Defining Attributes as Formulas

GeneXus™

GeneXus

average(attribute7)

attribute1 + attribute2

attribute3 – attribute4

Formulas

count(attribute6)

sum(attribute5)

max(...)

min(...)

Many times we need the application to solve a calculation involving values from certain attributes, constants, and/or functions.

For these cases, GeneXus provides Formulas.

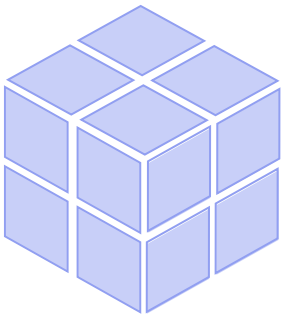
Two types of Formulas:

Global Formulas

Atribute = fx

Local (or inline) Formulas

Knowledge Base



Formulas can be defined in two ways:

GLOBALLY: the calculation will be available throughout the Knowledge Base.

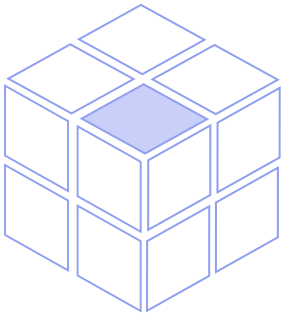
Two types of Formulas:

Global Formulas

Local (or inline) Formulas

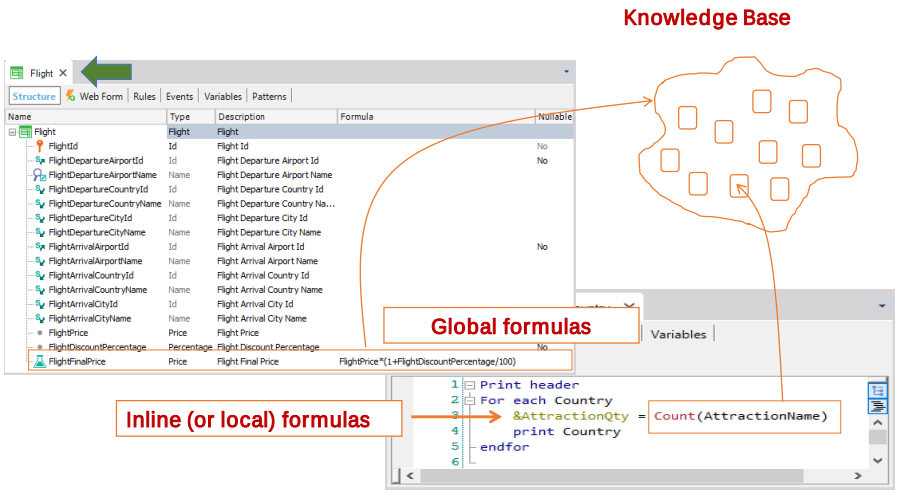
&Variable = fx

Knowledge Base



LOCALLY or INLINE: in this case, the calculation will be available only in the object in which it has been defined.

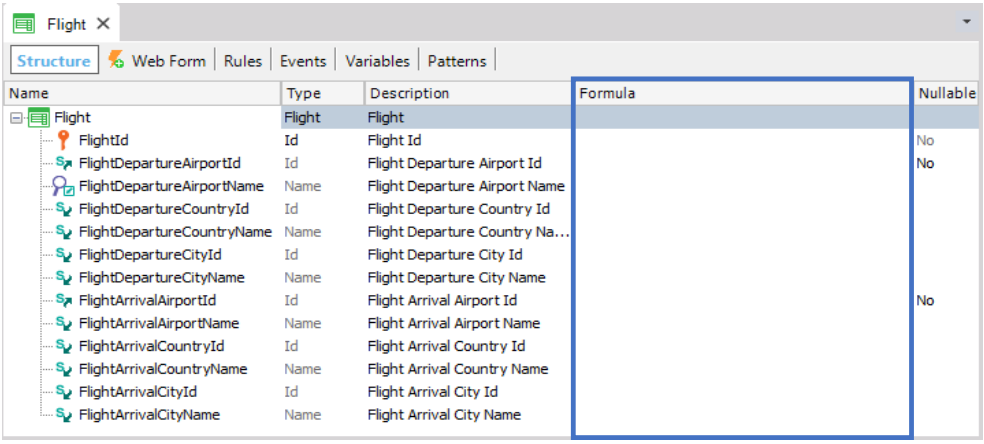
Two types of Formulas:



GeneXus

Global formulas

It is a calculation we define in association with an attribute; from then on it will be "virtual."



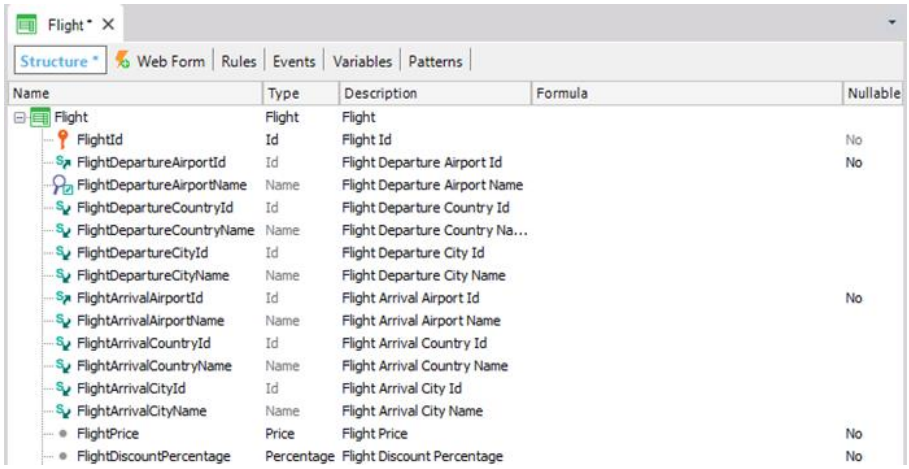
Name	Type	Description	Formula	Nullable
Flight	Flight	Flight		
FlightId	Id	Flight Id		No
FlightDepartureAirportId	Id	Flight Departure Airport Id		No
FlightDepartureAirportName	Name	Flight Departure Airport Name		
FlightDepartureCountryId	Id	Flight Departure Country Id		
FlightDepartureCountryName	Name	Flight Departure Country Na...		
FlightDepartureCityId	Id	Flight Departure City Id		
FlightDepartureCityName	Name	Flight Departure City Name		
FlightArrivalAirportId	Id	Flight Arrival Airport Id		No
FlightArrivalAirportName	Name	Flight Arrival Airport Name		
FlightArrivalCountryId	Id	Flight Arrival Country Id		
FlightArrivalCountryName	Name	Flight Arrival Country Name		
FlightArrivalCityId	Id	Flight Arrival City Id		
FlightArrivalCityName	Name	Flight Arrival City Name		

Let's start by explaining what a global formula is and how it is defined.

A global formula is a calculation defined in relation to an attribute.

Note that transaction structures contain a column titled "Formula."
If a calculation is defined in this column for an attribute, GeneXus will understand that this attribute is virtual; that is to say, it will not have to be physically created as a field in the associated table, because the attribute value will be obtained by making the calculation we've indicated.

Adding a global formula



Name	Type	Description	Formula	Nullable
Flight	Flight	Flight		
FlightId	Id	Flight Id		No
FlightDepartureAirportId	Id	Flight Departure Airport Id		No
FlightDepartureAirportName	Name	Flight Departure Airport Name		
FlightDepartureCountryId	Id	Flight Departure Country Id		
FlightDepartureCountryName	Name	Flight Departure Country Na...		
FlightDepartureCityId	Id	Flight Departure City Id		
FlightDepartureCityName	Name	Flight Departure City Name		
FlightArrivalAirportId	Id	Flight Arrival Airport Id		No
FlightArrivalAirportName	Name	Flight Arrival Airport Name		
FlightArrivalCountryId	Id	Flight Arrival Country Id		
FlightArrivalCountryName	Name	Flight Arrival Country Name		
FlightArrivalCityId	Id	Flight Arrival City Id		
FlightArrivalCityName	Name	Flight Arrival City Name		
FlightPrice	Price	Flight Price		No
FlightDiscountPercentage	Percentage	Flight Discount Percentage		No

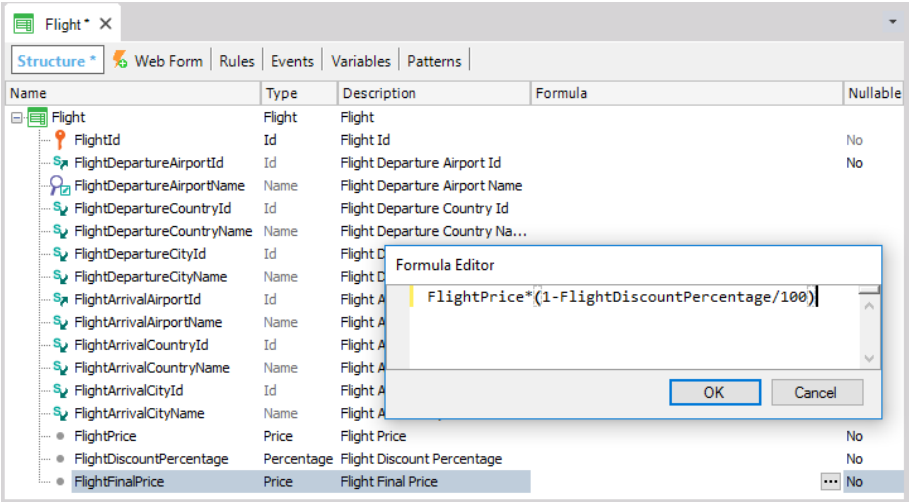
Let's see this with an example.

First, we will define a new attribute in the Flight transaction, in order to store the price of each flight.
So, we add the FlightPrice attribute.
And create the Price domain.

We save.

Now we will add another new attribute in the same transaction to store the discount applied to each flight. We call it: FlightDiscountPercentage... its data type will be a domain also called Percentage, numeric of 3 digits.

Adding a global formula

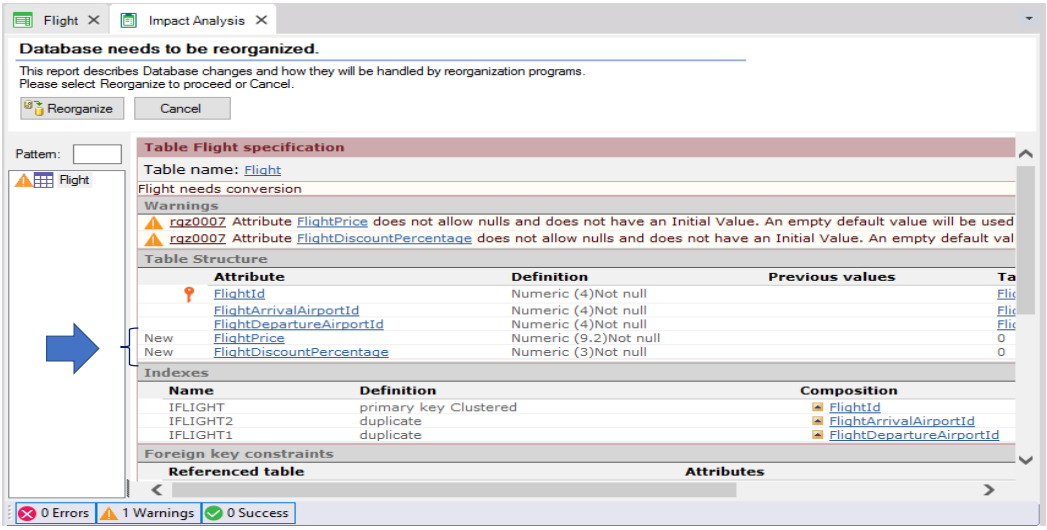


Lastly, we will add another attribute called FlightFinalPrice, based on the Price domain; this time, the attribute will be defined as a global formula.

To this end, in this attribute's “Formula” column, we will define the calculation necessary, so that it is always run and this attribute provides “the flight's current price”; that is to say, the price after subtracting the discount percentage stored in FlightDiscountPercentage from FlightPrice. So, in this formula column we will type the corresponding calculation.

Note that in this window we only have to type the calculation, not the assignment.

Impact Analysis



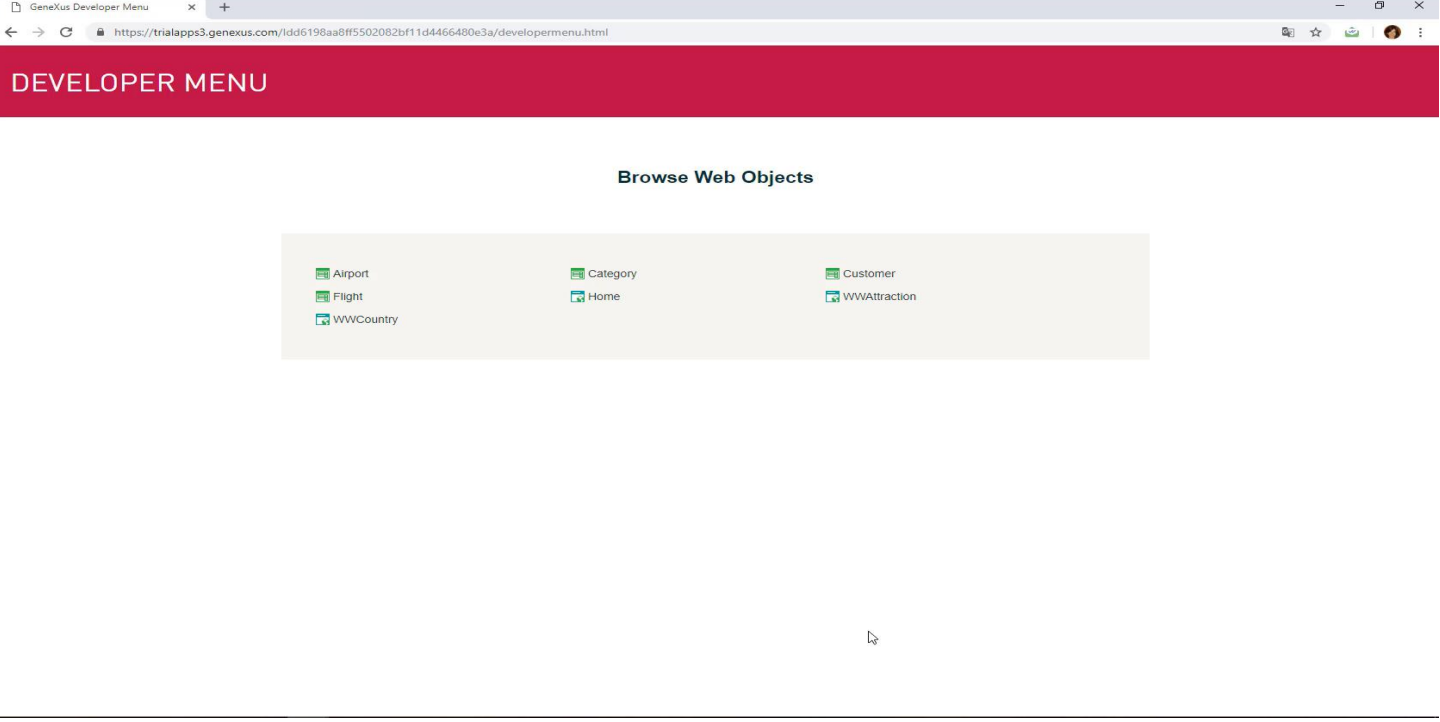
Now we press F5, which automatically saves all pending actions... and see what happens.

In the Flight physical table, only 2 fields are created, even though we have defined three new attributes in the transaction structure.

Due to the fact that the formula column contains a definition, this attribute is not added in the physical table.

Because the attribute is defined in the Knowledge Base with an associated formula, GeneXus can calculate its value. Also, in every object where this attribute is included, the calculation will be made and the result will be shown.

We reorganize... and see the application at runtime.



We run the Flight transaction, query flight number 1, and in this form we see the three new attributes created:

the flight price enabled for us to enter it,
the discount percentage, also enabled for us to enter it,
and the final price, disabled because it is the attribute defined as a formula, and its value is not entered; instead, it will be calculated and displayed.

Every attribute defined as a global formula will be read-only, and it will not be possible to enter a value for it. This happens because the attribute obtains its value from the associated calculation, which is run every time the attribute is used.

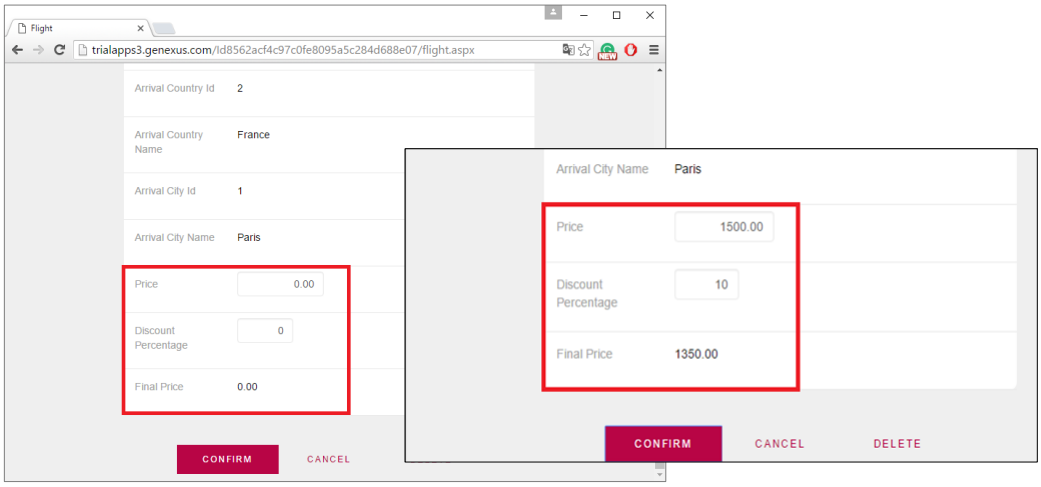
For this reason, there isn't a field in the physical table to store this attribute value. For this reason, there's no need for it to be editable.

We will enter a price for this flight, and a discount percentage: 10%.

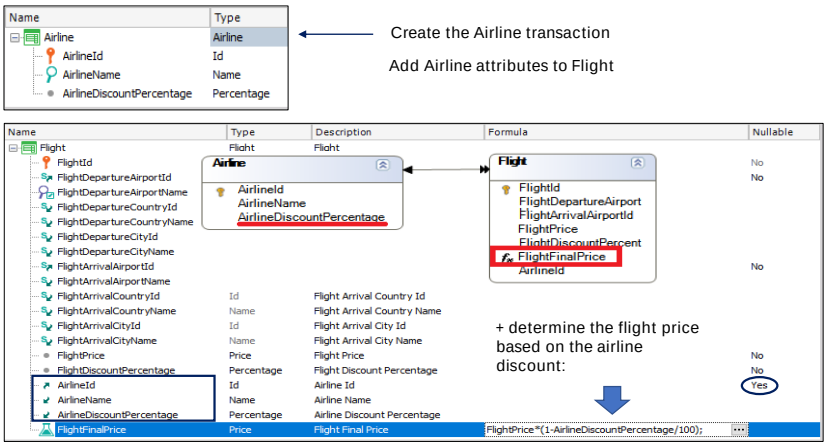
After leaving the field, we see that the formula is immediately run, and the final price of the flight is displayed with the discount applied.

Running the application with a global formula

FlightId 1



Using extended table attributes in a global formula



Let's return to GeneXus.
In this way, we have defined a global formula attribute.

Only attributes can be defined as global formulas in the way we've seen, using the Formula column in the transaction.

Something important to remember is that, even though in the example we have only used attributes from the transaction's own associated table -that is to say, its base table-, attributes from the extended table can also be used.

Let's see it.
We will create a new transaction called Airline to record the airlines.

We type:
- AirlineId
- AirlineName and...
- AirlineDiscountPercentage, to record the discount made by the airline for all its flights.

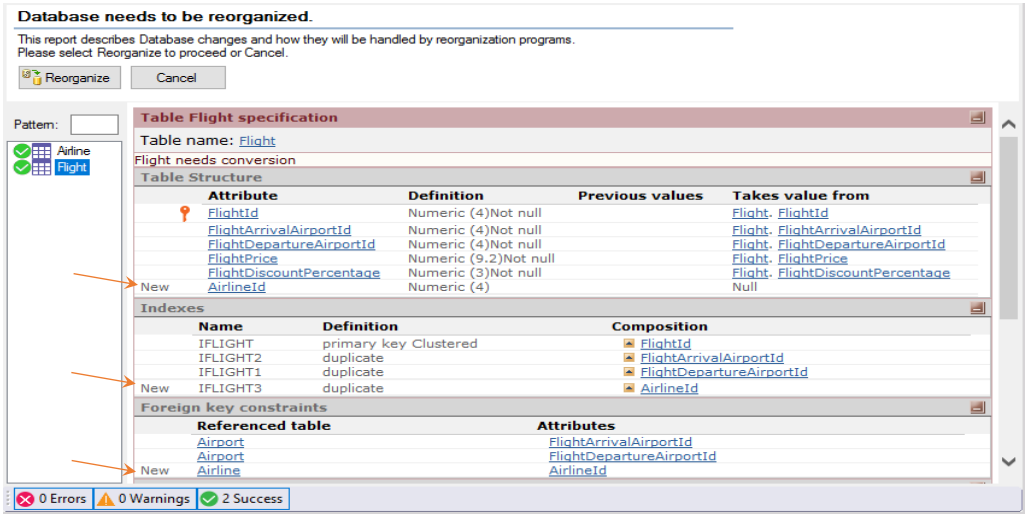
We save. Now we open the Flight transaction to assign an airline to every flight.

So, we add the AirlineId attribute, which here will be a foreign key... and change the value of its Nullable property to Yes... In this way, we can avoid indicating the flight's airline at this stage, because we still don't have any airlines recorded.

Later on we can change again the value of this Nullable property to No, so that it is mandatory to indicate the airline when entering or changing the details of a flight.
In addition, we add the AirlineName and AirlineDiscountPercentage attributes to also view this data in the form.

Now we will change the definition of this formula to have it calculate the final price of the flight by applying it the generic discount of the airline, instead of applying the discount of the flight itself.

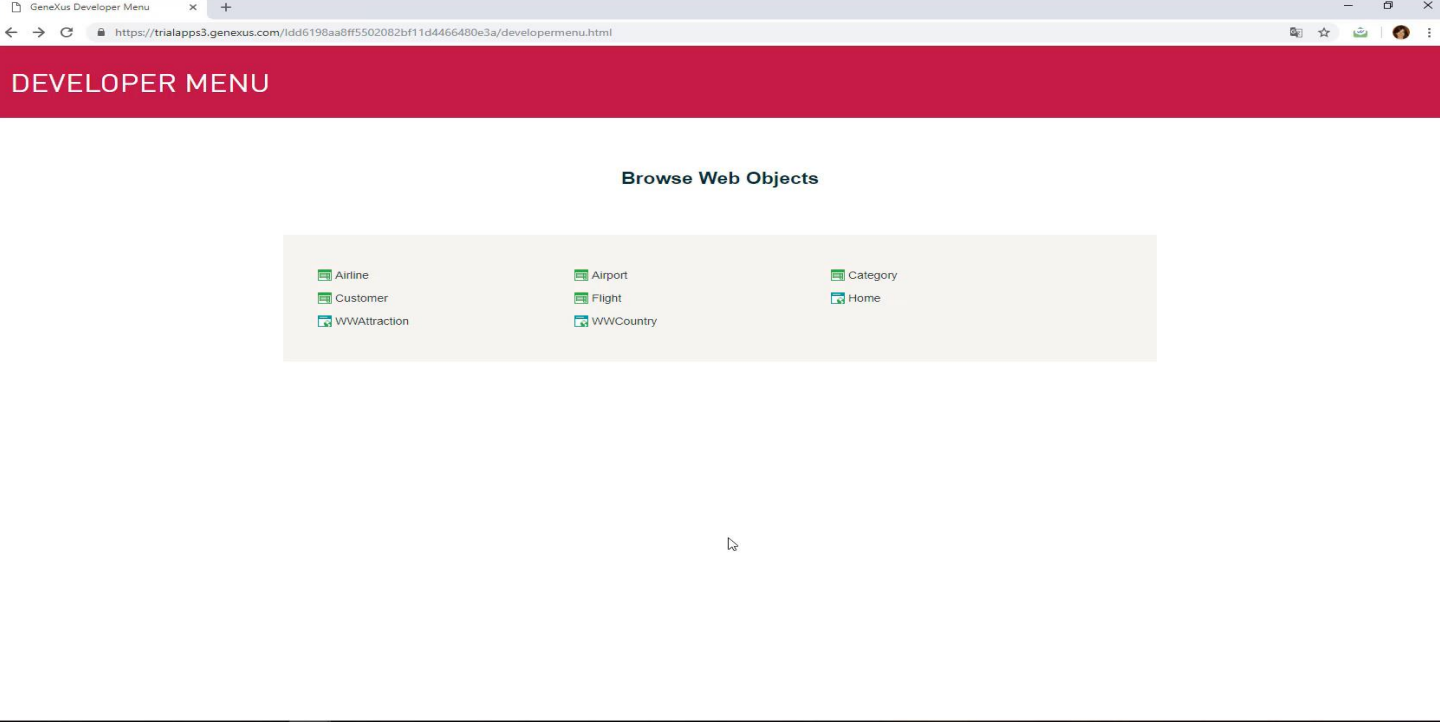
Impact Analysis



We press F5...

As we can see, the Airline physical table will be created with the three attributes defined, and in the Flight table the AirlineId foreign key will be created.

So, we reorganize and run...



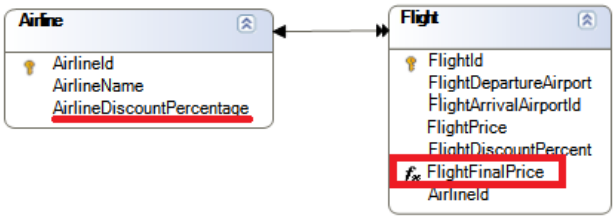
We run the Airline transaction and enter an airline, called TAM, with a 30% discount.

Now we will assign this airline to a flight.

So, we open the Flight transaction, flight number 1, and associate it with airline number 1...

The new final price of the flight, which is a global formula attribute, is calculated.

Testing the formula at runtime



Now it involves the discount percentage of the airline, which is an attribute of the extended table of the Flight base table.

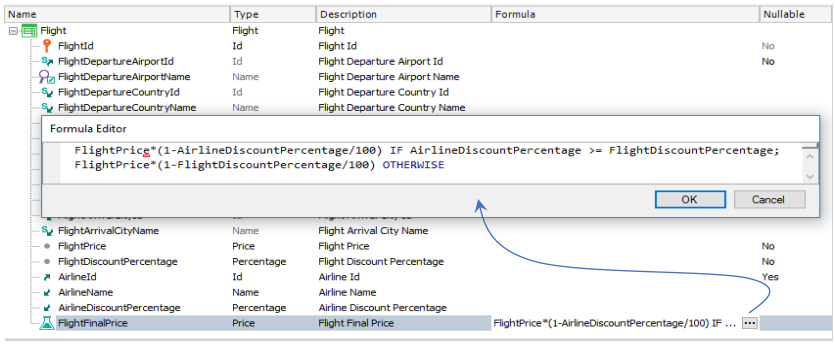
GeneXus

Adding conditions to the formula

Attribute = *fx* $\begin{cases} \text{expresión}_1 \textbf{ if } \text{condición}_1; \\ \text{expresión}_2 \textbf{ if } \text{condición}_2; \\ \ldots \\ \text{expresión}_n \textbf{ if } \text{condición}_n; \end{cases}$

We haven't mentioned yet that formulas can evaluate conditions, and the result can be calculated in different ways depending on whether these conditions are true or false.

Adding conditions to the formula



The result will be calculated in one way or another, depending on which condition is true

Let's see this.
To do so, we click on this button to edit the formula.

And we define that the highest discount percentage must be taken into account to calculate the final price of the flight, in order to make the best discount possible.

With this definition, if the airline has a higher discount for all its flights than the discount percentage of the flight itself, the airline discount will be considered to make the calculation.

GeneXus

We also add a default condition

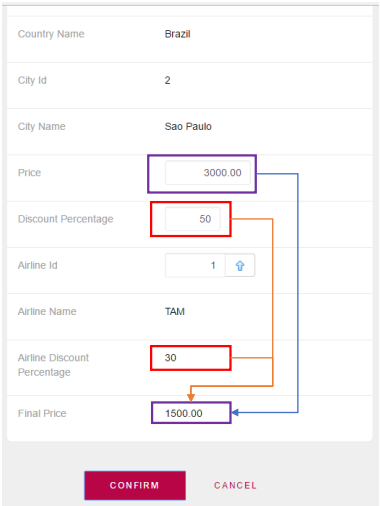
Attribute = *fx*

expresión$_1$ **if** condición$_1$;
expresión$_2$ **if** condición$_2$;
...
expresión$_n$ **if** condición$_n$;

Otherwise:

the discount percentage of the flight itself is used to make the calculation.

Testing the new definition of the formula



Note that formulas are written as expressions, so they end with a semicolon. To calculate the formula, GeneXus keeps the first expression that meets the condition. If no condition is met, and an otherwise clause has been added, it uses this one.

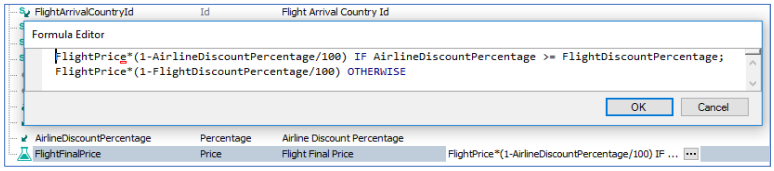
We move the Final Price attribute to the end of the list, so that the information is more clearly organized.

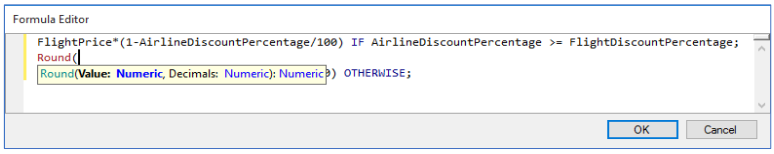
We press F5, and run the Flight transaction. In the first flight we set its discount percentage to be higher than the overall discount percentage of the airline; for example, 50%.

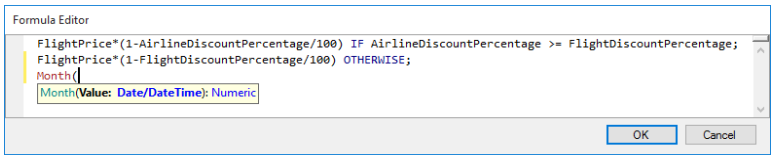
We exit this field and go to the airline field, so that it has everything necessary to calculate the formula.

The final price of the flight was calculated using the highest discount.

Horizontal formulas





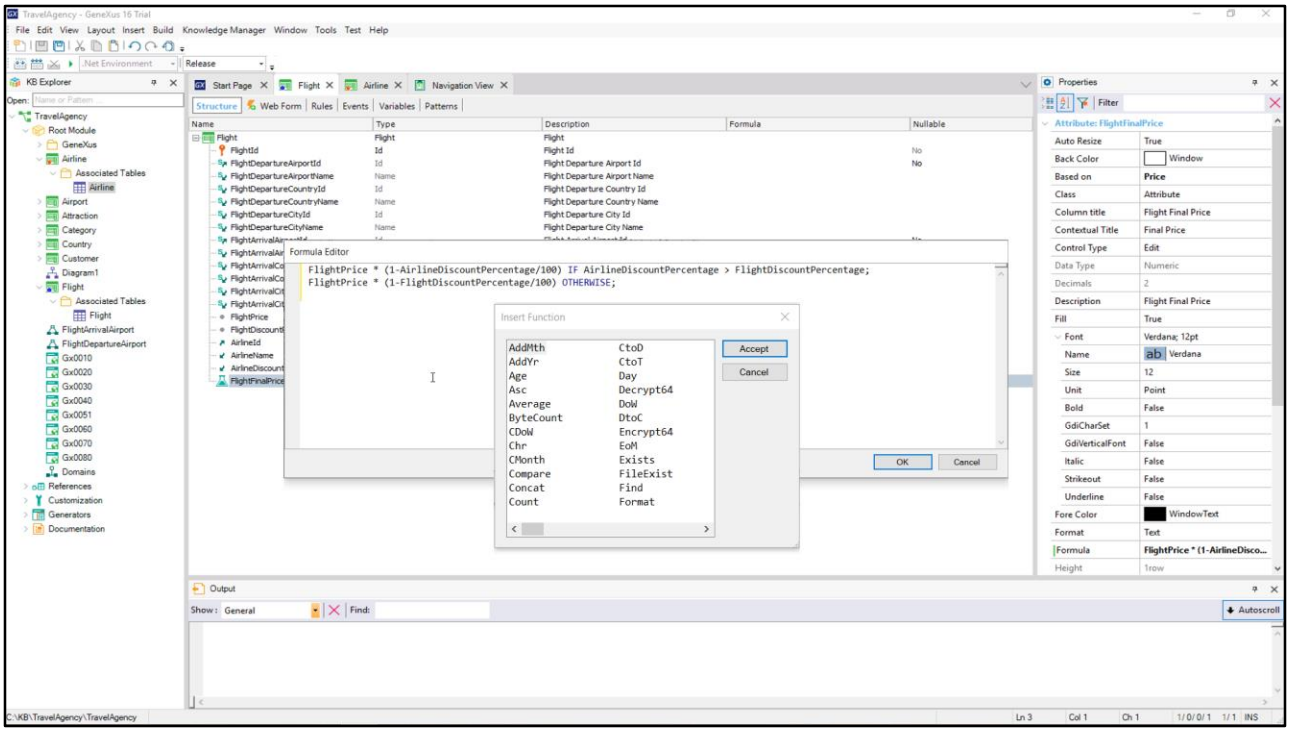


Let's return to GeneXus.

As we've seen, formulas can contain several lines followed by IF, and may contain a last line with OTHERWISE in case none of the above conditions are met.

In turn, even though in this example each result is obtained through a calculation, functions applied to attributes or expressions can also be used, such as Round, to obtain a rounded result.

Also, Month can be used to obtain the month of a date, etc... even a user-defined procedure can be called to return a value.



If we right-click on the formula editing dialog and choose the Function option... a new dialog will be displayed showing all the functions we could insert.

GeneXus Developer Menu | Search

wiki.genexus.com/commwiki/servlet/hsearch?functions,1

GeneXus Community Wiki MENU Login Register

functions All

Recents Wiki Home

Results 1 - 12 of about 1090 for functions. (0,06 seconds). Search tips

FTP Functions

FTP **Functions** anonymous . This **function** allows you to perform some actions on the servers via FTP (File Transfer Protocol) technology. You... **Function** GxFtpPut GxFtpStat **Function** GxFtpStat Category:Object **functions** Category:GeneXus X Help Category:GeneXus X Evolution 1 Help Category... Evolution 1 Help Object **functions** GeneXus X Help Closed anonymous dmarquez Public FTP **Functions**...

TYPE: Article LAST MODIFICATION: 31 July 2020 BY: anonymous OWNER: dmarquez STATUS: Closed

Functions in Procedures

Functions in Procedures anonymous . Add days syntax Add Days GXMLines **Function** GxMLines StrSearch **function** StrSearch AddMth **function** Add... **functions** Category:Object **functions** Category:GeneXus X Help Category:Procedure object Category:GeneXus X Evolution 1 Help Category:GeneXus X... Object **functions** Procedure object GeneXus X Help Closed anonymous dmarquez Public **Functions** in Procedures...

TYPE: Article LAST MODIFICATION: 31 July 2020 BY: anonymous OWNER: dmarquez STATUS: Closed

Functions in Transactions

Functions in Transactions anonymous . Add days syntax Add Days Hour **Function** Hour TAdd **Function** TAdd (Time adding) AddMth **function** Add... **Function** SubStr (SubString) See Also Standard and non standard **functions** Standard **Functions** Category:Object **functions** Category:GeneXus X Help... GeneXus X Evolution 3 Help GeneXus X Evolution 2 Help GeneXus X Evolution 1 Help Object **functions** GeneXus X Help Transaction object Closed...

TYPE: Article LAST MODIFICATION: 31 July 2020 BY: anonymous OWNER: dmarquez STATUS: Closed

Object functions

Object **functions** anonymous . GeneXus offers **functions** that can be used inside the different GeneXus objects sections... anonymous ceciliaf Public Object **functions**... to a variable or an attribute, or just evaluated or shown, etc. You can select Insert/**Function**... being in a specific object section...

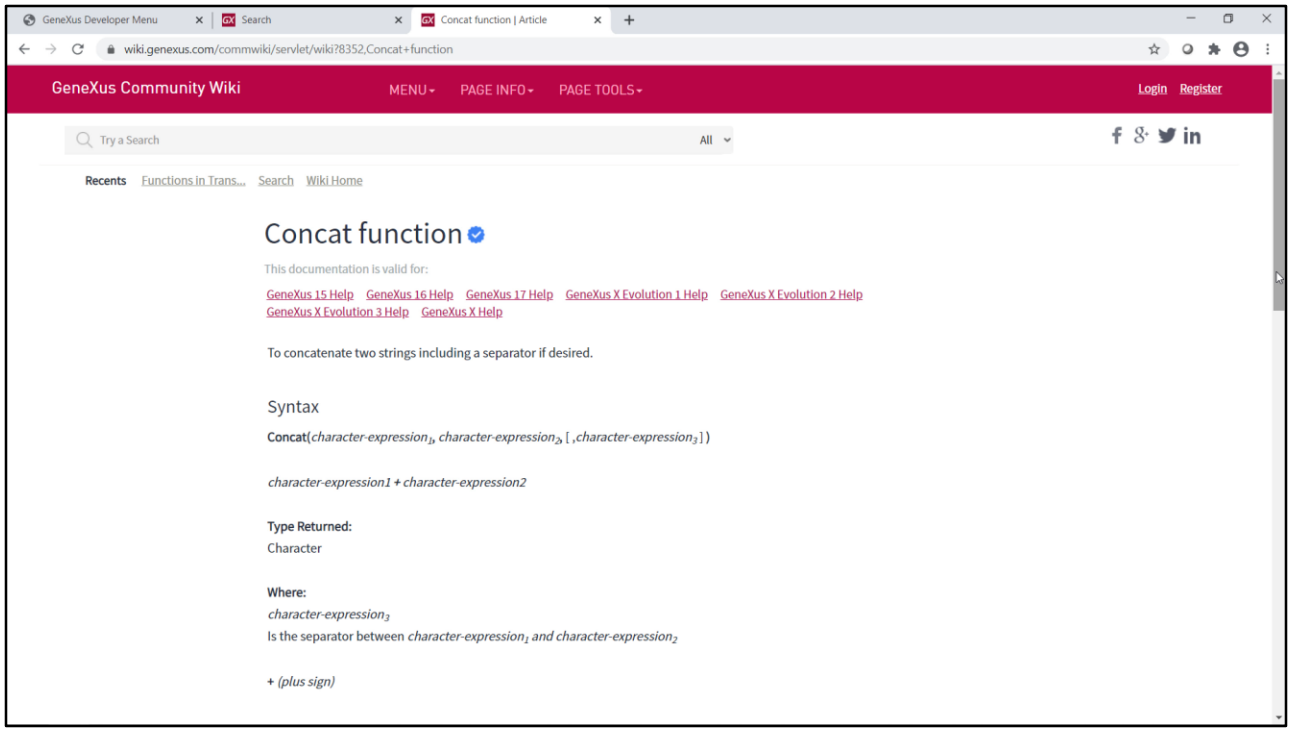
TYPE: Category LAST MODIFICATION: 31 July 2020 BY: anonymous OWNER: ceciliaf STATUS: Closed

Common dialog functions

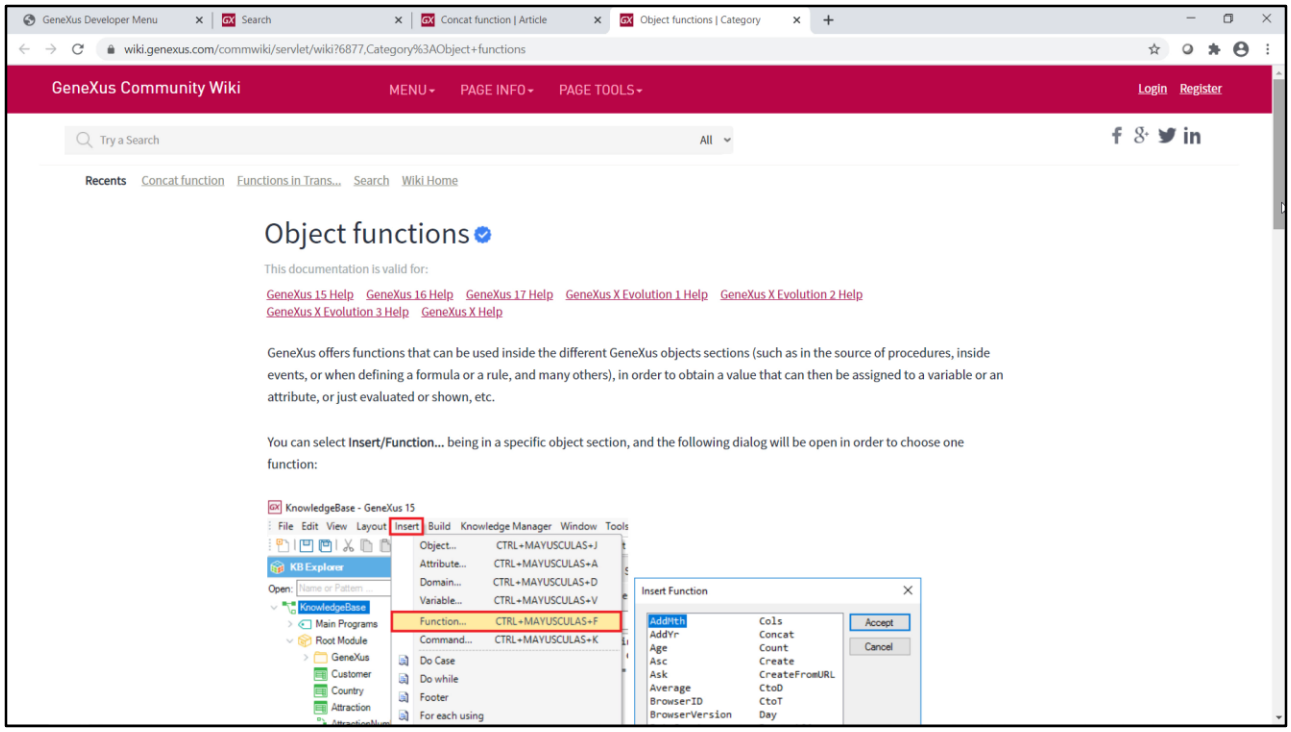
Common dialog **functions** anonymous . Common Dialogs are a set of dialogs used by different applications and are available in Windows. To be able to use this Common Dialogs, a set of standard **functions** have been made available. GxCalculate **Function** GxCalculate GxSelFile **Function**... **Function** GxSelColor GxSelPrn **Function** GxSelPrn GxSelDir **Function** GxSelDir Category:GeneXus X Evolution 1 Help Category:Object **functions**...

TYPE: Article LAST MODIFICATION: 31 July 2020 BY: anonymous OWNER: dmarquez STATUS: Closed

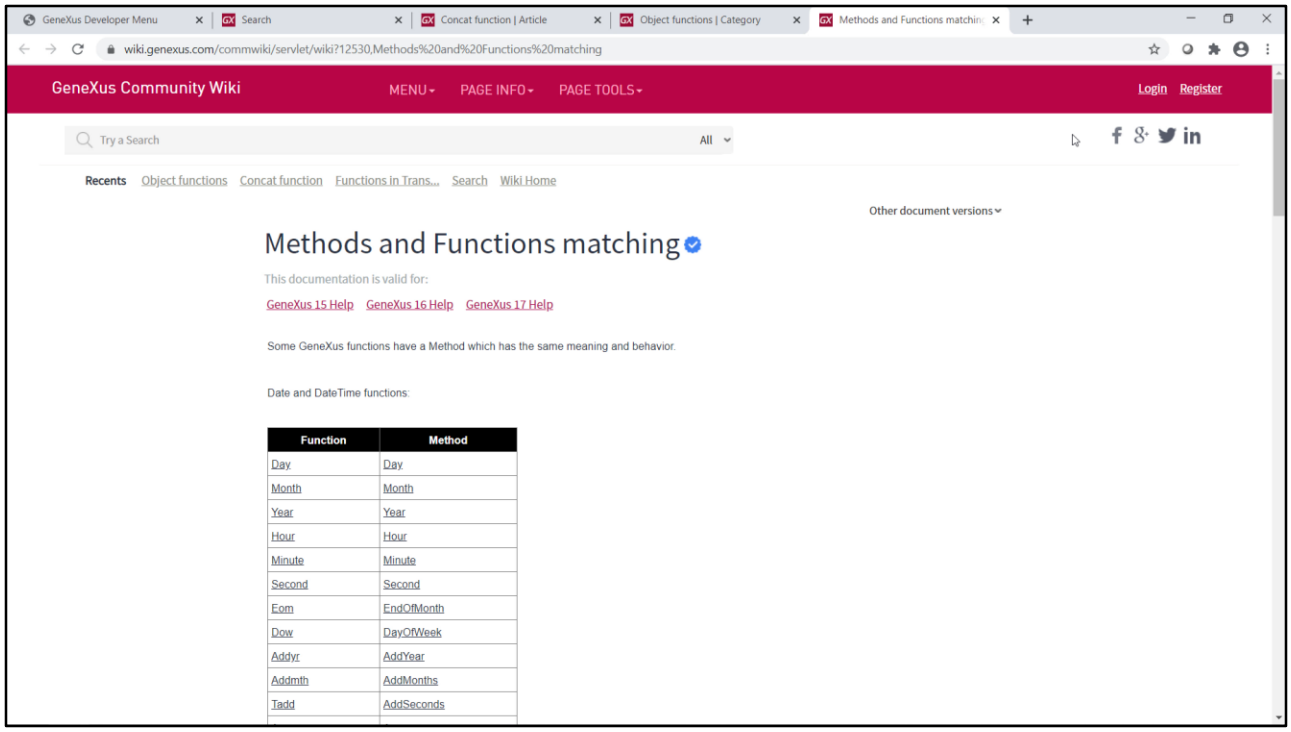
Let's go to the GeneXus Wiki to search for information about this: at wiki.genexus.com, in the search box we type "functions." Several entries are displayed with information about what we are looking for.



For example, if we click on “Functions in Transactions,” we will see all the functions that we can use in transactions, and if we click on any of them, we will get their complete information: use, syntax, types of data returned, examples, etc.



Now, let's look at the “Object Functions” entry: it explains how to access the functions corresponding to a certain object in which we are positioned, and gives an example that uses the Month() function to obtain the list of clients who have a birthday in the current month.



The entry “Methods and Functions matching” shows functions that have a method with the same meaning and behavior. We have already used the IsEmpty method to control whether a field is empty, but there are many others that we invite you to explore.

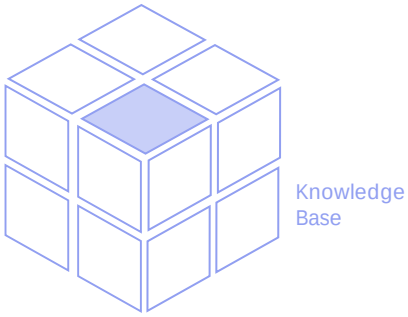
Aggregate formulas, used in:

Global Formulas

Attribute = *fx*

Local (or inline) Formulas

&Variable = *fx*



This type of formulas, which perform a calculation that is obtained from data of the record in which you are positioned (only one) and eventually from the associated records (by extended table), are usually called horizontal formulas.

Now we will see another type of formulas called Aggregate formulas.

We will explain them by defining examples of global formulas -that is to say, the corresponding calculations will be defined in relation to attributes, and therefore they will not be created as physical fields. As a result, our examples can also be assigned in another context, such as a variable, for example.
They may also be assigned locally in a certain section of an object (we will see this in another video).

GeneXus

Aggregate formulas: Count, Sum, Average, etc.

Example: Create a second level in the Flight transaction

Name	Type	Formula	Nullable
Flight	Flight		
FlightId	Id		No
FlightDepartureAirportId	Id		No
FlightDepartureAirportName	Name		
FlightDepartureCountryId	Id		
FlightDepartureCountryName	Name		
FlightDepartureCityId	Id		
FlightDepartureCityName	Name		
FlightArrivalAirportId	Id		No
FlightArrivalAirportName	Name		
FlightArrivalCountryId	Id		
FlightArrivalCountryName	Name		
FlightArrivalCityId	Id		
FlightArrivalCityName	Name		
FlightPrice	Price		No
FlightDiscountPercentage	Percentage		No
AirlineId	Id		Yes
AirlineName	Name		
AirlineDiscountPercentage	Percentage		
FlightFinalPrice	Price	FlightPrice*(1-AirlineDiscountPercentage/100) IF Airli...	
Seat	Seat		
FlightSeatId	Id		
FlightSeatLocation	Location		

We'll define this domain with enumerated values

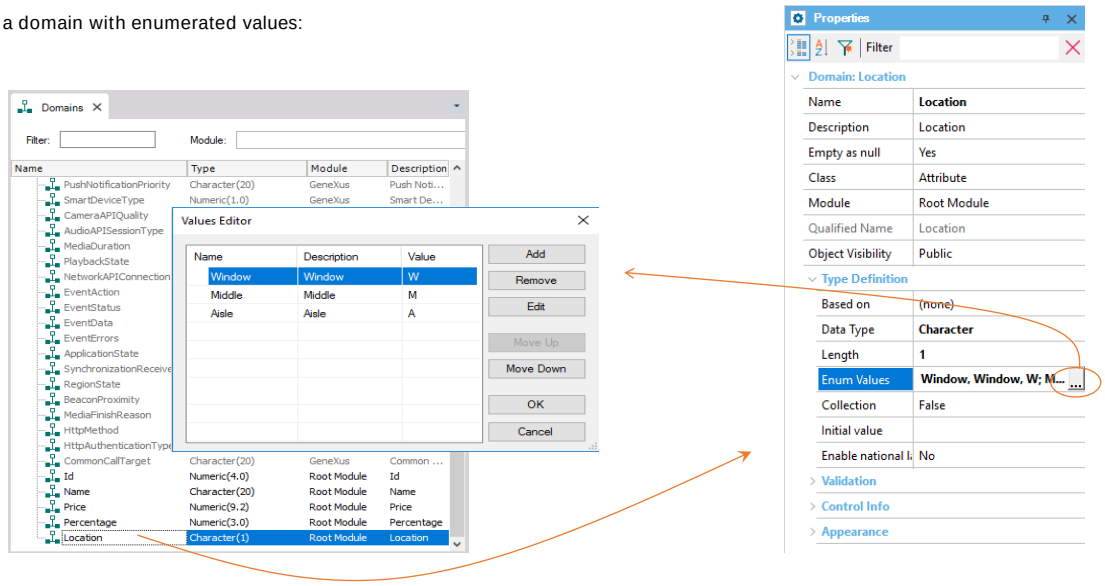
Now we will create a second level in the Flight transaction... and call it: Seat

As described by this level name, we will use it to record the seats available in the flight; for each seat, we will indicate if it is next to a window, an aisle, or in the middle. Next, we will need to know the total number of seats available in the flight.

We type a period and complete the attribute name: FlightSeatId
Now we create another attribute called FlightSeat Location... it will be a character of 1.

Aggregate formulas: Count, Sum, Average, etc..

Define a domain with enumerated values:



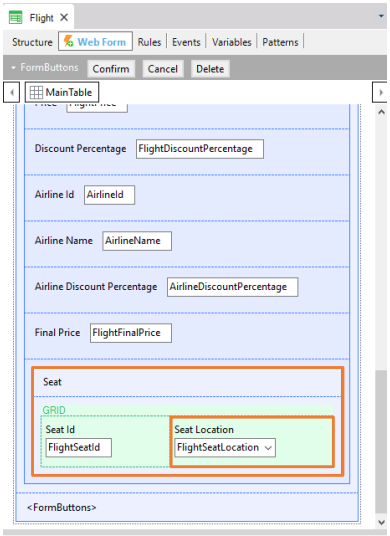
Now we edit the domains, to change a property of the Location domain that we've just created:

We locate the Enum Values property, and define the three values that this domain can take:

- Window... the character stored in this case will be “W”
- Middle... the character stored in this case will be “M”
- or Aisle... the character stored in this case will be “A”.

We click on OK.

Aggregate formulas: Count, Sum, Average, etc.

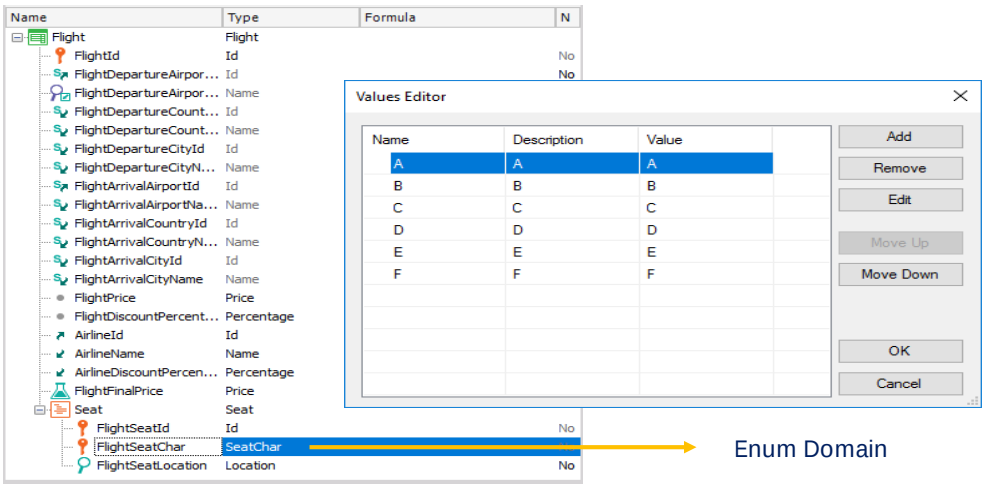


Look at the form of the Flight transaction. A grid has been added to enter the flight's seats, and for every seat we can indicate its position through a combo control.

This combo offers the values “window” “middle” or “aisle”, because they are the possible values defined for the domain of the FlightSeatLocation attribute.

Aggregate formulas: Count, Sum, Average, etc.

Change the Seat level key to better represent the seat. We want to identify it with a number + a letter from A to F.



Before pressing F5, let's look at the definition of the second level.

If the key is made up of FlightId plus FlightSeatId, for each flight, the seat numbers cannot be repeated. However, we need the seat number to be repeated, because it is identified by this number and a letter. In this way, we will have seats 1A, 1B, 1C, 2A, 2B, and so on.

So, we add a new FlightSeatChar attribute, and its domain will be SeatChar, a character of 1.

We will make this attribute part of the key to record the same seat numbers with different letters.

We will limit the possible letters from A to F, and to do so we will edit the SeatChar domain that we've just created...

We locate its Enum Values property and define the possible values:

In this case, the descriptions' values match the values stored. We click on OK.

Aggregate formulas: Count, Sum, Average, etc.

Add a new formula to count the number of seats:

Name	Type	Formula	Nullable
Flight	Flight		
FlightId	Id		No
FlightDepartureAirportId	Id		No
FlightDepartureAirportName	Name		
FlightDepartureCountryId	Id		
FlightDepartureCountryName	Name		
FlightDepartureCityId	Id		
FlightDepartureCityName	Name		
FlightArrivalAirportId	Id		No
FlightArrivalAirportName	Name		
FlightArrivalCountryId	Id		
FlightArrivalCountryName	Name		
FlightArrivalCityId	Id		
FlightArrivalCityName	Name		
FlightPrice	Price		No
FlightDiscountPercentage	Percentage		No
AirlineId	Id		Yes
AirlineName	Name		
AirlineDiscountPercentage	Percentage		
FlightFinalPrice	Price	FlightPrice*(1-AirlineDiscountPercenta...	
FlightCapacity	Numeric(4.0)	Count(FlightSeatLocation)	No
Seat	Seat		
FlightSeatId	Id		No
FlightSeatChar	SeatChar		No
FlightSeatLocation	Location		No

Now, to find out the maximum number of passengers allowed on a flight, according to the number of seats, we will define a new attribute in the first level. In its Formula column we will indicate the calculation consisting in counting the number of seats offered in the flight...

So, we create the FlightCapacity attribute, and its data type will be numeric of 4.

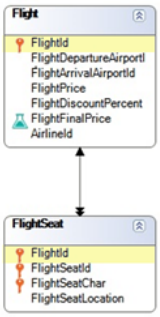
In its Formula column we type: Count... and between brackets we add an attribute that lets GeneXus know that we want to count the seats. To this end, we choose the FlightSeatLocation attribute that belongs to the transaction level containing the seats.

The Count formula will count data in memory or records of a table depending on the case. If we are inserting or updating a flight, the Count formula will count in memory the seats that the user has been recording.

FlightId	FlightDepartureAirportId	...	FlightPrice	AirlineId
1	1		1500	1
2	3	...	2500	2
3	1	...	1000	1
...	...	...	...	...

FlightId	FlightSeatId	FlightSeatChar	FlightSeatLocation
1	1	A	Window
1	1	B	Aisle
1	2	A	Window
1	2	B	Aisle
1	3	C	Middle
2	1	A	Window
2	1	B	Middle
3	...	...	...
...	...	...	...

count(FlightSeatLocation)



If the end user is not interacting with the transaction form, the Count formula will count the records in the FlightSeat table. If we are positioned on a given flight, GeneXus will only count the seats corresponding to that flight. That is, GeneXus automatically detects the relationship between the table where the formula attribute was defined and the table navigated by the formula, so to perform the calculation it will only take into account the related records. If no relationship is found, GeneXus will count all the records of the table navigated.

The attribute that is referred to inside the brackets of the formula gives GeneXus information about the level in memory that must be run through, or of the table that must be navigated to perform the calculation.

Aggregate formulas: Count, Sum, Average, etc.

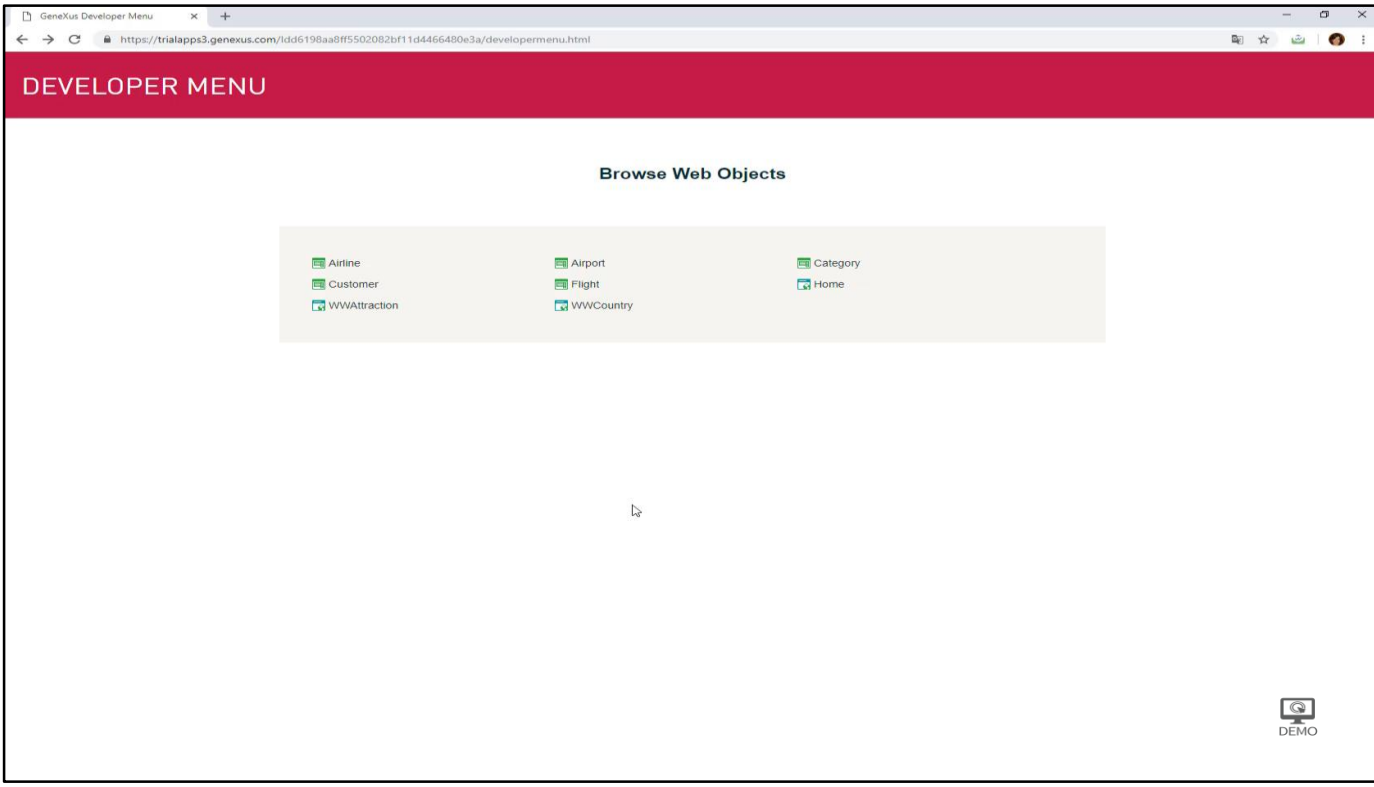
Name	Type	Formula	Nullable
Flight	Flight		
FlightId	Id		No
FlightDepartureAirportId	Id		No
FlightDepartureAirportName	Name		
FlightDepartureCountryId	Id		
FlightDepartureCountryName	Name		
FlightDepartureCityId	Id		
FlightDepartureCityName	Name		
FlightArrivalAirportId	Id		No
FlightArrivalAirportName	Name		
FlightArrivalCountryId	Id		
FlightArrivalCountryName	Name		
FlightArrivalCityId	Id		
FlightArrivalCityName	Name		
FlightPrice	Price		No
FlightDiscountPercentage	Percentage		No
AirlineId	Id		Yes
AirlineName	Name		
AirlineDiscountPercentage	Percentage		
FlightFinalPrice	Price	FlightPrice*(1-AirlineDiscountPercentage/100) IF Airli...	
Seat	Seat		
FlightSeatId	Id		
FlightSeatLocation	Location		

We'll define this domain with enumerated values

Let's try this at runtime by pressing F5...

As we can see, the FLIGHTSEAT physical table will be created, and it will be associated with the 2nd level of the Flight transaction; also, it will contain the attributes and they key that we've defined... note that the structure of the FLIGHT transaction will not be modified because the FlightCapacity attribute will not be physically created, as we expected.

We agree, so we click on reorganize...



We run the Flight transaction... query flight number 1 and enter some seats:

- 1A - window
- 1B – middle
- 1C – aisle
- 1D – window
- 1E – middle
- 1F – aisle

As we enter the seats, note that the total number of seats is updated every time we add a new seat to the flight.

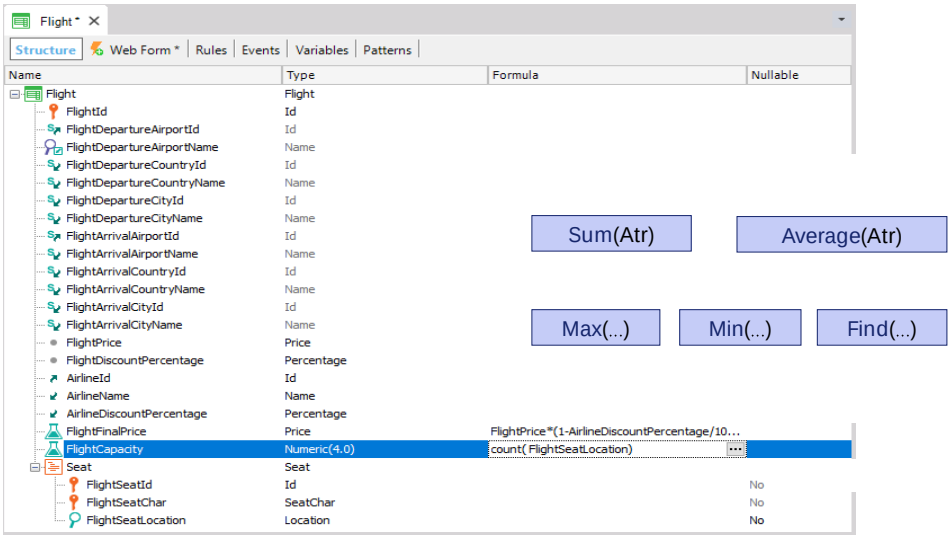
That is, it is triggered interactively in the Browser, as we add lines, and when we leave the attribute mentioned in the Count. Go back a little bit in the video, and look at what happened with the formula attribute.

Lastly, we add:

- 2A – window... and stop here...

Let's return to GeneXus.

Aggregate formulas: Count, Sum, Average, etc.

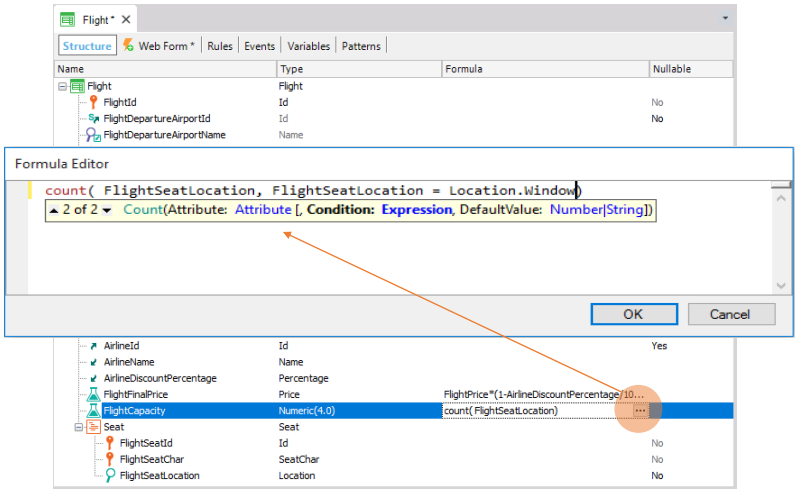


In addition, there are other Aggregate formulas that make operations by taking several records into account.

For example: Sum, to add the values of the specified attribute; Average, to find the average value of the specified attribute, and others such as Max, to find the maximum value of an attribute in the table to be navigated, and return the value of some other attribute for the corresponding record; Min, to do the same but with a minimum value, or Find, to find the value of an attribute according to some condition; for example, to find the Identifier value of an attraction whose name is “Eiffel Tower.” This will be frequently used later.

Aggregate formulas: Count, Sum, Average, etc.

Conditions can be added to count "certain" lines:

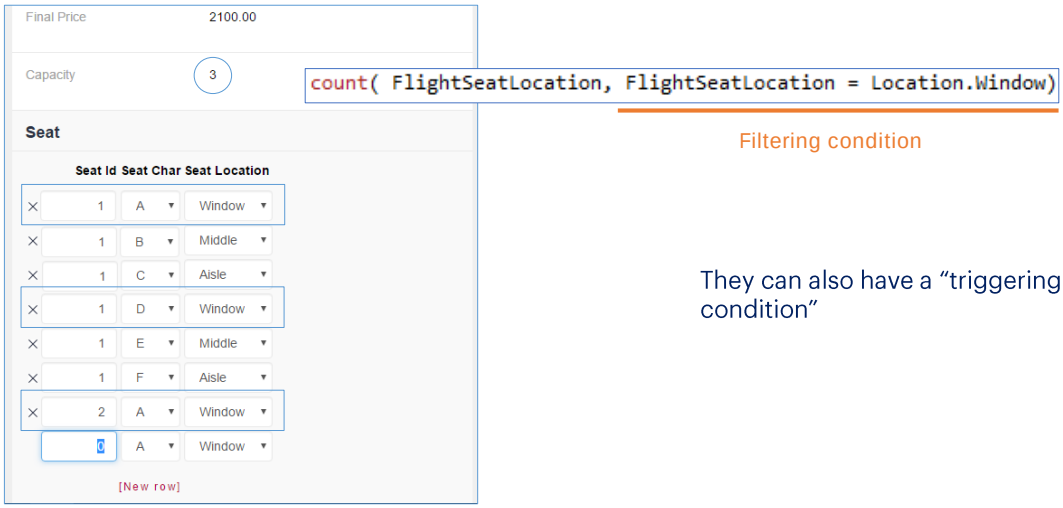


If we want to count not only the total seats in the flight we're positioned in, but also those that meet another condition, such as, for example, the number of seats next to the window, we can add this condition to the formula. In this way... since the FlightSeatLocation attribute belongs to the Location domain, and it has 3 enumerated values defined, the syntax to ask for the value taken by the attribute is as follows:

domain name, period, and the name associated with the value we want to filter by, which in this case is Window.

We click on OK.

Aggregate formulas: Count, Sum, Average, etc.



We press F5.

We run the Flight transaction, record number 1; the flight's capacity is now 3, and it corresponds to the number of seats located next to a window, which matches the seats we entered in the seat grid.

In sum, we have seen that in addition to the implicit condition (when there are related records), it is also possible to count, sum, search for, maximize or average; that is to say, add those records that comply with a certain explicit condition indicated by us. This condition is called “filtering condition” because it allows us to keep only those records that we're interested in.

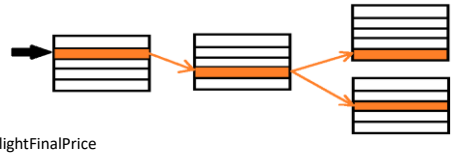
Lastly, we must remember that just like every other global formula, Aggregate formulas can have a “triggering condition.” That is to say, the formula is only calculated when this condition is met.

GeneXus

Summing up

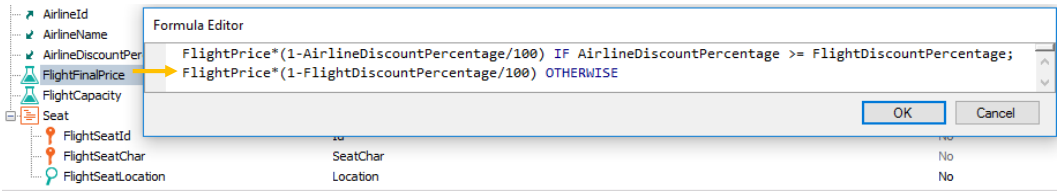
Horizontal:

- To make a calculation, they access a record and, occasionally, those related through an extended table.



FlightFinalPrice

Attribute = *expresión*$_1$ **if condición**$_1$;
expresión$_2$ **if condición**$_2$;
...
expresión$_n$ **if condición**$_n$;
expresión$_o$ **otherwise**;



In sum: we've seen two types of formulas:

Horizontal formulas - they access a record to make a calculation. Also, these records may be related through the extended table.

That was the case of FlightFinalPrice.

These attributes belonged to the FLIGHT table, and the others to the Airline table.

As we saw in the example, we could set a formula attribute to be calculated in different ways depending on the value of a condition.

GeneXus

Summing up

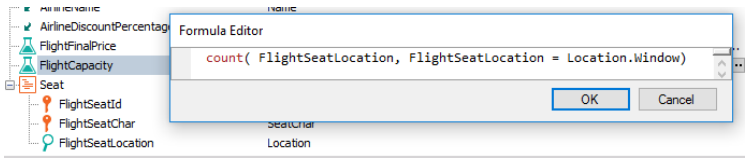
Aggregate:

- To make a calculation, they need to navigate many records of the same table.

FlightId	FlightDepartureAirportId	...
1	1	
2	3	...
3	1	...
...	...	...

FlightId	FlightSeatId		FlightSeatLocation
1	1	A	Window
1	1	B	Aisle
1	2	A	Window
1	2	B	Aisle
1	3	C	Middle
2	1	A	Window
2	1	B	Middle
3	...	...	...
...	...	...	...

- Example: FlightCapacity



Aggregate formulas - to make their calculation, they need to navigate many records in the same table.

That was the case of FlightCapacity.

From the FLIGHT table associated with the formula attribute, it made a calculation over the FLIGHTSEAT table that contains the FlightSeatLocation attribute.

In this case, since the formula attribute is associated with a table, Flight, which has a 1-to-many relationship with the table over which the Count operation will be made, only the related records will be counted. If the relationship didn't exist, they would all be counted. In addition, because we indicate conditions for the records to be counted, the related records will only be counted as long as they meet that condition.

GeneXus

Summing up

Attribute = **Count**(Attribute, condition, DefaultValue) **if** condición;

Sum(Expression, condition, DefaultValue) **if** condición;

Find(Expression, condition, DefaultValue) **if** condición;

…

The filtering condition is the second parameter in the formula, and as a third parameter we can indicate a default value; that is to say, the value that will be returned by the formula if no record is found to count, sum, etc.

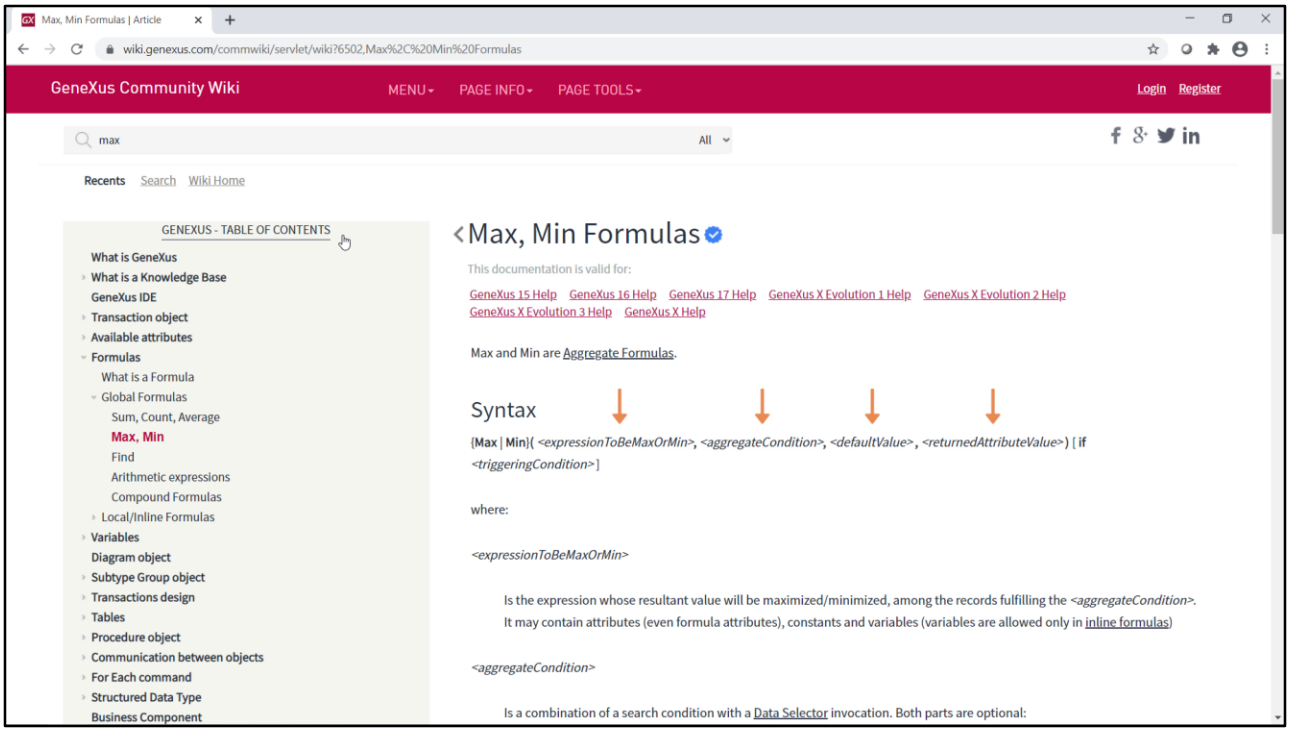
Just like horizontal formulas, aggregate formulas can be stated with conditions.

GeneXus

Summing up

```
Attribute  =  2 + Count(Attribute, condition, DefaultValue) *
              Sum(Expression, condition, DefaultValue) if condición;
              Atr1 + Atr2 * Atr3  otherwise;
              ...
```

Horizontal formulas can be combined with aggregate formulas, providing a high degree of expressiveness in calculations, but we won't talk about it in this course.

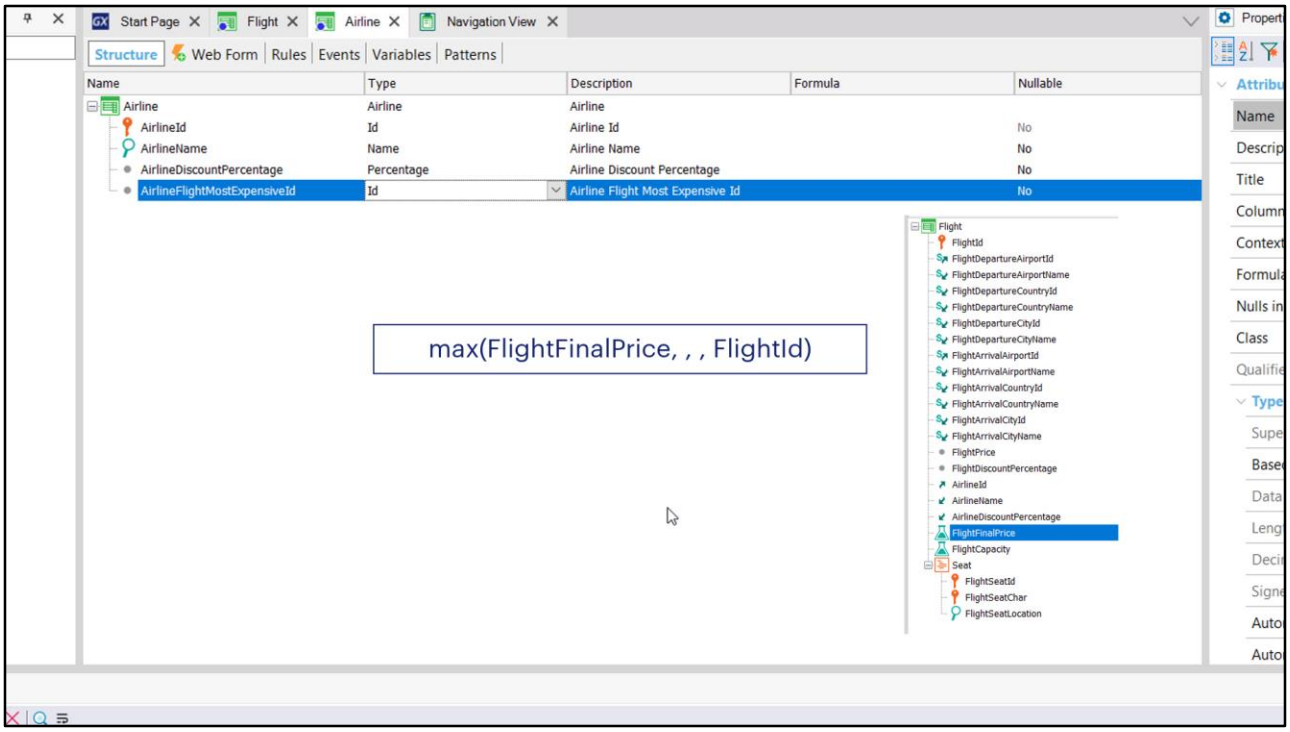


An aggregate formula can not only be specified if the table being navigated corresponds to a level of the same transaction. In the example we saw, the formula attribute, FlightCapacity, is on the first level of the Flight transaction, and the table being navigated will be the one corresponding to the second level.
But let's look at what would happen if, for example, we wanted to define a formula at the airline level, AirlineFlightMostExpensiveId, that would return the flight identifier of the most expensive airline flight.
The formula we need in that case is Max. If we look it up in the Wiki, we see that it has 4 parameters, of which only the first one (that indicates the value that will be maximized) is mandatory. In our case, it would be defined in this way:

max(FlightFinalPrice, , , FlightId)

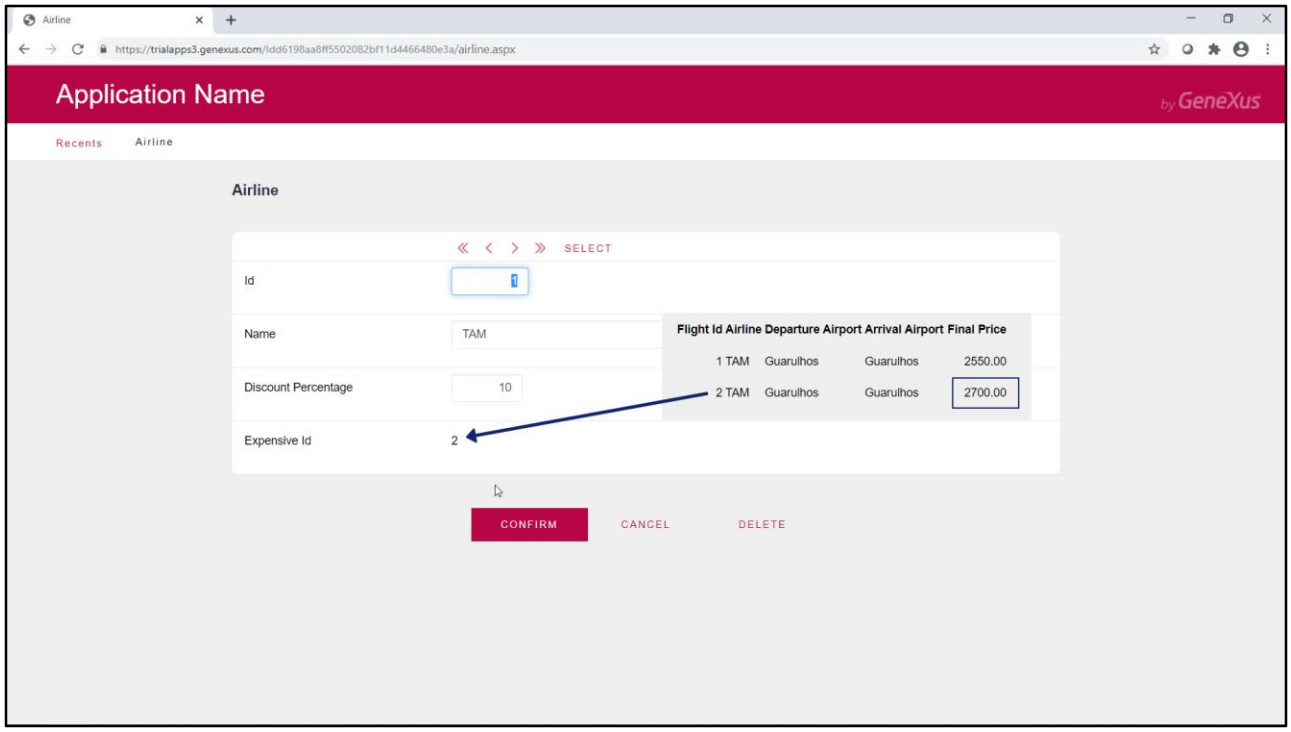
Where we are, of all the Flight records corresponding to the airline, keeping the one or the ones with the highest FlightFinalPrice attribute value (it doesn’t matter that it is also a formula attribute). And for the first of the maximum price records, it returns the value of the FlightId attribute. Since we didn't specify a second parameter, it will not apply any other filter condition on the records to be considered for maximization, and since we didn't include a third parameter, if it doesn't find any associated

record, it will return the empty value. But this will only happen if the airline does not have any associated flights.
If we see it running, when we open the Airline transaction the formula is triggered, which will navigate the flight table, which is not in memory at this time.



Where we are, of all the Flight records corresponding to the airline, keeping the one(s) with the highest FlightFinalPrice attribute value (it doesn’t matter that it is also a formula attribute). And for the first of the maximum price records, it returns the value of the FlightId attribute.

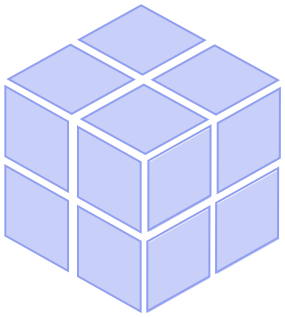
Since we didn't specify a second parameter, it will not apply any other filter condition on the records to be considered for maximization, and since we didn't include a third parameter, if it doesn't find any associated record, it will return the empty value. But this will only happen if the airline does not have any associated flights.



If we see it running, when we open the Airline transaction the formula is triggered; it will navigate the flight table which is not in memory at this time.

GeneXus

Knowledge Base



Global Formulas

Attribute = *fx*

Local (o inline) Formulas

Lastly, let's remove the filtering condition from the FlightCapacity attribute...

And send the new definitions to GeneXus Server.

The formulas discussed in this video are those specified at the attribute level; they are called “formula attributes” and are known throughout the knowledge base.

Later on we will study the others, which are equal but not assigned to an attribute: they are the so-called “local” or “inline” formulas.

Remember that formula attributes are virtual, i.e. they are not physically created as fields in the associated table. However, it is possible to modify the default behavior of these attributes and store their values, defining them as “redundant attributes” so that they will no longer be virtual. We invite you to read about how to do it in the GeneXus Wiki.

Remember that aggregation formulas don't always need to search for information in a table, but they also (as in the case of Flight that we studied) operate in memory.

Watch the next videos to learn how to use local or “inline” formulas.

GeneXus™

training.genexus.com
wiki.genexus.com