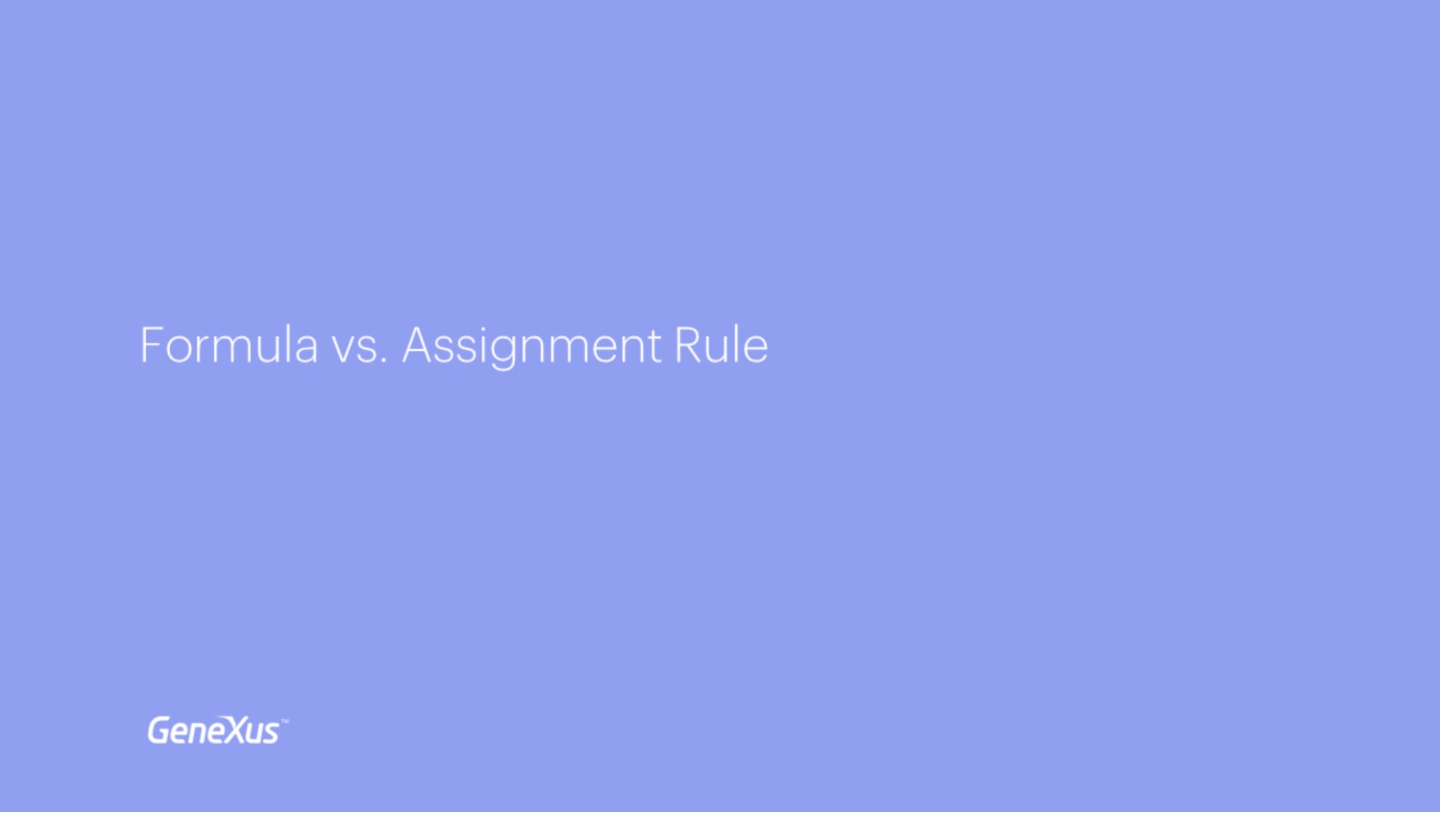
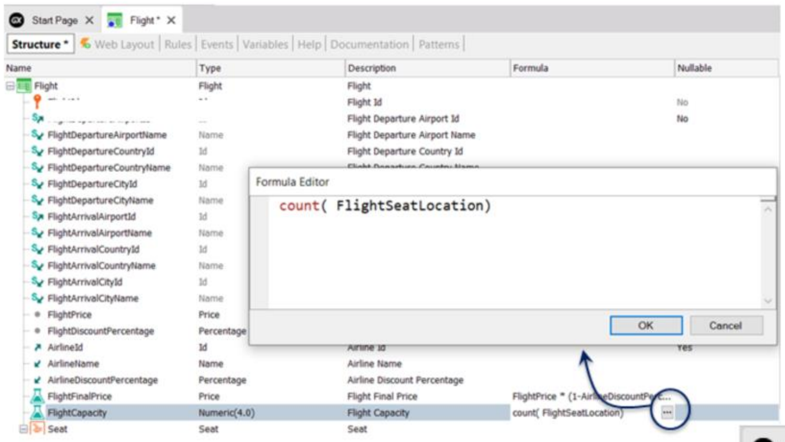


Formula vs. Assignment Rule

Next, we will take a closer look at some of the concepts of the formulas. Let's start by looking at the difference between defining an attribute through a calculation in a formula or in an assignment rule.

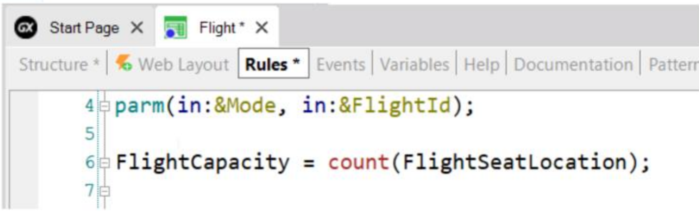
Considerations when defining an attribute based on a calculation: rule or formula?



Name	Type	Description	Formula	Nullable
Flight	Flight	Flight		
[illegible]	[illegible]	Flight Id		No
[illegible]	[illegible]	Flight Departure Airport Id		No
FlightDepartureAirportName	Name	Flight Departure Airport Name		
FlightDepartureCountryId	Id	Flight Departure Country Id		
FlightDepartureCountryName	Name	Flight Departure Country Name		
FlightDepartureCityId	Id			
FlightDepartureCityName	Name			
FlightArrivalAirportId	Id			
FlightArrivalAirportName	Name			
FlightArrivalCountryId	Id			
FlightArrivalCountryName	Name			
FlightArrivalCityId	Id			
FlightArrivalCityName	Name			
FlightPrice	Price			
FlightDiscountPercentage	Percentage			
AirlineId	Id	Airline Id		Yes
AirlineName	Name	Airline Name		
AirlineDiscountPercentage	Percentage	Airline Discount Percentage		
FlightFinalPrice	Price	Flight Final Price	FlightPrice * (1-AirlineDiscountPerc...	
FlightCapacity	Numeric(4.0)	Flight Capacity	count(FlightSeatLocation)	
Seat	Seat	Seat		

```
count( FlightSeatLocation)
```

Vs.

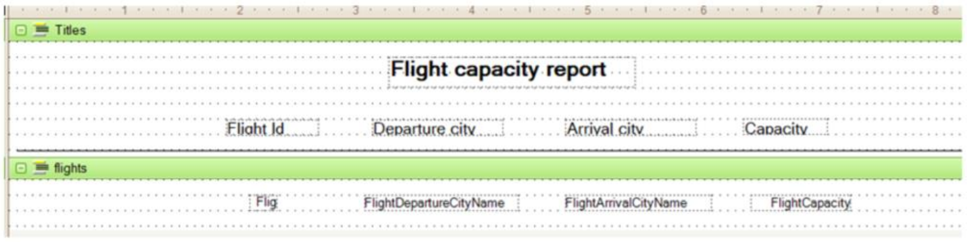


```
parm(in:&Mode, in:&FlightId);

FlightCapacity = count(FlightSeatLocation);

```

When the value of an attribute can be obtained by means of a calculation, we usually define it as a formula in the transaction structure.
Note, however, that we could also assign the calculation to the attribute by means of a rule.

What considerations should we take into account when deciding whether to assign the calculation by means of a rule or by defining the attribute as a formula? We have already seen that if the attribute is a formula, GeneXus does not create in its associated table (i.e. the table to which the formula attribute would belong if it were stored), a field to store the value, since it understands that its value can be obtained from the calculation. For this reason, we say that we consider the attribute as "virtual" since it is still present in the transaction structure, but not in the associated table.
We can see that in the definition of the table it appears as a "logical" attribute.

On the other hand, if the attribute value is assigned by a rule, it will still be present in the table simply by assigning a value to it, so it does not become a virtual attribute, but remains a stored attribute.



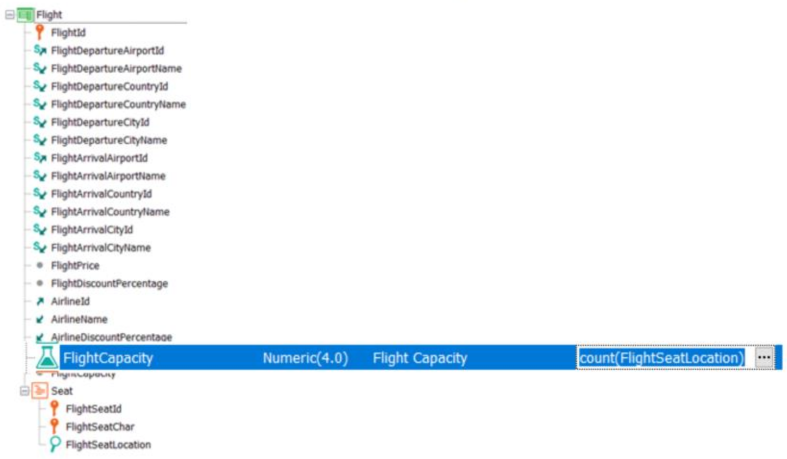
Titles

Flight capacity report

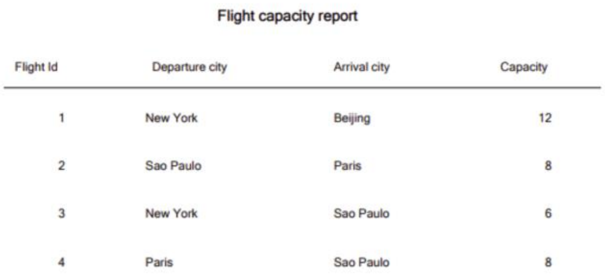
Flight Id | Departure city | Arrival city | Capacity

flights

Flig | FlightDepartureCityName | FlightArrivalCityName | FlightCapacity



```
Flight
  FlightId
  FlightDepartureAirportId
  FlightDepartureAirportName
  FlightDepartureCountryId
  FlightDepartureCountryName
  FlightDepartureCityId
  FlightDepartureCityName
  FlightArrivalAirportId
  FlightArrivalAirportName
  FlightArrivalCountryId
  FlightArrivalCountryName
  FlightArrivalCityId
  FlightArrivalCityName
  FlightPrice
  FlightDiscountPercentage
  AirlineId
  AirlineName
  AirlineDiscountPercentage
  FlightCapacity      Numeric(4.0)    Flight Capacity    count(FlightSeatLocation)
  Seat
    FlightSeatId
    FlightSeatChar
    FlightSeatLocation
```

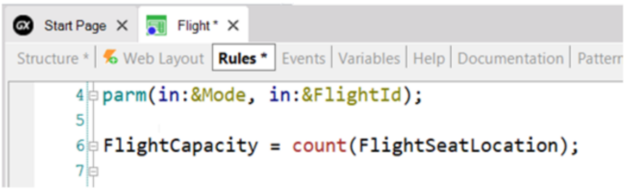


Flight capacity report

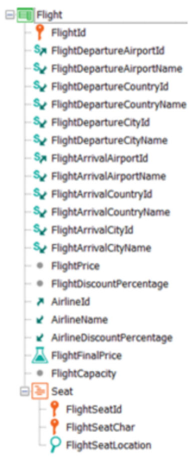
Flight Id	Departure city	Arrival city	Capacity
1	New York	Beijing	12
2	Sao Paulo	Paris	8
3	New York	Sao Paulo	6
4	Paris	Sao Paulo	8

So is it better to use an assignment rule than a formula? No, not necessarily because it depends on how the attribute will be used.

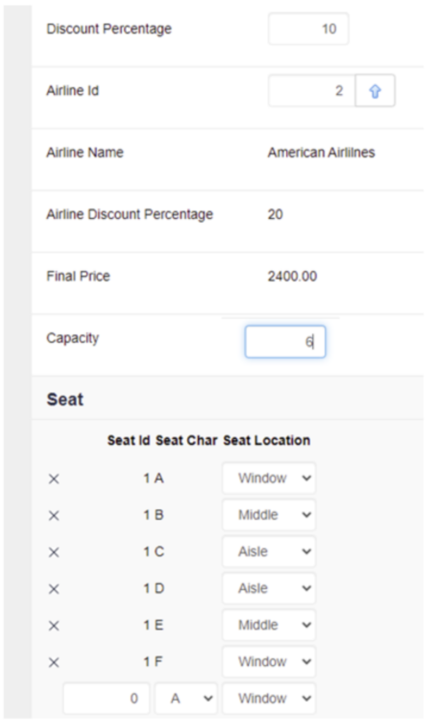
If the attribute is to be used in other objects and we need to make sure that its value is the actual result of the calculation, then we define it as a formula. As this definition is global to the knowledge base, when any object queries the attribute value, it accesses its calculation and it is triggered, updating the value on the fly.



```
parm(in:&Mode, in:&FlightId);

FlightCapacity = count(FlightSeatLocation);

```



```
Flight
  FlightId
  FlightDepartureAirportId
  FlightDepartureAirportName
  FlightDepartureCountryId
  FlightDepartureCountryName
  FlightDepartureCityId
  FlightDepartureCityName
  FlightArrivalAirportId
  FlightArrivalAirportName
  FlightArrivalCountryId
  FlightArrivalCountryName
  FlightArrivalCityId
  FlightArrivalCityName
  FlightPrice
  FlightDiscountPercentage
  AirlineId
  AirlineName
  AirlineDiscountPercentage
  FlightFinalPrice
  FlightCapacity
  Seat
    FlightSeatId
    FlightSeatChar
    FlightSeatLocation
```



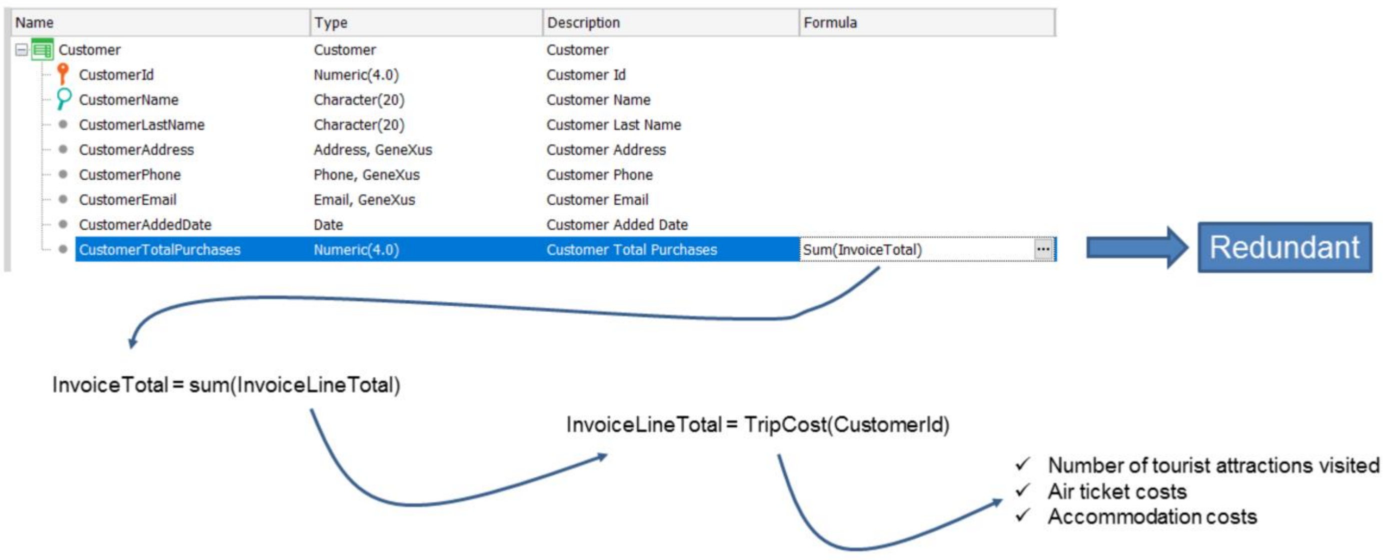
Discount Percentage	10
Airline Id	2
Airline Name	American Airlines
Airline Discount Percentage	20
Final Price	2400.00
Capacity	6

Seat

	Seat Id	Seat Char	Seat Location
×	1	A	Window
×	1	B	Middle
×	1	C	Aisle
×	1	D	Aisle
×	1	E	Middle
×	1	F	Window
	0	A	Window

If the attribute value is updated only by the transaction, then its value can be assigned locally, with a rule. The attribute is still stored and its value can also be changed through the form.

Considerations when defining an attribute based on a calculation



Name	Type	Description	Formula
Customer	Customer	Customer	
CustomerId	Numeric(4.0)	Customer Id	
CustomerName	Character(20)	Customer Name	
CustomerLastName	Character(20)	Customer Last Name	
CustomerAddress	Address, GeneXus	Customer Address	
CustomerPhone	Phone, GeneXus	Customer Phone	
CustomerEmail	Email, GeneXus	Customer Email	
CustomerAddedDate	Date	Customer Added Date	
CustomerTotalPurchases	Numeric(4.0)	Customer Total Purchases	Sum(InvoiceTotal)

Redundant
InvoiceTotal = sum(InvoiceLineTotal)
InvoiceLineTotal = TripCost(CustomerId)
✓ Number of tourist attractions visited
✓ Air ticket costs
✓ Accommodation costs

However, the fact that the attribute must be calculated every time is not always an advantage.

If the calculation has to be performed on many records each time and has to be done frequently, it may affect the application performance, in contexts where real-time feedback is required.

It can also be the case when the formula attribute is calculated from other formula attributes, so that a large number of calculations must be performed to obtain the value.

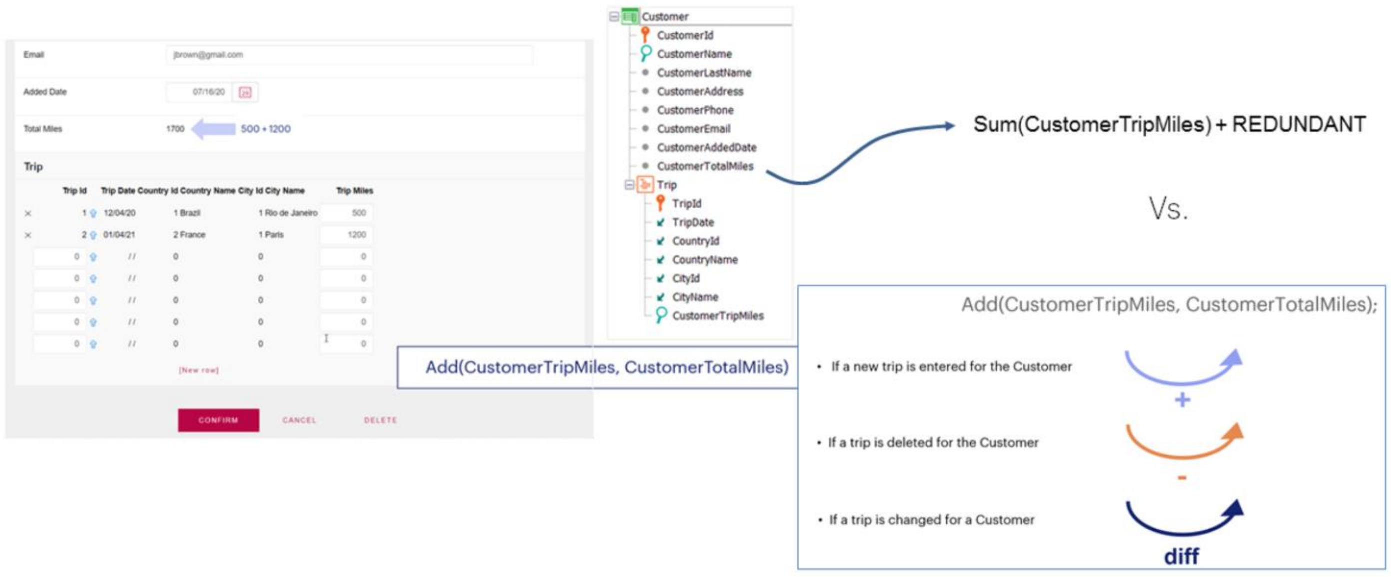
Suppose, for example, that we are calculating the total purchases of a customer of the travel agency, as the sum of the invoices made to the customer; the total of each invoice is calculated as the sum of his/her trips and in turn the cost of each trip is calculated by a procedure that takes into account the number of attractions visited, the cost of air tickets, the cost of accommodation, etc.
To list the total purchases of all the agency's clients in the last 5 years, it will probably be necessary to run through many records and perform many calculations, which may affect the response time of the system to obtain the result.

In this situation, it would be good to store the result of the customer's total purchases in a table, so that if something changes that affects the result, it is recalculated again and the new result is stored, but if nothing changes, the stored value can be used instead of always performing the calculation.
We do this by defining the formula attribute as redundant, i.e., although its value can be obtained by a calculation, the result of the calculation will be stored in the database and the value will be retrieved from there in the future.

Also for similar reasons, we can define an inferred attribute as redundant.

The definition of redundant attributes will be discussed later in another video.

Updating an attribute by means of a formula or a rule



Email	jbrown@gmail.com
Added Date	07/16/20
Total Miles	1700

Trip

	Trip Id	Trip Date	Country Id	Country Name	City Id	City Name	Trip Miles
×	1	12/04/20	1	Brazil	1	Rio de Janeiro	500
×	2	01/04/21	2	France	1	Paris	1200
	0	/ /	0		0		0
	0	/ /	0		0		0
	0	/ /	0		0		0
	0	/ /	0		0		0
	0	/ /	0		0		0

We have seen that we can define an attribute as a formula or assign its value with a rule, but we must also consider how its update mechanism will be in each case.

Remember the use of the Add (or Subtract) rule, which allowed us to keep the value of an attribute of the extended table up to date, performing the appropriate operation depending on whether we were inserting, deleting or modifying a record.

The attribute of the CustomerTotalMiles example, updated by the Add rule, is a stored attribute and therefore the value is immediately retrieved. However, because the Add rule is local to the Customer transaction, as we saw before, only the CustomerTotalMiles attribute will be updated if the Customer transaction screen or a business component of the transaction are executed.

If we want the value of the customer's total miles attribute to always be kept up to date, we should define it as a formula, in this case a Sum formula that adds the miles of each trip to the customer's total miles.

Although defining the attribute as a formula ensures that it is continuously updated, we lose the possibility of it being stored, and in order to obtain its value we may cause the performance problems mentioned above. To solve this, we could define the CustomerTotalMiles formula attribute as redundant and it will become a stored attribute. But when is the attribute value updated in the table?

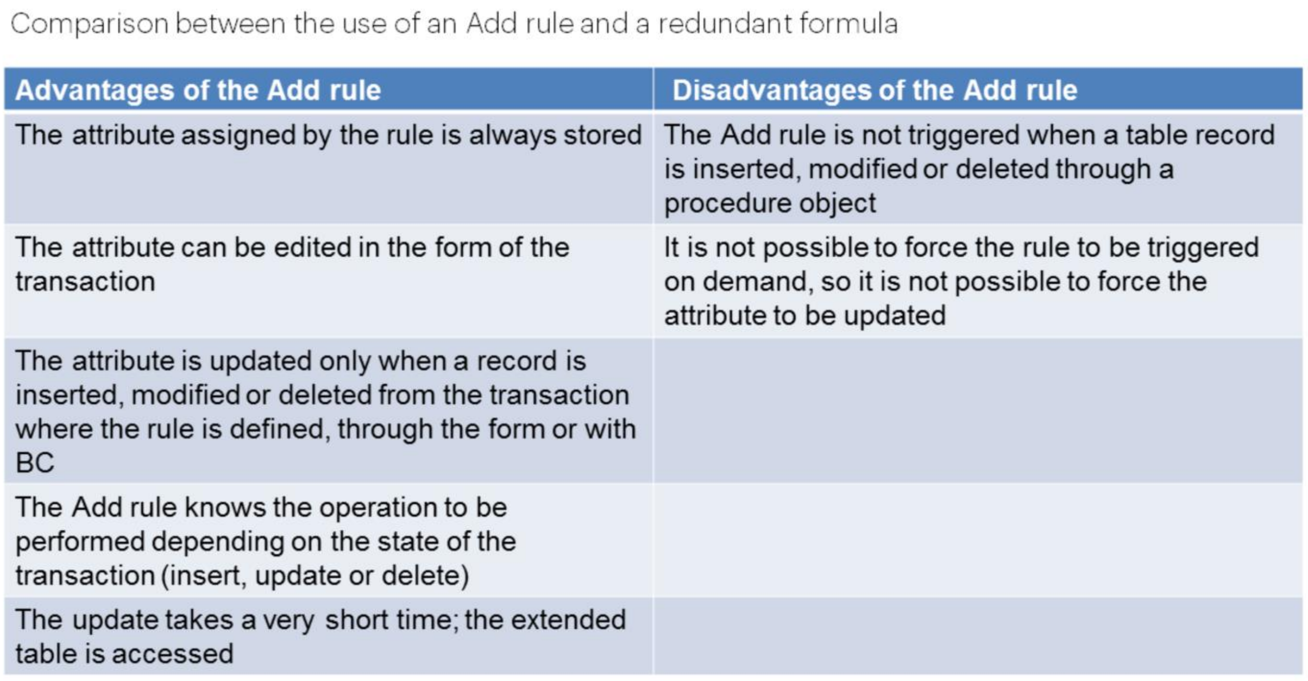
Updating an attribute by means of a formula or a rule (continued)

When we define a formula attribute as redundant, GeneXus automatically creates procedures in charge of updating its value and storing it in the database.

In the example we saw, when through the Customer transaction (or its Business Component) a customer trip is inserted or deleted or the value of the CustomerTripMiles is affected, the Sum formula is recalculated and the new value is stored in the Customer table.

When the value of some of the attributes that are part of the calculation of a redundant attribute is modified from the form of a transaction or from a Business Component, GeneXus triggers the procedure that updates its value.

Therefore, from the point of view of the update and that the attribute is stored, updating the attribute by means of an ADD rule and defining it as a redundant formula are **equivalent solutions**.



Comparison between the use of an Add rule and a redundant formula

Advantages of the Add rule	Disadvantages of the Add rule
The attribute assigned by the rule is always stored	The Add rule is not triggered when a table record is inserted, modified or deleted through a procedure object
The attribute can be edited in the form of the transaction	It is not possible to force the rule to be triggered on demand, so it is not possible to force the attribute to be updated
The attribute is updated only when a record is inserted, modified or deleted from the transaction where the rule is defined, through the form or with BC	
The Add rule knows the operation to be performed depending on the state of the transaction (insert, update or delete)	
The update takes a very short time; the extended table is accessed	

Even through from the point of view of updating and being stored, using an add rule or defining a formula attribute as redundant are equivalent solutions, there are several differences to consider between using one way or the other.

Now let’s look at some comparative tables to help us consider the pros and cons in each case.

First, let’s analyze the advantages and disadvantages of using an Add rule.

Advantages:

- The attribute assigned by the Add rule is always stored, so its value is immediately retrieved.
- The attribute can still be edited in the transaction's form
- The attribute is updated each time the rule is triggered; i.e. when a record is inserted, modified or deleted from the transaction where the rule was defined, either through its form or through Business Components
- The Add rule knows the operation to be performed depending on the state of the transaction; i.e. it knows if it has to add when a record is inserted, subtract if it is deleted or modify when an update is performed
- The update takes a very short time, since only tables belonging to the extended table are accessed.

Disadvantages:

- The Add rule is not triggered when a table record is inserted, modified or deleted through a procedure object, so in this case the attribute value is not updated
- It is not possible to force the rule to be triggered on demand, so it is not possible to force the attribute to be updated.

Comparison between the use of an Add rule and a redundant formula

Advantages of the redundant formula	Disadvantages of the redundant formula
The redundant global formula attribute is always stored	The attribute is read-only in the form
The attribute is updated when an attribute of the calculation is modified, or when a record is inserted, modified or deleted from the transaction where it was defined, through the form or with BC	The update takes a long time, since it involves executing several procedures and accessing several tables
Redundancy procedures know the operation to be performed to keep the attribute up to date	The KB size is increased due to the addition of the procedure objects created to maintain redundancy
It is possible to force the triggering of the redundancy update on demand by means of a special procedure	If anything changes in the formula definition, GeneXus must update the redundancy procedures

Now let's look at the advantages and disadvantages of defining a formula attribute as redundant.

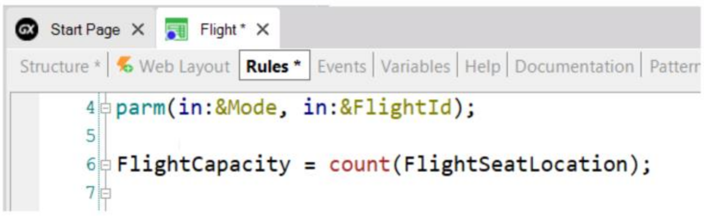
Advantages:

- The attribute defined as a redundant global formula is always stored
- It is updated when any attribute that is part of the calculation is modified, or when a record of the transaction where the formula attribute was defined is inserted, modified or deleted, either through the form or with Business Components
- Redundancy procedures know the operation to be performed to keep the attribute up to date; i.e. they know whether they have to add, subtract or how to modify the value
- It is possible to force the triggering of the redundancy update on demand by means of special procedures that GeneXus creates when we define an attribute as redundant.

Disadvantages:

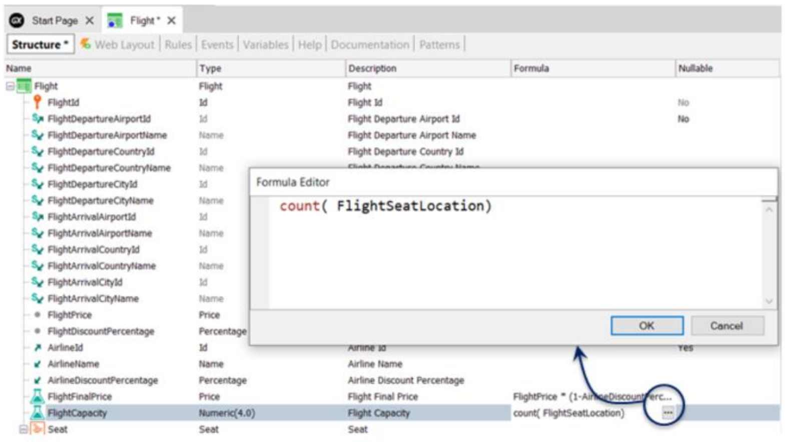
- The attribute is read-only in the form of the transaction where it was defined. Even though with redundancy it is stored, we cannot edit it.
- The redundancy update takes a long time, as it involves running several procedures and usually accessing several tables
- When a formula attribute is defined as redundant, the size of the KB is increased due to the addition of the procedures created to maintain redundancy
- If anything in the formula definition changes, GeneXus must keep the

redundancy procedures up to date.



```
parm(in:&Mode, in:&FlightId);

FlightCapacity = count(FlightSeatLocation);

```

Vs.



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			
FlightDepartureCityId	Id			
FlightDepartureCityName	Name			
FlightArrivalAirportId	Id			
FlightArrivalAirportName	Name			
FlightArrivalCountryId	Id			
FlightArrivalCountryName	Name			
FlightArrivalCityId	Id			
FlightArrivalCityName	Name			
FlightPrice	Price			
FlightDiscountPercentage	Percentage			
AirlineId	Id	Airline Id		Yes
AirlineName	Name	Airline Name		
AirlineDiscountPercentage	Percentage	Airline Discount Percentage		
FlightFinalPrice	Price	Flight Final Price	FlightPrice * (1-AirlineDiscountPerc...	
FlightCapacity	Numeric(4.0)	Flight Capacity	count(FlightSeatLocation)	
Seat	Seat	Seat		

```
count( FlightSeatLocation)
```

+ REDUNDANT

Therefore, as we said before, we must consider what is better, whether to update the value of the attribute by means of a rule or by defining it as a redundant formula, depending on the considerations we have just seen and how we will use the attribute.

In the following videos we will study the different types of formulas that we can define.

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