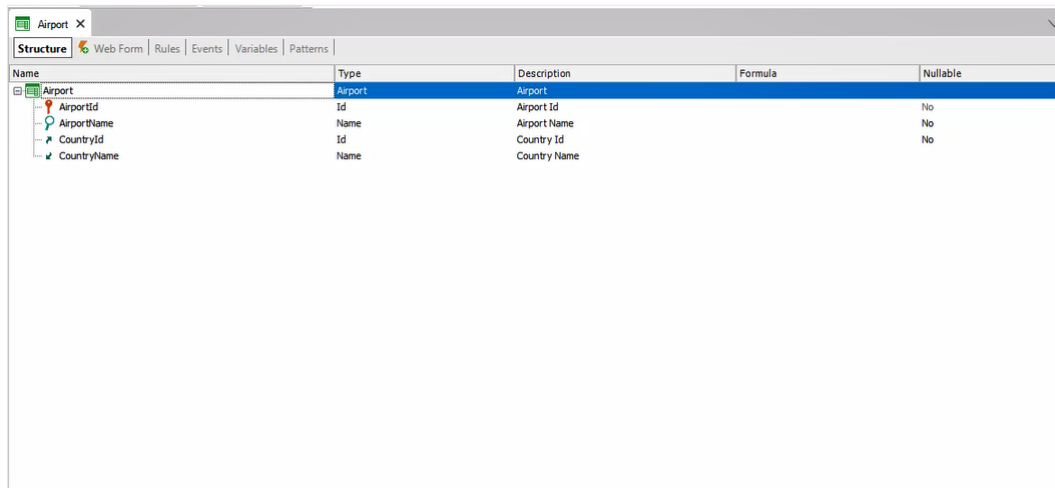


Effects of Adding an Attribute to a Table that Already Contains Data

GeneXus™

Trn Airport



Name	Type	Description	Formula	Nullable
Airport	Airport	Airport		
AirportId	Id	Airport Id		No
AirportName	Name	Airport Name		No
CountryId	Id	Country Id		No
CountryName	Name	Country Name		

Let's suppose that in our KB for the travel agency we have the Airport transaction in the following state and with these attributes. Also, in this transaction we want to add the CountryId and CountryName attributes of the Country transaction, so that each airport has an associated country.

Considering that the Airport table already has records before the reorganization, what value will the CountryId attribute have?

Nullable YES

Name	Type	Description	Formula	Nullable
Airport	Airport	Airport		
AirportId	Id	Airport Id		No
AirportName	Name	Airport Name		No
CountryId	Id	Country Id		Yes
CountryName	Name	Country Name		

AirportId	AirportName	CountryId
1	The Airport of Brazil	Null
2	The Airport of France	Null
3	The Airport of China	Null
...

Nullable NO

Name	Type	Description	Formula	Nullable
Airport	Airport	Airport		
AirportId	Id	Airport Id		No
AirportName	Name	Airport Name		No
CountryId	Id	Country Id		No
CountryName	Name	Country Name		

AirportId	AirportName	CountryId
1	The Airport of Brazil	0
2	The Airport of France	0
3	The Airport of China	0
...

Here we have two options, and it will depend on how we configure these CountryId attributes.

If we set CountryId to accept null values, when we reorganize and GeneXus adds this attribute in the table, the records that were already there will be left with this attribute as null.

Now, if we indicate in the transaction that that attribute does not accept nulls, what will happen? We see that when we reorganize it launches a warning, indicating that this attribute does not accept null values and does not have an initial value, so it will assign a default value to the CountryId attribute.

A record will be created in the table where this attribute is the primary key and in the table where we add it as a foreign key; it will be assigned a value 0 if the attribute is of numeric type, empty if it is of character type and False if it is of Boolean type.

In this case, a new record will be created in the Country table with CountryId = 0 and empty CountryName. And for all the records previously entered in the Airport table, that value will be assigned to the new CountryId attribute.

If a country with ID=0 is registered, it will be referenced.

We will always have the option of going record by record and changing this value to the one we want. Or create a procedure that runs through all the airports with a For Each, and modifies CountryId according to the conditions that we specify.

We could also assign to the new CountryId attribute, an initial value that we know already exists in our database, as we will see in a moment.

Name	Type	Description	Formula	Nullable
Airport	Airport	Airport		No
AirportId	Id	Airport Id		No
AirportName	Name	Airport Name		No
CountryId	Id	Country Id		No
CountryName	Name	Country Name		No
AirportInternational	Boolean	Airport International		No

id	1
Name	Paris Charles de Gaulle Airport
Country Id	2
Country Name	France
International	<input type="checkbox"/>

id	3
Name	Brazil
Country Id	1
Country Name	Brazil
International	<input type="checkbox"/>

id	4
Name	
Country Id	0
Country Name	
International	<input checked="" type="checkbox"/>

Let's see the following case.

Within the Airport transaction, we want to add an attribute that records whether the airport entered is international or not.

For this we add the attribute AirportInternational of Boolean type.

Suppose we want the value to be true by default. To this end, we create a Default rule.

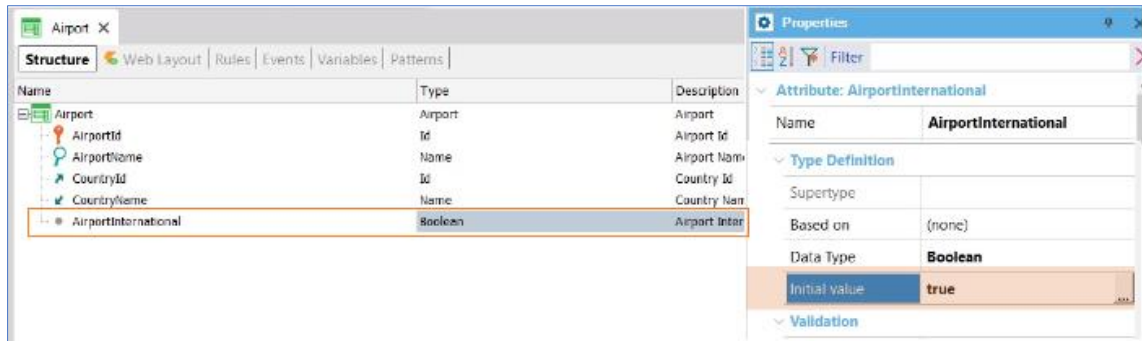
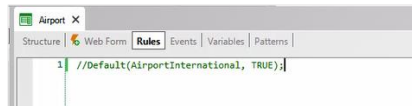
When we reorganize, we see again a Warning, which tells us that the attribute we have just created does not accept null values and has no initial value, so the empty value will be used as default.

If we look at the records previously entered in the Airport table, we see that in all of them it assigned a false value to this new attribute, since it is the default value assigned to this type of data.

If we want to enter a new record from the transaction, we see that by default it is set to true, as we assigned it in the rule, but it does not apply to the records we already had.

When this attribute is created, how can we have all previous records set to true? Through the Initial Value property.

Initial Value Property



If instead of using the Default rule, we set the Initial Value property of the attribute to true, it will not only apply to new records, but all existing ones will take that value.

This is one of the differences between using the Default rule and using the Initial Value property.

Regla Default vs Initial Value

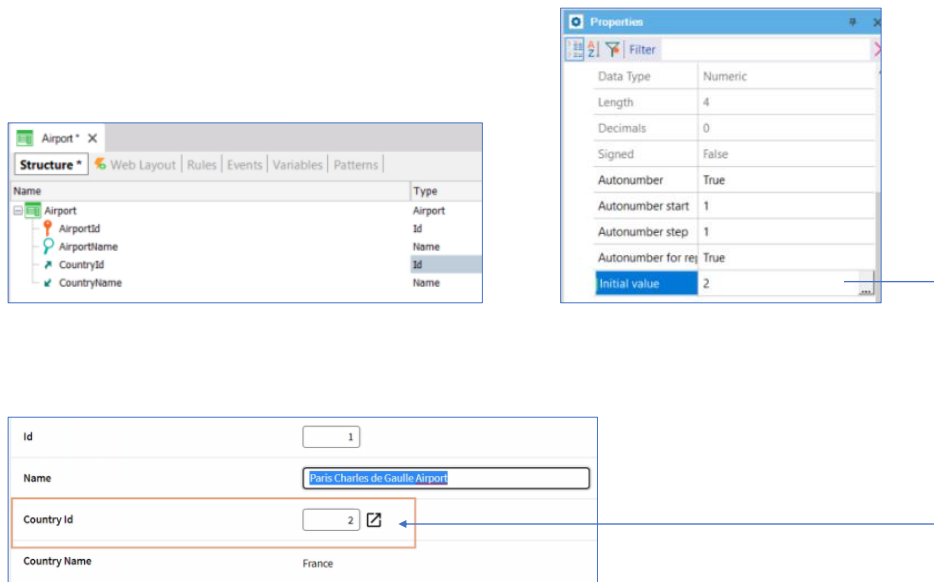
Default Rule:

For records already entered in the database, the new attribute takes a default value.
It belongs to the transaction.

Initial Value:

For records already entered in the database, the new attribute takes the value of the Initial Value property.
It belongs to the attribute.

Another difference is that the default rule is specific to the transaction, not the attribute. As for the Initial value property, it belongs to the attribute. And it will apply anywhere it is used, including transactions, procedures, BC, etc.



Let's go back for a moment to the case where we added CountryId to the Airport table; let's see what would happen if instead of doing it as we did, we had assigned an initial value. For example, suppose we know that all the airports already entered are from the country France. Since France has ID 2, we assign this value in the Initial Value of the CountryId attribute.

When running it we see that the Airport table is reorganized and added to the CountryId attribute, and when viewing the data we see that, in all the records previously entered, CountryId took the value 2, corresponding to France.

Even when we want to add a new record in Airport, CountryId will be initialized by default with the value 2, and we can modify it if we wish.

And what would happen if we had put in the Initial Value of CountryId a value that does not exist in the country table? For example, 20.

The screenshot illustrates the GeneXus development environment. On the left, the 'Structure' pane shows the 'Airport' table with attributes: AirportId (Id), AirportName (Name), CountryId (Id), and CountryName (Name). Below it, a form for adding a new record shows 'Id' as 1, 'Name' as 'Paris Charles de Gaulle Airport', and 'CountryId' as 20. A blue arrow points from the 'CountryId' field in the form to the 'CountryId' column in the table below. The table shows existing records with 'CountryId' values 1, 3, 7, 2, and 6, and corresponding names: Brazil, China, England, France, and Italy. A 'Properties' window for the 'CountryId' attribute shows its 'Initial value' is 20.

Id	Name
20	
1	Brazil
3	China
7	England
2	France
6	Italy

At runtime we see that, in the previously entered records, the CountryId attribute is assigned the value 20, even though we did not have any country with this ID. What GeneXus did in this case in order to assign CountryId the value 20, was to create a country in the Country table with ID 20 and empty name. Otherwise, there would be serious referential integrity conflicts.

If at some point we no longer want this attribute to be initialized with a value, we can modify this property and remove the value we entered. This will no longer impact the records we have entered.

It is important to remember that the initial value is assigned to the new attribute when the table is reorganized. If we didn't assign an initial value to the attribute when it was added and the reorganization is executed, then it will be necessary to run through all the records to assign a value to it.

As we also mentioned, when creating a new record, if there is an initial value assigned, the new record will be created with the attribute at that value.

Trn Airport

Name	Type	Description	Formula	Nullable
[-] Airport	Airport	Airport		
AirportId	Id	Airport Id		No
AirportName	Name	Airport Name		No
CountryId	Id	Country Id		No
CountryName	Name	Country Name		
AirportInternational	Boolean	Airport International		No
[-] Flight	Flight	Flight		
FlightId	Id	Flight Id		No
AirlineId	Id	Airline Id		
AirlineName	Name	Airline Name		
FlightFinalPrice	Price	Flight Final Price	FlightPrice * (1-AirlineDi...	
FlightCapacity	Numeric(4,0)	Flight Capacity	count(FlightSeatLocation)	

Tabla AirportFlight



Name	Type	Description	Formula
[-] AirportFlight Structure		Flight	
AirportId	Id	Airport Id	
FlightId	Id	Flight Id	

Now suppose we want to record from the Airport transaction the flights in each airport, so we enter a second level with the following attributes of the Flight transaction.

Since at the database level this structure does not generate changes within the Airport table, it does not impact previously entered records. A new AirportFlight table is generated and it will be empty.

Trn Airport

Name	Type	Description	Formula	Nullable
Airport	Airport	Airport		
AirportId	Id	Airport Id		No
AirportName	Name	Airport Name		No
CountryId	Id	Country Id		No
CountryName	Name	Country Name		
AirportInternational	Boolean	Airport International		No
AirportFlightCapacity	Numeric(4,0)	Airport Flight Capacity	sum(FlightCapacity)	
Flight	Flight	Flight		
FlightId	Id	Flight Id		No
AirlineId	Id	Airline Id		
AirlineName	Name	Airline Name		
FlightFinalPrice	Price	Flight Final Price	FlightPrice * (1-AirlineDis...	
FlightCapacity	Numeric(4,0)	Flight Capacity	count(FlightSeatLocation);	

Let's now assume that in the Flight transaction we have an attribute that counts the number of seats entered for that flight. Also, from Airport we want to know the total number of seats, adding up all the flights that have been entered in that airport, so as to obtain a total capacity.

For this, we have already created a formula attribute in the first level of the Airport transaction, with the SUM formula, which will add the value of the FlightCapacity attribute of the flights entered for each airport.

As we know, the attributes defined as a global formula are not stored in the database; instead, their value is calculated every time it is needed, but this data is not persisted.

Suppose you frequently need to access and work with this data. As it is calculated at runtime, if there are many records to be run through (for example, if there is a large number of flights entered), it can severely impact performance.

To solve this situation, GeneXus allows us to define a global formula attribute as redundant. This means that it will no longer be a virtual attribute; instead, it will be stored in its associated table and GeneXus will keep the knowledge that the attribute is a formula and how to calculate its value.

In this way, when executing the transaction that contains an attribute with a redundant formula, the formula will be evaluated and calculated, and the result will be stored in the database.

Then, when we need to query or work with that attribute again, GeneXus will retrieve the stored value from the database instead of spending time and effort to perform the calculation.

Redundant attribute

The screenshot shows the GeneXus Structure editor with two tabs: 'Airport' and 'Flight'. The 'Structure' tab is active, displaying a table of attributes. The 'AirportFlightCapacity' attribute is highlighted in blue, and its 'Redundant' checkbox is checked. A mouse cursor is visible over the checkbox.

Name	Redundant	Type	Description	Formula	Nullable
Airport		Airport	Airport		
AirportId		Id	Airport Id		No
AirportName		Name	Airport Name		No
CountryId		Id	Country Id		No
CountryName	<input type="checkbox"/>	Name	Country Name		
AirportInternational		Boolean	Airport International		No
AirportFlightCapacity	<input checked="" type="checkbox"/>	Numeric(4,0)	Airport Flight Capacity	sum(FlightCapacity)	
Flight		Flight	Flight		
FlightId		Id	Flight Id		No
AirlineId	<input type="checkbox"/>	Id	Airline Id		
AirlineName	<input type="checkbox"/>	Name	Airline Name		
FlightFinalPrice	<input type="checkbox"/>	Price	Flight Final Price	FlightPrice * (1-Airlin...	
FlightCapacity	<input type="checkbox"/>	Numeric(4,0)	Flight Capacity	count(FlightSeatLoca...	

To define this attribute as redundant, we right-click on the column headers row, select column chooser and there we see all the columns that we can include in the transaction editor. We choose redundant and drag it to the editor.

By means of a check box, it gives us the option to mark the attribute as redundant. We select it and run it again.

It will ask us to reorganize the database, since as we saw, the attribute we just marked as redundant will be added to it.

When a new Airport record is entered, the formula is triggered as usual in the transaction, adding up the total number of flights entered. And when confirming this value is also stored in the database.

What will happen to the Airport records that already existed in the database? In those records, what value will this attribute have in the table? When doing the reorganization and adding this attribute, GeneXus runs through the previously entered records one by one, triggering the formula and saving its value in the table. When viewing the previous records, the new attribute appears with its corresponding value, which as we said, is already stored in the database.

So far we have briefly seen how adding new attributes – whether they are inferred attributes, attributes with a redundant formula, or with a value in the Initial Value property – affects a transaction with previous records.

For more information, you can visit our [Wiki](#).

GeneXus[™]

training.genexus.com
wiki.genexus.com