Más sobre For eachs anidados

Casos y navegación

GeneXus

Lo primero que debe hacer GeneXus ante un caso de for eachs anidados es determinar las tablas base de cada uno.

Cuando el desarrollador indica Transacción Base, esto es inmediato.
Cuando no lo hace, GeneXus debe determinar cada tabla base a partir de los atributos presentes en cada for each. Este caso, más complejo, no será abordado en este video.

Luego, a partir de allí, define la navegación para resolver la consulta múltiple. Se dará uno de tres casos:

-Join
-Producto Cartesiano
-Corte de control

For each anidados

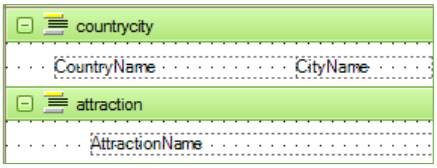
```
For each Attraction order AttractionName
      Where CountryId = 1
      &attractionName = AttractionName
      &attractionDescription = AttractionDescription

      For each Categories
            &categoryName = CategoryName
      Endfor

      When none
      ....
EndFor
```

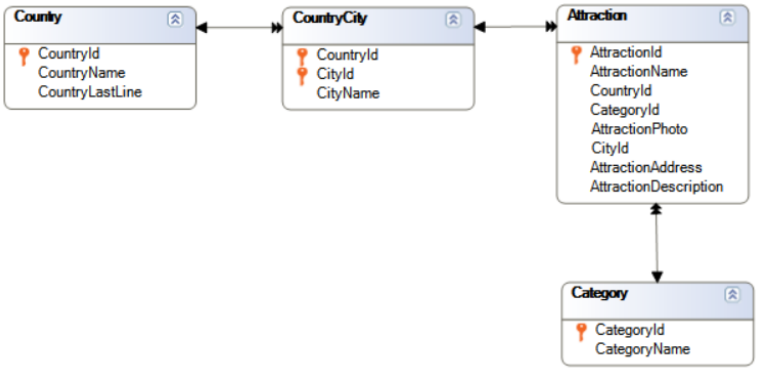
Como decíamos, lo primero que hace GeneXus al encontrar un par de for eachs anidados es determinar la tabla base de cada uno, en forma ordenada, de afuera hacia adentro, empezando por el más externo. Recién luego determina la navegación.

Por cada for each interviene la transacción base indicada, y únicamente los atributos propios de ese for each: tanto del orden, where, etc., así como los que estén en su cuerpo, exceptuando los que se encuentren dentro de un for each anidado. Es decir, quitando el for each anidado, se determina la tabla base como en el caso de un for each simple. No se toman en cuenta nunca los atributos de la cláusula When none. Todos los atributos deben pertenecer a la tabla extendida de la tabla base encontrada, la cual coincide con la tabla base asociada a la transacción base. Los atributos que no cumplan esto, no serán "instanciables", pues no se puede llegar a ellos.

For each anidados

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





En el ejemplo, se procede en este orden:

1- Se determina la tabla base del for each externo. Para eso se considera la transacción base indicada, o sea Country.City, y se verifica que los atributos presentes en el printblock (CountryName y CityName) pertenezcan a su tabla extendida. Si esto no es así, se produce una advertencia en el listado de navegación, que informará que algunos atributos no podrán instanciarse, pues no puede llegarse a ellos desde la tabla extendida de ese for each. En este caso CountryName y CityName pertenecen a la extendida de CountryCity, tabla base del for each.

2- Se determina la tabla base del for each anidado. Se considera la transacción base Attraction, y el atributo AttractionName presente en el printblock. No se toman en cuenta los atributos del for each exterior. Si no se hubiera escrito transacción base, entonces sí se consideraría algo relativo a los atributos del for each externo para determinar la tabla base del interno, pero no es el caso. Entonces se determina su tabla base como si fuera un for each independiente. Por tanto, su tabla base será Attraction.

For each anidados

Diferente tabla base

For each externo ⟷ For each anidado

Join

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

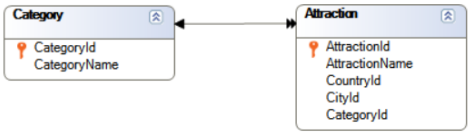


Tabla Category

CategoryId	CategoryName
1	Museum
2	Monument
3	Tourist site

Tabla 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

De la determinación de las tablas base, surgen los tres casos de For eachs anidados que ya hemos visto antes, y que aquí queremos conceptualizar.

Cuando las tablas base son distintas se abren dos posibilidades: o existe relación 1 a N directa o indirecta entre ellas, o no existe. En el primer caso, por cada registro del for each principal, el for each anidado ejecutará sus instrucciones solamente para los N registros relacionados. A esta operación de cortar la información de una tabla por la de otra, se la conoce como Join.

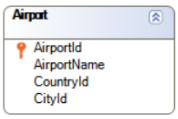
For each anidados

Diferente tabla base

For each externo — For each anidado

Producto cartesiano

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



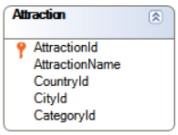


Tabla Airport

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

Tabla 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

En el segundo caso, cuando no existe relación, por cada registro considerado en el for each principal, el for each anidado ejecutará sus instrucciones para todos los registros de la otra tabla, dado que no encontró relación entre ellas. La operación se conoce como Producto Cartesiano.

For each anidados

Misma tabla base

Corte de control

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

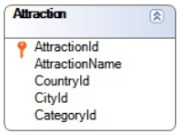


Tabla Attraction

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

Cuando las tablas base son la misma se tratará de un caso conocido como Corte de control: es cuando necesitamos agrupar la información de una tabla, ejecutar ciertas instrucciones que tienen en cuenta la información común del grupo y luego recorrer cada miembro del mismo, y ejecutar otras instrucciones, para, a continuación, pasar al siguiente grupo y repetir el proceso. En este caso es fundamental especificar los atributos que conforman el grupo, mediante la cláusula order.

Este caso se da porque se especificó la misma transacción base para el for each externo y para el anidado.
Si no se hubiesen especificado las transacciones bases de los for eachs, como dijimos, las calculará GeneXus a partir de los atributos que encuentre dentro de esos for eachs. Pero, solo inferirá que se quiere implementar un corte de control si las tablas bases encontradas por él son la misma. No entraremos en detalles en este curso.

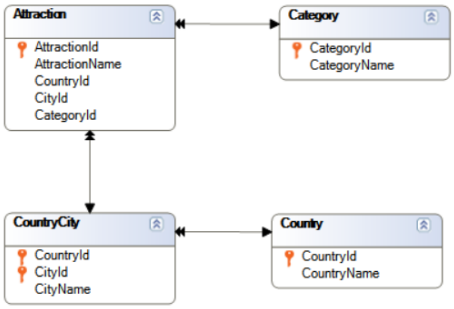
Veamos este otro caso.

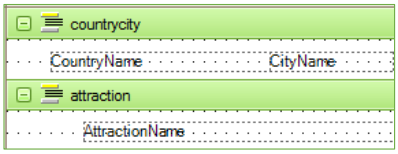
For each anidados

Diferente tabla base

For each externo ⟷ For each anidado

Corte de control






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

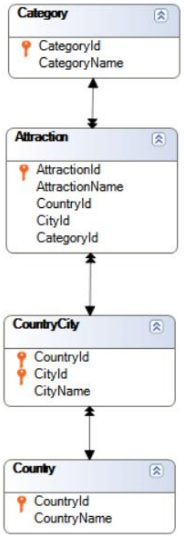
```
For each Attraction
    print attraction
    print countrycity
endfor
```

Observemos que aquí el segundo for each sería innecesario, porque por cada atracción en la que se esté posicionado en un momento dado en el for each externo, únicamente se tiene un registro de CountryCity relacionado. Sería lo mismo, entonces, no haber escrito el segundo for each, y directamente enviar a imprimir el printblock countrycity, que contiene CountryName y CityName, ambos pertenecientes a la tabla extendida de Attraction.

For each anidados

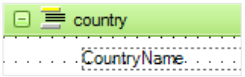
1

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



2

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



Analicemos ahora estos dos casos de relación 1 a N entre los for eachs, y por tanto, de JOIN.

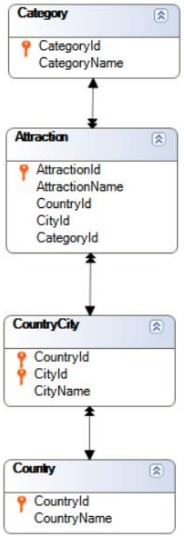
La primera es directa. Observemos que las tablas base del for each externo y anidado son CountryCity y Attraction, respectivamente, que están relacionadas por una relación 1 a N.

Se van a imprimir el nombre de país y ciudad, y por cada dupla, sus nombres de atracción

La segunda es indirecta. Las tablas base del for each externo y anidado son Country y Attraction. Si observamos bien cada país tiene N ciudades, a cada una de las cuales le corresponden N atracciones. Las tablas base de los for eachs no tienen una relación directa 1 a N, pero sí indirecta, a través de la tabla CountryCity. Dicho de otro modo, observemos que la tabla base del primer for each, Country, está incluida en la tabla extendida de la tabla base del for each anidado: Attraction. Entonces se realizará un join

For each anidados

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




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

Veamos los listados de navegación. Para el for each anidado, que en ambos casos navega Attraction, no se recorre toda la tabla. Observemos que en el primer caso, donde la relación es directa, vemos con el arroba que se va a filtrar por la clave foránea compuesta que es la que establece la relación, CountryId y CityId. En el segundo caso, vemos que será solamente por CountryId, que es parte de la clave foránea compuesta.

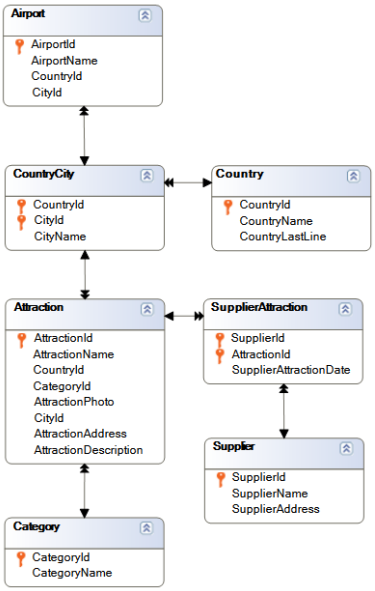
Obsérvese que en ambos casos, en lugar de ordenar la recorrida del for each anidado por la clave primaria de Attraction, que es AttractionId, lo hace por el atributo o conjunto de atributos relación, para los que cuenta con índice automáticamente creado, por clave foránea. De esta manera, el acceso a la base de datos estará optimizado.

Por tanto, cuando GeneXus determina que realizará un Join, intenta optimizar su navegación.

For each anidados

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

Join



Aquí vemos un tercer ejemplo. En este caso se han agregado proveedores los cuales ofrecen atracciones turísticas, y por ello aparecen dos tablas más: Supplier y su subordinada, SupplierAttraction.
Además tenemos los aeropuertos, que pertenecen a un país y ciudad.
Entonces si observamos el for each, tenemos que la tabla base del for each externo es SupplierAttraction y la tabla base del for each anidado es Airport. Supongamos para este ejemplo que en el printblock info se muestra el nombre del proveedor, el nombre de la atracción y la fecha en la que se mostrará esa atracción, y que en el segundo printblock se muestra el nombre del aeropuerto.

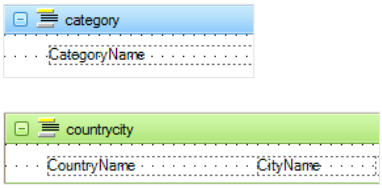
Observemos que existe una relación 1 a N indirecta. Es decir, por cada registro de SupplierAttraction, existirá uno solo de Attraction a partir del que obtenemos uno solo de CountryCity, que a su vez está relacionado con N registros de Airport (los N aeropuertos que se encuentran en ese país/ciudad, aunque en general, en la realidad, haya uno solo. En nuestro modelo puede haber varios).

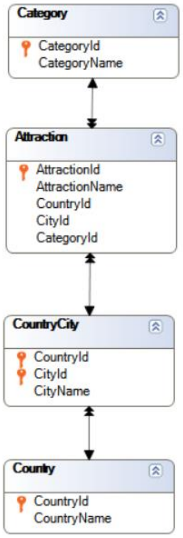
Hilando fino, aquí GeneXus encuentra que hay atributos en común entre la tabla extendida del for each principal y la tabla base del anidado. ¿Cuáles? El par {CountryId, CityId}. Y por ellos realizará el join.

Para optimizar vemos que está eligiendo el índice por clave foránea de la tabla Airport. Resumiendo, este es un caso en el que la relación entre las tablas base no es 1 a N directa, sino indirecta, a través de tablas intermediarias.

For each anidados

```
For each Category
    Print

    For each Country.City
        Print
    Endfor
Endfor
```





Producto Cartesiano

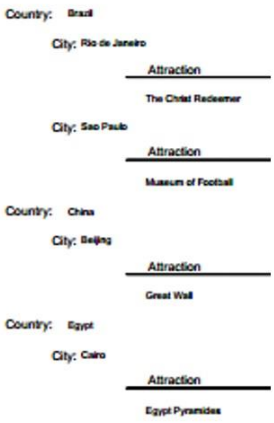
Veamos ahora este caso de for each anidados, con tablas base distintas y no relacionadas.

Aquí el primer for each navega Category, mientras que el anidado navega CountryCity. En este caso no hay una relación 1 a N de ningún tipo, contra lo que podría parecer. Si observamos el diagrama de tablas, vemos que una categoría tiene N atracciones, cada una de las cuales tiene un y solo un CountryCity. Uno podría pensar que entonces se listará para cada categoría, el país ciudad de cada una de sus atracciones. Eso sería así si la tabla base del for each anidado fuera Attraction. Pero no lo es, es CountryCity. Por lo que GeneXus entiende que no estamos buscando los registros relacionados por attraction. Si estuviéramos buscando esos, alcanzaría con especificar como transacción base a Attraction para el segundo for each. Este puede resultar un caso engañoso, por lo que es fundamental leer bien el listado de navegación para no llamarse a confusión. En el listado vemos claramente que no se está realizando ningún join.

En este caso GeneXus no logra encontrar una relación 1-N directa o indirecta entre las tablas y por lo tanto no aplica filtros implícitos a los registros del For each anidado. Es decir, realiza un producto cartesiano entre las tablas: para cada registro de la tabla base del for each externo (Category), considera todos los registros de la tabla base del anidado (CountryCity).

For each anidados

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

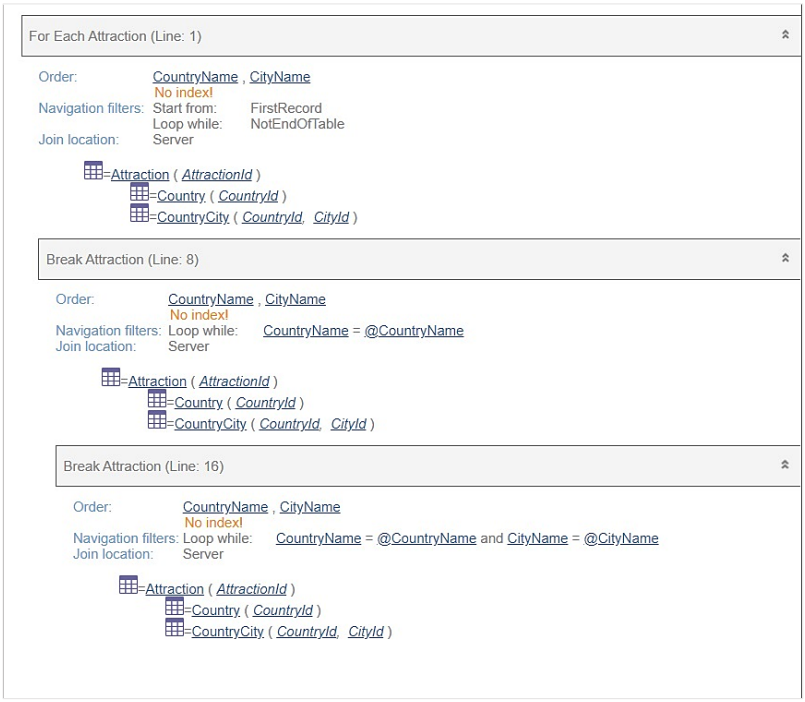


Aquí vemos un caso en que estamos queriendo listar cada país, y para él cada ciudad, y para ella cada atracción. La restricción: queremos hacerlo sólo para los países y ciudades para los que existen atracciones turísticas.

Es decir, tendremos que implementar un corte de control doble: donde primero agrupemos por país, y dentro de ese grupo, agrupemos luego por ciudad, y dentro de este último, mostremos los nombres de todas las atracciones. Para ello:

Tendremos que marcar los criterios de agrupamiento utilizando las cláusulas order. Recordemos que para un corte de control, el order tiene un peso muy fuerte: no sólo está marcando por qué atributo o atributos listar la información, sino que está especificando cómo ésta se va a agrupar.

Podríamos especificar un order para el for each más interno, pero ese order sí tendrá únicamente su uso convencional. Es decir, ese sí será utilizado únicamente para ordenar. Lo veremos en un momento.



Corte de control

Corte de control

Tenemos un doble corte de control, lo que implica tres for eachs.

Si observamos el listado de navegación, vemos la palabra Break para cada for each interno, indicando la misma tabla base, Attraction, y por tanto, un corte de control.

Además, recorrerá esa tabla base una única vez, para lo cuál necesita ordenar por la concatenación de los atributos que aparezcan en los orders de los for eachs. Es por eso que elige CountryName, CityName.

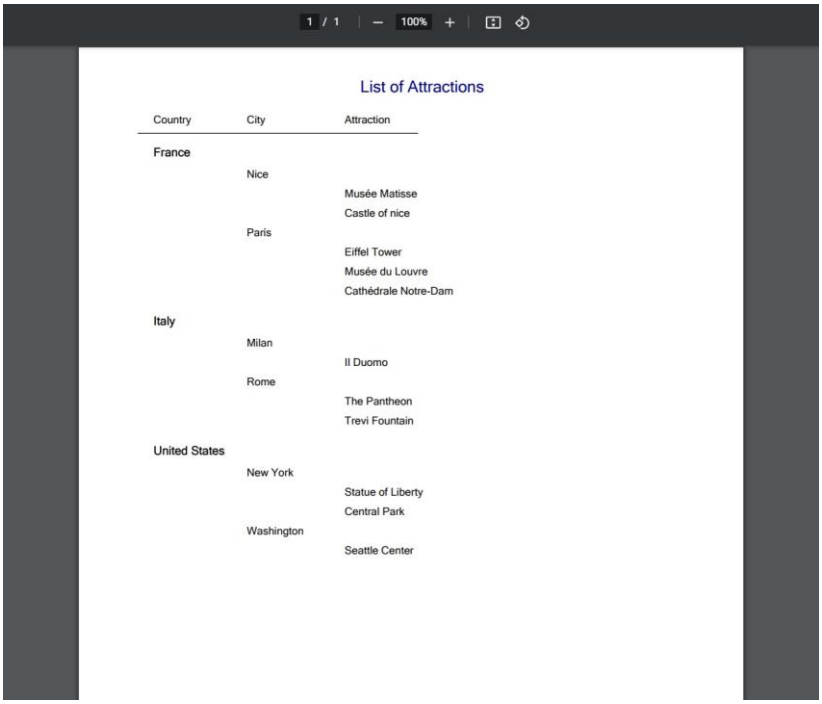
Obsérvese que en el segundo for each corta por país, iterando sobre el país en el que se encuentra posicionado en el primer for each. Y el tercer for each corta por país y ciudad, iterando sobre la ciudad en la que se encuentra posicionado en el segundo for each.

Aquí vemos un ejemplo de cómo se mostraría si estos fueran los datos. Primero vemos el país, y para él vemos la primera de sus ciudades de acuerdo a su nombre, y las atracciones turísticas de ese país y ciudad. Luego vemos la siguiente ciudad del mismo país, y sus atracciones, y recién cuando no quedan más ciudades de ese país, se muestra el siguiente, alfabéticamente y se vuelve a empezar. País, ciudad,

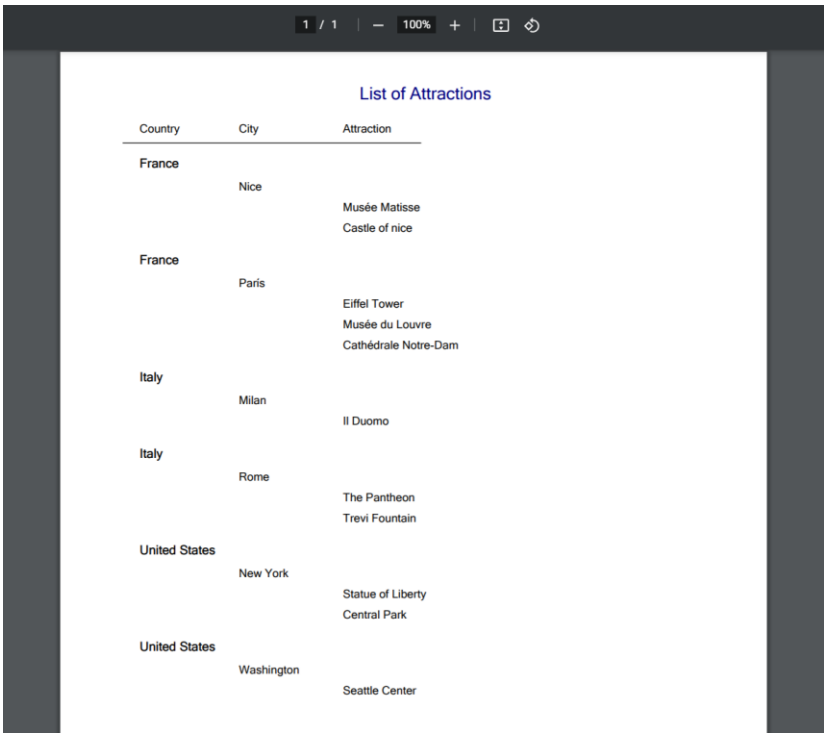
atracciones.

Piense cuál será la ejecución del listado anterior si en lugar de haber ordenado el primer for each por CountryName y el segundo por CityName, hubiéramos ordenado los dos o solo el primero por el par CountryName, CityName.

Observe que en ese caso el listado de navegación diferirá de este que ve arriba, en el segundo for each. Loop while allí dirá "CountryName = @CountryName and CityName = @CityName".



Esto significa que en ejecución, ahora la misma información, se verá de este otro modo. Es decir, sale el país... su primera ciudad... y sus atracciones turísticas. Luego, el mismo país y su segunda ciudad... y sus atracciones turísticas, y así sucesivamente hasta que cambia el país y vuelve a pasar lo mismo para el siguiente.
Claramente la información se está presentando distinta. Solo por haber colocado CityName en el primer o en el segundo for each.

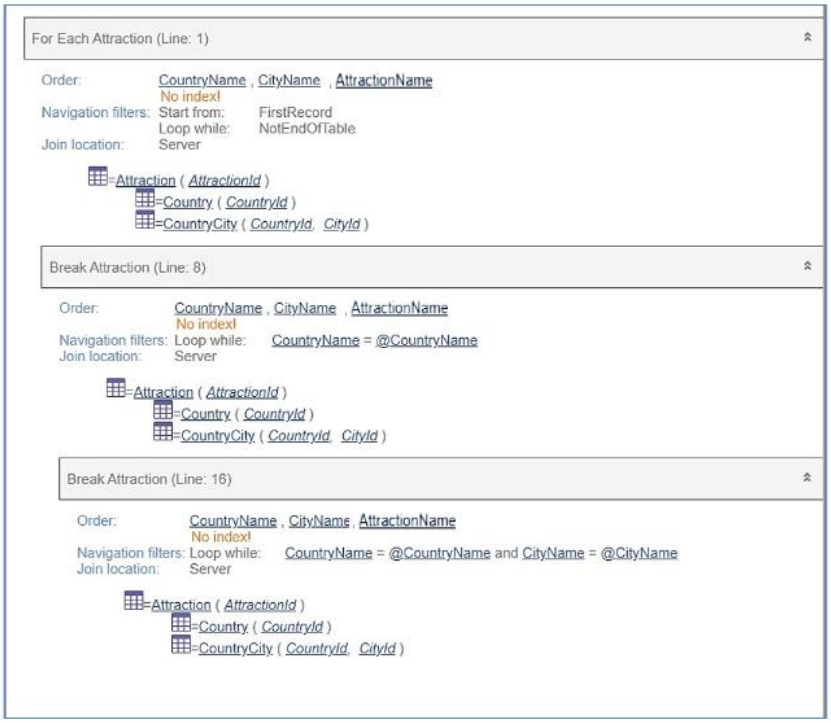


Ahora supongamos que sobre el primer ejemplo, en el que cortábamos por país y luego por ciudad, para luego listar las atracciones, le agregamos al último for each, el más interno, un order por AttractionName. Es decir, querremos que las atracciones de un país y ciudad se ordenen por nombre de atracción.

El listado de navegación cambiará, y se agregará AttractionName al order de los tres for eachs, quedando CountryName, CityName y AttractionName.

Y si ejecutamos con este cambio, vemos exactamente lo que decíamos: se está cortando por nombre de país, por eso sale primero Francia y recién mucho después Italia, y dentro de Francia, se está cortando por CityName o sea por ciudad, por eso sale primero Niza que está alfabéticamente antes que Paris, y dentro de cada uno de esos subgrupos, las atracciones se ordenan alfabéticamente:

Al contrario de lo que sucedía en el primer caso, cuando no teníamos el order por AttractionName en el último for each.

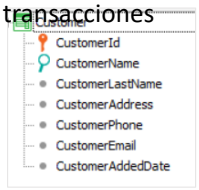


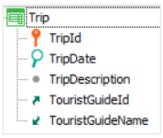
Como vimos, los criterios de corte los vemos en Navigation Filters. Aquí estamos diciendo que ordene la recorrida por el conjunto CountryName, CityName, AttractionName y que ejecute el cuerpo del primer for each, donde solamente se imprime el CountryName, y luego ejecute el segundo for each, en el que, mientras no cambie el CountryName, debe imprimir el printblock que contiene solo a CityName, pero inmediatamente ejecutar el for each interno, donde mientras no cambie el par countryName cityName, debe imprimir el AttractionName.

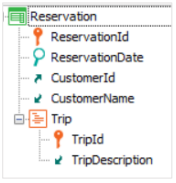
Es decir, observamos que los filtros se mantienen exactamente los mismos que cuando no ordenábamos el for each más interno por AttractionName, por lo que, como decíamos, este order que acabamos de ingresar solo sirve para ordenar, y no para realizar un corte

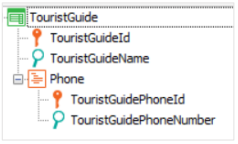
Caso de estudio

Diseño de transacciones









Source

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

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

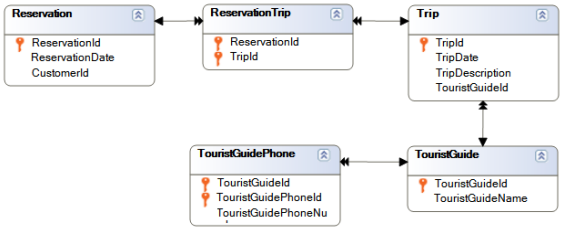
En un video anterior hemos analizado un caso de estudio para poder determinar la forma en que GeneXus aplica los filtros cuando se reciben como atributo en la regla Parm.

Retomando ese mismo caso, veremos la forma de evitar que ese filtro se aplique.

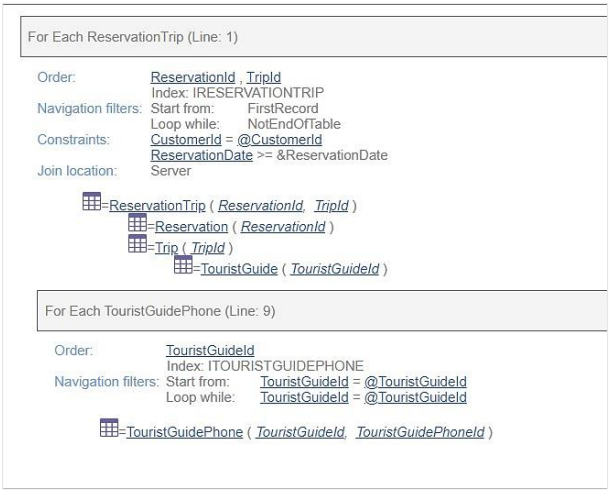
Partímos entonces del diseño de transacciones que vemos, y necesitamos obtener un listado que muestre, para un cliente dado, y a partir de una fecha dada, todas las excursiones que tiene reservadas, y para cada una de ellas los teléfonos de contacto del guía de turismo a cargo.

Para lograrlo se propone el source que se muestra.

Caso de estudio




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

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



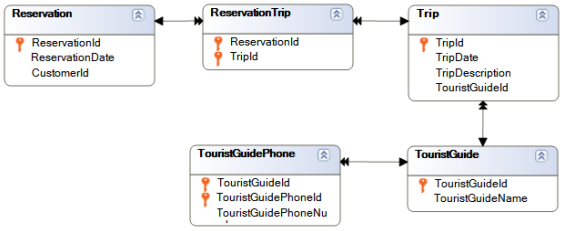
Sabiendo que la table base del For each externo es RESERVATIONTRIP, y que la tabla base del For each interno es TOURISTGUIDEPHONES., ¿establece filtros implícitos para la información que utilizará? Sabemos que sí, que mostrará los teléfonos del guía de cada excursión

.¿Por qué? GeneXus busca si hay relación entre la tabla extendida del for each externo, y la tabla base del for each anidado:

Es una manera de buscar una relación 1 a N, aunque en este caso indirecta. Si cada RESERVATIONTRIP tiene un TouristGuideId, y en la tabla a ser naveg ada también hay un TouristGuideId, entonces GeneXus entiende que por la relación entre la información, se tratará del mismo. Y por eso realiza el join.

Lo vemos claramente en el listado de navegación donde siempre el @ está indicando información contextual (observe el @CustomerId, que refiere al valor recibido en el atributo CustomerId en la regla Parm). Este @TouristGuideId refiere al valor del atributo de ese nombre de la tabla TRIP accedida por el primer for each.

Caso de estudio




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

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

Si esto no fuera lo deseado por el programador, ¿hay manera de evitar esos filtros automáticos?

La respuesta es sí, utilizando una subrutina para encapsular el código de este segundo for each. Al hacer esto, el listado de navegación lo reflejará.

Las subrutinas son bloques de código, que se definen en un objeto mediante el comando Sub. El cual podrá ser llamado luego, dentro del mismo objeto y las veces que queramos, mediante el comando Do.
En próximos videos entraremos en detalles sobre su implementación.

GeneXus™

training.genexus.com
wiki.genexus.com