How to List Related Information

Nested For Each Commands

GeneXus™

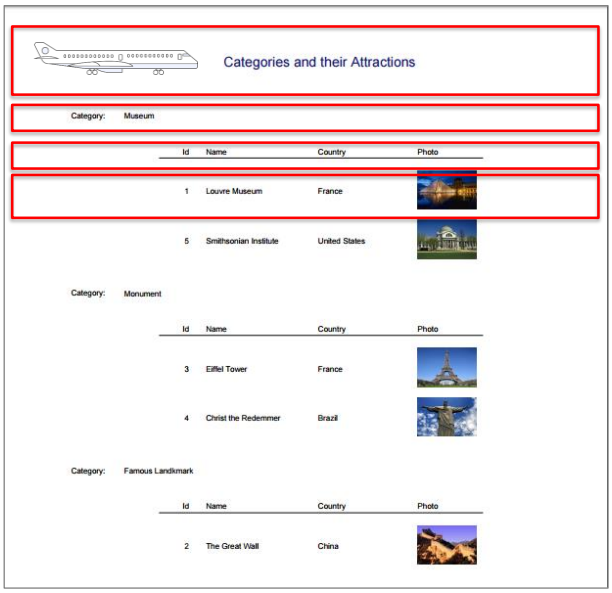
The listing below is required:

Title

Categories

ColumnTitles

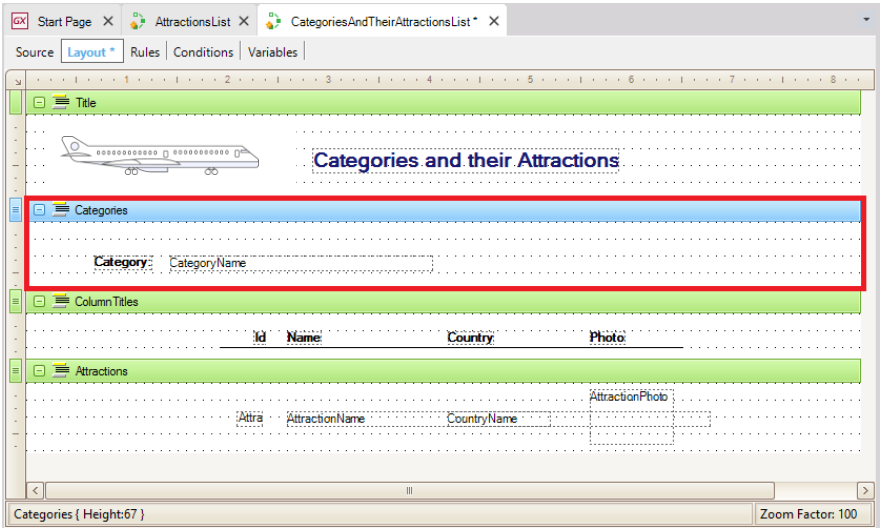
Attractions



Now, suppose that the travel agency has requested a list that shows all the tourist attraction categories, and for each category, all its attractions.

Note that the greatest difference between this listing and the one we had previously implemented is that now we want to group the attractions by category.

Layout



In the Layout, we added a new printblock.

Categories is the name we give to this new printblock. There we will insert a Text Block... Category ...and an attribute... CategoryName. The other printblocks will remain unchanged.

Note that we have two printblocks with fixed contents: Title and ColumnTitles, and two printblocks with variable contents that will have to be extracted from the database: Categories and Attractions. Both contain attributes. Categories has CategoryName, from the CATEGORY table, and Attractions contains all these attributes, that we had noticed belonged to the ATTRACTION extended table.

Now we move on to the Source.

Source

For each

CategoryId	CategoryName
1	Museum
2	Monument
3	Famous Landmark

For each

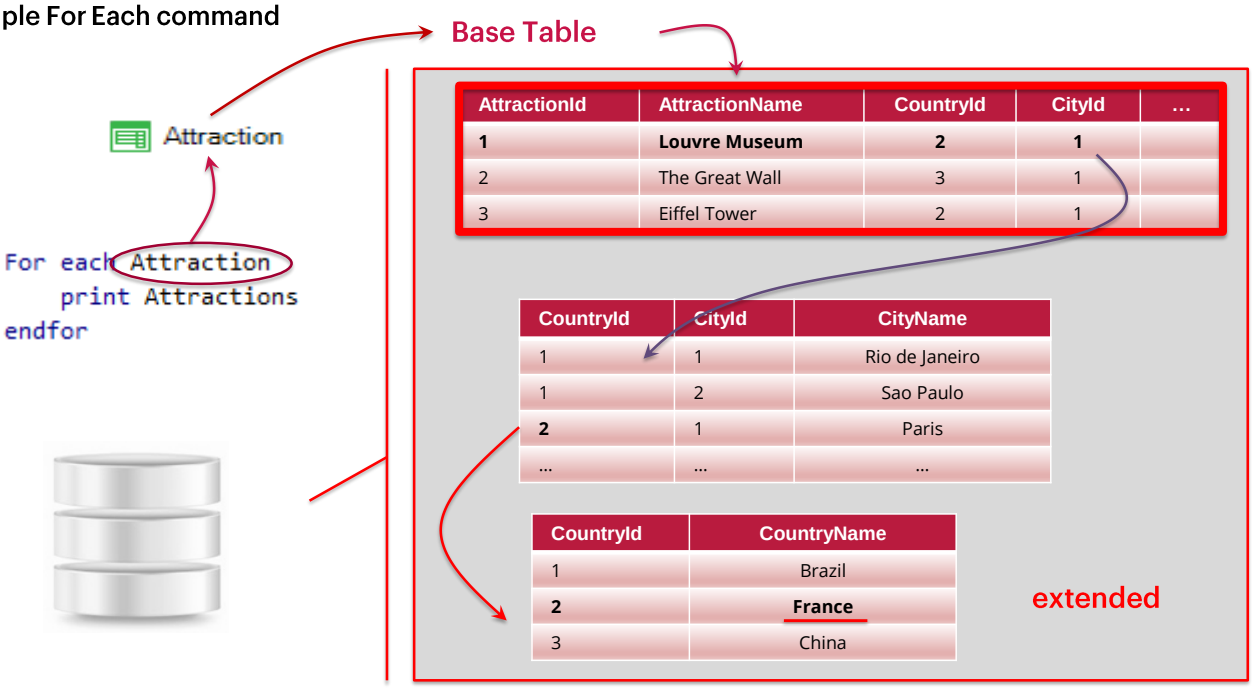
	AttractionId	AttractionName	CountryId	CategoryId	…
→	1	Louvre Museum	2	1	
	2	The Great Wall	3	3	
	3	Eiffel Tower	2	2	
	4	Christ the Redeemer	1	2	
→	5	Smithsonian Institute	4	1	

endfor

endfor

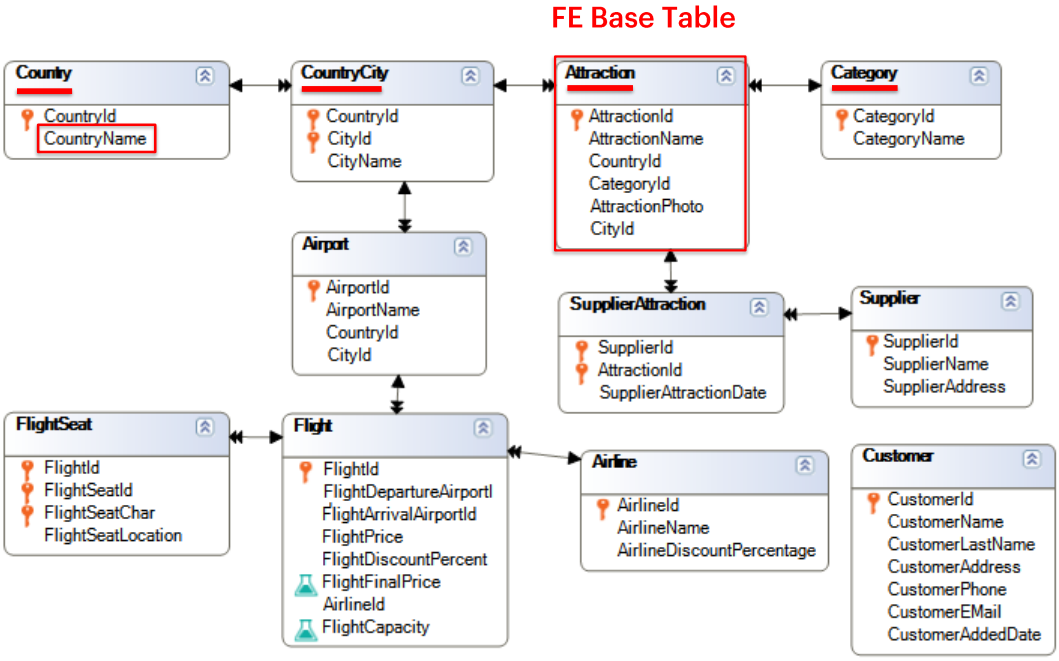
As we have to navigate categories and for each one of them **navigate several** attractions (the ones that belong to this category), this listing is different from that the one we have developed previously.

BEFORE:
Simple For Each command



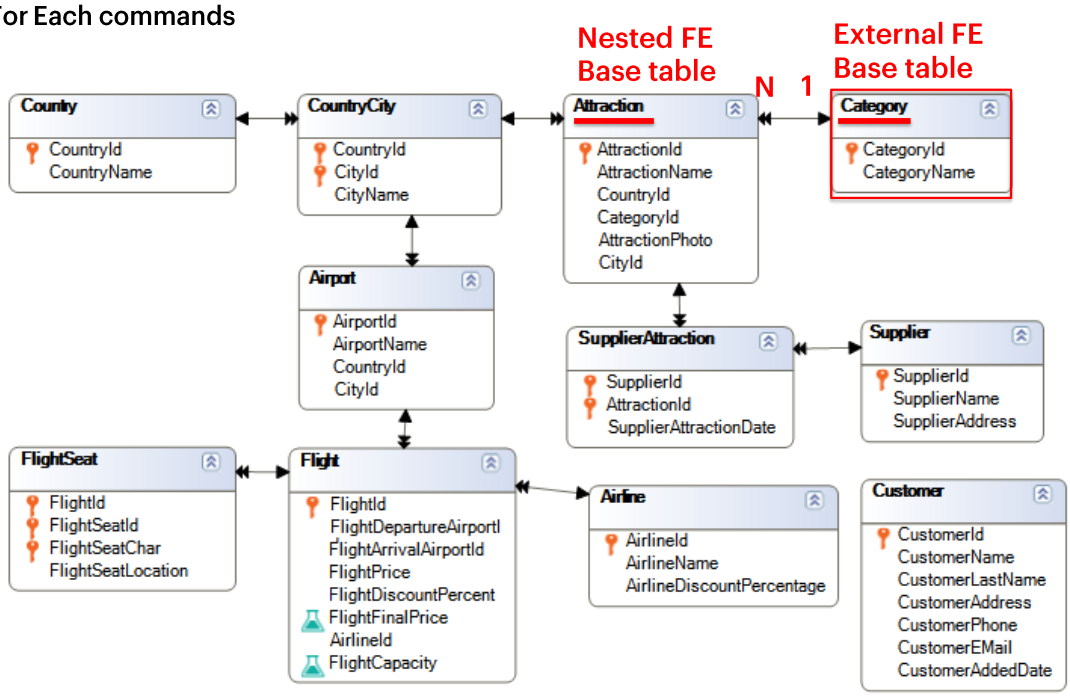
In the previous list we navigated attractions and since each attraction has only one country, we could retrieve the name of each attraction's country, because it was in the extended table of the base table that we were navigating.

BEFORE:
Simple For Each command



As long as the information we want to retrieve is available in the extended table of the base table we're navigating, **we can reference it directly in the For Each command.** That was the case of CountryName.

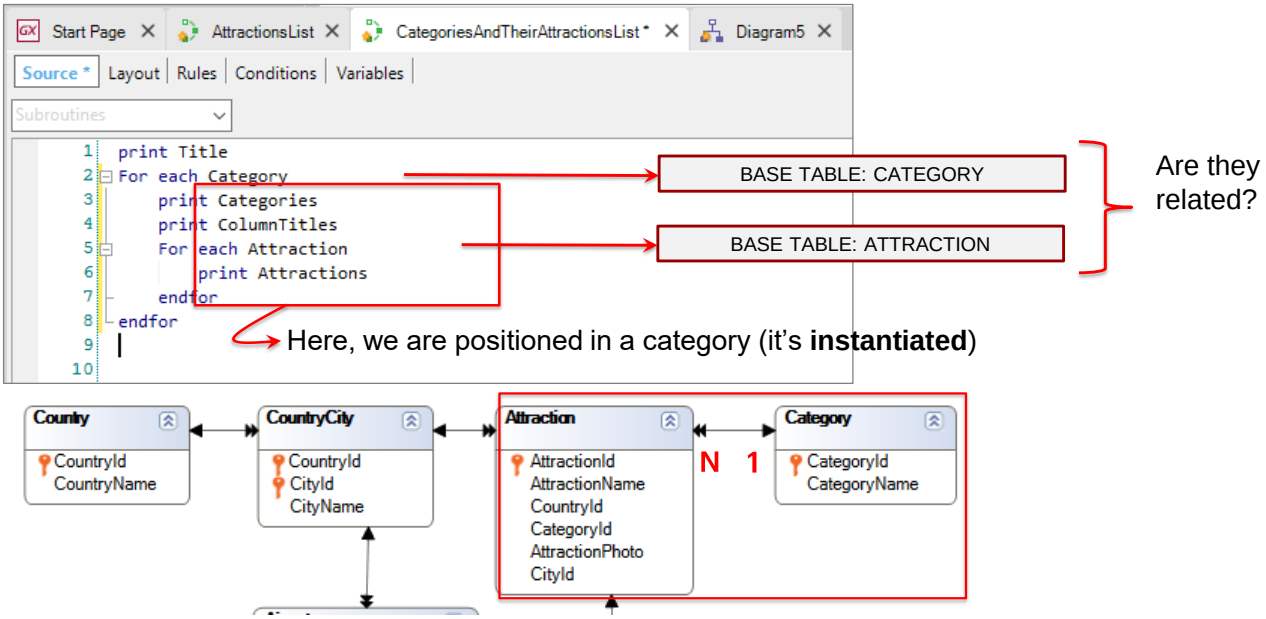
NOW:
Nested For Each commands



On the other hand, if we're navigating a table –in our case, Category– and for each accessed record we need to navigate **several related records which are saved in another table that doesn't belong to the extended table of the table we're navigating or running through,** as in this case, with the ATTRACTION table, we will need to write **another For Each command inside the first one, to run through the group of related records.**

That is to say, we will have a For Each command **nested** inside the other.

Designing the Source



We return to the Source of our procedure and start to write the first For Each command to navigate and show categories.

Next to the For Each command we type Category... Remember that here goes the **base transaction**; that is to say, the name of the transaction level whose information we want to navigate.

What do we want to do first with each category accessed by the For Each command? Print it. So, inside the For Each command we type: Print Categories.

Since the Categories printblock only includes the CategoryName attribute, and GeneXus has inferred that the **base table** of the For Each command is CATEGORY, and CategoryName is included in the extended table of this base table (because in this example it is in the table itself), everything will be in the correct order and it will be possible to retrieve the data. Otherwise, GeneXus would give an error.

After printing the category, we want to navigate **its** group tourist attractions ... therefore, we need to type the second For Each command, to run through the N attractions of the category we're navigating.

But right before navigating the group of attractions in that category, we will have to show the titles of the attractions that will be displayed, so we type the instruction **Print ColumTitles.**

Now we type the second For Each command, inside the body of the first one...

Next, we type Attraction because it is the name of the transaction whose associated table we want to navigate now. Inside the For Each command, we type Print Attractions.

Next, we type **Endfor** to close this navigation and **Endfor** again to close the first one.

How did GeneXus know which attractions had to be shown for each category if we didn't explicitly indicate anything about it?

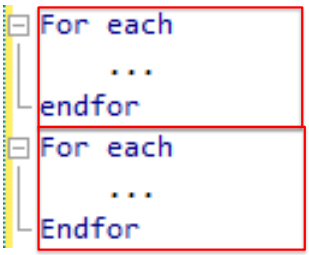
Let's look at the For Each commands. We know that a For Each command runs through N records in a table and, for each one of them, runs a series of instructions; those found inside the For each command. Inside this “body” of the first For Each command, we will be positioned each time on a single category. We say that the category is **instantiated**, every time. It's a **certain** category. Only when the execution of the instructions in the body is completed, it moves on to the following category.

Therefore, before starting to execute the nested For Each command, GeneXus already knows the category in which it is positioned at that moment.

That's why we wrote a For Each command that navigates the attractions, without adding a **where** clause to filter those attractions meeting the condition that their category must match the category we're positioned in within the first For Each command.

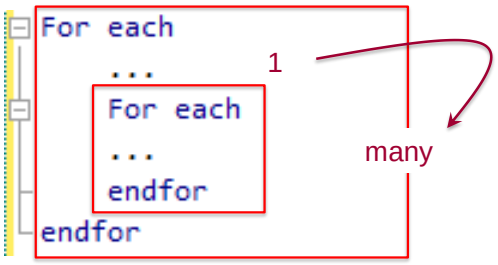
Implicit filter in nested For Each commands

Parallel For Each commands



Independent navigations

Nested For Each commands



Related navigations?

How did GeneXus determine that filter without us having to write it? The answer is in the way For Each commands are written.

If two For Each commands are written one after the other, they are **independent of each other**.

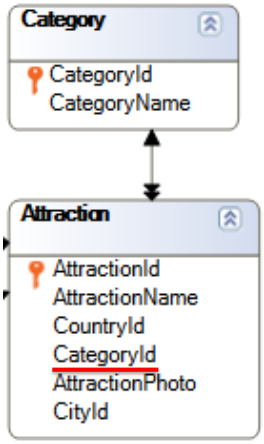
On the other hand, we type one For Each command **inside** another For Each command because for each record of the first navigation we want to run through a set of records in the second one.

When we write nested For Each commands, GeneXus determines for every For Each command the base table that will be navigated... **and then looks for relationships between that information**.

Implicit filter in nested For Each commands

Primary Key

CategoryId	CategoryName
1	Museum
2	Monument
3	Famous Landmark



Foreign Key

AttractionId	AttractionName	CountryId	CategoryId	...
1	Louvre Museum	2	1	
2	The Great Wall	3	3	
3	Eiffel Tower	2	2	
4	Christ the Redeemer	1	2	
5	Smithsonian Institute	4	1	

In our case, the base table of the external For Each command is CATEGORY, and the base table of the internal For Each command is ATTRACTION. GeneXus knows that there's a common attribute between both tables: This common attribute is **CategoryId, which is a primary key in CATEGORY and foreign key in ATTRACTION.**

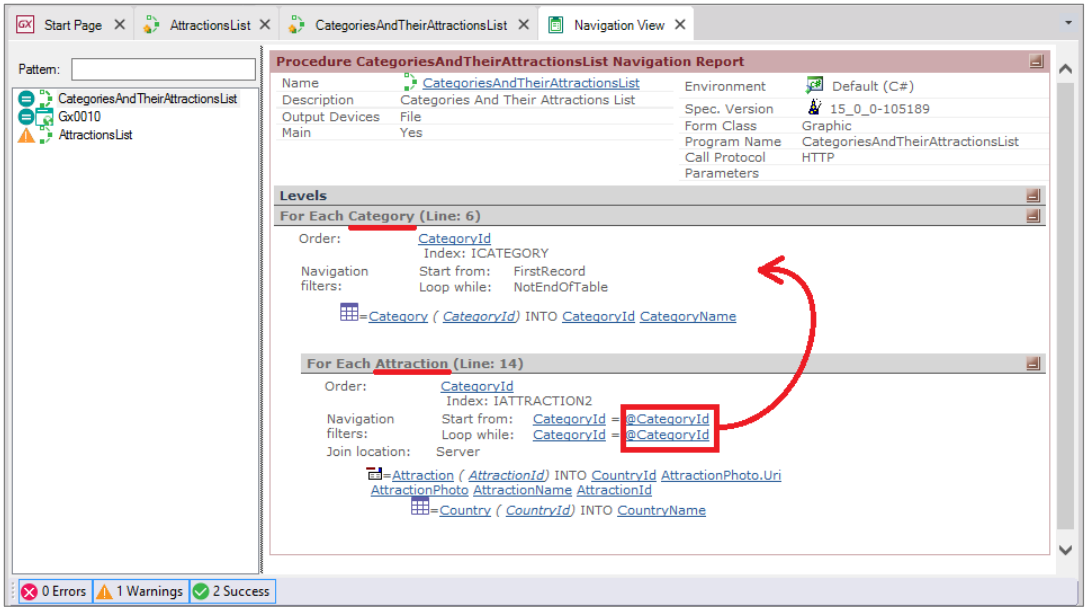
In this way, the CategoryId attribute relates the tables ATTRACTION and CATEGORY, as we can see here in the diagram, establishing a 1 to N relationship. That is to say, for every category there are many related attractions.

Implicit filter in nested For Each commands

Therefore, for every category navigated in the external For Each command, GeneXus runs the For Each command that navigates the attractions table, **filtering only those attractions whose CategoryId value matches the CategoryId value of the category we're positioned in.**

It's **exactly** as if in the internal For Each command we had written **Where CategoryId=CategoryId**.... but we don't have to type it because GeneXus detects it and applies it.

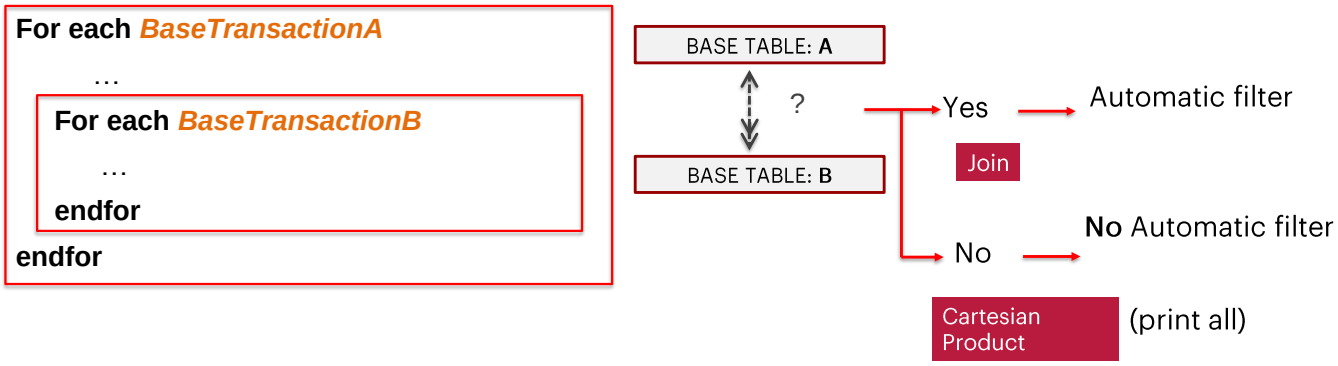
Navigation List



If we open the Navigation List of this procedure, we can see that it provides information about the two For Each commands: the base table of the external one is Category, and that of the nested one is Attraction. In addition, we can see that categories are retrieved in the order specified by their identifier, CategoryId, and that attractions are also ordered by this attribute, but in this table it is a foreign key. It is the attribute that relates them, and that's why we see in the navigation filters that only the attractions in this category will be retrieved.

We have seen how easy it is to obtain information and display it in a report... but procedures can do much more than that. We will see that later on.

Review



As a review, let's remember that when we type nested For Each commands, GeneXus determines, for every one of them, the base table that it will navigate... and looks for any relationship between these base tables.

If the answer is Yes, as we've seen in the list shown in this class, it will apply an automatic filter to the records run through by the nested For Each command. This case of nested For Each commands where information is filtered according to a relationship criterion is called **Join**.

Review

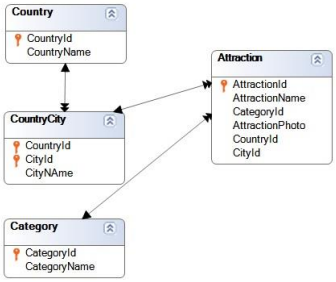
Join

```
For each Country.City
    Print countrycity
```

1

N

```
For each Attraction
    Print attraction
Endfor
```



Direct 1-N

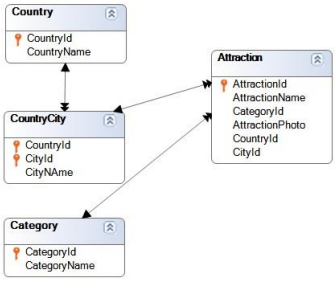
Indirect 1-N

```
for each Country
    print country
```

1

N

```
    For each Attraction
        Print attraction
    Endfor
Endfor
```



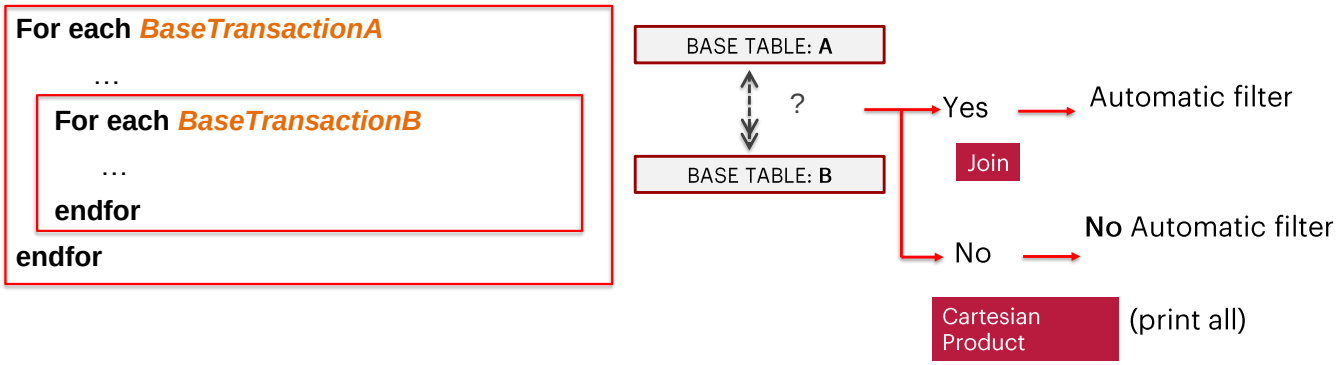
Here we can distinguish two cases of 1-N relationships between both tables.

The first one is direct. Note that the base tables of the external and nested For Each command are CountryCity and Attraction, respectively, which are linked by a 1-N relationship.

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

Review

- Related information



If, on the other hand, the answer to the question about the existence of a relationship was No, No filter would be applied. All the records of the nested For Each command for every record of the external For Each command would be printed. This type of nested For Each commands where no implicit relationship is found is called **Cartesian Product**. Of course, the developer can always add explicit filter conditions by typing them directly in the For Each command with Where clauses.

In these cases, we assumed that the tables were different. In the following video, we will see what happens when the tables of the external and nested For Each commands are the same table.

GeneXus™

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