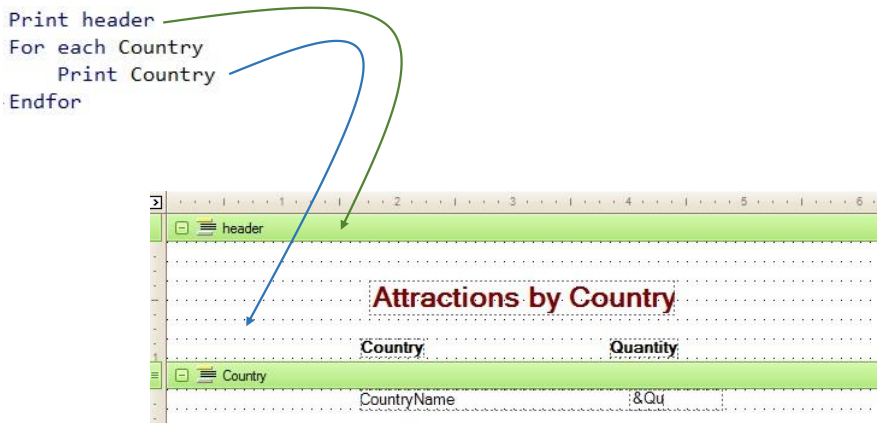
Inline formulas

GeneXus™

We create a list to meet the requirement



In a previous class we had distinguished global formulas from local or inline formulas.

Global formulas were defined at the attribute levels in a transaction structure; that is to say, it was indicated that a certain attribute was always calculated in a certain way. As a result, when later on it was necessary to retrieve the attribute value in any object, this formula was evaluated to get its result. That's why formula attributes were virtual attributes that weren't stored in tables.

On the other hand, formulas can also be used locally; that is to say, they can be set to be evaluated only in the object code where they are located. For example, in a procedure Source. We may assign the result of a formula to a variable.

Let's see some use examples.

Suppose that the travel agency requests a list that shows all country names and for each country, the number of tourist attractions they offer. To solve this request, we will create a procedure objec. We're positioned in a procedure Source and the first thing we will do is print a header with a title. To do so, we type this instruction: Print header. Header will be a printblock that we don't have yet. So, let's create it.

We open the Layout section and rename the default printblock to “header”. Now we can insert a textblock from scratch and format it as we want. Also, we

may open the list of attractions that we had before, copy&paste from the textblock to keep its format, and change only the Text property.
Here we can see that we have already added two texts to the printblock for the list columns and one line.

Let's go back to the Source to continue implementing the report. Below the instruction that prints the header we must add a For Each command to access the table that stores the countries and show them.
Within the For Each command we type: Print Country in order to show the CountryName attribute in a printblock with that name.

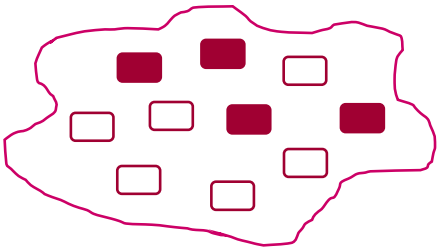
Therefore, we need to create that printblock. Inside it we add the CountryName attribute... aligning it with the column of Country title. Let's go back to the Source. This For Each command definition navigates the entire country table and shows the name of each country queried.

Global Formulas vs. Inline Formulas (local)

- **Global** Formulas

Attribute = *fx*

(in the Transaction structure)



Knowledge Base

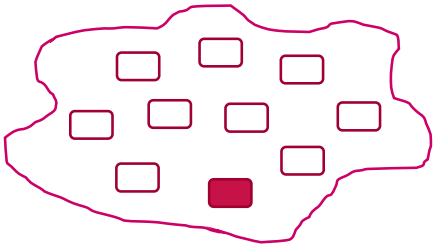
- Any object of the KB can access the calculation.
- The formula is evaluated every time an object uses the attribute.
- The attribute is no longer stored in the database.

Global Formulas vs. Inline Formulas (local)

- **Inline** (or **local**) Formulas

&Variable = *fx*

(in the code of an object)



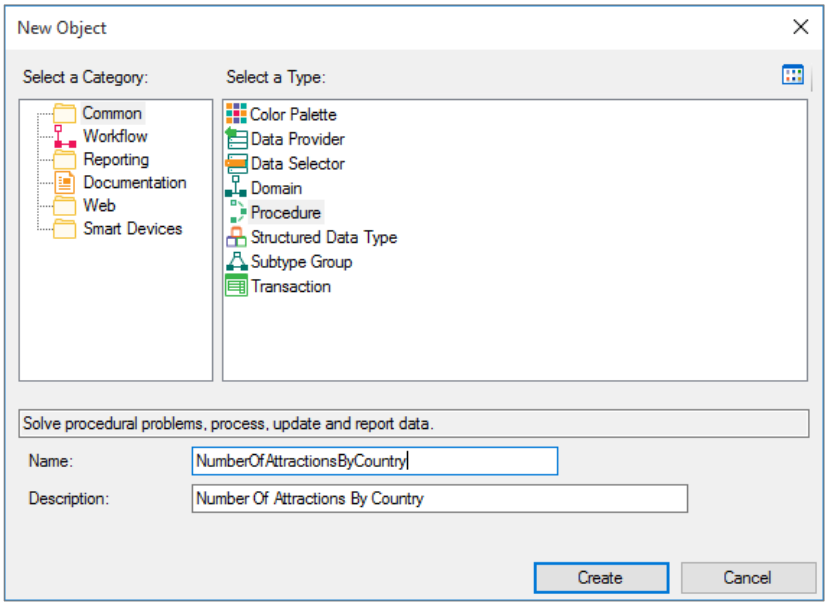
Knowledge Base

- The calculation is only accessed by the object that has it defined.
- They are the equivalent of a function that returns a value.

The only requirement that hasn't been met is that for each country, its number of attractions must be printed. How can we implement this?

Let's see an example...

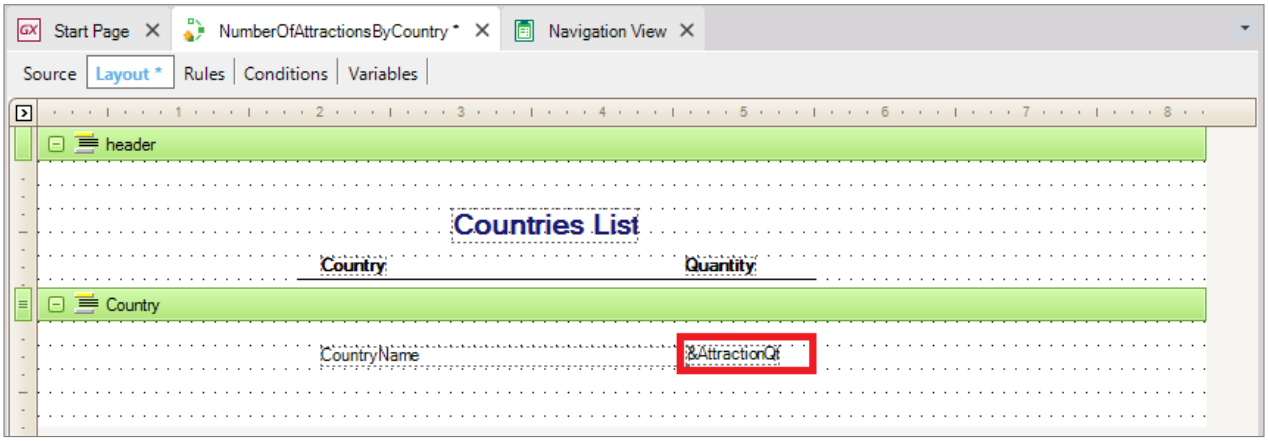
- We're asked to create a list of countries with the number of attractions of each one of them.
- We create a procedure object…



We will calculate the number with an "inline" formula... and display the result.

We define its layout

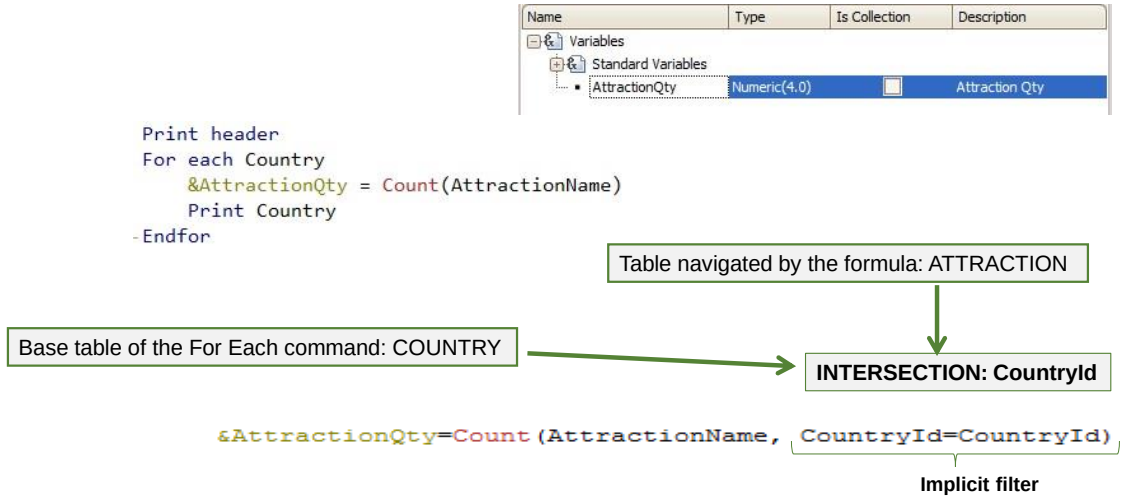
- ...with an &AttractionQty variable to show the number of attractions.



We will create a variable to assign it the result returned by a Count formula, and then we will show the variable in the printblock. We create a new variable called AttractionQty, of Numeric(4) type.

Remember that variables are memory spaces which are local to the object where they are created. This means that they only exist while this object is being executed. When the execution ends, they disappear.

We define an inline formula



Let's return to the Source and within the For Each command, right before printing, to the variable: &AttractionQty... we assign the result of a Count formula... that will count attractions.

Between the brackets we need to include an attribute that lets GeneXus know the table from which we want it to count. Since it belongs to ATTRACTION, we will choose an attribute that we know is included in that table. For example, AttractionName.

Let's examine this code to see how it works. Formulas determine the table that will be navigated by the attribute(s) referenced between brackets. In this case, we have included the AttractionName attribute because we want to count attractions and GeneXus understands exactly that. However, the attractions we want to count are not all the attractions in the table. Instead, we want to count the attractions of the country where we're positioned in every time the For Each command is executed. If the formula wasn't inside a For Each command, or if we weren't already positioned in a table, all attractions would be counted. But since the formula is created inside a For Each command, that is to say, it is located in a context where a table is being run through (the countries table), it will be influenced by that context. Thus, not ALL the attractions in the table will be counted. Instead, only those related to the country being processed in each iteration of the For Each command will be counted.

In other words: GeneXus determines the table that must be navigated by the formula according to its attributes, and then looks for a relationship between that table and the For Each command base table (including its extended table).

In this case, there is a common attribute between both navigations: CountryId. So, for each country found by the For Each command, its attractions are counted.

We run the procedure and see how the formula is calculated

Countries List

Country	Quantity
Brazil	1
France	3
China	2
United States	1

If we have already added the Matisse Museum in France and the Forbidden City in China to the attractions:
The For Each command is executed and for the first country, Brazil, the attractions with the same CountryId are counted...
Next, the For Each command moves on to the next record, 2- France, and counts the attractions in CountryId = 2... It finds three.

Next, it iterates the following record, 3 – China, and counts 2 attractions. Lastly, only one attraction is found for the United States. In sum, it works as if we had defined this explicit filter condition in the formula.

Where one CountryId belongs to the table navigated by the formula and the other belongs to the For Each command table. It doesn't have to be written because GeneXus detects it automatically. In addition, it will realize that it is convenient to run through the table to be navigated by the formula by ordering it by that filter attribute. Otherwise, as we can see in this example, for each country it will have to run through the entire attraction table to count the corresponding ones, every time.

All these inferences made by GeneXus are shown in the navigation list.
As we can see, it indicates that the For Each command will run through the Country table, and will count the attractions in each country. To do so, it will have to navigate the Attraction table by CountryId. Now we only need to add to the Country printblock the &AttractionQty variable calculated as first instruction in the For Each command body. In this way, right after the variable receives the number of attractions, it is displayed next to the country name.

We add this. Now we run the report. Before running it, we set the corresponding properties to issue it in a web environment and PDF format. If we press here we can see, of all the properties, only the ones we modified. The rule output_file has to be included. If we don't remember it, we can look for it from the Insert option in the menu, or from the Toolbox, by dragging it. And replacing the file name... we give it the same name as the object... and a format... "pdf".

We save and press F5.
In this way, we have met the requirement. As we have seen, to determine the table to be navigated by the formula, the attributes referenced in the For Each command were not taken into account. Only the attributes included in the formula definition were considered.
Likewise, to determine the table to be navigated by the For Each command, all the For Each command attributes were taken into account except for the attributes referenced in the formula.

Another use of the inline formula

- Requirement: to list all the countries that have more than 2 attractions to visit.
- We will add a Where clause to the For Each command and use the formula for the condition:

```
Print header
For each Country
    Where Count(AttractionName) > 2
    &AttractionQty = Count(AttractionName)
    Print Country
Endfor
```

Suppose that now we're asked to list all the countries that have more than two tourist attractions. We save this procedure with another name... The condition that we're asked to comply with to show the countries is that they have more than two tourist attractions. So, we add a Where clause to the For Each command...where we want to filter those countries whose number of attractions: is greater than 2.

For those that comply with this condition, we will keep printing the same data: country name and number of attractions (that will be more than two).

At runtime...

Countries List

Country	Quantity
France	3

Remember that:

- The formulas that we've seen before (Sum, Average, Max and others) can also be used as inline formulas.

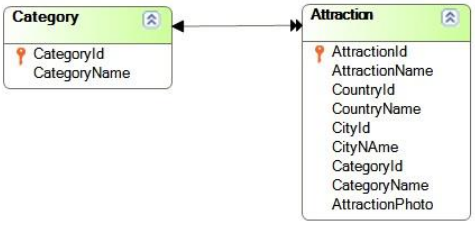
DEMO

Since this procedure was based on the other, everything necessary is configured to print it as a PDF, so we run it and see the result.
Only France is displayed, with three attractions as we expected.

This completes two examples of use of "inline" formulas to easily obtain calculations. In this example we only talked about the Count formula, but all the formulas that we've studied could have been used as inline formulas; for example, Sum, Average, Max, among others.

Inline formulas + Unique clause

It also allows joining data with inline formulas.



It returns the list (with no repeated elements) of all categories, each one with the corresponding number of attractions recorded.

```
Print Title
For each Attraction
     Unique CategoryId
     &Quantity = Count(AttractionId)
     Print Categories
Endfor
```

Base table of the For each command: ATTRACTION

Table navigated by the formula: ATTRACTION

We have previously seen the concept and an example of use of the Unique clause within a For Each command.

We had defined a list showing the categories that have registered tourist attractions, but without repeating the names of those categories. We will now add one more requirement: Next to each category name we want to see its number of attractions.

To calculate the number of attractions, we now declare the following Inline formula &Quantity=Count(AttractionId)

Note that the base table of the For Each command is ATTRACTION, the same as the base table of the Inline formula.
Therefore, the Count formula will add from the context an implicit condition in its evaluation: It will count all attractions for the attribute declared in the Unique clause.

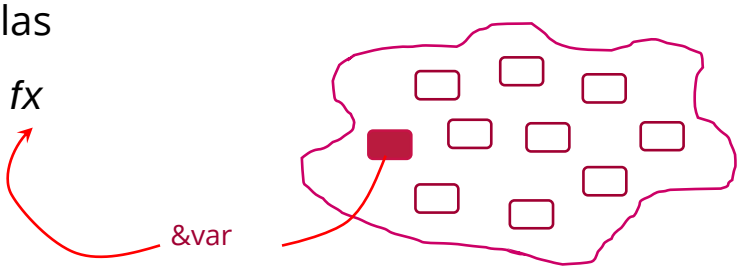
The navigation shows that the formula will count all attraction instances for the declared (Given) CategoryId attribute in the Unique clause.

Global Formulas vs. Inline Formulas (local)

- Inline formulas can use variables from the object where they are stated.

Inline(or **local**)Formulas

&Variable = *fx*



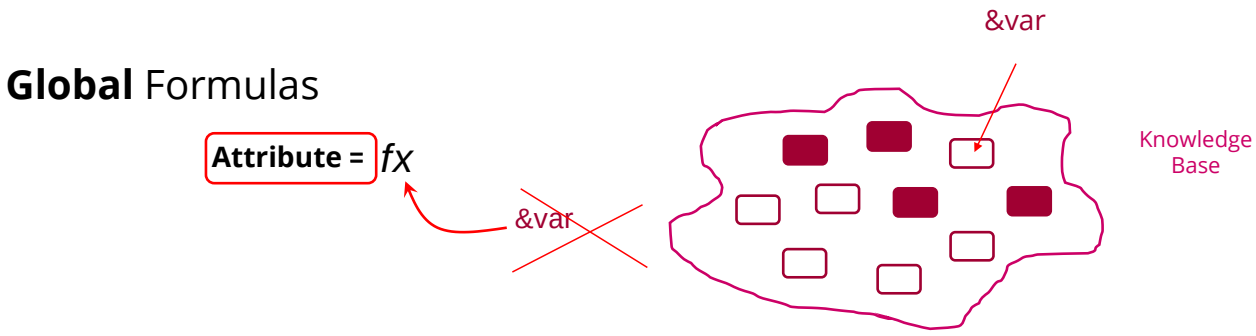
e.g. &AttractionsQty = Count(AttractionName)

It should be taken into account that since inline formulas are only calculated in the object in which they are written, their calculation can include variables defined in that object. For example...

This doesn't happen in global formulas, where variables can't be used to make the calculation because they are attributes that can be used in any object, and variables only have local scope.

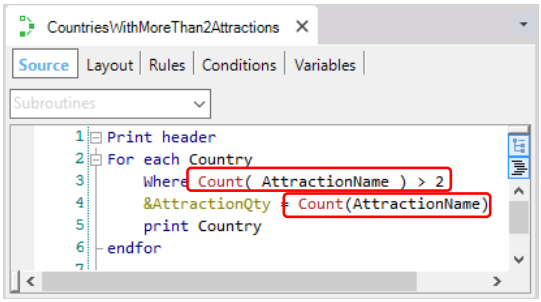
Global Formulas vs. Inline Formulas (local)

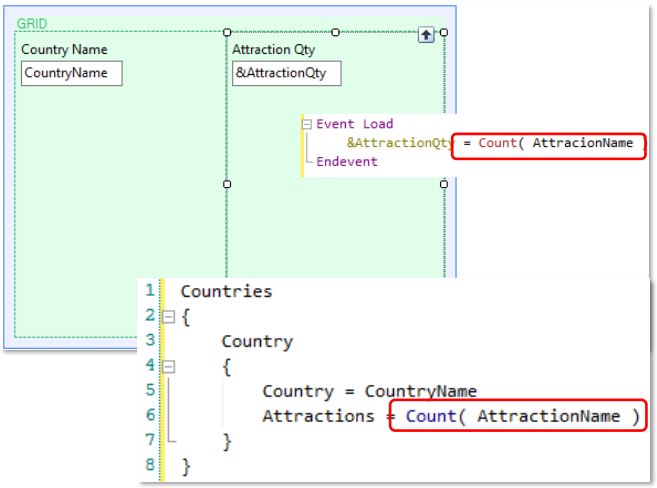
- Global formulas don't allow using variables for the calculation because they are attributes that can be used in any object, and variables only have local scope.



In sum...

- An “inline” formula is a formula that we state as a specific instruction within a certain piece of code, such as a procedure Source, web panel event, Data Provider Source, etc.





In summary:

An “inline” formula is a formula stated as a particular instruction in a certain code; for example, in a procedure Source, in a web panel event, in a Data Provider Source, and so on. The formula is only known in the object where it was stated.

That's why it is also called local formula. It is calculated when the object is executed and then its value disappears. They are different than global formulas (stated for attributes in transactions), which are calculated every time an attribute value is queried inside any object at runtime.

In sum...

- The formula is only known in the object where it was defined.
- For this reason, we also call it local formula. It is calculated when the object is run and then its value disappears.
- They are different from global formulas (those stated for attributes in transactions), which are calculated every time the value of an attribute within any object is queried at runtime.

GeneXus™

training.genexus.com
wiki.genexus.com