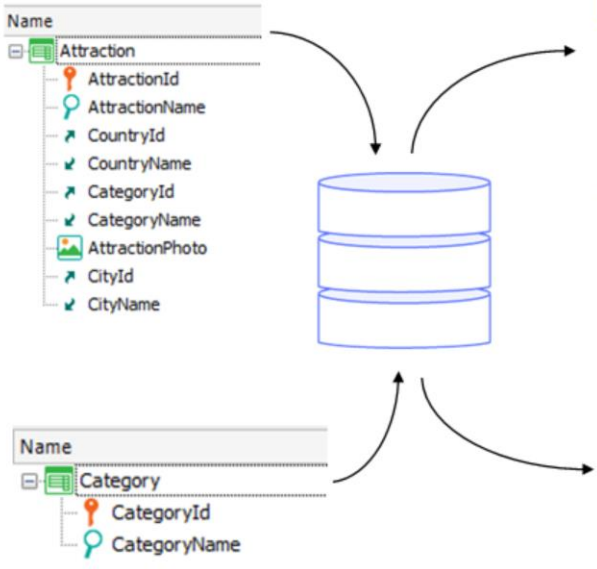
Populating with data from the transaction itself

GeneXus 16

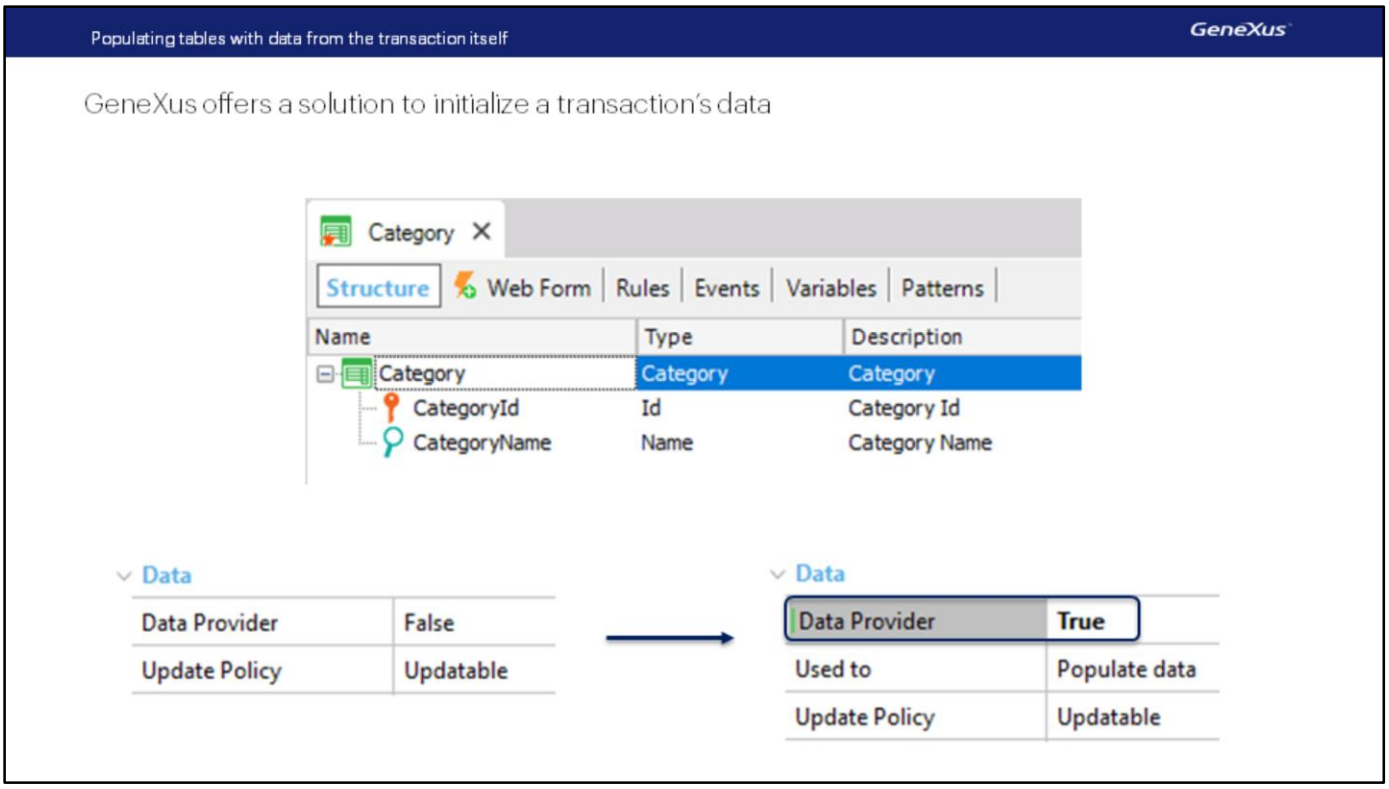
Transactions and tables



AttractionId	AttractionName	CountryId	CategoryId
1	Louvre Museum	2	1
2	The Great Wall	3	3
3	Eiffel Tower	2	2
...	...	...	...

CategoryId	CategoryName
1	Museum
2	Monument
3	Tourist Site

When we create a transaction, by default GeneXus will create associated tables to store the data entered from its screen.

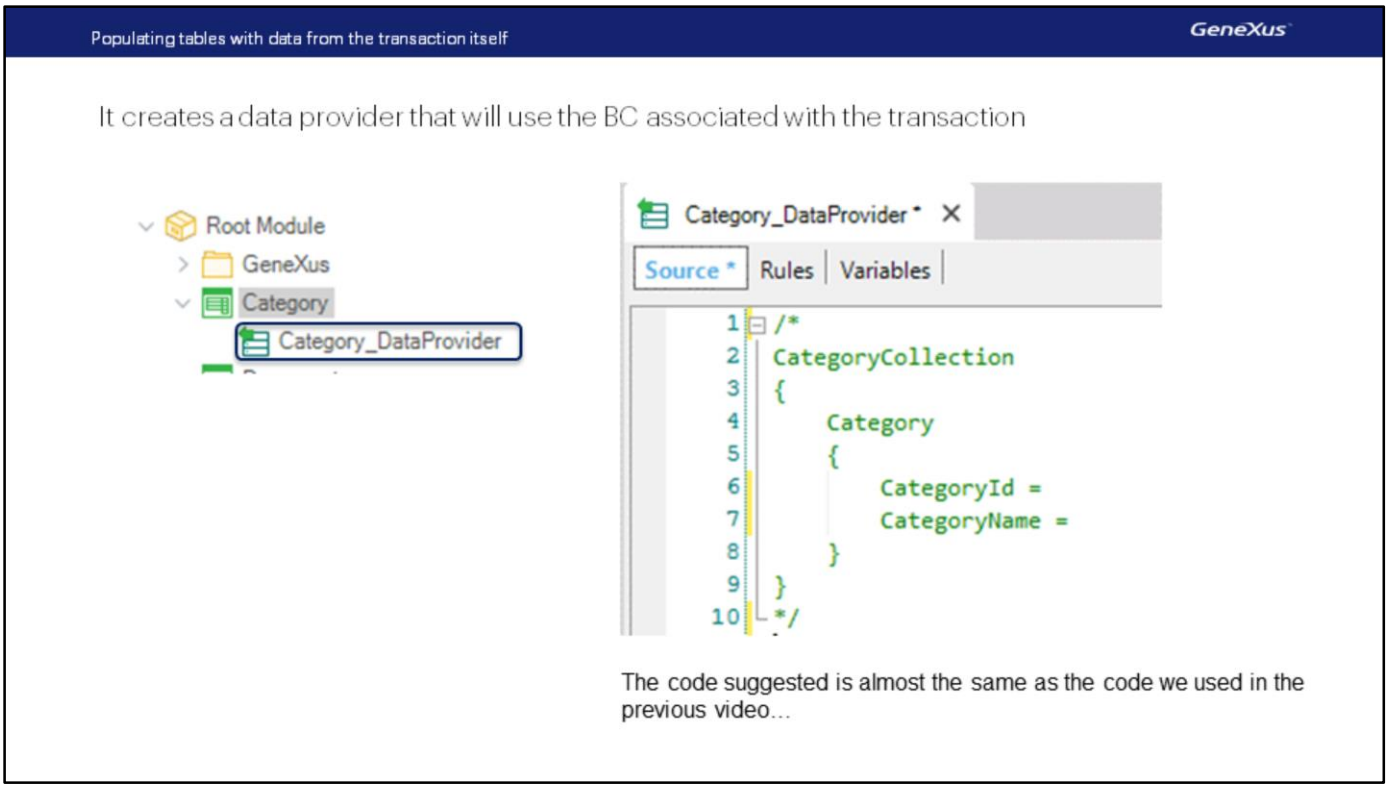


In the previous video, we saw that these tables could be initialized using the Business Component associated with the transaction, through a collection variable loaded using a Data Provider.

But GeneXus already offers a solution to initialize the data corresponding to a transaction, with no need for us to perform all the previous steps manually (obtain the Business Component, create the Data Provider and the collection variable, invoke the Data Provider, make the Insert).

To do so, the transaction has a property, located under the Data group, and called **Data Provider**. Let's see it with the Category transaction. By default, it will be set to False. We will change it to True.

In this way, we're indicating that there will be an associated Data Provider. Also, in this new property, we indicate that we will use it to initialize the table data.

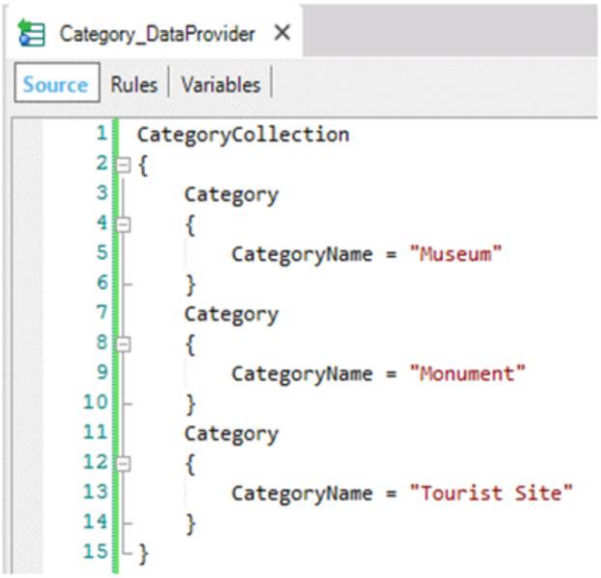


When we save, we see that an object of Data Provider type called Category_DataProvider has been created.

In addition, if the Business Component property of the transaction hadn't been set to True, it would have been automatically set to True in order to create the Business Component associated with the transaction.

If we open the Data Provider, we can see that it offers the code for us to complete the categories data.

We reuse the code that we used to complete the DP

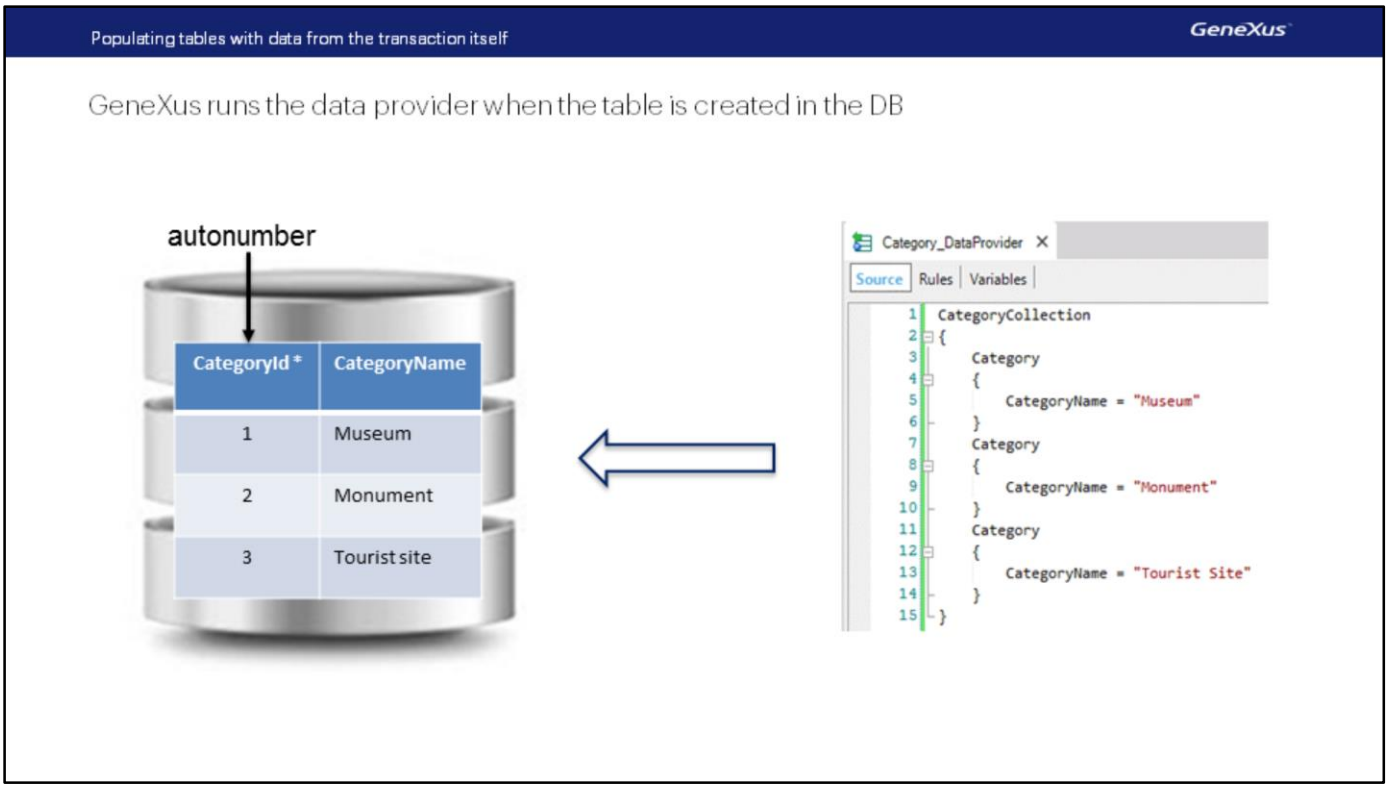


This code is almost a copy of what we did manually before.

So, we copy the code that we had written in the new Data Provider.

In this way, we have defined how the Data Provider will assign values to the new categories that will be created. What we haven't figure out yet is the moment when this Data Provider will be invoked to perform the task. The right moment will be when the table is created in the database.

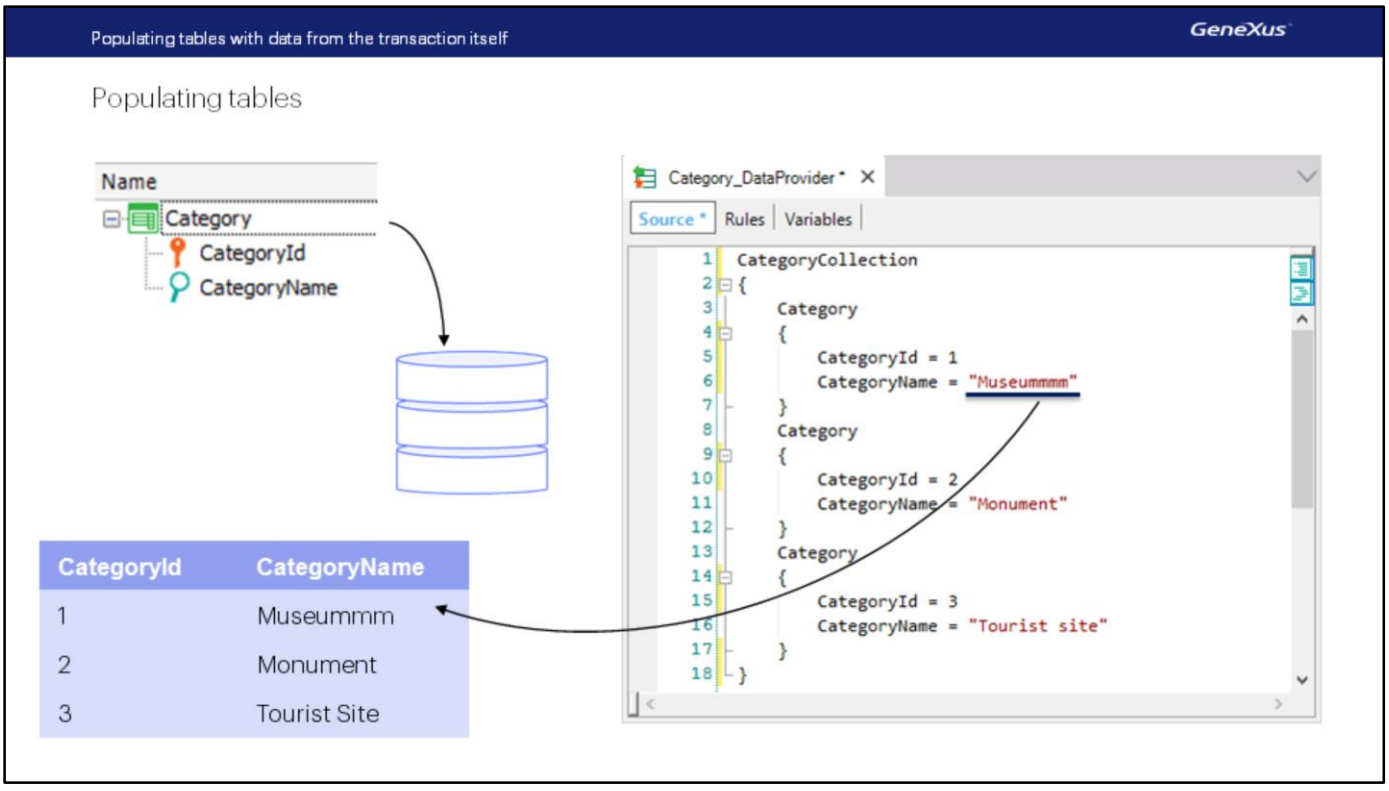
However, in order not to overload the process of creating tables, GeneXus will delay running the Data Provider until the moment the application is executed, which will be when we really need the data to be in the table.



In this case, the table has already been created and will not have data because we will delete it. We click on Remove Data to delete the data in Category and Attraction.

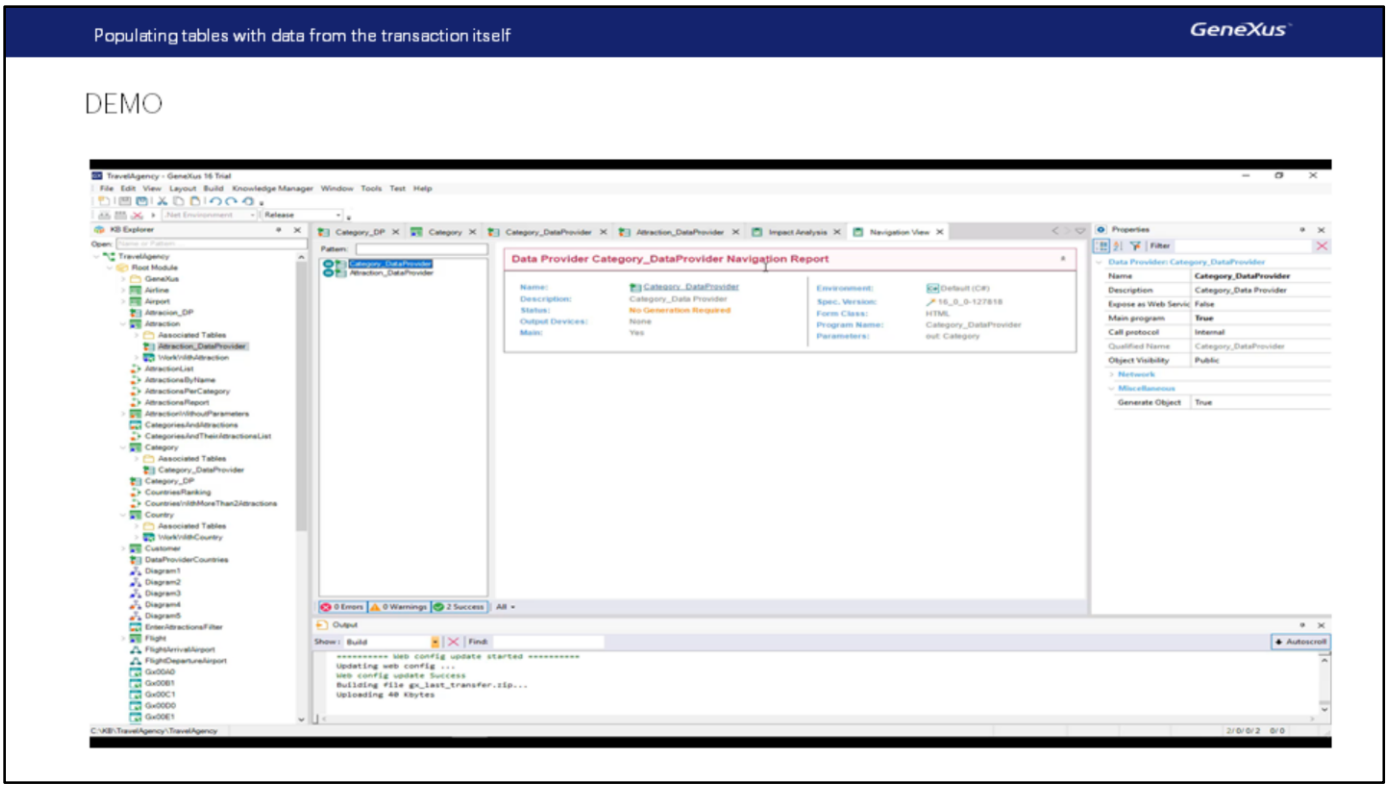
However, since we have just enabled the initialization of data in the transaction, the next time it is run GeneXus will run the Data Provider.

Note that if the table already had data in that moment, because the identifier had been set as autonumber, it would add new records. In other words, it doesn't matter if there was already a Museum category, it will insert another one. It isn't our case because we had the precaution to empty the table first. However, if the identifier wasn't autonumbered, we would have to indicate its value for each new group of the Data Provider, and if there are already records in the table with the values we are adding, they will be updated.



When trying to perform the insertion using a Business Component, it will find a duplicate key and there what it does is to perform an update. Therefore, in this example, it would change the name value of the "Museum" category for this one with many "m" letters.

Well, to populate the Attraction transaction with data we'll do the same thing we did before with Category.



[DEMO: https://youtu.be/MhzFWrA7UIw]

We run what we have done so far. F5.

Note that the navigation list informs that the initialization program of the Category table will have to be run. And that of the Attraction table as well.

And when it is executed we see that it did run those programs and we have data in the tables again (look at the identifiers! Remember they are autonumbered).

If now or later we need to change the initialization Data Provider, by adding, for example, a category that we hadn't thought of at first, after pressing F5 GeneXus will realize that the Data Provider has changed and will run it again.

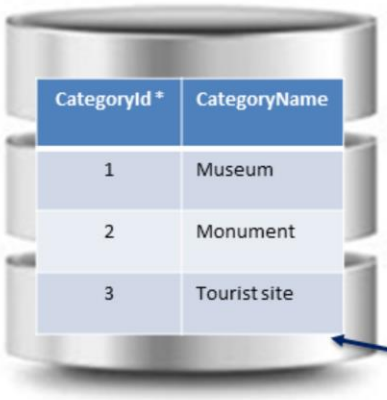
But in doing so, as the categories Museum, Monument and Tourist site already existed in the table, with autonumbered IDs, then it will reinsert them with new IDs, in addition to inserting the new one. We have a way to avoid it, by creating a Unique index for the category name, so that it is checked before inserting the record that this value is not repeated. We will not do it here.

We run and manually delete the new duplicate records.

Note that in all other aspects, the transaction keeps working as usual. That is to say, we will continue inserting and deleting its data from the screen as usual.

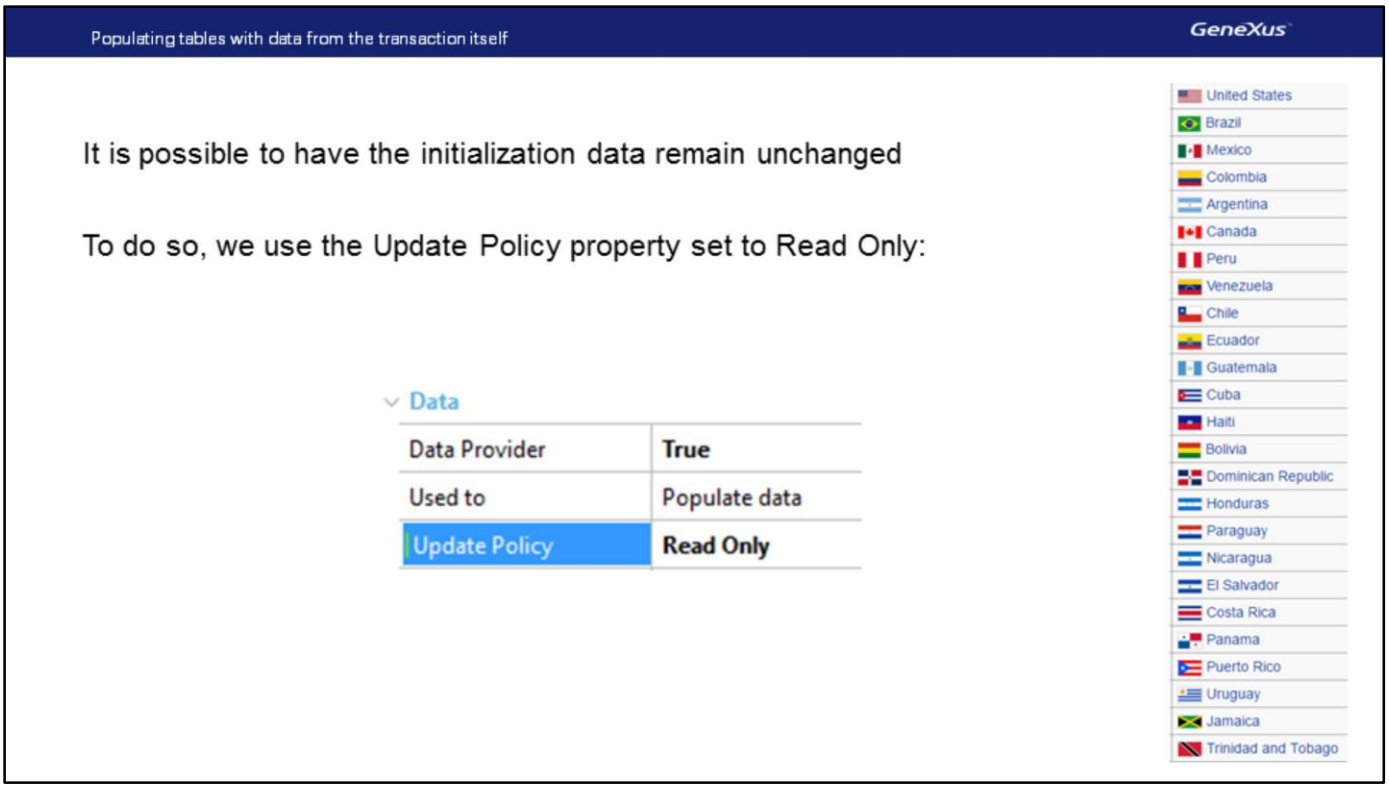
Also, its rules will be executed, and we will be able to use the associated Business Component as before. What we've seen only affects the initialization of its data.

If the DP is changed, GeneXus becomes aware of it and runs it again

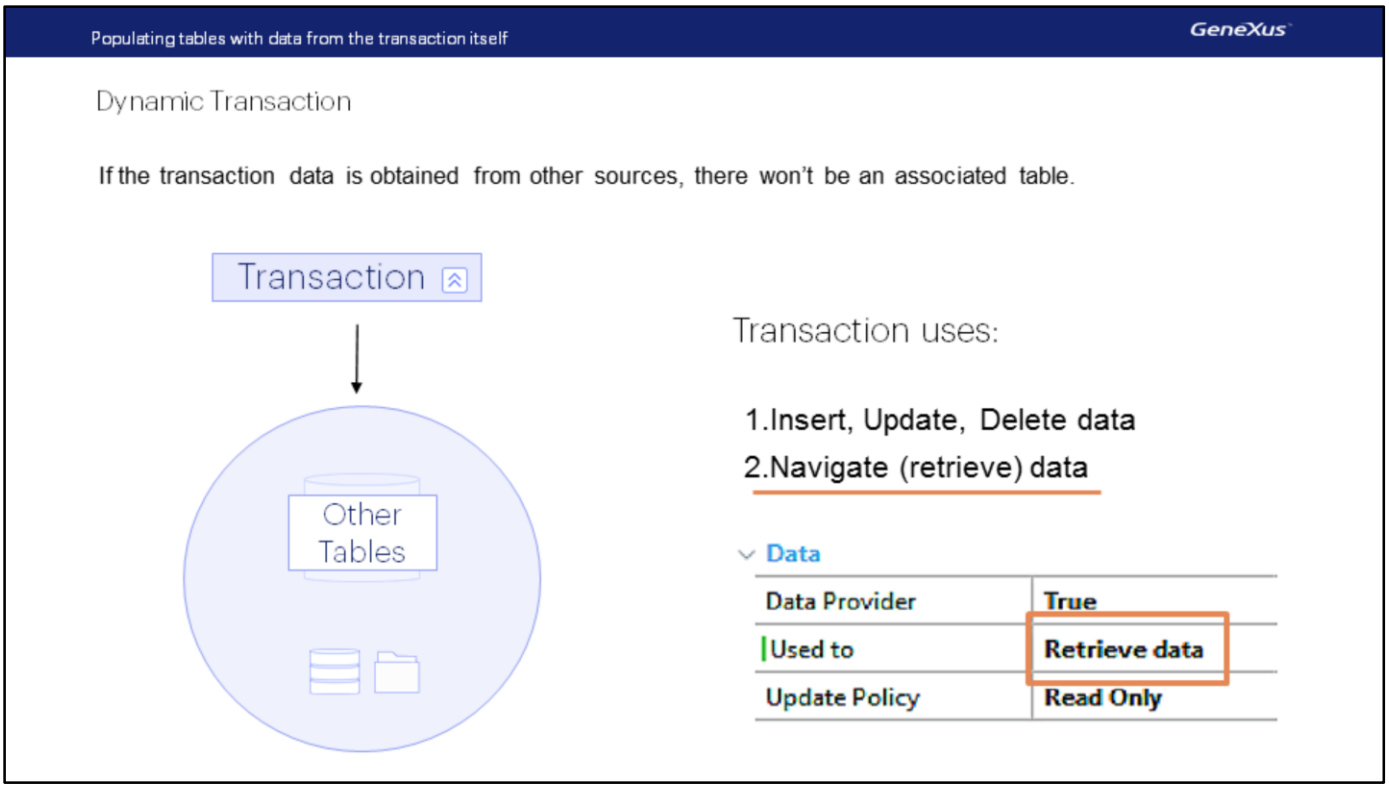


CategoryId *	CategoryName
1	Museum
2	Monument
3	Tourist site

```
CategoryCollection
{
    Category
    {
        CategoryName = "Museum"
    }
    Category
    {
        CategoryName = "Monument"
    }
    Category
    {
        CategoryName = "Tourist site"
    }
    Category
    {
        CategoryName = "Nature reserve"
    }
}
```



But if the transaction contains information that doesn't change over time, such as, for example, countries, states or provinces of a country, a system's parameters and so on, it is not necessary that the transaction or Business Component allow us to update their data. To make sure that the data is not changed, we set the “Update Policy” property to Read Only.



So far, we have used the transaction as usual, where the transaction has its associated "table".

On the other hand, we have other cases where the transaction data is obtained form other sources, which may be complex queries to the database that include looking for information in several tables, or querying other databases, and so on.

In this case, the transaction will not have this associated table because the Data Provider is responsible for obtaining this data. To use it in this way, we set the Used to property to "Retrieve data".

These transactions are called "dynamic transactions", and will not be included in this course.

In sum, to populate a table with data we wouldn't do it manually as we did in the previous video; instead, we would use the Data Provider property of the transaction.

Lastly, we Commit the changes in GeneXus Server.

GeneXus™

The power of doing.

Videos	training.genexus.com
Documentation	wiki.genexus.com
Certifications	training.genexus.com/certifications