Actualización de base de datos

Business Components: diferencias entre métodos

GeneXus™

Methods to operate on the database

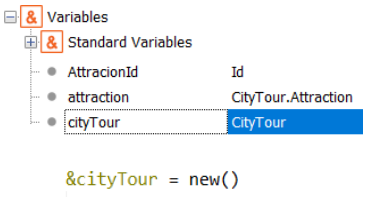
- Load(), Update(), Delete(), Insert()
- Save(), Asignation instead of Load, InsertOrUpdate()

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

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

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

TrnMode.Insert



TrnMode.Udpate

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

TrnMode.Delete

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

Retomemos el final del video donde comparábamos los business components de 1 y 2 niveles y continuemos.

"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().

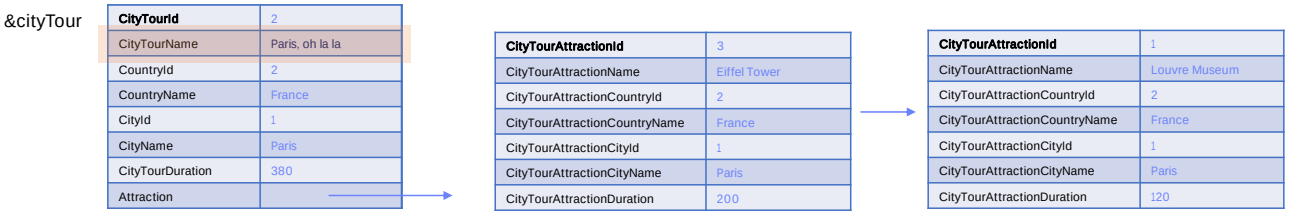
Si la variable se carga con Load() pasa inmediatamente a estar en modo Update.
Después de realizada alguna operación sobre ella, si es exitosa, estará en modo Update si se insertó o modificó, y en modo Delete si se le aplicó el método Delete().

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.”

Methods to operate on the database

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

```
&cityTour.Load(2)
&cityTour.CityTourName = "Paris, oh la la"
...
&cityTour.Save()
```



Con este ejemplo quedará bastante claro.

Observemos que tras realizar el Load, si existe un city tour 2 la variable cargará la información de la base de datos y quedará en modo Update, por lo que el Save querrá actualizar. Actualizará únicamente los elementos que difieran, en este caso, únicamente el CityTourName.

Methods to operate on the database

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

```
&cityTour.Load(2)
&cityTour.CityTourName = "Paris, oh la la"
...
&cityTour.Save()
```

```
&cityTour.CityTourId = 2
&cityTour.CityTourName = "Paris, oh la la"
...
&cityTour.Save()
```

```
&cityTour.CityTourId = 2
&cityTour.CityTourName = "Paris, o
...
&cityTour.Update()
```

&cityTour

CityTourId	2
CityTourName	Paris, oh la la
CountryId	
CountryName	
CityId	
CityName	
CityTourDuration	
Attraction	

&cityTour_aux

CityTourId	2
CityTourName	Paris, oh la la
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	200

CityTourAttractionId	1
CityTourAttractionName	Louvre Museum
CityTourAttractionCountryId	2
CityTourAttractionCountryName	France
CityTourAttractionCityId	1
CityTourAttractionCityName	Paris
CityTourAttractionDuration	120

¿Pero qué pasaría si en lugar de utilizar el método Load, directamente asignásemos valor a la clave primaria?
Va a depender del modo en el que se encontraba la variable. Si no estaba en modo Update, por ejemplo porque solamente se la definió en el objeto y esto es lo primero que se hace con ella, entonces el Save intentará insertar un city tour 2, lo que fallará por clave primaria duplicada.

En cambio si se utiliza el método Update en lugar del Save, no habrá problema. A primera vista podría parecer que sí, porque la variable &cityTour tendrá solamente dos elementos no vacíos. Ni siquiera están allí las dos líneas que sabemos tiene el cityTour 2. Podemos entonces pensar que esto hará un desastre en la base de datos al ejecutar el Update.
Sin embargo, el Update es inteligente, y sabe que como la variable está en modo Insert, eso significa que no tiene cargados todos los datos (porque de haber ejecutado un Load la variable estaría en modo Update y no Insert) entonces solamente debe modificar de la base de datos los datos efectivamente asignados en el BC, y ningún otro.
Internamente carga en una variable BC auxiliar los datos, que por supuesto, queda en Update; les modifica los que se han modificado en la variable que está en modo Insert, en este caso solo el CityTourName, y hace un Save(), que como la variable auxiliar está en Update por haber sido cargada... actualiza.

En general no es buena práctica trabajar de este modo, asignando directamente la clave primaria de la variable BC sin realizar un Load

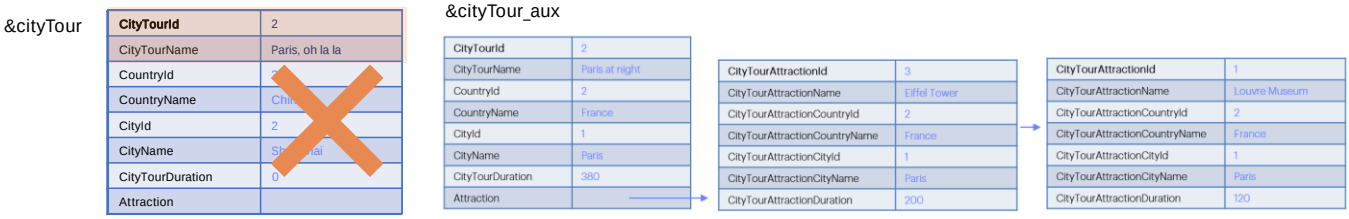
Methods to operate on the database

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

```
&cityTour.Load(2)
&cityTour.CityTourName = "Paris, oh la la
...
&cityTour.Save()
```

```
&cityTour.CityTourId = 2
&cityTour.CityTourName = "Paris, oh la la
...
&cityTour.Save()
```

```
&cityTour.Load(5)
&cityTour.CityTourId = 2
&cityTour.CityTourName = "Paris, o
...
&cityTour.Update()
```



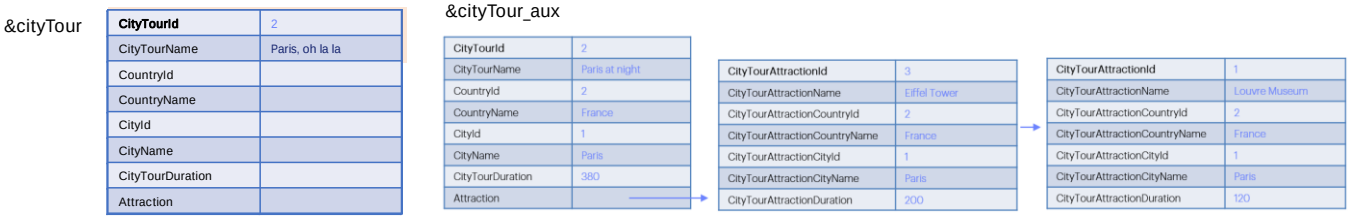
, puesto que si la variable se encontraba en modo Update porque, por ejemplo, se la había utilizado antes (imaginemos que se la había cargado con el city tour 5 que solo tiene cabezal), entonces, si no somos cuidadosos, podemos dejarle valores basura que provienen de ese antes.

La asignación es la única manera cuando el business component se carga en un Data Provider, como veremos a continuación, aprovechando para estudiar el método InsertOrUpdate.

Methods to operate on the database

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

```
&cityTour = new()
&cityTour.CityTourId = 2
&cityTour.CityTourName = "Paris, c
...
&cityTour.Update()
```



La fuerte recomendación es que si se va a utilizar la asignación, entonces antes se pida nuevo espacio de memoria para la variable, de modo que esté limpia y en Insert.

La asignación es la única manera cuando el business component se carga en un Data Provider, como veremos a continuación, aprovechando para estudiar el método InsertOrUpdate.

Methods to operate on the database

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

Insert()
Update()
Delete()
InsertOrUpdate()

FOR COLLECTION OF BCs TOO!

```
Parm( in: &cityTour )

If &cityTour.InsertOrUpdate()
   Commit
else
   Rollback
endif
```

```
If not &cityTour.Insert()
    for &message in &cityTour.GetMessages()
  if &message.Id = "DuplicatePrimaryKey"
      &cityTour.Update()
  endif
  endfor
endif
If &cityTour.Sucess
   Commit
else
   Rollback
```

Este método se utiliza cuando no sabemos si existe o no en la base de datos la información que estamos manipulando en el BC.

Por ejemplo, cuando en un procedimiento se recibe en una variable business component. Dentro del código no tenemos idea de qué información vendrá allí cargada. Sí sabemos que la variable estará en modo Insert, pues es nueva en este objeto, así sea parámetro. No recibe el modo en el que estaba en el objeto llamador. Solo recibe el valor de sus elementos.

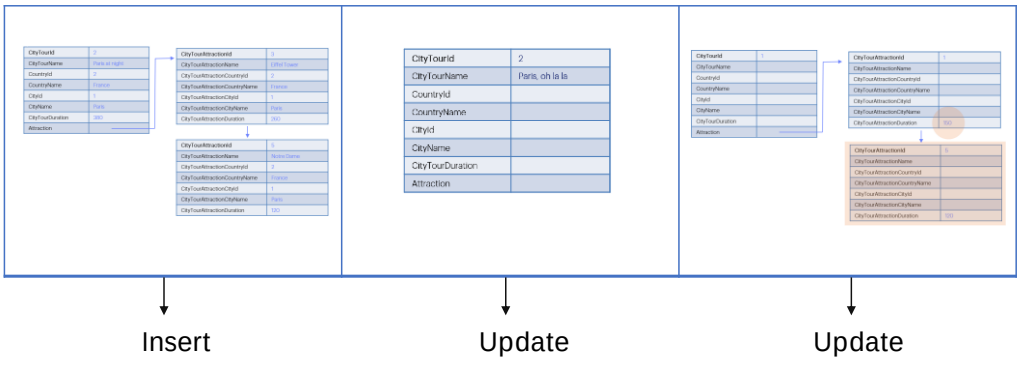
Entonces, si intentáramos hacer un Save(), solo será exitoso si la clave primaria del BC no se encuentra en la base de datos. De lo contrario fallará por clave duplicada (dado que como la variable estará indefectiblemente en modo Insert, lo que intentará el Save es insertar). Por tanto, si el objeto que llamó al procedimiento lo hizo con la intención de que se realizara una actualización, estamos en problemas.

El método InsertOrUpdate nos evita tener que programar nosotros las dos posibilidades. Es decir, intentar insertar, y si eso falla por clave duplicada, y no por otra cosa, entonces intentar el Update. Esto es ni más ni menos lo que hace el

InsertOrUpdate().

Recordemos, además, que tanto el Insert, como el Update, el Delete y el InsertOrUpdate pueden utilizarse sobre variables de tipo colección de business components, que es el otro caso donde veremos resultará muy útil este último método.

&cityTours




```
&cityTours = GetCityTours(...)
If &cityTours.InsertOrUpdate()
   Commit
else
   Rollback
endif
```

Data Provider Source?

Aquí tenemos una variable de tipo colección de business components, que podría estar cargada en base a lo que devuelve un Data Provider, o estar cargada por código, en este mismo programa. En este caso, el Data Provider devolverá una colección de ítems de tipo el Business Component CityTour- y querremos impactar en la base de datos lo realizado en cada business component de la colección.

Puede que el primer ítem de la colección corresponda a información a ser insertada (en este caso un cabezal y sus dos líneas); en el segundo ítem a información a ser modificada (por ejemplo solamente el CityTourName del cabezal); y en el último también a info a ser modificada (en el ejemplo se modifica una línea y se agrega otra).

En este caso la operación no dependerá del modo de cada variable BC, porque ese modo será en todos los casos Insert, ¿ve por qué?
Por este motivo no podemos recorrer la colección e ir ejecutando Save (sólo funcionará bien para el primer elemento, el que se quiere insertar, pero fallará para los demás, ya que intentará insertar y encontrará clave

duplicada).

La operación, por tanto, deberá depender no del modo de la variable, sino de la existencia o no de la clave primaria en la base de datos. Aquí el método adecuado, por tanto, otra vez es el InsertOrUpdate. Al aplicárselo a la colección, se lo aplicará a cada ítem. Para el primero, intentará Insertar y lo logrará. Para el segundo, intentará insertar pero encontrará clave duplicada, por lo que intentará actualizar, y lo logrará, y lo mismo para el último.

Podemos preguntar por el resultado de la operación, si fue exitosa para toda la colección, entonces commiteamos, y de lo contrario, podríamos hacer Rollback si no deseamos que queden algunos registros modificados y otros no.

En base a lo visto aquí, ¿cómo declararía el Data Provider para que cargue los datos como los mostramos?

DP Source

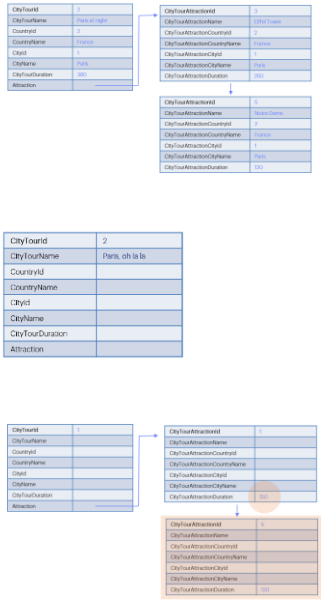
Output

Infer Structure	No
Output	**CityTour**
Collection	**True**
Collection Name	GetCityTours

GetCityTours *

Source * | Rules | Variables

```
1  CityTour
2  {
3       CityTourId = 2
4       CityTourName = Paris at night
5       CountryId = find(CountryId, CountryName = "France")
6       CityId = find(CityId, CityName = "Paris")
7       Attraction
8       {
9            CityTourAttractionId = 3
10           CityTourAttractionDuration = 260
11      }
12      Attraction
13      {
14           CityTourAttractionId = 5
15           CityTourAttractionDuration = 120
16      }
17 }
18 CityTour
19 {
20      CityTourId = 2
21      CityTourName = "Paris, oh la la"
22 }
23 CityTour
24 {
25      CityTourId = 1
26
27      Attraction
28      {
29           CityTourAttractionId = 1
30           CityTourAttractionDuration = 150
31      }
32      Attraction
33      {
34           CityTourAttractionId = 5
35           CityTourAttractionDuration = 120
36      }
37 }
38
```



Aquí lo mostramos, sin entrar en detalles. El Data provider devuelve una colección de tipo de datos el Business component CityTour. ¡Colección!

Estamos suponiendo que CityTourId no es autonumerado, y que el 2 no existe. Observemos que tenemos 3 grupos CityTour, para los tres ítems de la colección que serán devueltos. El primero carga todos los datos de cabezal y las dos líneas.

El segundo supone que la operación de Insert sobre el primero ya se realizó, por lo que para el mismo CityTour, lo que va a intentar es modificar su nombre.

Y el tercero modifica el valor de la línea de Id 1, y agregar la línea de id 5 (aquí no se nota la diferencia, porque los atributos almacenados del nivel Attraction son esos dos únicamente. Pero si la tabla tuviera más atributos, aquí aparecerían todos (o todos aquellos a los que queramos darle valor), mientras que aquí solamente los que queremos modificar.

GeneXus™

training.genexus.com
wiki.genexus.com