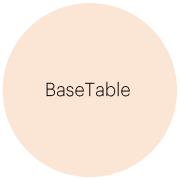
Lógica de consulta a la base de datos con GeneXus

Determinación de tablas base

GeneXus

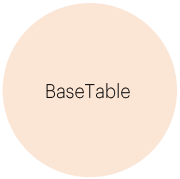



```
For each BaseTrn1, ... , BaseTrnn
    skip expression1 count expression2
    order att1, att2, ... , attn [when condition]
    order att1, att2, ... , attn [when condition]
    order none [when condition]
    unique att1, att2, ... , attn
    using DataSelector ( parm1, parm2, ... , parmn )
    where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

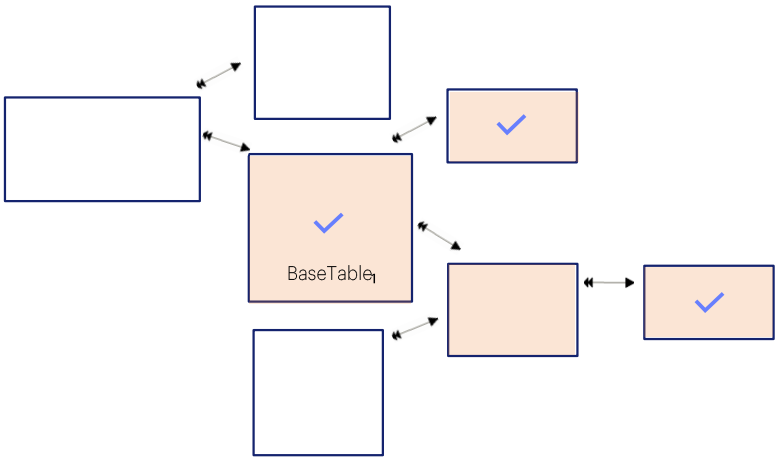
extended table

Nos concentraremos en el comando for each. Si éste tiene transacción base definida entonces el problema ya está resuelto. La pregunta es qué sucede cuando no.

¿Cómo se encuentra la tabla base? GeneXus busca los atributos presentes en todos estos lugares y a partir de ellos busca una tabla extendida que los contenga a todos.

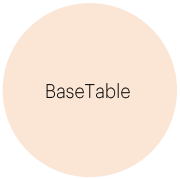



```
For each BaseTrn1, ... , BaseTrnn
    skip expression1 count expression2
    order att1, att2, ... , attn [when condition]
    order att1, att2, ... , attn [when condition]
    order none [when condition]
    unique att1, att2, ... , attn
    using DataSelector ( parm1, parm2, ... , parmn )
    where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

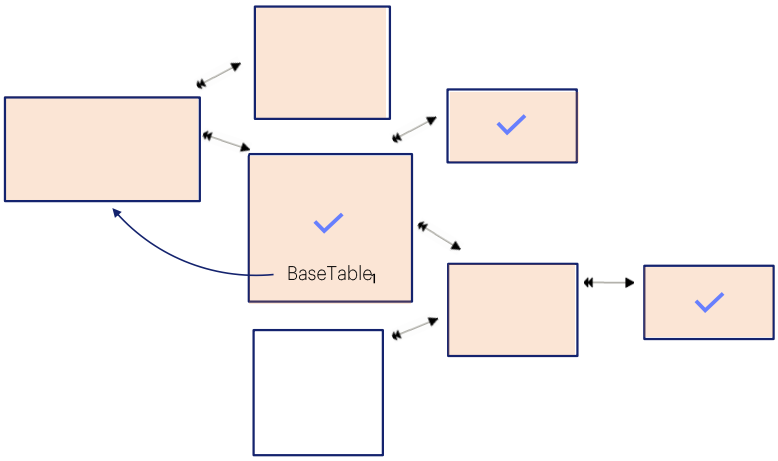


Por ejemplo, si los atributos que aparecen en estos lugares son de estas tablas, ¿cuál será la tabla base del for each?

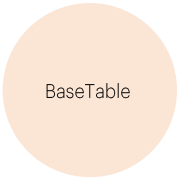
Será esta, porque su tabla extendida los contiene a todos...



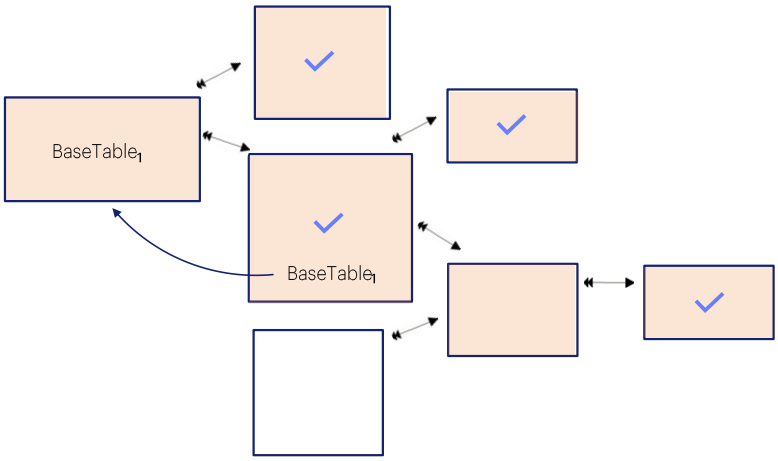

```
For each BaseTrn1, ... , BaseTrnn
    skip expression1 count expression2
    order att1, att2, ... , attn [when condition]
    order att1, att2, ... , attn [when condition]
    order none [when condition]
    unique att1, att2, ... , attn
    using DataSelector ( parm1, parm2, ... , parmn )
    where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



Podríamos objetar que si eligiera a esta otra como tabla base también se cumpliría que su extendida contiene a todos los atributos, así que es preciso agregar una condición: que de todas las tablas extendidas que contengan a **todos** los atributos sea la mínima. Allí queda zanjado el problema; ahora sí 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
```



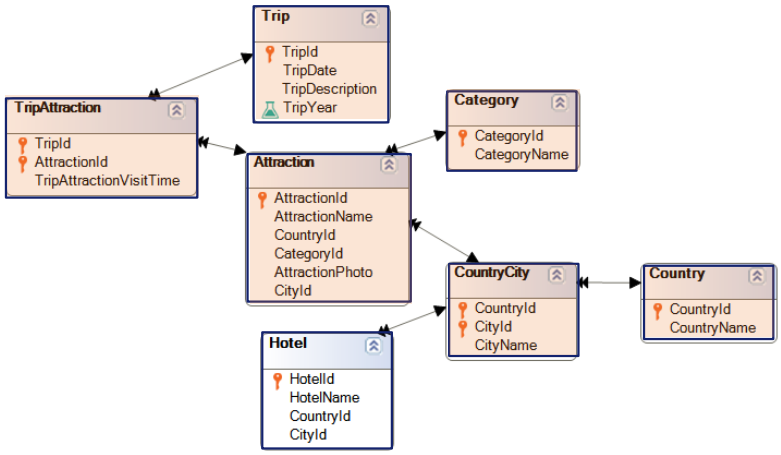
Por supuesto, si apareciera un atributo de esta tabla, la tabla base pasaría a ser esta otra.

Pensémoslo en este ejemplo...

BaseTable

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

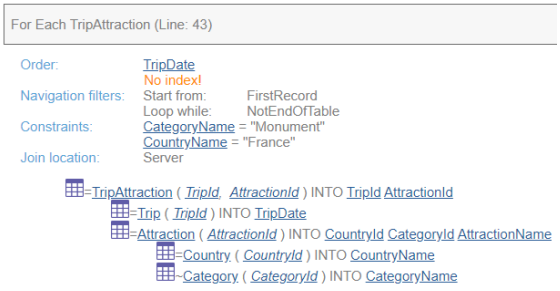
```
For each BaseTrn1, ... , BaseTrnn
    skip expression1 count expression2
    order att1, att2, ... , attn [when condition]
    order att1, att2, ... , attn [when condition]
    order none [when condition]
    unique att1, att2, ... , attn
    using DataSelector ( parm1, parm2, ... , parmn )
    where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

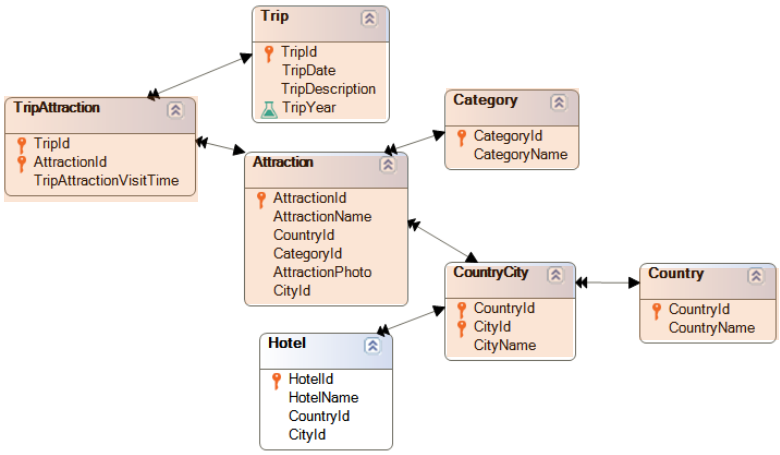


No tenemos transacción base. Este atributo está aquí, este otro aquí, este otro aquí, y en el código principal aparece este otro. El listado de navegación mostrará...

BaseTable

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

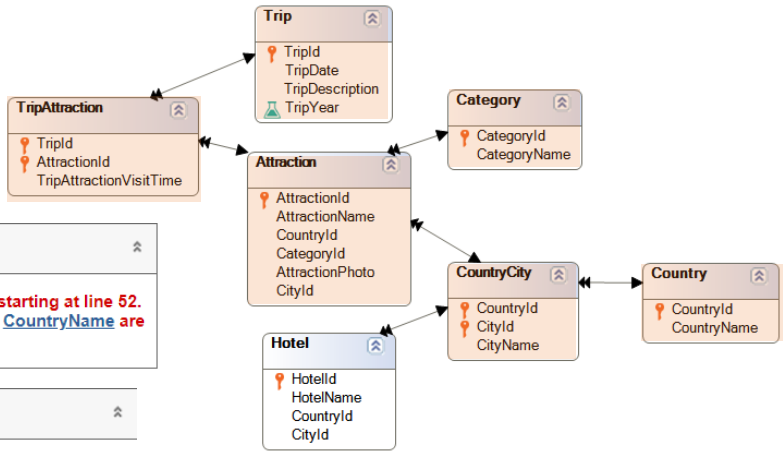


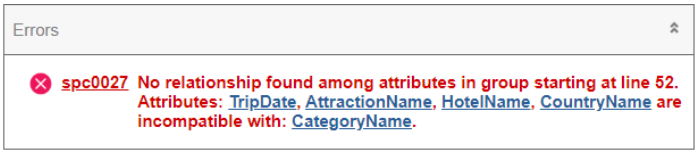


Vemos que está eligiendo, efectivamente, a TripAttraction como tabla base.

BaseTable

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





Ahora, ¿qué pasa si no existe ninguna tabla extendida que los contenga a todos?

Por ejemplo, si agregamos en el printblock a HotelName.

GeneXus arrojará un error y no se podrá generar el objeto.

Observemos que si hubiéramos especificado transacción base, por ejemplo TripAttraction, entonces en lugar de un error veremos que se nos advierte que el atributo HotelName no será alcanzable, pero la tabla base estará perfectamente determinada por la transacción base.

BaseTable

For each

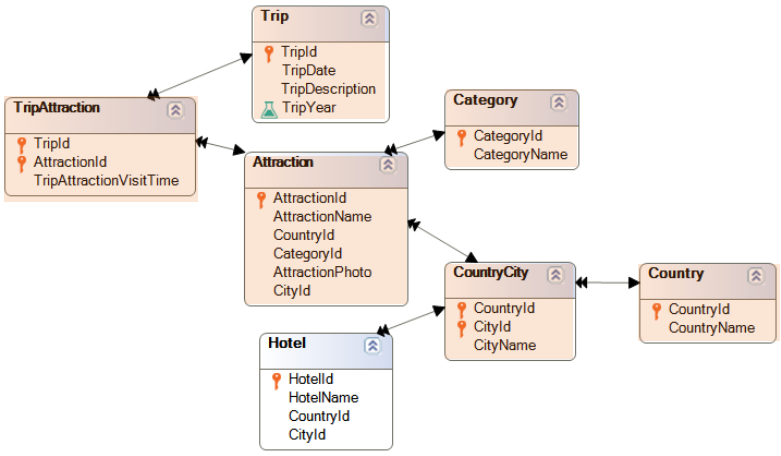
Grid with base table
(with base table)

Data Provider Group
(with base table)

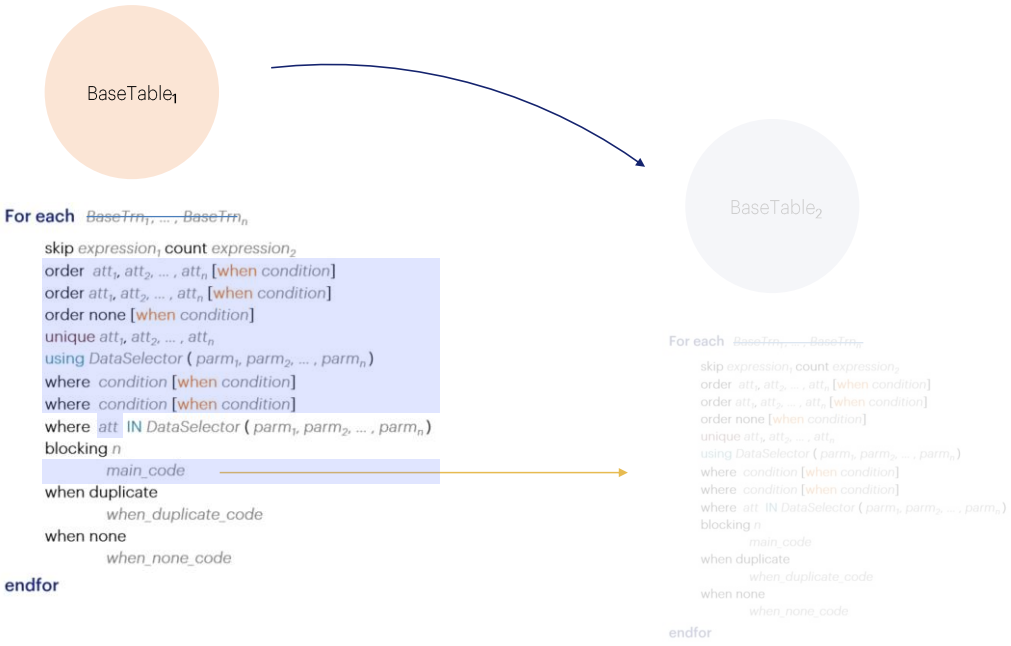
Data Selector

Formula

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



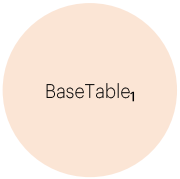
Todo esto que vimos para un For each aplica también a un Grid (con tabla base), o a un grupo de Data Provider. Y lo mismo para un Data Selector ejecutado como consulta independiente, o una fórmula.



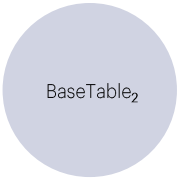
Ahora, ¿qué sucede cuando existe un For each anidado?

La tabla base del for each principal se determina sin considerar en absoluto al For each anidado. Es decir, como si éste no existiera dentro del código principal.

¿Y cómo se determina la tabla base del For Each anidado? Esta siempre se determina luego de determinarse la del for each que lo contiene. El caso a analizar es cuando no hay transacción base.



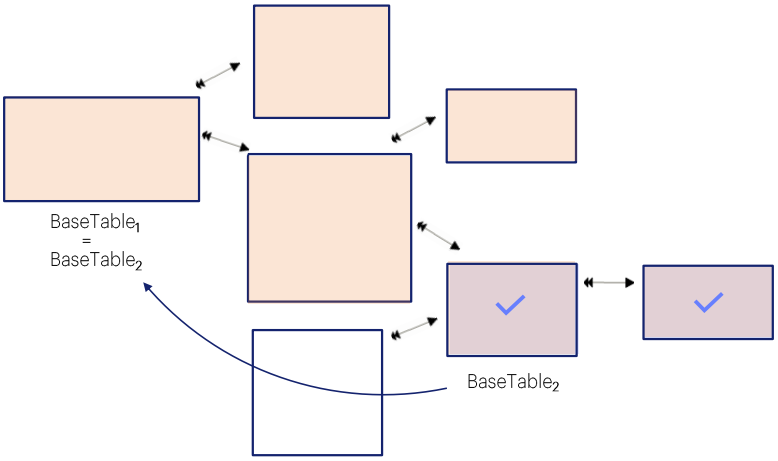

```
For each BaseTrn1, ... , BaseTrnn
    skip expression1 count expression2
    order att1, att2, ... , attn [when condition]
    order att1, att2, ... , attn [when condition]
    order none [when condition]
    unique att1, att2, ... , attn
    using DataSelector ( parm1, parm2, ... , parmn )
    where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

Podría pensarse que de manera idéntica. Considerando los atributos que aquí se encuentren...

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

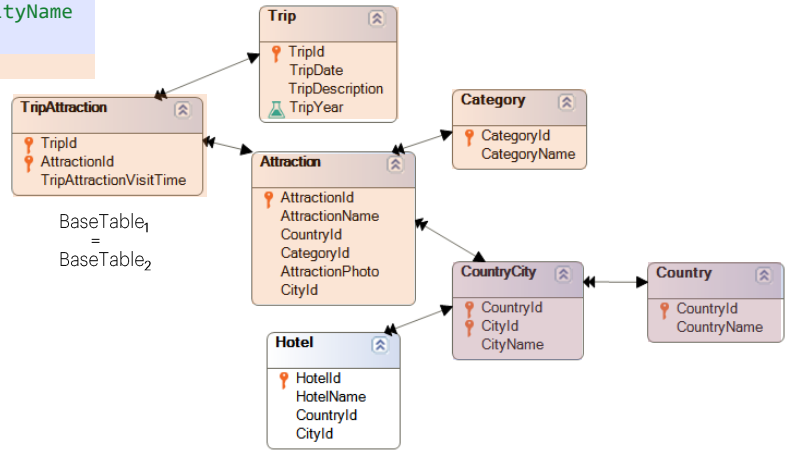


(supongamos estos)… y encontrando la mínima tabla extendida que los contenga, en forma independiente del for each principal.

Sin embargo no será exactamente así. De hecho de todos los atributos extraídos de los lugares consabidos, primero se quitan todos los que también formen parte de la extendida del for each principal, y se opera solo con los que quedan para encontrar la mínima tabla extendida que los contenga.

En este caso si quitamos del conjunto de atributos a calcular los que pertenecen a la tabla extendida del principal nos quedamos con… ¡ninguno! Conjunto vacío. En ese caso la tabla base del for each anidado pasa a ser la misma que la del for each principal. Y se tratará, entonces, de un corte de control.

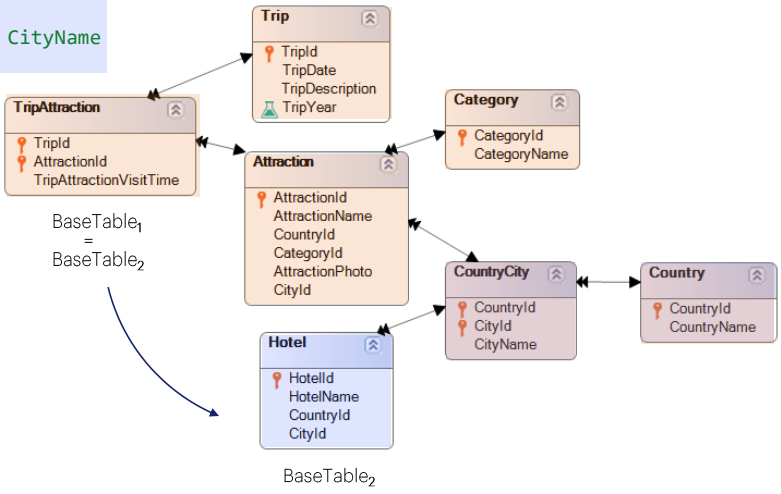
```
for each
   order CategoryName
   where TripDate > &today
   where AttractionName > 'N'
      print PB1 //CategoryName
      for each
         print PB2 //CountryName, CityName
      endfor
endfor
```



Aquí vemos ejemplo concreto. Ya sea que se especifique para el for each principal transacción base Trip.Attraction o que no, en cualquier caso, por los atributos involucrados la tabla base será TripAttraction. Y para el for each anidado, dado que los atributos que se utilizan en él son de esta tabla y de esta otra, su tabla base sería esta... pero resulta que se encuentra en la extendida del principal, por lo que GeneXus elegirá como su tabla base la misma, TripAttraction.

Y se realizará un corte de control por CategoryName.

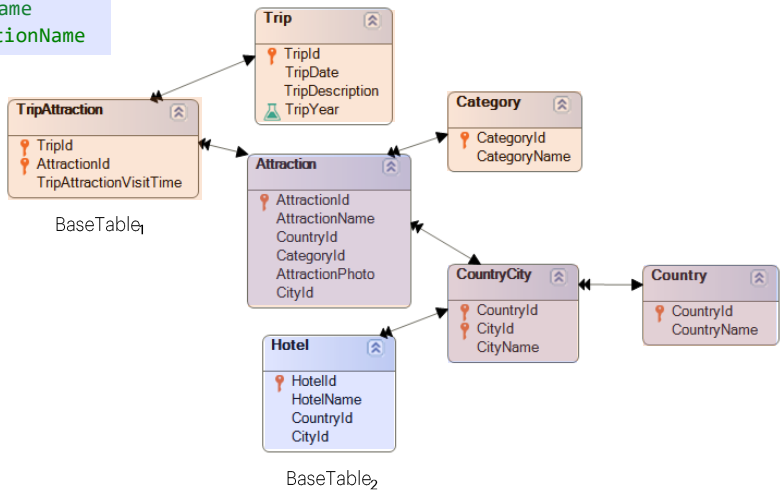
```
for each
   order CategoryName
   where TripDate > &today
   where AttractionName > 'N'
      print PB1 //CategoryName
      for each
         order HotelName
            print PB2 //CountryName, CityName
      endfor
endfor
```



Si se agregara, por ejemplo, un order por HotelName, entonces allí las cosas cambian, y la tabla base del for each anidado será Hotel.

Observemos que aquí de estos tres atributos del For each, tendremos que quitar este y este para el cálculo de la tabla base, porque están ya en la extendida de TripAttraction. Solo nos queda HotelName para calcular la mínima tabla extendida que lo contiene.

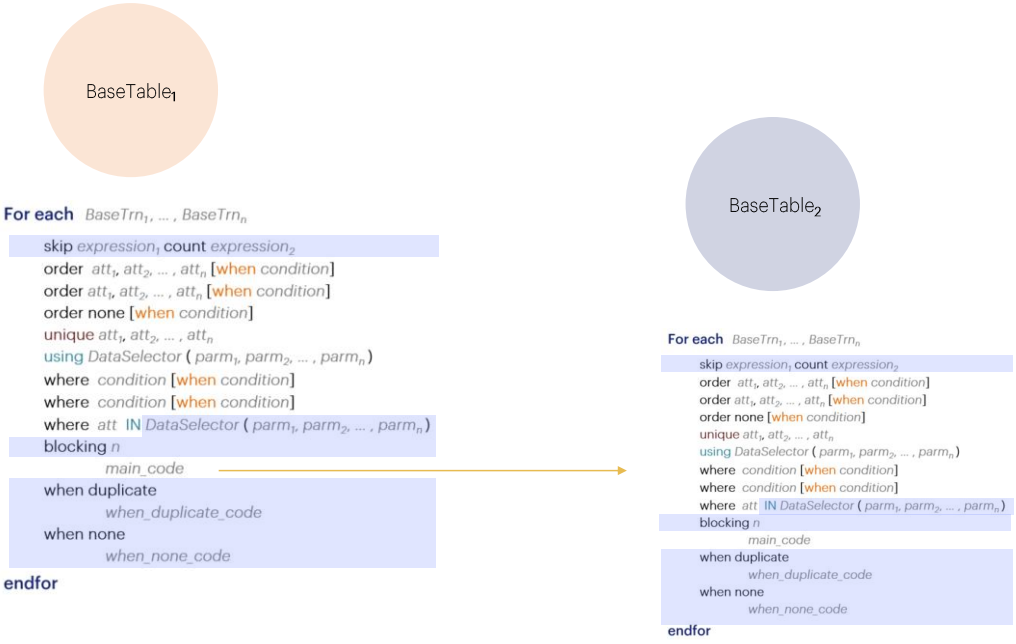
```
for each
   order CategoryName
   where TripDate > &today
   where AttractionName > 'N'
      print PB1 //CategoryName
      for each
         order HotelName
            print PB2 //CountryName, CityName
      endfor          // HotelName, AttractionName
endfor
```



Esto es así porque podríamos eventualmente en el for each anidado utilizar atributos de la tabla extendida del principal, para hacer algo con los valores perfectamente determinados de esos atributos.

Por ejemplo, imaginemos que en el print block del for each anidado agregamos HotelName, que no hace ninguna diferencia, pero además AttractionName. Para la determinación de la tabla base del anidado no participarán ni CountryName, ni CityName ni AttractionName, por estar en la extendida del principal. Claramente la tabla base será Hotel ¿Qué se ejecutará? Para cada TripAttraction que pase los filtros habrá un valor de AttractionName y ese valor es el que se mostrará para cada registro del for each anidado.

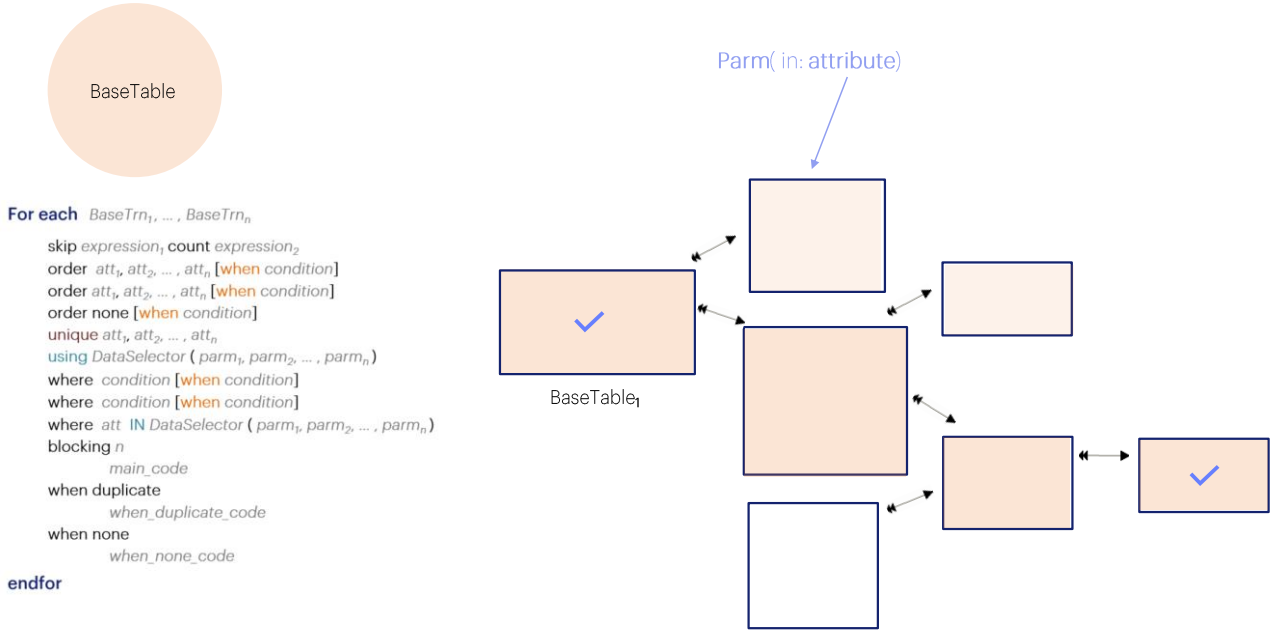
Así para todos los hoteles de la ciudad de la atracción del trip, saldrán ese país y ciudad junto con el nombre del hotel y el nombre de esa atracción, siempre la misma para todos estos registros.



Es importante tener presente que los atributos que se encuentren en estos lugares no participarán de la determinación de la tabla base de la consulta respectiva.

Por supuesto puede especificarse transacción base para uno sí y el otro no.

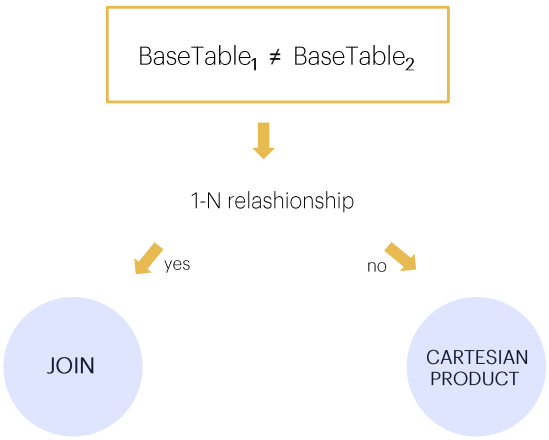
Después de determinarse las tablas base, recién **después**, GeneXus resuelve la navegación. En particular qué condiciones impondrá implícitamente. Por ejemplo...

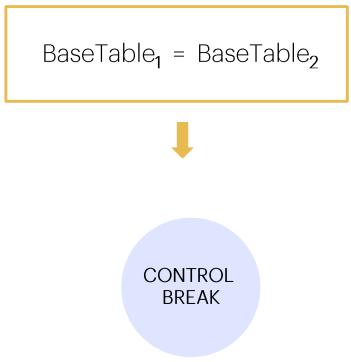



```
For each BaseTrn1, ... , BaseTrnn
    skip expression1 count expression2
    order att1, att2, ... , attn [when condition]
    order att1, att2, ... , attn [when condition]
    order none [when condition]
    unique att1, att2, ... , attn
    using DataSelector ( parm1, parm2, ... , parmn )
    where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

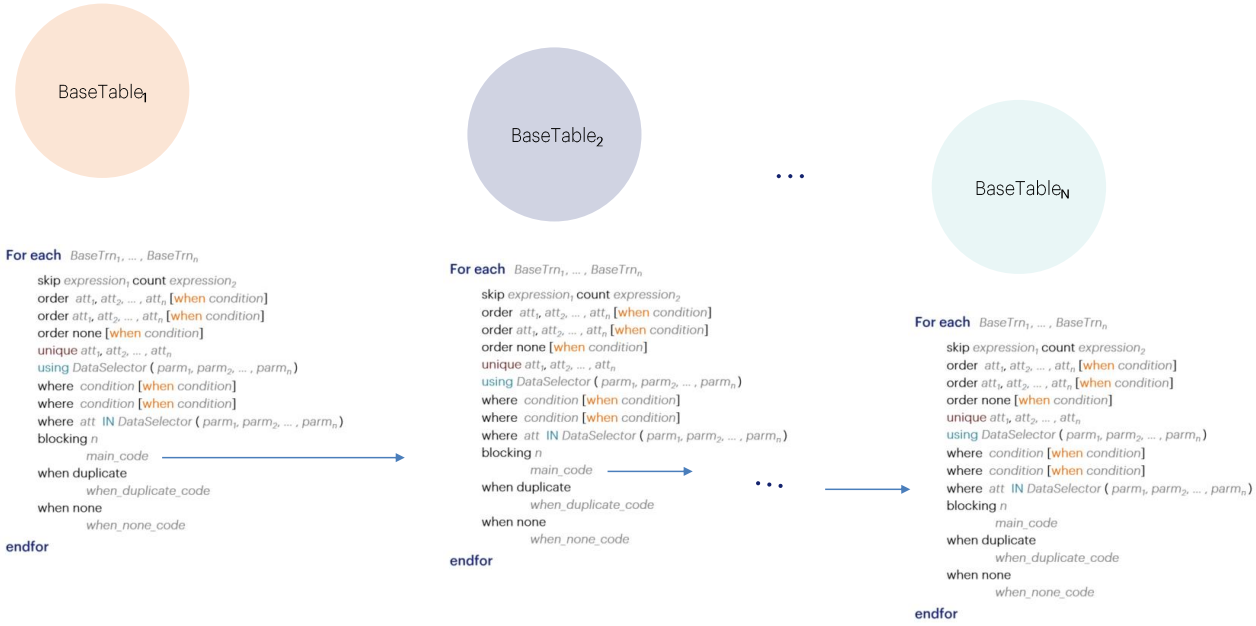
Recordemos que si dentro del for each no se requiere para nada acceder a una tabla particular de la extendida, por ejemplo supongamos que solo se utilizan atributos de estas tablas, entonces solo se accederá a estas de la extendida y no a todas. Estas dos quedarán sin ser accedidas.

Por tanto, si, por ejemplo, recibiéramos en la regla parm en un atributo de una de ellas, ese filtro automático por igualdad no se aplicará al For each. Porque las condiciones implícitas se colocan DESPUÉS que la navegación fue definida.





El otro ejemplo claro: después de determinarse las tablas base, recién después, se define la navegación concreta. Es decir, primero hay que saber cuáles son las tablas base para luego determinar el caso. Por ejemplo, buscar si hay o no relación 1 a N directa o indirecta.



¿Y para determinar la tabla base de un for each anidado en un nivel N de anidamiento?

Como resultará lógico, puede utilizar atributos de las extendidas de sus ancestros, no solo de su padre. Se hacen inferencias abuelo-hijo, no solamente padre-hijo. En realidad se hacen inferencias con cualquier ancestro.

Por otro lado los cortes de control son únicamente con el nivel padre. Lo que puede ocurrir es que el nivel padre sea a su vez corte de control con su propio padre.

Con esto damos por concluido el tema.

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