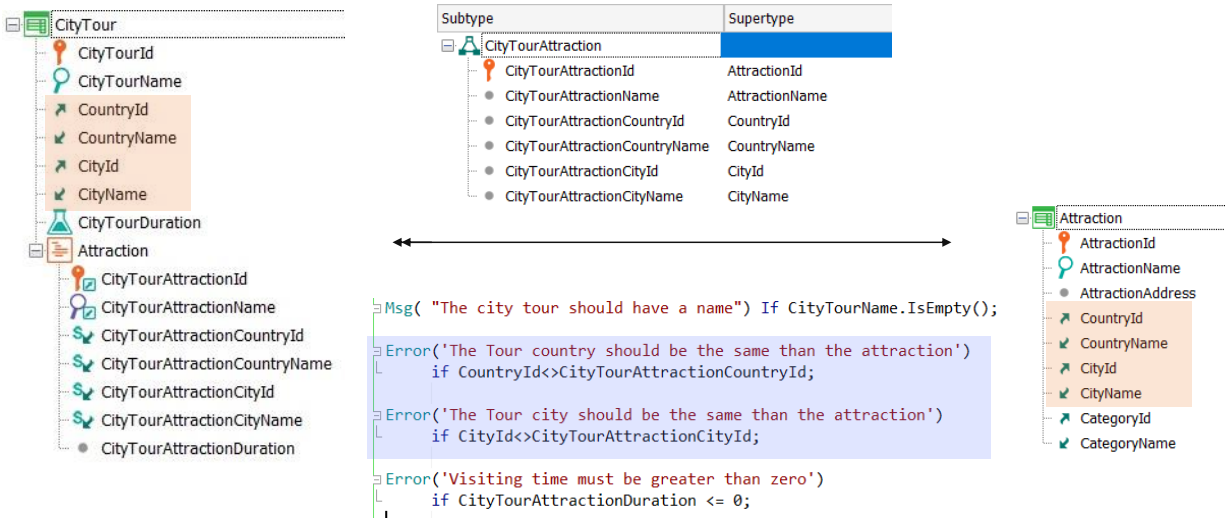
Actualización de la base de datos

Usando Business Component de dos niveles

GeneXus

Two-level BC



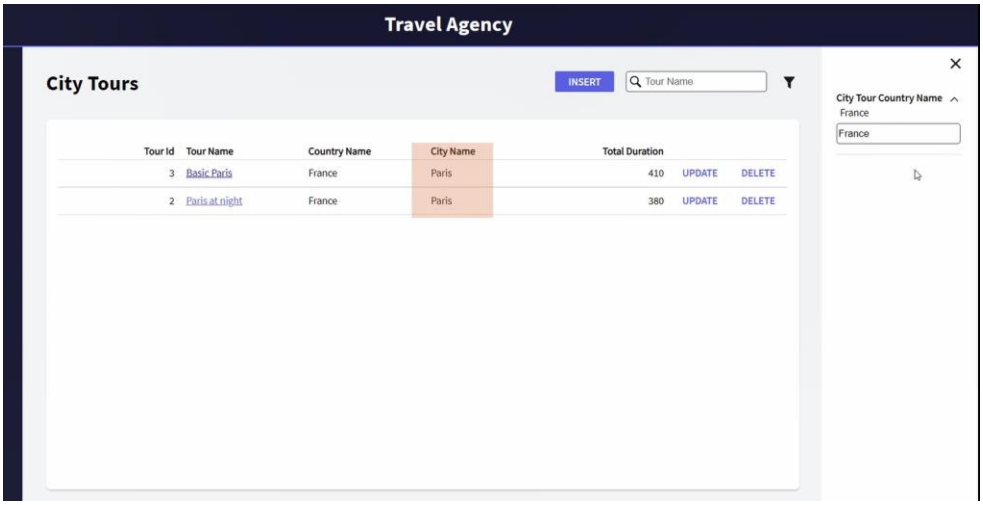
Supongamos que hemos creado una transacción CityTour para representar los tours que se ofrecen a los clientes de la agencia de viajes para visitar las diferentes atracciones turísticas de una ciudad determinada.

Se trata de una transacción de dos niveles: en el primero además del nombre del tour, se especifica su país y ciudad, y el tiempo estimado de duración del tour completo, que es la suma de los tiempos estimados de visita de cada atracción.

El subnivel indica las atracciones turísticas que incluye el tour, para las que hubo que definir un grupo de subtipos, ya que necesitamos especificar en el primer nivel el país y ciudad del city tour y las atracciones también tienen un país y ciudad, que deberemos controlar que coincidan con los del tour.
Observemos que CityTourAttracionId es subtipo de AttractionId, por lo que es como si se tratara del mismo atributo, y por ello será clave foránea en la tabla del segundo nivel de CityTour.

Además, esa tabla contará con el atributo secundario CityTourAttractionDuration, para especificar cuánto tiempo se estima que durará la visita a esa atracción.

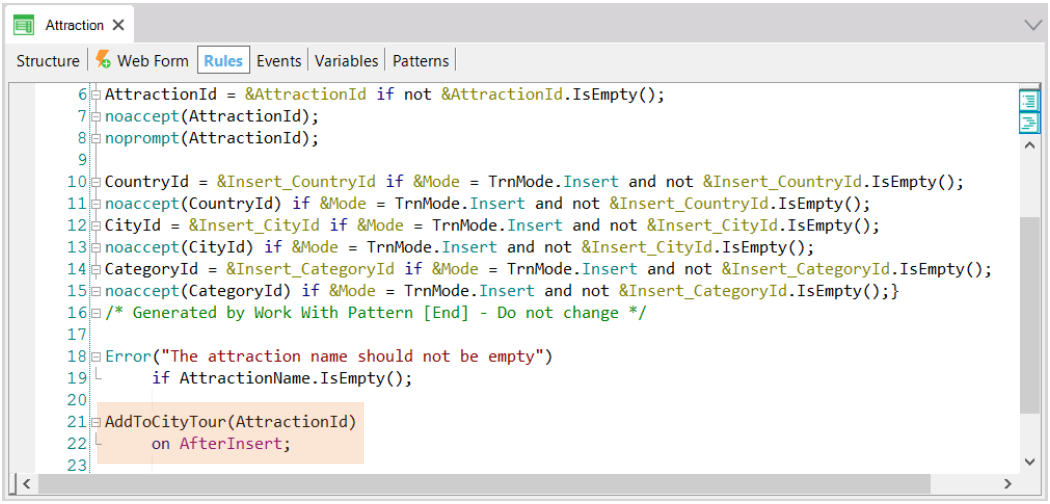
Insert many lines



Hemos aplicado el pattern work with a CityTour y también a las atracciones, para trabajar más cómodos. Y hemos cargado algunos datos.
Por ejemplo veamos que para París tenemos creados solo dos city tours. En el primero tenemos al Louvre y a la torre Eiffel, y en el segundo únicamente a la torre Eiffel.

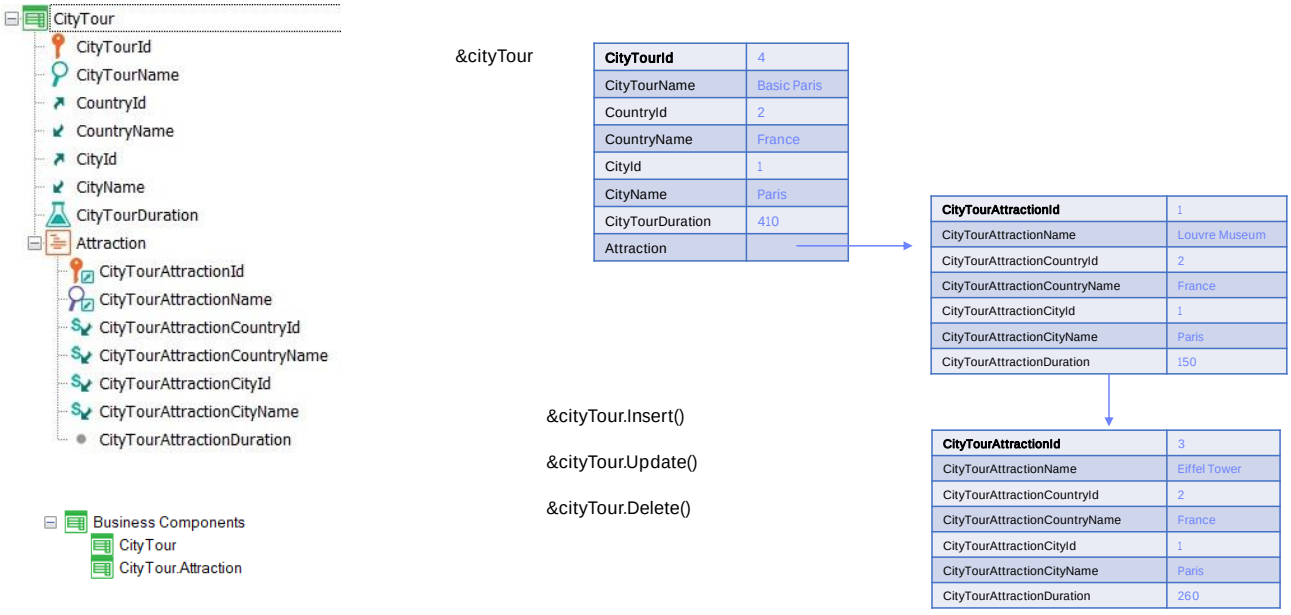
Supongamos que se desea que cuando el usuario del backoffice esté trabajando con las atracciones turísticas para ingresar una nueva al sistema, -por ejemplo la catedral de Notre Dame- automáticamente ésta se agregue a todos los city tours correspondientes a la ciudad de la atracción –que en este caso es París-, con una duración fija para la visita de 120 minutos, que luego podrá modificarse. ¿Cómo conseguimos este comportamiento?

Insert many lines



En definitiva, ni bien se inserta una nueva atracción a través de esta transacción, necesitamos que automáticamente para todos los city tours que tengan su mismo país y ciudad, se inserte una línea nueva con esa atracción, de tiempo 120 minutos.

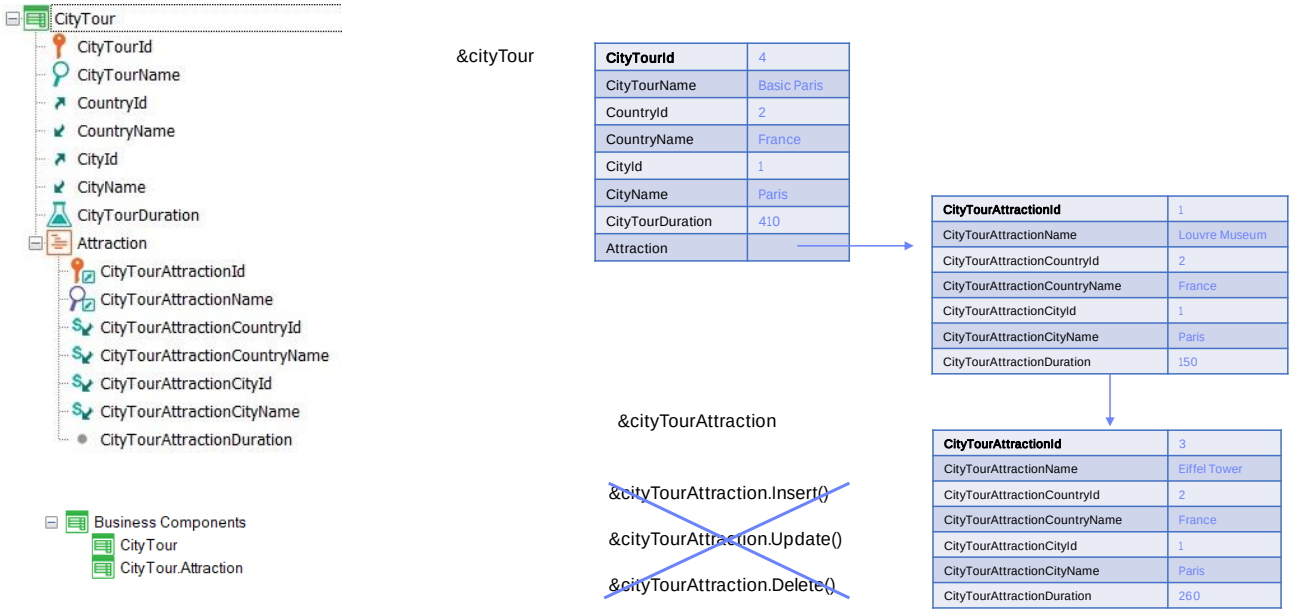
Entonces llamaremos en AfterInsert, es decir, inmediatamente después de que se haya insertado en la tabla Attraction la nueva atracción, a un procedimiento al que le pasaremos el identificador de esa atracción ingresada, y que será el encargado de insertar la línea para los city tours que encuentre con el mismo país y ciudad.



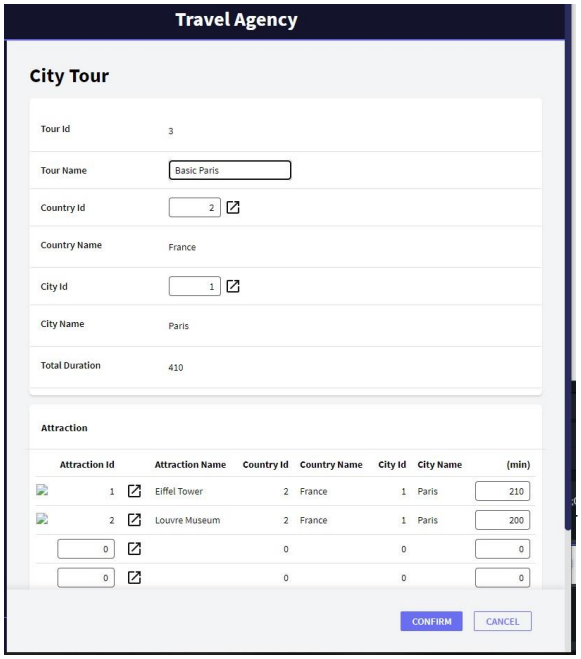
Al prender la propiedad Business Component de la transacción se crearán automáticamente en la KB dos business components y no solo uno, como podría pensarse.

El primero es el esperable, con una estructura como la que mostramos. Así, si el city tour 4 tiene dos líneas, la estructura de una variable de tipo de datos el business component CityTour será como la que aquí representamos, donde el último elemento, Attraction, es una colección de líneas. Cada línea corresponderá, a su vez, a un dato estructurado. ¿Con qué estructura? La del business component CityTour.Attraction, que es la de cada línea de la transacción.

Es decir, mientras que el business component correspondiente a la transacción como un todo es el que nos permitirá realizar las operaciones de Insert, Update y Delete sobre la base de datos, el que se crea para las líneas es solamente para ser utilizado como estructura de datos.



No permitirá esas otras operaciones. Esto puede parecer confuso a primera vista.



Pero así como cuando queremos insertar una nueva línea a través de la pantalla de la transacción CityTour debemos primero instanciar el cabezal, luego insertar la línea y finalmente presionar el Confirm global, lo mismo sucederá con los business components. Para trabajar con sus líneas, por ejemplo agregando una, habrá que trabajar con el cabezal y sus líneas, agregando la línea y luego haciendo la operación deseada sobre el todo, es decir, sobre la variable BC de la transacción.

Insert a line


```
&cityTour.Load(2)

&attraction = new()

&attraction.CityTourAttractionId = 5
&attraction.CityTourAttractionDuration = 120

&cityTour.Attraction.add(&attraction)

&cityTour.Update()
```

CityTourId	2
CityTourName	Paris at night
CountryId	2
CountryName	France
CityId	1
CityName	Paris
CityTourDuration	380
Attraction	

CityTourAttractionId	3
CityTourAttractionName	Eiffel Tower
CityTourAttractionCountryId	2
CityTourAttractionCountryName	France
CityTourAttractionCityId	1
CityTourAttractionCityName	Paris
CityTourAttractionDuration	260

CityTourAttractionId	5
CityTourAttractionName	Notre Dame
CityTourAttractionCountryId	2
CityTourAttractionCountryName	France
CityTourAttractionCityId	1
CityTourAttractionCityName	Paris
CityTourAttractionDuration	120

Entonces, si queremos insertar una nueva atracción para el city tour 2, necesitaremos una variable &cityTour del tipo de datos el Business component de la transacción CityTour. Le aplicamos la operación Load, tras lo cual obtenemos la variable cargada con los datos de la base de datos: en este caso el cabezal y su única línea.
Luego, como necesitaremos agregar una nueva línea, definimos una variable &attraction del tipo de datos el business component correspondiente a las líneas. Cargamos los valores de sus elementos y solo nos resta agregar esta estructura a la colección de líneas. Esto lo hacemos con el método add, aplicado ni más ni menos que a la colección Attraction.

Una vez que tenemos la información como la queremos en la variable &cityTour, realizamos sobre ella la operación de Update, porque el cityTour de Id 2 ya existía, solamente queremos modificarlo agregándole la línea. Esta operación será la que insertará realmente la línea en la base de datos.

Insert many lines

```
parm(in:&AttracionId);
```

AddToCityTour

Source | Layout | Rules | Conditions | Variables

Subroutines

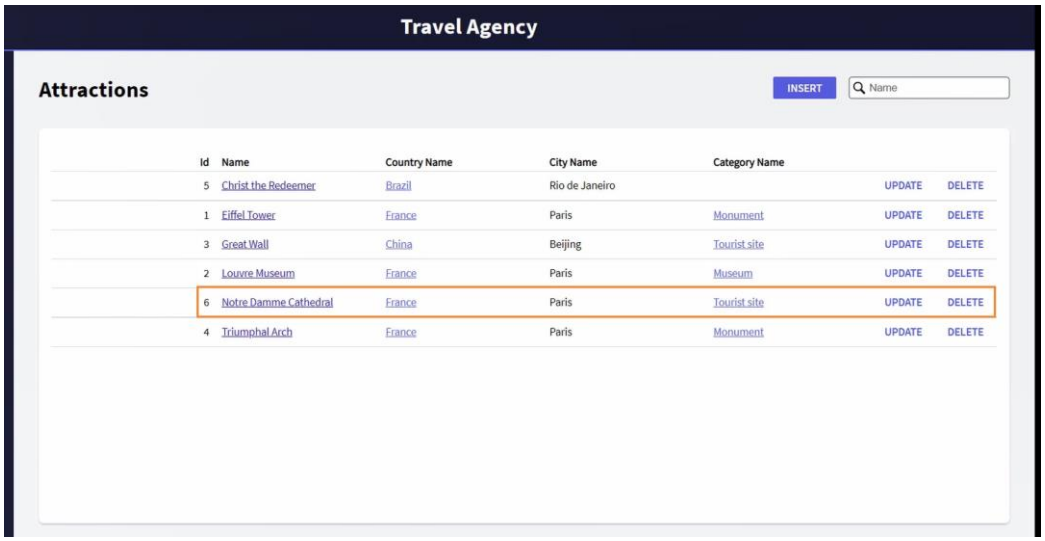
```
1  /*The attraction received by parameter will be inserted in all the City tours with the same
2    country and city of the attraction */
3
4  for each CityTour
5        where CountryId = find(CountryId, AttractionId = &AttracionId)
6        where CityId = find(CityId, AttractionId = &AttracionId)
7
8        &cityTour.Load(CityTourId)
9
10       &attraction = new()
11       &attraction.CityTourAttractionId = &AttracionId
12       &attraction.CityTourAttractionDuration = 120
13
14       //Add the attraction to the City Tour lines
15       &cityTour.Attraction.Add(&attraction)
16
17       If &cityTour.Update()
18            commit
19       endif
20 endfor
21
```

Si vamos a ver el procedimiento que implementamos, ya entenderemos todo.
Recibimos al id de atracción en la variable &AttractionId.

Estamos iterando en el for each por todos los city tours del país y ciudad de la atracción recibida –aquí queda claro por qué tenemos que llamar al procedimiento luego de que se haya insertado la atracción en la tabla Attraction; de lo contrario no encontrará su país y ciudad para hacer este filtro–.

&cityTour es una variable de tipo de datos el Business component CityTour. La cargamos a partir del id de city tour encontrado en el for each. Solo tenemos que agregarle la línea con la atracción.
Para ello definimos una variable del tipo de datos el business component correspondiente a las líneas. Creamos nuevo espacio de memoria para esta variable, le asignamos los elementos almacenables: esto es el id de atracción y la duración de la visita.
Y luego solo nos resta agregar esa línea a la colección de líneas del CityTour.
Por último realizamos la operación de Update que devolverá True si fue exitosa y en ese caso commiteamos.
Luego se pasa a la siguiente iteración.

Delete many lines

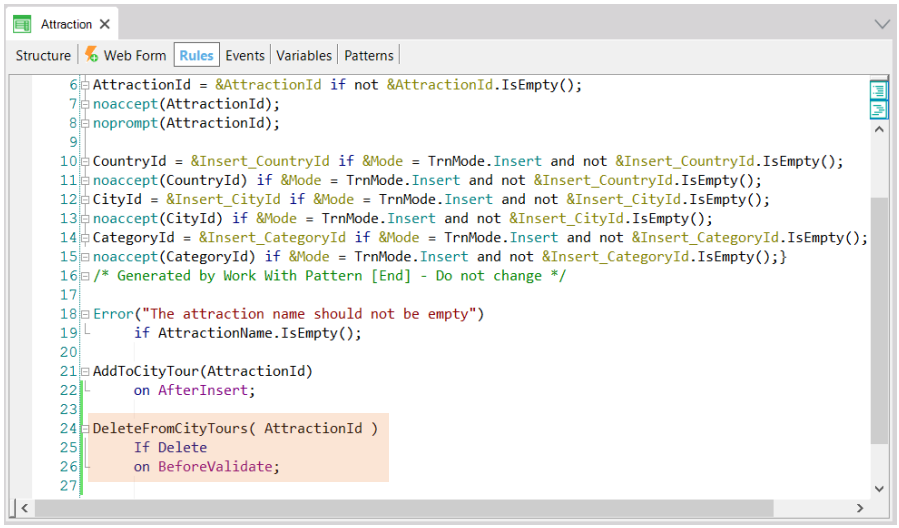


Ahora vamos a estudiar cómo eliminar líneas a través del Buisiness Component.

Veamos que tenemos en el Work with Attractions la catedral de Notre Dame que habíamos insertado previamente. Si vamos a los city tours, vemos que hay dos de París que la contienen: este y este otro. Lo que queremos, entonces, ahora, es que se nos permita eliminar la atracción –normalmente esto no nos sería permitido debido a que se va a chequear la integridad referencial, por tanto, que no existan city tours que contengan a esta atracción, para permitir su eliminación–. Por lo que vamos a querer que al presionar Delete primero se eliminen todas aquellas líneas de city tours en las que se encuentre la atracción, para luego pasar a eliminar, sí, la atracción.

Confirmo, y ahora vemos que se eliminó de este city tour, y también del que habíamos visto antes. La pregunta es cómo lo implementamos.

Delete many lines

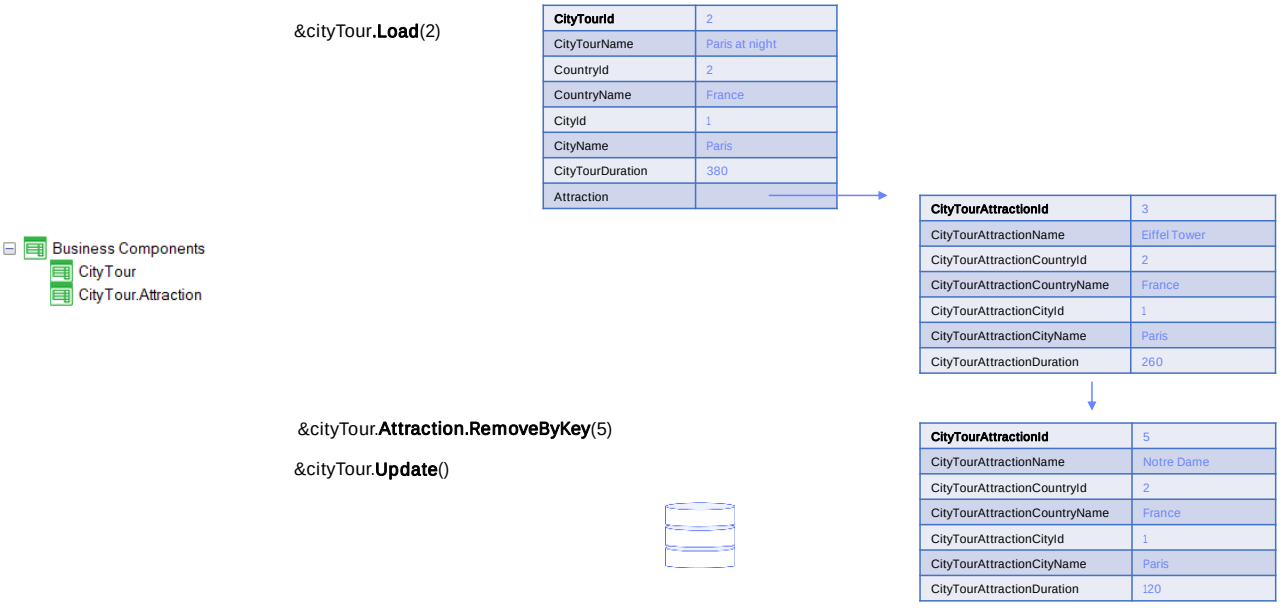


Desde la transacción Attraction, cuando estamos en modo Delete, es decir, estamos queriendo eliminar la atracción y antes de que se validen los datos, es decir, antes de que se realicen los chequeos de integridad referencial, llamamos a un procedimiento, pasándole el identificador de atracción; y ese procedimiento es el que se va a tener que encargar de eliminar todas las líneas de city tours que la contengan.

¿Cómo hacemos esto?

Cuando queremos eliminar una línea utilizando la transacción primero tenemos que cargar el city tour, que quedará en modo Update, luego eliminar la línea y finalmente presionar el Confirm para que esa eliminación se lleve realmente a cabo en la base de datos. Aquí será análogo.

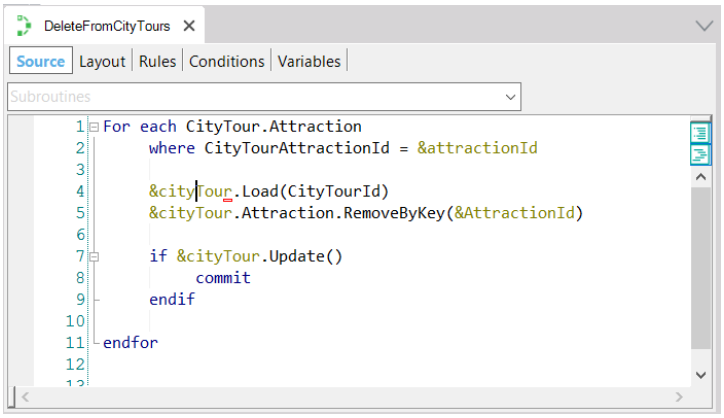
Delete a line

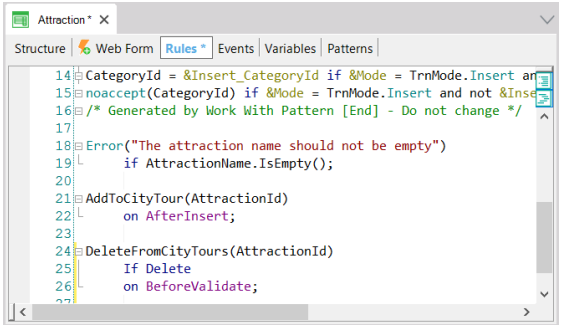


Si queremos eliminar la Catedral de Notre Dame del city tour 2, en una variable del tipo de datos el Business Component CityTour cargamos la estructura utilizando el método Load, y luego eliminamos la línea de la colección Attraction. ¿Cómo? Con el método RemoveByKey. Como lo indica su nombre, este método va a buscar en la colección aquel ítem que corresponda al valor de clave indicado. En nuestro caso el de Id 5 que es de la Catedral de Notre Dame. Esta eliminación se ha realizado únicamente en memoria. Ahora nos falta impactarla en la base de datos. Para ello tenemos que actualizar el city tour, y por ello utilizamos el método Update. Por supuesto, también podríamos haber utilizado el Save(). Todo esto es equivalente a haber presionado el botón de Confirm en la transacción.

Delete many lines

```
parm( in: &AttractionId );
```

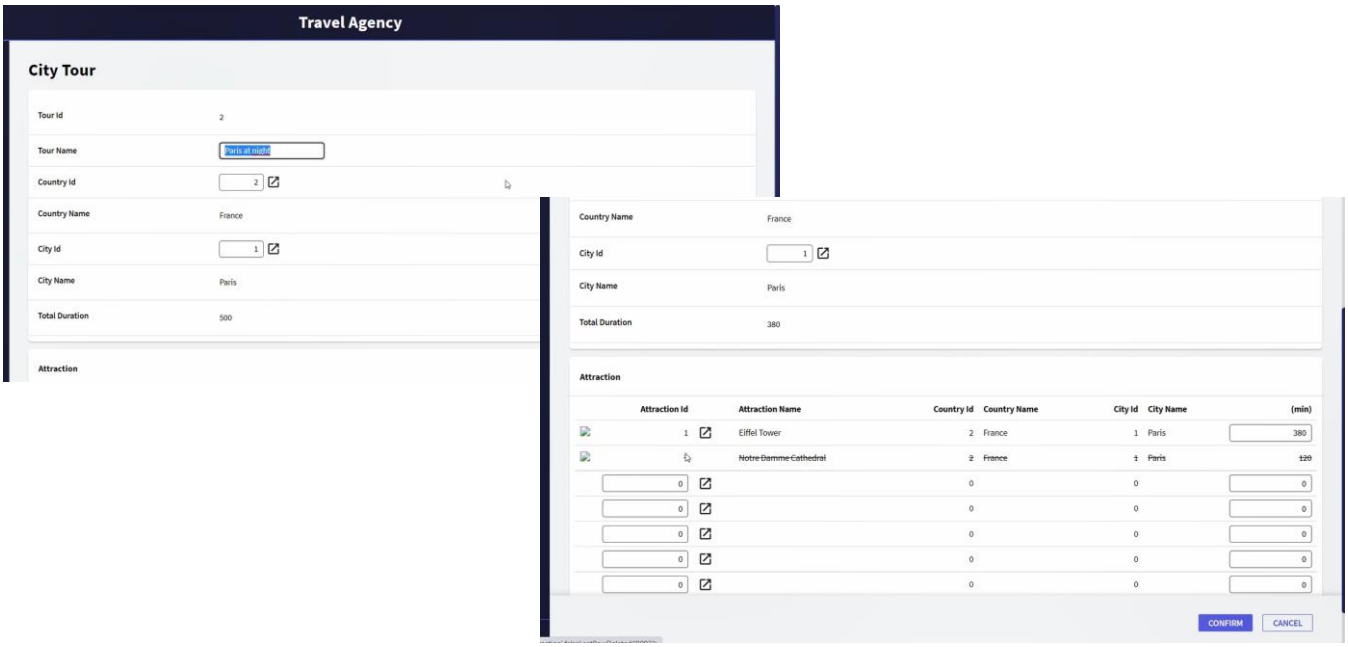




Si ahora vamos a ver cómo implementamos nuestro procedimiento en GeneXus: vemos que recibimos en la variable &AttractionId, el identificador de atracción que va a ser eliminado. Hacemos un for each buscando en las líneas de city tour si existe alguna que corresponda a ese attraction id. En caso de encontrarla, entonces en una variable city tour, del tipo de datos el Business Component correspondiente a la transacción cargamos el CityTourId que corresponda a esa línea, y luego lo que hacemos es aplicarle el método RemoveByKey a la colección de atracciones de ese city tour para eliminar la de identificador attraction id recibido por parámetro.
Luego hacemos el Update; si fue exitoso commiteamos.

Con esto estamos seguros de que habremos eliminado todas las líneas de city tours que tuvieran a esa atracción entre sus datos. Entonces cuando este procedimiento termina su ejecución retorna el control aquí, se hace la validación de la atracción correspondiente que ahora no va a tener información relacionada y se pasa a eliminar sin dar ningún tipo de falla de integridad.

Update a line

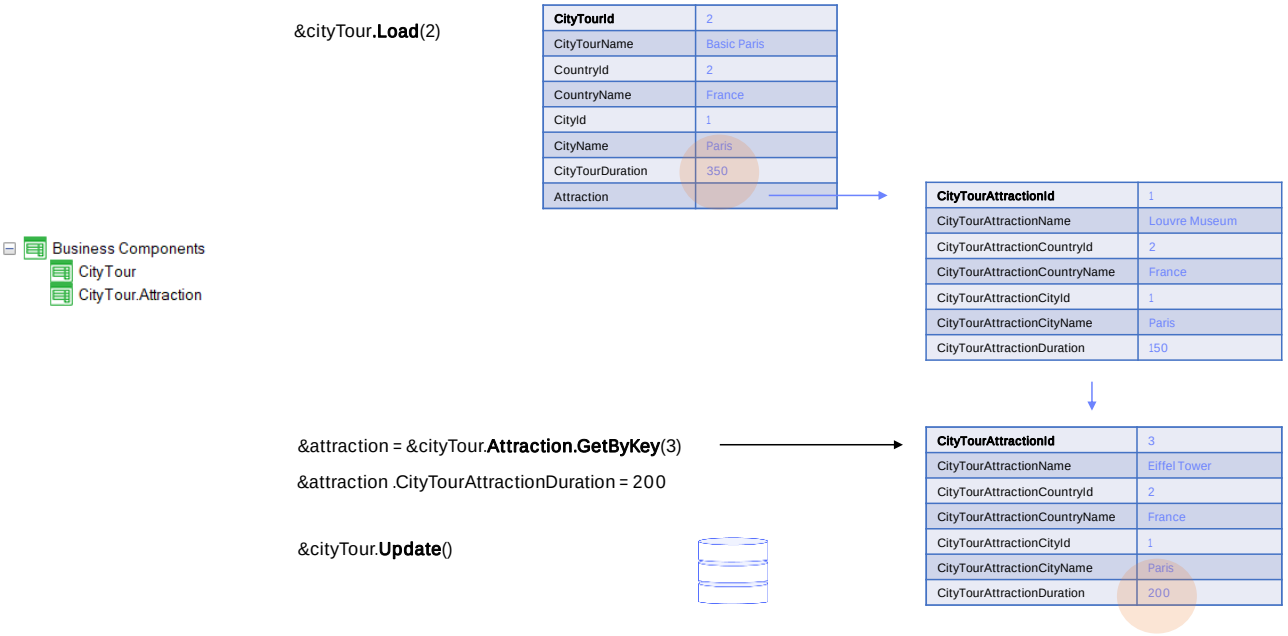


Nos queda por estudiar el último caso, que es cómo hacemos para actualizar líneas de una transacción pero a través del Business component.

Si lo hiciéramos a través de la transacción editaríamos, en este caso el TourId 2. Supongamos que nos interesa modificar la duración de la visita a la torre Eiffel, pasando de 260 minutos a 200, y una vez que hacemos esto lo que nos resta es confirmar, para que la actualización se lleve a cabo.

Algo así vamos a tener que hacer a través del business compontent: cargarlo, modificar la línea y actualizar.

Update a line



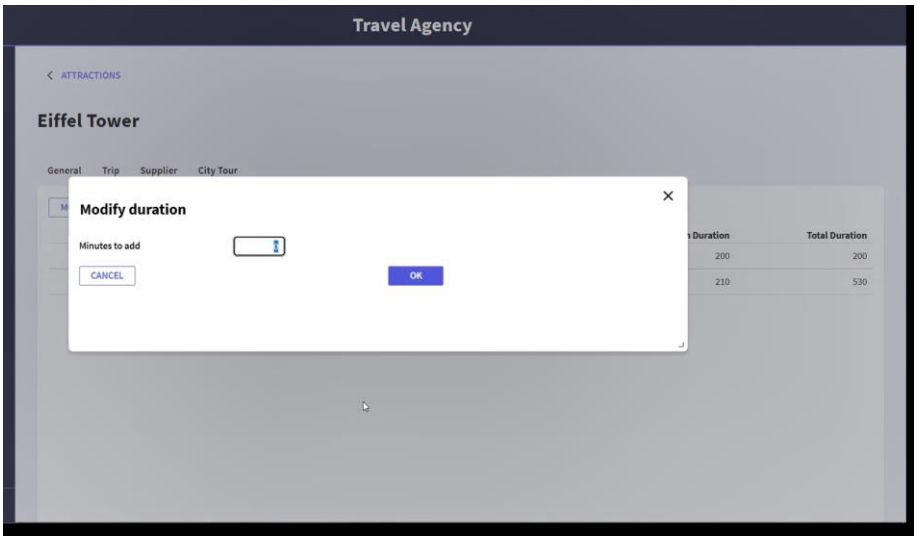
Cargamos entonces la variable &cityTour con el city tour de id 2.

Debemos acceder al ítem de la colección que corresponde a la atracción 3. Para ello, contamos con el método GetByKey, que aplicado a la colección Attraction, esto es, a la de las líneas de la transacción, nos devolverá una referencia directa a esa posición de memoria. Por tanto se la asignamos a la variable &attraction del tipo de datos el business component correspondiente a las líneas. De esta manera la variable &attraction no será una copia de esa línea, sino que será exactamente esa línea.

Ahora sólo necesitamos modificar el elemento CityTourAttractionDuration de esa variable y esto repercutirá directamente en la variable &cityTour.

Nos falta impactar esto en la base de datos, y para ello pedimos que se realice la operación de Update del Business Component. Observemos que como consecuencia, además de actualizase en la base de datos el registro correspondiente a esa línea, la variable &cityTour también es actualizada, y el elemento CityTourDuration que correspondía a una fórmula en el cabezal de la transacción que sumaba los tiempos de visita de las líneas, también es refrescado en la variable.

Update many lines



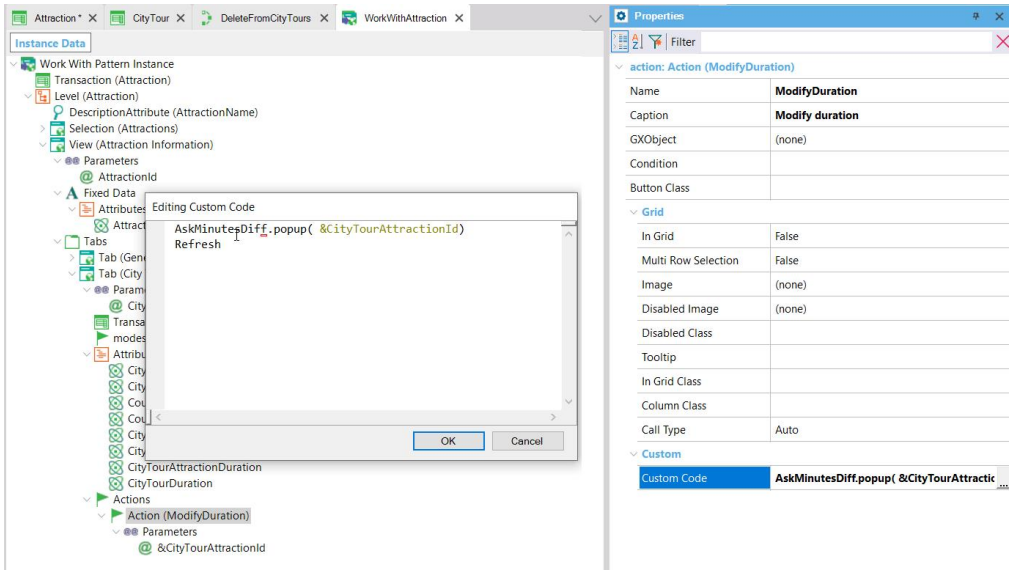
Veamos ahora cómo pusimos en práctica la modificación de líneas a través de business component en nuestra aplicación.

Estamos posicionados en el Work with de Attractions. Si editamos una atracción determinada, por ejemplo la torre Eiffel, y vamos a la solapa que nos muestra todos los city tours en los cuales esa atracción se encuentra, vemos que nos está mostrando, entre otra información, la duración en minutos de la visita a esa atracción en cada uno de los city tours.

Queríamos ofrecer desde aquí la posibilidad de modificar esa duración, sumándole o restándole minutos en todos los city tours en los que se encuentre. Así, supongamos que queremos agregar 10 minutos en cada uno. Presentamos esta pantalla al usuario donde indica que quiere sumarle 10 minutos y vemos que efectivamente se ha realizado esa adición.

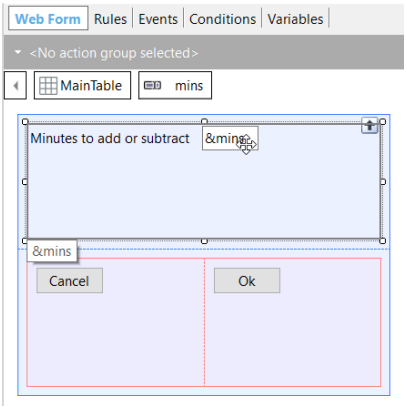
¿Qué es lo que se hizo por detrás? Se accedió a cada uno de esos dos city tours, se empezó por el primero, se accedió a la línea correspondiente a esta atracción, y se le sumaron 10 minutos al tiempo estimado. Y se hizo lo mismo para esta otra. Es decir, hubo que modificar una línea de cada uno de estos dos city tours.

Update many lines

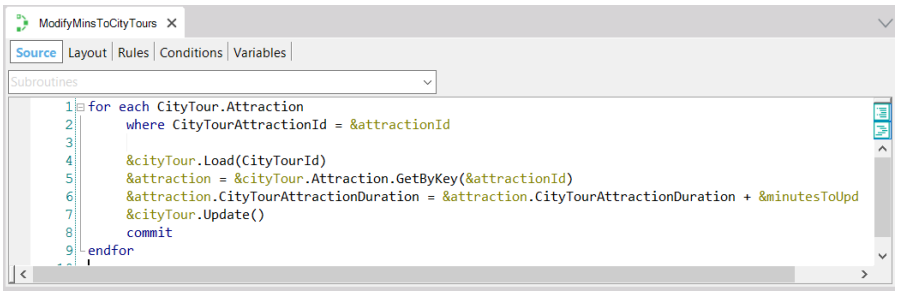


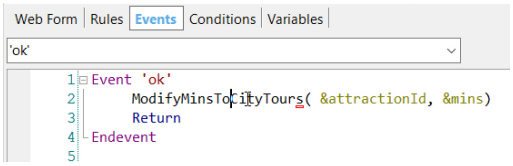
Si vamos a ver cómo lo implementamos en GeneXus, vemos que en el Work with Attraction agregamos para el tab correspondiente a los city tours de la atracción, una acción, ModifyDuration, cuyo código es la invocación al web panel de este nombre, pasándole el identificador de atracción, y cuando ese web panel se termina de ejecutar se hace un refresh para mostrar en el grid que veíamos los datos refrescados.

Update many lines




```
parm( in: &attractionId, in: &minutesToUpd);
```





Vamos a ver ese web panel, que es el que pide al usuario en esta variable la modificación en minutos y lo que hace es invocar a un procedimiento que es el que efectivamente hace la modificación en los datos que correspondan. El procedimiento necesita recibir el id de atracción y los minutos que queremos sumar o restar a esa atracción en los distintos city tours en los que se encuentre.

Entonces vamos a visualizar cómo programamos ese procedimiento.

Vemos que está recibiendo en estas dos variables la información, y lo que hace es: en un for each recorre las atracciones de los city tours, filtrando por la atracción recibida por parámetro. Entonces, para cada uno de los registros que encuentre lo que hace es cargar en la variable business component &cityTour el city tour del que se trate; se obtiene en la variable &attraction del tipo CityTour.Attraction –es decir, el business component correspondiente a las líneas- el ítem de la colección Attraction que corresponde a la clave &AtrractionId. Luego lo que se hace es modificar la duración: el elemento CityTourAttractionDuration para ese ítem, sumándole al valor que tenía, la cantidad en minutos recibida por parámetro. Y luego simplemente se hace un Update, commiteando.

Summary

Insert a line

```
&BC.Load(PKAttribute)

&lineBC = new()

&lineBC.PKLineAtt = …
&lineBC.LineAtt2 = …
…
&lineBC.LineAttn = …

&BC.Lines.Add(&lineBC)

&BC.Update()
```

Delete a line

```
&BC.Load(PKAttribute)

&BC.Lines.RemoveByKey(PKLineAtt)

&BC.Update()
```

Update a line

```
&BC.Load(PKAttribute)

&lineBC = &BC.Lines.GetByKey(PKLineAtt)
&lineBC.LineAtt2 = …
&lineBC.LineAttn = …

&BC.Update()
```

Aquí vemos un resumen de lo que hemos visto relativo a cómo insertar, eliminar y modificar una línea en un business component de dos niveles.

En todos los casos la operación es Update, dado que supusimos que el cabezal ya existía.

Siempre, entonces, como primera cosa cargamos la variable business component sobre la que queremos trabajar. Para el caso de la inserción de una línea se crea una variable de tipo el business component de las líneas; se reserva nuevo espacio de memoria; se le asigna valor a todos los elementos de la línea que correspondan a atributos de la tabla a los que les queramos dar valor –los que no asignemos quedarán nulos o vacíos– y luego se agrega la línea a la colección de líneas. Este agregado también es por referencia, de modo que debemos tener mucho cuidado de hacer el new() para crear nueva memoria antes de trabajar con cada línea.

Para el caso de la eliminación se cuenta con el método RemoveByKey, que recibe por parámetro el identificador de la línea que desea eliminarse.

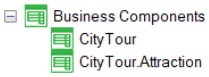
Y por último, para poder modificar algún atributo o atributos de una línea, contamos con el método GetByKey de la colección de líneas, que también recibe por parámetro de id de la línea que se quiere obtener. Este método devuelve una referencia al business component correspondiente a esa línea, por lo que debemos asignar el método a una variable de ese tipo. Luego, al modificar la variable ya estaremos modificando ese ítem de la colección.

Insert a new header + lines

```
&cityTour = new()

&cityTour.CityTourId = 2
&cityTour.CityTourName = 'Paris at night'
&cityTour. CountryId = 2
&cityTour.CityId = 1
```

CityTourId	2
CityTourName	Paris at night
CountryId	2
CountryName	France
CityId	1
CityName	Paris
CityTourDuration	380
Attraction	




```
&attraction = new()
&attraction.CityTourAttractionId = find( AttractionId, AttractionName = "Eiffel Tower")
&attraction.CityTourAttractionDuration = 260

&cityTour.Attraction.add(&attraction)
```

CityTourAttractionId	3
CityTourAttractionName	Eiffel Tower
CityTourAttractionCountryId	2
CityTourAttractionCountryName	France
CityTourAttractionCityId	1
CityTourAttractionCityName	Paris
CityTourAttractionDuration	260

```
&attraction = new()
&attraction.CityTourAttractionId = find( AttractionId, AttractionName = "Notre Dame")
&attraction.CityTourAttractionDuration = 120

&cityTour.Attraction.add(&attraction)

&cityTour.Insert()
```

CityTourAttractionId	5
CityTourAttractionName	Notre Dame
CityTourAttractionCountryId	2
CityTourAttractionCountryName	France
CityTourAttractionCityId	1
CityTourAttractionCityName	Paris
CityTourAttractionDuration	120

Si queremos insertar un nuevo cabezal y sus líneas, por ejemplo, el city tour 2, asumiendo que no existía, empezamos por definir la variable BC y asignarle memoria nueva.

Luego asignamos valor para todos los elementos que corresponden a atributos del cabezal no inferidos ni fórmulas.

Luego creamos un nuevo espacio de memoria para una variable &attraction del tipo el business component de las líneas. Le asignamos los valores para los atributos no inferidos y agregamos esa variable a la colección de las líneas del business component global.

Hacemos lo mismo para ingresar una segunda línea. Observemos que aquí sí es obligatorio reservar nuevo espacio de memoria, porque de lo contrario, al hacer las asignaciones estaremos sobrescribiendo la línea anterior.
Agregamos la nueva variable a la colección de Attractions del BC y por último ejecutamos la operación de Insert, para que se inserte cabezal y las dos líneas en la base de datos.

Si no hubo ningún error, la variable quedará cargada con todos los datos, los que ingresamos y los inferidos y fórmula.

GeneXus™

training.genexus.com
wiki.genexus.com