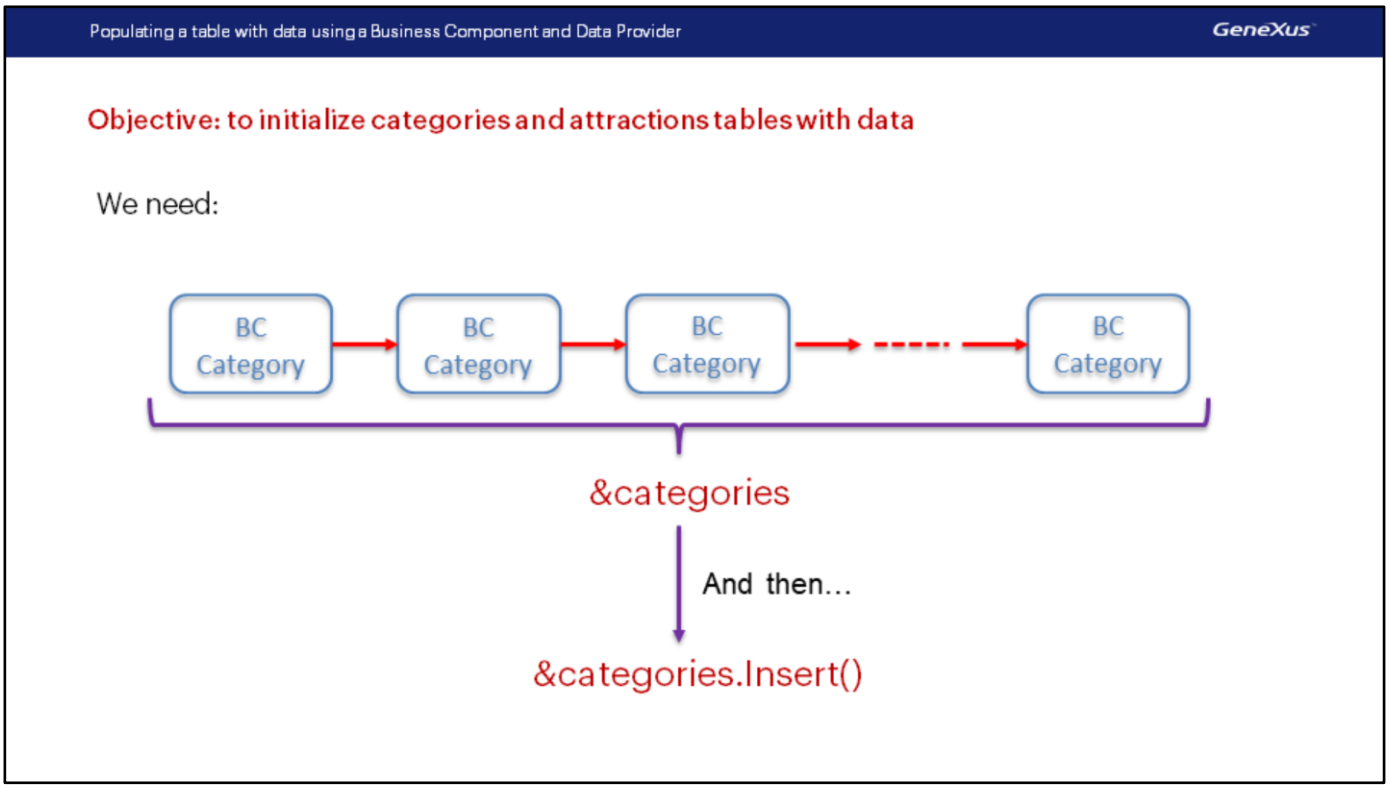
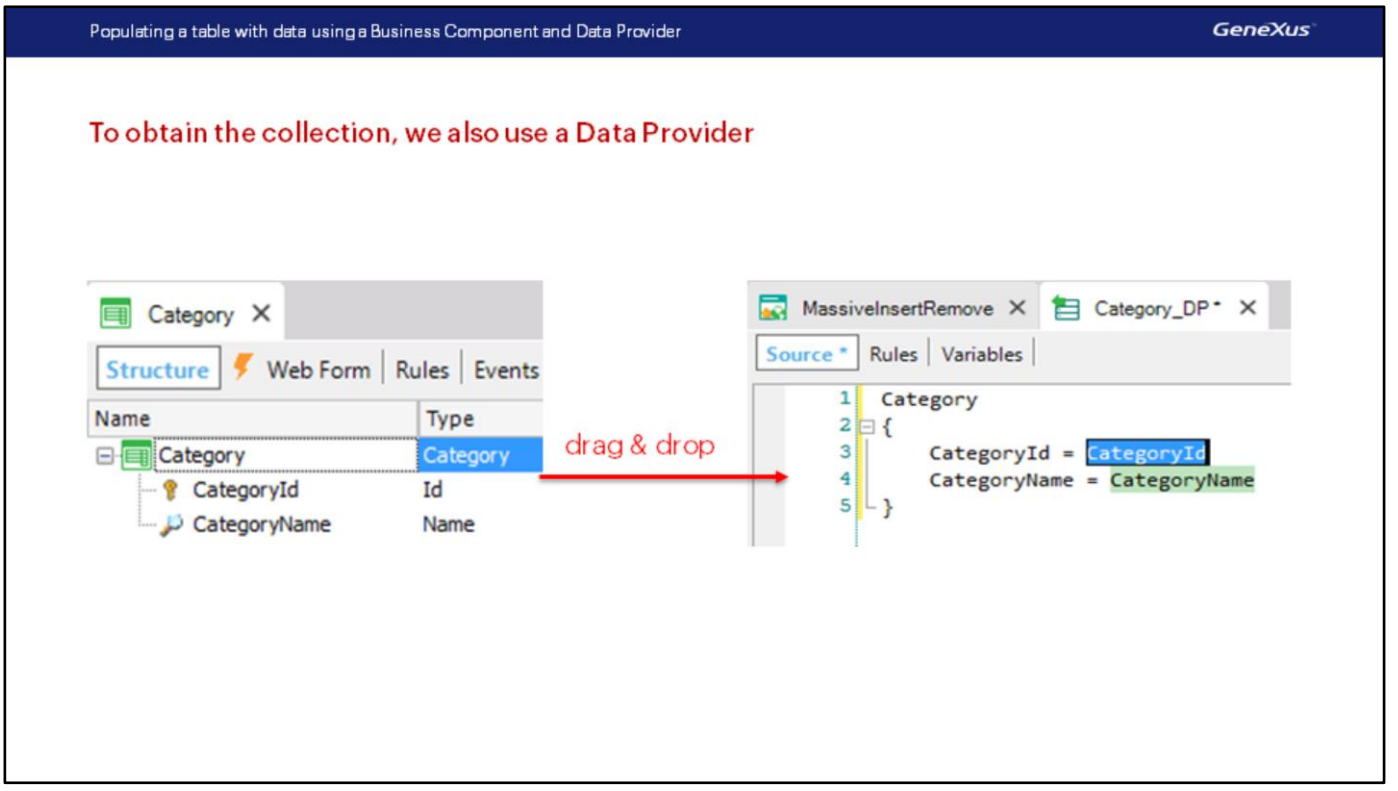
Populating with data using a Business Component and Data Provider

GeneXus 16



If we could obtain a collection variable of Business Component items corresponding to Category, loaded with the categories to be added to the database, we will only have to apply the Insert() method to this collection variable, because as we said in the previous video, this will allow us to Insert all the items, that is to say, all the Business Components in the collection.

So, now we only need to obtain this collection. How do we go about it?



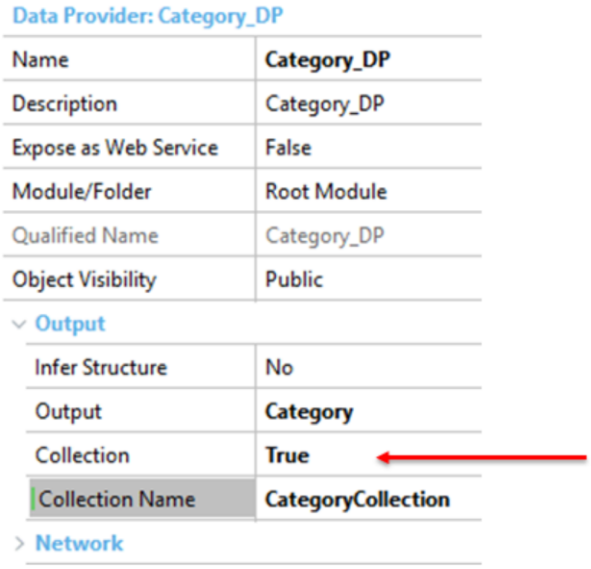
So far, we've known that a Data Provider allows us to return structured data, of both simple and collection type. In this case, we want to return a category collection, but these categories are not structured data types; instead, they are **Business Components**.

However, the structure of a Business Component is exactly the same as that of an SDT. Therefore, Data Providers will also allow us to load and return Business Components, of both simple and collection type.

We drag the Category transaction to the Data Provider Source and see that it writes the transaction structure. Note that to the left we have the Business Component's elements, which will be saved in memory, and that have the same name as the attributes even though they are not attributes. Meanwhile, the attributes of the corresponding table are now displayed on the right side. From there, the Data Provider will obtain the data to load the BC that is saved in memory.

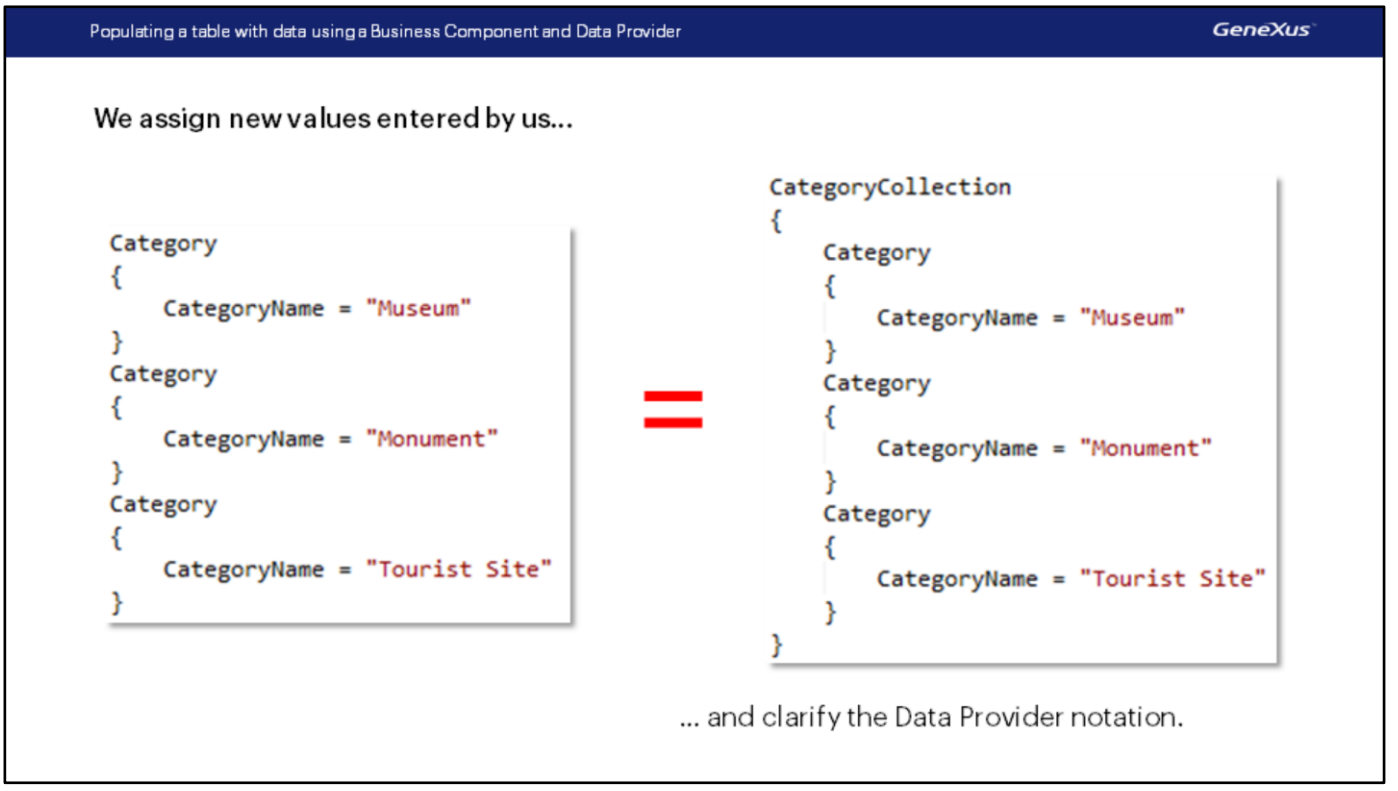
If this is what we wanted, the Data Provider should return a collection of this Business Component, because the table has many records.

To have the DP return a collection, we use the Collection property



In the properties we can see that the Output property now has the Business Component value, but the Collection property is not set to True as we need.

So, we change it and the new Collection name property is displayed. By default, it takes the Data Provider name. We change it to CategoryCollection.

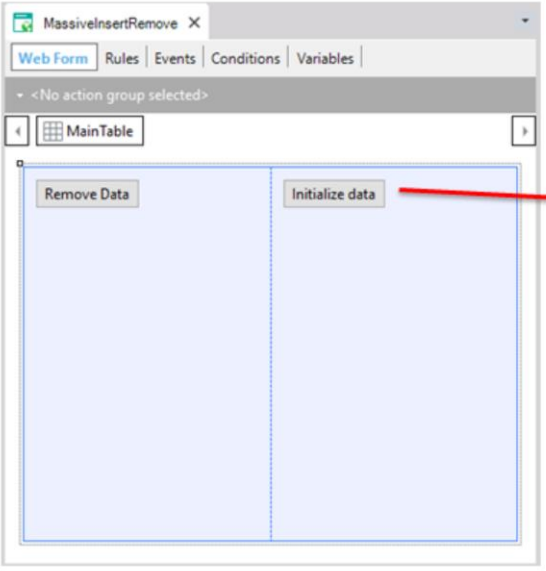


In addition, we don't want to load this collection with data from the database; instead, we want to assign it new values entered by us.

Therefore, we enter groups associated with the collection items one by one. Since CategoryId is an autonumbered attribute, we don't need to assign it a value when we want to insert a record. This is what we will do next, so we simply delete this assignment.

And since we want to return a collection called CategoryCollection –even though it's not necessary because by setting the Collection property to True, the Data Provider knows that it will return a collection– to clarify the code we can explicitly indicate what GeneXus has already inferred: to do so, we enclose all the Category groups in the CategoryCollection group corresponding to the collection.

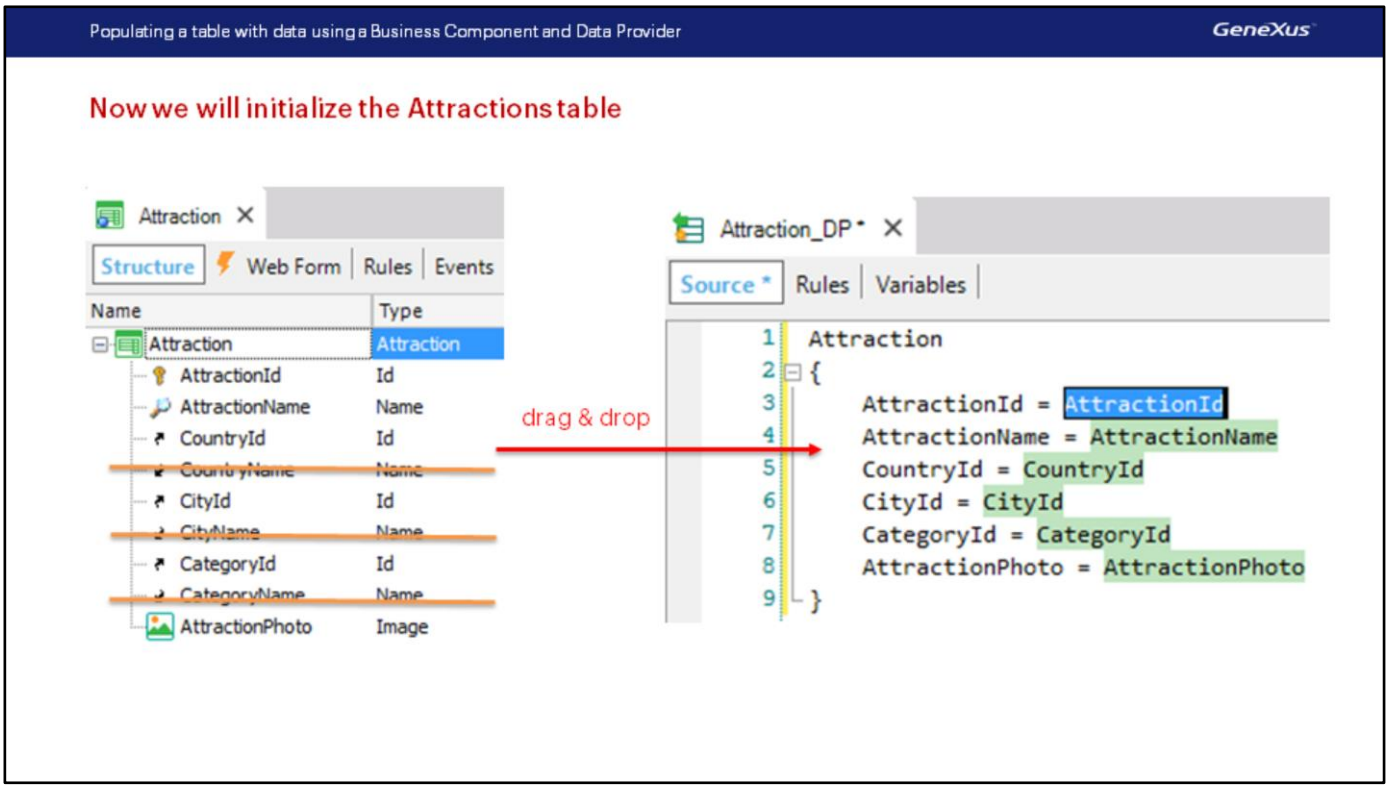
We invoke the Data Provider from a web panel




```
Event 'Initialize data'
    &Categories = Category_DP()
    &Categories.Insert()
    Commit
Endevent
```

And apply the Insert() method to the categories collection

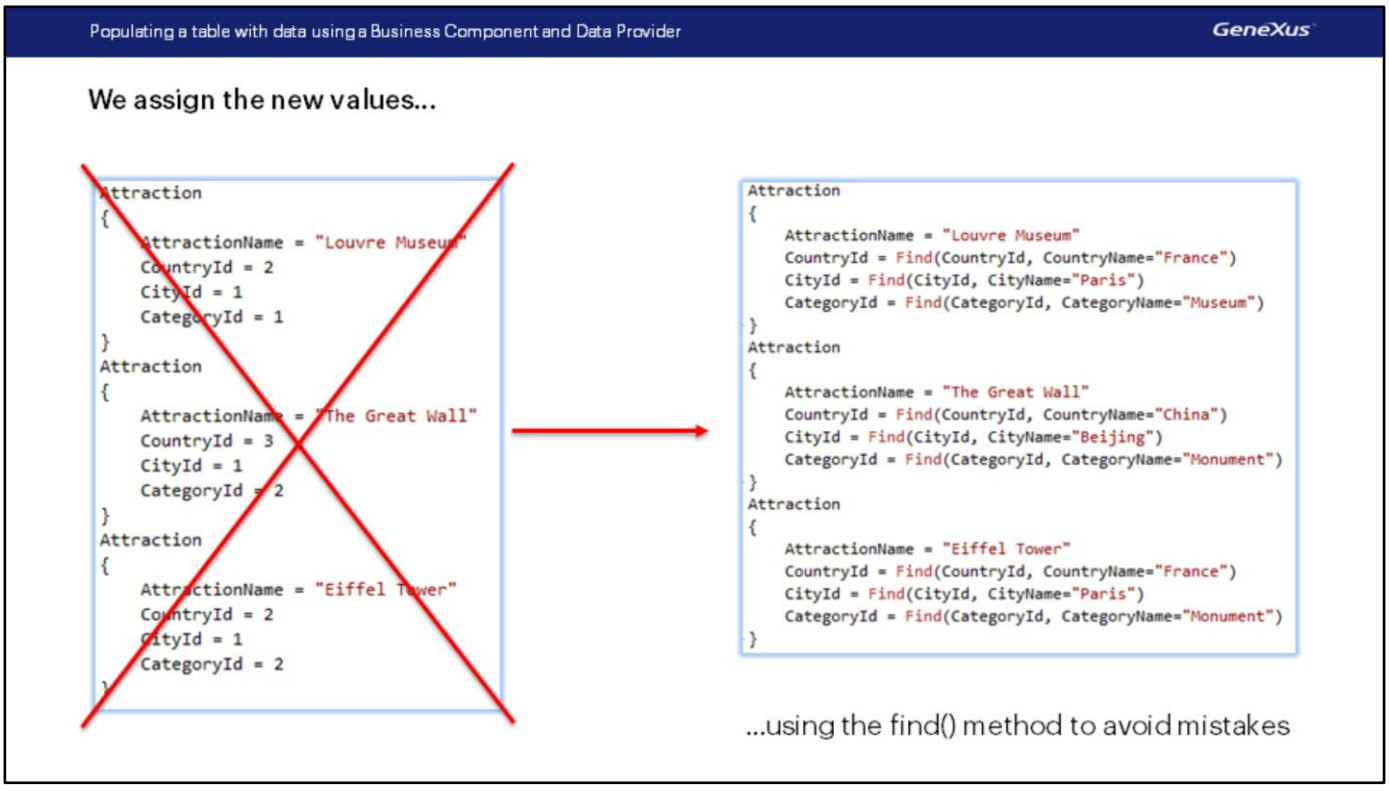
Now we only need to invoke this Data Provider from the event associated with the web panel button and enter this in the database.



Next, we will have to initialize the attractions table. Likewise, we will create a Data Provider called Attraction_DP.

We drag the Transaction (from which we had already obtained the Business Component) and see that every element of the Business Component is initialized by default with the corresponding attribute in the table.

Once again, we see that only the attributes physically present in the table are taken into account, and that the attributes inferred in the transaction or formulas are not included.

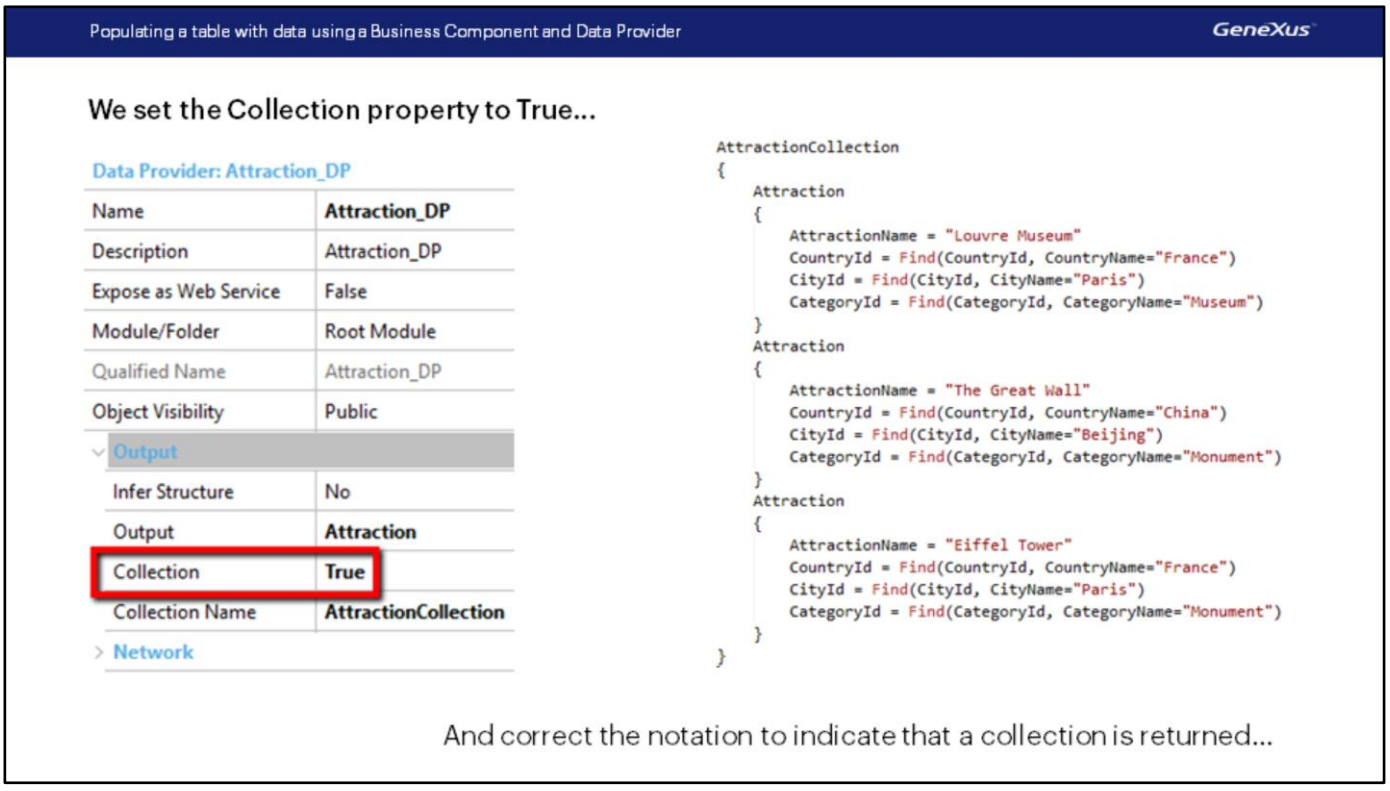


Since we're not interested in loading existing attractions (because we run this Data Provider to load the initial data), we delete all these attributes and enter these values manually. In addition, since the ID is autonumbered, we don't need to assign a value to this Business Component element either. The attractions' photos will be assigned later, so we also remove this attribute.

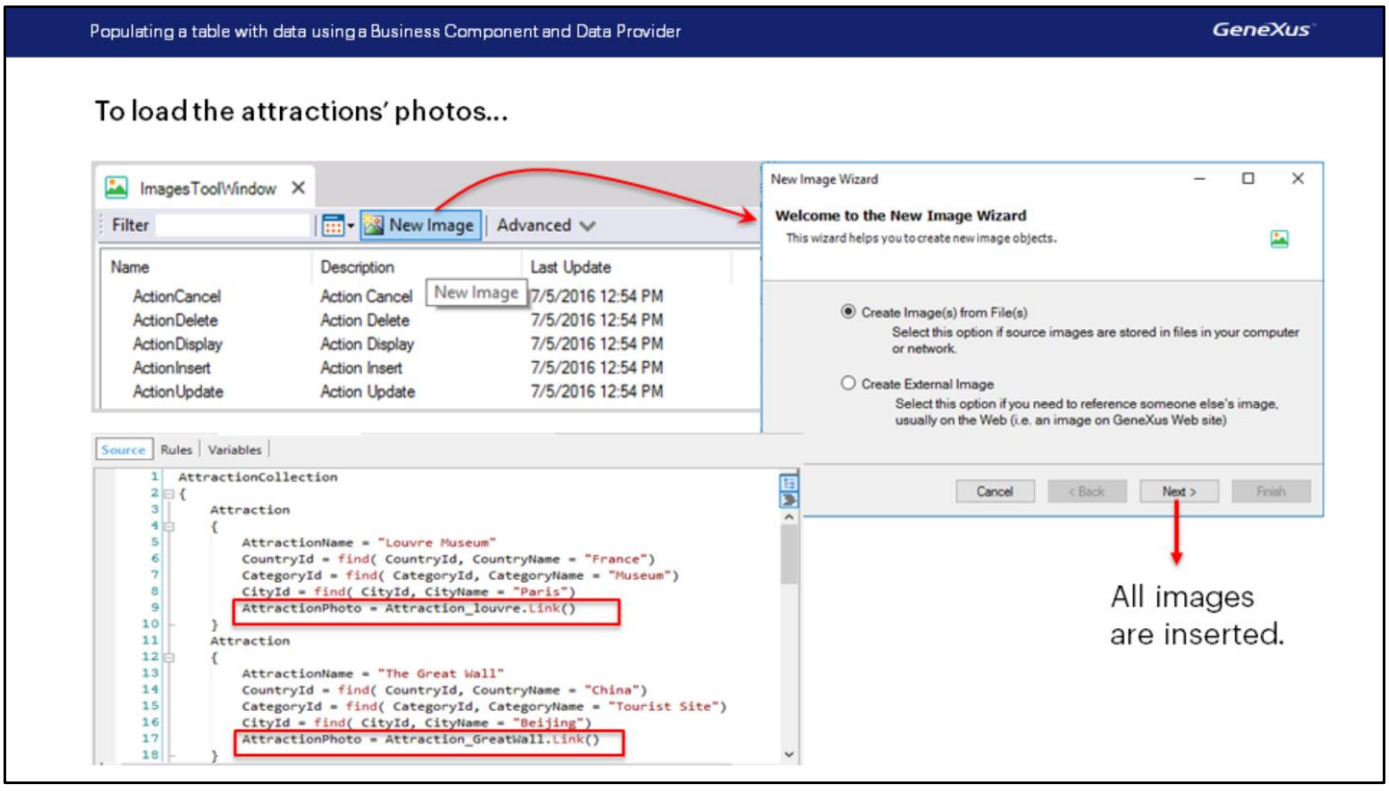
We're assigning the CountryId, CityId and CategoryId values by heart, which means that they may not exist in the corresponding tables. If any of the values doesn't exist, when trying to insert the records with the Business Component, the corresponding referential integrity checks will be triggered and the insertion will fail.

To avoid assigning values that may not exist, we will use the Find formula to find the correct identifiers based on the name of the country, city or category.

Note that the Find formulas are accessing the database only to search for the identifiers corresponding to the names we've used, but the rest of the values assigned to the Business Component are fixed.



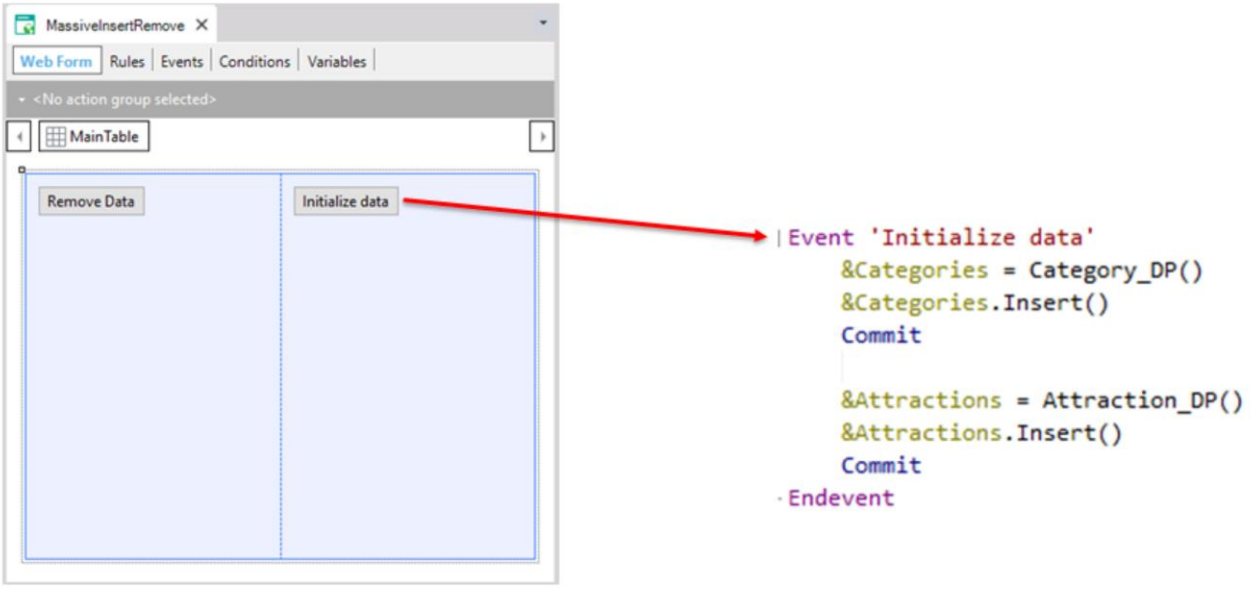
Just like we did with the categories Data Provider, we must set the Collection property to True because we will return many attractions. Also, we will adjust the notation in the source by enclosing the groups inside the AttractionCollection group to indicate that it is an attraction collection.



To also load the attractions' photos, we may insert them first as image objects in the KB...

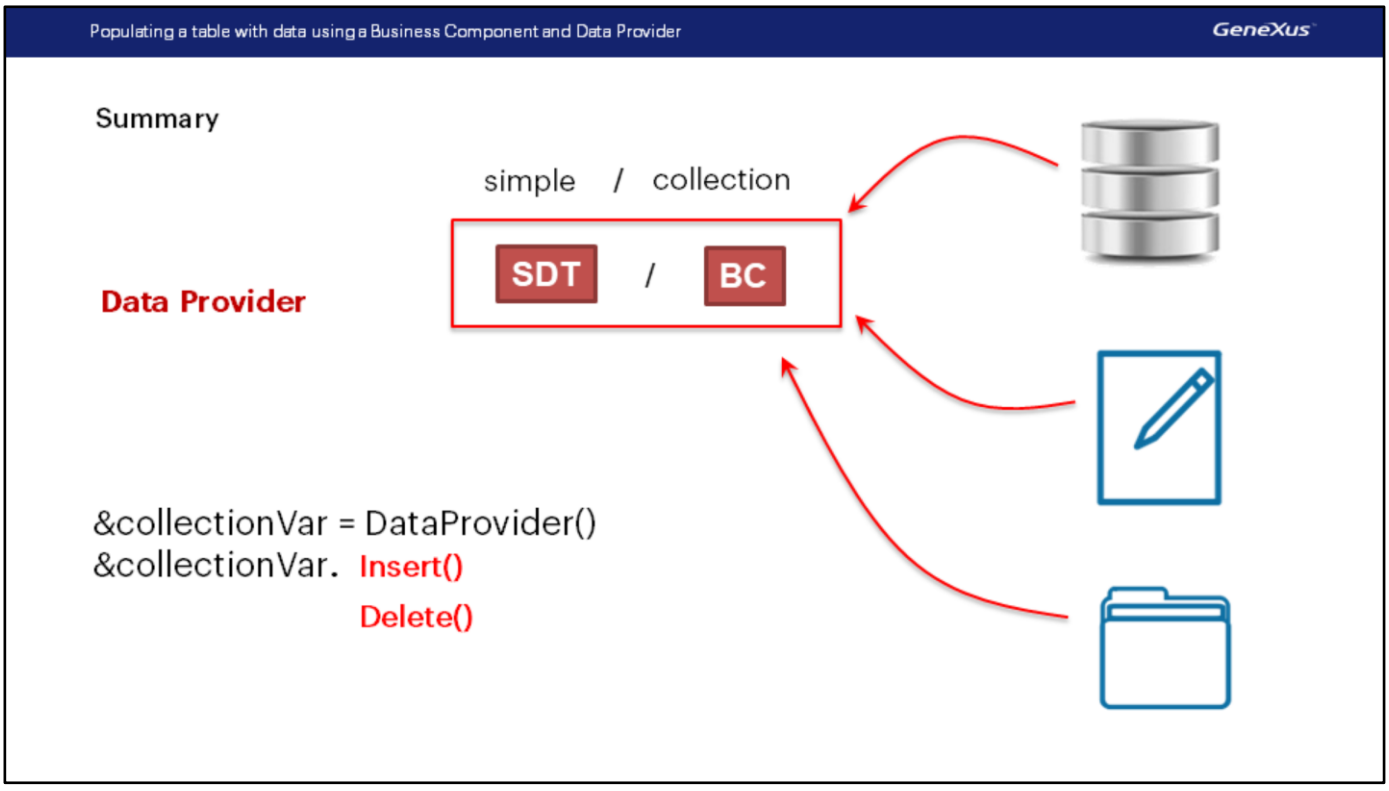
Next, for each Data Provider group we may simply assign the name of the image (dot link) to AttractionPhoto.

We add the DP invocation in the web panel event



Now we only have to invoke the Data Provider so that it returns the loaded collection.

Note that to be able to insert attractions, the categories must have been created first; for this reason, the order is the one we used in the event code.



Here we saw how a Data Provider not only allows loading a structure with data from the database, but also from fixed data.

In addition, it can also do it from other external sources, as you will see in more advanced courses.

We've also seen that a Data Provider allows loading the structure of a Business Component (and not only of an SDT) that can be of simple or collection type.

Lastly, we saw that if the structure is of collection type, we can apply methods that affect all the collection items in a single operation, such as insert() and delete().

GeneXus™

The power of doing.

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