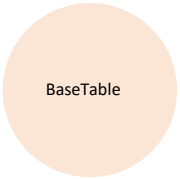
Lógica de consulta à base de dados com GeneXus

Determinação de tabelas base

GeneXus™

Aqui veremos como GeneXus determina a tabela base que será navegada para satisfazer uma consulta.

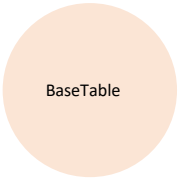



```
For each BaseTrn1, ... , BaseTrnn
    skip expression1 count expression2
    order att1, att2, ... , attn [when condition]
    order att1, att2, ... , attn [when condition]
    order none [when condition]
    unique att1, att2, ... , attn
    using DataSelector ( parm1, parm2, ... , parmn )
    where 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
```

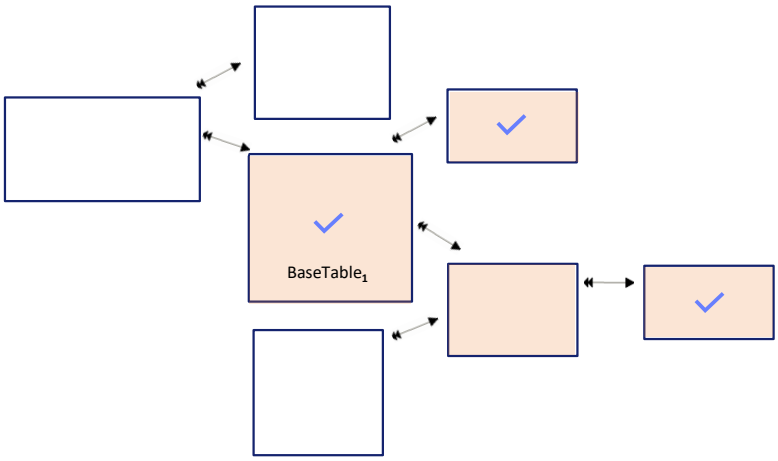
extended table

Vamos nos concentrar no comando for each. Se ele tiver transação base definida, então o problema já está resolvido. A pergunta é o que acontece quando não tiver.

Como é encontrada a tabela base? GeneXus busca os atributos presentes em todos estes lugares e a partir deles busca uma tabela estendida que contenha todos eles.

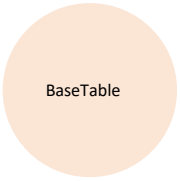



```
For each BaseTrn1, ... , BaseTrnn
    skip expression1 count expression2
    order att1, att2, ... , attn [when condition]
    order att1, att2, ... , attn [when condition]
    order none [when condition]
    unique att1, att2, ... , attn
    using DataSelector ( parm1, parm2, ... , parmn )
    where 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
```

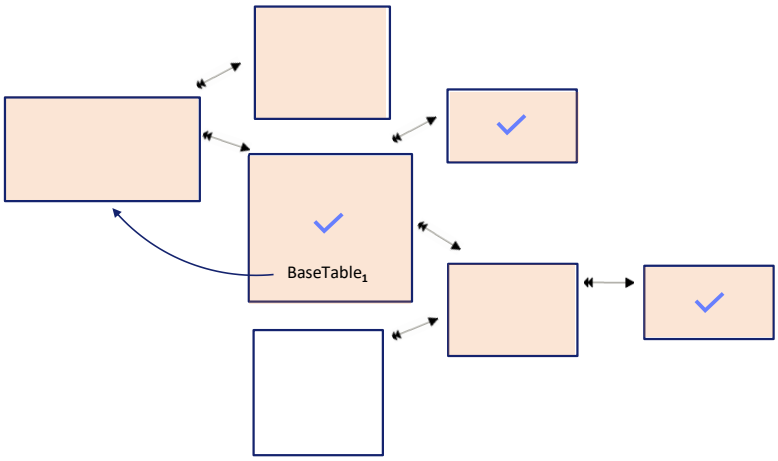


Por exemplo, se os atributos que aparecem nestes lugares são destas tabelas, qual será a tabela base do for each?

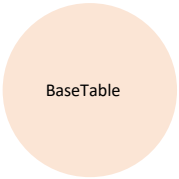
Será esta, porque sua tabela estendida contém todos eles…



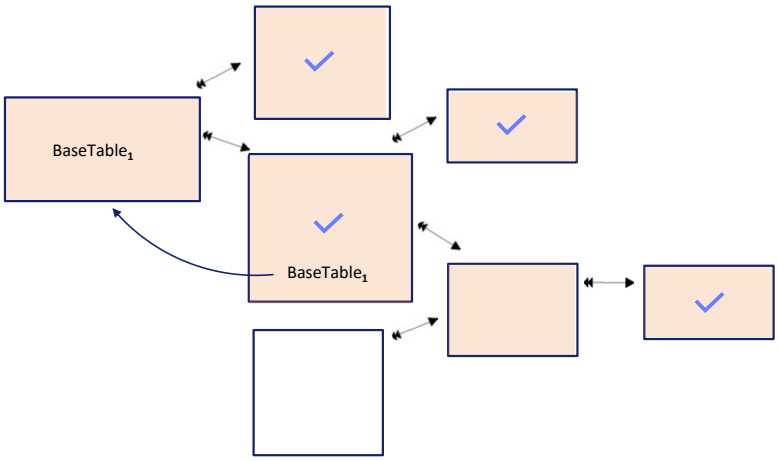

```
For each BaseTrn1, ... , BaseTrnn
    skip expression1 count expression2
    order att1, att2, ... , attn [when condition]
    order att1, att2, ... , attn [when condition]
    order none [when condition]
    unique att1, att2, ... , attn
    using DataSelector ( parm1, parm2, ... , parmn )
    where 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
```



Poderíamos contestar que se escolhesse esta outra como tabela base, também seria verdade que sua estendida contém todos os atributos, então é necessário adicionar uma condição: que de todas as tabelas estendidas que contenham **todos** os atributos, seja a mínima. Aí fica resolvido o problema; agora sim será esta.

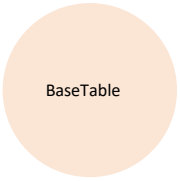



```
For each BaseTrn1, ... , BaseTrnn
    skip expression1 count expression2
    order att1, att2, ... , attn [when condition]
    order att1, att2, ... , attn [when condition]
    order none [when condition]
    unique att1, att2, ... , attn
    using DataSelector ( parm1, parm2, ... , parmn )
    where 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
```



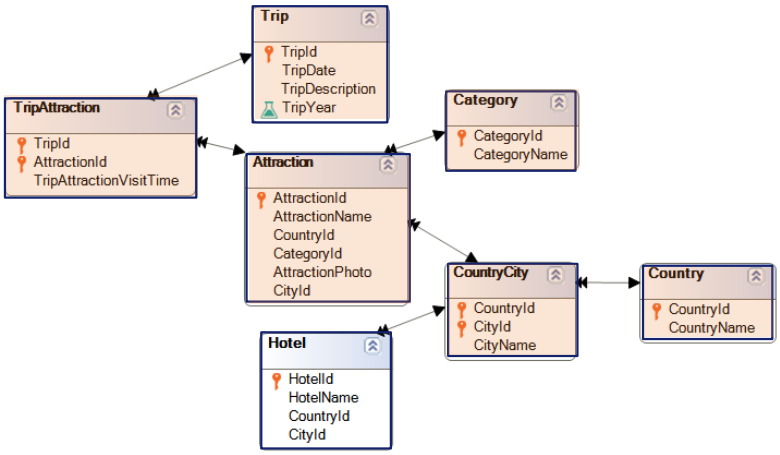
Claro, se aparecesse um atributo desta tabela, a tabela base passaria a ser esta outra.

Vamos pensar neste exemplo…




```
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
order TripDate
where CategoryName = "Monument"
where CountryName = "France"
   print PB1 //AttractionName
endfor
```



Não temos transação base. Este atributo está aqui, este outro aqui, este outro aqui, e no código principal aparece este outro. A lista de navegação mostrará…

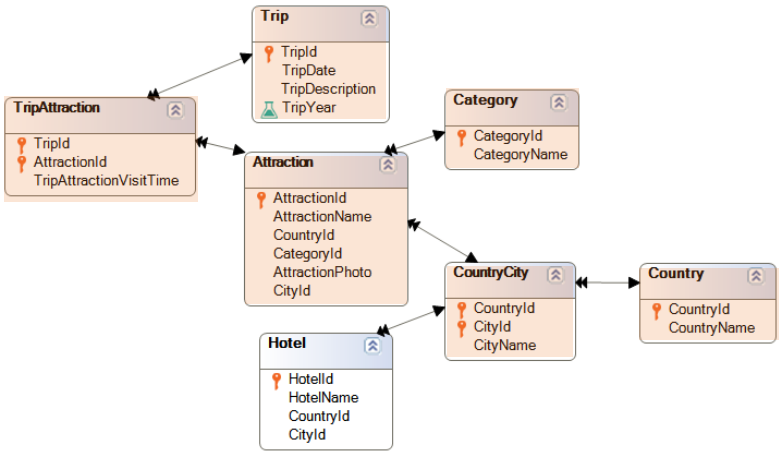
BaseTable

```
for each
order TripDate
where CategoryName = "Monument"
where CountryName = "France"
   print PB1 //AttractionName
endfor
```

For Each TripAttraction (Line: 43)

Order: TripDate
No index!
Navigation filters: Start from: FirstRecord
Loop while: NotEndOfTable
Constraints: CategoryName = "Monument"
CountryName = "France"
Join location: Server

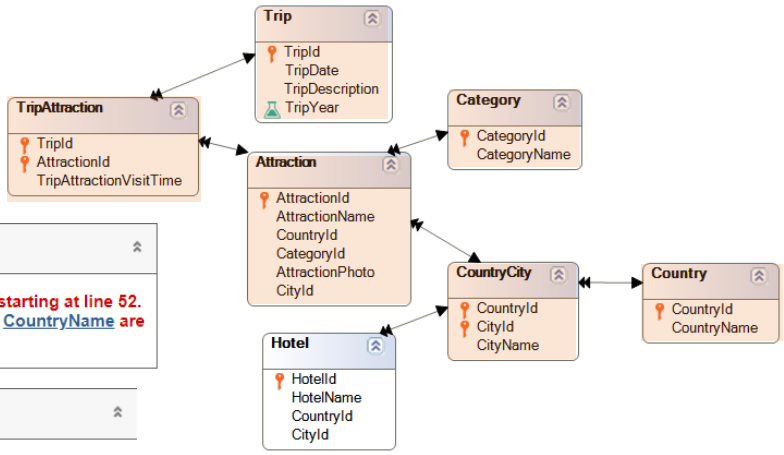
=TripAttraction (*TripId*, *AttractionId*) INTO TripId AttractionId
=Trip (*TripId*) INTO TripDate
=Attraction (*AttractionId*) INTO CountryId CategoryId AttractionName
=Country (*CountryId*) INTO CountryName
=Category (*CategoryId*) INTO CategoryName

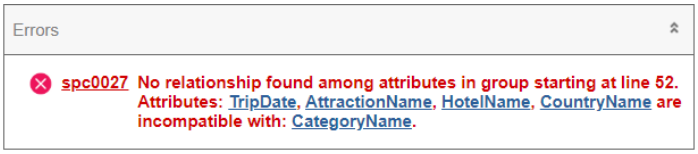


Vemos que está escolhendo, efetivamente, TripAttraction como tabela base.

BaseTable

```
for each Trip.Attraction
order TripDate
where CategoryName = "Monument"
where CountryName = "France"
   print PB1 //AttractionName, HotelName
endfor
```





Agora, o que acontece se não houver nenhuma tabela estendida que contenha todos eles?

Por exemplo, se adicionarmos no printblock o HotelName.

GeneXus lançará um erro e não será possível gerar o objeto.

Observemos que, se tivéssemos especificado transação base, por exemplo TripAttraction, então em vez de um erro, veríamos que somos avisados de que o atributo HotelName não será alcançável, mas a tabela base estará perfeitamente determinada pela transação base.

BaseTable

```
for each
order TripDate
where CategoryName = "Monument"
where CountryName = "France"
   print PB1 //AttractionName
endfor
```

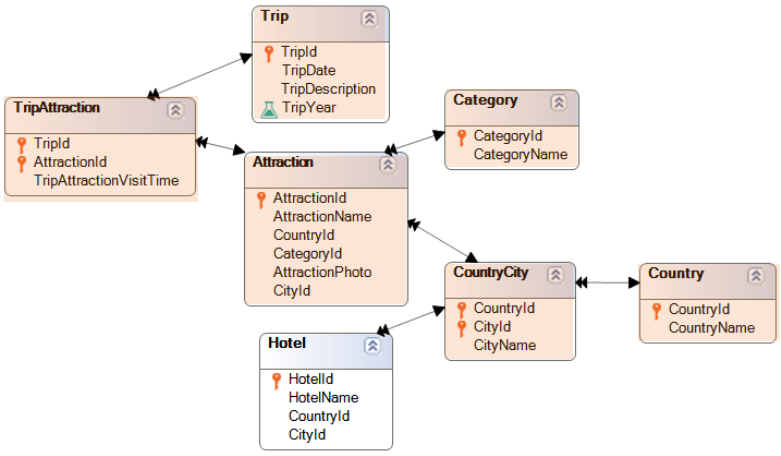
For each

Grid with base table
(with base table)

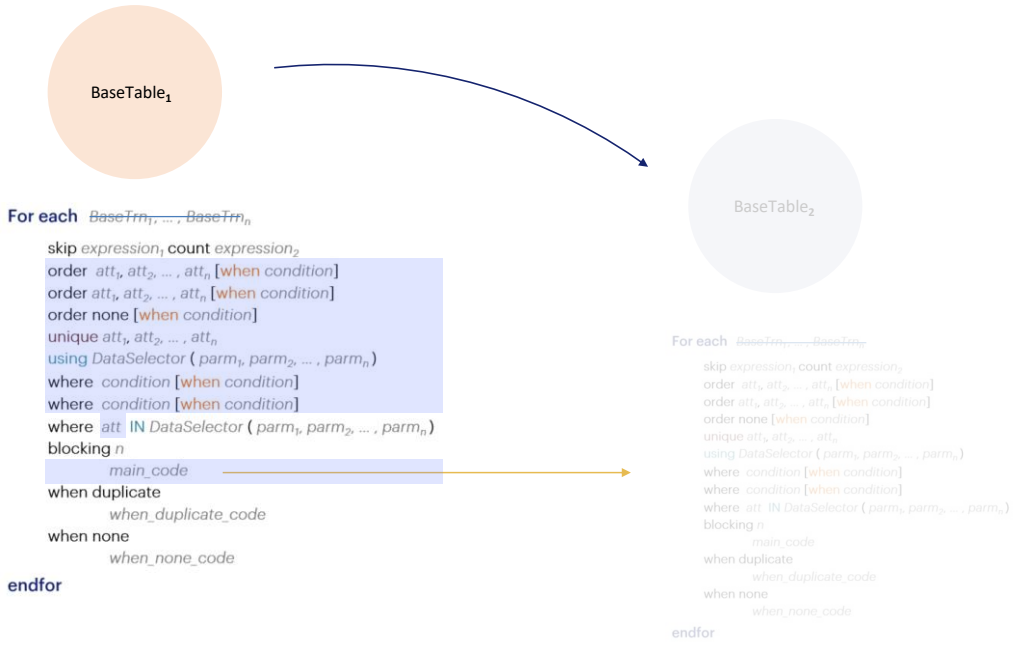
Data Provider Group
(with base table)

Data Selector

Formula



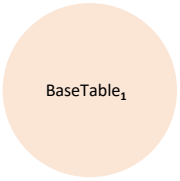
Tudo isto que vimos para um For each também se aplica a um Grid (com tabela base), ou a um grupo de Data Provider. E o mesmo para um Data Selector executado como consulta independente, ou uma fórmula.



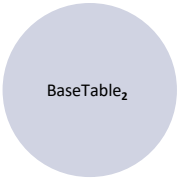
Agora, o que acontece quando há um For each aninhado?

A tabela base do for each principal é determinada sem considerar de forma alguma o For each aninhado. Ou seja, como se ele não existisse dentro do código principal.

E como é determinada a tabela base do For Each aninhado? Esta sempre é determinada **depois** de determinar a do for each que o contém. O caso a analisar é quando não há transação base.



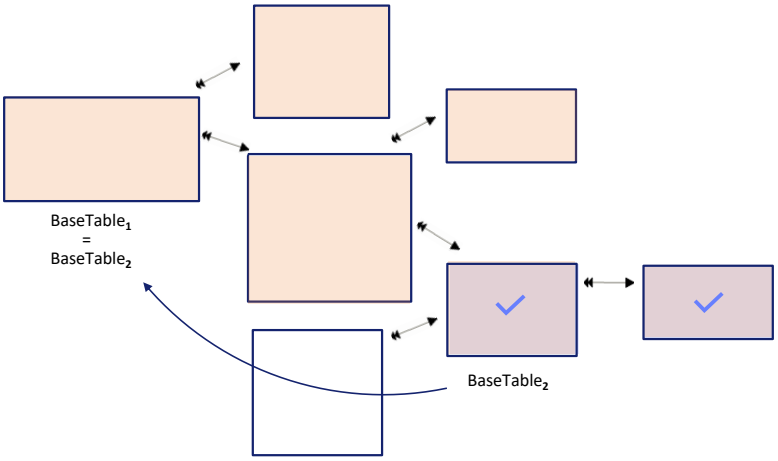

```
For each BaseTrn1, ... , BaseTrnn
    skip expression1 count expression2
    order att1, att2, ... , attn [when condition]
    order att1, att2, ... , attn [when condition]
    order none [when condition]
    unique att1, att2, ... , attn
    using DataSelector ( parm1, parm2, ... , parmn )
    where 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
```




```
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
```

Pode-se pensar que é determinada sua tabela base de maneira idêntica.
Considerando os atributos que aqui se encontram…

```
for each
   …
   for each
      …
   endfor
   …
endfor
```

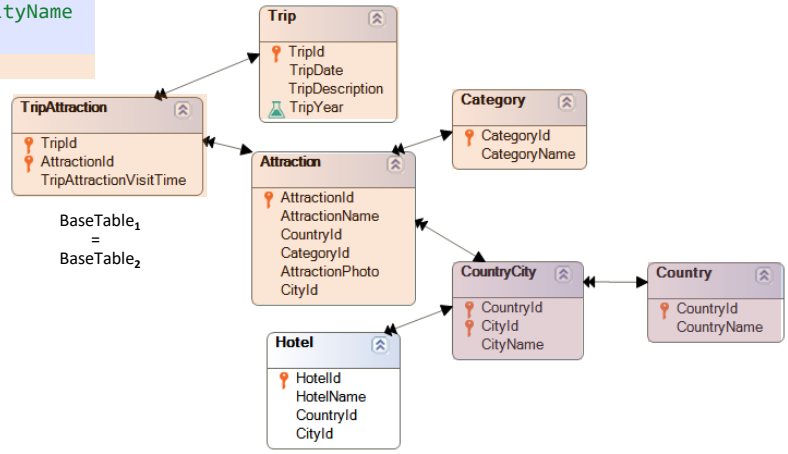


(suponhamos estes)… e encontrando a mínima tabela estendida que os contenha, de forma independente do for each principal.

No entanto, não será exatamente assim. De fato, de todos os atributos extraídos dos locais conhecidos, primeiro são removidos todos aqueles que também fazem parte da estendida do for each principal, e opera-se apenas com os restantes para encontrar a mínima tabela estendida que os contenha.

Neste caso, se retirarmos do conjunto de atributos a calcular aqueles que pertencem à tabela estendida do principal, ficamos com... nenhum! Conjunto vazio. Neste caso, a tabela base do for each aninhado passa a ser a mesma do for each principal. E será, então, um corte de controle.

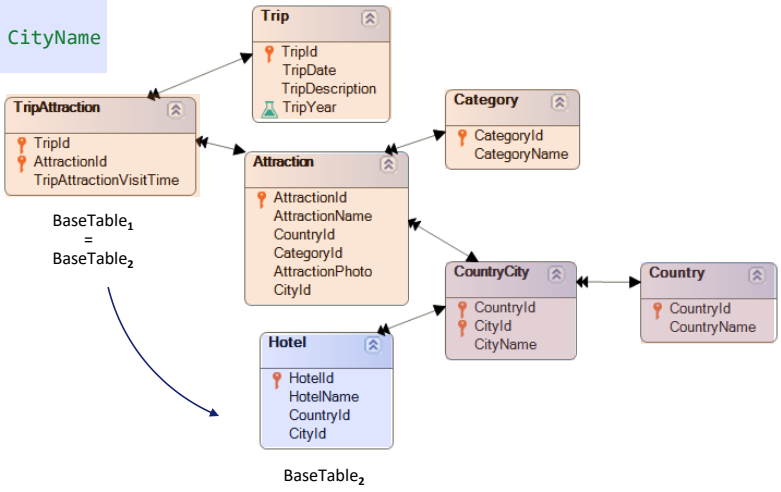
```
for each
   order CategoryName
   where TripDate > &today
   where AttractionName > 'N'
      print PB1 //CategoryName
      for each
         print PB2 //CountryName, CityName
      endfor
endfor
```



Aqui vemos um exemplo concreto. Se é especificado para o for each principal transação base Trip.Attraction ou não, em qualquer caso, pelos atributos envolvidos a tabela base será TripAttraction. E para o for each aninhado, dado que os atributos utilizados nele são desta tabela e desta outra, sua tabela base seria esta... mas acontece que se encontra na estendida do principal, então GeneXus irá escolher como sua tabela base a mesma, TripAttraction.

E será realizado um corte de controle por CategoryName.

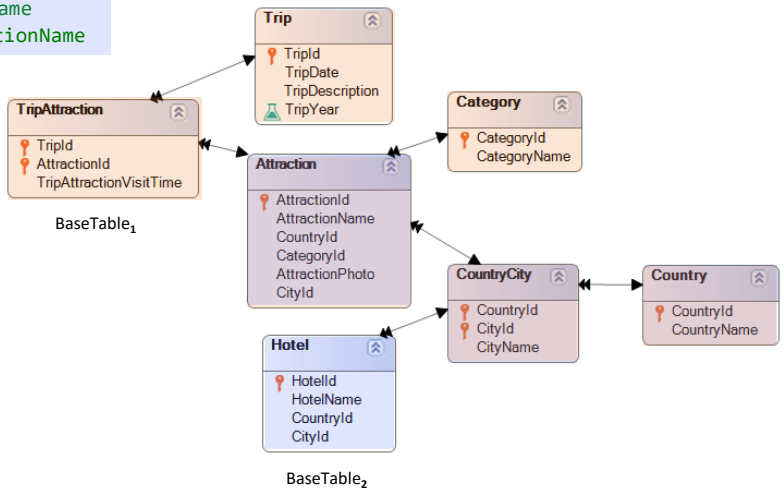
```
for each
   order CategoryName
   where TripDate > &today
   where AttractionName > 'N'
      print PB1 //CategoryName
      for each
         order HotelName
            print PB2 //CountryName, CityName
      endfor
endfor
```



Se foi adicionada, por exemplo, uma order por HotelName, então ali as coisas mudam, e a tabela base do for each aninhado será Hotel.

Observemos que aqui destes três atributos do For each, teremos que retirar este e este para o cálculo da tabela base, pois já estão na estendida de TripAttraction. Só nos resta HotelName para calcular a mínima tabela estendida que o contém.

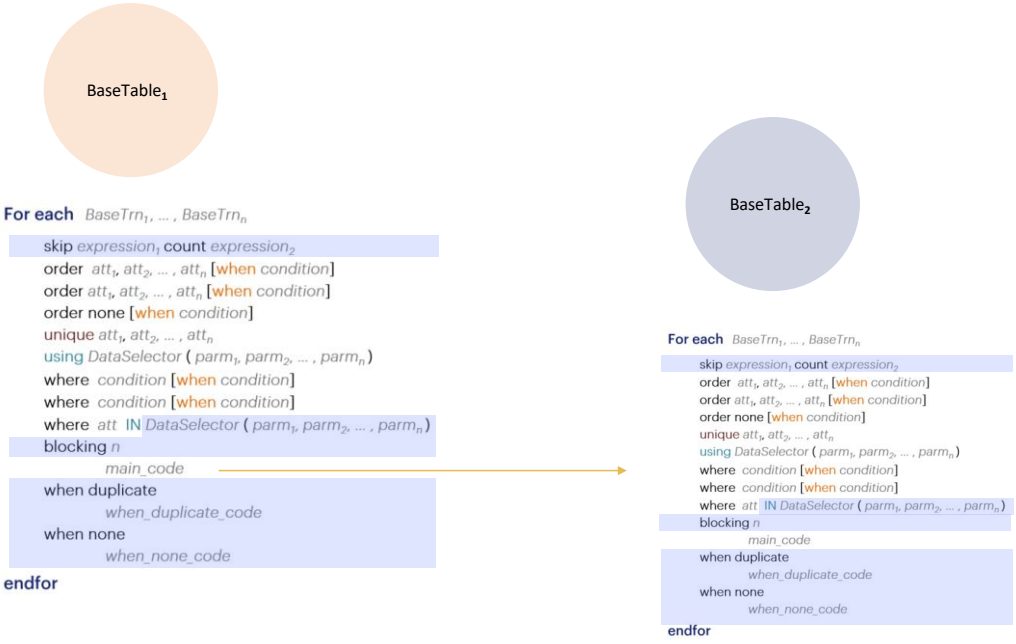
```
for each
   order CategoryName
   where TripDate > &today
   where AttractionName > 'N'
      print PB1 //CategoryName
      for each
         order HotelName
            print PB2 //CountryName, CityName
      endfor          // HotelName, AttractionName
endfor
```



Isto é assim porque poderíamos, eventualmente, no for each aninhado utilizar atributos da tabela estendida do principal, para fazer algo com os valores **perfeitamente determinados** desses atributos.

Por exemplo, vamos imaginar que no print block do for each aninhado adicionamos HotelName, que não faz nenhuma diferença, mas também AttractionName. Para a determinação da tabela base do aninhado não participarão nem CountryName, nem CityName nem AttractionName, por estarem na estendida do principal. Claramente a tabela base será Hotel. O que será executado? Para cada TripAttraction que passar pelos filtros haverá **um** valor de AttractionName e **esse** valor é o que será exibido para cada registro do for each aninhado.

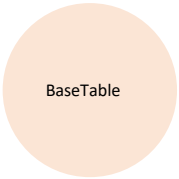
Assim, para todos os hotéis da cidade da atração da trip, aparecerão esse país e cidade juntamente com o nome do hotel e o nome **dessa** atração, sempre a mesma para todos estes registros.



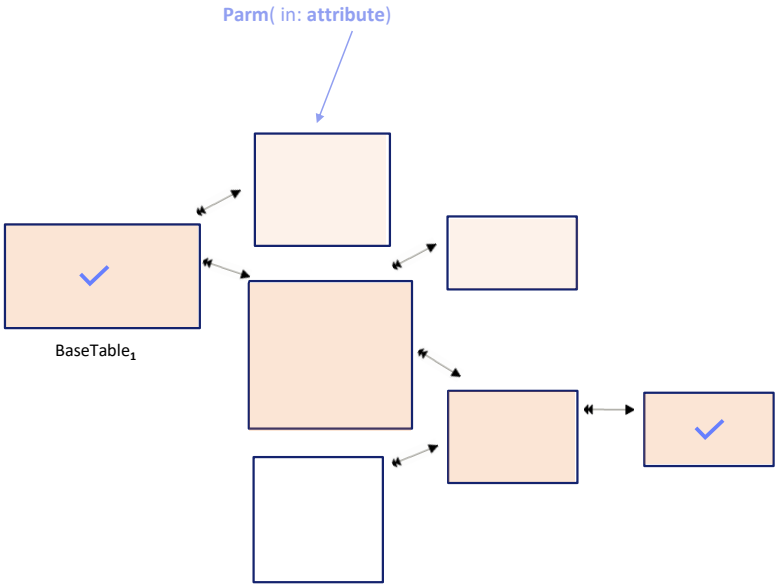
É importante ter em mente que os atributos que se encontrem nestes locais não participarão da determinação da tabela base da consulta respectiva.

Claro que pode ser especificada transação base para um e o outro não, ou para nenhum, ou para ambos.

Depois de determinar as tabelas base, somente **após**, GeneXus resolve a navegação. Em particular, quais condições aplicará implicitamente. Por exemplo…

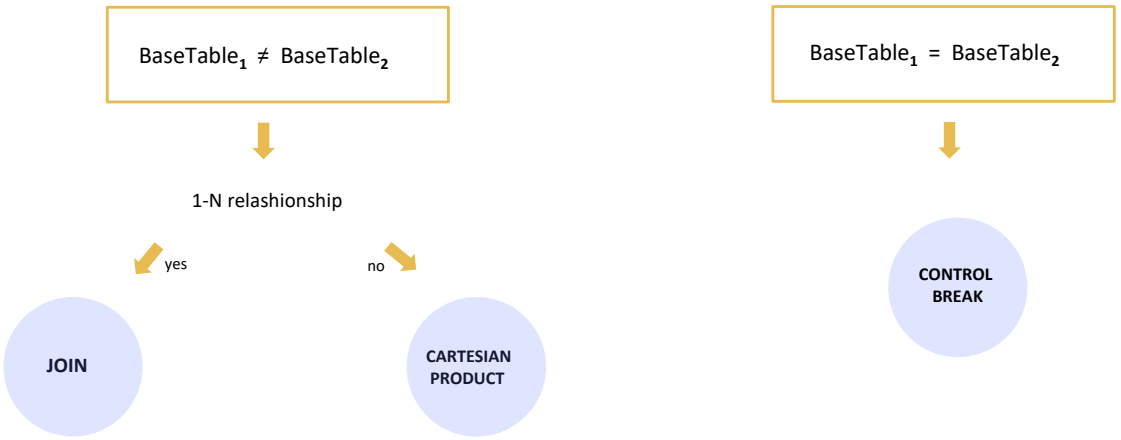



```
For each BaseTrn1, ... , BaseTrnn
    skip expression1 count expression2
    order att1, att2, ... , attn [when condition]
    order att1, att2, ... , attn [when condition]
    order none [when condition]
    unique att1, att2, ... , attn
    using DataSelector ( parm1, parm2, ... , parmn )
    where 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
```

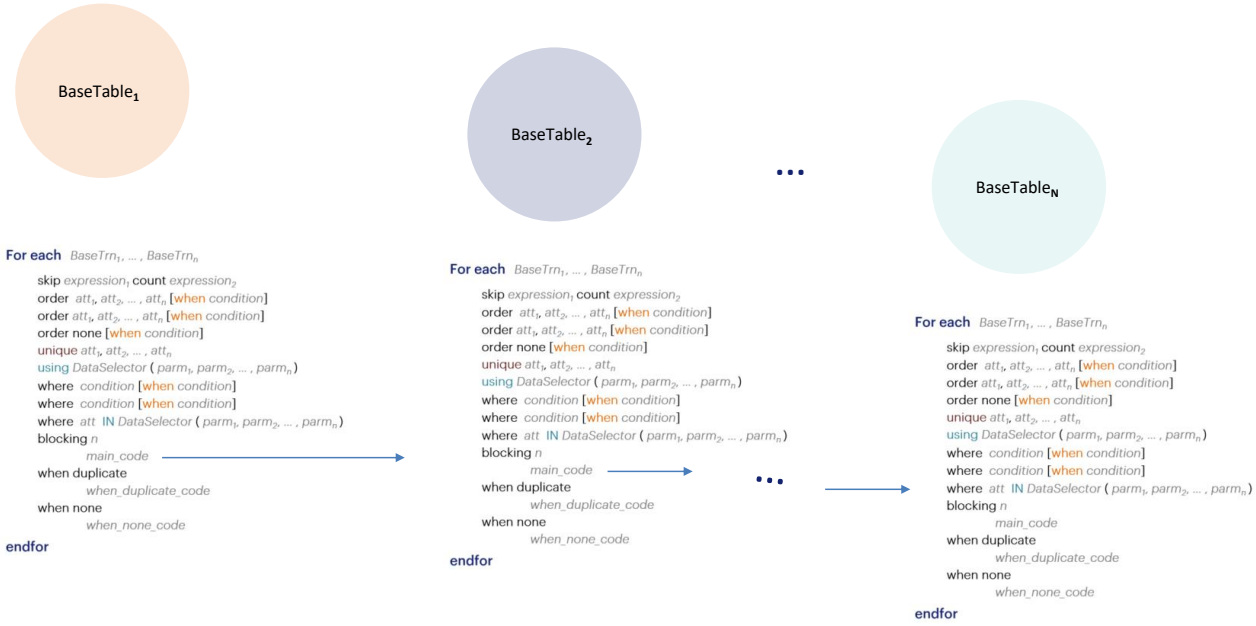


Lembremos que se dentro do for each não for necessário de jeito nenhum acessar uma tabela particular da estendida, por exemplo, suponhamos que apenas são utilizados atributos destas tabelas, então somente serão acessadas estas da estendida e não todas. Estas duas permanecerão sem serem acessadas.

Portanto, se, por exemplo, recebêssemos na regra parm em um atributo de uma delas, esse filtro automático por igualdade não será aplicado ao For each. Porque as condições implícitas são colocadas DEPOIS que a navegação foi definida.



O outro exemplo claro: depois de determinadas as tabelas base, só depois, se define a navegação concreta. Ou seja, primeiro tem que saber quais são as tabelas base para depois determinar o caso. Por exemplo, procurar se existe ou não relação 1 a N direta ou indireta.



E para determinar a tabela base de um for each aninhado em um nível N de aninhamento?

Como parecerá lógico, pode utilizar atributos das estendidas de seus ascendentes, não apenas de seu pai. São feitas inferências avô-filho, não apenas pai-filho. Na verdade, são feitas inferências com qualquer ascendente.

Por outro lado, os cortes de controle são unicamente com o nível pai. O que pode acontecer é que o nível pai seja, por sua vez, corte de controle com seu próprio pai.

Com isto damos por concluído o tema.

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