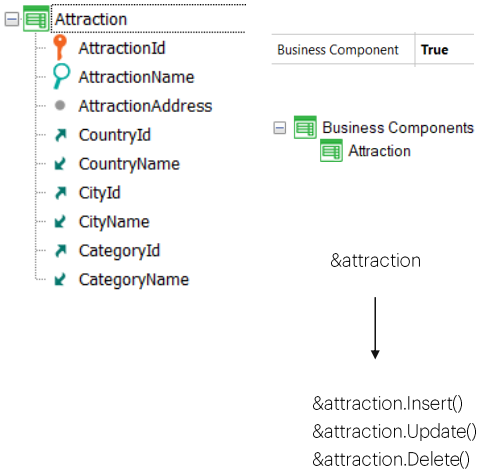
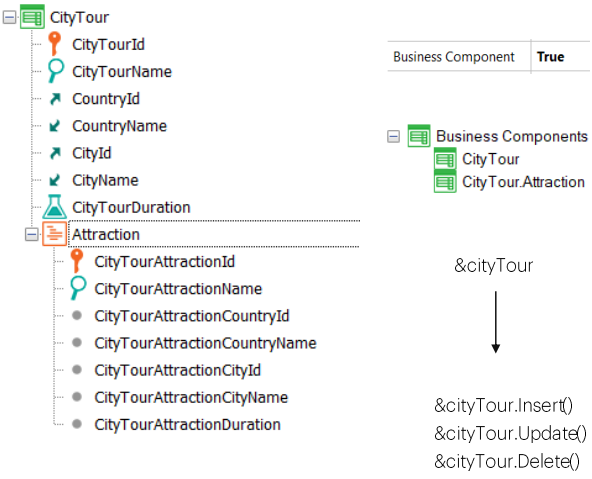
Actualización de base de datos

Comparación entre business component de un nivel y de dos

GeneXus™



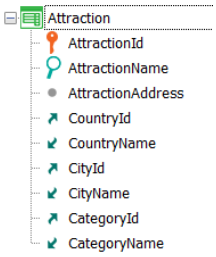


Habíamos visto cómo insertar, modificar y eliminar información de la base de datos proveniente de transacciones de uno y dos niveles a través de business components.

En todos los casos debíamos cargar la estructura de datos del Business Component en una variable y realizar sobre ella la operación deseada, de manera análoga a como lo hacíamos con la transacción.

La diferencia entre Business Component de un nivel o dos venía dada únicamente en relación al trabajo con las líneas.

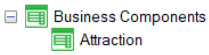
Single-level BC / Header of a Two-level BC



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

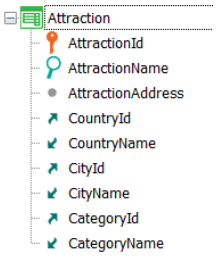
&attraction

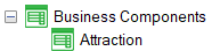
AttractionId	3
AttractionName	Eiffel Tower
CountryId	2
CountryName	France
CityId	1
CityName	Paris
CategoryId	2
CategoryName	Monument

En relación al primer nivel la forma de trabajo es idéntica.

Así, por ejemplo, para insertar una atracción pedíamos memoria, asignábamos valor a los elementos de la estructura y luego invocábamos al método Insert, que insertaba el nuevo registro en la base de datos, si todo estaba bien. La variable nos quedaba cargada con los valores actuales de la base de datos y en modo Update.

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





Insert

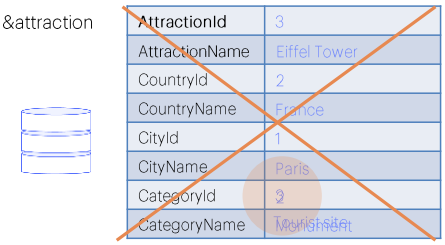
```
&attraction = new()
&attraction.AttractionName  = "Eiffel Tower"
&attraction.CountryId       = find( CountryId, CountryName = "France" )
&attraction.CityId          = find( CityId, CityName = "Paris" )
&attraction.CategoryId      = find( CategoryId, CategoryName = "Monument" )
&attraction.Insert()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

Update

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

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

Delete

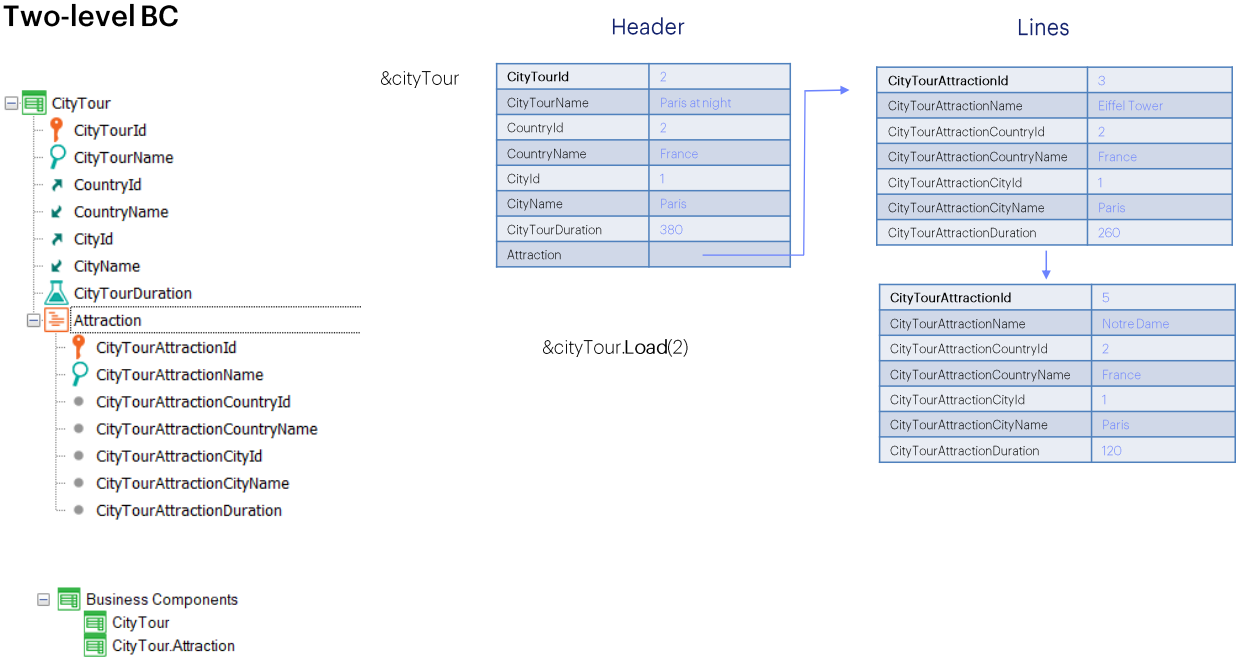
```
&attraction.Load(3)
&attraction.Delete()

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



Para actualizar una atracción, cargábamos primero la variable con el método Load, asignábamos el nuevo valor al elemento u elementos que queríamos modificar y luego invocábamos al método Update, para realizar la actualización del registro en la base de datos. Si ésta es exitosa, la variable nos queda cargada con los valores actuales, y en modo Update.

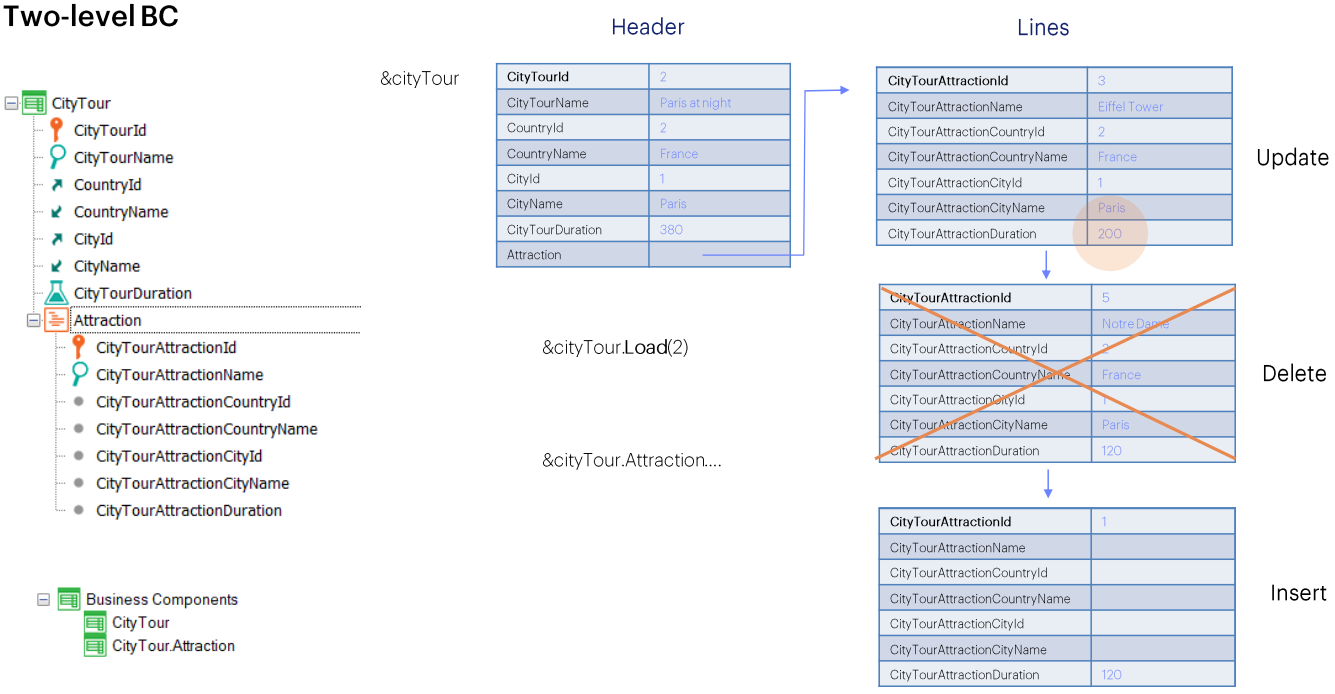
Por último, para eliminar una atracción, la cargábamos en la variable con el método Load, y luego invocábamos al método Delete, que era el que efectivamente la eliminaba de la base de datos si todo estaba bien. La variable queda cargada con los valores que tenía el registro antes de ser eliminado y en modo Delete.

Two-level BC



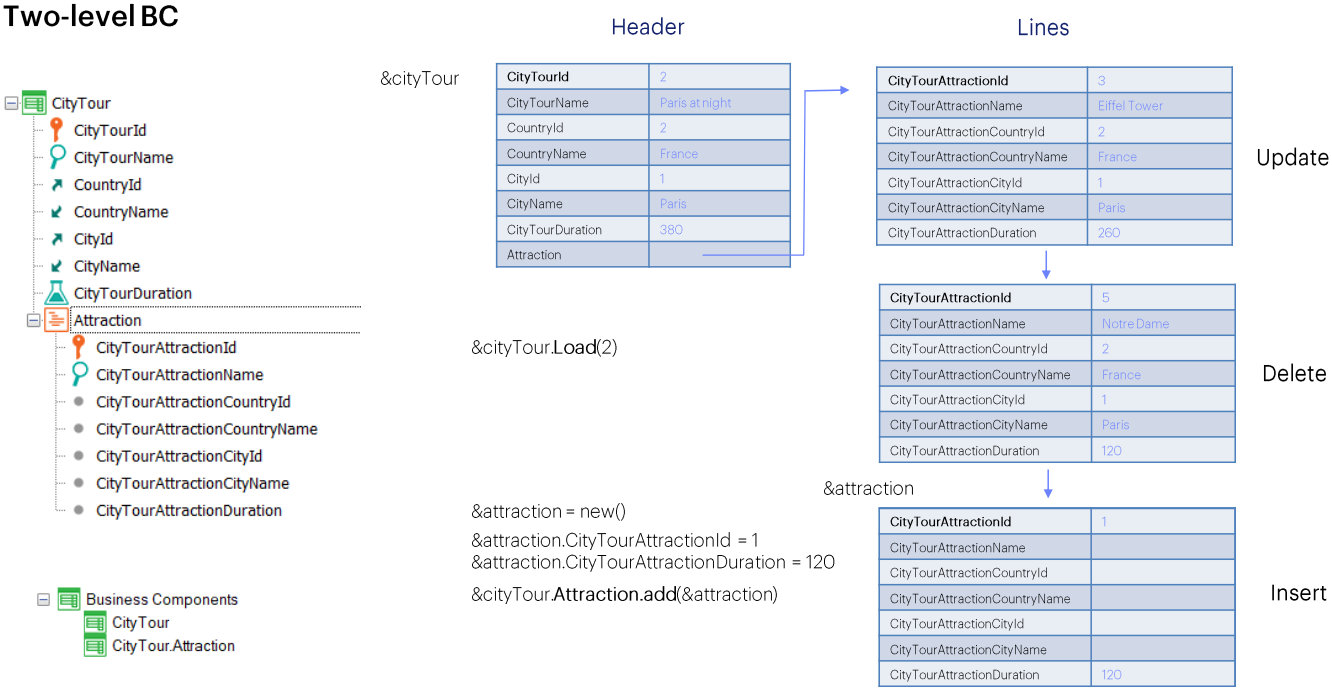
Y para insertar, actualizar o eliminar líneas, si el cabezal existía entonces primero cargábamos en la variable BC cabezal y líneas, y luego manipulábamos la colección de líneas.

Two-level BC



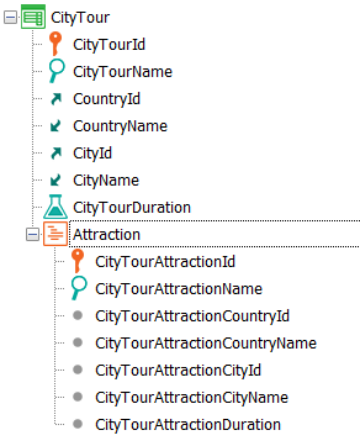
Por ejemplo, si el city tour 2 tiene dos líneas, una para la Torre Eiffel y otra para la catedral de Notre Dame y quisiéramos agregar una nueva, para el museo del Louvre, modificar la duración de la visita a la torre Eiffel y eliminar del tour la visita a Notre Dame... es decir, insertar, modificar y eliminar una línea... teníamos que trabajar con la colección de líneas...

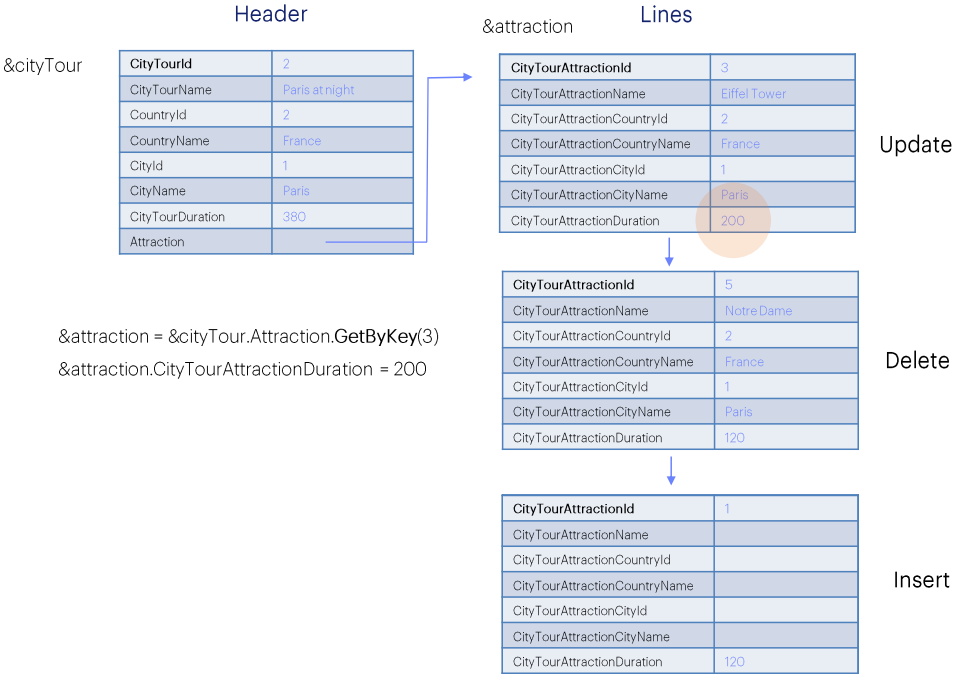
Two-level BC



Para insertar una había que pedir espacio de memoria para una variable del tipo el BC de las líneas, que tendrá sus elementos vacíos. Asignar valor a sus elementos y luego utilizar el método add de colecciones para insertar la variable &attraction.

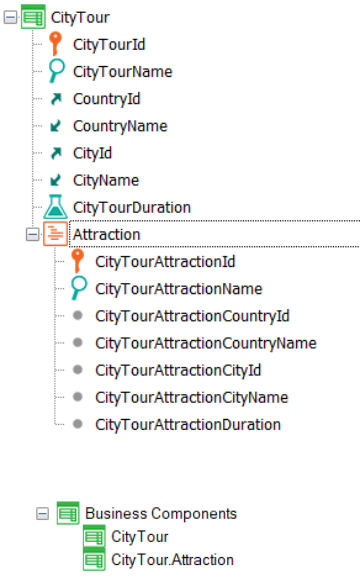
Two-level BC

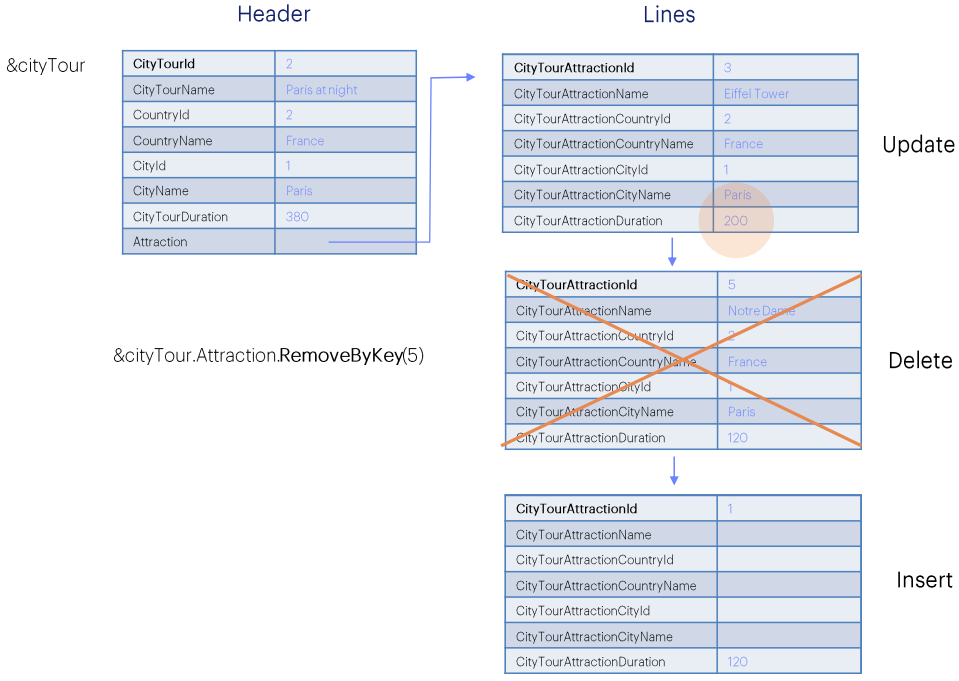




Para modificar una línea primero la obtenemos en la variable &attraction BC de las líneas, con el método GetByKey. Luego simplemente actualizamos el elemento que nos interese.

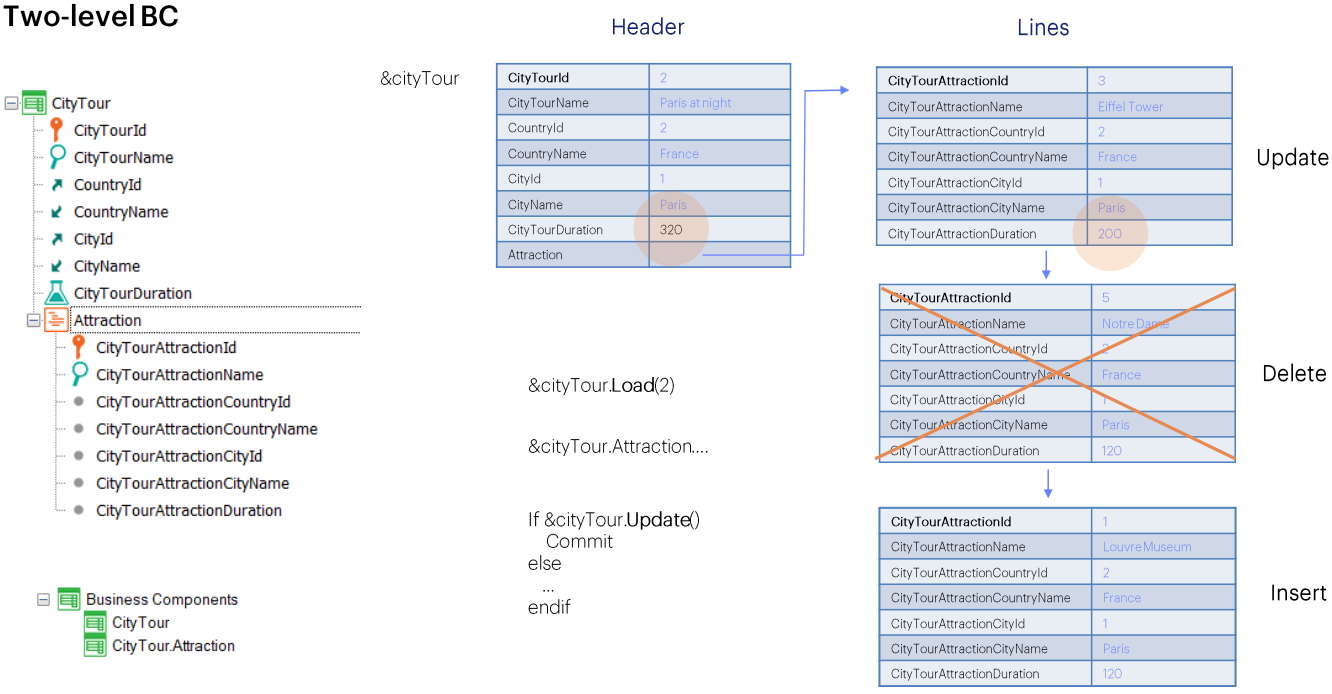
Two-level BC





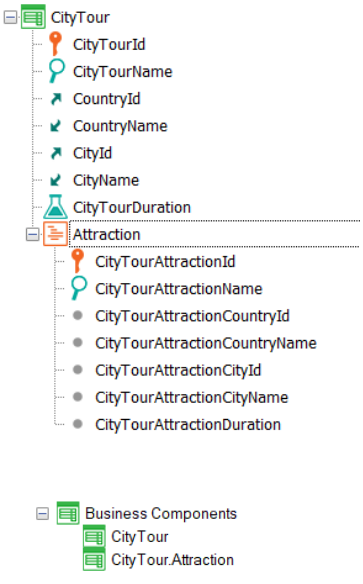
Y por último, para eliminar una línea, utilizamos el método RemoveByKey de la colección de líneas de la variable &cityTour.

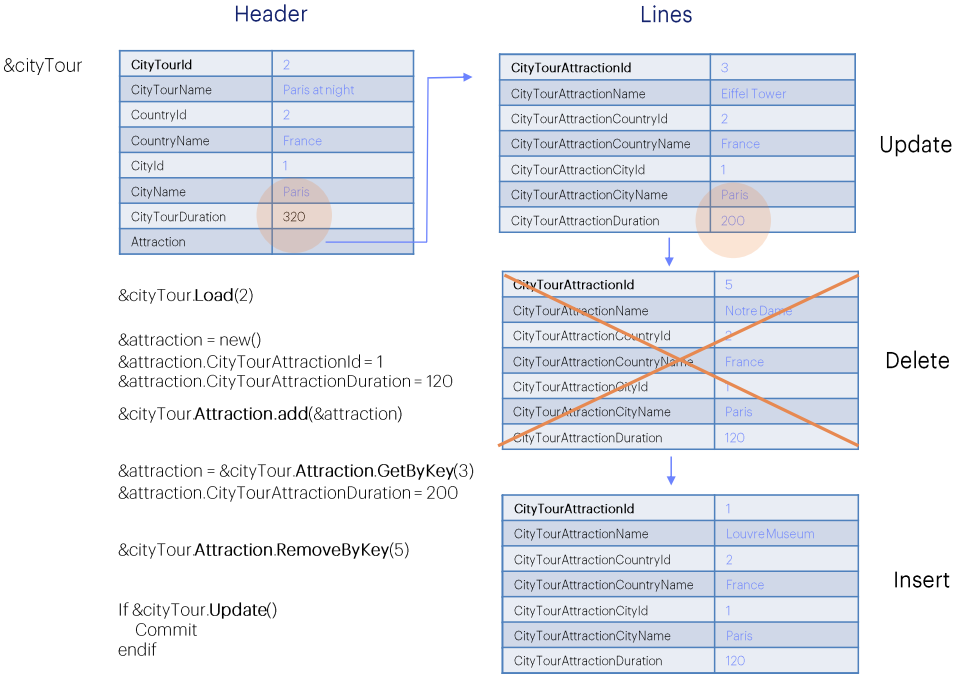
Two-level BC



Después de haber realizado las tres operaciones sobre las líneas, invocamos al método Update que es el que efectivamente impacta lo realizado en la variable en la base de datos, ejecutando las reglas de la transacción, así como los controles de integridad referencial.
De ser exitosa, la variable nos queda cargada con los valores actuales, incluyendo el caso de atributos inferidos y fórmulas. Y queda, lógicamente, en modo Update.

Two-level BC

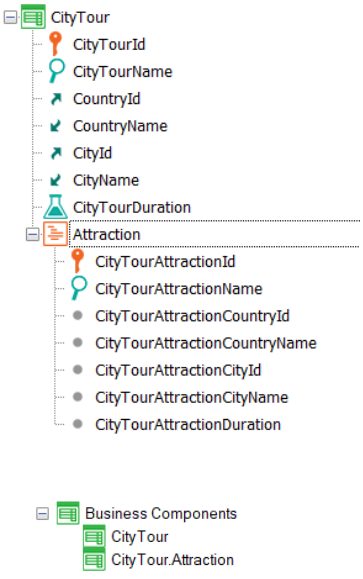


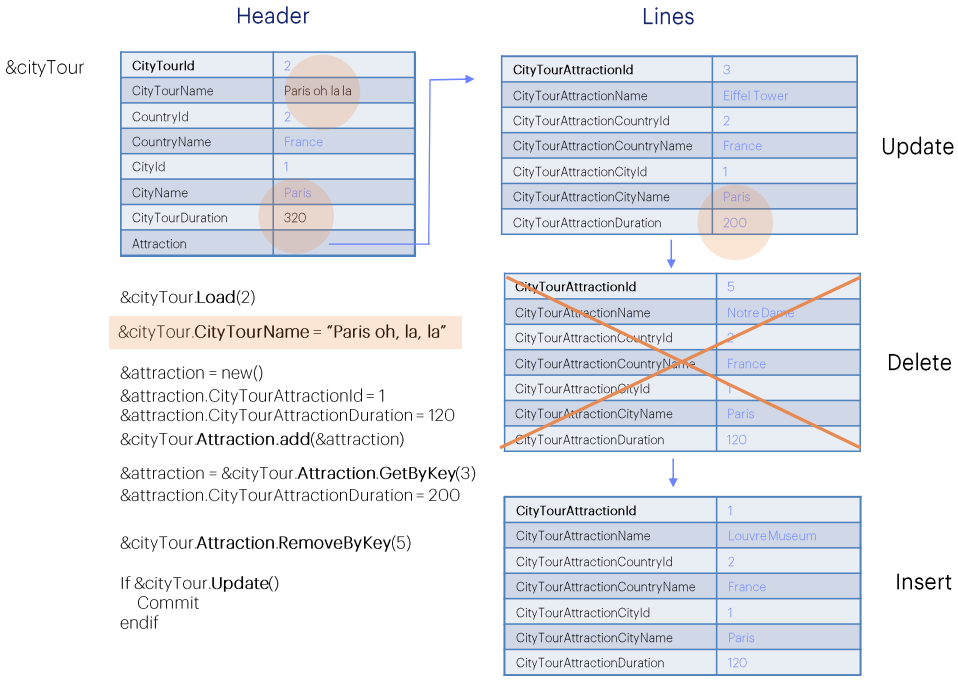



```
&cityTour.Load(2)

&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
```

Aquí vemos el código completo del programa, en el orden en el que fuimos operando con las líneas.

Two-level BC





Por supuesto nada nos impide actualizar también información del cabezal.

En este caso le cambiamos el nombre al city tour antes de trabajar con las líneas, luego trabajamos con las líneas y luego al hacer Update...

Observemos que aquí el orden no importa. Podríamos haber modificado el atributo del cabezal en cualquier otro momento antes del Update.

Methods to operate on the database

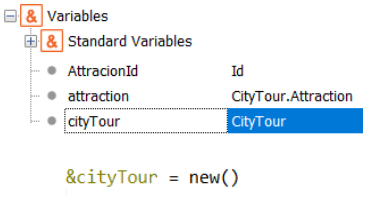
- Load(), Update(), Delete(), Insert()
- Save(), Asignation instead of Load, InsertOrUpdate()

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

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

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

TrnMode.Insert



TrnMode.Udpate

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

TrnMode.Delete

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

Hemos estado utilizando repetidamente los métodos: Load para cargar información existente en un BC, en combinación con los métodos Update() o Delete() para actualizar o eliminar, y el método Insert() para insertar.

Pero podemos utilizar otras formas. Solo hay que tener claras las diferencias y casos de uso.

Por ejemplo, el método Save nos ahorra especificar la operación de la que se trata. Se parece más al botón de Confirm de la transacción. Va a depender del modo en el que se encuentre la variable, si intentará insertar o actualizar: si se encuentra en modo Insert intentará Insertar y si está en Update actualizar.

Por ejemplo, toda variable BC definida en un objeto está en modo Insert, y lo mismo ocurre cuando se le destina nuevo espacio de memoria con new().

Si la variable se carga con Load() pasa inmediatamente a estar en modo Update.
Después de realizada alguna operación sobre ella, si es exitosa, estará en modo Update si se insertó o modificó, y en modo Delete si se le aplicó el método Delete().

Entonces, para saber qué operación se intentó realizar al encontrarse este Save, debemos conocer el contexto de esta variable. En cambio con los

métodos Insert y Update eso no es necesario. Independientemente del modo querrá insertar o actualizar la información en la base de datos.

More about the differences between methods and...

Para interiorizarnos un poco más en las diferencias entre los distintos métodos incluyendo cómo trabajar con Business Components cargados a partir de Data Providers, lo invitamos a ver el video correspondiente.

training.genexus.com
wiki.genexus.com
training.genexus.com/certifications