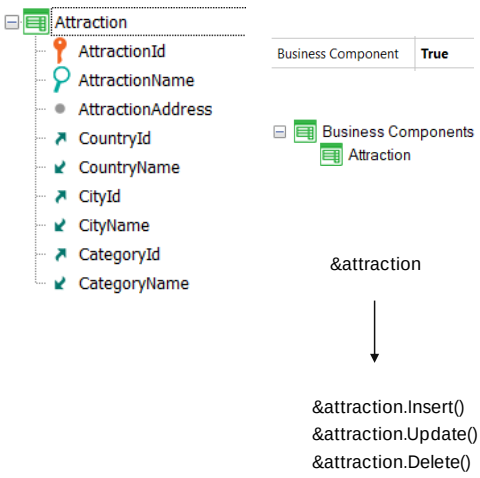
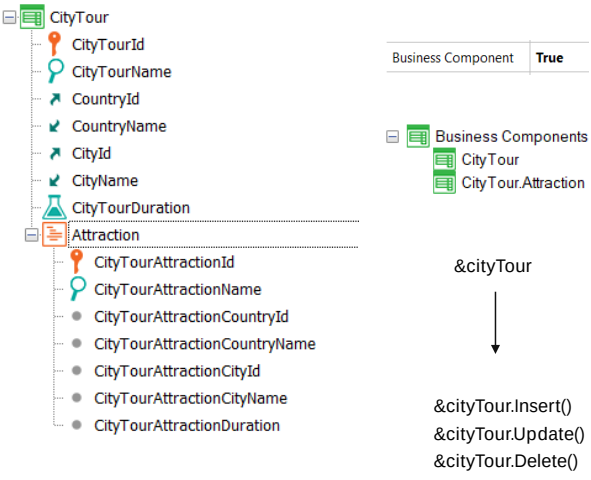
Database Update

Comparison Between One- and Two-level Business Components

GeneXus™

Single-level BC / Header of a Two-level BC



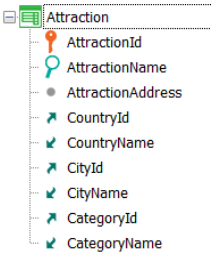


We had seen how to insert, change and delete database information from one- and two-level transactions through Business Components.

In all cases, the Business Component's data structure had to be loaded into a variable and the desired operation had to be performed on it, in the same way as with the transaction.

The difference between one- or two-level Business Components was limited to working with the lines.

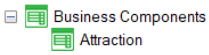
Single-level BC / Header of a Two-level BC



Insert

```
&attraction = new()
&attraction.AttractionName  = "Eiffel Tower"
&attraction.CountryId       = find( CountryId, CountryName = "France" )
&attraction.CityId          = find( CityId, CityName = "Paris" )
&attraction.CategoryId      = find( CategoryId, CategoryName = "Monument" )
&attraction.Insert()
   Commit
endif
```

Update



Delete

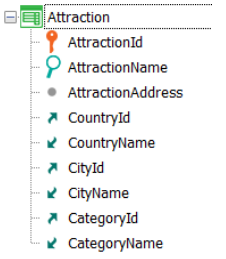
&attraction

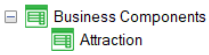
AttractionId	3
AttractionName	Eiffel Tower
CountryId	2
CountryName	France
CityId	1
CityName	Paris
CategoryId	2
CategoryName	Monument

As for the first level, the way of working is identical.

So, for example, to insert an attraction we would ask for memory, assign a value to the elements of the structure and then invoke the Insert method, which inserted the new record into the database, if everything was correct. The variable was loaded with the current values from the database, in Update mode.

Single-level BC / Header of a Two-level BC





Insert

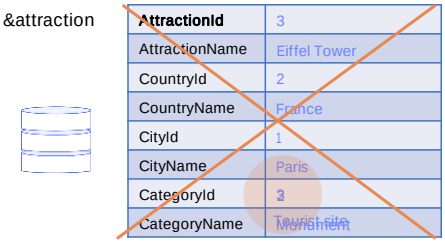
```
&attraction = new()
&attraction.AttractionName  = "Eiffel Tower"
&attraction.CountryId       = find( CountryId, CountryName = "France" )
&attraction.CityId          = find( CityId, CityName = "Paris" )
&attraction.CategoryId      = find( CategoryId, CategoryName = "Monument" )

&attraction.Insert()
   Commit
endif
```

Update

```
&attraction = Load(3)
&attraction.CategoryId  = find( CategoryId, CategoryName = "Tourist site" )

If &attraction.Update()
   Commit
endif
```

Delete

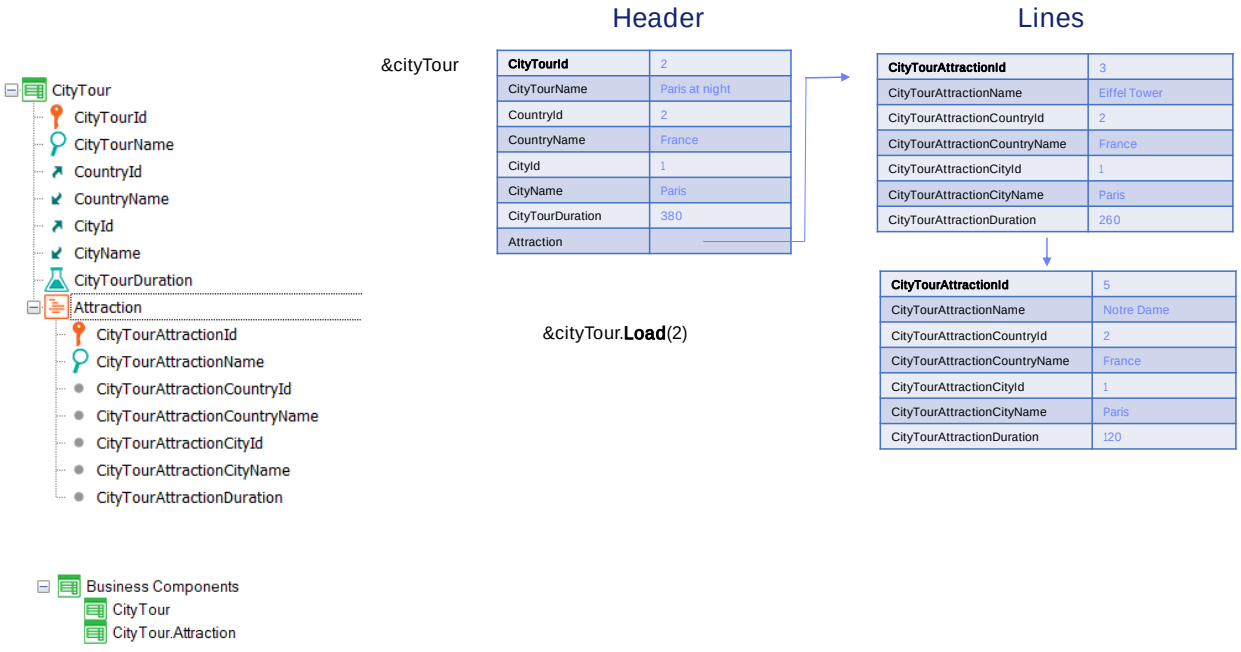
```
&attraction.Load(3)
&attraction.Delete()

If &attraction.Success()
  Commit
endif
```



To update an attraction, we first loaded the variable with the Load method, assigned the new value to the element or elements we wanted to change, and then invoked the Update method to update the record in the database. If this was successful, the variable was loaded with the current values, in Update mode.

Finally, to delete an attraction, we loaded it into the variable with the Load method, and then invoked the Delete method, which effectively removed it from the database if everything was correct. The variable was loaded with the values that the record had before being deleted and in Delete mode.

Two-level BC

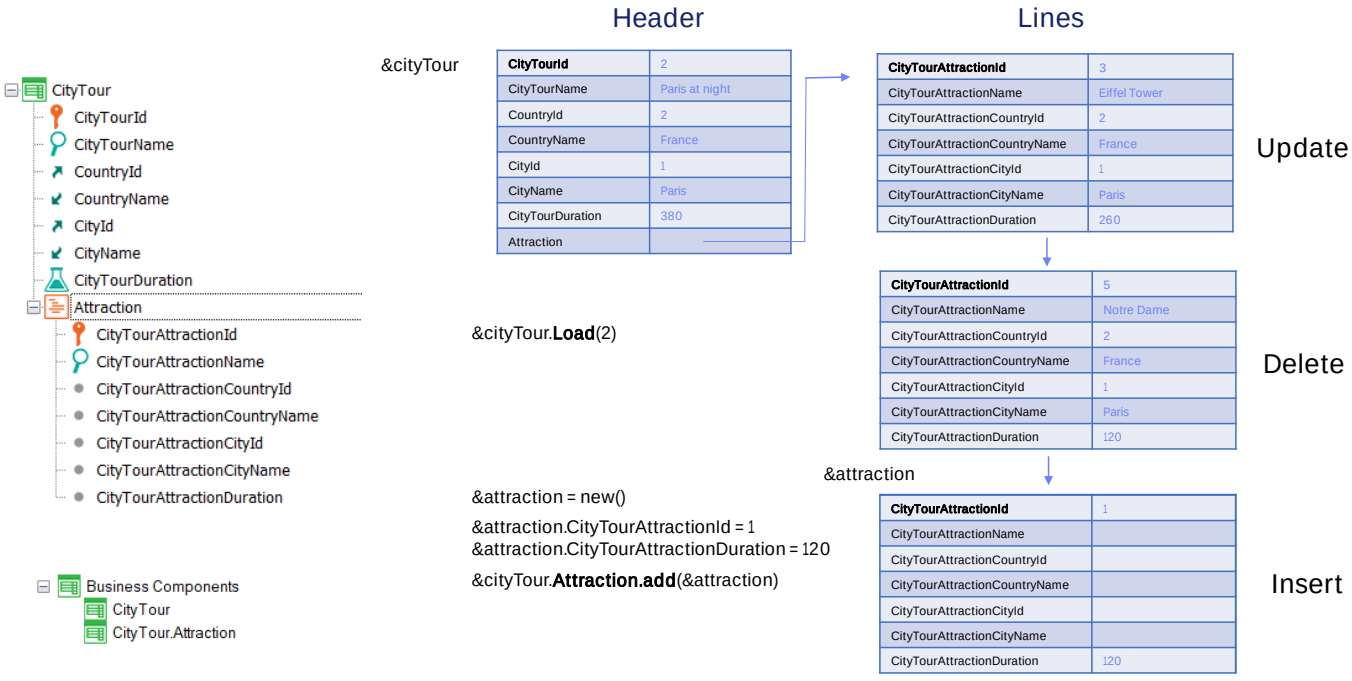


And to insert, update or delete lines, if the header already existed we first loaded the header and lines in the BC variable, and then manipulated the collection of lines.

Two-level BC

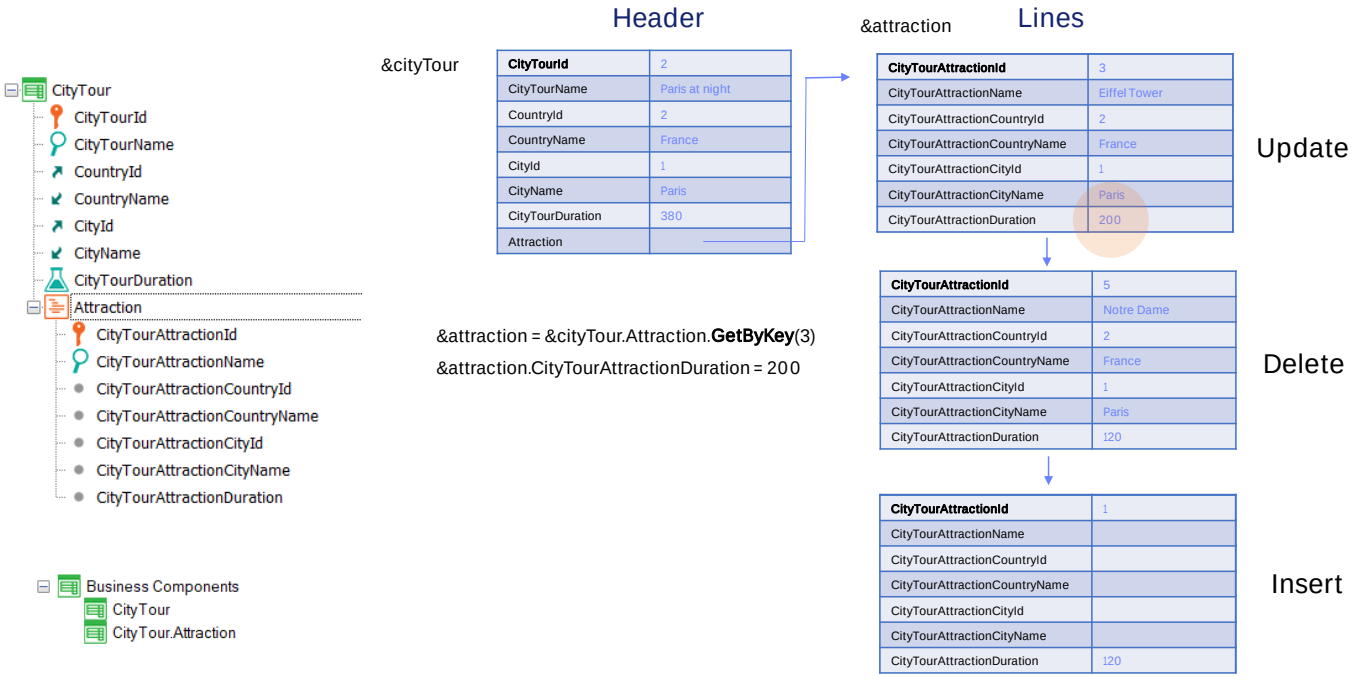
For example, if city tour 2 has two lines, one for the Eiffel Tower and another for Notre Dame cathedral and we would like to add a new one –for the Louvre museum, for example–, change the duration of the visit to the Eiffel Tower, and remove the visit to Notre Dame from the tour... that is, insert, change and delete one line... we had to work with the collection of lines...

Two-level BC



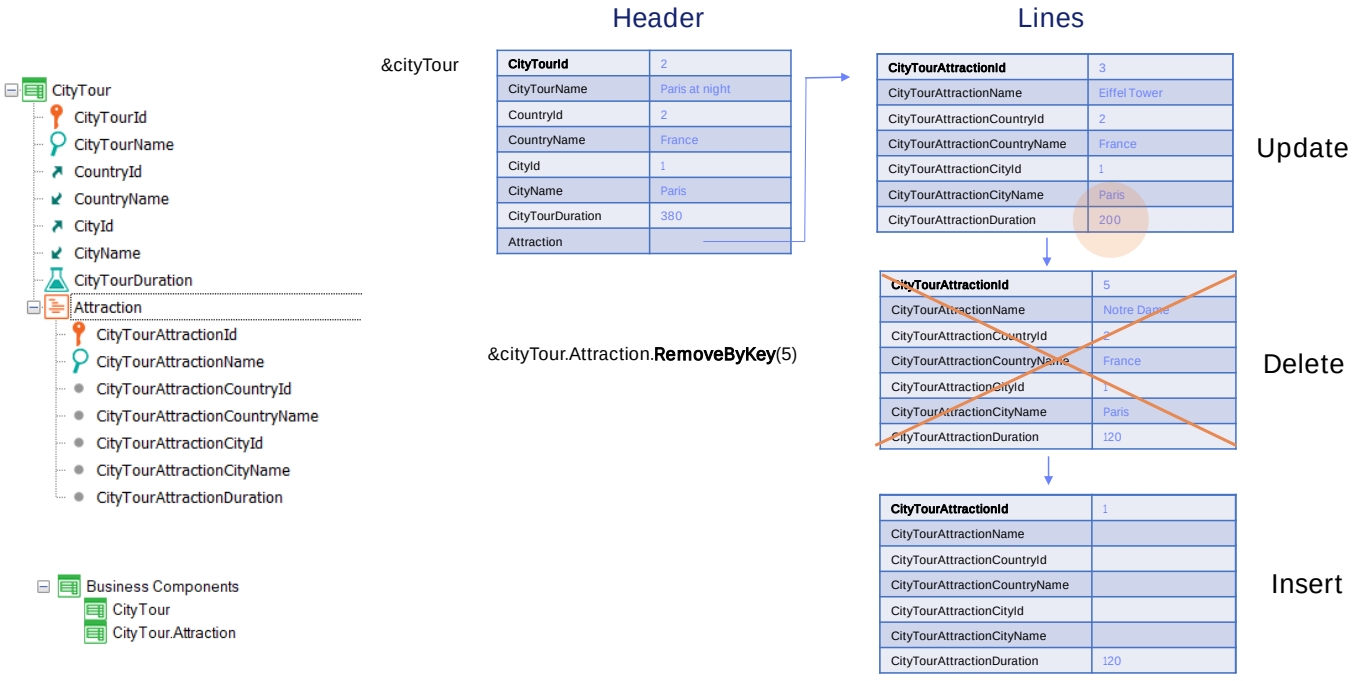
To insert one it was necessary to ask for memory space for a variable of line BC type, which will have blank elements. Assign a value to its elements and then use the add collection method to insert the &attraction variable.

Two-level BC



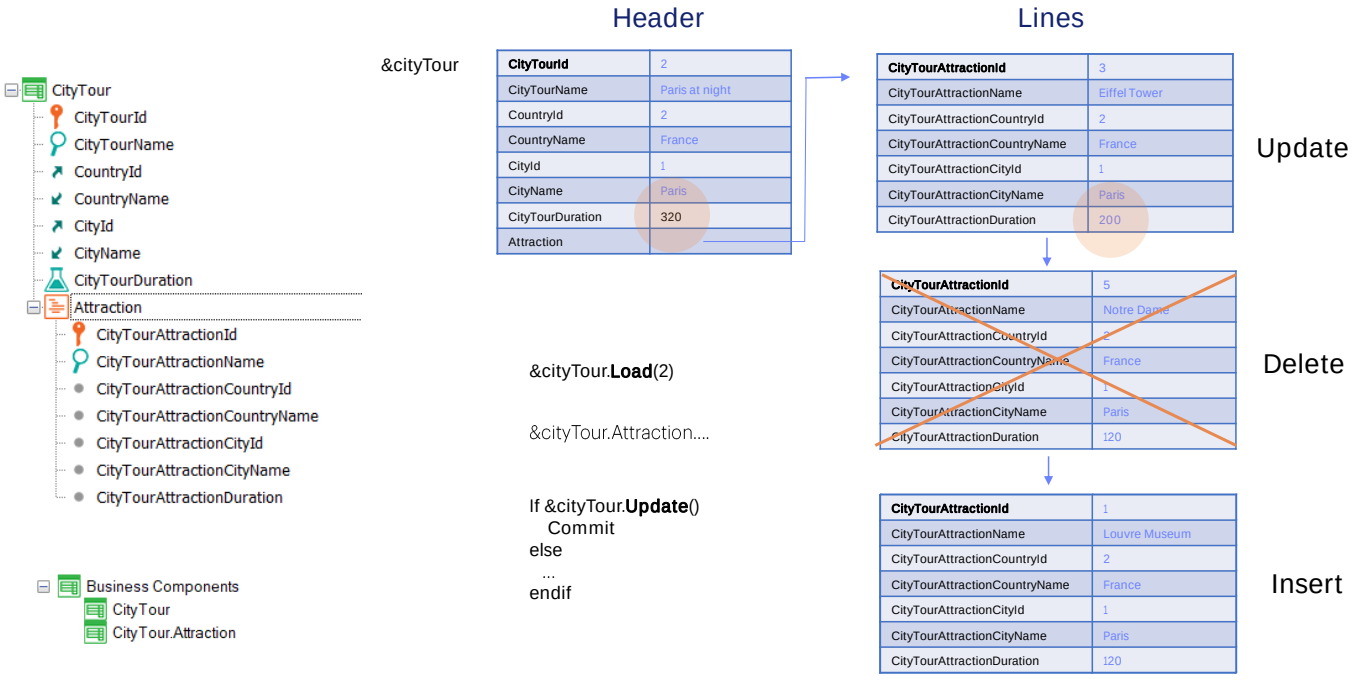
To change a line, first it is obtained in the &attraction BC variable of the lines using the GetByKey method. Then you just have to update the element that you're interested in.

Two-level BC



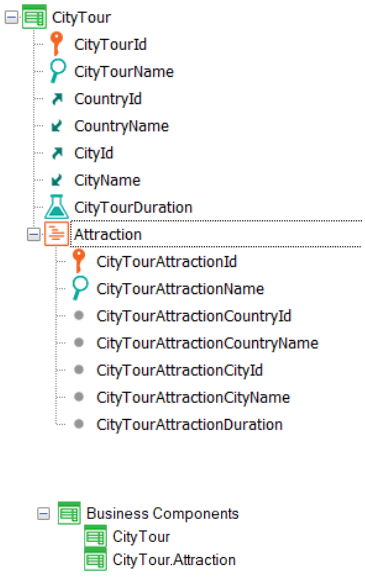
And finally, to delete a line, you have to use the RemoveByKey method from the line collection of the &cityTour variable.

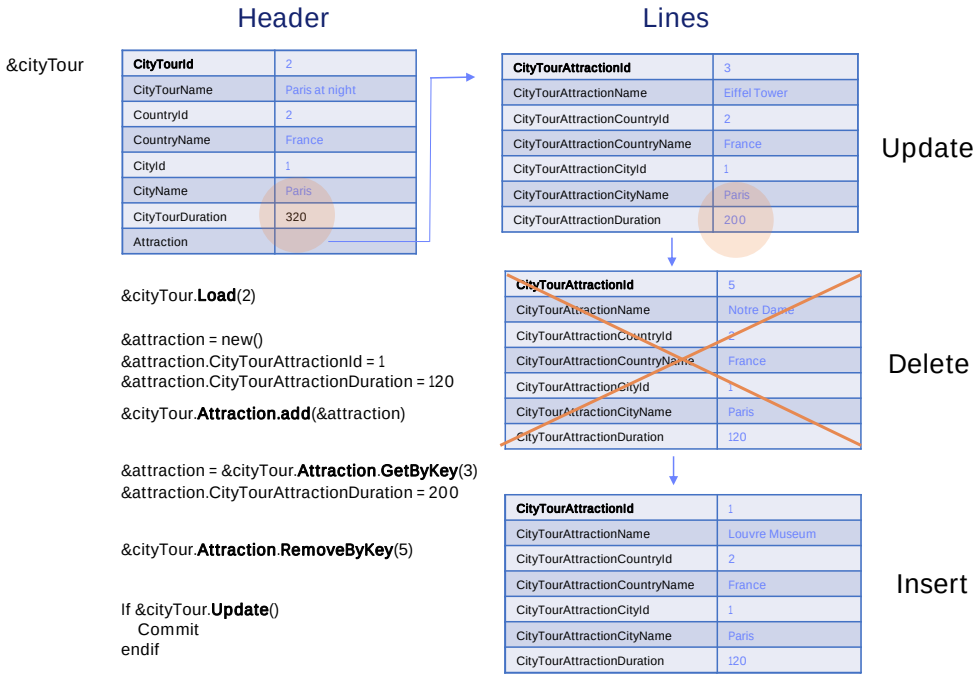
Two-level BC



After performing the three operations on the lines, the Update method is invoked, which is the one that effectively impacts what has been done to the variable in the database, executing the transaction rules, as well as the referential integrity controls.
If successful, the variable is loaded with the current values, including the case of inferred attributes and formulas. And it is left, of course, in Update mode.

Two-level BC

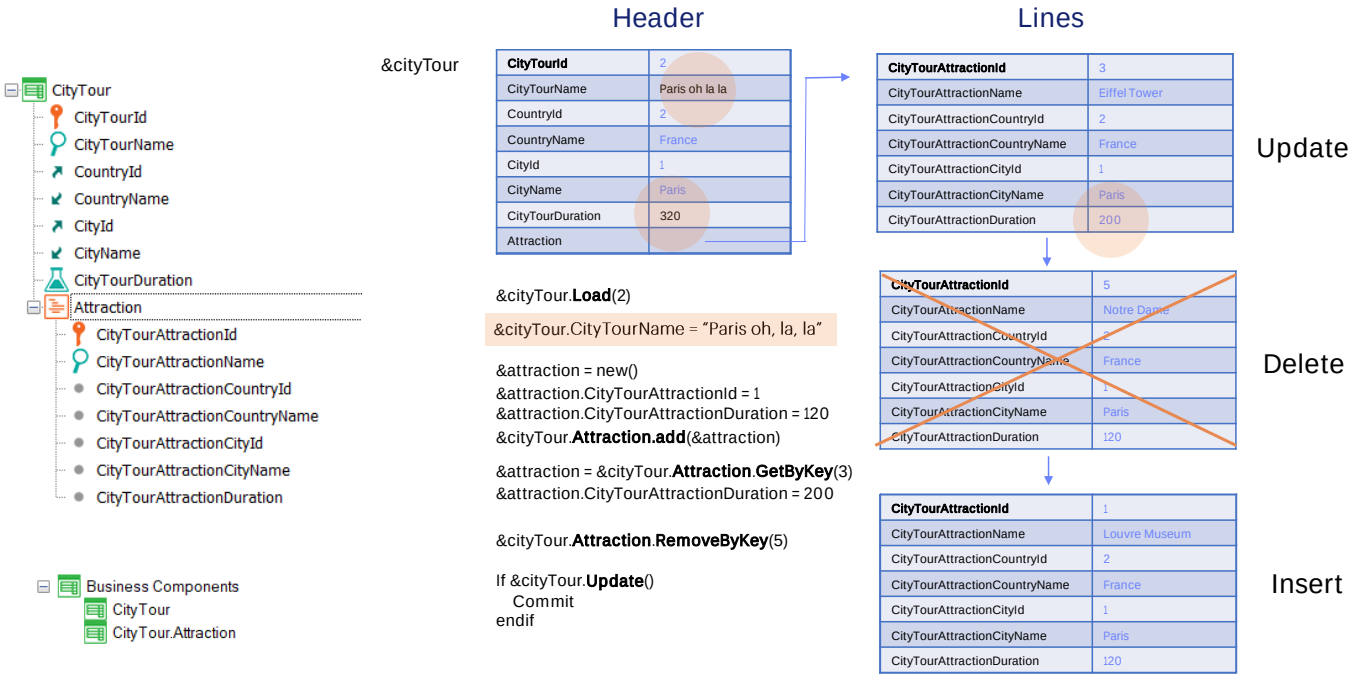





```
&cityTour.Load(2)

&attraction = new()
&attraction.CityTourAttractionId = 1
&attraction.CityTourAttractionDuration = 120
&cityTour.Attraction.add(&attraction)

&attraction = &cityTour.Attraction.GetByKey(3)
&attraction.CityTourAttractionDuration = 200

&cityTour.Attraction.RemoveByKey(5)

If &cityTour.Update()
   Commit
endif
```

Here we see the entire code of the program, in the order in which the lines of code operate.

Two-level BC



Of course, the header data can be updated as well.

In this case, you need to change the name of the city tour before working with the lines, then work with the lines, and when doing an Update...

Note that here the order doesn't matter. The header attribute could have been changed at any other time before the Update.

Methods to operate on the database

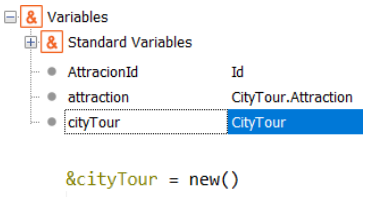
- Load(), Update(), Delete(), Insert()
- Save(), Assignment instead of Load, InsertOrUpdate()

```
&cityTour ...
...
&cityTour.Save()
```

```
&cityTour ...
...
&cityTour.Insert()
```

```
&cityTour ...
...
&cityTour.Update()
```

TrnMode.Insert



TrnMode.Udpate

```
&cityTour.Load(2)
&cityTour.Insert()
&cityTour.Update()
```

TrnMode.Delete

```
&cityTour.Delete()
```

The methods have been used repeatedly: Load method to load existing information into a BC, combined with the Update() or Delete() methods to update or delete, and the Insert() method to insert.

But other ways can be used. You just have to be clear about their differences and use cases.

For example, the Save method avoids specifying the operation in question. It's more like the Confirm button of the transaction. It will depend on the mode the variable is in, whether it will try to insert or update: if it is in Insert mode it will try to Insert, and if it is in Update mode it will try to update.

For example, every BC variable defined in an object is in Insert mode, and the same happens when new memory space is allocated to it with new().

If the variable is loaded with Load(), it immediately goes into Update mode.
After performing some operation on it, if it is successful, it will be in Update mode if it was inserted or modified, and in Delete mode if the Delete() method was applied.

Then, in order to know what operation was attempted when this Save was found, we must know the context of this variable. With the Insert and Update methods, however, this is not necessary. Regardless of the mode, you want to insert or update information in the database.

GeneXus™

training.genexus.com
wiki.genexus.com