

More about
Nested For Eachs
Cases and navigation

GeneXus™ 16

Estimated reading time: 20 min

Base Table

Base Tables and Navigation

- ✓ When there are nested For Each commands, GeneXus must first determine the base tables of each one of them.
 - When the developer indicates the Base Transaction, it is done right away.
 - Otherwise, GeneXus has to determine each base table according to the attributes included in every For Each command. This case is more complex and will not be addressed in this course.
- ✓ Next, GeneXus defines the necessary navigation to solve the multiple query. One of three options will be applied:
 - Join
 - Cartesian product
 - Control break

External FE Base Table

For each

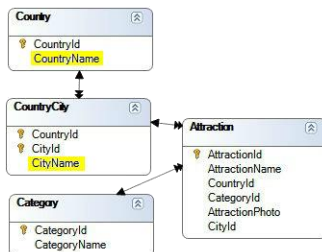


When none

```

...
endifor

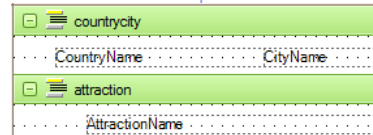
```



```

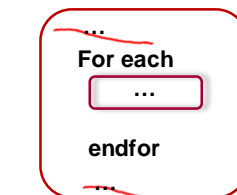
For each Country.City
Print countrycity
For each Attraction
Print attraction
Endfor
Endfor

```



Nested For each Base Table

For each

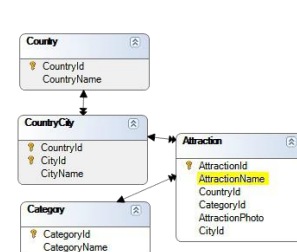


When none

```

...
endifor

```



As we have said, the first thing GeneXus does when it finds a couple of nested For Each commands is determine the base table of each one of them, in an ordered manner and from the outside in, starting from the outermost one. Only then it determines how navigation will take place.

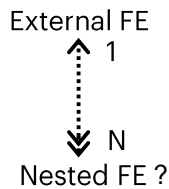
In every For Each command, the indicated base transaction and only the attributes that belong to this For Each command come into play: from the Order, Where, etc. as well as those in its body, except for those included in a nested For Each command. That is to say, removing the nested For Each command, the base table is determined as in a simple For Each command. The attributes of the When none clause are never taken into account. All attributes must belong to the extended table of the base table found. The attributes that don't meet this condition will not be "instantiable" because they can't be reached.

In the example, it is done in this order:

- 1) The **base table of the external For Each command** is determined. To this end, the indicated base transaction is considered; that is to say, Country.City, and it is checked whether the attributes included in the printblock (CountryName and CityName) belong to its extended table. Otherwise, a warning is displayed in the navigation list to inform the user that some attributes cannot be instantiated, because they can't be reached from the extended table of that For Each command. In this case, CountryName and CityName belong to the extended table of CountryCity, the base table of the For Each command.
- 2) The **base table of the nested For Each command** is determined. The Attraction base transaction is considered, as well as the AttractionName attribute included in the printblock. If a base transaction hadn't been written, something related to the external For Each command attributes would be considered to determine the base table of the internal For Each command, but this is not the case. Thus, its base table is determined as if it was a standalone For Each command. Therefore, its base table will be Attraction.

Cases and Navigation

- **Base tables \neq**



Is there an implicit relationship that associates External For Each with N records of the nested For Each?

Yes

Join: some records of the nested FE are retrieved: the related ones. **(Case 1)**

No

Cartesian product: all records of the nested one are retrieved. **(Case 2)**

- **Base tables =**

Control break: When we want to retrieve data in groups. **(Case 3)**

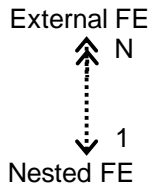
The three examples of nested For Each commands that we've seen before are based on the determination of base tables. Now we will examine them more closely.

When the **base tables** are **different**, there are two possibilities: whether or not there is a **direct or indirect 1 to N relationship** between them. In the first case, for each record of the main For Each, the nested For Each will run its instructions only for the N associated records. The operation that cuts the data from a table by that of another one is known as a **Join**.

In the second case, when there is no relationship, for each record considered in the main For Each, the nested For Each will run its instructions for all the records of the other table because it has found no relationship between them. This operation is known as **Cartesian Product**.

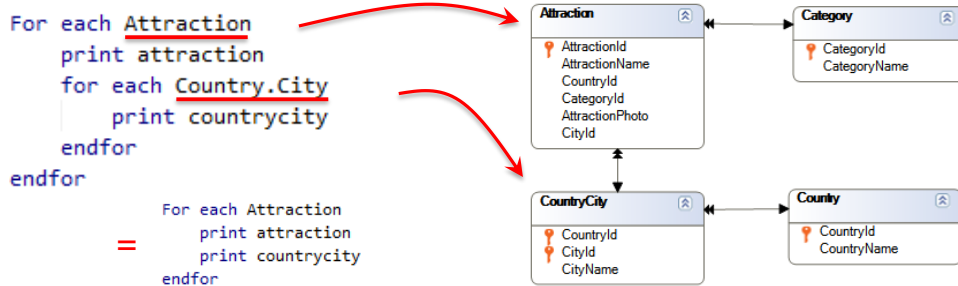
When the **base tables** are **the same**, an operation known as **Control Break** is performed: it takes place when we need to group the data from a table, run certain instructions that consider the group's common data and run through each member, and then run other instructions to move on to the next group and repeat the process. In this case we must indicate the attributes that make up the group using the **order** clause.

- Base tables ≠



This is an unusual, badly programmed case: when the base table of the nested For Each command is part of the extended table of the external For Each command.

(Case 4)

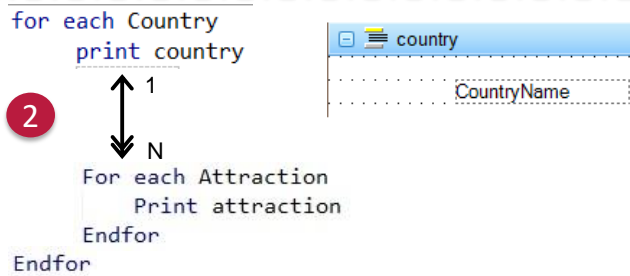
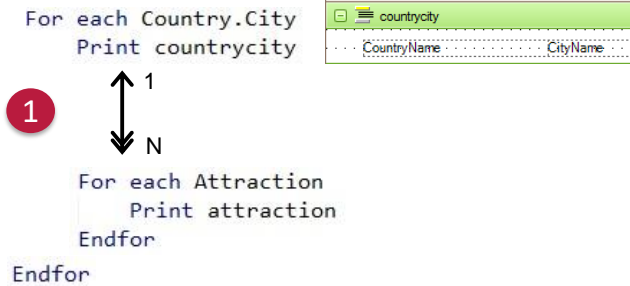


This case only appears when a base transaction is used for the nested For Each command. Otherwise, GeneXus will have to calculate it on its own. In this situation, for the nested For Each command it will choose the same base table as that of the parent, implementing a control break. We won't talk about it in this course.

Note that here the second For Each command would be unnecessary, because for every attraction selected at a given moment in the external For Each command, there is only one related CountryCity record. So, this would be the same as not writing the second For Each command, and sending to print the CountryCity printblock that contains CountryName and CityName, which both belong to the Attraction extended table.

Examples of each case

Base tables ≠



Here we have two examples of 1 to N relationships.

The first one is direct. Note that the base tables of the external and nested For Each are CountryCity and Attraction, respectively; they are related by a 1 to N relationship.

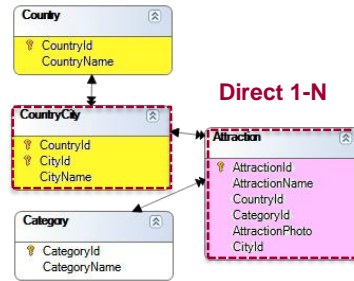
The second one is indirect. The base tables of the external and nested For Each are Country and Attraction, which don't have a direct 1 to N relationship. They do have an indirect relationship through the CountryCity table. In other words: note that the base table of the first For Each (Country) is included in the extended table of the nested For Each base table (Attraction).

1

```

For Each CountryCity (Line: 9)
Order:      CountryId , CityId
Index:      ICOUNTRYCITY
Navigation Start from: FirstRecord
filters:    Loop while: NotEndOfTable
Join location: Server
    =CountryCity ( CountryId, CityId) INTO CityId CountryId CityName
    =Country ( CountryId) INTO CountryName

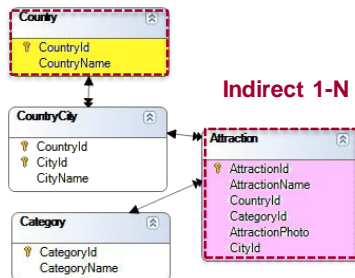
For Each Attraction (Line: 15)
Order:      CountryId , CityId
Index:      IATTRACTION1
Navigation Start from: CountryId = @CountryId
filters:    from:      CityId = @CityId
            CountryId = @CountryId
            Loop while: CityId = @CityId
            =Attraction ( AttractionId) INTO AttractionName
    
```



Direct 1-N

Join

2



Indirect 1-N

```

For Each Country (Line: 33)
Order:      CountryId
Index:      ICOUNTRY
Navigation Start from: FirstRecord
filters:    Loop while: NotEndOfTable
    =Country ( CountryId)

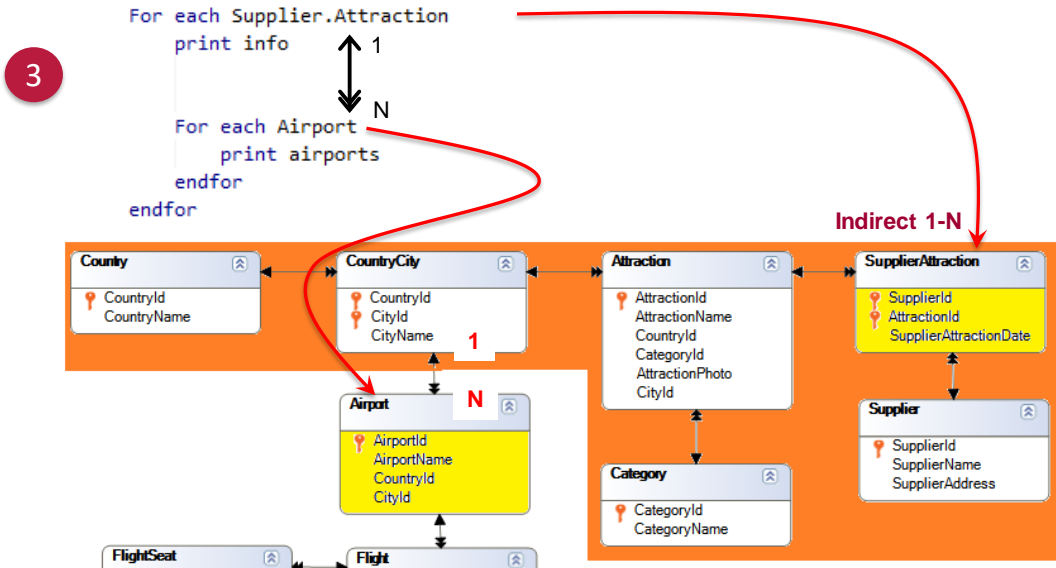
For Each Attraction (Line: 40)
Order:      CountryId
Index:      IATTRACTION1
Navigation Start from: CountryId = @CountryId
filters:    Loop CountryId = @CountryId
            while:
            =Attraction ( AttractionId)
    
```

Navigation lists clearly indicate the Join: the entire table is not run through for the nested For Each. Note that in both cases instead of ordering the navigation by the primary key of Attraction, which is AttractionId, it does so by the relation attribute or set of attributes. To do so, it has an index automatically created by foreign key. In this way, the database access will be optimized.

Therefore, after determining that it will make a Join, GeneXus tries to optimize its navigation.

Base tables ≠

Join



Here we see a third example, where the base table of the external For Each command is SupplierAttraction, and the base table of the nested For Each command is Airport. Note that there is an indirect 1 to N relationship. That is to say, for every SupplierAttraction record, there will be only one Attraction record, given that we obtain only one from CountryCity, which in turn is related to N records of Airport (the N airports found in that country/city, even though, in general, there is actually only one. In this model there may be several).

Base tables ≠

3

```

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

Join

For Each SupplierAttraction (Line: 23)

Order: [SupplierId](#) , [AttractionId](#)
 Index: ISUPPLIERATTRACTION

Navigation Start from: FirstRecord
 filters: Loop while: NotEndOfTable
 Join location: Server

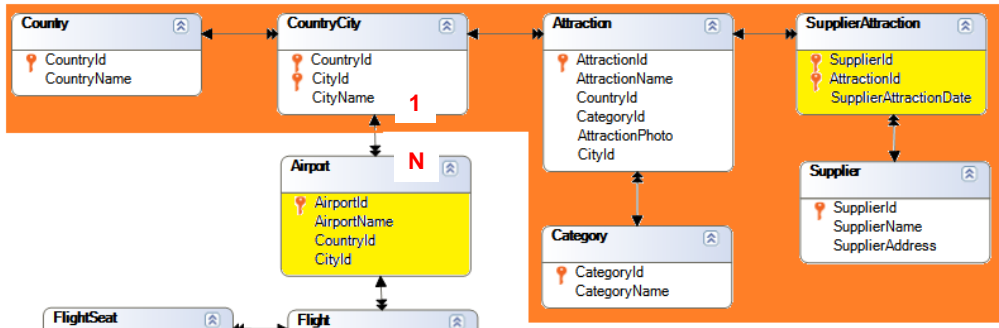
=SupplierAttraction ([SupplierId](#) , [AttractionId](#))
 =Attraction ([AttractionId](#))

For Each Airport (Line: 27)

Order: [CountryId](#) , [CityId](#)
 Index: IAIRPORT1

Navigation Start from: [CountryId = @CountryId](#)
 filters: [CityId = @CityId](#)
 Loop while: [CountryId = @CountryId](#)
 [CityId = @CityId](#)

=Airport ([AirportId](#))



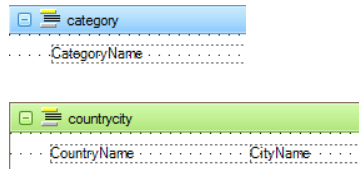
Here GeneXus finds attributes in common between the extended table of the main For Each command and the base table of the nested For Each command. Which ones? The pair {CountryId, CityId}. It will make the Join through them.

Base tables ≠

```

For each Category
  Print
Endfor

For each Country.City
  Print
Endfor
    
```



For Each Category (Line: 23)

Order: [CategoryId](#)
 Index: ICATEGORY

Navigation Start from: FirstRecord
 filters: Loop while: NotEndOfTal

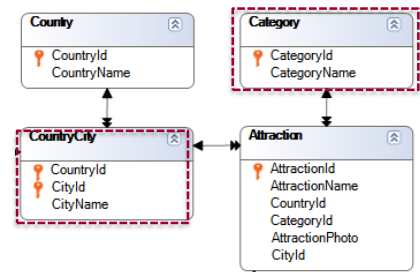
= [Category](#) ([CategoryId](#))

For Each CountryCity (Line: 27)

Order: [CountryId](#) , [CityId](#)
 Index: ICOUNTRYCITY

= [CountryCity](#) ([CountryId](#) , [CityId](#))

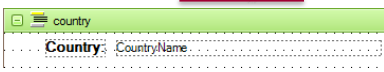
Cartesian Product





In this case, GeneXus can't find a direct or indirect 1-N relationship between the tables; therefore, it doesn't apply implicit filters to the nested For Each records. That is to say, it makes a Cartesian Product between the tables: for each base table record of the external For Each (Category), it considers all records of the nested For Each base table (CountryCity).


Base tables =

```

For each Attraction order CountryName
  Print
  
  Country: CountryName

For each Attraction order CityName
  Print
  
  City: CityName

  Print
  
  Attraction

  For each Attraction
    Print
    
    AttractionName
  Endfor
Endfor
Endfor

```

Control Break

```

Country: Brazil
  City: Rio de Janeiro
    Attraction
    The Christ Redeemer
  City: Sao Paulo
    Attraction
    Museum of Football
Country: China
  City: Beijing
    Attraction
    Great Wall
Country: Egypt
  City: Cairo
    Attraction
    Egypt Pyramids

```

In this case we want to list all countries; for each one of them we want to list their cities, and for each city we want to list their attractions. The only restriction is that we want to do it only for the countries and cities that have tourist attractions recorded.

That is to say, we will have to implement a **double control break**, in which first we **group** by country, and within it we will then **group** by city. Within the latter group, we will **show** the names of all attractions. To do so, we will:

Define the grouping criteria using **order clauses**. Remember that the order is very important in a control break; it not only indicates the attribute or attributes used to list data, but also sets how to group it.

We could indicate an order for the innermost For Each, but this order will only be used in the conventional manner. That is, it will be used only for ordering purposes.

Base tables =

Control Break

For Each Attraction (Line: 17)
 Order: CountryName, CityName
 ! No index
 Navigation Start from: FirstRecord
 filters: Loop while: NotEndOfTable
 Join location: Server
 =Attraction(AttractionId)
 =Country(CountryId)
 ~CountryCity(CountryId, CityId)

Break Attraction (Line: 21)
 Order: CountryName, CityName
 ! No index
 Navigation Loop while: CountryName = @CountryName
 filters:
 Join location: Server
 =Attraction(AttractionId)
 =Country(CountryId)
 ~CountryCity(CountryId, CityId)

Break Attraction (Line: 23)
 Order: CountryName, CityName
 ! No index
 Navigation Loop while: CountryName = @CountryName and CityName = @CityName
 filters:
 Join location: Server
 =Attraction(AttractionId)
 =Country(CountryId)
 ~CountryCity(CountryId, CityId)

A single order! The table is run through only once and is "broken"

We have a double control break, which implies three For Each Commands. In the order of the first one, the outermost group is established. The innermost group is established in the second one.

If we look at the navigation list, we can see the word Break for every internal For Each, indicating the same base table, Attraction, and therefore, a control break.

In addition, it will run through this base table only once. To do so, it needs to order by the concatenation of the attributes included in the orders of the For Each commands. That's why it chooses CountryName, CityName.

Note that in the second For Each, the break is performed by country, iterating on the country it is positioned in the first For Each. In the third For Each, the break is performed by city, iterating on the city it is positioned in the second For Each.

Think about how the previous list will be executed if instead of ordering the first For Each by CountryName and the second by CityName, we order by the pair CountryName, CityName. Note that in this case the navigation list will be different from the one shown above, in the second For Each. There, Loop while will read "CountryName = @CountryName and CityName = @CityName".

GeneXus™

The power of doing.

Videos	training.genexus.com
Documentation	wiki.genexus.com
Certifications	training.genexus.com/certifications