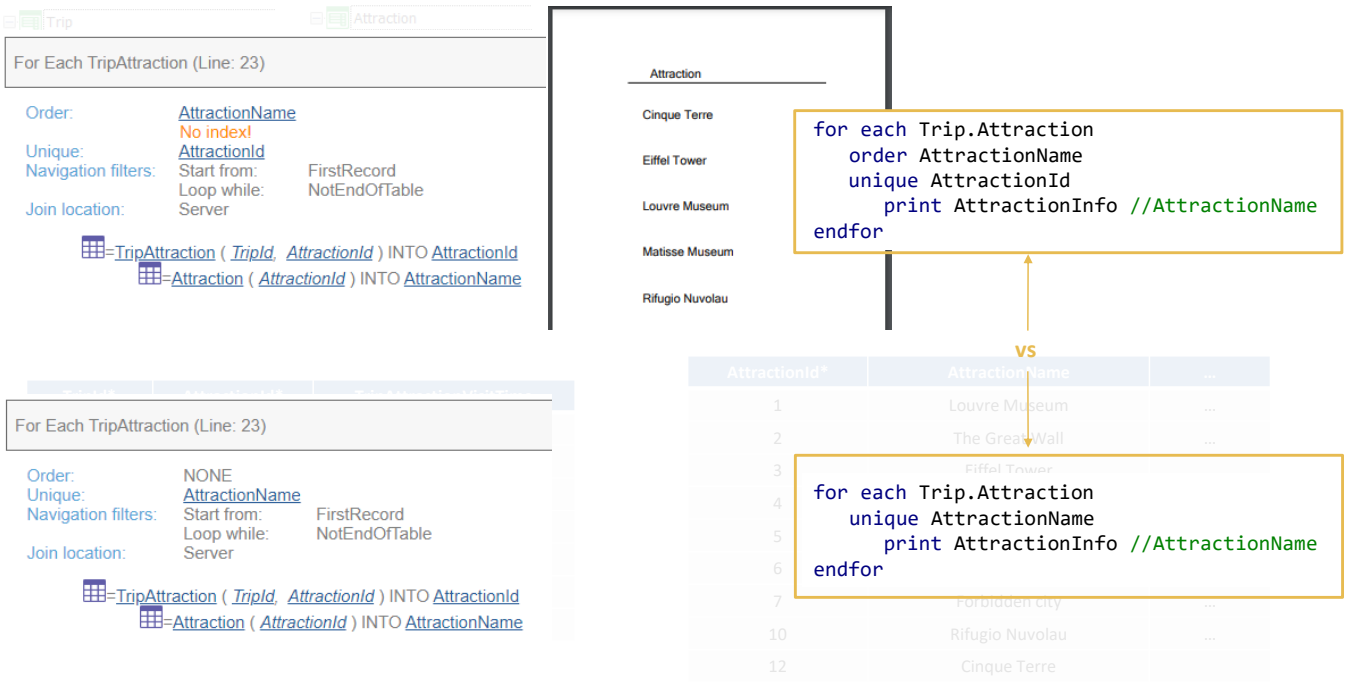
```
for each Trip.Attraction
   order AttractionName
   unique AttractionId
      print AttractionInfo //AttractionName
endfor
```

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	

...poderíamos adicionar a cláusula order.

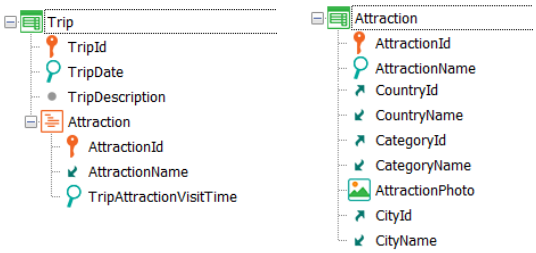
E assim vemos a lista de navegação. O valor único que se busca continua sendo AttractionId, mas o resultado da consulta será ordenado por AttractionName.



Em vez de fazer isto, poderíamos utilizar na cláusula unique o atributo AttractionName diretamente?

Sim, mas vamos observar duas coisas. Por um lado, fazer isto não garante que a lista seja também ordenada por AttractionName. Observemos que a lista de navegação está indicando Order NONE. Ocorre que GeneXus não tem conhecimento da existência de um índice por AttractionName. Portanto, ainda deveríamos ordenar por AttractionName se é isso que queremos.

Mas por outro lado, vamos pensar no que aconteceria se tivéssemos na base de dados duas atrações com o mesmo nome.




```
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

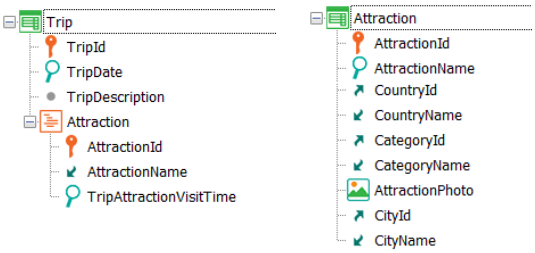
AttractionId*	AttractionName	...
1	Louvre Museum	...
2	The Great Wall	...
3	Eiffel Tower	...
4	Christ the Redemmer	...
5	Smithsonian Institute	...
6	Matisse Museum	....
7	Forbidden city	...
8	Louvre Museum	
10	Rifugio Nuvolau	...
12	Cinque Terre	

Por exemplo, a 1 e a 8. Se não tivermos um índice unique por AttractionName, isto será permitido. E observemos que temos a atração 1 em 2 trips, e a 8 em 1.

Entre este For each e este outro a única diferença é a cláusula unique.

No primeiro caso, estes dois registros serão trabalhados em conjunto, e será listado Louvre Museum; e este será trabalhado por outro lado, em outro grupo, e será listado novamente Louvre Museum.

Por outro lado, no segundo caso, os três registros estarão dentro do mesmo grupo, e será listado apenas uma vez Louvre Museum, embora corresponda a duas atrações diferentes.



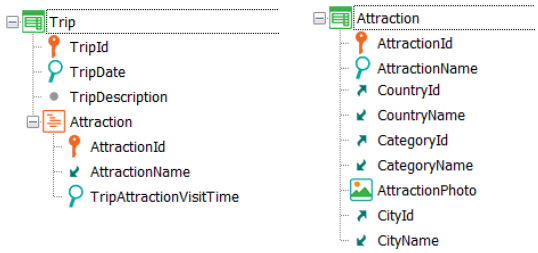

```
for each Trip.Attraction
   order AttractionName
   unique AttractionId
      print AttractionInfo //AttractionName
endfor

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	

Vemos isso muito claramente se imaginarmos os dados assim, com a tabela estendida como uma supertabela.




```
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	

Se a ordenarmos por AttractionName vemos com mais clareza... Observemos que não há problema algum em que o atributo da cláusula unique esteja na tabela estendida e não na base.

Resumindo, não será a mesma coisa pedir valores únicos para AttractionId que para AttractionName.

Se existe o índice unique, então sim, o resultado de ambos os for eachs será exatamente o mesmo, pois não ocorrerá que possa existir este registro 8.

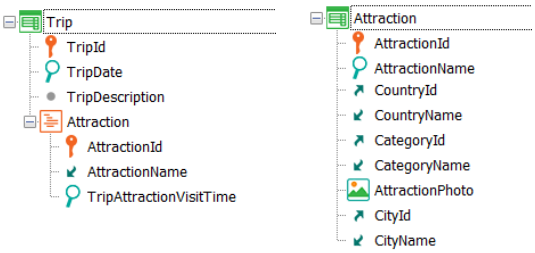
Navigation group

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

A*	B*	C	D	E	F	G	H	I

Além disso, podemos não apenas ficar com um dos registros que se repetem para fazer algo com a informação que não varia (como imprimi-la), mas também podemos executar fórmulas de agregação sobre os repetidos, que, por exemplo, contam eles. A fórmula deve navegar, é claro, a mesma tabela.

Assim, é tomado o primeiro grupo e é executada a count sobre seus registros (dará 3 neste caso). E é impresso D e E, informação que é única para esse grupo, e 3.
Em seguida, o próximo grupo, para o qual count dará 2.
Em seguida, o terceiro, para o qual dará 1. E por último o quarto, para o qual dará 3.

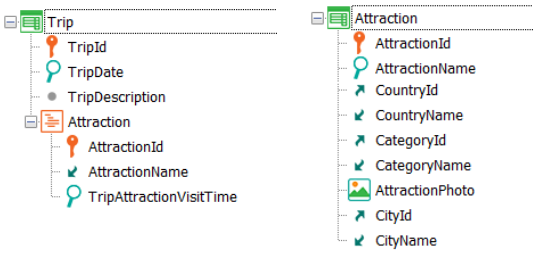



```
for each Trip.Attraction
   unique AttractionId
      &qty = Count(TripAttractionVisitTime)
      print AttractionInfo //AttractionName, &qty
endfor
```

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	

Neste exemplo, além de querer ficar com AttractionId não repetido para listar seu nome, queremos contar quantas vezes aparece repetido. Resumindo, em quantas 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

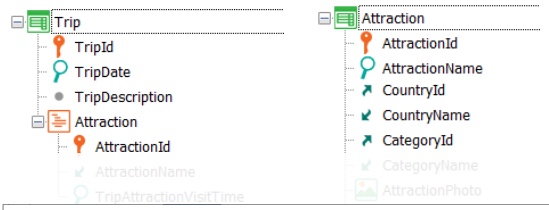
Navigation to evaluate: count(TripAttractionVisitTime)

Given: AttractionId
Index: ITRIPATTRACTION
Group by: AttractionId

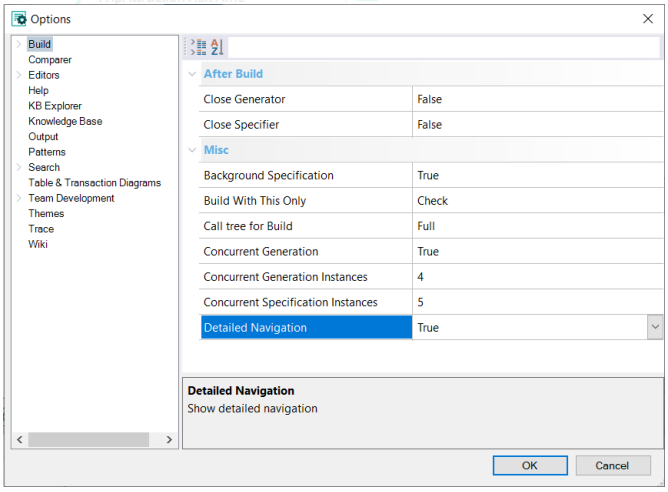
=TripAttraction (*AttractionId*)

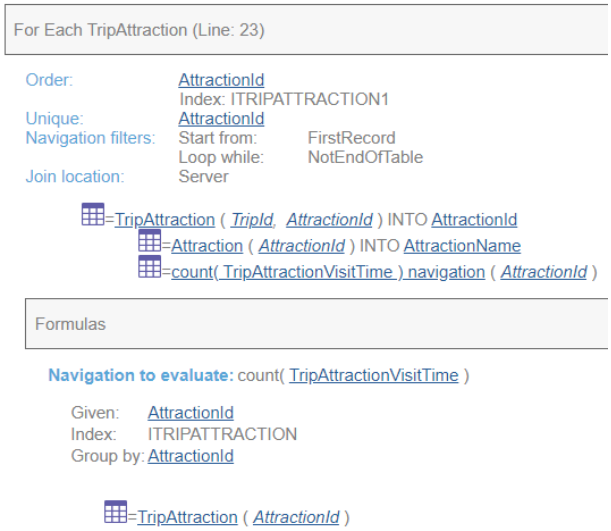
TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
5	8	60
2	10	120
2	12	90

A fórmula Count está utilizando o atributo secundário da tabela que se quer navegar, pelo que ao determinar isto, a fórmula Count terá um comportamento especial: irá agrupar pelo atributo de unique, como vemos na lista de navegação.




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





Lembremos que para que a lista nos mostre a navegação da fórmula devemos ativar a navegação detalhada... através 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*)

Então, para cada grupo de repetidos, serão contados os registros para aquele AttractionId dado, o de cada grupo. Assim, obtém-se o primeiro grupo com AttractionId repetido, são contados seus registros, aqueles que tenham o mesmo AttractionId, e são impressos na saída o nome de atração e essa quantidade.

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	

Em seguida, o próximo grupo, para o qual a count dá 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	

Em seguida, o próximo, que dá 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	

Então o próximo com o mesmo nome do primeiro, dá 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	

O próximo também dá 1, e o último também.

Em vez disso, se no 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 na cláusula unique, então o grupo correspondente ao 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*)

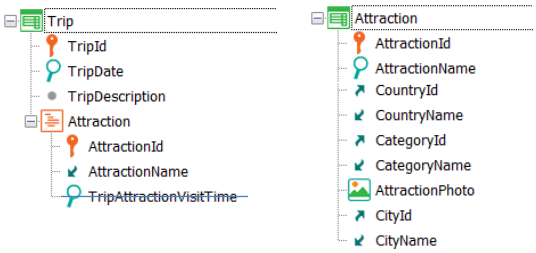
Formulas

Navigation to evaluate: count(TripAttractionVisitTime)

Given: AttractionName
Index: ITRIPATTRACTION
Group by: AttractionName

=TripAttraction
=Attraction (*AttractionId*)

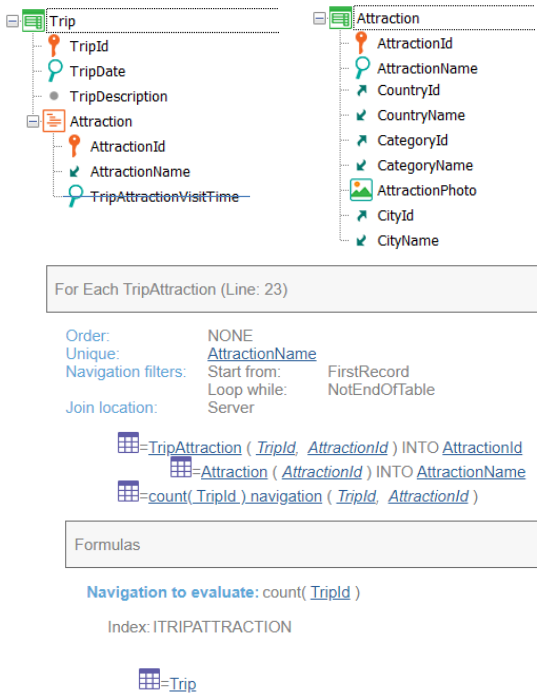
Na navegação vemos o Given e o Group by.



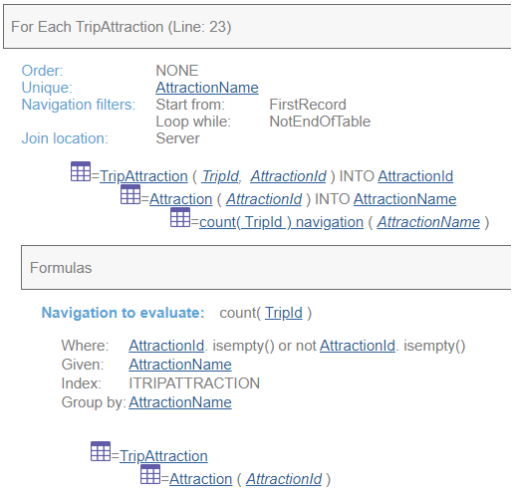

```
for each Trip.Attraction
   unique AttractionName
      &qty = Count(TripAttractionVisitTime)

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

Vejamos este caso em particular. Se na tabela que queremos navegar não há nenhum atributo secundário, então possivelmente tenhamos que fazer algo para que GeneXus entenda que é desejado navegar essa tabela para a fórmula.

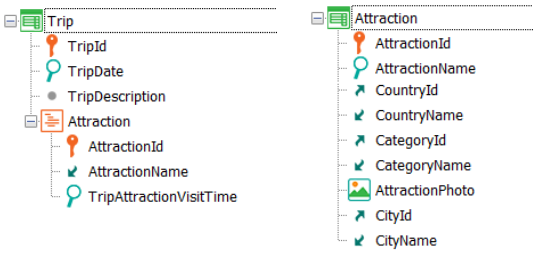



```
for each Trip.Attraction
   unique AttractionName
      &qty = Count(TripId), AttractionId.IsEmpty() or
                                 not AttractionId.IsEmpty())
      print AttractionInfo //AttractionName, &qty
endfor
```



Ou seja, se, por exemplo, colocamos para Count o atributo TripId e em unique deixamos AttractionName, pode ser que GeneXus não escolha a tabela TripAttraction para resolver a fórmula Count, mas Trip, e o resultado não será o desejado.
Vejamos que é informado que navegará a tabela Trip e contará todas as trips então, pois não há nenhuma condição informada para a fórmula.

Precisamos que escolha navegar TripAttraction para que faça o que queremos. Como não temos trn base para as fórmulas, podemos usar um truque: adicionar uma condição que seja sempre true e que contenha um atributo que faça determinar a tabela base que queremos. Por exemplo, esta condição que utiliza AttractionId e que será sempre true.
Observemos a lista de navegação indicando o que queremos. Agora sim está navegando TripAttraction e também está agrupando pelo AttractionName dado no for each, portanto, contando apenas as tripattractions do mesmo 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	...

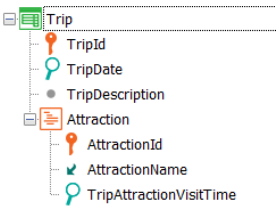
TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	6	180
4	6	240
3	8	60

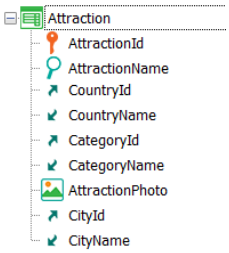
AttractionId*	AttractionName	...
3	Eiffel Tower	...
1	Louvre Museum	...
8	Louvre Museum	....
6	Matisse Museum	....

Vamos agora um pouco mais adiante. Sabemos que podemos especificar vários atributos na cláusula unique, e que eles não precisam pertencer à tabela base do for each, como neste exemplo.

Queremos contar a quantidade de trips que em uma mesma data incluem visita a um mesmo nome de atração. Ou seja, para o mesmo TripDate e AttractionName, quantos registros existem em TripAttraction.

Se estes são os dados das tabelas (mostramos apenas os registros relevantes), vemos que para Eiffel Tower haverá apenas um registro em TripAttraction: o da trip 1, que é nesta data.






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

01/01/2023 Eiffel Tower 1

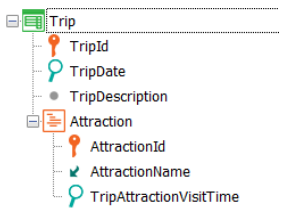
01/01/2023 Louvre Museum 2

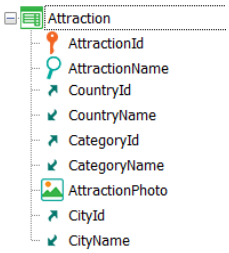
TripId*	TripDate	TripDescription
1	01/01/2023	...
2	04/04/2023	...
3	01/01/2023	...
4	05/05/2023	...

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	....

O Louvre Museum temos nestes 3 registros. Se formos olhar as datas, para a trip 1 e para a 3 são as mesmas, então na saída teremos…






```
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	...

01/01/2023 Eiffel Tower 1

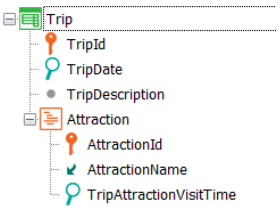
01/01/2023 Louvre Museum 2

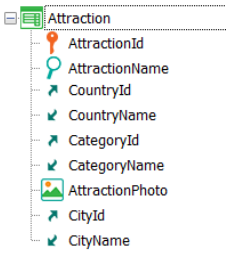
04/04/2023 Louvre Museum 1

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	....

E para a 2 é esta outra, então aparecerá na saída isto.






```
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

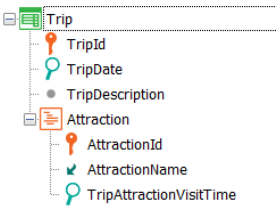
04/04/2023 Louvre Museum 1

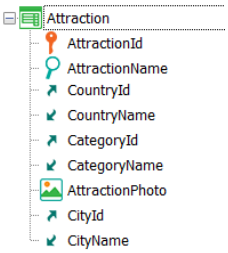
01/01/2023 Matisse Museum 1

05/05/2023 Matisse Museum 1

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

E por último para Matisse Museum: temos a trip 3 e a 4, que, por terem datas diferentes, vão dar origem a dois prints na saída.






```
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	...

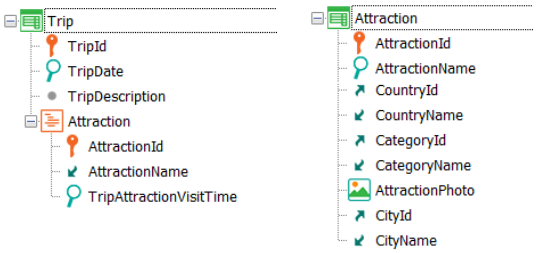
TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	6	180
4	6	240
3	8	60

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	....

Neste exemplo, nenhum dos dois atributos da cláusula unique pertence à tabela base do For each.

Também poderíamos utilizar fórmulas na cláusula unique, desde que sejam globais.




```
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	...

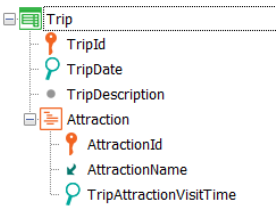
TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	6	180
4	6	240
3	8	60

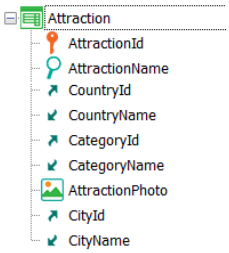
AttractionId*	AttractionName	...
3	Eiffel Tower	...
1	Louvre Museum	...
8	Louvre Museum	....
6	Matisse Museum	....

Por exemplo, vamos imaginar que queremos contar as trips por nome de atração, mas de acordo com o ano da excursão. Ou seja, para cada nome de atração, por ano, em quantas trips está.

Seremos tentados a escrever esta cláusula unique, de tal forma que se trocarmos, por exemplo, o ano da trip 2, por este outro, então a saída teria que ser a seguinte:

AttractionName Eiffel Tower está apenas em uma trip, então é listado seu ano e a count dará 1.






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

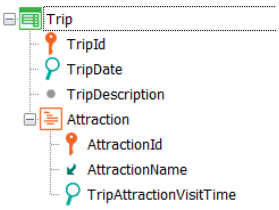
2023	Eiffel Tower	1
2023	Louvre Museum	2

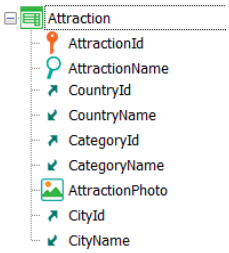
TripId*	TripDate	TripDescription
1	01/01/2023	...
2	04/04/~~2023~~ 2024	...
3	01/01/2023	...
4	05/05/2023	...

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	....

Depois vem Louvre Museum que está em 3 trips. O ano da 1 e da 3 coincidem, então na saída será visto isso.






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

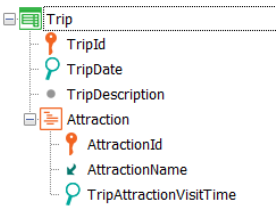
2023	Eiffel Tower	1
2023	Louvre Museum	2
2024	Louvre Museum	1

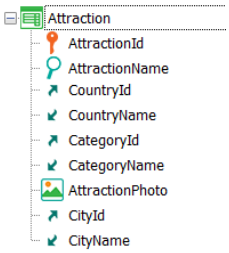
TripId*	TripDate	TripDescription
1	01/01/2023	...
2	04/04/2023 2024	...
3	01/01/2023	...
4	05/05/2023	...

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	....

Enquanto para a trip 2 o ano é outro, então será mostrado na saída isso.






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

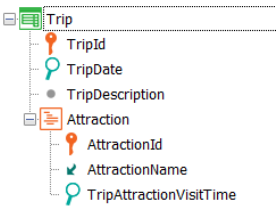
TripId*	TripDate	TripDescription
1	01/01/2023	...
2	04/04/~~2023~~ 2024	...
3	01/01/2023	...
4	05/05/2023	...

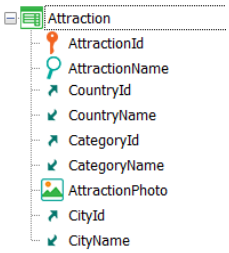
TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	6	180
4	6	240
3	8	60

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	....

Então passamos para o último AttractionName, que está em duas trips, a 3 e a 4, que são do mesmo ano, portanto na saída veremos.





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

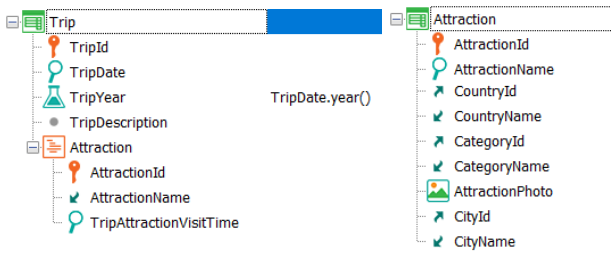
TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	6	180
4	6	240
3	8	60

```
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
```

O problema é que não poderemos utilizar uma expressão na cláusula unique. Só podemos colocar atributos, como fica claro na sintaxe.

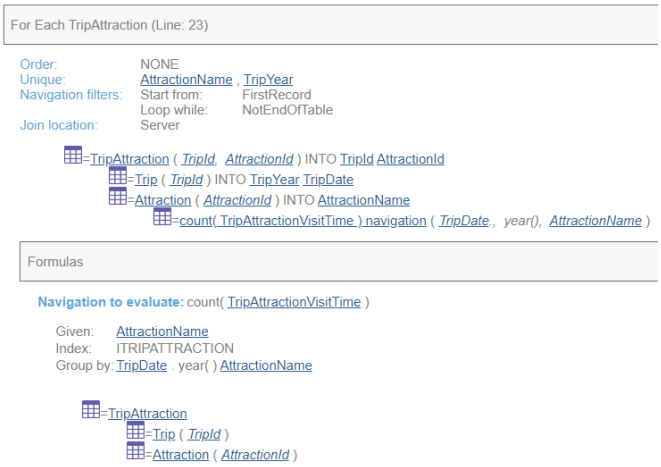
Mas esses atributos podem muito bem ser atributos fórmula.



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
```



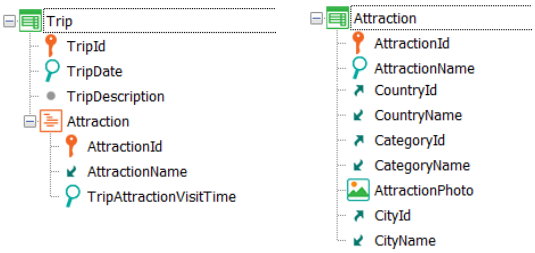
Portanto, em nosso exemplo, se tivéssemos um atributo TripYear, fórmula, e a utilizássemos na cláusula unique, tudo funcionará conforme o esperado, como nos mostra a lista de navegação.

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 restrição, então, que só podem ser utilizados atributos (que podem ser fórmulas) mas não expressões.

Outra restrição: só podem ser incluídos no código principal ou corpo do for each atributos que tenham valores únicos para aqueles da cláusula unique.




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

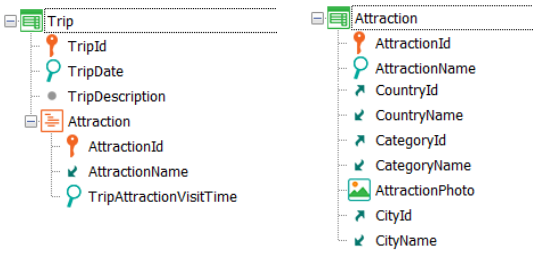
Louvre Museum	2
Louvre Museum	1

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	....

Neste exemplo onde estamos solicitando valores únicos de tripattraction de acordo com AttractionId, vemos, por exemplo, que estes dois registros serão processados uma vez e será exibido Louvre Museum junto com 2 como resultado da count. E por outro lado este outro registro será processado sozinho, mostrando Louvre Museum e 1 na saída.

No printblock foi possível ser colocado AttractionName porque para cada AttractionId da cláusula unique é único seu valor. Também poderíamos colocar CountryName, CityName, CategoryName, ou seja, atributos únicos para AttractionId. Mas se tivéssemos colocado TripAttractionVisitTime ou TripDate, nos lançaria um erro.




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

Louvre Museum 3

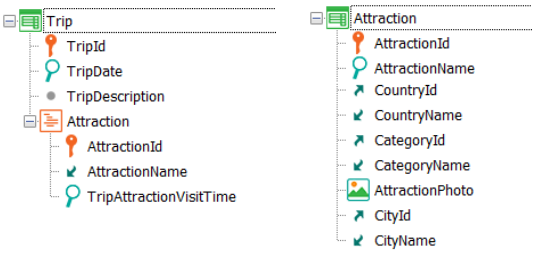
TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	6	180
4	6	240
3	8	60

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

Há casos mais sutis que podem nos levar ao engano. Por exemplo, o que aconteceria se em vez de AttractionId colocássemos AttractionName na unique?

Com este código não haverá problema, mas mostrará algo diferente do anterior se houver atrações com nome repetido, como neste caso.
Porque agrupará estes 3 registros e exibirá 3 como resultado da contagem.

O que aconteceria se, por distração, colocássemos AttractionId no printblock?
Teremos o mesmo erro que se colocássemos TripId ou TripDate.
Ocorre que, de um AttractionName não é obtido um único AttractionId.




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

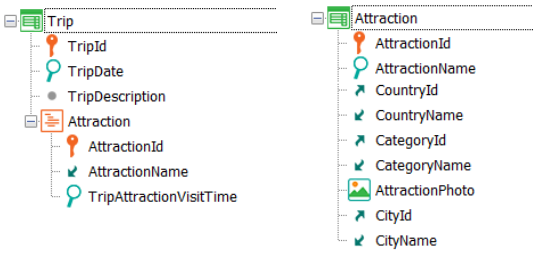
For each *BaseTrn$_1$, ... , BaseTrn$_n$*

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

A restrição aplica-se apenas ao corpo do for each, não às outras cláusulas. Em particular, não se aplica aos filtros. Ou seja, aplica-se ao que acontece após os registros serem ordenados e filtrados.

Assim poderíamos querer, por exemplo, listar cada nome de atração que está em trips posteriores à data de hoje, juntamente com a quantidade de trips em que ela se encontra. E com a 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	....

Se os dados forem estes e a data da variável &today for esta, então vamos pensar em qual deverá ser o resultado da consulta.

É ordenada a consulta da tabela TripAttraction por AttractionId, mas também são considerados apenas uma vez os registros que compartilham o mesmo AttractionName. Neste caso, para o primeiro registro serão esses quatro. Mas desses, quantos passarão pelo filtro? O primeiro não passa, o segundo sim, o terceiro não e o quarto sim.

Se a fórmula Count inclui também a mesma condição, então na saída será exibido Louvre Museum, e 2.
Se, em vez disso, não tivéssemos adicionado a mesma condição de filtro à fórmula, sairá Louvre Museum e 4.

Então vamos para o próximo registro a ser processado no For each. É o único que nos resta, cuja data não atende ao filtro, portanto, nada mais é processado. O resultado final será este (dependendo se foi adicionada ou não a condição de filtro à fórmula).

Trip Attraction

For Each TripAttraction (Line: 16)

Order: AttractionId
Index: ITRIPATTRACTION1
Unique: AttractionName
Navigation filters: Start from: FirstRecord
Loop while: NotEndOfTable
Constraints: TripDate > &Today
Join location: Server

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

Formulas

Navigation to evaluate: count(TripAttractionVisitTime)

Where: TripDate > &Today
Given: AttractionName &Today
Index: ITRIPATTRACTION
Group by: AttractionName

=TripAttraction
=Trip (*TripId*)
=Attraction (*AttractionId*)

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

Formulas

Navigation to evaluate: count(TripAttractionVisitTime)

Given: AttractionName
Index: ITRIPATTRACTION
Group by: AttractionName

=TripAttraction
=Attraction (*AttractionId*)

De fato, se observamos a lista de navegação, vemos claramente que a fórmula é calculada como desejamos. Observemos o Group by e como no Where é mostrado o filtro sobre os registros a serem contados.

Se removermos a condição da Count, então a lista de navegação informará este outro para a fórmula.

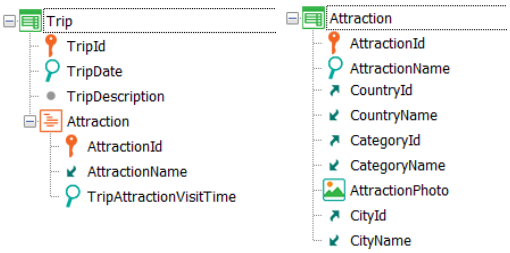
Restrictions

```
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
```

```
For each    ≠ Base table

endfor
```

Por último, vamos mencionar a última restrição: se utilizarmos cláusula unique, só faz sentido aninhar outro for each se não tiver a mesma tabela base. Ou seja, não podemos utilizar unique em cortes de controle, como se poderia em princípio 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

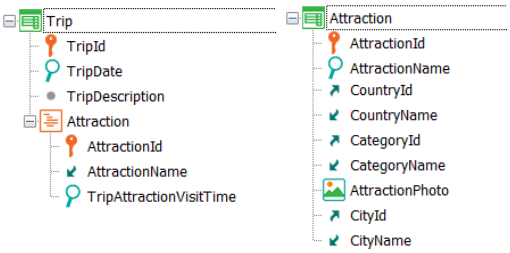
TripId*	TripDate	TripDescription
1	01/01/2023	...
2	04/04/2023	...
3	01/01/2023	...
4	05/05/2023	...

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	....

Assim, por exemplo, se quisermos uma lista das atrações que estão em trips, com a duração da visita à atração em cada uma dessas trips, não teremos outra alternativa a não ser implementá-la como o clássico corte de controle.

Percorremos TripAttraction ordenada por AttractionName e para o primeiro registro de TripAttraction com o primeiro AttractionName imprimimos o nome da atração e então iteramos pelos registros com o mesmo AttractionName, aqui apenas este. Imprimimos a data da trip e o tempo de visita, e passamos para o próximo grupo…




```
For each Trip.Attraction
   order AttractionName
      print AttractionInfo //AttractionName
      For each Trip.Attraction
         print TripAttractionInfo //TripDate TripAttractionVisitTime
      endfor
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	8	60
4	8	240

Eiffel Tower

01/01/2023 120

Louvre Museum

01/01/2023 180

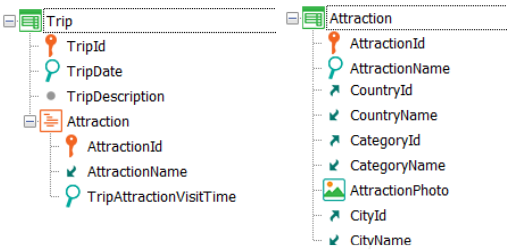
04/04/2023 200

01/01/2023 60

05/05/2023 240

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

...que é o que corresponde a estes registros com o mesmo nome em Attraction. Imprimimos o nome e, novamente, iteramos com o for each aninhado, enquanto não mude o AttractionName. E assim temos na saída...



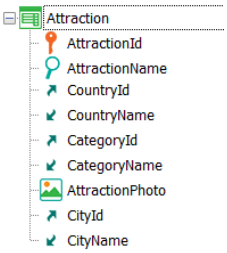

```
For each Trip.Attraction
   order AttractionName
      print AttractionInfo //AttractionName
      For each Trip.Attraction
         print TripAttractionInfo //TripDate TripAttractionVisitTime
      endfor
endfor
```

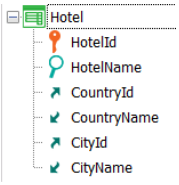
```
For each Trip.Attraction
   unique AttractionName
      print AttractionInfo //AttractionName
      For each Trip.Attraction
         print TripAttractionInfo //TripDate TripAttractionVisitTime
      endfor
endfor
```

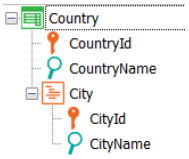
Errors

spc0211 Unique clause in break group not supported in group starting at line 24.

Agora, se quiséssemos implementar a mesma coisa com a cláusula unique, GeneXus não permitirá. A lista de navegação nos mostrará este erro.








```
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	...

Mas podemos aninhar um for each que navegue outra tabela.

Por exemplo, vejamos este caso em que estamos percorrendo a tabela de atrações, ordenando por CountryName, da estendida, e pedindo para processar uma única vez todos os registros que repitam o valor de CountryName.

Para eles, queremos contar as atrações do mesmo país, imprimir esse país com o número de atrações e navegar na tabela Hotel imprimindo os nomes de hotéis do mesmo país.

Assim, com estes dados, veremos na saída o seguinte. Aqui temos a tabela Attraction ordenada por CountryName.
Há apenas um registro para o país do primeiro registro: é listado então seu país e 1 para a conta. Como não há nenhum hotel de Brazil, passará para o próximo registro em Attraction, que será o 3, correspondente a France. Existem 3 registros com o país France, então na saída será visto... E então é executado o for each aninhado, que imprimirá os hotéis de France. Portanto, será visto na saída…

E por último teremos…

For Each Attraction (Line: 47)

Order: CountryName
No index!
Unique: CountryName
Navigation filters: Start from: FirstRecord
Loop while: NotEndOfTable
Join location: Server

=Attraction (*AttractionId*) INTO CountryId
=Country (*CountryId*) INTO CountryName
=count(AttractionName) navigation (*CountryName*)

Formulas

Navigation to evaluate: count(AttractionName)

Given: CountryName
Index: IATTRACTION
Group by: CountryName

=Attraction
=Country (*CountryId*)

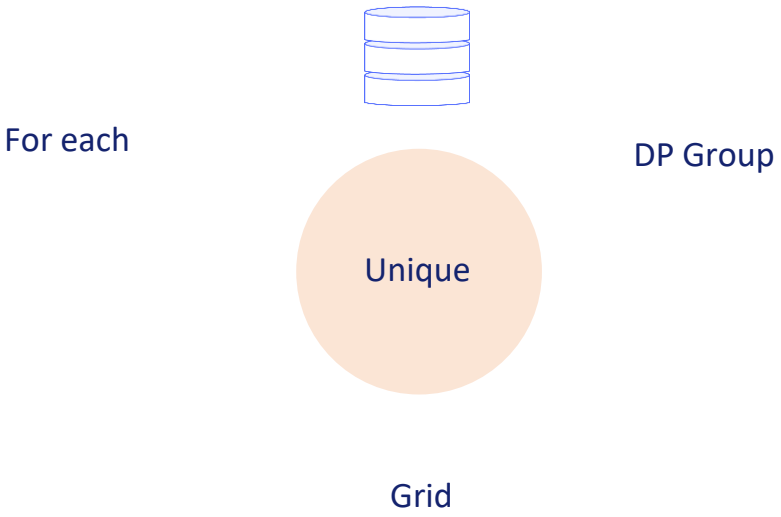
For Each Hotel (Line: 54)

Order: HotelId
Index: IHOTEL
Constraints: CountryName = @CountryName
Join location: Server

=Hotel (*HotelId*) INTO CountryId HotelName
=Country (*CountryId*) INTO CountryName

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

Se observarmos a lista de navegação, ficará claro para nós que está sendo implementado o que queríamos. Observemos a Constraint do for each aninhado por CountryName.



Por último, para encerrar, mencionar novamente que, embora nos concentramos na cláusula unique para o comando For each, sua lógica é válida para todas as outras formas de consulta.

GeneXus™

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