Mais sobre For eachs aninhados

Casos e navegação

GeneXus

A primeira coisa que GeneXus deve fazer em um caso de for eachs aninhados é determinar as tabelas base de cada um.

Quando o desenvolvedor indica Transação Base, isto é imediato.
Quando não o faz, GeneXus deve determinar cada tabela base a partir dos atributos presentes em cada for each. Este caso, mais complexo, não será abordado neste vídeo.

Então, a partir daí, define a navegação para resolver a consulta múltipla. Ocorrerá um de três casos:

-Join
-Produto Cartesiano
-Corte de controle

For each aninhados

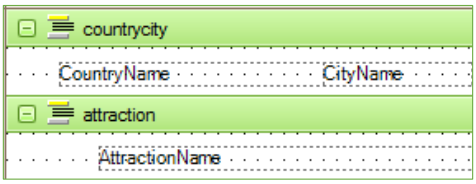
```
For each Attraction order AttractionName
      Where CountryId = 1
      &attractionName = AttractionName
      &attractionDescription = AttractionDescription

      For each Categories
            &categoryName = CategoryName
      Endfor

      When none
      ....
EndFor
```

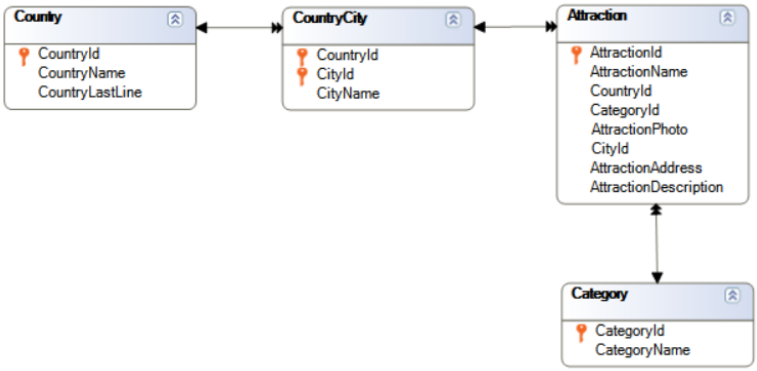
Como dissemos, a primeira coisa que GeneXus faz ao encontrar um par de for eachs aninhados é determinar a tabela base de cada um, de maneira ordenada, de fora para dentro, começando pelo mais externo. Só então determina a navegação.

Para cada for each, intervém a transação base indicada, e apenas os atributos próprios daquele for each: tanto da ordem, where, etc., bem como aqueles que estão em seu corpo, exceto aqueles que se encontram dentro de um for each aninhado. Ou seja, removendo o for each aninhado, se determina a tabela base como no caso de um for each simples. Nunca são levados em consideração os atributos da cláusula When none. Todos os atributos devem pertencer à tabela estendida da tabela base encontrada, a qual coincide com a tabela base associada à transação base. Os atributos que não cumpram com isto não serão "instanciáveis", pois não é possível chegar a eles.

For each aninhados

```
For each Country.City
        print countrycity
        For each Attraction
              Print attraction
        Endfor
EndFor
```





No exemplo, se procede nesta ordem:

1- É determinada a tabela base do for each externo. Para isso, é considerada a transação base indicada, ou seja, Country.City, e verifica-se se os atributos presentes no printblock (CountryName e CityName) pertencem à sua tabela estendida. Caso contrário, é produzido um aviso na lista de navegação, que informará que alguns atributos não poderão ser instanciados, pois não podem ser alcançados a partir da tabela estendida desse for each. Neste caso, CountryName e CityName pertencem à estendida CountryCity, tabela base do for each.

2- É determinada a tabela base do for each aninhado. É considerada a transação base Attraction e o atributo AttractionName presente no printblock. Não são levados em consideração os atributos do for each exterior. Se não tivesse sido escrito transação base, então sim, seria considerado algo relativo aos atributos do for each externo para determinar a tabela base do interno, mas este não é o caso. Em seguida, é determinada sua tabela base como se fosse um for each independente. Portanto, sua tabela base será Attraction.

For each aninhados

Tabela base diferente

For each externo ⟷ For each aninhado

Join

```
For Each Category
        ...
        For each Attraction
              ...
        Endfor
Endfor
```

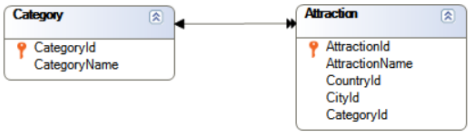


Tabela Category

CategoryId	CategoryName
1	Museum
2	Monument
3	Tourist site

Tabela Attraction

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	3
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	3
5	Christ the Redemmer	1	2	2

A partir da determinação das tabelas base, surgem os três casos de For eachs aninhados que já vimos antes e que queremos conceituar aqui.

Quando as tabelas base são diferentes, são abertas duas possibilidades: ou existe relação 1 para N direta ou indireta entre elas, ou não existe. No primeiro caso, para cada registro do for each principal, o for each aninhado executará suas instruções apenas para os N registros relacionados. A esta operação de cortar a informação de uma tabela pela de outra, se conhece como Join.

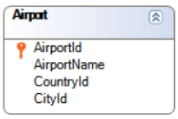
For each aninhados

Tabela base diferente

For each externo — For each aninhado

Produto cartesiano

```
For each Airport
      .....
      For each Attraction
            ....
      Endfor
EndFor
```



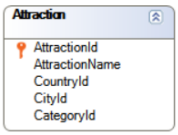


Tabela Airport

AirportId	AirportName	CountryId
1	Guarulhos	2
2	Charles de Gaulle	1
3	Tegel	3

Tabela Attraction

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	3
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	3
5	Christ the Redemmer	1	2	2

No segundo caso, quando não existe relação, para cada registro considerado no for each principal, o for each aninhado executará suas instruções para todos os registros da outra tabela, uma vez que não encontrou relação entre elas. A operação é conhecida como Produto Cartesiano.

For each aninhados

Mesma tabela base

Corte de controle

```
For each Attraction order CategoryId
      .....
      For each Attraction
            ....
      Endfor
Endfor
```

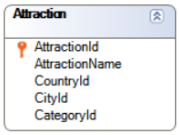


Tabela Attraction

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
6	The Centre Pompidou	2	1	1
7	Quai Branly	2	1	1
3	Eiffel Tower	2	1	2
5	Christ the Redemmer	1	2	2
2	The Great Wall	3	1	3
4	Forbidden city	3	1	3

Quando as tabelas base forem iguais, será um caso conhecido como Corte de controle: é quando precisamos agrupar as informações de uma tabela, executar certas instruções que levam em consideração a informação comum do grupo e depois percorrer cada membro do mesmo, e executar outras instruções para, em seguida, passar para o próximo grupo e repetir o processo. Neste caso, é fundamental especificar os atributos que compõem o grupo, utilizando a cláusula order.

Este caso ocorre porque foi especificada a mesma transação base para o for each externo e para o aninhado.
Se não tivessem sido especificadas as transações bases dos for eachs, como dissemos, GeneXus as calculará a partir dos atributos que encontre dentro desses for eachs. Porém, apenas inferirá que se deseja implementar um corte de controle se as tabelas bases encontradas por ele forem a mesma. Não entraremos em detalhes neste curso.

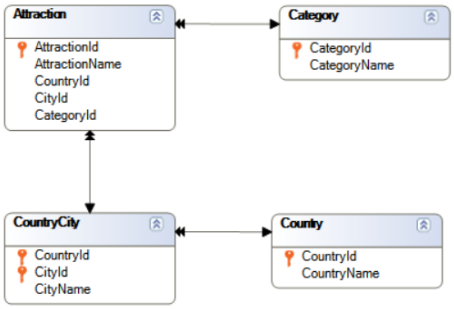
Vejamos este outro caso.

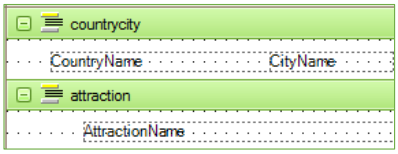
For each aninhados

Tabela base diferente

For each externo ⟷ For each aninhado

Corte de controle






```
For each Attraction
    print attraction
    For each Country City
        print countrycity
    endfor
endfor
```

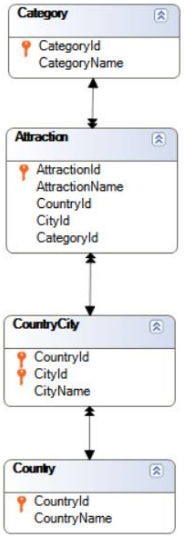
```
For each Attraction
    print attraction
    print countrycity
endfor
```

Observemos que aqui o segundo for each seria desnecessário, pois para cada atração na qual se está posicionado em um determinado momento no for each externo, existe apenas um registro de CountryCity relacionado. Seria o mesmo, então, não ter escrito o segundo for each, e diretamente enviar para imprimir o printblock countrycity, que contém CountryName e CityName, ambos pertencentes à tabela estendida de Attraction.

For each aninhados

1

```
For each Country.City
    Print countrycity
                ↕ 1 – N Direto
    For each Attraction
        Print attraction
    Endfor
Endfor
```



2

```
For each Country
    Print country
                ↕ 1 – N Indireto
    For each Attraction
        Print attraction
    Endfor
Endfor
```

Vamos analisar agora estes dois casos de relação 1 para N entre os for eachs e, portanto, de JOIN.

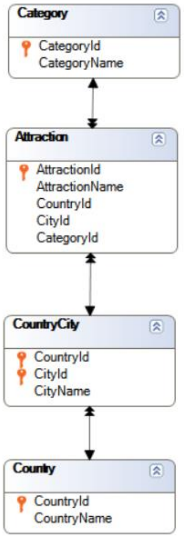
A primeira é direta. Observemos que as tabelas base do for each externo e aninhado são CountryCity e Attraction, respectivamente, que estão relacionadas por uma relação 1 para N.

Serão impressos o nome do país e cidade, e para cada par, seus nomes de atração

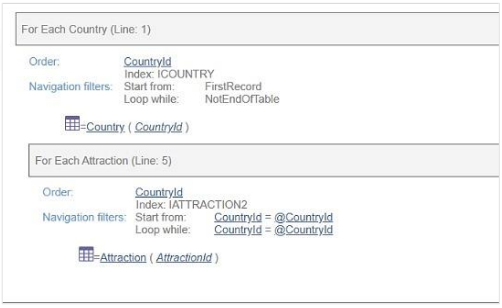
A segunda é indireta. As tabelas base do for each externo e aninhado são Country e Attraction. Se observamos bem, cada país tem N cidades, a cada uma das quais correspondem N atrações. As tabelas base dos for eachs não têm uma relação direta 1 para N, mas sim indireta, por meio da tabela CountryCity. Em outras palavras, observemos que a tabela base do primeiro for each, Country, está incluída na tabela estendida da tabela base do for each aninhado: Attraction. Então será realizado um join

For each aninhados

```
For each Country.City
    Print countrycity
                ↕ 1 – N Direto
    For each Attraction
        Print attraction
    Endfor
Endfor
```




```
For each Country
    Print country
                ↕ 1 – N Indireto
    For each Attraction
        Print attraction
    Endfor
Endfor
```



Vejamos as listas de navegação. Para o for each aninhado, que em ambos os casos navega Attraction, não é percorrida a tabela inteira. Observemos que no primeiro caso, onde a relação é direta, vemos com a arroba que será filtrado pela chave estrangeira composta que é a que estabelece a relação, CountryId e CityId. No segundo caso, vemos que será apenas por CountryId, que faz parte da chave estrangeira composta.

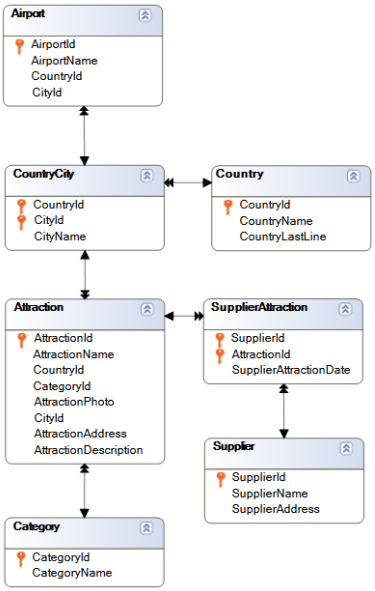
Observe que em ambos os casos, em vez de ordenar o percurso do for each aninhado pela chave primária de Attraction, que é AttractionId, o faz pelo atributo ou conjunto de atributos relação, para os quais conta com índice criado automaticamente, por chave estrangeira. Desta forma, o acesso à base de dados será otimizado.

Portanto, quando GeneXus determina que realizará um Join, tenta otimizar sua navegação.

For each aninhados

```
For each Supplier.Attraction
    print info
                ↕  1 – N Indireto
    For each Airport
        print airports
    endfor
endfor
```

Join



Aqui vemos um terceiro exemplo. Neste caso, foram adicionados fornecedores que oferecem atrações turísticas e, portanto, aparecem mais duas tabelas: Supplier e sua subordinada, SupplierAttraction.
Também temos os aeroportos, que pertencem a um país e cidade.
Então, se observamos o for each, temos que a tabela base do for each externo é SupplierAttraction e a tabela base do for each aninhado é Airport. Suponhamos, para este exemplo, que no printblock info é exibido o nome do fornecedor, o nome da atração e a data em que será exibida essa atração, e que no segundo printblock é exibido o nome do aeroporto.

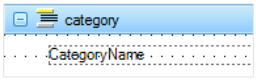
Observemos que existe uma relação 1 para N indireta. Ou seja, para cada registro de SupplierAttraction, existirá apenas um de Attraction a partir do qual obtemos apenas um de CountryCity, que por sua vez está relacionado com N registros de Airport (os N aeroportos que se encontram nesse país/cidade, embora geralmente, na realidade exista apenas um. No nosso modelo, podem existir vários).

Indo mais fundo, aqui GeneXus descobre que há atributos em comum entre a tabela estendida do for each principal e a tabela base do aninhado. Quais? O par {CountryId, CityId}. E por eles realizará o join.

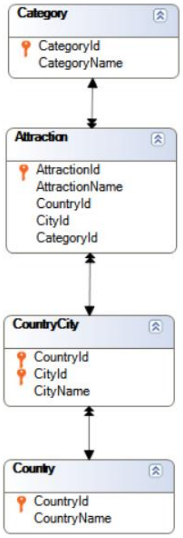
Para otimizar vemos que está escolhendo o índice por chave estrangeira da tabela Airport. Resumindo, este é um caso em que a relação entre as tabelas base não é 1 para N direta, mas indireta, por meio de tabelas intermediárias.

For each aninhados

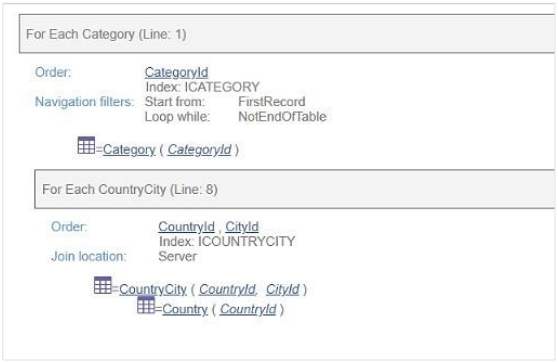
```
For each Category
    Print
```




```
    For each Country.City
        Print
    Endfor
Endfor
```



Produto Cartesiano



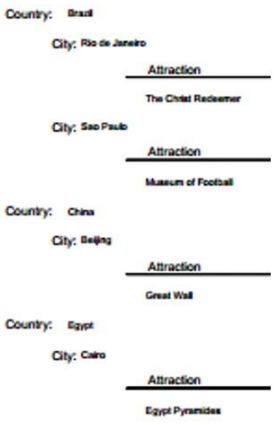
Vejamos agora este caso de for each aninhados, com tabelas base diferentes e não relacionadas.

Aqui, o primeiro for each navega Category, enquanto o aninhado navega CountryCity. Neste caso, não há uma relação 1 para N de nenhum tipo, ao contrário do que possa parecer. Se observamos o diagrama de tabelas, vemos que uma categoria tem N atrações, cada uma das quais tem um e apenas um CountryCity. Poderia se pensar que, então, será listada para cada categoria, o país cidade de cada uma de suas atrações. Isso seria assim se a tabela base do for each aninhado fosse Attraction. Mas não é, é CountryCity. Portanto, GeneXus entende que não estamos buscando os registros relacionados por attraction. Se estivéssemos buscando por eles, bastaria especificar como transação base Attraction para o segundo for each. Este pode ser um caso enganoso, por isso é essencial ler bem a lista de navegação para não se confundir. Na lista, vemos claramente que não está sendo realizado nenhum join.

Neste caso, GeneXus não consegue encontrar uma relação 1-N direta ou indireta entre as tabelas e, portanto, não aplica filtros implícitos aos registros do For each aninhado. Ou seja, realiza um produto cartesiano entre as tabelas: para cada registro da tabela base do for each externo (Category), considera todos os registros da tabela base do aninhado (CountryCity).

For each aninhados

```
For each Attraction order CountryName
      Print                                  ──►
      For each Attraction order CityName
            Print                            ──►
            Print                            ──►
            For each Attraction
                  Print                      ──►
            Endfor
      Endfor
Endfor
```

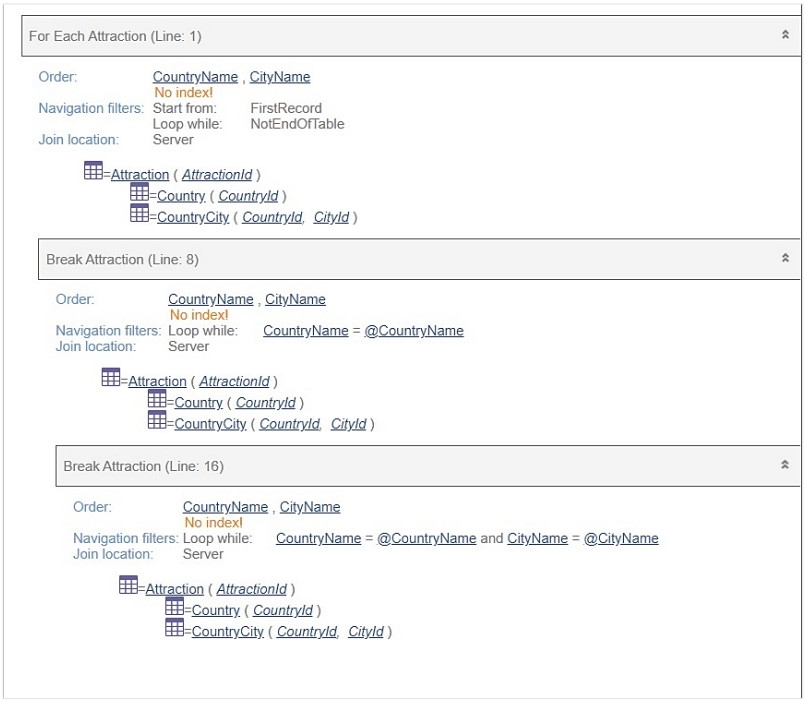


Aqui vemos um caso em que queremos listar cada país, e para ele cada cidade, e para ela cada atração. A restrição: queremos fazê-lo apenas para os países e cidades para os quais existem atrações turísticas.

Ou seja, teremos que implementar um corte de controle duplo: onde primeiro agrupamos por país, e dentro desse grupo, agrupamos por cidade, e dentro deste último, mostramos os nomes de todas as atrações. Para isso:

Teremos que marcar os critérios de agrupamento utilizando as cláusulas order. Lembremos que, para um corte de controle, a order tem um peso muito forte: não está apenas marcando por qual atributo ou atributos listar a informação, mas também está especificando como esta será agrupada.

Poderíamos especificar uma order para o for each mais interno, mas essa order terá unicamente seu uso convencional. Ou seja, essa sim será utilizada apenas para ordenar. Veremos em um momento.



Corte de controle

Corte de controle

Temos um corte de controle duplo, o que implica três for eachs.

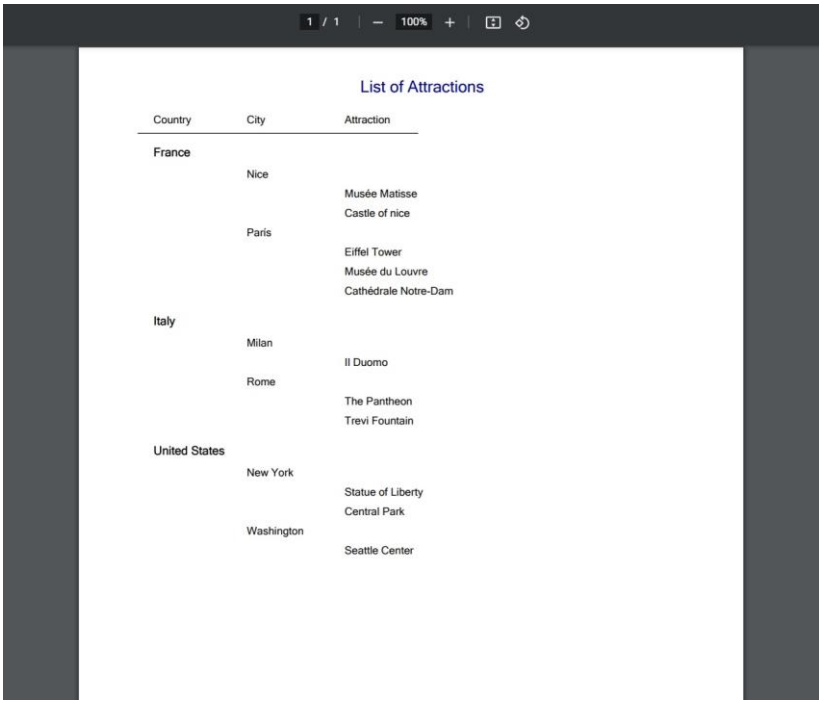
Se observamos a lista de navegação, vemos a palavra Break para cada for each interno, indicando a mesma tabela base, Attraction e, portanto, um corte de controle.

Além disso, percorrerá essa tabela base apenas uma vez, para o que precisa ordenar pela concatenação dos atributos que aparecem nas orders dos for eachs. É por isso que escolhe CountryName, CityName.

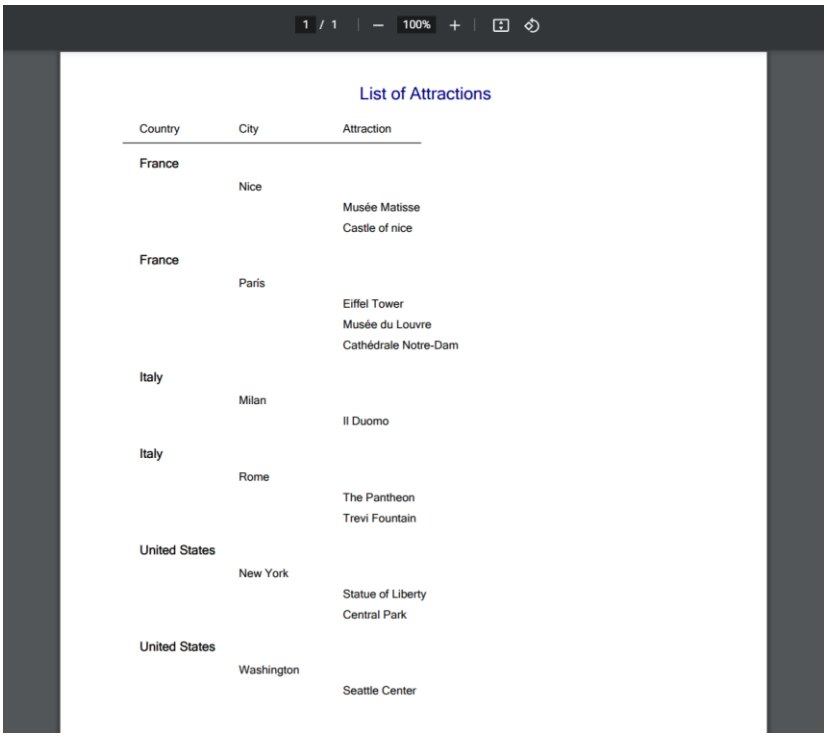
Observe que no segundo for each corta por país, iterando sobre o país no qual se encontra posicionado no primeiro for each. E o terceiro for each corta por país e cidade, iterando sobre a cidade na qual se encontra posicionado no segundo for each.

Aqui vemos um exemplo de como seria exibido se estes fossem os dados. Primeiro vemos o país, e para ele vemos a primeira de suas cidades de acordo com seu nome, e as atrações turísticas desse país e cidade. Então vemos a próxima cidade do mesmo país, e suas atrações, e somente quando não houver mais cidades desse país, o próximo é exibido, em ordem alfabética e começa novamente. País, cidade, atrações.

Pense em como será a execução da lista anterior se em vez de ordenar o primeiro for each por CountryName e o segundo por CityName, tivéssemos ordenado os dois ou apenas o primeiro pelo par CountryName, CityName.
Observe que, neste caso, a lista de navegação será diferente desta que vê acima, no segundo for each. Loop while ali dirá "CountryName = @CountryName and CityName = @CityName".



Isto significa que, em execução, agora a mesma informação será vista deste outro modo. Ou seja, sai o país... sua primeira cidade... e suas atrações turísticas. Depois, o mesmo país e sua segunda cidade... e suas atrações turísticas, e assim por diante, até que muda o país e volta a ocorrer o mesmo para o próximo.
É evidente que as informações estão sendo apresentadas de forma diferente. Apenas por ter colocado CityName no primeiro ou no segundo for each.

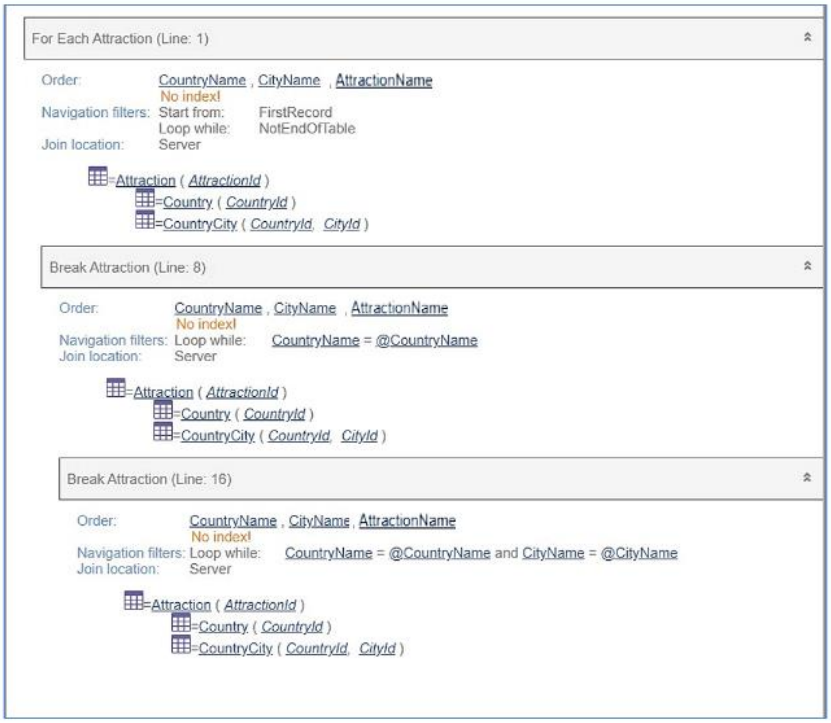


Agora suponhamos que sobre o primeiro exemplo, no qual cortávamos por país e depois por cidade, para então listar as atrações, adicionamos ao último for each, o mais interno, uma order por AttractionName. Ou seja, queremos que as atrações de um país e cidade sejam ordenadas por nome de atração.

A lista de navegação mudará e será adicionado AttractionName à ordem dos três for each, ficando CountryName, CityName e AttractionName.

E se executarmos com esta mudança, veremos exatamente o que dissemos: está sendo cortado por nome de país, por isso sai primeiro França e só muito depois Itália, e dentro de França, está sendo cortado por CityName, ou seja por cidade, é por isso que sai primeiro Nice, que está alfabeticamente antes de Paris, e dentro de cada um desses subgrupos, as atrações são ordenadas alfabeticamente:

Ao contrário do que acontecia no primeiro caso, quando não tínhamos a order por AttractionName no último for each.

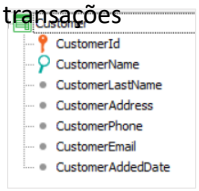


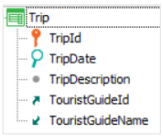
Como vimos, os critérios de corte vemos em Navigation Filters. Aqui estamos dizendo que ordene a navegação pelo conjunto CountryName, CityName, AttractionName e que execute o corpo do primeiro for each, onde apenas é impresso o CountryName, e depois execute o segundo for each, no qual, desde que não mude o CountryName, deve imprimir o printblock que contém apenas CityName, mas imediatamente executar o for each interno, onde enquanto não mude o par countryName cityName, deve imprimir o AttractionName.

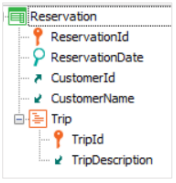
Ou seja, observamos que os filtros permanecem exatamente os mesmos de quando não ordenávamos o for each mais interno por AttractionName, portanto, como dissemos, esta order que acabamos de inserir serve apenas para ordenar, e não para realizar um corte

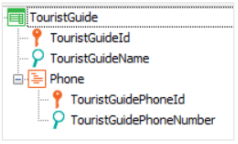
Caso de estudo

Desenho de transações









Source

```
Parm(in:&ReservationDate, in: CustomerId);

For each Reservation.Trip
     Where ReservationDate >= &ReservationDate
     Print Trips
     For each TouristGuide.Phone
          Print TouristGuidesPhones
     Endfor
Endfor
```

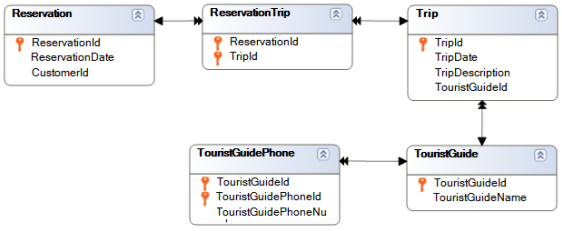
Em um vídeo anterior analisamos um caso de estudo para determinar a forma como GeneXus aplica os filtros quando são recebidos como atributo na regra Parm.

Retomando esse mesmo caso, veremos a forma de evitar que esse filtro seja aplicado.

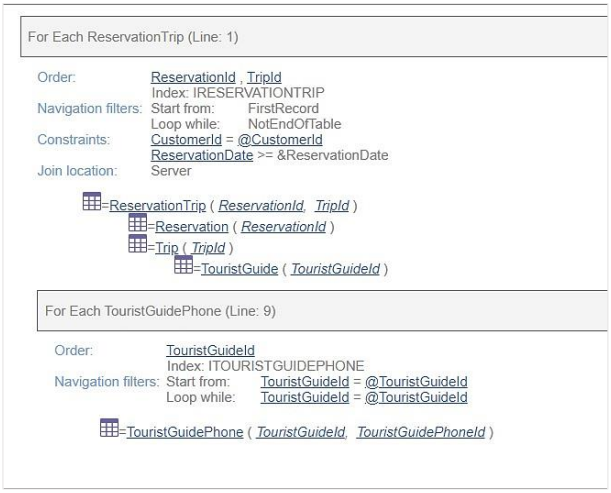
Começamos então do desenho de transações que vemos, e precisamos obter uma lista que mostre, para um determinado cliente, e a partir de uma determinada data, todas as excursões que tem reservadas, e para cada uma delas os telefones de contato do guia turístico responsável.

Para isso, é proposto o source apresentado.

Caso de estudo




```
Parm(in:&ReservationDate, in: CustomerId);

For each Reservation.Trip
     Where ReservationDate >= &ReservationDate
     Print Trips
     For each TouristGuide.Phone
          Print TouristGuidesPhones
     Endfor
Endfor
```



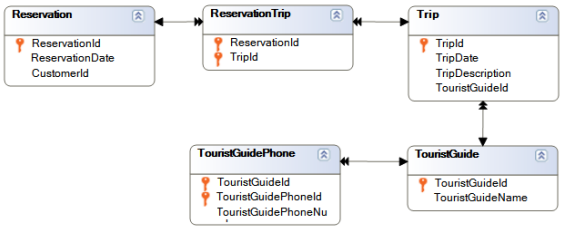
Sabendo que a tabela base do For each externo é RESERVATIONTRIP e que a tabela base do For each interno é TOURISTGUIDEPHONES., estabelece filtros implícitos para a informação que utilizará? Sabemos que sim, que mostrará os telefones do guia de cada excursão

Por quê? GeneXus busca se existe relação entre a tabela estendida do for each externo, e a tabela base do for each aninhado:

É uma forma de buscar uma relação 1 para N, embora neste caso indireta. Se cada RESERVATIONTRIP tem um TouristGuideId, e na tabela a ser navegada
também existe um TouristGuideId, então GeneXus entende que pela relação entre a informação, se tratará do mesmo. E por isso realiza o join.

Vemos isso claramente na lista de navegação onde sempre o @ está indicando informação contextual (observe o @CustomerId, que se refere ao valor recebido no atributo CustomerId na regra Parm). Este @TouristGuideId se refere ao valor do atributo desse nome da tabela TRIP acessada pelo primeiro for each.

Caso de estudo




```
For each Reservation.Trip
     Where ReservationDate >= &ReservationDate
     Print Trips
     Do "ListTouristGuidePhones"
Endfor

Sub "ListTouristGuidePhones"
     For each TouristGuide.Phone
          Print TouristGuidesPhones
     Endfor
EndSub
```

Se não for isto que o programador deseja, há uma maneira de evitar esses filtros automáticos?

A resposta é sim, utilizando uma sub-rotina para encapsular o código deste segundo for each. Ao fazer isso, a lista de navegação irá refleti-lo.

As sub-rotinas são blocos de código, que são definidos em um objeto usando o comando Sub. O qual poderá ser chamado posteriormente, dentro do mesmo objeto e quantas vezes quisermos, através do comando Do.
Em vídeos futuros entraremos em detalhes sobre sua implementação.

GeneXus™

training.genexus.com
wiki.genexus.com