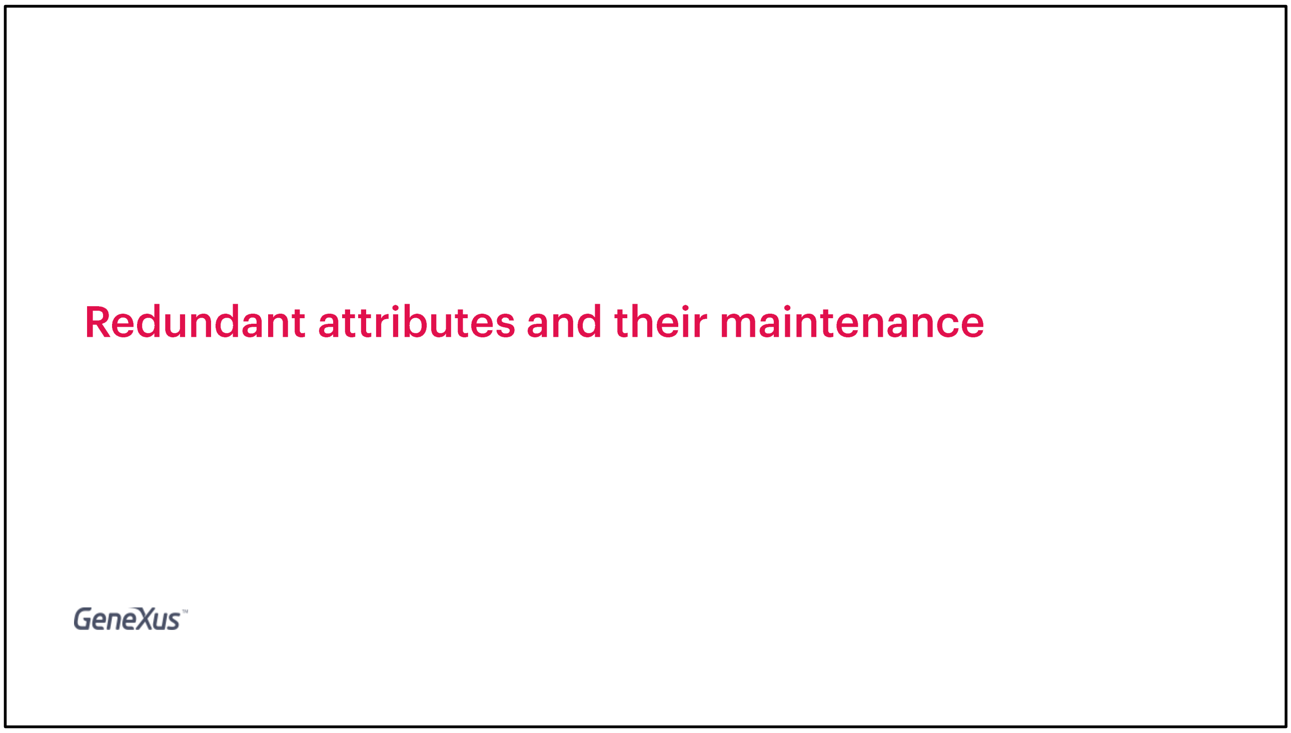
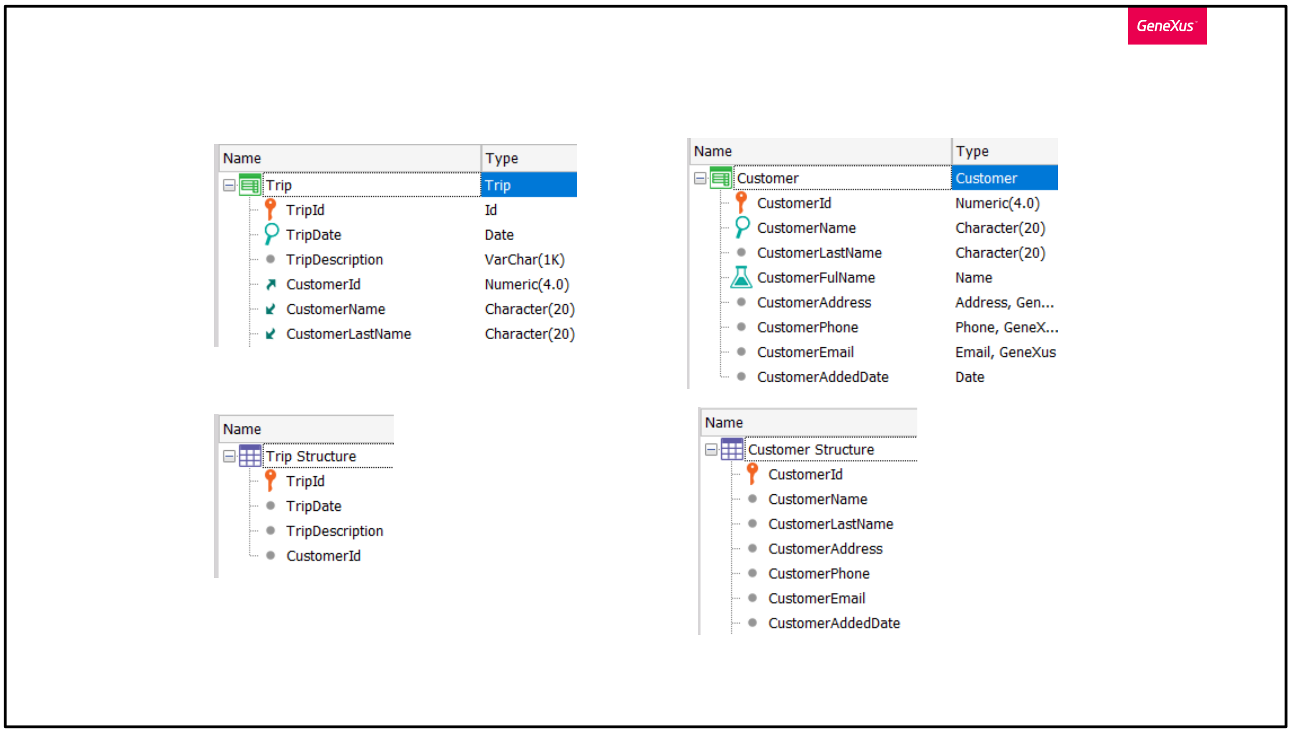


Redundant attributes and their maintenance

GeneXus™

In this video, we will see how to define inferred attributes or formulas–which, by definition, are not stored–as redundant and make them attributes of a database table.



Name	Type
Trip	Trip
TripId	Id
TripDate	Date
TripDescription	VarChar(1K)
CustomerId	Numeric(4.0)
CustomerName	Character(20)
CustomerLastName	Character(20)

Name	Type
Customer	Customer
CustomerId	Numeric(4.0)
CustomerName	Character(20)
CustomerLastName	Character(20)
CustomerFulName	Name
CustomerAddress	Address, Gen...
CustomerPhone	Phone, GeneX...
CustomerEmail	Email, GeneXus
CustomerAddedDate	Date

Name
Trip Structure
TripId
TripDate
TripDescription
CustomerId

Name
Customer Structure
CustomerId
CustomerName
CustomerLastName
CustomerAddress
CustomerPhone
CustomerEmail
CustomerAddedDate

As we know, GeneXus automatically normalizes the database in Third Normal Form, which implies that the only attributes that can be in more than one table are those attributes that are a primary key and perform foreign key functions.

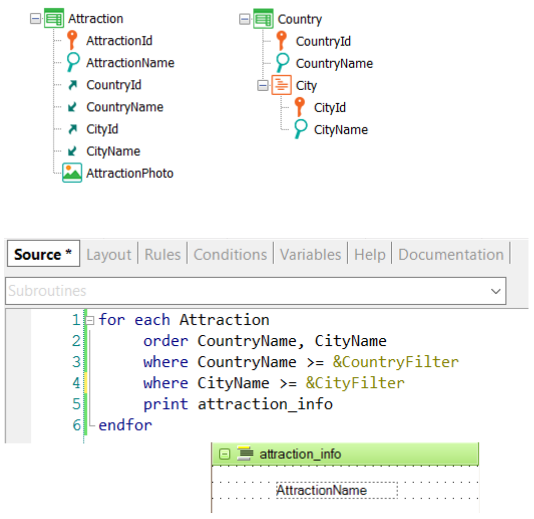
The rest of the attributes, which are called secondary, are stored in a single table and determined by the primary key. If they are added to a different transaction, GeneXus will infer them, retrieving their value from the table where they are stored by means of the foreign key.

In addition, when we define an attribute as a formula in a transaction, it is no longer stored and becomes a virtual attribute.

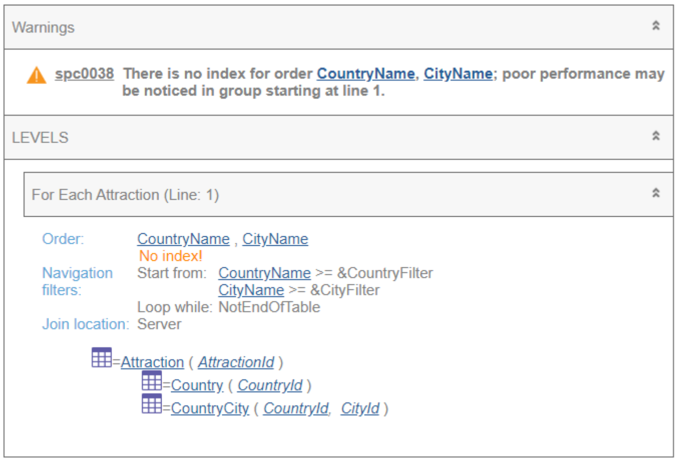
However, for performance reasons, in some cases we want to allow an inferred attribute to be stored in the table associated with the transaction where it is inferred, or a formula attribute that must perform many calculations and is time consuming whenever its value is retrieved to be stored in its associated table in order to obtain the value more quickly.

GeneXus allows storing an attribute that by default is not stored in a table, defining it as redundant.

Referencial redundancy



```
for each Attraction
     order CountryName, CityName
     where CountryName >= &CountryFilter
     where CityName >= &CityFilter
     print attraction_info
endfor
```


Warnings

spc0038 There is no index for order CountryName, CityName; poor performance may be noticed in group starting at line 1.

LEVELS

For Each Attraction (Line: 1)

```
Order:             CountryName , CityName
                   No index!
Navigation         Start from:  CountryName >= &CountryFilter
filters:                        CityName >= &CityFilter
                   Loop while:  NotEndOfTable
Join location:     Server

     =Attraction ( AttractionId )
         =Country ( CountryId )
         =CountryCity ( CountryId,  CityId )
```

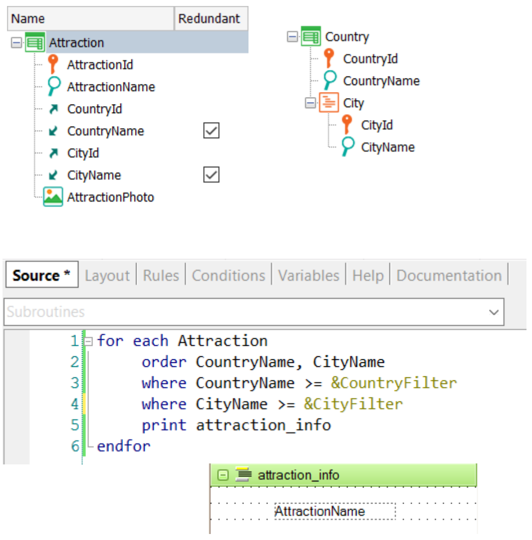
When we make an inferred attribute redundant, it is called referential redundancy.

This approach to attribute redundancy was mainly motivated by the need to improve performance.

Let's suppose that the table of tourist attractions had millions of records, and we would like to retrieve those corresponding to countries whose name is alphabetically later than a given value, and whose city name is later than another.
For optimization purposes, we know that sorting by filter attributes is the most convenient option.

If we look at the navigation list, on one hand we see that the search is optimized in terms of navigation filters, but note that two joins will have to be made since CountryName and CityName are not in the Attraction table being run through. However, if they were in the table...

Referencial redundancy



Name	Redundant
Attraction	
AttractionId	
AttractionName	
CountryId	
CountryName	✓
CityId	
CityName	✓
AttractionPhoto	

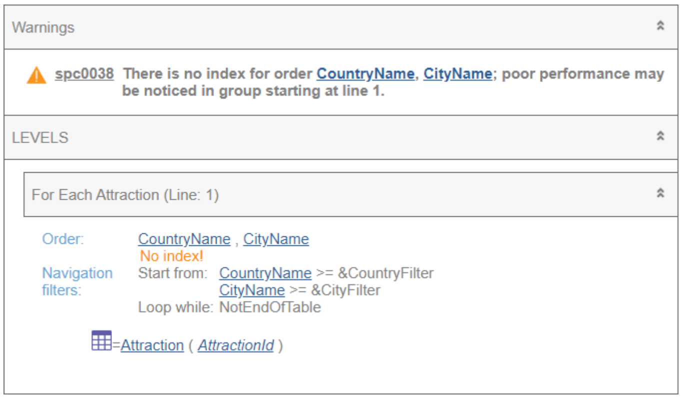
Country
- CountryId
- CountryName
- City
  - CityId
  - CityName

Source * | Layout | Rules | Conditions | Variables | Help | Documentation

```
for each Attraction
     order CountryName, CityName
     where CountryName >= &CountryFilter
     where CityName >= &CityFilter
     print attraction_info
endfor
```

attraction_info

AttractionName



Warnings

spc0038 There is no index for order CountryName, CityName; poor performance may be noticed in group starting at line 1.

LEVELS

For Each Attraction (Line: 1)

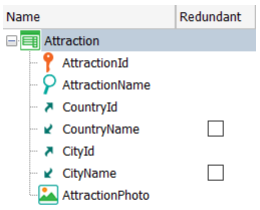
```
Order:          CountryName , CityName
                No index!
Navigation      Start from:  CountryName >= &CountryFilter
filters:                     CityName >= &CityFilter
                Loop while:  NotEndOfTable

        =Attraction ( AttractionId )
```

...there would be no join to perform, so the performance would improve.

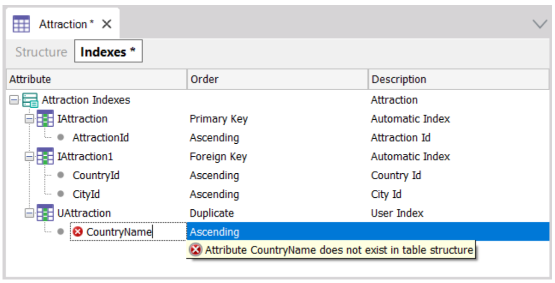
In both cases, we are informed that because there is no index for the order we defined, we could notice performance problems.The intelligence and capabilities provided will depend on the DBMS being used. We know that DBMSs now are much more intelligent than in the past and have strategies to optimize searches.
However, in some cases to solve a query it will be necessary to create a temporary index that will be deleted after the query, and so on every time the query is executed.

If this were the case, and we needed, precisely to avoid this continuous creation of an index and its subsequent elimination, to create a user index for these attributes...

Referencial redundancy



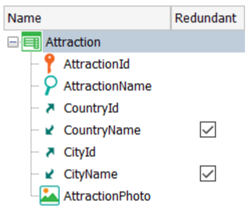
Name	Redundant
Attraction	
AttractionId	
AttractionName	
CountryId	
CountryName	☐
CityId	
CityName	☐
AttractionPhoto	



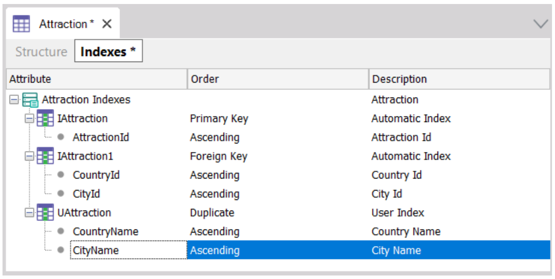
Attraction * | Structure | Indexes *

Attribute	Order	Description
Attraction Indexes		Attraction
IAttraction	Primary Key	Automatic Index
AttractionId	Ascending	Attraction Id
IAttraction1	Foreign Key	Automatic Index
CountryId	Ascending	Country Id
CityId	Ascending	City Id
UAttraction	Duplicate	User Index
CountryName	Ascending	

Attribute CountryName does not exist in table structure



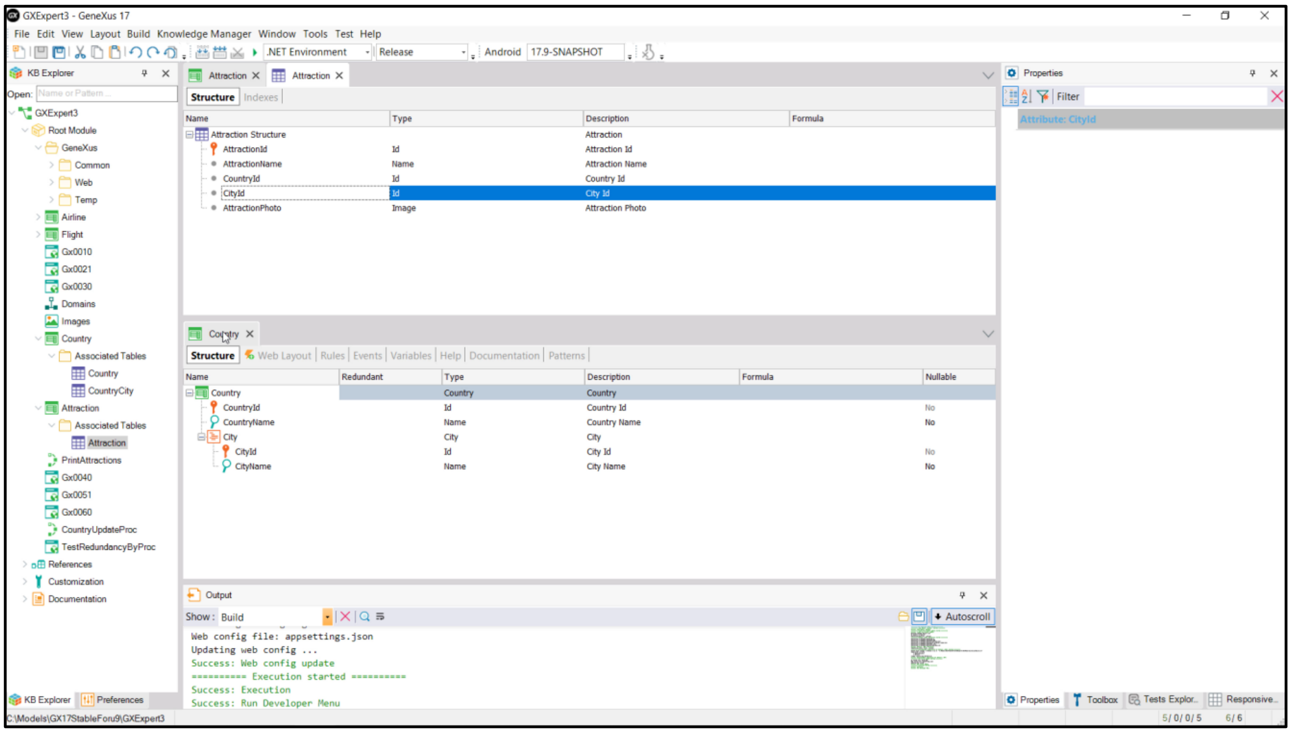
Name	Redundant
Attraction	
AttractionId	
AttractionName	
CountryId	
CountryName	☑
CityId	
CityName	☑
AttractionPhoto	



Attraction * | Structure | Indexes *

Attribute	Order	Description
Attraction Indexes		Attraction
IAttraction	Primary Key	Automatic Index
AttractionId	Ascending	Attraction Id
IAttraction1	Foreign Key	Automatic Index
CountryId	Ascending	Country Id
CityId	Ascending	City Id
UAttraction	Duplicate	User Index
CountryName	Ascending	Country Name
CityName	Ascending	City Name

...we will only be able to do it if both are in the Attraction table.



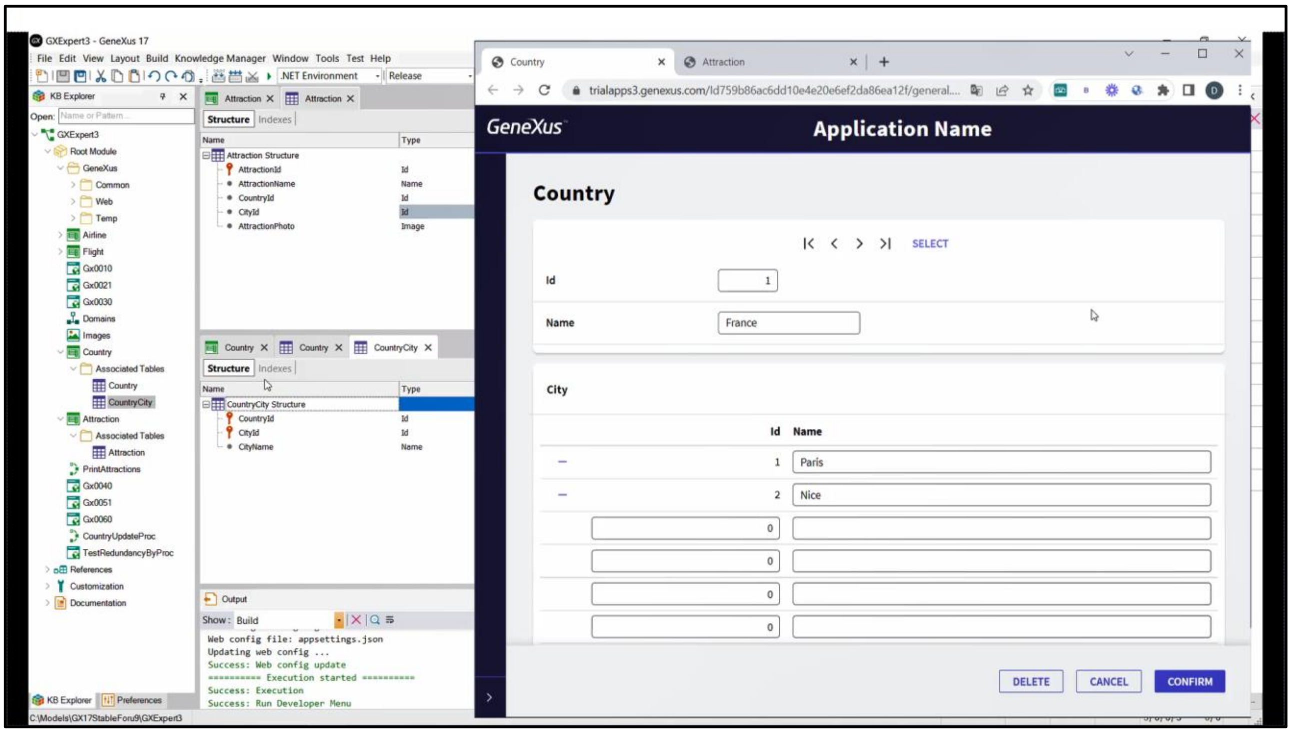
Name	Type	Description	Formula
Attraction Structure		Attraction	
AttractionId	Id	Attraction Id	
AttractionName	Name	Attraction Name	
CountryId	Id	Country Id	
CityId	Id	City Id	
AttractionPhoto	Image	Attraction Photo	

Name	Redundant	Type	Description	Formula	Nullable
Country		Country	Country		
CountryId		Id	Country Id		No
CountryName		Name	Country Name		No
City		City	City		
CityId		Id	City Id		No
CityName		Name	City Name		No

```
Web config file: appsettings.json
Updating web config ...
Success: Web config update
========== Execution started ==========
Success: Execution
Success: Run Developer Menu
```

Let's see in GeneXus how to declare redundancy and its effects.

We have the Attractions transaction that infers, as usual, the values of CountryName and CityName from the foreign keys. In fact, if we look at the structure of the attractions table we can see that these attributes are not present, and only the foreign keys are there.



Name	Type
Attraction Structure	
AttractionId	Id
AttractionName	Name
CountryId	Id
CityId	Id
AttractionPhoto	Image

Name	Type
CountryCity Structure	
CountryId	Id
CityId	Id
CityName	Name

```
Web config file: appsettings.json
Updating web config ...
Success: Web config update
========== Execution started ==========
Success: Execution
Success: Run Developer Menu
```

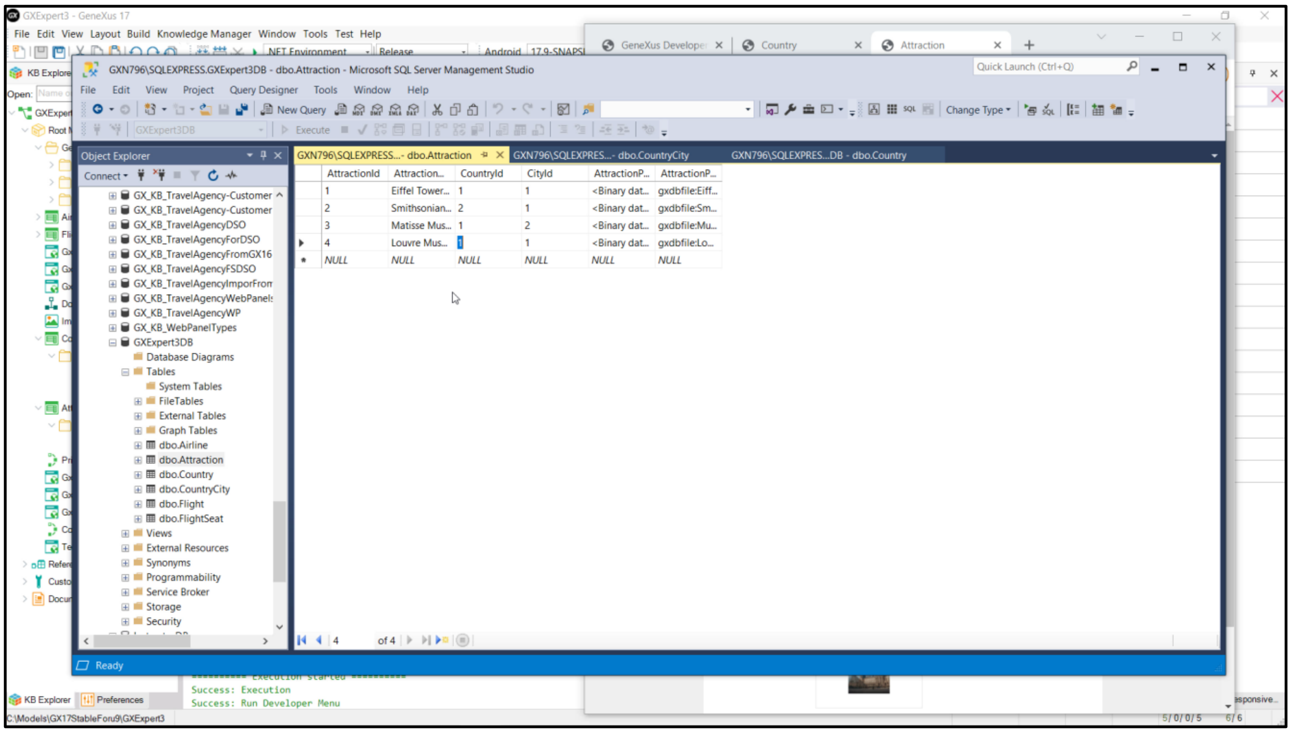
Country

Id	1
Name	France

City

	Id	Name
–	1	Paris
–	2	Nice
	0	
	0	
	0	
	0	

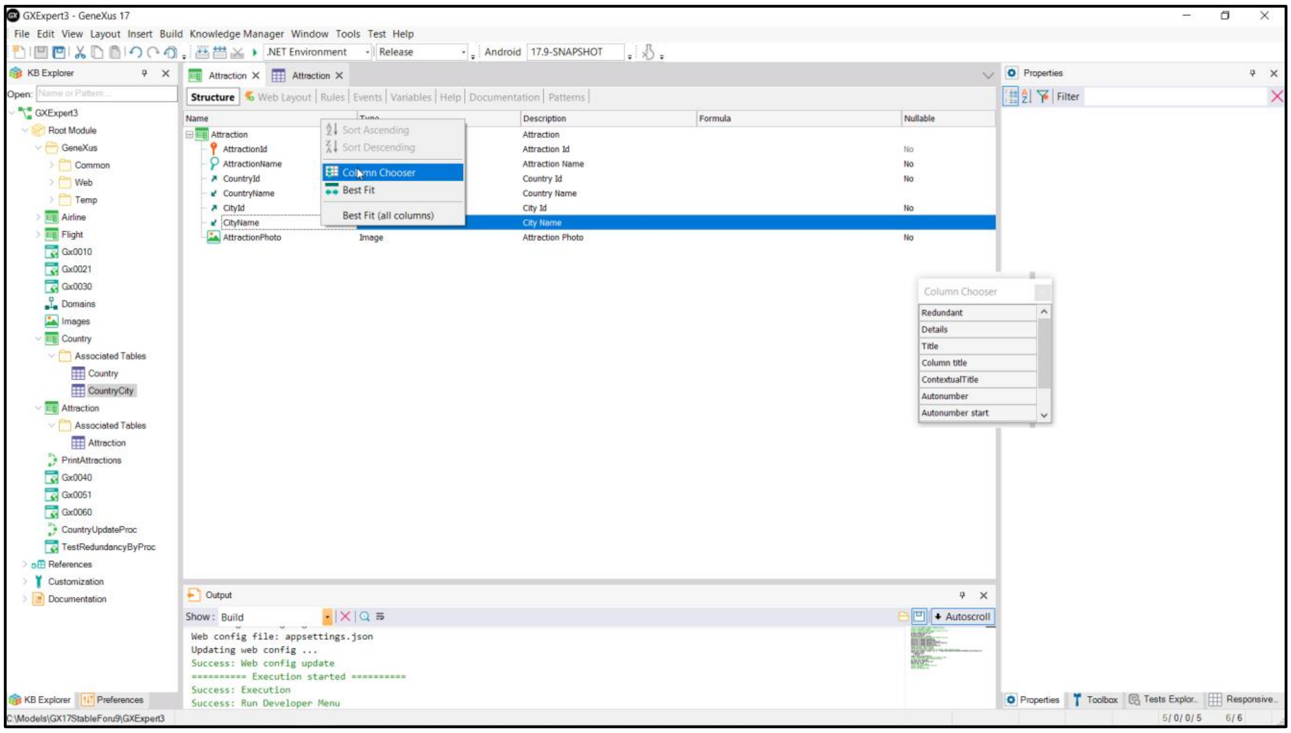
We already have some data loaded at runtime: two countries with their cities. Let's pay special attention to 1, France, with its two cities Paris and Nice. Then we see that we have 3 attractions in France: two in Paris and one in Nice.



AttractionId	Attraction...	CountryId	CityId	AttractionP...	AttractionP...
1	Eiffel Tower...	1	1	<Binary dat...	gxdbfile:Eiff...
2	Smithsonian...	2	1	<Binary dat...	gxdbfile:Sm...
3	Matisse Mus...	1	2	<Binary dat...	gxdbfile:Mu...
4	Louvre Mus...	1	1	<Binary dat...	gxdbfile:Lo...
NULL	NULL	NULL	NULL	NULL	NULL

```
Success: Execution
Success: Run Developer Menu
```

If we look for the data of the tables in SQL Server... we see that in the Attractions table there are only the foreign keys.

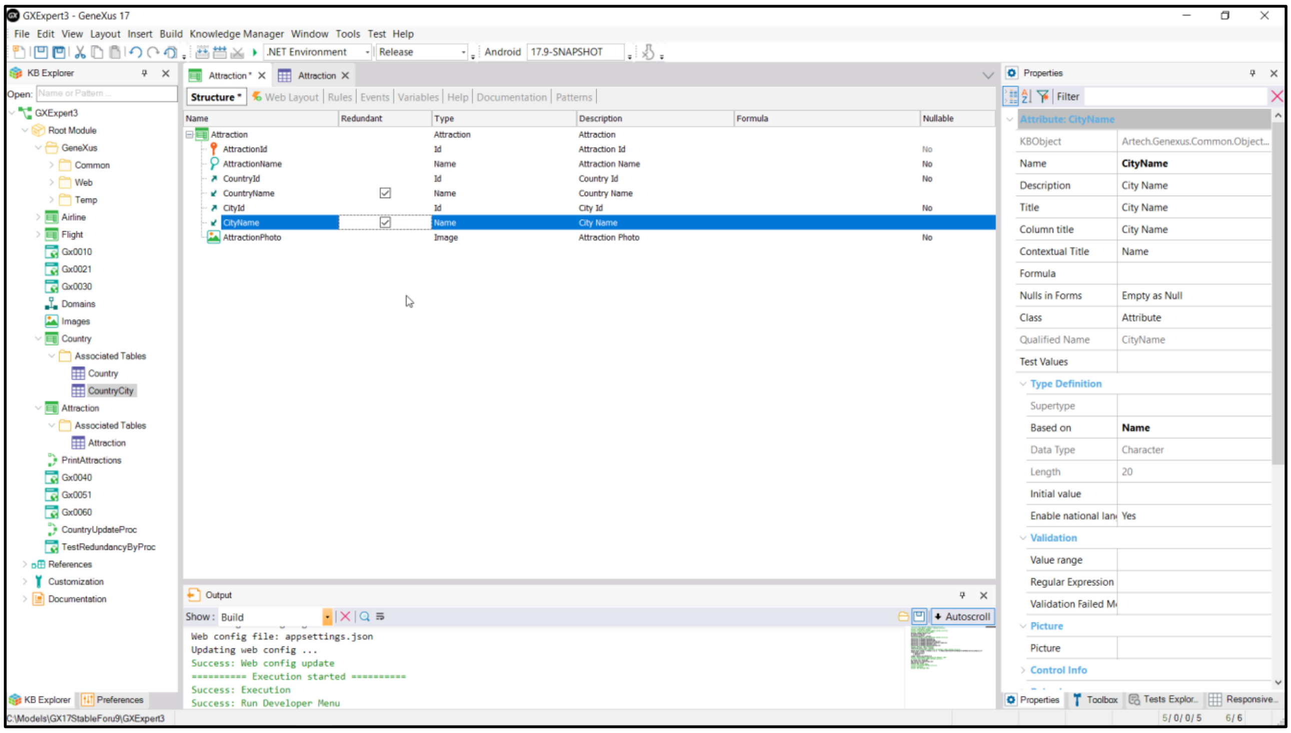


Name	Type	Description	Formula	Nullable
Attraction		Attraction		
AttractionId		Attraction Id		No
AttractionName		Attraction Name		No
CountryId		Country Id		No
CountryName		Country Name		
CityId		City Id		No
CityName		City Name		
AttractionPhoto	Image	Attraction Photo		No

```
Web config file: appsettings.json
Updating web config ...
Success: Web config update
========== Execution started ==========
Success: Execution
Success: Run Developer Menu
```

C:\Models\GX17StableForu9\GXExpert3

Well, now in Attraction we want to define the two inferred attributes as redundant. To do so, in the editor of the transaction structure, we add this column...



Name	Redundant	Type	Description	Formula	Nullable
Attraction		Attraction	Attraction		
AttractionId		Id	Attraction Id		No
AttractionName		Name	Attraction Name		No
CountryId		Id	Country Id		No
CountryName	✓	Name	Country Name		
CityId		Id	City Id		No
CityName	✓	Name	City Name		
AttractionPhoto		Image	Attraction Photo		No

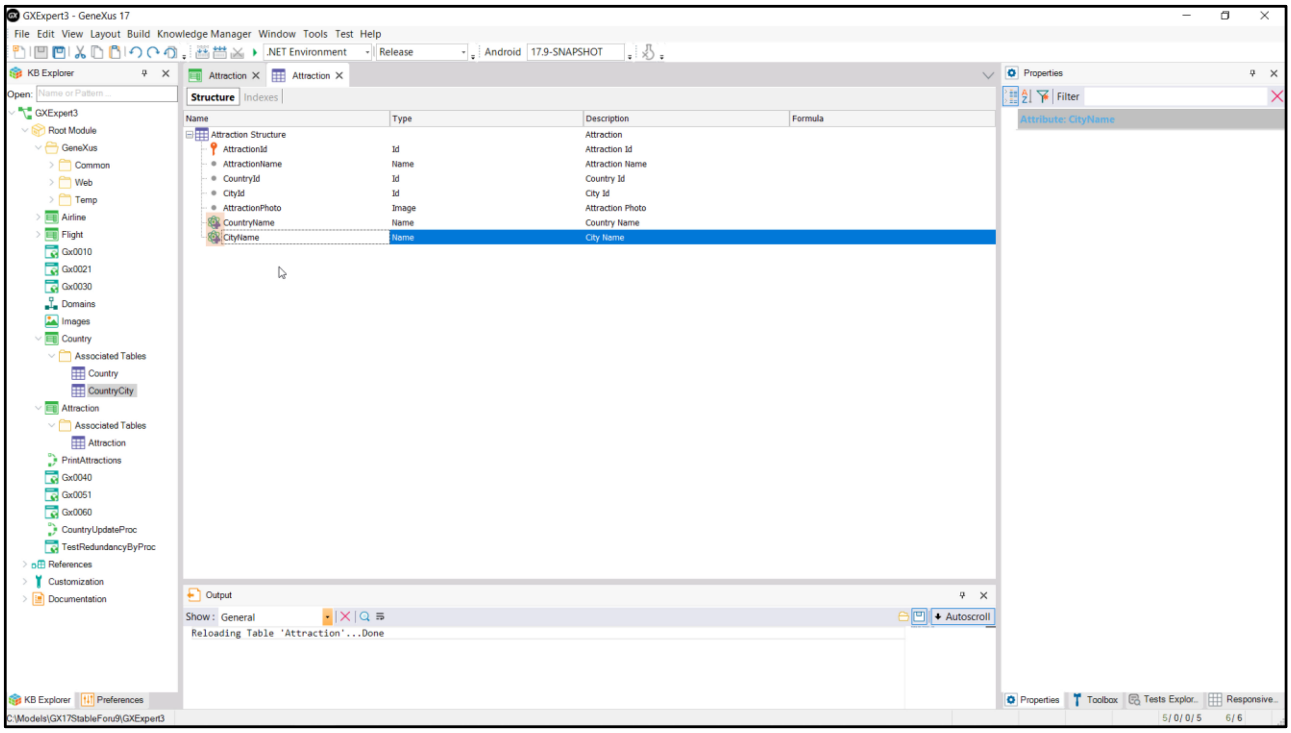
Attribute: CityName

KBObject	Artech.Genexus.Common.Object...
Name	CityName
Description	City Name
Title	City Name
Column title	City Name
Contextual Title	Name
Formula	
Nulls in Forms	Empty as Null
Class	Attribute
Qualified Name	CityName
Test Values	
Type Definition	
Supertype	
Based on	Name
Data Type	Character
Length	20
Initial value	
Enable national lan	Yes
Validation	
Value range	
Regular Expression	
Validation Failed M	
Picture	
Picture	
Control Info	

```
Web config file: appsettings.json
Updating web config ...
Success: Web config update
========== Execution started ==========
Success: Execution
Success: Run Developer Menu
```

... which provides check boxes to indicate which attributes of the transaction structure we want to make redundant. It only offers us to make the inferred attributes redundant. If there were formula attributes, it would also offer us to make them redundant, as we will see.

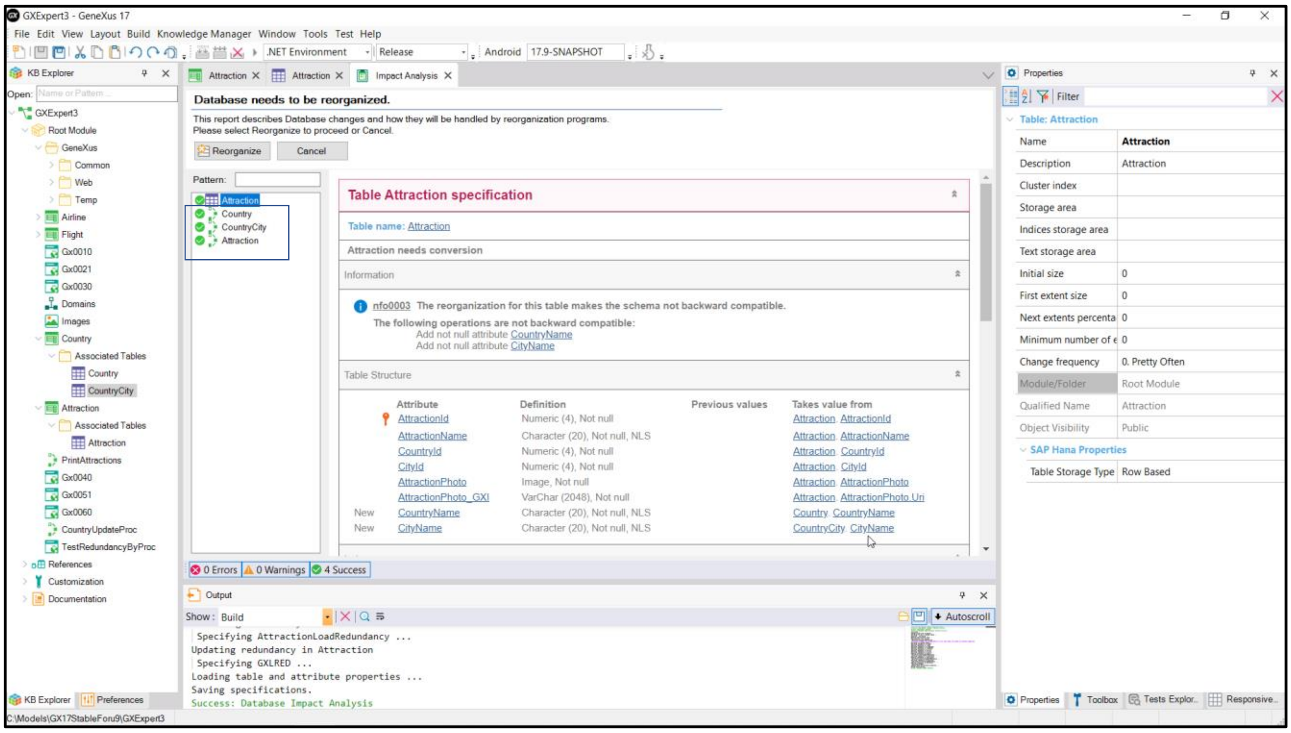
We are going to select both, because we want both to be redundant in the Attraction table.



Name	Type	Description	Formula
Attraction Structure		Attraction	
AttractionId	Id	Attraction Id	
AttractionName	Name	Attraction Name	
CountryId	Id	Country Id	
CityId	Id	City Id	
AttractionPhoto	Image	Attraction Photo	
CountryName	Name	Country Name	
CityName	Name	City Name	

```
Reloading Table 'Attraction'...Done
```

Let's save. We see that the attributes have been added to the table structure and it is indicated in this way that they are redundant.



Database needs to be reorganized.

This report describes Database changes and how they will be handled by reorganization programs.
Please select Reorganize to proceed or Cancel.

Table Attraction specification

Table name: Attraction

Attraction needs conversion

Information

nfo0003 The reorganization for this table makes the schema not backward compatible.
The following operations are not backward compatible:
Add not null attribute CountryName
Add not null attribute CityName

Table Structure

	Attribute	Definition	Previous values	Takes value from
	AttractionId	Numeric (4), Not null		Attraction. AttractionId
	AttractionName	Character (20), Not null, NLS		Attraction. AttractionName
	CountryId	Numeric (4), Not null		Attraction. CountryId
	CityId	Numeric (4), Not null		Attraction. CityId
	AttractionPhoto	Image, Not null		Attraction. AttractionPhoto
	AttractionPhoto_GXI	VarChar (2048), Not null		Attraction. AttractionPhoto.Uri
New	CountryName	Character (20), Not null, NLS		Country. CountryName
New	CityName	Character (20), Not null, NLS		CountryCity. CityName

0 Errors 0 Warnings 4 Success

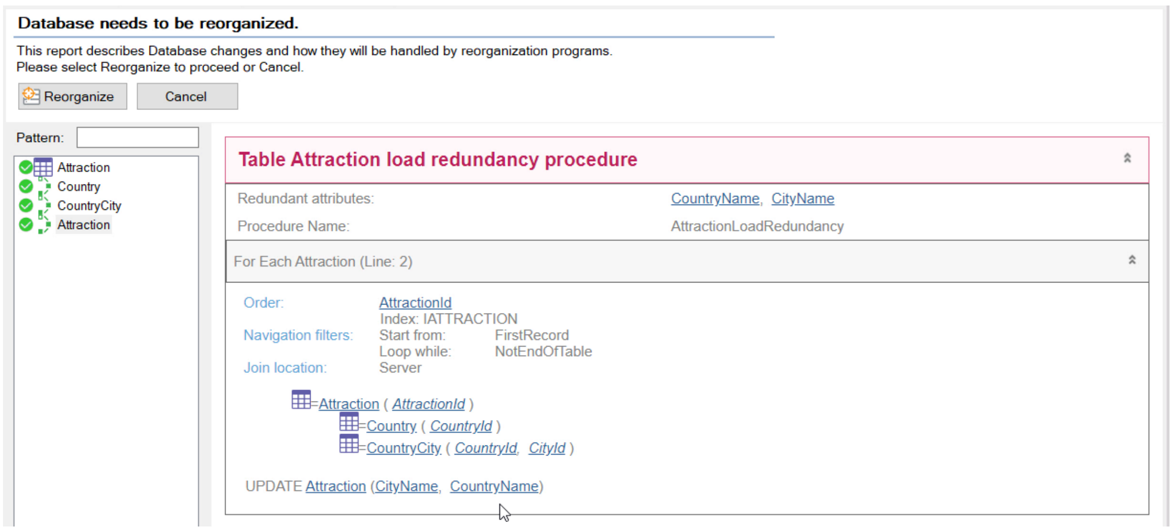
Output — Show: Build

```
 Specifying AttractionLoadRedundancy ...
Updating redundancy in Attraction
 Specifying GXLRED ...
Loading table and attribute properties ...
Saving specifications.
Success: Database Impact Analysis
```

Properties — Table: Attraction

Name	Attraction
Description	Attraction
Cluster index	
Storage area	
Indices storage area	
Text storage area	
Initial size	0
First extent size	0
Next extents percenta	0
Minimum number of e	0
Change frequency	0. Pretty Often
Module/Folder	Root Module
Qualified Name	Attraction
Object Visibility	Public
Table Storage Type	Row Based

If we now execute it, clearly we will have to reorganize the Attraction table to add those attributes that so far were inferred but now we want them to be stored.
Note that it is indicated where their values will be taken from.
Well, we expected this. But, what are these three internal procedures that GeneXus will create?



Database needs to be reorganized.

This report describes Database changes and how they will be handled by reorganization programs.
Please select Reorganize to proceed or Cancel.

Pattern:

- Attraction
- Country
- CountryCity
- Attraction

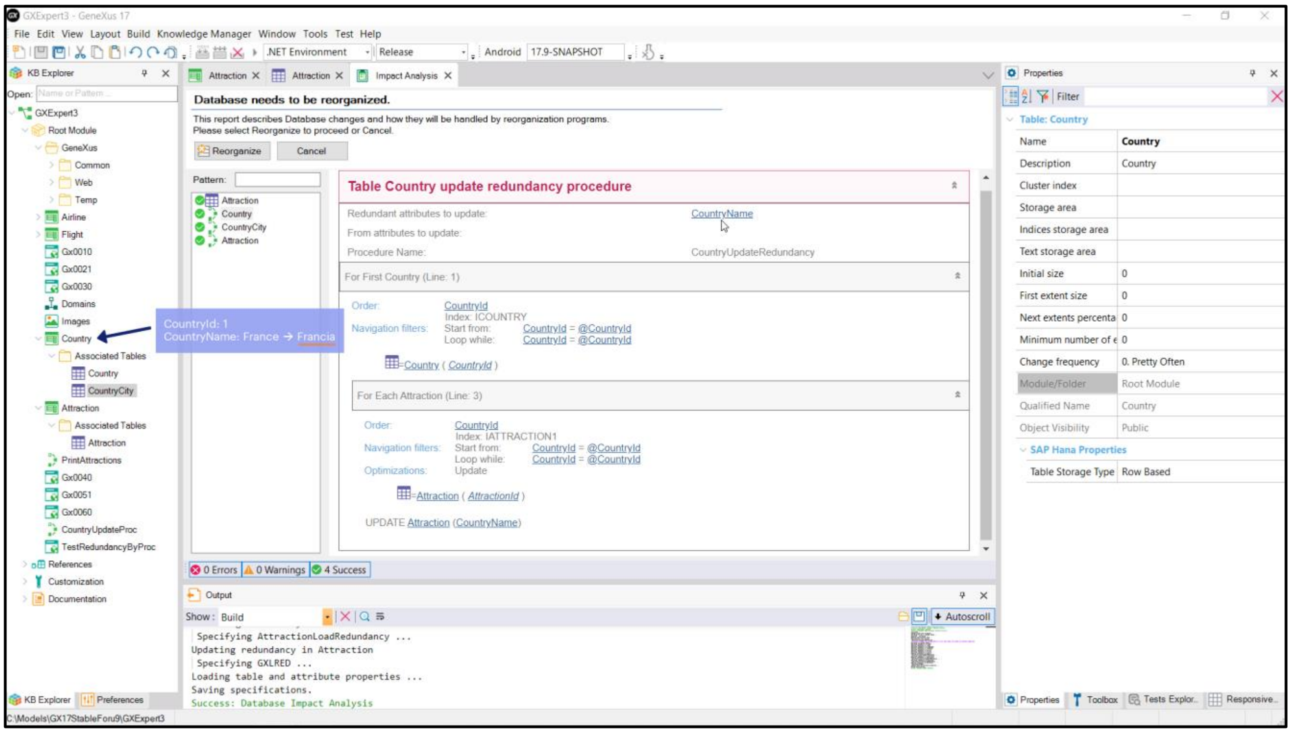
Table Attraction load redundancy procedure

Redundant attributes:	CountryName, CityName
Procedure Name:	AttractionLoadRedundancy

For Each Attraction (Line: 2)

```
Order:               AttractionId
                     Index: IATTRACTION
Navigation filters:  Start from:   FirstRecord
                     Loop while:   NotEndOfTable
Join location:       Server

        =Attraction ( AttractionId )
            =Country ( CountryId )
            =CountryCity ( CountryId, CityId )

UPDATE Attraction (CityName, CountryName)
```

Let's start by examining the last one. Its name is AttractionLoadRedundancy. Here it says that it is the procedure for loading the redundancies of the Attraction table. And it tells us which redundant attributes it has to load. What this procedure will do is to run through the entire table of attractions. For each one, it will go to Country to find the value of the CountryName attribute and store it in the redundant attribute of this Attraction table. Also, it will go to the cities table to find the value of the CityName attribute and store it in the new redundant CityName attribute in this Attraction table.
In sum, it must automatically execute this procedure after reorganizing the Attraction table so that the redundant attributes are loaded with the corresponding values.



Database needs to be reorganized.

This report describes Database changes and how they will be handled by reorganization programs.
Please select Reorganize to proceed or Cancel.

Table Country update redundancy procedure

Redundant attributes to update: CountryName
From attributes to update:
Procedure Name: CountryUpdateRedundancy

For First Country (Line: 1)

Order: CountryId
Index: ICOUNTRY
Navigation filters: Start from: CountryId = @CountryId
Loop while: CountryId = @CountryId

=Country (CountryId)

For Each Attraction (Line: 3)

Order: CountryId
Index: IATTRACTION1
Navigation filters: Start from: CountryId = @CountryId
Loop while: CountryId = @CountryId
Optimizations: Update

=Attraction (AttractionId)

UPDATE Attraction (CountryName)

0 Errors | 0 Warnings | 4 Success

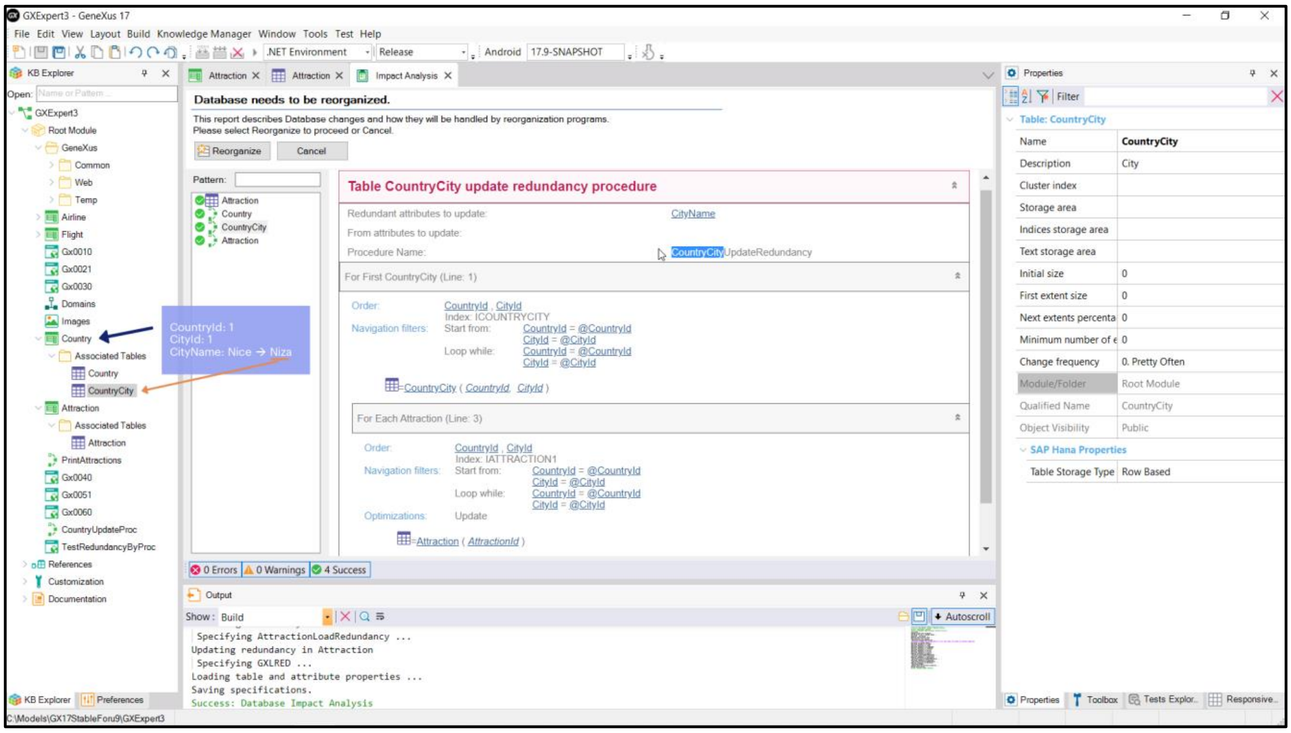
```
 Specifying AttractionLoadRedundancy ...
Updating redundancy in Attraction
 Specifying GXLRED ...
Loading table and attribute properties ...
Saving specifications.
Success: Database Impact Analysis
```

Table: Country

Name	**Country**
Description	Country
Cluster index	
Storage area	
Indices storage area	
Text storage area	
Initial size	0
First extent size	0
Next extents percenta	0
Minimum number of e	0
Change frequency	0. Pretty Often
Module/Folder	Root Module
Qualified Name	Country
Object Visibility	Public
SAP Hana Properties	
Table Storage Type	Row Based

Ok, but what about these two procedures?

If we look at the first one, we see that it is a procedure to update the redundancies of the Country table. We know that the CountryName attribute of this table is redundant in Attraction. This means that if we execute the transaction and change the name of a country, this change should also be made for all the attractions that correspond to that country in order to keep the redundancy up to date. This is what this procedure will do, which will be invoked transparently every time the user changes the value of CountryName for a CountryId through the Country transaction (or its business component). Here we can see that the procedure receives as parameter the value of the primary key to instantiate that record; then it runs through the table of attractions filtering by that value, and for each attraction it updates the CountryName in that table.



Database needs to be reorganized.

This report describes Database changes and how they will be handled by reorganization programs.
Please select Reorganize to proceed or Cancel.

Pattern:
- Attraction
- Country
- CountryCity
- Attraction

CountryId: 1
CityId: 1
CityName: Nice → Niza

Table CountryCity update redundancy procedure

Redundant attributes to update: CityName
From attributes to update:
Procedure Name: CountryCityUpdateRedundancy

For First CountryCity (Line: 1)

```
Order:               CountryId , CityId
                     Index: ICOUNTRYCITY
Navigation filters:  Start from:   CountryId = @CountryId
                                   CityId = @CityId
                     Loop while:   CountryId = @CountryId
                                   CityId = @CityId

     =CountryCity ( CountryId, CityId )
```

For Each Attraction (Line: 3)

```
Order:               CountryId , CityId
                     Index: IATTRACTION1
Navigation filters:  Start from:   CountryId = @CountryId
                                   CityId = @CityId
                     Loop while:   CountryId = @CountryId
                                   CityId = @CityId
Optimizations:       Update

     =Attraction ( AttractionId )
```

0 Errors | 0 Warnings | 4 Success

Output — Show: Build

```
 Specifying AttractionLoadRedundancy ...
Updating redundancy in Attraction
 Specifying GXLRED ...
Loading table and attribute properties ...
Saving specifications.
Success: Database Impact Analysis
```

KB Explorer tree: GXExpert3, Root Module, GeneXus, Common, Web, Temp, Airline, Flight, Gx0010, Gx0021, Gx0030, Domains, Images, Country, Associated Tables, Country, CountryCity, Attraction, Associated Tables, Attraction, PrintAttractions, Gx0040, Gx0051, Gx0060, CountryUpdateProc, TestRedundancyByProc, References, Customization, Documentation

C:\Models\GX17StableForu9\GXExpert3

Table: CountryCity

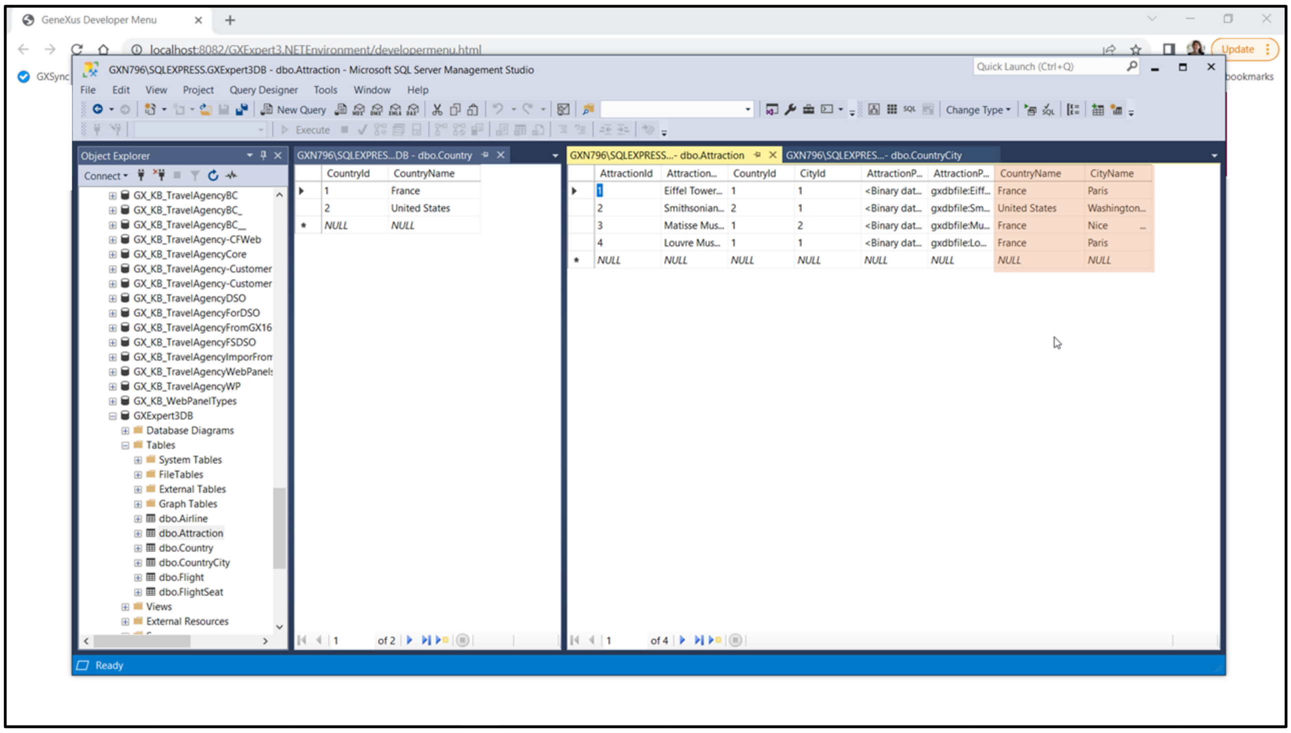
Name	CountryCity
Description	City
Cluster index	
Storage area	
Indices storage area	
Text storage area	
Initial size	0
First extent size	0
Next extents percenta	0
Minimum number of e	0
Change frequency	0. Pretty Often
Module/Folder	Root Module
Qualified Name	CountryCity
Object Visibility	Public

SAP Hana Properties

Table Storage Type	Row Based

What about this other procedure? The same goes, but to keep the redundancies relative to the cities table updated. That is to say, when the name of a city in a country is modified from the Country transaction, this procedure will be automatically and transparently executed. The country ID and the city ID will be sent to it by parameter, this table record will be accessed, and for each matching Attraction record, the value of the redundant attribute CityName will be modified in Attraction.

These procedures will then be created in this reorganization, and the logic we just explained will be added to the transactions.
We reorganize.

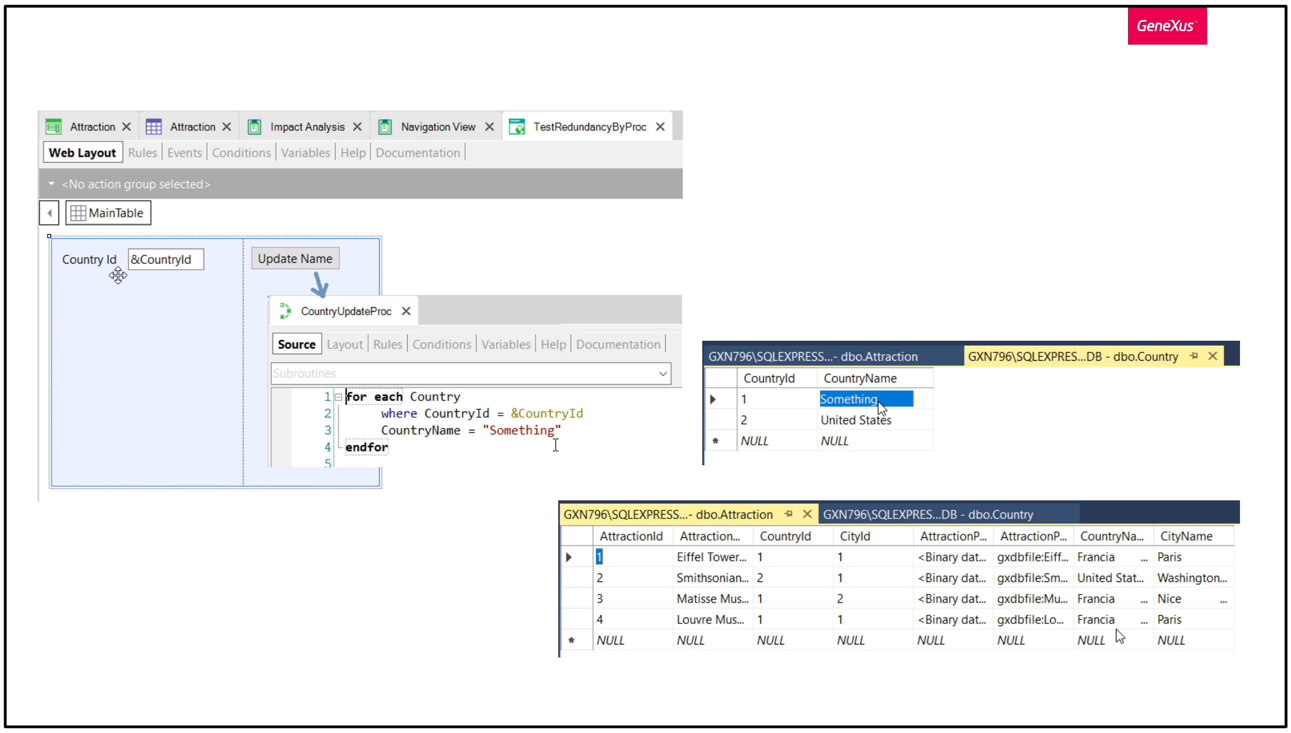

GXN796\SQLEXPRESS.GXExpert3DB - dbo.Attraction - Microsoft SQL Server Management Studio
GXN796\SQLEXPRES...DB - dbo.Country
GXN796\SQLEXPRESS...- dbo.Attraction
GXN796\SQLEXPRES...- dbo.CountryCity

	CountryId	CountryName
▶	1	France
	2	United States
*	NULL	NULL

	AttractionId	Attraction...	CountryId	CityId	AttractionP...	AttractionP...	CountryName	CityName
▶	1	Eiffel Tower...	1	1	<Binary dat...	gxdbfile:Eiff...	France	Paris
	2	Smithsonian...	2	1	<Binary dat...	gxdbfile:Sm...	United States	Washington...
	3	Matisse Mus...	1	2	<Binary dat...	gxdbfile:Mu...	France	Nice
	4	Louvre Mus...	1	1	<Binary dat...	gxdbfile:Lo...	France	Paris
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Now let's see the table in SQL Server. The redundant attributes were added and assigned their corresponding value.
We are now going to change the name of the country France to its name in Spanish, Francia. We will do it through the transaction.
Let's look at the table data... the name in the transaction table has changed... and let's see what happened to the redundant attribute... just as we expected, it was updated.
Likewise, if we are going to modify the name of a city, for example Nice, writing it now in Spanish (Niza)... we see that in the cities table it has been changed obviously, and now in the Attractions table... too.

That was due to the modification made through the Country transaction.


GeneXus
Attraction
Attraction
Impact Analysis
Navigation View
TestRedundancyByProc
Web Layout | Rules | Events | Conditions | Variables | Help | Documentation
<No action group selected>
MainTable
Country Id
&CountryId
Update Name
CountryUpdateProc
Source | Layout | Rules | Conditions | Variables | Help | Documentation
Subroutines

```
for each Country
     where CountryId = &CountryId
     CountryName = "Something"
endfor
```

GXN796\SQLEXPRESS...- dbo.Attraction | GXN796\SQLEXPRES...DB - dbo.Country

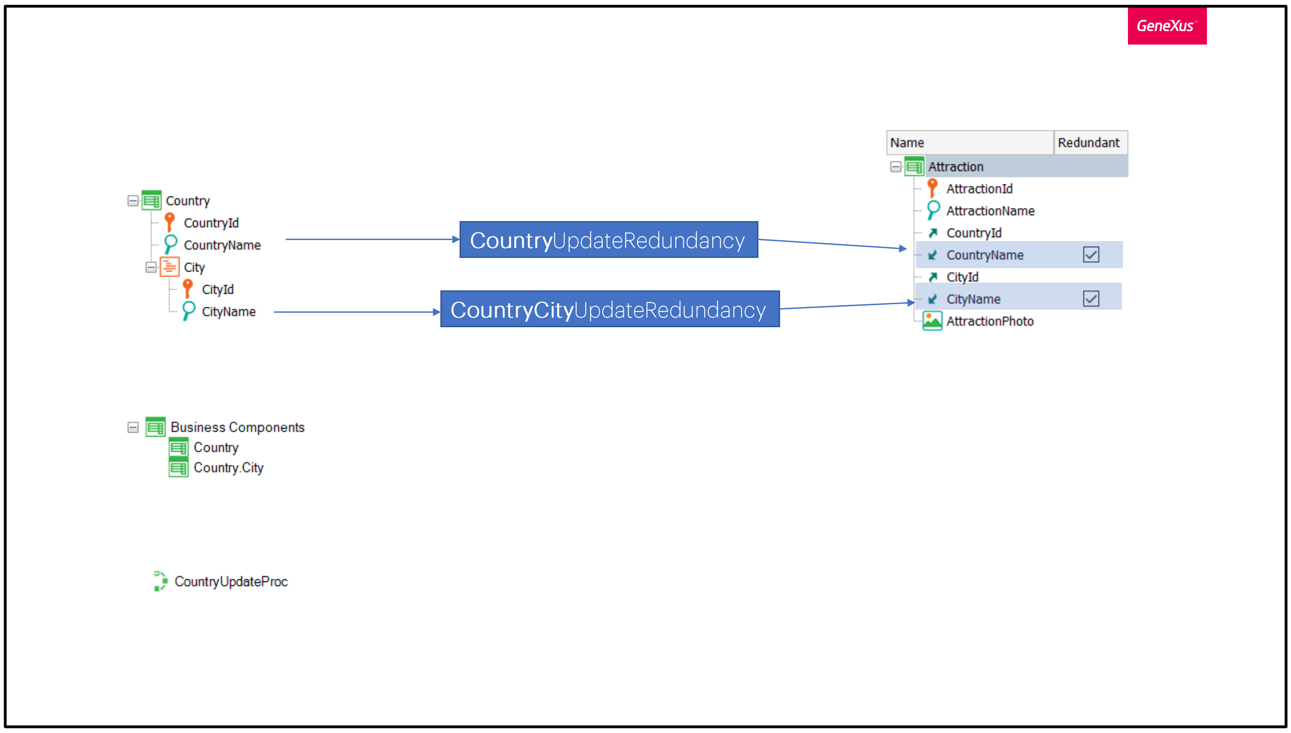
	CountryId	CountryName
▶	1	Something
	2	United States
*	NULL	NULL

GXN796\SQLEXPRESS...- dbo.Attraction | GXN796\SQLEXPRES...DB - dbo.Country

	AttractionId	Attraction...	CountryId	CityId	AttractionP...	AttractionP...	CountryNa...	CityName
▶	1	Eiffel Tower...	1	1	<Binary dat...	gxdbfile:Eiff...	Francia ...	Paris
	2	Smithsonian...	2	1	<Binary dat...	gxdbfile:Sm...	United Stat...	Washington...
	3	Matisse Mus...	1	2	<Binary dat...	gxdbfile:Mu...	Francia ...	Nice ...
	4	Louvre Mus...	1	1	<Binary dat...	gxdbfile:Lo...	Francia ...	Paris
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Let's see what happens if we do it through a procedure... We have a web panel for the user to enter a country ID, and when pressing the button we invoke a procedure that receives that ID and goes to the Country table to change the value of the CountryName attribute for the country with that ID to “Something.” Let's run it to test it.

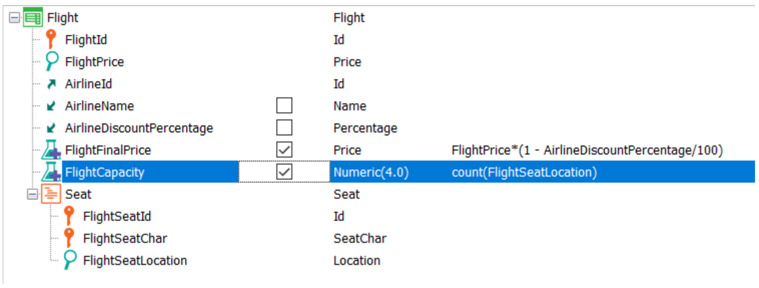
We choose the country with ID 1, which is France. In the Country table, its value was indeed changed to the one assigned by the procedure. But now let's see if the procedure kept the redundancy in Attraction up to date. No, it did not.



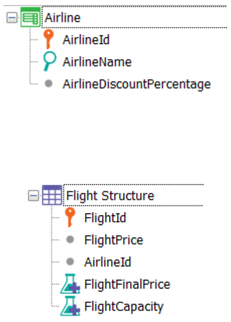
GeneXus invokes those procedures that we saw whose names ended in UpdateRedundancy only when the modifications are made through the transaction (or the business component, logically). It doesn't invoke them when the modifications are made in any other way.

So, we must be careful. In case of modifying attributes that are redundant in other tables through another way than the transaction or the business component, the developer will be responsible for keeping the redundant attributes up to date. GeneXus won’t do it.

Formulas redundancy



Flight		Flight	
FlightId		Id	
FlightPrice		Price	
AirlineId		Id	
AirlineName	☐	Name	
AirlineDiscountPercentage	☐	Percentage	
FlightFinalPrice	✓	Price	FlightPrice*(1 - AirlineDiscountPercentage/100)
FlightCapacity	✓	Numeric(4.0)	count(FlightSeatLocation)
Seat		Seat	
FlightSeatId		Id	
FlightSeatChar		SeatChar	
FlightSeatLocation		Location	



Airline
- AirlineId
- AirlineName
- AirlineDiscountPercentage

Flight Structure
- FlightId
- FlightPrice
- AirlineId
- FlightFinalPrice
- FlightCapacity

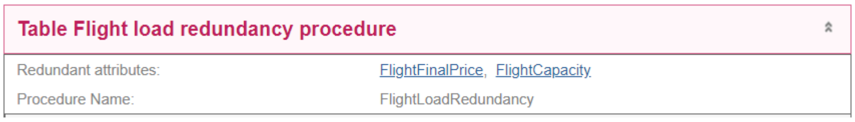


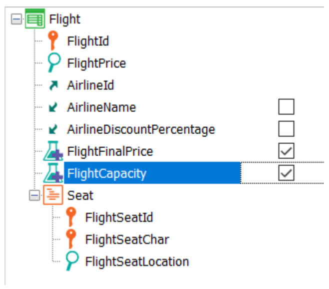
Table Flight load redundancy procedure

Redundant attributes:	FlightFinalPrice, FlightCapacity
Procedure Name:	FlightLoadRedundancy

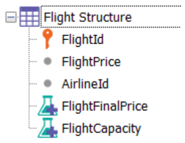
The other case of redundancies that we had already mentioned was that of formulas. In this example, we see that in addition to offering redundancy of the inferred attributes, it also offers redundancy of the two formulas defined for the transaction: the horizontal one that applies a discount to the price of the flight according to the discount percentage set by the airline, and the aggregate one that counts the number of seats on the flight.

Defining these formulas as redundant will add the attributes to the table. In addition, obviously, in the reorganization GeneXus must also create a procedure to load them with the corresponding values.

Formulas redundancy



- Flight
  - FlightId
  - FlightPrice
  - AirlineId
  - AirlineName ☐
  - AirlineDiscountPercentage ☐
  - FlightFinalPrice ✓
  - FlightCapacity ✓
  - Seat
    - FlightSeatId
    - FlightSeatChar
    - FlightSeatLocation



- Flight Structure
  - FlightId
  - FlightPrice
  - AirlineId
  - FlightFinalPrice
  - FlightCapacity

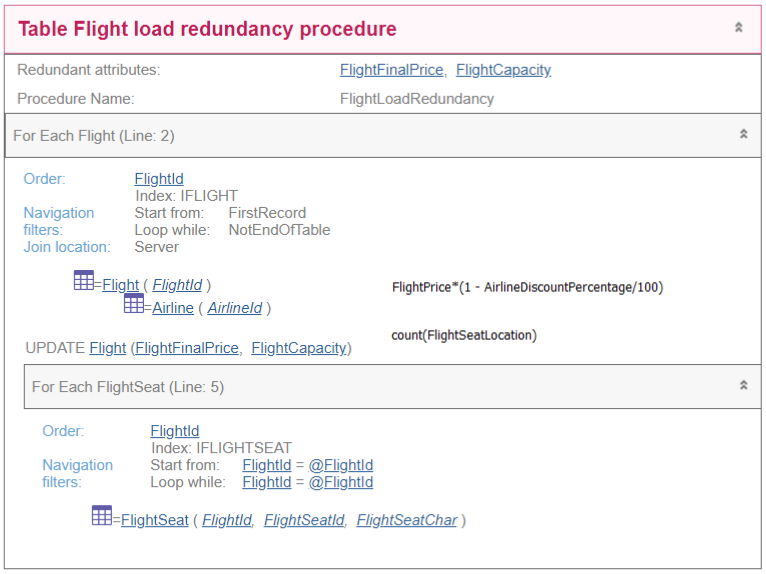


Table Flight load redundancy procedure

Redundant attributes: FlightFinalPrice, FlightCapacity
Procedure Name: FlightLoadRedundancy

For Each Flight (Line: 2)

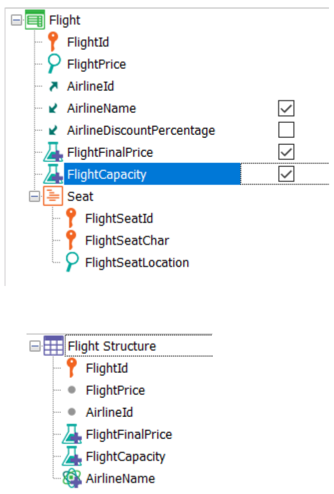
```
Order:        FlightId
              Index: IFLIGHT
Navigation    Start from:   FirstRecord
filters:      Loop while:   NotEndOfTable
Join location: Server

    =Flight ( FlightId )                FlightPrice*(1 - AirlineDiscountPercentage/100)
        =Airline ( AirlineId )
                                        count(FlightSeatLocation)
UPDATE Flight (FlightFinalPrice, FlightCapacity)
```

For Each FlightSeat (Line: 5)

```
Order:        FlightId
              Index: IFLIGHTSEAT
Navigation    Start from:   FlightId = @FlightId
filters:      Loop while:   FlightId = @FlightId

    =FlightSeat ( FlightId, FlightSeatId, FlightSeatChar )
```

That is, a procedure where for each record in the Flight table it will trigger the calculation of each formula... for FlightFinalPrice it will have to go to Airline to find the value of AirlineDiscountPercentage, and store its value in the redundant attribute; for FlightCapacity it will have to count the associated records in the FlightSeat table, and store the result in the redundant attribute.

Formulas redundancy



- Flight
  - FlightId
  - FlightPrice
  - AirlineId
  - AirlineName ✓
  - AirlineDiscountPercentage ☐
  - FlightFinalPrice ✓
  - FlightCapacity ✓
  - Seat
    - FlightSeatId
    - FlightSeatChar
    - FlightSeatLocation

- Flight Structure
  - FlightId
  - FlightPrice
  - AirlineId
  - FlightFinalPrice
  - FlightCapacity
  - AirlineName

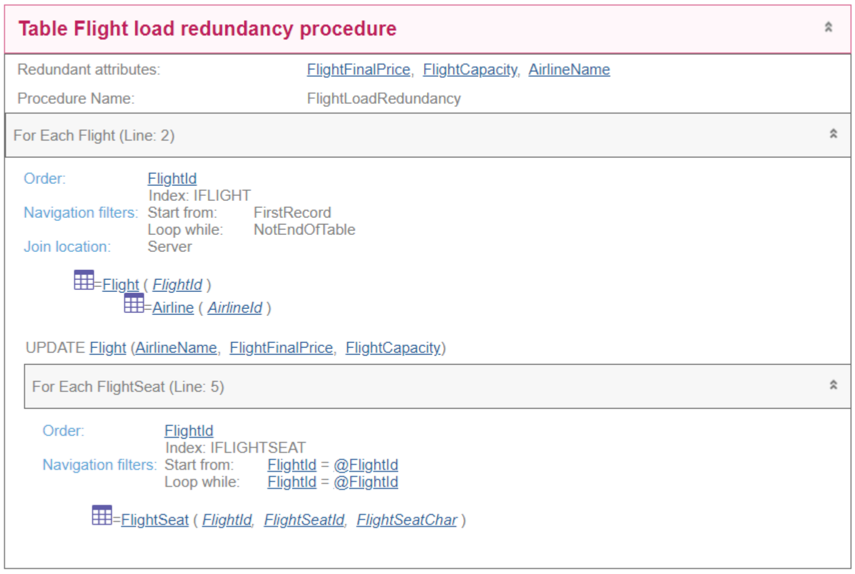


Table Flight load redundancy procedure

Redundant attributes:	FlightFinalPrice, FlightCapacity, AirlineName
Procedure Name:	FlightLoadRedundancy

For Each Flight (Line: 2)

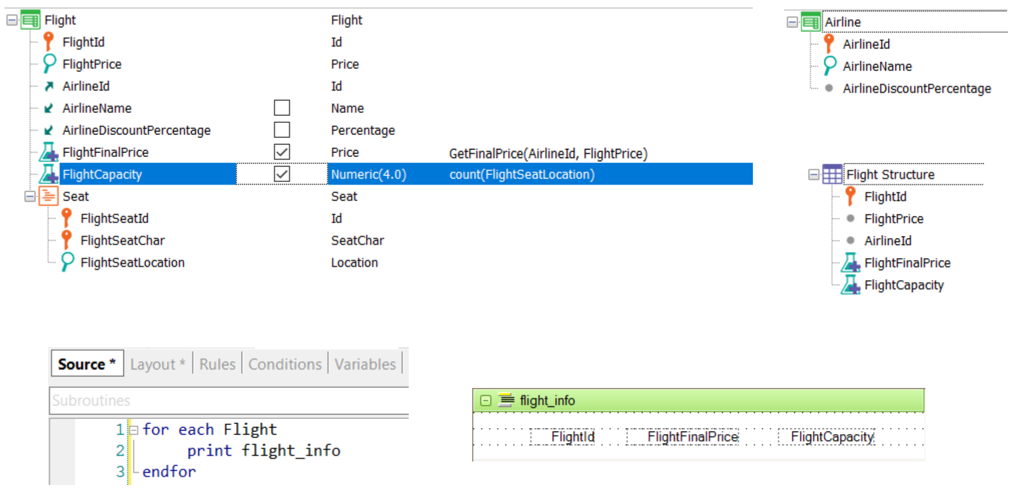
```
Order:               FlightId
                     Index: IFLIGHT
Navigation filters:  Start from:    FirstRecord
                     Loop while:    NotEndOfTable
Join location:       Server

        =Flight ( FlightId )
            =Airline ( AirlineId )

UPDATE Flight (AirlineName, FlightFinalPrice, FlightCapacity)
```

For Each FlightSeat (Line: 5)

```
Order:               FlightId
                     Index: IFLIGHTSEAT
Navigation filters:  Start from:    FlightId = @FlightId
                     Loop while:    FlightId = @FlightId

        =FlightSeat ( FlightId, FlightSeatId, FlightSeatChar )
```

If we also define, for example, AirlineName as redundant–that is, a referential redundancy–its value would be loaded into the table in this same procedure. That's why the name of the procedure is always the concatenation of the name of the table where the redundancies are, in this case Flight, and LoadRedundancy. That is to say, in this procedure all the redundancies of the table will be loaded: the referential ones and those of formulas.
And it will be the procedure automatically invoked in the reorganization, after the new attributes are added to the database table.

Formulas redundancy



Flight		Flight	
FlightId		Id	
FlightPrice		Price	
AirlineId		Id	
AirlineName	☐	Name	
AirlineDiscountPercentage	☐	Percentage	
FlightFinalPrice	✓	Price	GetFinalPrice(AirlineId, FlightPrice)
FlightCapacity	✓	Numeric(4.0)	count(FlightSeatLocation)
Seat		Seat	
FlightSeatId		Id	
FlightSeatChar		SeatChar	
FlightSeatLocation		Location	

Airline
- AirlineId
- AirlineName
- AirlineDiscountPercentage

Flight Structure
- FlightId
- FlightPrice
- AirlineId
- FlightFinalPrice
- FlightCapacity

Source * | Layout * | Rules | Conditions | Variables

Subroutines

```
for each Flight
      print flight_info
endfor
```

flight_info

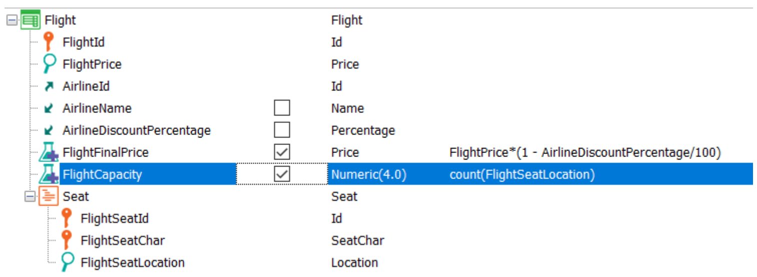
FlightId FlightFinalPrice FlightCapacity

Let's focus only on the formulas again. Of course, once they have been made redundant, when information on flight price or capacity is needed, the formula will not be triggered again, but the stored value will be retrieved, and that is exactly the point.

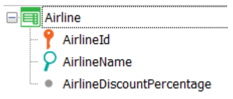
Hypothetically, if we had to run this list millions of times, and if there were thousands of seats for each flight, calculating the FlightCapacity formula every time could become a performance problem. Having it redundant saves us from having to make the calculation for each query. There is only the cost of the calculation to load the redundant attribute the first time, and to keep it up to date afterwards.

For the horizontal formula of the example, the usefulness of defining it as redundant doesn't seem too clear, at least in terms of performance, unless we need, for example, a panel or web panel where the user wants to filter the flights shown in a grid by flight price.
The case would become more interesting if the horizontal calculation involved many tables of the extended one. Or, even more so, if it were a horizontal formula of the kind that is solved by invoking a procedure that does perform a complex calculation.

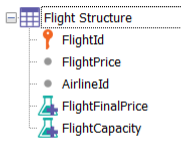
Formulas redundancy



Flight		Flight	
FlightId		Id	
FlightPrice		Price	
AirlineId		Id	
AirlineName	☐	Name	
AirlineDiscountPercentage	☐	Percentage	
FlightFinalPrice	☑	Price	FlightPrice*(1 - AirlineDiscountPercentage/100)
FlightCapacity	☑	Numeric(4.0)	count(FlightSeatLocation)
Seat		Seat	
FlightSeatId		Id	
FlightSeatChar		SeatChar	
FlightSeatLocation		Location	



```
Airline
  AirlineId
  AirlineName
  AirlineDiscountPercentage
```



```
Flight Structure
  FlightId
  FlightPrice
  AirlineId
  FlightFinalPrice
  FlightCapacity
```

```
For each Flight
      FlightPrice *= 1.1
endfor
```

```
For each Flight.Seat
      where FlighId = 4
      Delete
endfor
```

```
new
      FlightId = 12503
      FlightPrice = 500
      AirlineId = find(AirlineId, AirlineName = "Air Europe")
      new
            FlighSeatId = 1
            FlightChar = 'B'
            FlightSeatLocation = Location.Window
      endnew
endnew
```

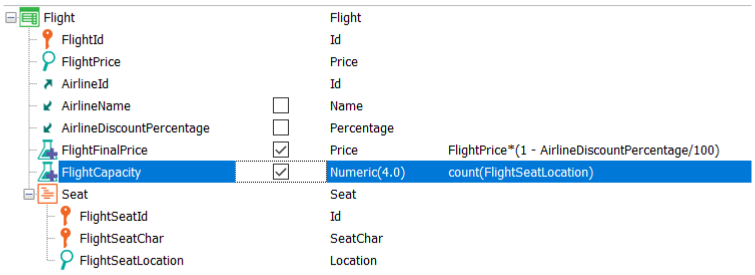
If there is a clear advantage to redundant formula attributes, what is the disadvantage?

That they should always be kept up to date, and that update has a cost. If the value of FlightPrice is changed through the Flight transaction (or its business component) the horizontal formula involving it will be triggered again as usual. The same happens if a line is added or deleted: the FlightCapacity formula is triggered again. In both cases, its value is stored in the redundant attributes and the developer doesn't have to worry about doing it.

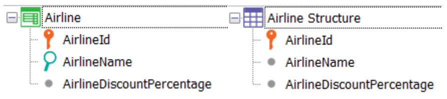
However, as in the case of referential redundancies, if the modification is done with a procedure, as in these examples, GeneXus will do nothing. Here the developer will have to worry about updating the redundant attributes by adding code.

GeneXus

Formulas redundancy

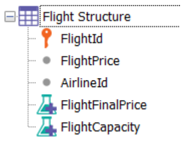


Flight		Flight	
FlightId		Id	
FlightPrice		Price	
AirlineId		Id	
AirlineName	☐	Name	
AirlineDiscountPercentage	☐	Percentage	
FlightFinalPrice	✓	Price	FlightPrice*(1 - AirlineDiscountPercentage/100)
FlightCapacity	✓	Numeric(4.0)	count(FlightSeatLocation)
Seat		Seat	
FlightSeatId		Id	
FlightSeatChar		SeatChar	
FlightSeatLocation		Location	



Airline
- AirlineId
- AirlineName
- AirlineDiscountPercentage

Airline Structure
- AirlineId
- AirlineName
- AirlineDiscountPercentage



Flight Structure
- FlightId
- FlightPrice
- AirlineId
- FlightFinalPrice
- FlightCapacity

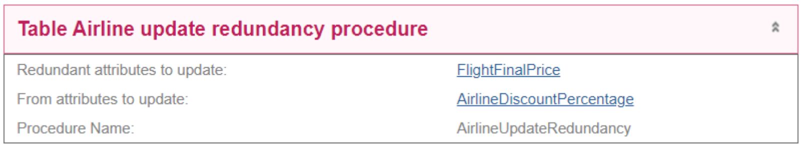


Table Airline update redundancy procedure

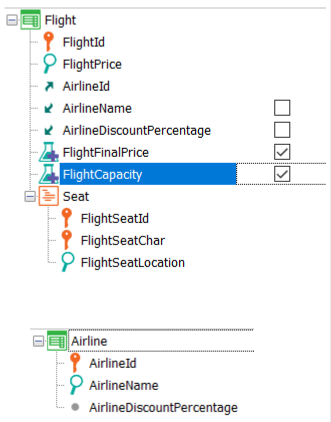
Redundant attributes to update:	FlightFinalPrice
From attributes to update:	AirlineDiscountPercentage
Procedure Name:	AirlineUpdateRedundancy

What happens if the user enters the Airline transaction and modifies the discount percentage for an airline? (or does it through the business component).

In Flight the redundant formula FlightFinalPrice depends on that value, so the redundant value stored for all Flight records belonging to that airline should be updated.

GeneXus will create in the reorganization the procedure whose name is the concatenation of the table name, in this case Airline, and UpdateRedundancy, as it did with referential redundancies.

Formulas redundancy



```
Flight
    FlightId
    FlightPrice
    AirlineId
    AirlineName                    ☐
    AirlineDiscountPercentage      ☐
    FlightFinalPrice               ☑
    FlightCapacity                 ☑
  Seat
      FlightSeatId
      FlightSeatChar
      FlightSeatLocation

Airline
    AirlineId
    AirlineName
    AirlineDiscountPercentage
```

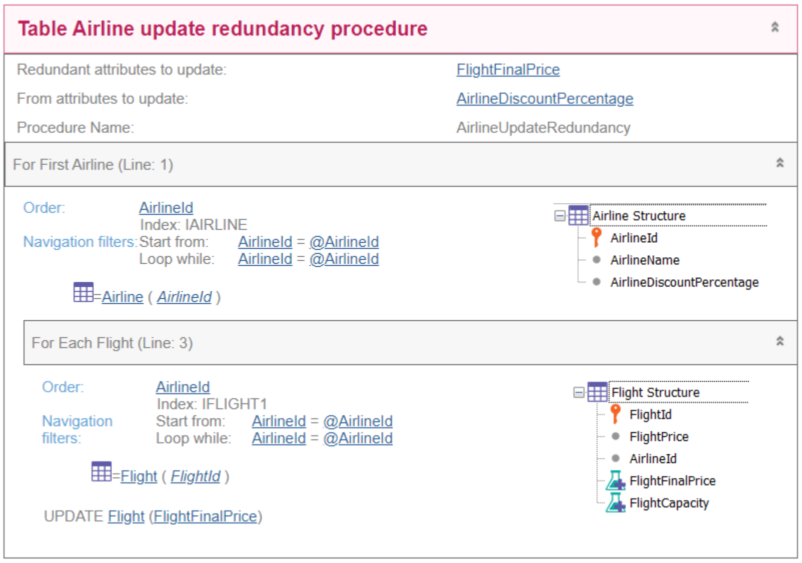


Table Airline update redundancy procedure

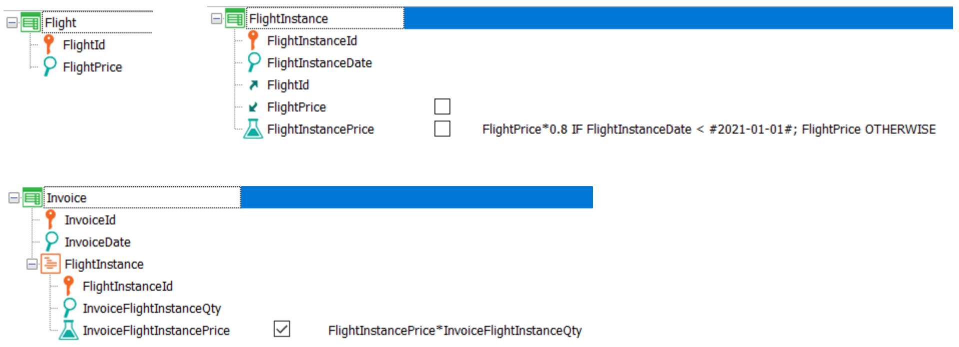
Redundant attributes to update:	FlightFinalPrice
From attributes to update:	AirlineDiscountPercentage
Procedure Name:	AirlineUpdateRedundancy

```
For First Airline (Line: 1)

Order:              AirlineId
                    Index: IAIRLINE
Navigation filters: Start from:   AirlineId = @AirlineId
                    Loop while:   AirlineId = @AirlineId

      =Airline ( AirlineId )

Airline Structure
    AirlineId
    AirlineName
    AirlineDiscountPercentage

For Each Flight (Line: 3)

Order:              AirlineId
                    Index: IFLIGHT1
Navigation          Start from:   AirlineId = @AirlineId
filters:            Loop while:   AirlineId = @AirlineId

      =Flight ( FlightId )

UPDATE Flight (FlightFinalPrice)

Flight Structure
    FlightId
    FlightPrice
    AirlineId
    FlightFinalPrice
    FlightCapacity
```

This procedure will be invoked from the Airline transaction by passing it the ID and the procedure will access the corresponding record. Next, it will run through the Flight table filtering by that airline and trigger the FlightFinalPrice formula again, storing its result in the table.

All this implies a performance cost. Therefore, a very careful assessment should be made of when it is convenient to make a formula redundant and when it is not.

Formulas redundancy: limitations



```
Flight
  FlightId
  FlightPrice

FlightInstance
  FlightInstanceId
  FlightInstanceDate
  FlightId
  FlightPrice               ☐
  FlightInstancePrice       ☐   FlightPrice*0.8 IF FlightInstanceDate < #2021-01-01#; FlightPrice OTHERWISE

Invoice
  InvoiceId
  InvoiceDate
  FlightInstance
    FlightInstanceId
    InvoiceFlightInstanceQty
    InvoiceFlightInstancePrice   ☑   FlightInstancePrice*InvoiceFlightInstanceQty
```

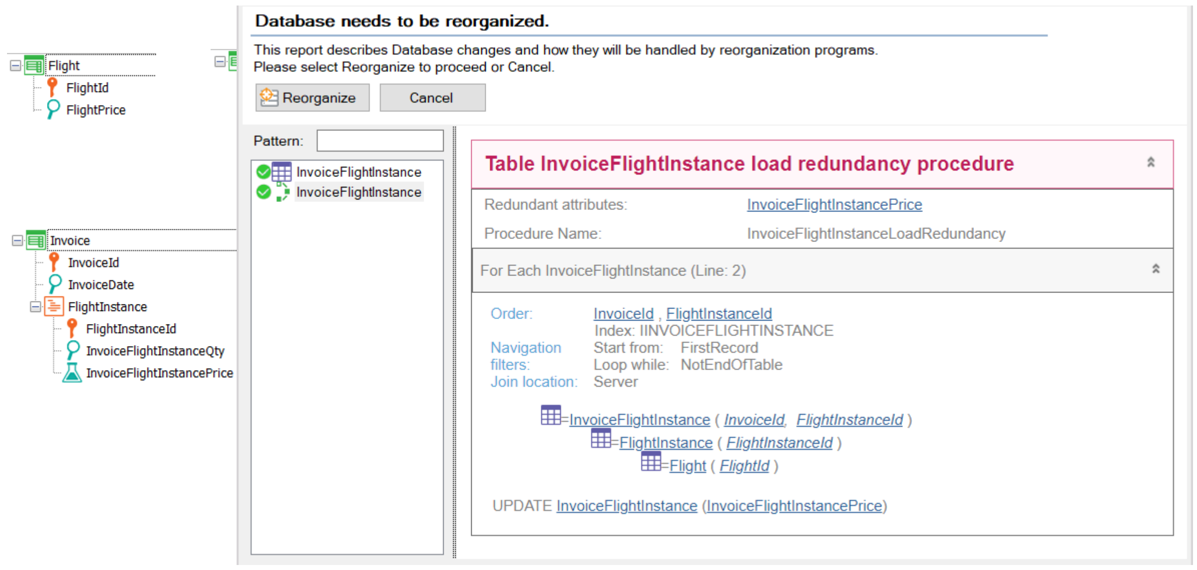
For the moment, the maintenance of redundant formulas has some limitations.

For example, when we have these three related transactions, where the Flight transaction records the generic information of a flight, such as its price, but the FlightInstance transaction is the one corresponding to the actual flight on a given date. In it, the price of the actual flight is obtained with a formula that takes into account the list price of the flight according to the date.

And then we have a transaction to record flight invoicing. The lines record the flights for which tickets are being purchased, and how many tickets for each one. Then this formula calculates the price for each line, using the inferred formula attribute of FlightInstance, which in turn is a formula, as we said.

Suppose we want to make this formula redundant in the InvoiceFlightInstance table. If we only make it redundant...

Formulas redundancy: limitations



Database needs to be reorganized.

This report describes Database changes and how they will be handled by reorganization programs.
Please select Reorganize to proceed or Cancel.

Table InvoiceFlightInstance load redundancy procedure

Redundant attributes: InvoiceFlightInstancePrice
Procedure Name: InvoiceFlightInstanceLoadRedundancy

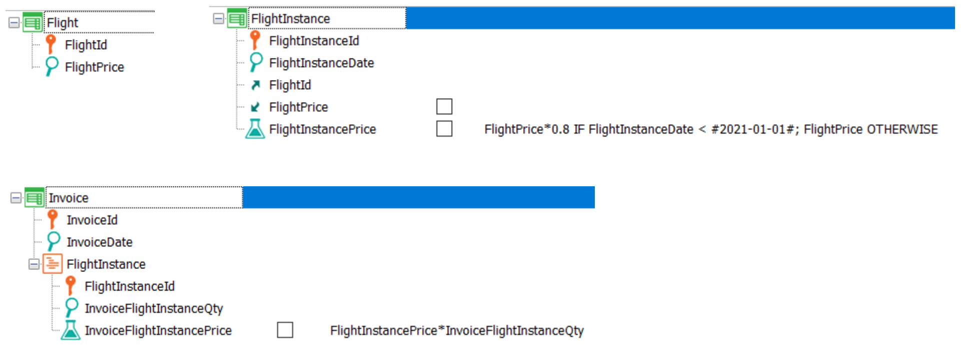
For Each InvoiceFlightInstance (Line: 2)

```
Order:              InvoiceId , FlightInstanceId
                    Index: IINVOICEFLIGHTINSTANCE
Navigation          Start from:   FirstRecord
filters:            Loop while:   NotEndOfTable
Join location:      Server

        =InvoiceFlightInstance ( InvoiceId,  FlightInstanceId )
              =FlightInstance ( FlightInstanceId )
                    =Flight ( FlightId )

UPDATE InvoiceFlightInstance (InvoiceFlightInstancePrice)
```

...the procedure for loading the redundancy will be handled correctly, by going to the InvoiceFlightInstance table and triggering the horizontal formula for each record, and storing it.

However, redundancy update procedures will not be created.

Formulas redundancy: limitations



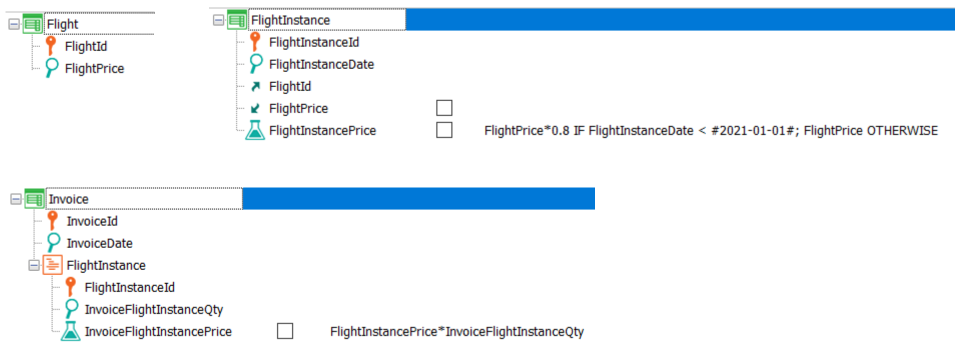
```
Flight
  FlightId
  FlightPrice

FlightInstance
  FlightInstanceId
  FlightInstanceDate
  FlightId
  FlightPrice          ☐
  FlightInstancePrice  ☐   FlightPrice*0.8 IF FlightInstanceDate < #2021-01-01#; FlightPrice OTHERWISE

Invoice
  InvoiceId
  InvoiceDate
  FlightInstance
    FlightInstanceId
    InvoiceFlightInstanceQty
    InvoiceFlightInstancePrice  ☐   FlightInstancePrice*InvoiceFlightInstanceQty
```

What do we mean?

That if the user executes the FlightInstance transaction and changes, for example, the value of FlightInstanceDate (suppose that from a lower date than this to a later one), since the horizontal formula will be triggered again and its value will be changed, we would expect the transaction to invoke a procedure–FlightInstanceUpdateRedundancy–that will go to the FlightInstance table and update the corresponding redundancies. But it won't.

Similarly, we would expect that if the user changes the value of the FlightPrice attribute through the Flight transaction there would also be a FlightUpdateRedundancy procedure that goes to the InvoiceFlightInstance table to recalculate and re-store the redundancies of the formula attribute that should be changed when the FlightInstancePrice is changed, when applicable. It won’t do it either.

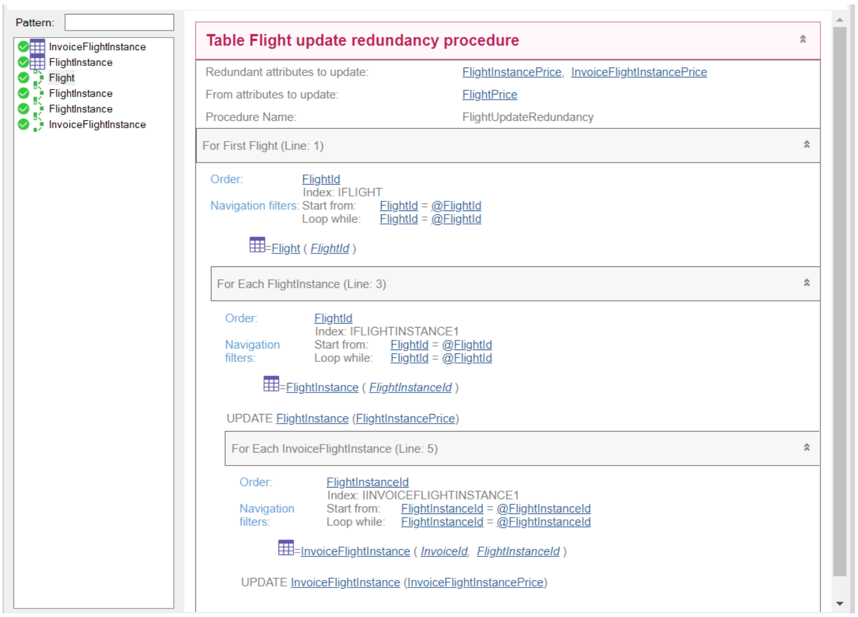
Formulas redundancy: limitations


Flight
FlightId
FlightPrice

FlightInstance
FlightInstanceId
FlightInstanceDate
FlightId
FlightPrice ☐
FlightInstancePrice ☐ FlightPrice*0.8 IF FlightInstanceDate < #2021-01-01#; FlightPrice OTHERWISE

Invoice
InvoiceId
InvoiceDate
FlightInstance
FlightInstanceId
InvoiceFlightInstanceQty
InvoiceFlightInstancePrice ☐ FlightInstancePrice*InvoiceFlightInstanceQty

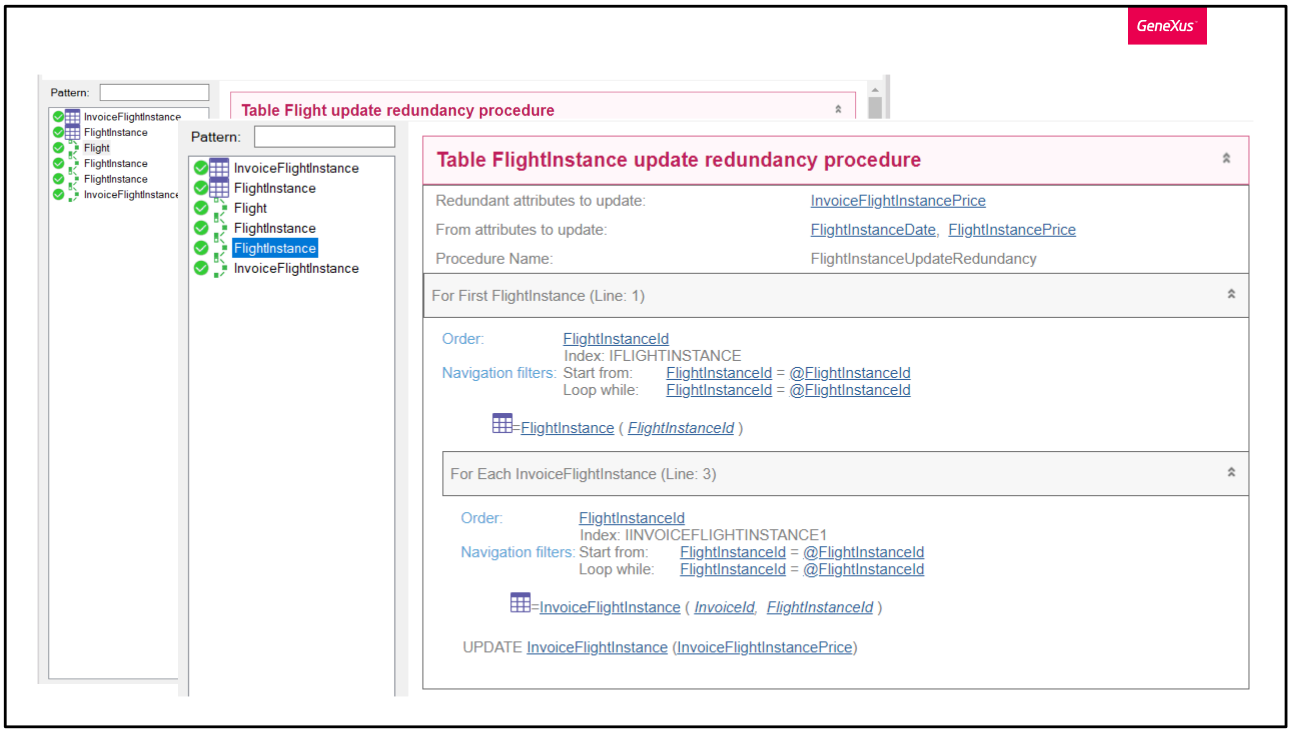
But what if we also make the involved horizontal formula redundant?



```
Pattern:
InvoiceFlightInstance
FlightInstance
Flight
FlightInstance
FlightInstance
InvoiceFlightInstance

Table Flight update redundancy procedure
Redundant attributes to update:   FlightInstancePrice, InvoiceFlightInstancePrice
From attributes to update:        FlightPrice
Procedure Name:                   FlightUpdateRedundancy

For First Flight (Line: 1)
  Order:              FlightId
                      Index: IFLIGHT
  Navigation filters: Start from:   FlightId = @FlightId
                      Loop while:   FlightId = @FlightId

      =Flight ( FlightId )

  For Each FlightInstance (Line: 3)
    Order:        FlightId
                  Index: IFLIGHTINSTANCE1
    Navigation    Start from:   FlightId = @FlightId
    filters:      Loop while:   FlightId = @FlightId

        =FlightInstance ( FlightInstanceId )

    UPDATE FlightInstance (FlightInstancePrice)

    For Each InvoiceFlightInstance (Line: 5)
      Order:        FlightInstanceId
                    Index: IINVOICEFLIGHTINSTANCE1
      Navigation    Start from:   FlightInstanceId = @FlightInstanceId
      filters:      Loop while:   FlightInstanceId = @FlightInstanceId

          =InvoiceFlightInstance ( InvoiceId, FlightInstanceId )

      UPDATE InvoiceFlightInstance (InvoiceFlightInstancePrice)
```

We will see when reorganizing that in addition to reporting the changes in both tables to add the two formulas as redundant, and the two procedures for loading the redundancies, two UpdateRedundancy procedures will be created.

The first one that is triggered from the Flight transaction when the price is changed and will update the first redundant formula in FlightInstance. Then, from it, the second one in InvoiceFlightInstance.



Pattern:

- InvoiceFlightInstance
- FlightInstance
- Flight
- FlightInstance
- FlightInstance
- InvoiceFlightInstance

Table Flight update redundancy procedure

Table FlightInstance update redundancy procedure

Redundant attributes to update:	InvoiceFlightInstancePrice
From attributes to update:	FlightInstanceDate, FlightInstancePrice
Procedure Name:	FlightInstanceUpdateRedundancy

For First FlightInstance (Line: 1)

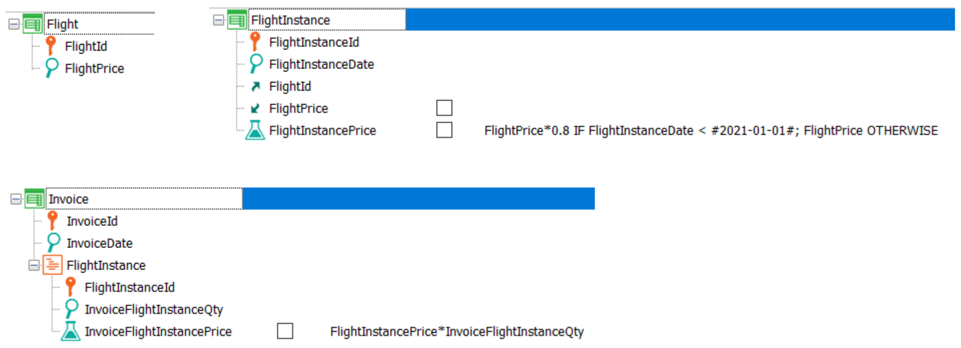
```
Order:              FlightInstanceId
                    Index: IFLIGHTINSTANCE
Navigation filters: Start from:   FlightInstanceId = @FlightInstanceId
                    Loop while:   FlightInstanceId = @FlightInstanceId

        =FlightInstance ( FlightInstanceId )
```

For Each InvoiceFlightInstance (Line: 3)

```
Order:              FlightInstanceId
                    Index: IINVOICEFLIGHTINSTANCE1
Navigation filters: Start from:   FlightInstanceId = @FlightInstanceId
                    Loop while:   FlightInstanceId = @FlightInstanceId

        =InvoiceFlightInstance ( InvoiceId, FlightInstanceId )

UPDATE InvoiceFlightInstance (InvoiceFlightInstancePrice)
```

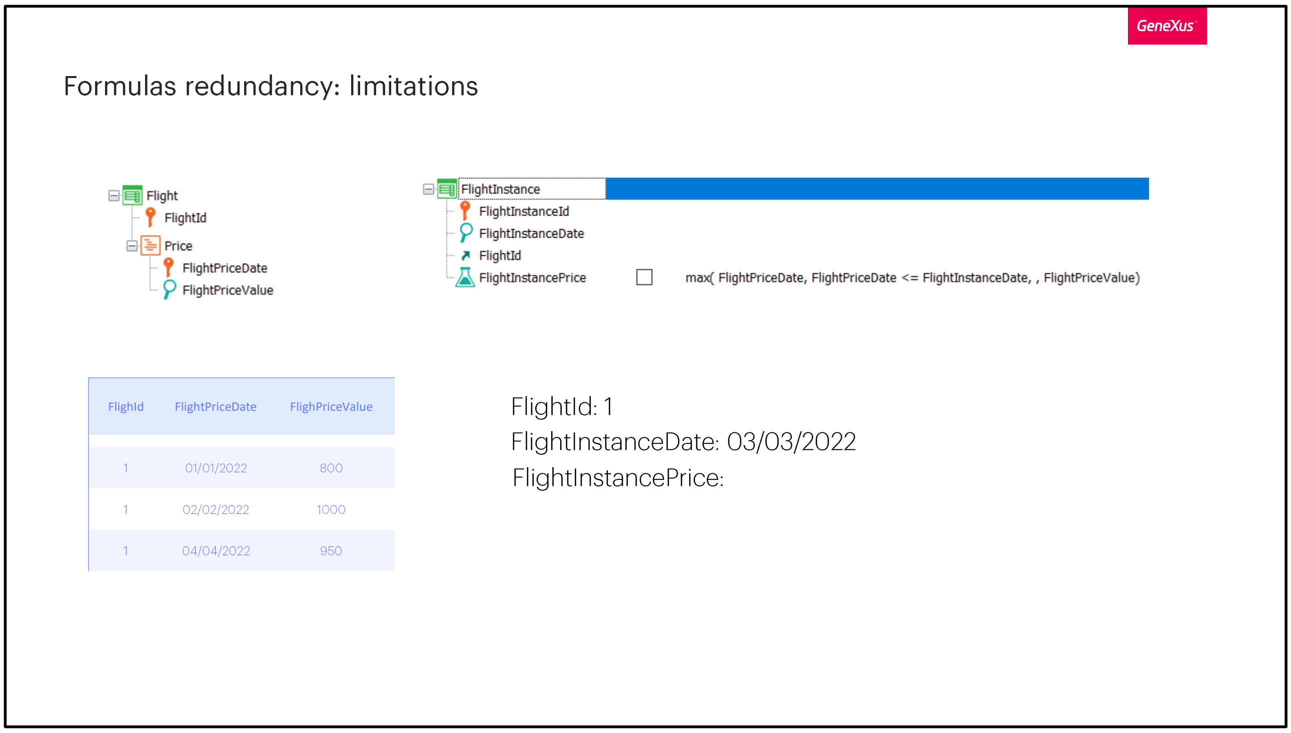
The second procedure will be executed when the flight date is changed from the FlightInstance transaction, and will update the redundant attribute in InvoiceFlightInstance.

Formulas redundancy: limitations

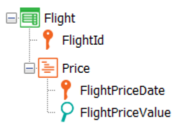


```
Flight
    FlightId
    FlightPrice

FlightInstance
    FlightInstanceId
    FlightInstanceDate
    FlightId
    FlightPrice              ☐
    FlightInstancePrice      ☐    FlightPrice*0.8 IF FlightInstanceDate < #2021-01-01#; FlightPrice OTHERWISE

Invoice
    InvoiceId
    InvoiceDate
    FlightInstance
        FlightInstanceId
        InvoiceFlightInstanceQty
        InvoiceFlightInstancePrice   ☐    FlightInstancePrice*InvoiceFlightInstanceQty
```

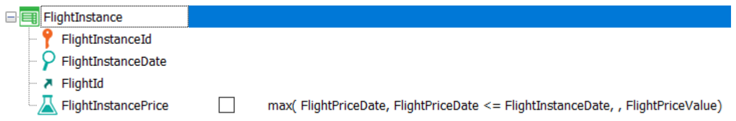
So, if we have a formula whose calculation involves another one, in order to maintain its redundancy automatically, we need to define the inner formula as redundant as well.



Formulas redundancy: limitations



```
Flight
  FlightId
  Price
    FlightPriceDate
    FlightPriceValue
```



```
FlightInstance
  FlightInstanceId
  FlightInstanceDate
  FlightId
  FlightInstancePrice    ☐    max( FlightPriceDate, FlightPriceDate <= FlightInstanceDate, , FlightPriceValue)
```

FlighId	FlightPriceDate	FlighPriceValue
1	01/01/2022	800
1	02/02/2022	1000
1	04/04/2022	950

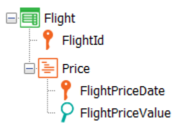
FlightId: 1
FlightInstanceDate: 03/03/2022
FlightInstancePrice:

But there are more limitations. For example, in most cases aggregate/select formulas cannot be maintained through these update procedures.

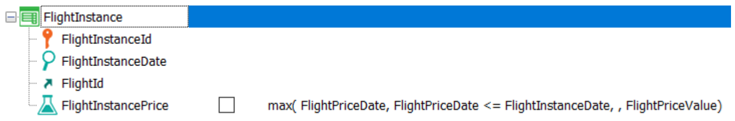
If we maintain in Flight a list of flight prices by date, for example, in FlightInstance we will have to calculate the price of a particular flight according to the price that corresponds to the actual flight date. That is, we calculate the price according to a max formula.
So, with this data, if we are entering an instance of flight 1, with this date, the max formula will be left with this record...

GeneXus

Formulas redundancy: limitations



Flight
- FlightId
- Price
  - FlightPriceDate
  - FlightPriceValue



FlightInstance
- FlightInstanceId
- FlightInstanceDate
- FlightId
- FlightInstancePrice ☐ max(FlightPriceDate, FlightPriceDate <= FlightInstanceDate, , FlightPriceValue)

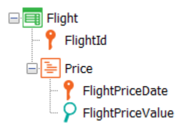
FlighId	FlightPriceDate	FlighPriceValue
1	01/01/2022	800
1	02/02/2022	1000
1	04/04/2022	950

FlightId: 1
FlightInstanceDate: 03/03/2022 LIST
FlightInstancePrice:

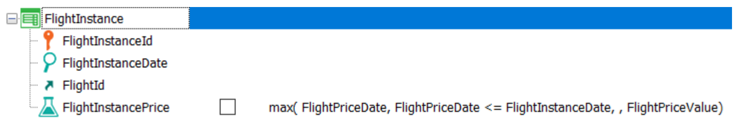
... so it will return the value 1000 for the flight price. If it were being printed in a list, 1000 would be displayed.

GeneXus

Formulas redundancy: limitations



```
Flight
    FlightId
    Price
        FlightPriceDate
        FlightPriceValue
```



```
FlightInstance
    FlightInstanceId
    FlightInstanceDate
    FlightId
    FlightInstancePrice    ☐    max( FlightPriceDate, FlightPriceDate <= FlightInstanceDate, , FlightPriceValue)
```

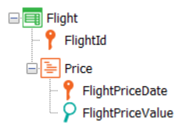
FlighId	FlightPriceDate	FlighPriceValue
1	01/01/2022	800
1	02/02/2022	1500
1	04/04/2022	950

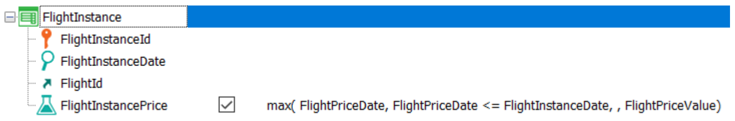
FlightId: 1
FlightInstanceDate: 03/03/2022 LIST
FlightInstancePrice: 1500

FlightPriceUpdateRedundancy?

If later on the user goes to the Flight transaction and changes 1000 to 1500 in the line corresponding to this record, since the formula is virtual, when the list is executed again, the max will be triggered and 1500 will be listed.

Formulas redundancy: limitations





```
max( FlightPriceDate, FlightPriceDate <= FlightInstanceDate, , FlightPriceValue)
```

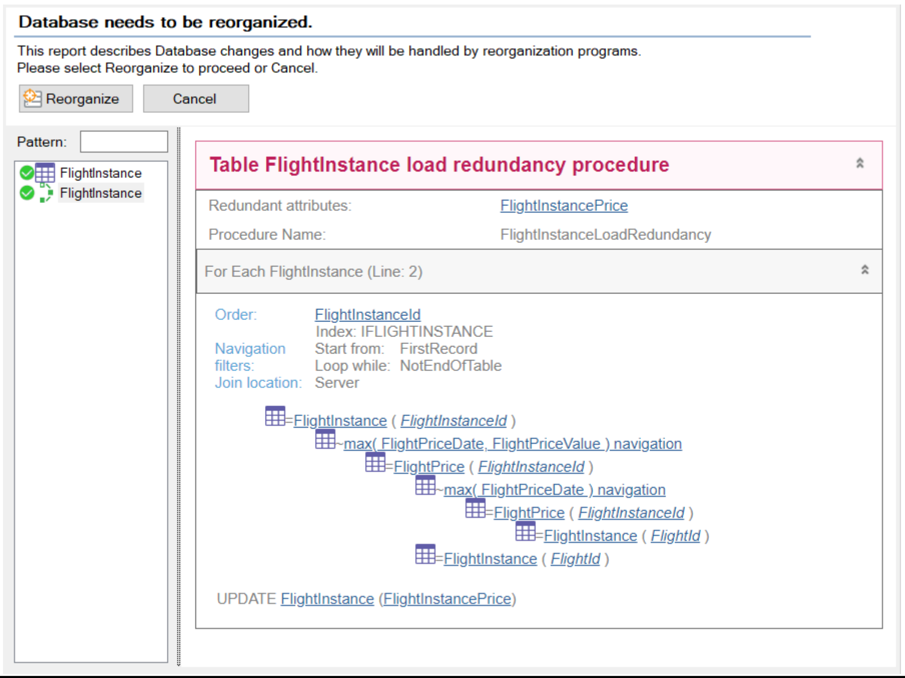
FlighId	FlightPriceDate	FlighPriceValue
1	01/01/2022	800
1	02/02/2022	1500
1	04/04/2022	950

FlightId: 1
FlightInstanceDate: 03/03/2022 LIST
FlightInstancePrice: 1500

FlightPriceUpdateRedundancy?

But what if we make the max formula redundant? We would expect that a redundancy update procedure would be created for the FlightPrice table, so when the value of FlightPriceValue is modified through the Flight transaction, the FlightInstance table would be accessed looking for which records would be affected by the change to modify the value of FlightInstancePrice. But when trying to figure out which records would be affected, we see how difficult it is to determine this.

Therefore, in this case, GeneXus will not create this redundancy maintenance program. We see this clearly when we define the attribute as redundant and look at the reorg report.



Database needs to be reorganized.

This report describes Database changes and how they will be handled by reorganization programs.
Please select Reorganize to proceed or Cancel.

Reorganize | Cancel

Pattern:

FlightInstance
FlightInstance

Table FlightInstance load redundancy procedure

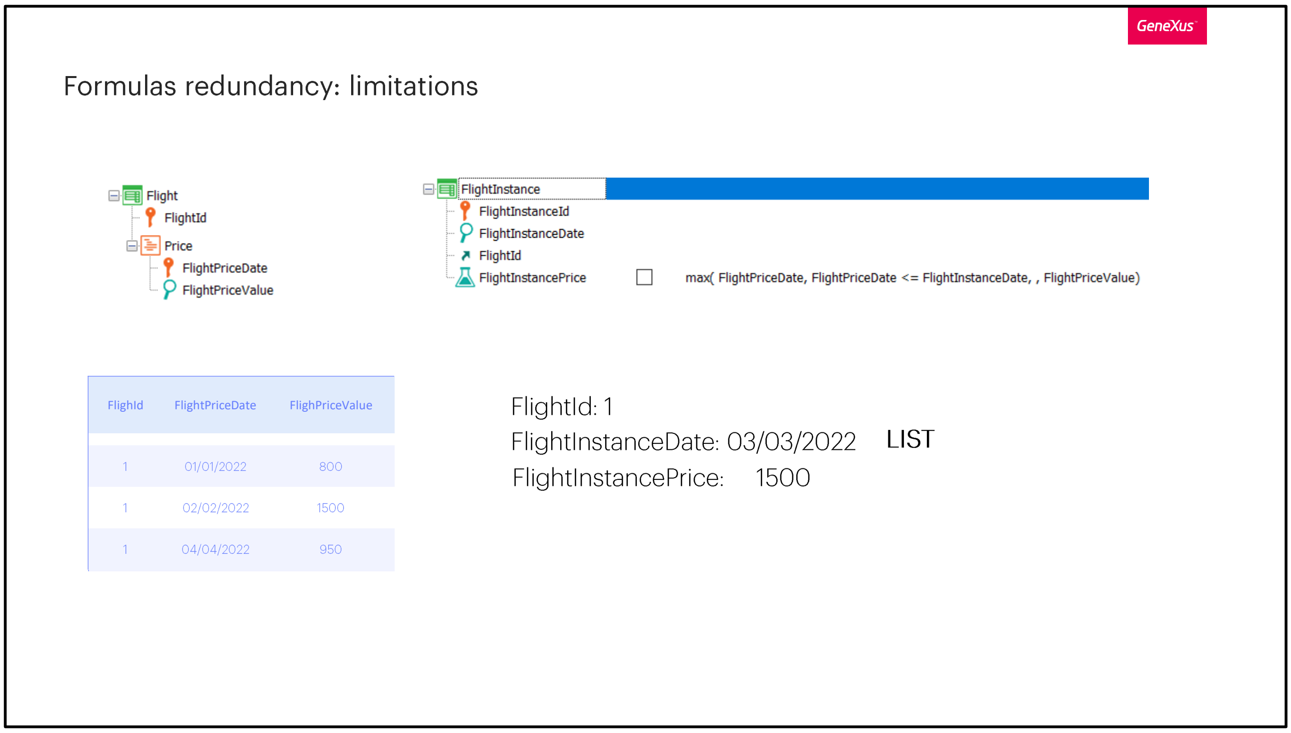
Redundant attributes:	FlightInstancePrice
Procedure Name:	FlightInstanceLoadRedundancy

For Each FlightInstance (Line: 2)

```
Order:            FlightInstanceId
                  Index: IFLIGHTINSTANCE
Navigation        Start from:   FirstRecord
filters:          Loop while:   NotEndOfTable
Join location:    Server

=FlightInstance ( FlightInstanceId )
    ~max( FlightPriceDate, FlightPriceValue ) navigation
        =FlightPrice ( FlightInstanceId )
            ~max( FlightPriceDate ) navigation
                =FlightPrice ( FlightInstanceId )
                    =FlightInstance ( FlightId )
            =FlightInstance ( FlightId )

UPDATE FlightInstance (FlightInstancePrice)
```

It will only create the redundancy load procedure, but not the update procedure.

So we recommend to always read this impact analysis report to find out which redundant formulas will be maintained and which ones will not.



Formulas redundancy: limitations

GeneXus

```
Flight
  FlightId
  Price
    FlightPriceDate
    FlightPriceValue
```

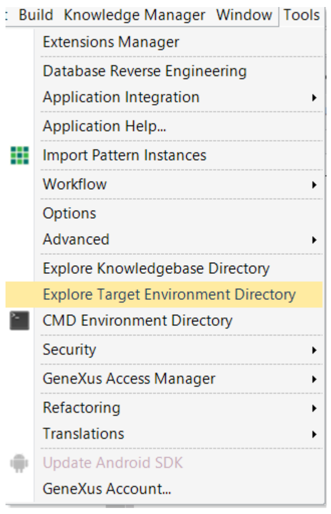
```
FlightInstance
  FlightInstanceId
  FlightInstanceDate
  FlightId
  FlightInstancePrice    ☐    max( FlightPriceDate, FlightPriceDate <= FlightInstanceDate, , FlightPriceValue)
```

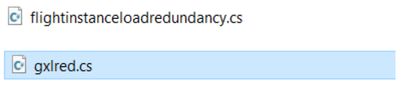
FlighId	FlightPriceDate	FlighPriceValue
1	01/01/2022	800
1	02/02/2022	1500
1	04/04/2022	950

FlightId: 1
FlightInstanceDate: 03/03/2022 LIST
FlightInstancePrice: 1500

As a side note: if the user accesses the FlightInstance transaction and changes the value of FlightInstanceDate, of course the redundant attribute will be kept up to date. The reason is that inside the transaction the formula will be triggered as usual and its value will be stored. The problem occurs when an attribute involved in the formula calculation is modified from another transaction, not the one of the formula. In this case, from the Flight transaction.

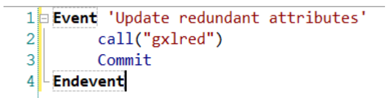
Rebuild redundancy





flightinstanceloadredundancy.cs

gxlred.cs



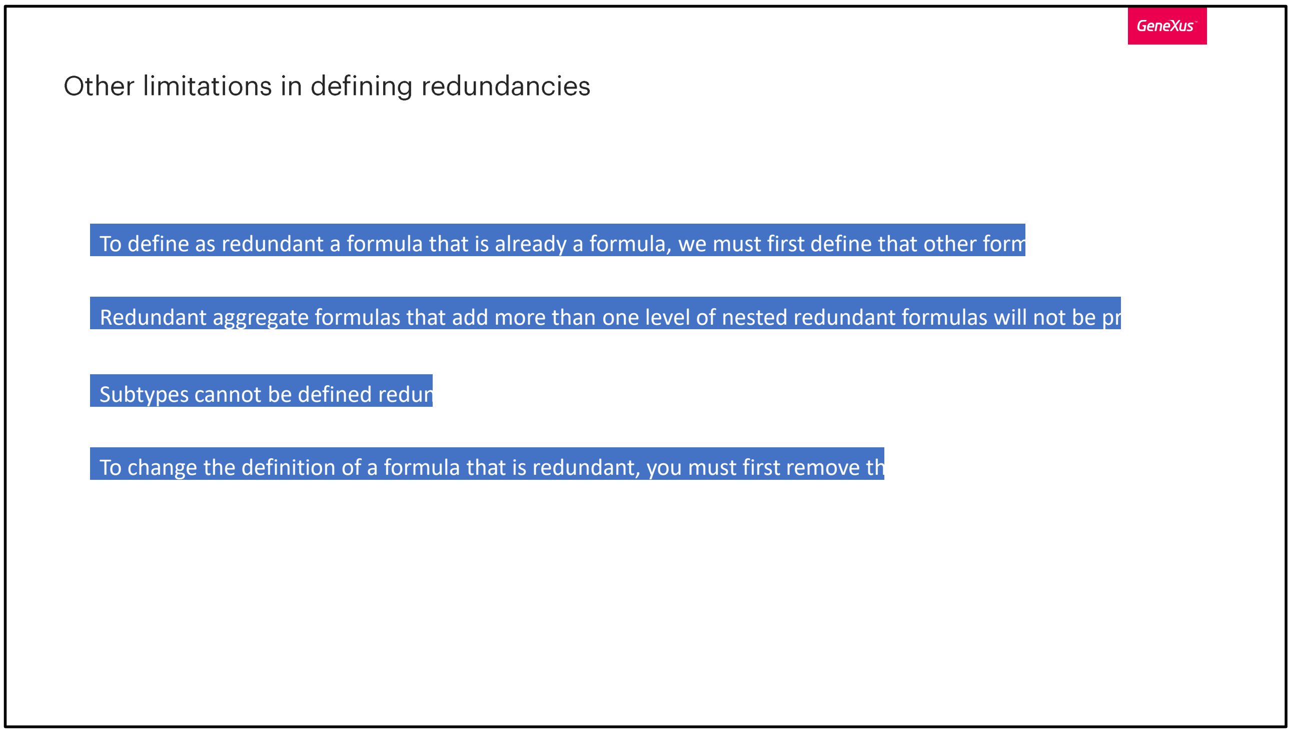
```
Event 'Update redundant attributes'
      call("gxlred")
      Commit
Endevent
```

If we needed that attribute to be redundant anyway, knowing that the redundancy will not be automatically kept up to date from Flight, we can always invoke the redundancy loading procedure created by GeneXus.

If we look for the files in the environment directory, we will find the redundancy loading procedure for each table with redundant attributes. The one that ends with LoadRedundancy; table name: load redundancy. In the last case, the flightinstanceloadredundancy, which we can invoke as a call to an external procedure.

In addition, even though it is not listed in the impact analysis report, GeneXus creates a procedure named gxlred, for GeneXus Load Redundancy, which invokes each of the LoadRedundancy procedures of each table with redundancies.

So if at any time we want to make sure that all redundancies defined in the KB are updated, we can, for example, invoke this program from an event.



GeneXus

Other limitations in defining redundancies

To define as redundant a formula that is already a formula, we must first define that other form

Redundant aggregate formulas that add more than one level of nested redundant formulas will not be pr

Subtypes cannot be defined redun

To change the definition of a formula that is redundant, you must first remove th

There are other limitations for defining redundant attributes that must be considered.

Some of them are listed here and you can read more about them in the wiki.

GeneXus™

training.genexus.com
wiki.genexus.com