Compound Formulas

GeneXus

Compound Formulas

Attribute = expression$_1$ **if condition$_1$**;
expression$_2$ **if condition$_2$**;
...
expression$_n$ **if condition$_n$**;
expression$_o$ **otherwise**;

```
Occupancy.Low IF count(FlightSeatLocation) < 5;
Occupancy.Medium IF count(FlightSeatLocation) >5 and count(FlightSeatLocation) < 8;
Occupancy.High OTHERWISE
```

conditions: valid logic expressions, including
- attributes from extended table
- constants, functions, logic operators (AND, OR, NOT) and relational operators (>, >=, <, <= , =, <>, like)

Attribute = ***Count***(*Attribute, condition, default* **Value**) **if** condition;

Sum(*Expression, condition, default* **Value**) **if** condition;

Find(*Expression, condition, default* **Value**) **if** condition;

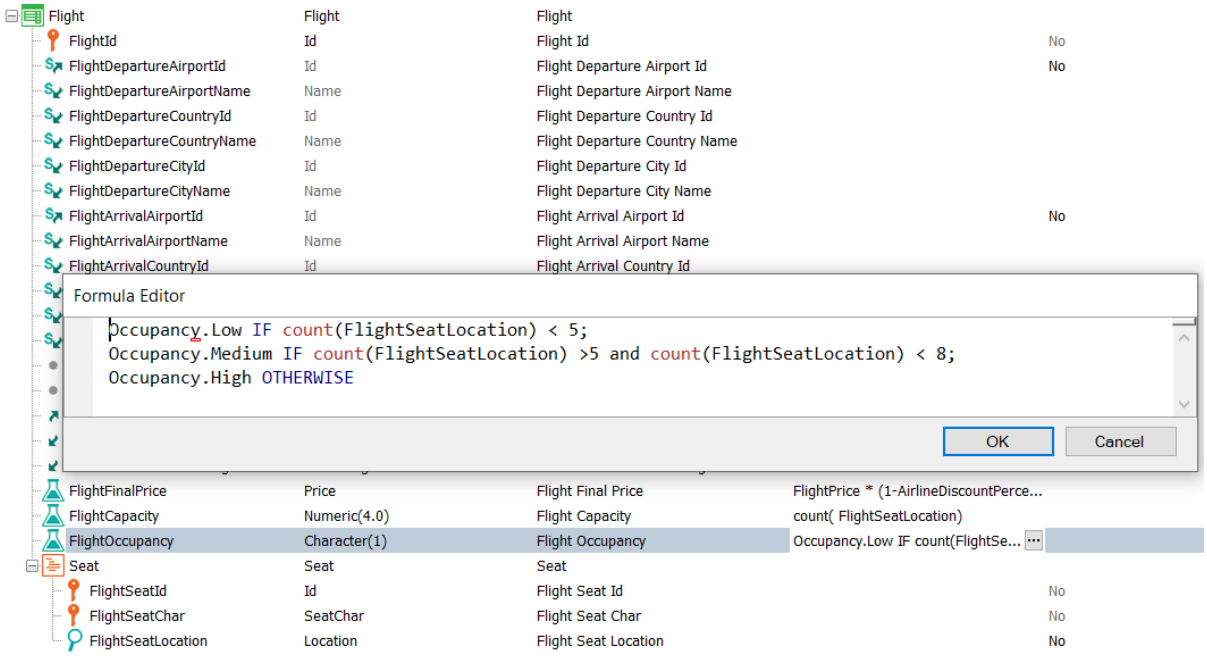
Compound formulas include several conditional aggregate formulas and may also contain horizontal expressions.

In this case, each expression can be an aggregate formula or a horizontal formula. If all the expressions included are horizontal formulas, then the defined formula is not compound but horizontal.

Conditions are any valid logical expression, and may contain attributes belonging to the extended table of the table associated with the attribute being defined as a formula, constants, functions, logical operators (and, or, not) and relational operators (>, >=, <, <=, =, <> and like). The first condition that evaluates to True will cause the result of the formula to be that of the expression to the left of that condition (the others will not continue to be evaluated).

When none of the conditions evaluated are True, if there is an *expression* with an otherwise clause, the result of the formula will be that of the expression preceding this clause.

Example



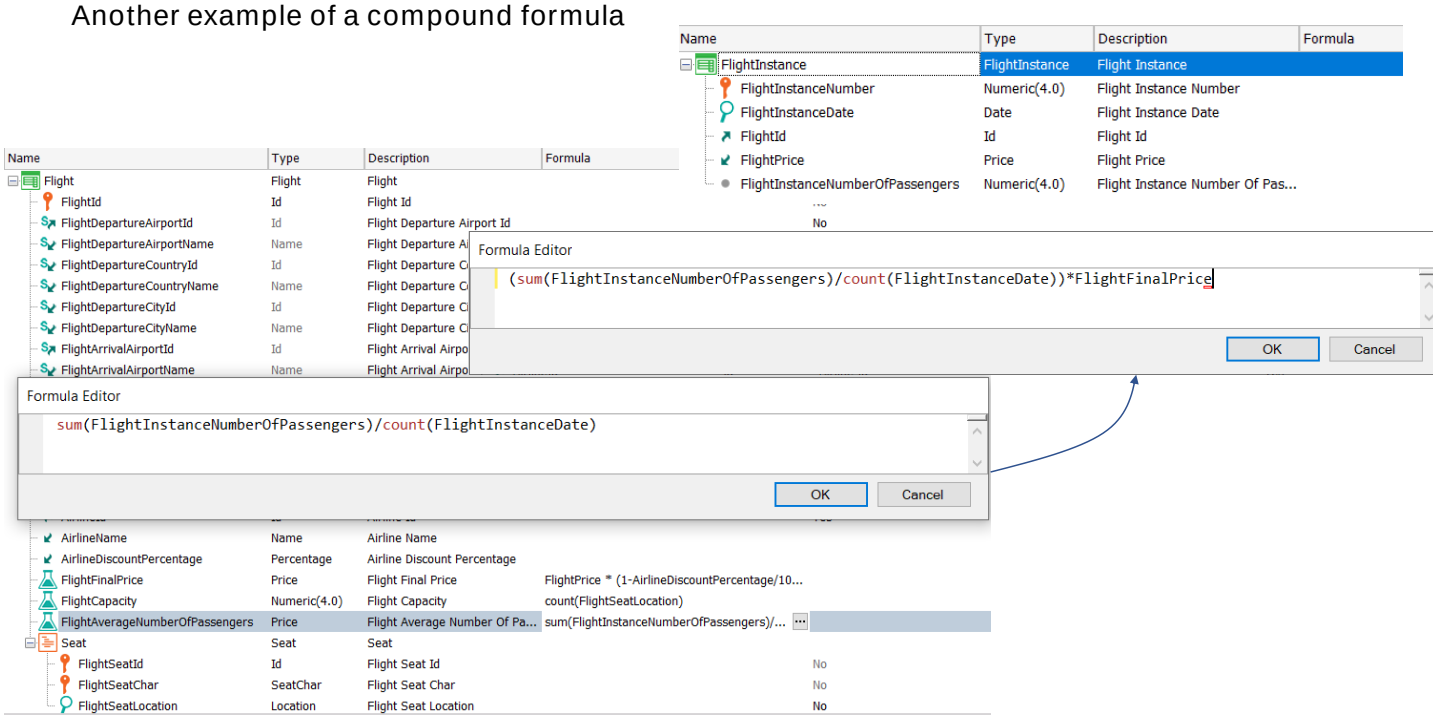
Let's see an example of this type of compound formulas in our travel agency reality.

Here we see that the FlightOccupancy attribute was defined based on horizontal expressions that assign the corresponding value of the Occupancy domain (Low, Medium or High), depending on the number of seats on the flight, which are calculated with aggregate count formulas.

In particular, in our case, we could have replaced the aggregate formulas with the FlightCapacity attribute, but it is perfectly valid to leave it as it is defined.

In this implementation, the structure is that of a horizontal formula and the aggregate ones were included in the triggering conditions.

Another example of a compound formula



In this example, we want to calculate the average number of passengers that traveled on a given flight.

Remember that unlike the Flight transaction where we defined flights in a generic way, in the FlightInstance transaction we model the actual instances of a particular flight, with a date, flight number, number of passengers, etc.

To calculate the average number of passengers that took each flight, we must add the total number of passengers for all instances of that flight and divide it by the number of flight instances.

We define the FlightAverageNumberOfPassengers attribute in the Flight transaction as a global formula, calculated as the quotient of an aggregate sum formula, which adds the FlightInstanceNumberOfPassengers attribute and divides the result by the number of instances of the flight, calculated as an aggregate Count formula that uses the FlightInstanceDate attribute to count the instances.

Note that since the formula has been defined as global in the Flight transaction, its context is the table associated with the formula attribute, i.e. the FLIGHT table. Therefore, the result will be the average number of passengers of the instances of the particular flight in which you are positioned.

Also, remember that we have the aggregate Average formula with which we could have done this calculation, but we did it this way to prove that it is possible to compose formulas to create compound formulas.

And we could continue to compose calculations, for example, if we were interested in the average revenue per flight, we could have multiplied the average number of passengers by the final price of the flight, the FlightFinalPrice attribute.

Note that this attribute is in turn a horizontal formula, so GeneXus can easily perform complex calculations such as the ones described.

Examples of other compound formulas

Attribute = **Max**(...) if condition1;
(2 * $attr_x$) + 100 if condition2;
Sum($attr_y$) otherwise

Attribute = **Find**(...) if condition1;
1 otherwise

Attribute = procedure(...) if condition1;
Min(...) if condition2;
10 if condition3

Attribute = 2 + **Count**(Attribute, condition, default **Value**) *

Sum(Expression, condition, default **Value**) **if** condition;

$Attr_1$ + **$Attr_2$** * **$Attr_3$** otherwise;

Compound formulas provide great flexibility for defining calculations, and make it possible to model a large number of situations.

In this video, we saw how convenient it is to use formulas to save code, taking advantage of the simplicity of declarative programming.

GeneXus™

training.genexus.com
wiki.genexus.com