Procedures and Listings.

Command to access the database

GeneXus 16

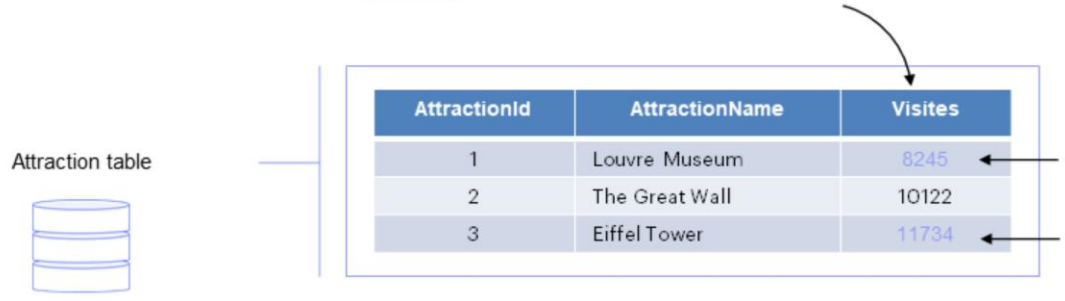
Procedure object

- ✓ It allows defining processes to access and navigate tables in the database with various objectives...

We will see several possible objectives.

Procedure object

- We may need to navigate the records of a certain table that comply with certain conditions, and for these records update a certain attribute with a given value.



Procedure object

- ✓ Navigate certain table and print all its data in a PDF list, ordered by some criterion.

Attraction table

AttractionId	AttractionName	CountryId	...
1	Louvre Museum	2	...
2	The Great Wall	3	...
3	Eiffel Tower	2	...

Attractions List

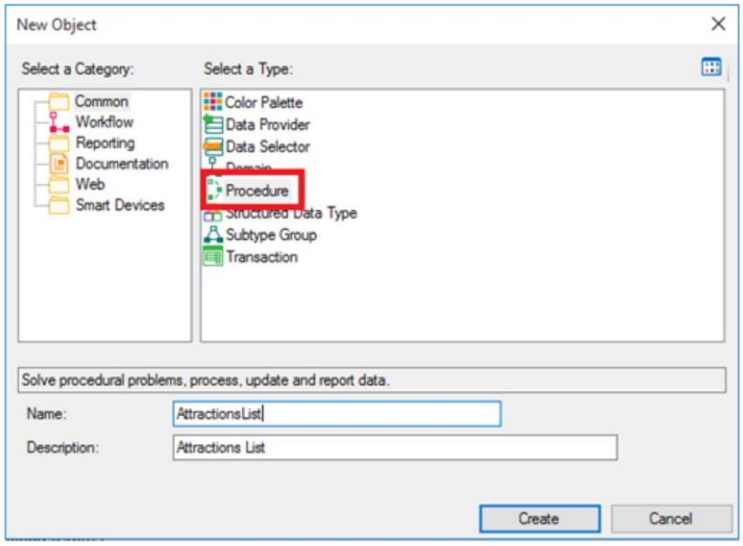
Id	Name	Country	Photo
1	Louvre Museum	France	
2	The Great Wall	China	
3	Eiffel Tower	France	

Procedure object

- Define specific processes that contain searches, calculations and database updates, and print that information.

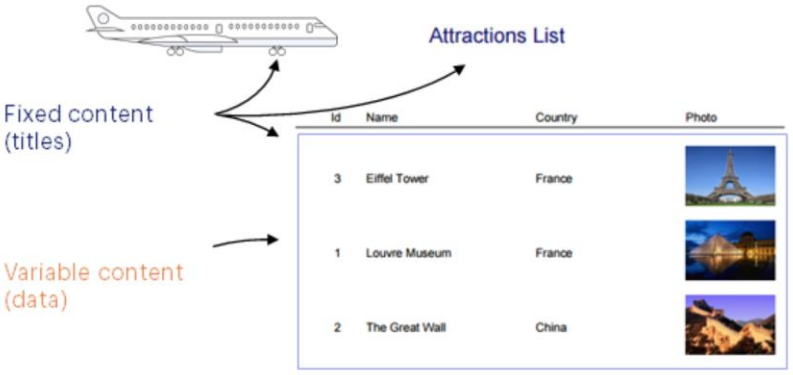
We solve all this by creating Procedures in our knowledge base (GeneXus **Procedure** object).

Procedure object



Example: list data in PDF format

- If you wish: list in a PDF file all the tourist attractions of the travel agency, in alphabetical order.

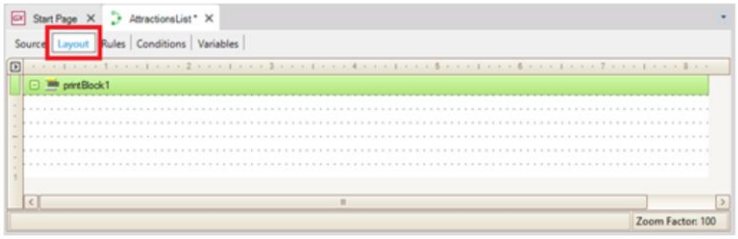


Let's start by defining a procedure to list all the tourist attractions that the travel agency has to offer, in alphabetical order.

Procedure: Source y Layout

- Source: for the instructions to be executed

- Layout: for designing the output

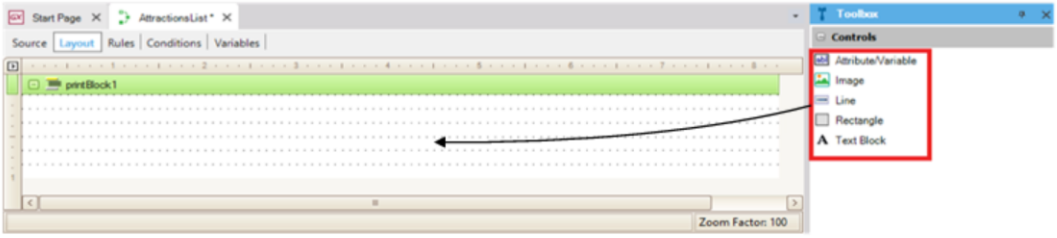


Once the object has been created, we see that GeneXus takes us to a section called Source. Here is where we'll type commands and orders that enable the procedure to meet its objective, which in this case is to print a list of tourist attractions.

Now let's look at this other section called Layout. The layout is the place where the output is designed; that is to say, where we set how we want to view our data.

Layout

- It is made up of printblocks



- One already appears by default. It is called printBlock1.

It is made up of printblocks and inside the printblocks we will include what we want to show.

We may want to show titles, lines, rectangles, images, as well as attribute or variable values. To do so, we will drag them to the printblock from the toolbar.

Note that a printblock is automatically included in the layout.

With this printblock we can display a title or today's date, or we can add more printblocks in this section, as we will see.

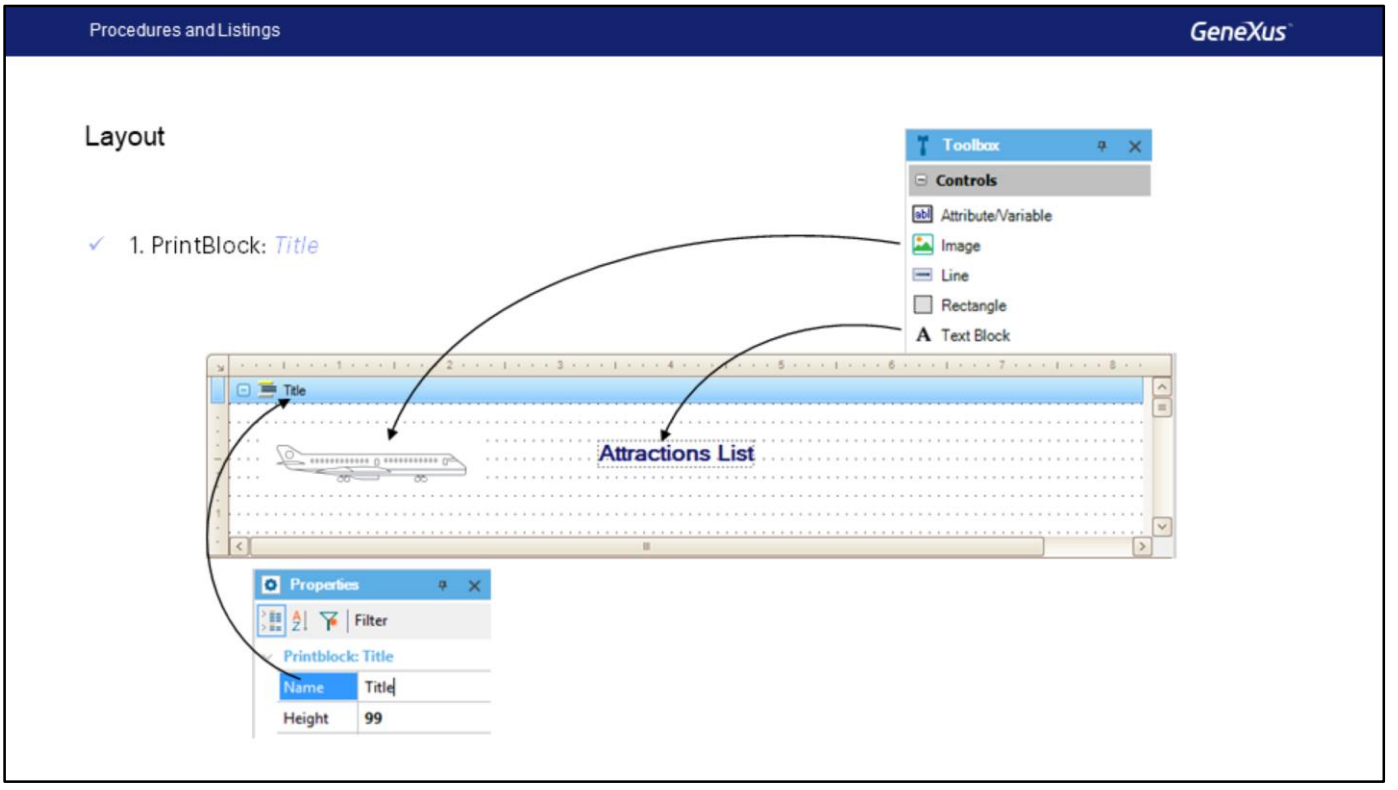
Layout

- ✓ Printblocks for our listing:

We may define three printblocks:

- one with the list title and its image, which could be named Title
- another one to show the column titles with the line below them, that could be named ColumnTitles
- and a third printblock where we will display the tourist attraction details, which will be named Attractions.

So, let's start to define this.



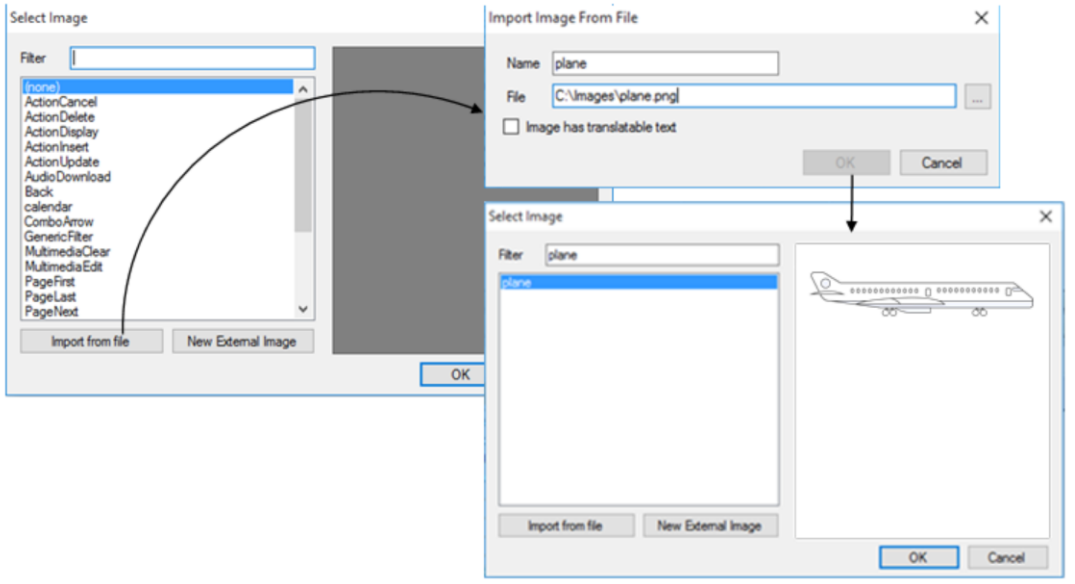
We can use the printblock that was created when we created the procedure object for the title and image.

Let's start by the title. To do so, from the Toolbox we drag the Text Block control... edit its properties... in the Text property we type "AtractionsList". We also change its color, MidnightBlue and font... Size 14, Bold=True, and select its position in relation to the margins.

We will give this printblock a clear name that represents what is being displayed. To do so, we select the printblock properties and edit its Name property by assigning it the name "Title".

Now, let's insert the image with the plane on the left by dragging the Image control from the Toolbox and dropping it where we want it to be placed. Doing this...

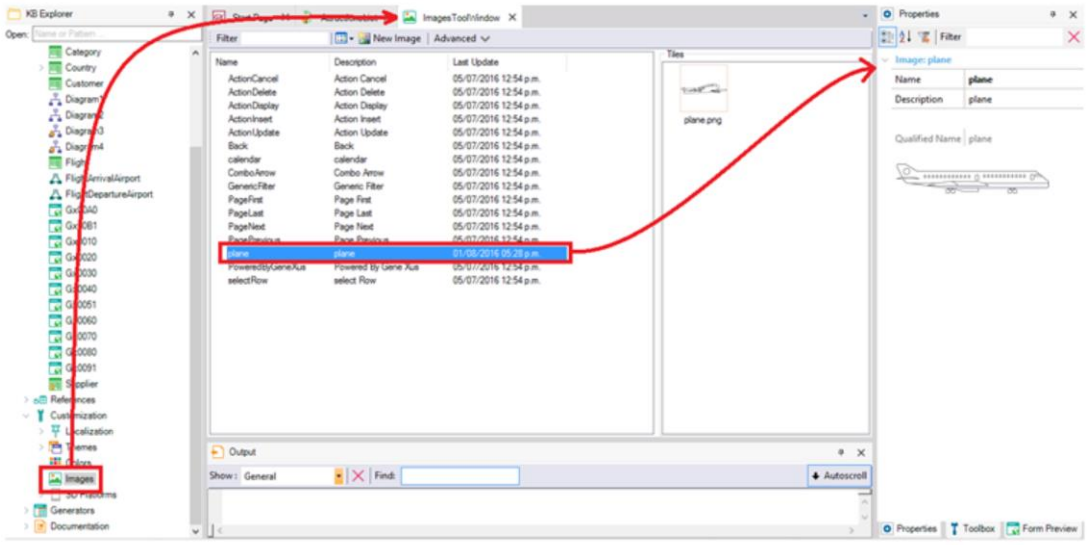
Images in the Knowledge Base



...a window is opened for us to select one of the images existing in the knowledge base, or add a new one, for example, by importing it from a file.

The "Import from file" button allows us to explore our file system and select the image, which will be created as a GeneXus object of **Image** type with the image filename as default name. From then on we will be able to freely use the image in our KB.

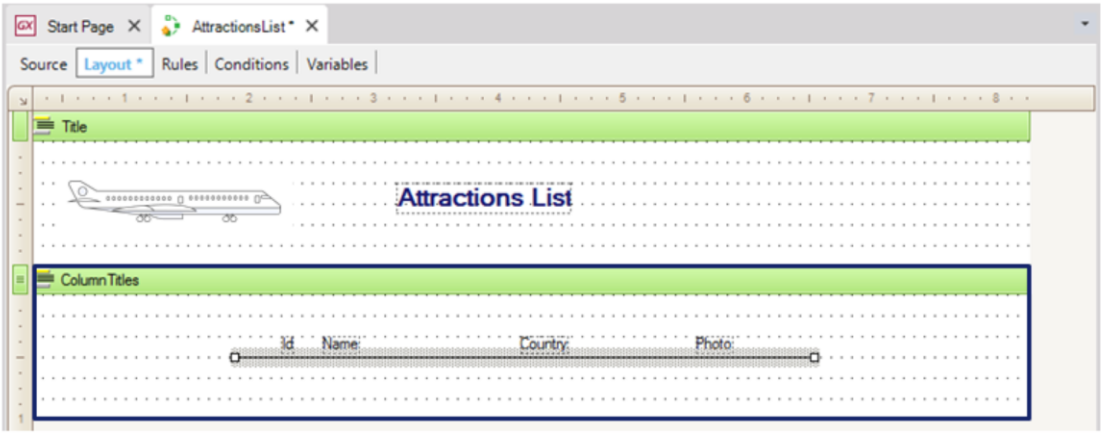
Images in the Knowledge Base



Note that from KB Explorer/Customization/Images we can access all the images in the KB, among which is the image of the plane.

Layout

✓ 2. PrintBlock: *ColumnTitles*



Now we will create another printblock to include the column titles, with a line below them. If we right-click **on a certain printblock** and select the "Insert Printblock" option a new printblock will be inserted **below it**.

The way in which printblocks are ordered in the Layout is not important because it doesn't mean that they must be printed in that order. We **determine when to print each printblock** in the code that we type in the procedure Source. We will see it soon.

Now we will give the name "ColumnTitles" to this new printblock.
And in this new printblock we will insert a TextBlock for every text that we want to show as column title.

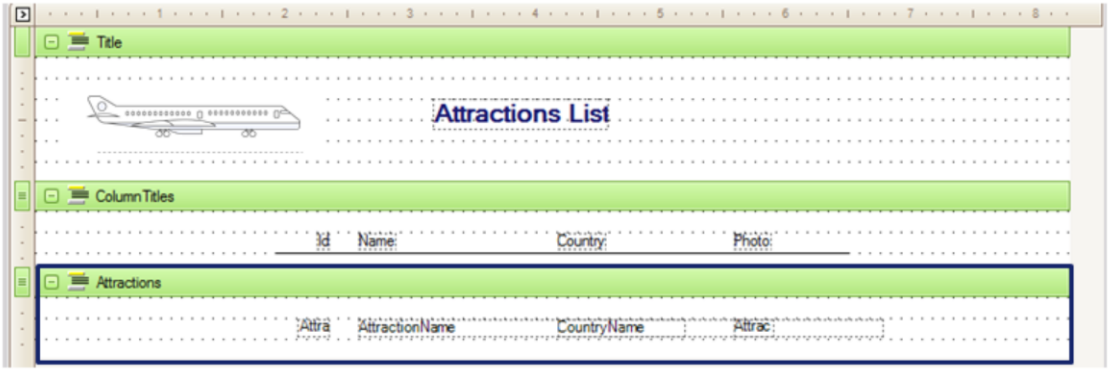
So, from the Toolbox we drag the textblock, and in its Text **property** we type "Id". We add another Textblock, and in its Text property we enter the text "Name". And create another Textblock to show the text "Country". Lastly, we create a Textblock for the title "Photo".

We place the controls in the positions we want... They can be aligned by selecting them all at once and then: Menu/Layout/Align/Bottom.

Lastly, we will insert a line below these column titles. So, we return to the Toolbox and drag a "Line" control. We drag it from here... and give it the length we want...

Layout

✓ 3. PrintBlock: *Attractions*



We still have to add the third Printblock that we had mentioned, to show the tourist attraction details. So, we insert a new Printblock and call it: Attractions.

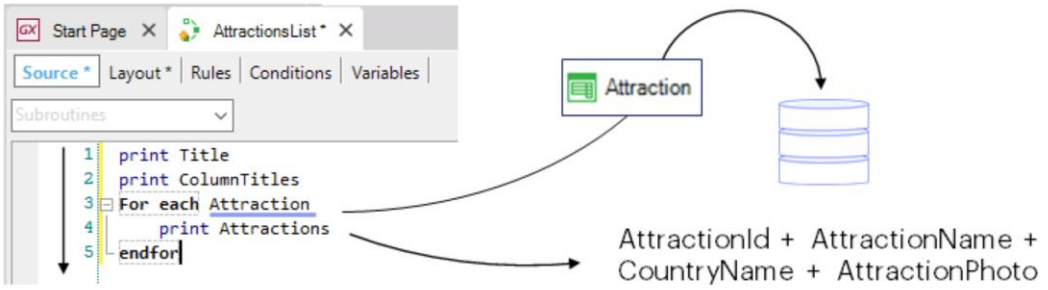
Since the data is stored in attributes, we return to the Toolbox, select a control of "Attribute/Variable" type and drag it below the "Id" title.

In the window that is opened we choose a variable or attribute to show in the control. We see that in addition to &today, a procedure has these other system variables.

We can also insert attributes in a printblock from the **Insert / Attribute** option.

Source

- ✓ Commands:
  - *Print* to print a printblock, and
  - *For each* to go over a table and its extended table to do something with each data record.



The design of how the data will be displayed in the list is ready.

Now we have to type the necessary code to obtain the right information from the database and have printblocks printed in the order we want.
Let's go to the Source option...

The first thing we want to print is the report title, so we type “print Title”. Since the instructions we type in the Source will be executed downwards, **this instruction will be the first one to be executed.** With it, we're sending the contents of the printblock called Title -the list's title- to print. The Print command must always be followed by the name of a printblock defined in the Layout.

Next, we want to print the column titles, so we have to give the order to print the “ColumnTitles” printblock...

With these two instructions we have given the order to print the fixed part of the report; that is to say, the part that will not change with the data: the part containing the report title and the plane image, and the part containing the column titles.

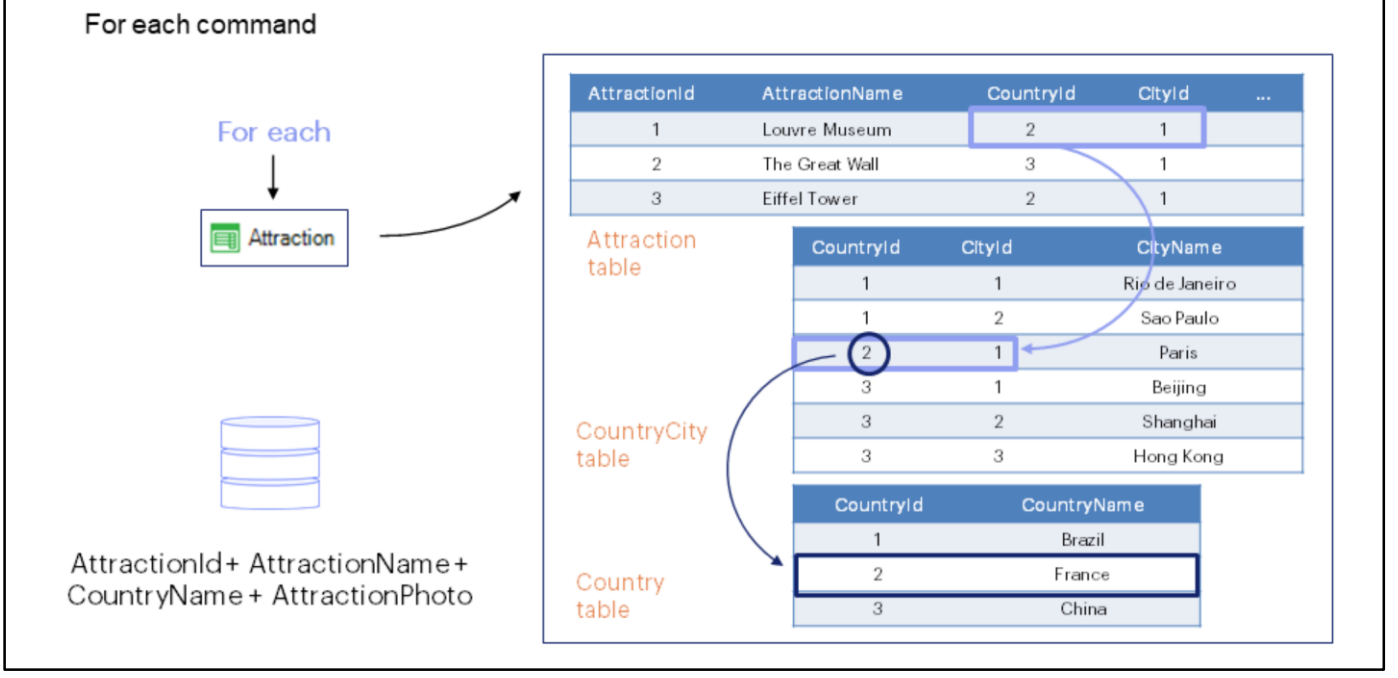
Now we have to print the attractions' data that is stored in the database. To do so, we must access the physical table that has this information stored; that is to say, the table associated with the Attraction transaction.

The command that allows us to access a physical table is the “For Each” command. The physical table that is accessed is called base table of the For Each command.

So we type the For Each command... Next to it: Attraction. Why do we type Attraction next to the For Each command?
Because it is the name of the attraction **whose associated physical table** we want to navigate...

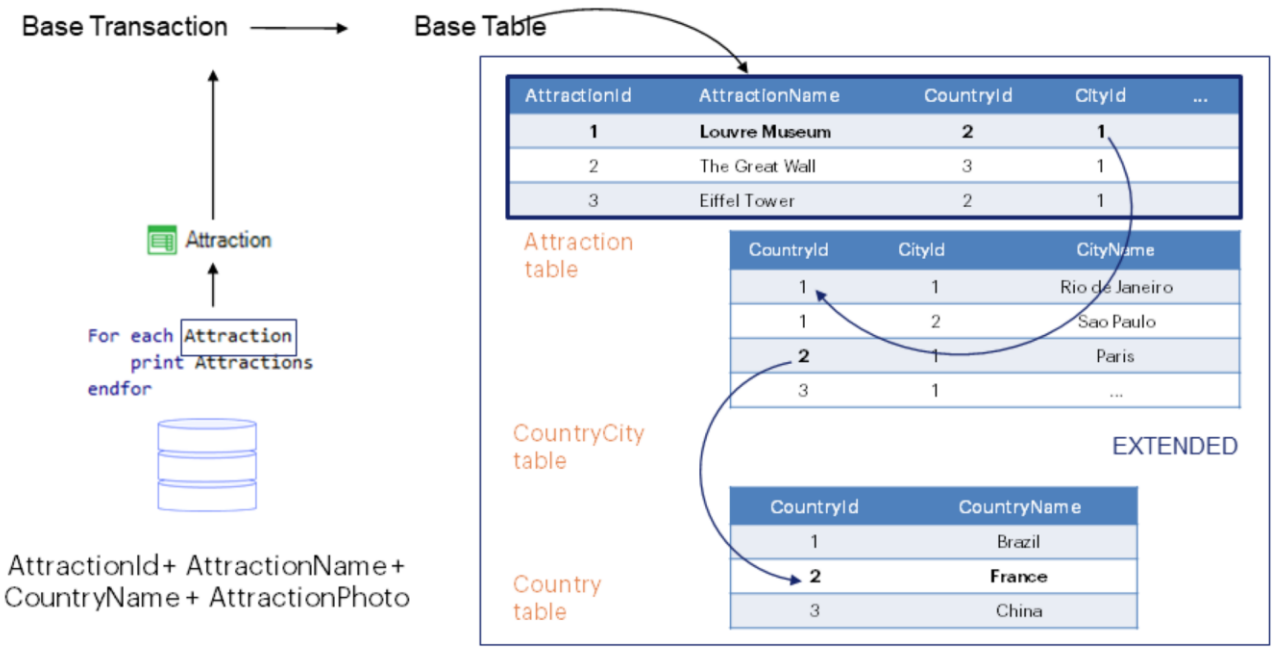
... and now... since we want to print, for every tourist attraction, the content of the attributes AttractionId,

AttractionName, CountryName, and AttractionPhoto we type the order to print the “Attractions” printblock that contains them. We type Print Attractions.



In this way, we have told GeneXus to navigate the ATTRACTION physical table, which corresponds to the Attraction transaction.

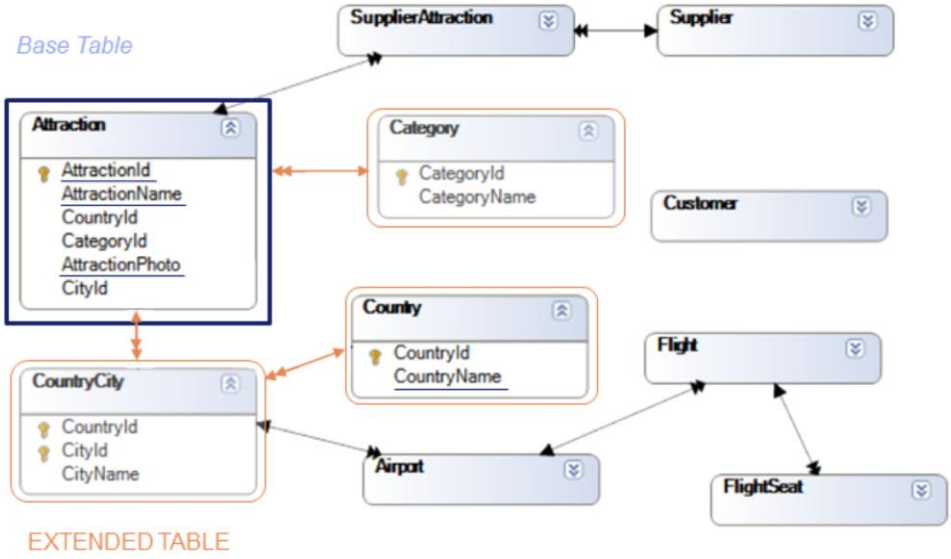
Since within the For Each command we have invoked a printblock containing attributes of the ATTRACTION and COUNTRY tables, applying the concept of extended table, for each navigated attraction, the COUNTRYCITY table will be accessed, and from it the COUNTRY table, to obtain the name of the country where this attraction is located.



In this way, we have told GeneXus to navigate the ATTRACTION physical table, which corresponds to the Attraction transaction.

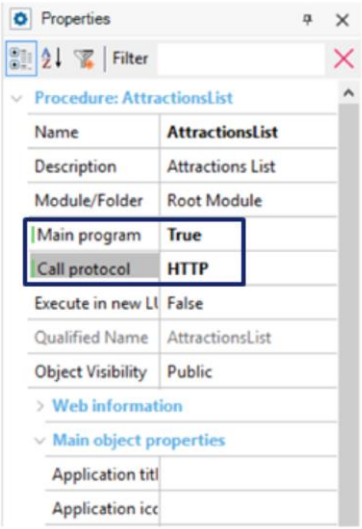
Since within the For Each command we have invoked a printblock containing attributes of the ATTRACTION and COUNTRY tables, applying the concept of extended table, for each navigated attraction, the COUNTRYCITY table will be accessed, and from it the COUNTRY table, to obtain the name of the country where this attraction is located.

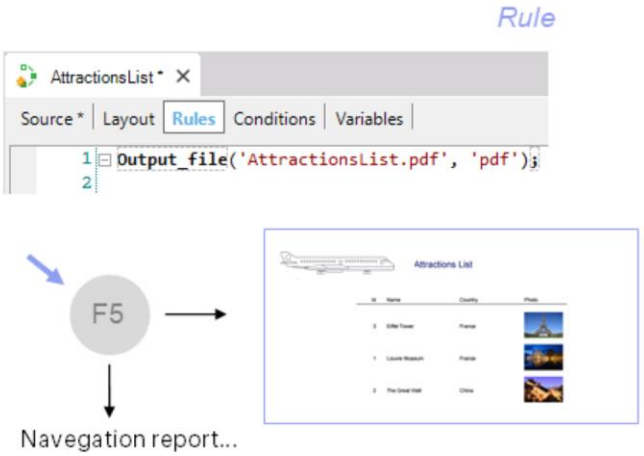
Base Transaction, Base Table and Extended Table of the For each command



Here we have a diagram with the relations between the tables in our knowledge base.
Inside the For each we have the AttractionId, AttractionName, AttractionPhoto and CountryName attributes. The first three belong to the base table of the For each, while the last one belongs to one of the tables of the extended table.

Procedure object's Properties





We will run it to see the result.

First, we need to set some necessary properties to print the list in PDF format. We open the report's properties and set the "Main program" to True.
Next, in the "Call protocol" property we select "HTTP".

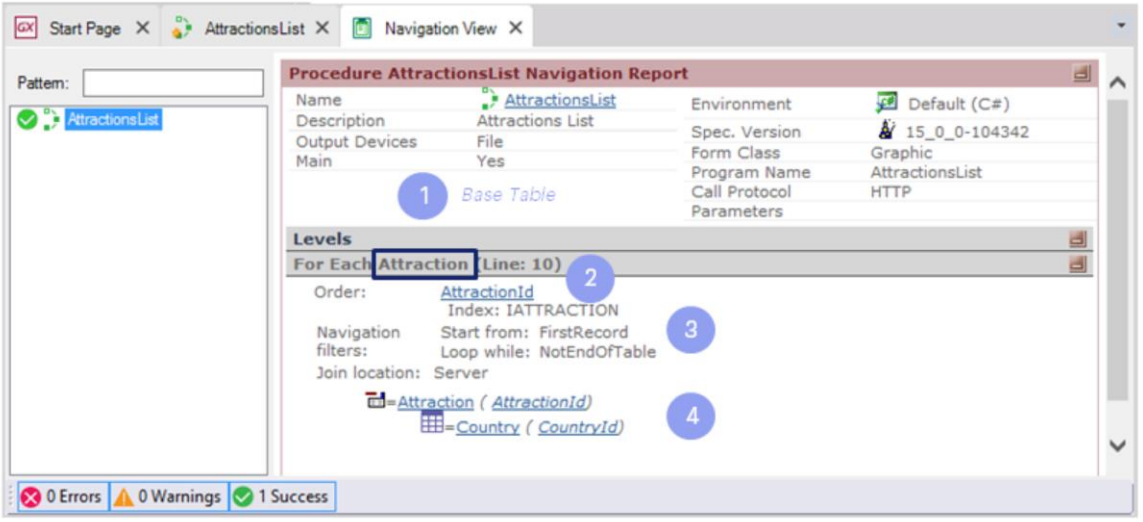
Lastly, we have to insert the OutputFile rule in the rules section... because as we can see, an object of this type also allows defining some rules –even if fewer than in a transaction–, so we select **Insert/Rule**... And we complete it by entering the filename of the "AttractionsList.PDF" list followed by the format that will be used: "PDF".

We save... and now we can run it.

The list is created! ...With the format selected... all the tourist attractions that we had entered are included in the list, each one with the name of the country they belong to and a photo.

In addition, a window "Navigation View" is opened in GeneXus with a report...

Navigation Report



The **physical table that will be navigated by the For Each command,** as well as other decisions made by GeneXus, are displayed in the **procedure's navigation list.**

This list is automatically created when the procedure to be run is generated. In this case, it was after pressing F5.

In it, GeneXus shows how it accesses the information in the database.

1. We can see that next to “For Each”, it also says Attraction, Podemos ver que al lado de donde dice “For Each”, to indicate that it is the **For Each command base table.**

Remember that the For Each command runs through a physical table; that's why the Attraction name displayed in the navigation list is that of the ATTRACTION physical table, not the name of the base transaction that we've written in the procedure. GeneXus infers this table because it is associated with the base transaction we've indicated.

2. It also indicates that to order the list of attractions the AttractionId attribute was used (it is the primary key of the Attraction table).

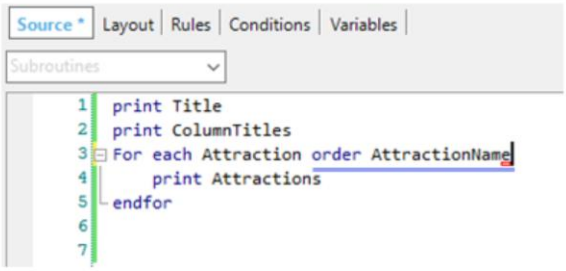
3. It also indicates that it has run through all the records in the table: because it started by the first record and iterated until reaching the end of the table.
That is to say, all the attractions were displayed...

4. lastly.... it indicates that the table **navigated** was Attraction and it had to **access** Country to retrieve information, because in our list we show the country name.

How to change the order of data

- The request was:

to list, on a PDF file all the tourist attractions of the travel agency, in alphabetical order.



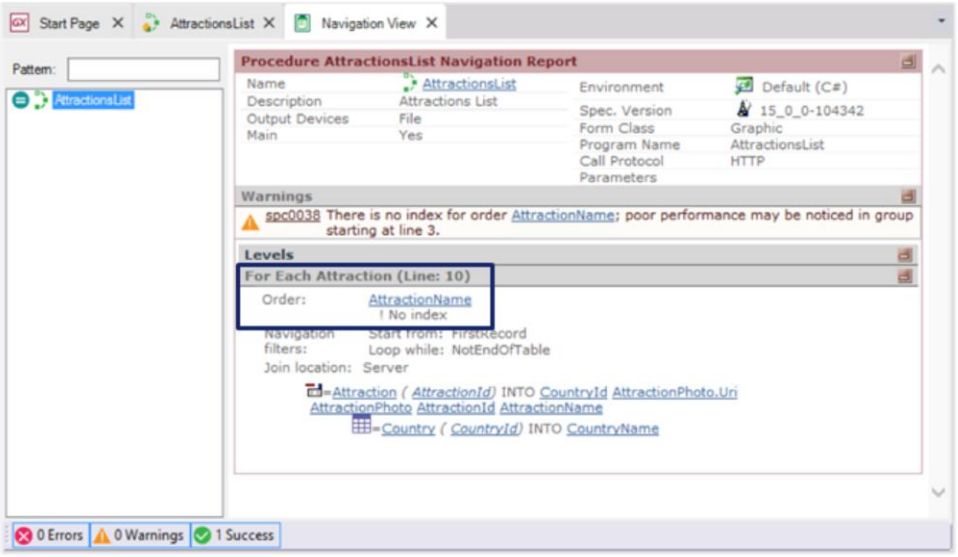
...it is possible to sort by any attribute of the *extended table* of the table that goes through the For each.

```
For each Attraction order CountryName
    print Attractions
endfor
```

Something we still had pending was that attraction should be listed in alphabetical order, by name of the attraction.

We can do this by simply writing the clause “order AttractionName” next to “For each Attraction”.

Navegation Report



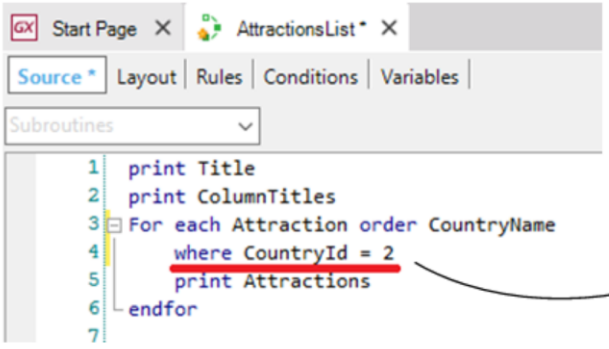
For now, we don't pay attention to the warning displayed in the list.

But we should note how the navigation listing informs us why the attribute will be ordered at the Output.

Just as we have added the **"order" optional clause** to the For each, the syntax of the For each also allows us to add several other optional clauses and definitions, as we will see.

How to define filters

- ✓ The requirement is: a listing of all tourist attractions in France.




```
where CountryName = "France"
```

Navegation report...

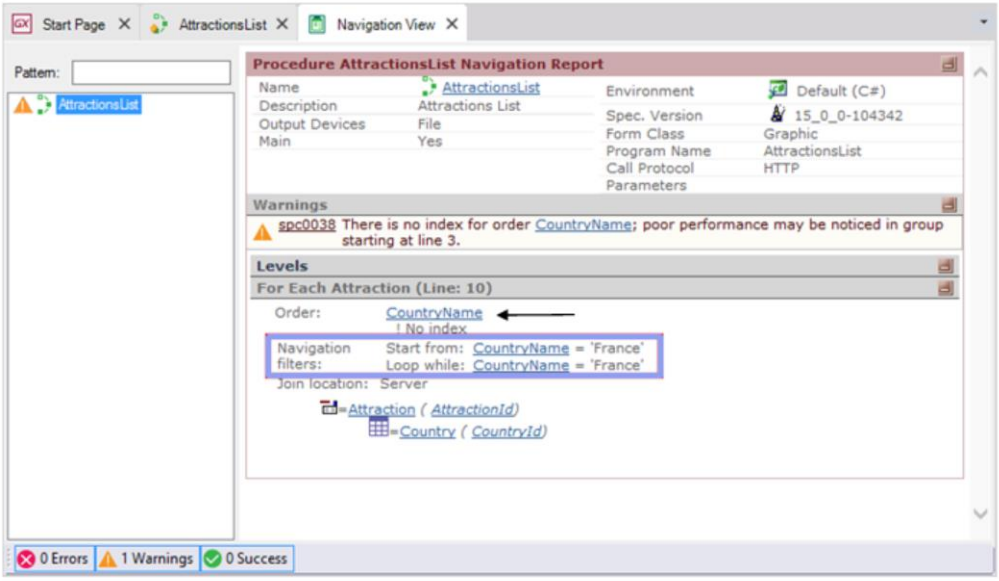
For example, what would happen if the travel agency asked us to list only the tourist attractions of France?

To the For Each command we would only add a clause called **Where**, to have it filter and show only the data that meets the desired condition.

So, we click on the line after the For Each command and type Where...CountryId=2, because we know that France's ID is 2.

Instead of filtering by Country identifier, we could also have written **Where CountryName="France"**.

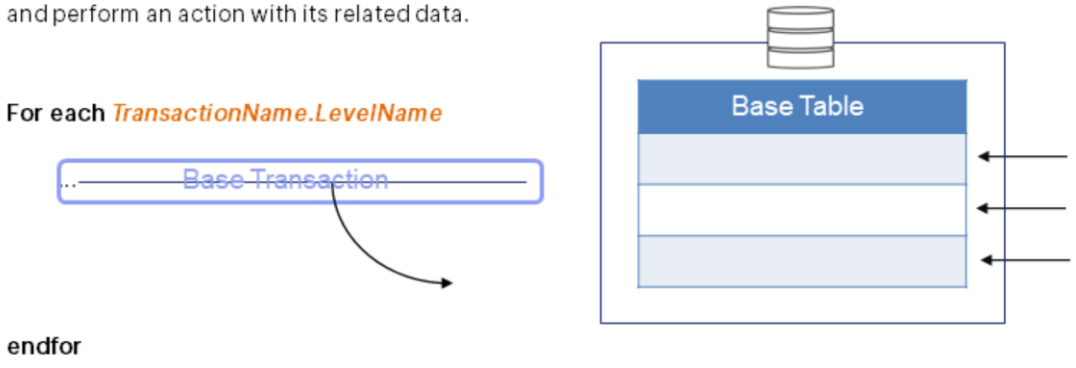
How to define filters: navigation report



We see that the Attractions table is no longer run through entirely. Since we're ordering by CountryName, to keep the countries called ‘France’ it only has to run through part of the table, not all of it. It is similar to when we look for the word 'France' in the dictionary. We don't look in the entire dictionary. Instead, we go straight to letter “F”.

Conceptualizing

- The For Each command is used to run through every record of a table and perform an action with its related data.



The For Each command is used to run through every record of a table and perform an action with its related data.
To this end, we indicate the name of the transaction, or, more precisely, the name of the transaction level whose associated table we want to run through.

This level indication is called **base transaction** of the For Each command.

From that level, GeneXus will infer the table to run through, which is called **base table** of the For Each command.

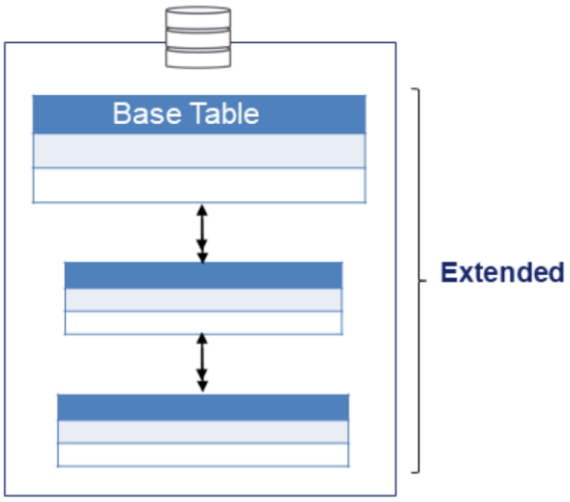
Conceptualizing

For each *TransactionName.LevelName*

...

Attributes present here must belong to the *extended table* of the base table to run through

endfor



The set of attributes between For each and Endfor must belong to the extended table of the base table to run through.

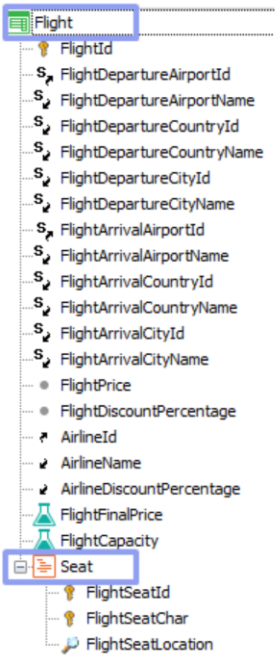
Conceptualizing

- ✓ Base Transaction:
  - 1st trn Level = TransactionName

```
For each Flight
Where FlightDepartureAirportId = 1
...
Endfor

For each Flight.Seat
Where FlightId = 15
...
Endfor
```

  - Nested trn Level = TransactionName.LevelName



In the first example we want to navigate the table of the flights departing from airport 1. For the first level, the transaction name matches the level name.

In the second example we want to navigate the seats on flight 15.

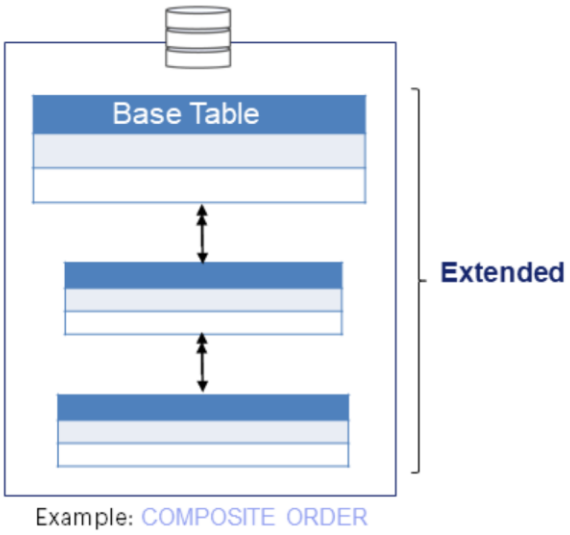
For each syntax

```
For each      BaseTransaction
    order  att1, att2, ... , attn
    where  condition1
    where  condition2
    ...
    where  conditionn
        MainCode
endfor
```

Here is a summary of what we've seen so far about the For Each command. We will expand this syntax as we talk about more topics.

For each syntax: order

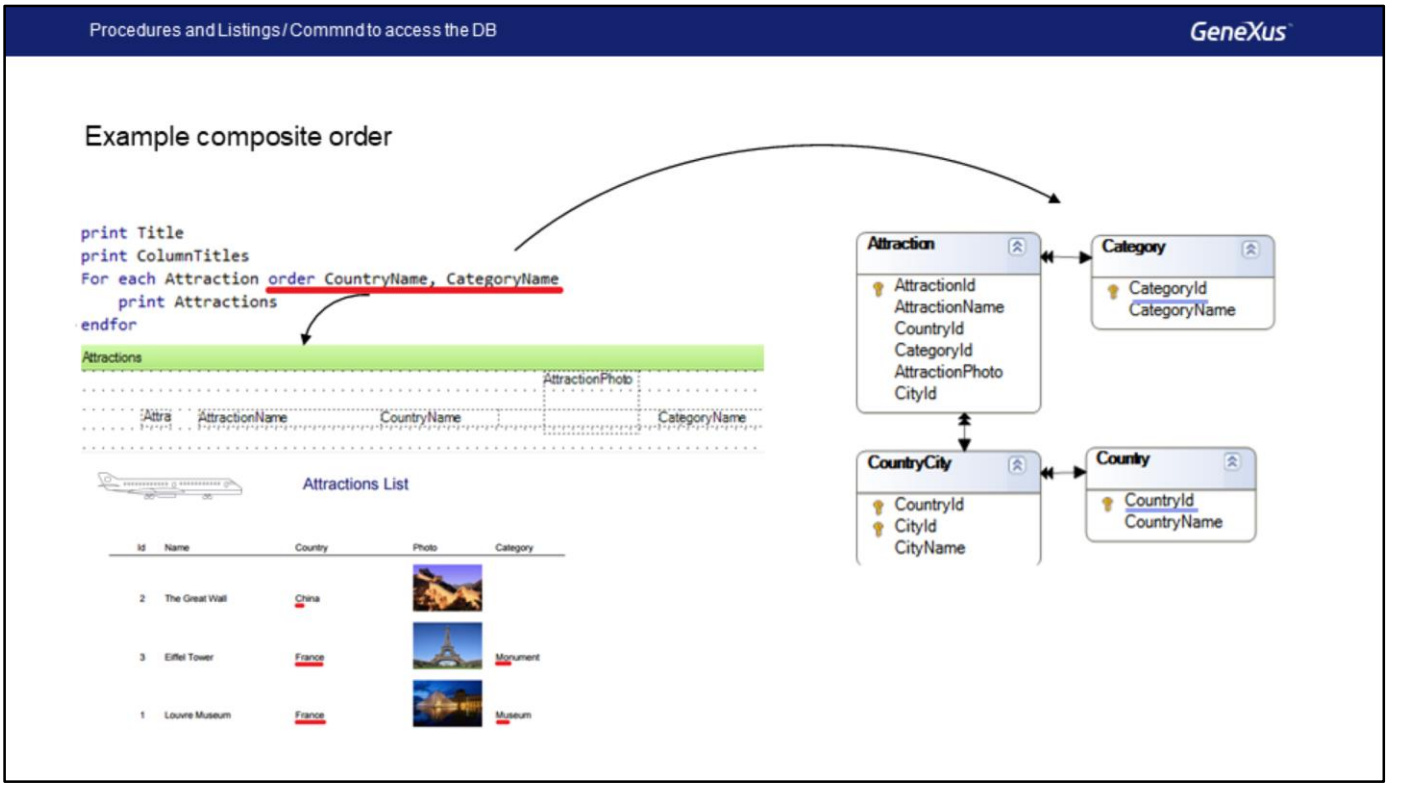
```
For each      BaseTransaction
   order att1, att2, ... , attn
   where condition1
   where condition2
   ...
   where conditionn
      MainCode
endfor
```



Example: COMPOSITE ORDER

With the **Order** clause we can indicate the criterion used to order the information returned by the For Each command. The order can depend on the attributes in the base table of the For Each command or its extended table.

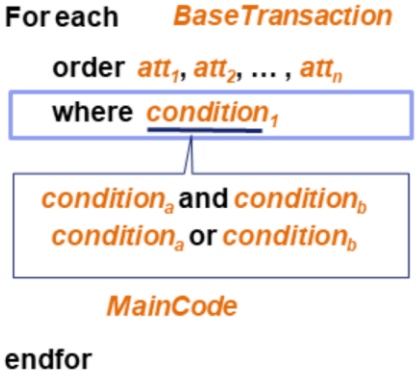
As we can see, we can order by a single attribute or by several attributes.

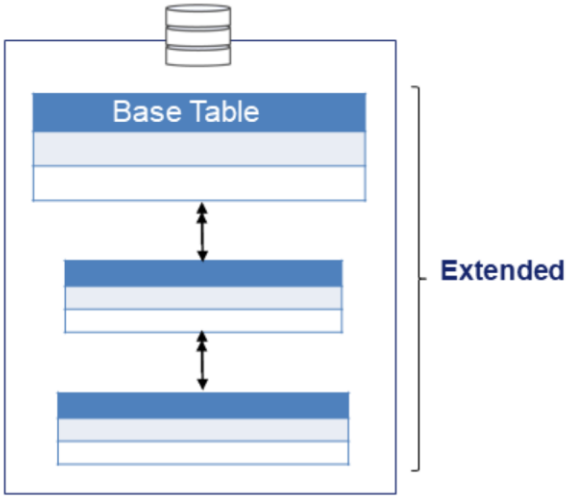


For example, if the tourist attraction category was also shown in the list, and we wanted to order it by country name, and within the attractions of the same country, by category name... we would type both attributes in this order: first CountryName and then CategoryName...

Here, CountryName and CategoryNameare not included in the base table, Attraction; instead, they are included in tables of the extended table.

For each syntax: where





To filter the information returned by the For Each command, the **Where** clause is used. In it, we indicate the condition that the records must meet in order to be selected.

The condition can be complex and include several conditions joined by AND or OR; that is to say, for example:

- $Condition_a$ **AND** $Condition_b$: it means that both conditions must be met at the same time.
- $Condition_a$ **OR** $Condition_b$: it means that if one of them is met, it is enough for the record being evaluated to pass the filter.

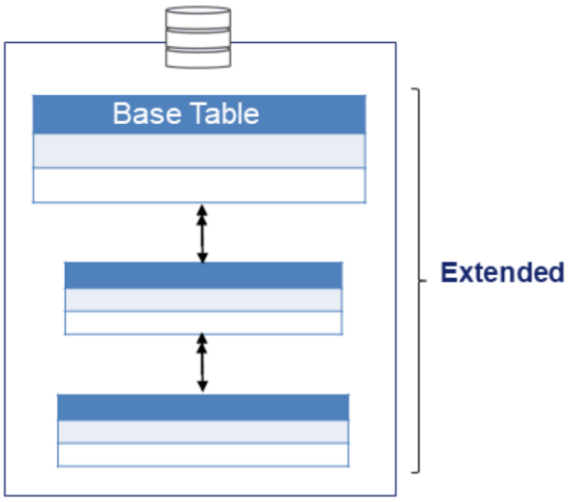
For each syntax: where

```
For each      BaseTransaction
    order att1, att2, ... , attn
    where condition1
    where condition2
    ...
    where conditionn
        MainCode
endfor
```

and

and

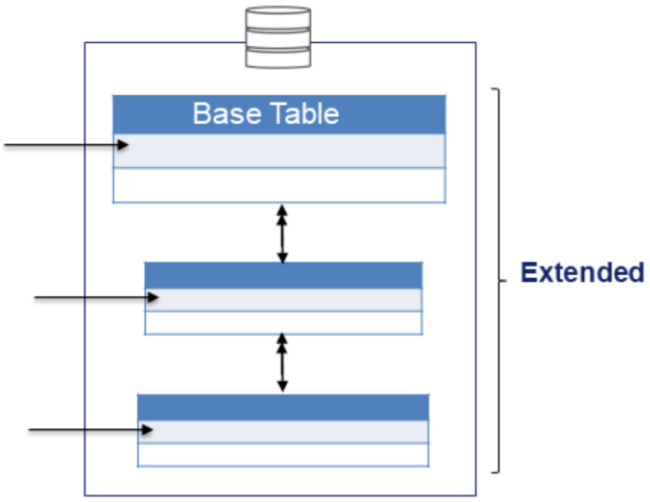
and



We can also add several **Where** clauses, which is the same as writing only one, with its conditions joined by **AND**.

For each syntax: Main Code

For each *BaseTransaction*
order $att_1, att_2, \ldots, att_n$
where $condition_1$
where $condition_2$
…
where $condition_n$
MainCode
endfor



Example...

Within the For Each command, in its **main code**, we type the commands that we want to run in sequence to perform, step by step, what we need in the record of the base table in which it is positioned in each moment... and those associated by extended table.

For each syntax: Main Code

```
print Title
print ColumnTitles
For each Attraction order CountryName, CategoryName
    print Attractions
endfor
```

Attractions List

Id	Name	Country	Photo	Category
2	The Great Wall	China		
3	Eiffel Tower	France		Monument
1	Louvre Museum	France		Museum

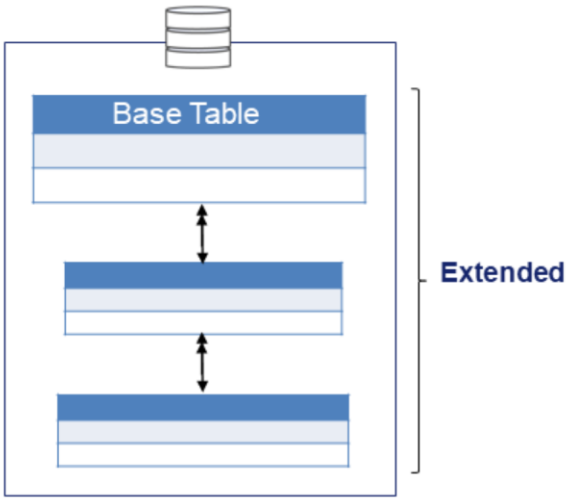
For example, print a printblock.

For each syntax

For each *BaseTransaction*
order att_1, att_2, … , att_n
where $condition_1$
where $condition_2$
…
where $condition_n$

MainCode

endfor



Therefore, this is how the For Each command structure looks so far.

The command accepts more clauses and options. Some of them will be explained in other classes, and others will be addressed in other courses

GeneXus™

The power of doing.

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