

Actualización de base de datos usando Business Components

Cómo actualizar los datos al modo de una transacción, pero por código, sin su pantalla

GeneXus™ 16

Escenario

- Usamos las transacciones para INS, UPD, DLT, a través de su pantalla.

The image shows two screenshots of a 'Category' form. The left screenshot shows the 'Id' field with a cursor and the 'Name' field empty. Below the form, a red arrow points to the 'CONFIRM' button, labeled 'Ins'. The right screenshot shows the 'Id' field with the value '2' and the 'Name' field with the value 'Monument'. Below the form, red arrows point to the 'CONFIRM', 'DELETE', and 'DELETED' buttons, labeled 'Upd' and 'Dlt' respectively.

- Contamos con otra manera de hacerlo: los Business Components (BCs)

Previamente habíamos usado las transacciones para insertar, modificar o eliminar registros de la base de datos, pero siempre a través de su pantalla, con los botones y controles disponibles para ello.

Aquí veremos cómo podemos realizar las mismas operaciones, pero usando comandos escritos en un código, desde cualquier objeto GeneXus, utilizando el concepto de Business Components.

Nuevos requerimientos

- Nueva categoría "Tourist site" para representar atracciones muy visitadas, independientemente de que se trate de monumentos, museos, etc.
- Para todas las atracciones de Beijing de categoría "Monument" cambiarla por "Tourist site"

Veamos esto con un ejemplo. Supongamos que en nuestra agencia de viajes se presenta el caso de que algunas atracciones son tan visitadas por los turistas, que queremos clasificarlas de otra manera para armar excursiones a esos lugares en particular.

Por ejemplo en Beijing, dado la popularidad de sus monumentos, no queremos categorizar más los monumentos como tales, sino como "sitios de interés", categoría que no teníamos y que nos servirá para identificar aquellas atracciones que son muy concurridas, independientemente del tipo de atracción con el que estaban clasificadas anteriormente.

Necesitaremos crear esa nueva categoría y pasar todas las atracciones de Beijing que tuvieran la categoría "Monument" a la categoría "Tourist site"

Agregamos regla Error

The screenshot displays the GeneXus IDE interface for two entities: Category and Attraction. For each entity, the 'Structure' view shows the database fields, and the 'Rules' view shows a custom error rule.

Category Entity Structure:

Name	Type	Description	Formula	Nullable
Category	Category	Category		
CategoryId	Id	Category Id		No
CategoryName	Name	Category Name		No

Category Error Rule:

```
1 Error( "Enter de category name, please")
2 |   if CategoryName.IsEmpty();
3
```

Attraction Entity Structure:

Name	Type	Description	Formula	Nullable
Attraction	Attraction	Attraction		
AttractionId	Id	Attraction Id		No
AttractionName	Name	Attraction Name		No
CountryId	Id	Country Id		No
CountryName	Name	Country Name		
CategoryId	Id	Category Id		Yes
CategoryName	Name	Category Name		
AttractionPhoto	Image	Attraction Photo		No
CityId	Id	City Id		Yes
CityName	Name	City Name		

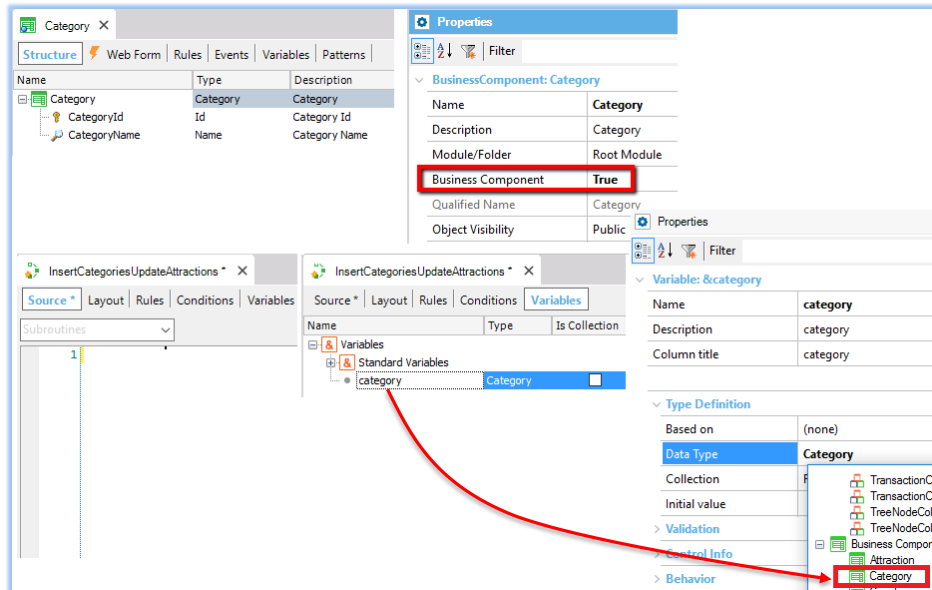
Attraction Error Rule:

```
1 Error( "Enter the attraction name, please" )
2 |   if AttractionName.IsEmpty();
3
```

Para esto deberemos insertar una nueva categoría en la tabla CATEGORY y luego recorrer las atracciones de la tabla ATTRACTION, filtrando por ciudad igual a "Beijing" y por categoría igual a "Monument", para cambiar el código de categoría por el de "Tourist site".

Antes de continuar, vemos que hemos agregado una regla error tanto a la transacción Category como a Attraction, para controlar que el nombre no quede vacío. Explicaremos para qué usaremos esto en un momento.

Business Component de transacción Category



Para implementar la creación de la nueva categoría y la actualización de las atracciones, vamos a crear un objeto procedimiento al que llamamos InsertCategoryUpdateAttractions.

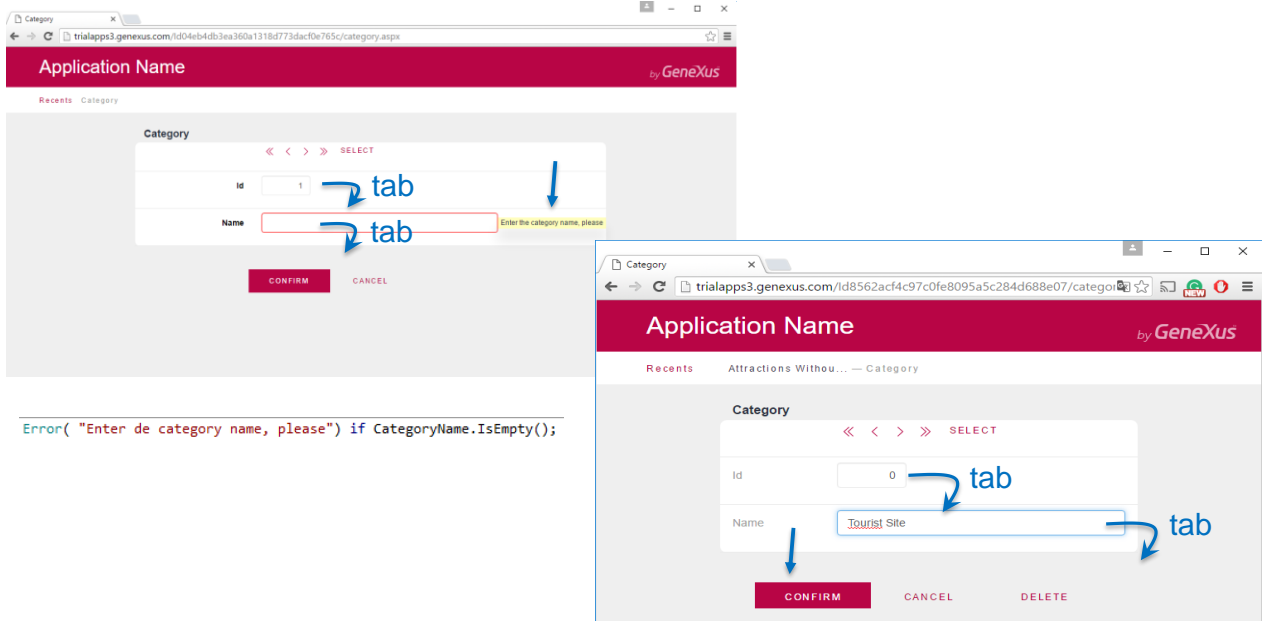
Lo primero que tenemos que hacer ahora es crear la categoría "Tourist site" y para ello, vamos a utilizar un Business Component de Category. Para crearlo, vamos a la transacción Category, ubicamos la propiedad **Business Component** y le asignamos el valor True.

Al hacer esto, GeneXus creará una estructura a partir de la transacción Category, que tendrá asociadas funcionalidades similares a las de la transacción.

Para poder usar esta definición, debemos crear una variable del tipo de este business component de Category. Así que vamos a la solapa de variables del procedimiento, creamos una variable de nombre &category y vemos que GeneXus le asigna automáticamente el tipo de datos Category. Este tipo de datos corresponde al Business Component Category, creado a partir de la transacción Category.

Si vamos a las propiedades de la variable y hacemos clic en Data Type, vemos que encontramos un grupo denominado Business Component y si lo abrimos encontraremos el Business Component Category, con un ícono igual al de una transacción. Confirmamos entonces que GeneXus creó un tipo de datos nuevo basado en la transacción Category y veremos que comparte muchas características con ella.

Cómo insertaríamos con la transacción Category



```
Error( "Enter de category name, please") if CategoryName.IsEmpty();
```

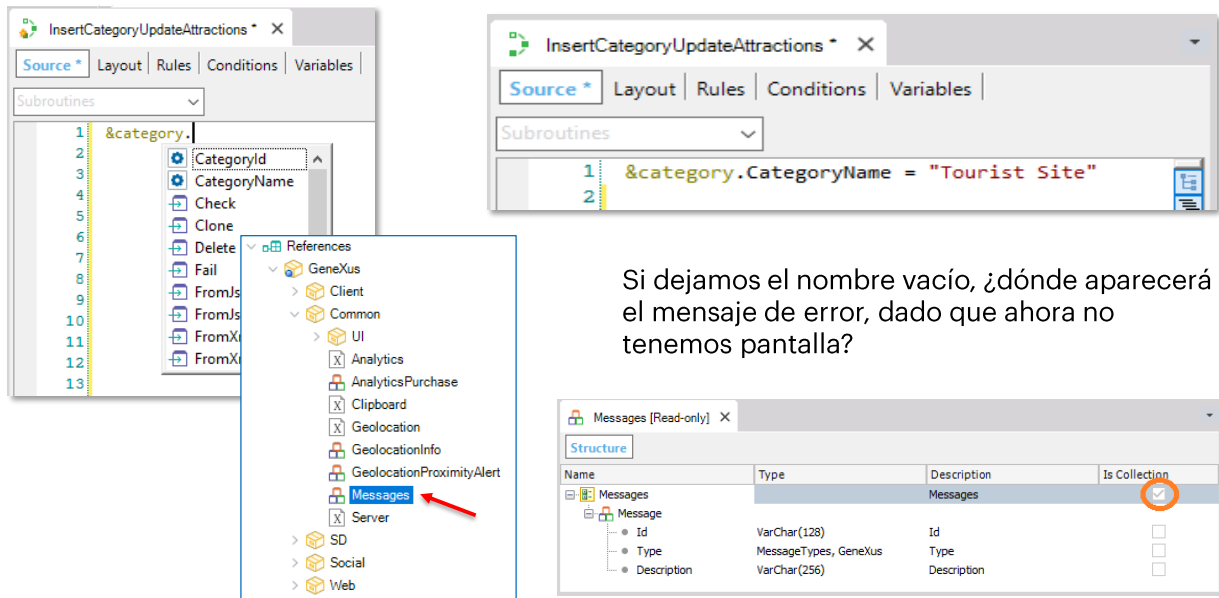
Recordemos que si ejecutamos la transacción Category, por defecto se está esperando que el usuario inserte un nuevo registro.

Lo primero que haríamos sería posicionarnos en el identificador de categoría, pero como el atributo CategoryId es autonumerado, no es necesario asignarle valor, de modo que damos tab para salir del campo, dejando al mismo vacío.

Luego nos corresponde escribir el nombre de la categoría. ¿Qué pasa si decidiéramos dejarla vacía? ¡Se disparará la regla Error que controla que no dejemos al atributo CategoryName sin valor!

En el nombre de la categoría escribiríamos "Tourist site" y por último, una vez asignados los valores que queríamos a los atributos de la categoría, presionamos el botón de Confirmar, para que los datos se salven en la base de datos y damos por concluida la inserción.

Cómo insertamos con el Business Component Category



Si dejamos el nombre vacío, ¿dónde aparecerá el mensaje de error, dado que ahora no tenemos pantalla?

Name	Type	Description	Is Collection
Messages		Messages	<input checked="" type="checkbox"/>
Message			
Id	VarChar(128)	Id	<input type="checkbox"/>
Type	MessageTypes, GeneXus	Type	<input type="checkbox"/>
Description	VarChar(256)	Description	<input type="checkbox"/>

Cuando utilizamos un Business Component, la secuencia de operaciones es exactamente la misma. Para insertar un registro nuevo, lo que hacemos sería asignarle valores a los campos de la categoría y confirmar los cambios.

Para esto, insertamos en el source la variable `&category`. Si presionamos el punto, vemos que aparecen varios métodos disponibles (como Save, Delete, Check) y que también aparecen `CategoryId` y `CategoryName`, a los que podremos asignarles valor.

Como sabemos que el identificador de categoría es autonumerado, no le asignamos valor, y por defecto también asume que vamos a insertar.

Al igual que como lo hicimos con la pantalla de la transacción, continuamos con el nombre de la categoría.

Lo interesante de trabajar con Business Components es que también se dispararán las reglas definidas en la transacción que se usó para crearlo. Por lo tanto, si en nuestro ejemplo hubiéramos omitido asignarle valor al nombre de la categoría, se hubiera disparado la regla Error que controla esto, al igual que sucede cuando usamos la pantalla de la transacción Category. Sólo que esto no será cuando se encuentre esta asignación en el código, sino cuando se le diga al business component que grabe.

Debemos aclarar que no todas las reglas serán disparadas, ya que como estamos ejecutando el business component a través de código, no podrá dispararse ninguna regla que llame a algún objeto que tenga pantalla (por ejemplo a otra transacción o a un web panel), ni tampoco se disparará la regla Parm si hubiera sido definida en la transacción.

Está claro entonces que si dejamos la categoría sin nombre no se insertará el registro, pero ¿dónde aparecerá el mensaje de error, dado que ahora no tenemos pantalla? En toda base de conocimiento GeneXus crea un SDT predefinido, llamado Messages, que es una colección de ítems.

Cómo insertamos con el Business Component Category

The image displays two windows side-by-side. The left window is the GeneXus IDE, titled 'InsertCategoriesUpdateAttractions'. It has tabs for 'Source', 'Layout', 'Rules', 'Conditions', and 'Variables'. A dropdown menu shows 'Subroutines'. The code editor contains the following lines:
1 &category.CategoryName = "Tourist site"
2 &category.Save()
3
4 |
5
The right window is a web browser showing a 'Category' form. The form has a header 'Application Name by GeneXus'. Below it, there are navigation buttons '<<', '<', '>', '>>' and a 'SELECT' button. The form fields are 'Id' (value 0) and 'Name' (value 'Tourist Site'). At the bottom, there are three buttons: 'CONFIRM', 'CANCEL', and 'DELETE'. Red arrows point from the code in the IDE to the 'Name' field and the 'CONFIRM' button in the web application.

Continuando con nuestro ejemplo, si estuviéramos trabajando con la pantalla de la transacción, una vez que asignamos el nombre de la categoría debemos presionar el botón Confirmar. Con Business Components usamos el método `Save()`:

```
&category.CategoryName = "Tourist site"  
&category.Save()
```

Por cada variable business component, como `&category`, cuando se realiza una operación `Save` se carga en memoria una colección de mensajes, con todos los mensajes de advertencia o error que se produzcan como consecuencia de esa operación.

Tenemos manera de obtenerla y recorrerla, de forma sencillísima, pero no lo veremos aquí.

Cómo cambiarle la categoría a la atracción 2, "Muralla China"

Business Component

```

InsertCategoriesUpdateAttractions * X
Source * Layout Rules Conditions Variables *
Subroutines
1 &category.CategoryName = "Tourist site"
2 &category.Save()
3
4 &attraction.Load(2)
5 &attraction.CategoryId = &category.CategoryId
6
7 &attraction.Save()
8

```

vs

Transacción

Aquí estamos usando el valor que se le dio al id de la categoría que creamos antes con el BC

Hasta aquí tenemos creada la categoría "Tourist site", procederemos ahora a cambiarle la categoría a los monumentos de China.

Supongamos que solamente tenemos que cambiarle la categoría a la Muralla China, que tiene el id= 2. ¿En la transacción cómo hacemos? Abrimos la transacción Attraction, ponemos el 2 en el Id, salimos del campo, y GeneXus nos trae todos los datos de la Muralla China.

Luego cambiamos los valores que nos interesan, en este caso, la categoría y salvamos (presionando Confirm).

Al confirmar se realiza el cambio, disparándose todas las reglas que correspondan. Por ejemplo, si se hubiera borrado el nombre a la atracción, habría fallado la actualización. Y por otro lado, si se hubiera cambiado el id de categoría por una inexistente, como GeneXus controlará la consistencia de los valores entre tablas relacionadas (la integridad referencial), se hubiera dado un error y no se hubiera realizado la modificación.

Este mismo comportamiento ocurrirá con el business component.

Para posicionarnos en una atracción, utilizamos el comando Load y entre paréntesis ponemos el id de la atracción, en este caso el 2.

Luego asignamos el nuevo valor de la categoría (que es el valor del id de la categoría "Tourist site" que creamos anteriormente) y finalmente hacemos un Save().

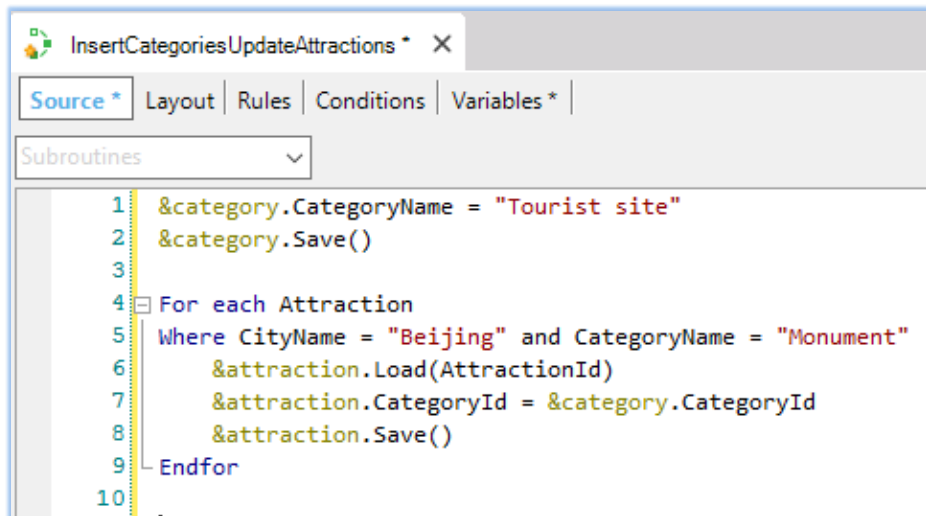
```
&Attraction.Load(2)
```

```
//Aquí estamos usando el valor que se le dio al id de la categoría que creamos antes con el BC.
&Attraction.CategoryId = &category.CategoryId
```

```
&Attraction.Save()
```

Al haber escrito el Load de un registro existente, el Save sabe que estamos queriendo actualizar (no insertar).

Cómo cambiarle la categoría a todas las atracciones de Beijing que eran monuments... con business component Attraction



```
InsertCategoriesUpdateAttractions * X
Source * | Layout | Rules | Conditions | Variables * |
Subroutines
1  &category.CategoryName = "Tourist site"
2  &category.Save()
3
4  For each Attraction
5  Where CityName = "Beijing" and CategoryName = "Monument"
6    &attraction.Load(AttractionId)
7    &attraction.CategoryId = &category.CategoryId
8    &attraction.Save()
9  Endfor
10
```

Pero nuestro requerimiento no es cambiar solamente la categoría de la Muralla China, sino de todas las atracciones de Beijing que fueran monumentos. Para eso recorreremos todas esas atracciones.

Como ahora estamos recorriendo registros con un for each, haremos el Load del Id de atracción en el que nos encontremos en cada iteración.

Cómo saber si grabación de Business Component fue exitosa y cómo dar por válidas las operaciones en la base de datos

```

1  &category.CategoryName = "Tourist site"
2  &category.Save()
3
4  If &category.Success()
5    For each Attraction
6      Where CityName = "Beijing" and CategoryName = "Monument"
7      &attraction.Load(AttractionId)
8      &attraction.CategoryId = &category.CategoryId
9      &attraction.Save()
10   Endfor
11   Commit
12 Endif
13

```

Aquí se grabó exitosamente la categoría y se modificaron las atracciones que no fallaron

Da por buenas todas las operaciones sobre la BD

Como cada Save() puede ser exitoso o no, dependiendo justamente de las reglas del negocio y de la integridad referencial, los registros pueden haberse insertado o modificado en la base de datos o no.

Para saber si un comando Save() fue o no exitoso, contamos con el método Success(), que nos devuelve True si fue exitoso y False en caso contrario. Que el comando Save() falle, podría ser por haberse disparado una regla error que no dejó salvar, o se cayó el sistema en el momento de salvar y el registro no quedó guardado. O falló la integridad referencial.

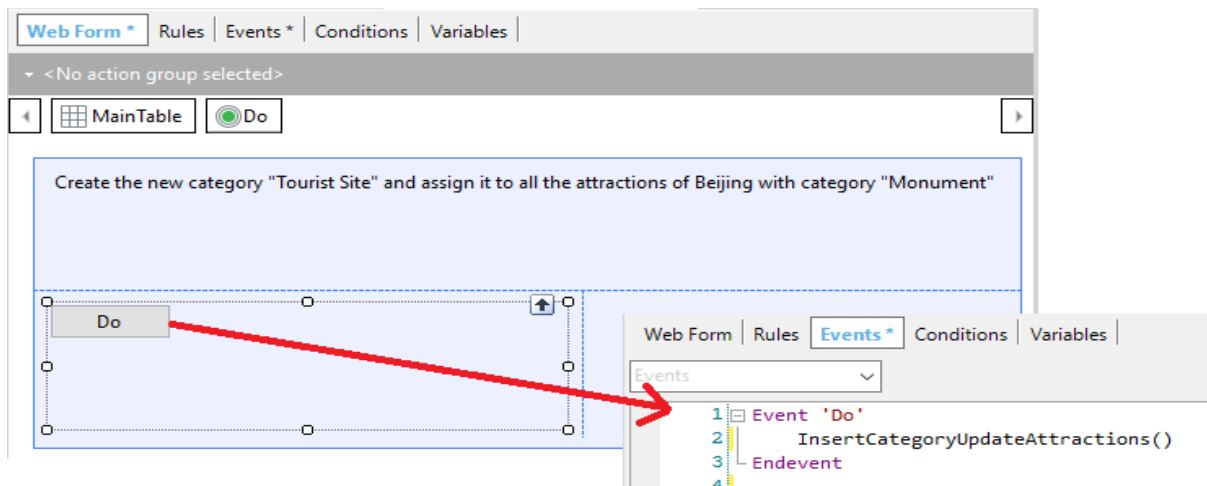
Si la inserción de la categoría nueva no fue exitosa, evidentemente el Save de las atracciones nos fallará, porque esa categoría no existirá. Así que condicionamos la modificación de las atracciones al resultado del método Success().

Sólo nos estaría quedando decirle a la base de datos que los registros que hemos insertado o actualizado están correctos y queremos que queden ingresados de forma tal que no se pierdan ante una caída del sistema. Esto lo hacemos agregando el comando Commit.

Llegado este momento sabemos que se grabó exitosamente la categoría y se modificaron las atracciones (para todas aquellas que no dieran error, por alguna regla de negocio que lo impidiera). Ahora necesitamos que la base de datos dé por buenas todas estas actualizaciones. Hemos insertado un registro y modificado otros. Si hubiera un fallo del sistema (por ejemplo, debido a un corte de energía eléctrica), las bases de datos se recuperan y deshacen todas las operaciones que no llegaron a ejecutar un Commit.

El comando **Commit** es un comando especial que nos permite indicar a la base de datos que finaliza el bloque de operaciones que queremos que se realicen todas juntas (y en caso de no poderse, que se deshagan todas). Al ejecutarse este comando, la base de datos salva los datos de tal manera que no puedan perderse ante una caída del sistema o falla de energía.

Invocando al procedimiento desde un Web Panel



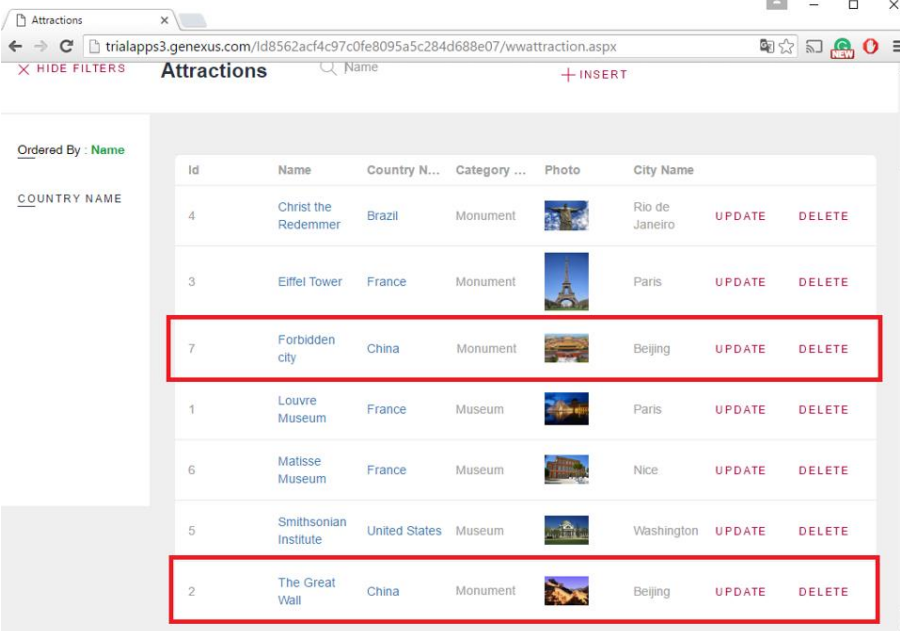
Para ejecutar nuestro procedimiento, necesitamos que otro objeto GeneXus lo invoque.

Esto lo podemos hacer desde un objeto Web panel. Ya habíamos visto que un Web Panel es una pantalla que podemos diseñar de acuerdo a nuestras necesidades para cumplir muchas funciones, como por ejemplo ver datos de la base de datos, confeccionar una pantalla de inicio para nuestro sistema o para ingresar datos.

En nuestro caso vamos a crear un web panel con un botón que al presionarlo llame al procedimiento que creamos anteriormente.








En el evento asociado al botón, invocamos al procedimiento `InsertCategoriesUpdateAttractions`, que no requiere parámetros.

ANTES



Ordered By: **Name**

COUNTRY NAME

Id	Name	Country N...	Category ...	Photo	City Name	UPDATE	DELETE
4	Christ the Redemmer	Brazil	Monument		Rio de Janeiro	UPDATE	DELETE
3	Eiffel Tower	France	Monument		Paris	UPDATE	DELETE
7	Forbidden city	China	Monument		Beijing	UPDATE	DELETE
1	Louvre Museum	France	Museum		Paris	UPDATE	DELETE
6	Matisse Museum	France	Museum		Nice	UPDATE	DELETE
5	Smithsonian Institute	United States	Museum		Washington	UPDATE	DELETE
2	The Great Wall	China	Monument		Beijing	UPDATE	DELETE

Ejecutamos la aplicación. Observemos que las atracciones de China tiene la categoría Monument.

Si vemos la información detallada, confirmamos que ambas están ubicadas en Beijing.

DESPUÉS

F5

The screenshot shows a web application interface with two browser windows. The left window displays a sidebar with a 'Do' button. A red arrow points from this button to the right window, which shows a table of attractions. The table has the following data:

Id	Name	Country N...	Category ...	Photo	City Name	UPDATE	DELETE
4	Christ the Redenmer	Brazil	Monument		Rio de Janeiro	UPDATE	DELETE
3	Eiffel Tower	France	Monument		Paris	UPDATE	DELETE
7	Forbidden city	China	Tourist Site		Beijing	UPDATE	DELETE
1	Louvre Museum	France	Museum		Paris	UPDATE	DELETE
6	Matisse Museum	France	Museum		Nice	UPDATE	DELETE
5	Smithsonian Institute	United States	Museum		Washington	UPDATE	DELETE
2	The Great Wall	China	Tourist Site		Beijing	UPDATE	DELETE

Si abrimos el webpanel CategoriesAndAttractions, presionamos el botón Do y volvemos a las atracciones, vemos que se creó la atracción Tourist Site y se asignó a las dos atracciones que eran de Beijing y de categoría Monumento.

Vemos que en el caso de la categoría, GeneXus infirió que tenía que hacer un Insert y en el segundo, cuando cambiamos el valor de la categoría de la atracción, que se trataba de un Update, al igual que lo hace con una transacción. Es por ello que en el procedimiento utilizamos el método Save(), que tiene la inteligencia de guardar los datos, sin importar cuál era la operación que se había realizado.

Métodos Insert() y Update() para especializar el Save()

```
&category.CategoryName = "Tourist site"
&category.Insert()

If &category.Success()
  For each Attraction
    Where CityName = "Beijing" and CategoryName = "Monument"
    &attraction.Load(AttractionId)
    &attraction.CategoryId = &category.CategoryId
    &attraction.Update()
  Endfor
  Commit
Endif
```

`&attraction.AttractionId = AttractionId`

Para intentar un Insert y si falla un Update: &attraction.InsertOrUpdate()

InsertOrUpdate() – Intenta un insert y si no puede intenta un update

Vimos que el método Save() tiene la inteligencia de guardar la operación, sin importar si se trataba de una inserción o una modificación de los datos.

Pero también contamos con métodos Insert() y Update() que nos permiten indicar específicamente la operación que deseamos realizar.

Modificamos el código para utilizar estos métodos, Insert para el caso de las categorías y Update para las atracciones.

Notemos que en el caso de las atracciones, no hubiéramos necesitado el Load y podríamos haber directamente escrito:

```
&Attraction.AttractionId = AttractionId
```

Ya que como aclaramos que queremos hacer un Update de atracciones, GeneXus sabe que queremos actualizar un registro.

Además de las operaciones Insert() y Update() para indicar que queremos que se realice una inserción o una modificación específicamente, contamos también con la operación InsertOrUpdate(), mediante la cual, al asignarle valores nuevos a los atributos, se intentará hacer un Insert, y si no puede (porque por ejemplo ya existe un registro con esa clave), se intentará hacer un Update.

Insertando y actualizando en la BD desde un webpanel

```

1 Event 'Do'
2   //InsertCategoriesUpdateAttractions()
3   &category.CategoryName = "Tourist site"
4   &category.Insert()
5
6   If &category.Success()
7     For each Attraction
8       Where CityName = "Beijing" and CategoryName = "Monument"
9       //&attraction.Load(AttractionId)
10      &attraction.AttractionId = AttractionId
11      &attraction.CategoryId = &category.CategoryId
12      &attraction.Update()
13    Endfor
14    Commit
15  Endif
16 Endevent

```

```

Event 'Undo'
&categoryId = find( CategoryId, CategoryName = "Monument")
For each Attraction
  where CityName = "Beijing" and CategoryName = "Tourist Site"
  &attraction.Load(AttractionId)
  &attraction.CategoryId = &categoryId
  &attraction.Save()
endifor

&categoryId = find( CategoryId, CategoryName = "Tourist Site")
&category.Load(&categoryId)
&category.Delete()

If &category.Success()
  Commit
else
  Rollback
endif
endevent

```

Otra cosa que es importante notar, es que como vimos, es posible insertar y actualizar registros de dos tablas distintas dentro del mismo objeto (en este caso un procedimiento), pero también se podría haber colocado este código directamente en el evento del web panel, porque las actualizaciones a la base de datos a través de business components pueden realizarse desde cualquier objeto (con algunas restricciones en el caso de las transacciones).

Si quisiéramos deshacer lo hecho, agregamos botón "Undo" al web panel y en él tendríamos que borrar la categoría "Tourist site" recién creada, y volver a dejar la categoría de las atracciones a las que habíamos cambiado por "Tourist site" como "Monuments".

¿Cómo borraríamos la categoría mediante la transacción? Primero la ubicaríamos, eligiendo su ID. Y luego presionaríamos el botón **Delete**. Usando los business components será igual, a través de un método con ese nombre.

Si no supiéramos el ID, lo buscamos usando la fórmula Find. La fórmula find es una fórmula inline que nos permite encontrar el primer registro que cumpla la condición indicada, y para ese registro, devuelve el valor del atributo que especificamos.

Si elimináramos la categoría "Tourist Site" a través de la transacción, la eliminación nos fallará, debido a que hay atracciones que tienen asignadas esa categoría.

El Business Component hará exactamente lo mismo, controlando la integridad referencial en el momento en que se ejecute el código del Delete(). No se puede borrar la categoría "Tourist Site" porque existen atracciones con esa categoría, que nos quedarían colgadas.

Por lo que antes de hacer esto, tendremos que cambiarles la categoría a esas atracciones, restituyéndoles la anterior que tenían: "Monument".

Como en general no recordamos los valores de los identificadores pero sí los nombres, es más seguro, antes que confiar en nuestra memoria buscarlos en la base de datos a partir del nombre. Así, en nuestro caso para recuperar el valor de Id de la categoría Monument utilizamos la fórmula find.

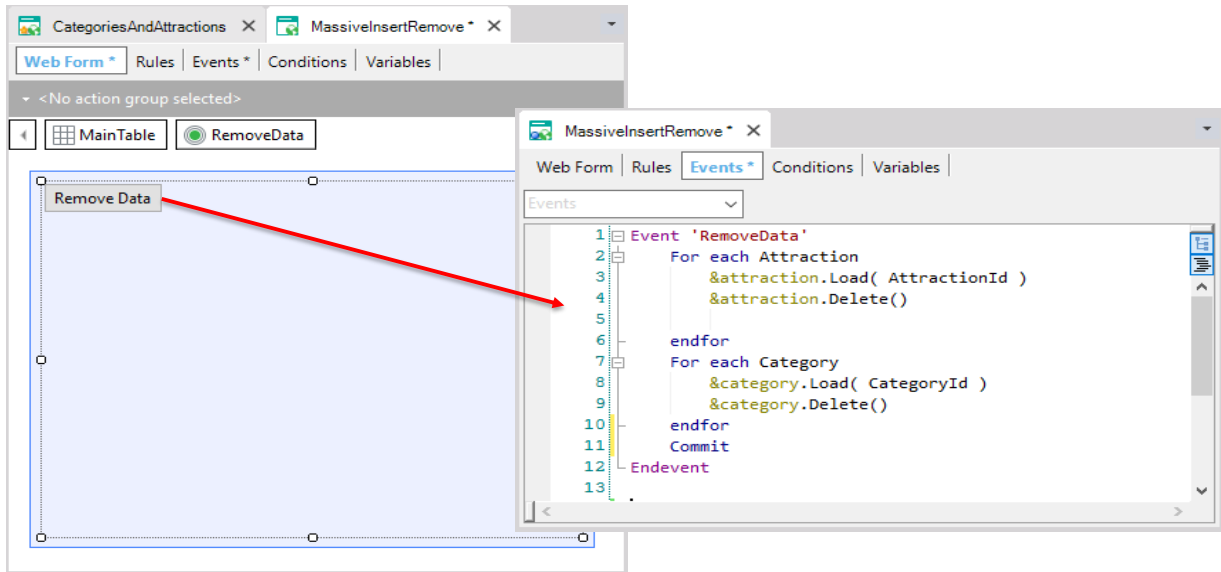
Para cada atracción de Beijing con la categoría "Tourist Site" cargamos en la variable business component

&Attraction todos sus datos, y modificamos únicamente el de CategoryId, asignándole la correspondiente a "Monument" y actualizamos.

Ahora sí podemos eliminar la categoría "Tourist Site". Y por último, no olvidemos el commit, para confirmar que las eliminaciones se hagan definitivamente sobre la base de datos, y no suceda que ante una caída del sistema, debido al rollback que hacen las bases de datos al recuperarse, los registros que habían sido eliminados se restituyan.

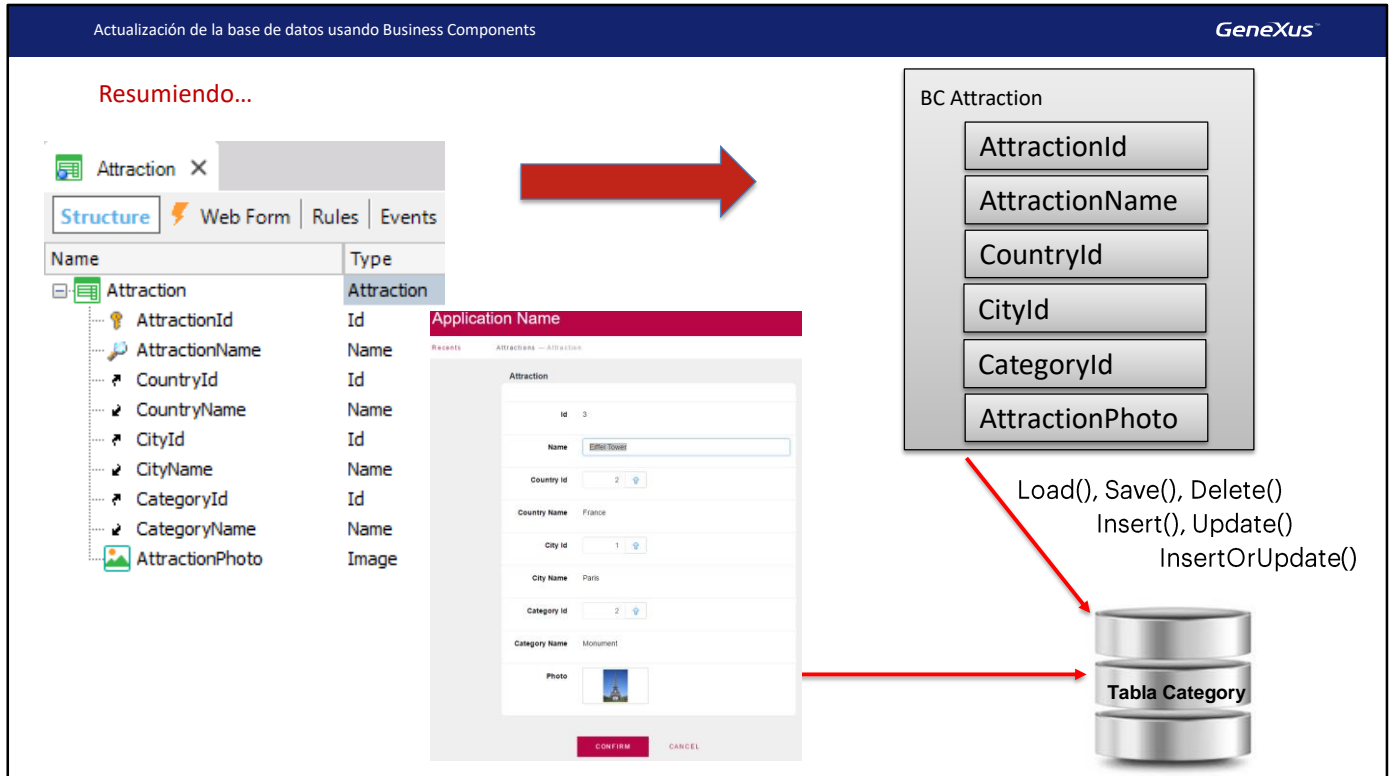
Si alguna de las atracciones falló en su actualización, también fallará el Delete de &category, así que podemos directamente preguntar si fue exitosa la eliminación de la categoría para hacer el commit. Y en caso contrario, ninguna modificación sobre la base de datos anterior queremos que quede efectuada, por lo que podemos provocar el rollback, es decir, deshacer todo lo que se hubiera hecho.

Inserción y eliminación masivas



Observar que si quisiera borrar todas las atracciones anteriores, en vez de cambiarles la categoría...se debería eliminar todos esos registros de la tabla ATTRACTION. Esto podríamos hacerlo a través de un for each con Delete. Para ver esto, vamos a creamos un Web panel "MassiveInsertRemove con un botón "Remove data".

Notemos que es importante el orden en que se ejecutan las eliminaciones, ya que como sabemos que los business components controlan la integridad referencial, no podremos borrar primero las categorías.



Resumiendo lo que hemos visto, aprendimos que el Business Component es una especie de tipo de datos especial, construido a partir de una transacción, como un espejo de ella: permite hacer todo lo que la transacción hace, salvo lo que tenga que ver con su pantalla.

Es por ello que una vez que tenemos el Business Component (espejo) de la trn, podemos definir una variable de ese tipo de datos, que será como un SDT, estructurado, con igual estructura que la transacción. Y a través de los métodos Load, Save y Delete (o Insert, Update y Delete), podremos realizar las mismas operaciones que interactivamente utilizamos en las transacciones, asegurándonos que las reglas de negocio se disparen "como si" se hubiera ejecutado la transacción. ¿Cuáles reglas de negocio? Las que no tienen relación con trabajar interactivamente a través de una pantalla, ni invocan a otra pantalla.

Es por ello que decimos que ejecutar un Business Component es como ejecutar la transacción en forma "silenciosa".

Para una transacción de un solo nivel como Attraction, la estructura del Business Component contendrá no sólo un elemento por atributo presente físicamente en la tabla asociada, sino también los atributos como CategoryName, o CountryName o CityName que aparecen en la estructura de la transacción como inferidos a través de las llaves foráneas. Aparecen con la misma función que en las transacciones: poder inferir valores luego de hacer Load.

ADVERTENCIA

The screenshot shows the GeneXus IDE interface for the 'Attraction' transaction. The main window displays a table of attributes with columns for Name, Type, Description, Formula, and Nullable. The 'CategoryId' attribute is selected, and its 'Nullable' property is set to 'Yes'. The Properties window on the right shows the 'Attribute: CategoryId' with a table of properties. A red box highlights the 'Empty as null' and 'Yes' options in the Properties window.

Name	Type	Description	Formula	Nullable
Attraction	Attraction	Attraction		
AttractionId	Id	Attraction Id		No
AttractionName	Name	Attraction Name		No
CountryId	Id	Country Id		No
CountryName	Name	Country Name		
CategoryId	Id	Category Id		Yes
CategoryName	Name	Category Name		
AttractionPhoto	Image	Attraction Photo		
CityId	Id	City Id		
CityName	Name	City Name		
AttractionAddress	Address, GeneXus	Attraction Address		

Name	CategoryId
Description	Category Id
Empty as null	Yes
Column name	Category Id

Detengámonos un momento aquí para realizar una advertencia.

En nuestra transacción Attraction hemos permitido al atributo CategoryId, que será clave foránea en la tabla asociada, asumir valores null (cambiando el valor de la propiedad Nullable de No a Yes). Con esto, como vimos, lográbamos que el usuario no estuviera obligado a ingresar siempre una categoría para una atracción. Puesto que si logramos almacenar un valor Null en la base de datos entonces la integridad referencial no se chequea.

Vemos que en esta propiedad se está indicando que cuando el atributo queda vacío, se almacene un null en su lugar.

Insertemos, así, una atracción a través de la transacción Attraction o la paralela... bien, pudimos hacerlo.

ADVERTENCIA

Create the new category "Tourist site" and assign it to all the attractions of Beijing with category "Monument"

Do Undo

Insert Attraction

```
Event 'Insert Attraction'  
  &Attraction.AttractionName = "Louvre"  
  &Attraction.CountryId = 2  
  &Attraction.CityId = 1  
  if &Attraction.Insert()  
    Commit  
  endif  
Endevent
```

Error en ejecución → &Attraction.CategoryId quedó vacío en BD, no null

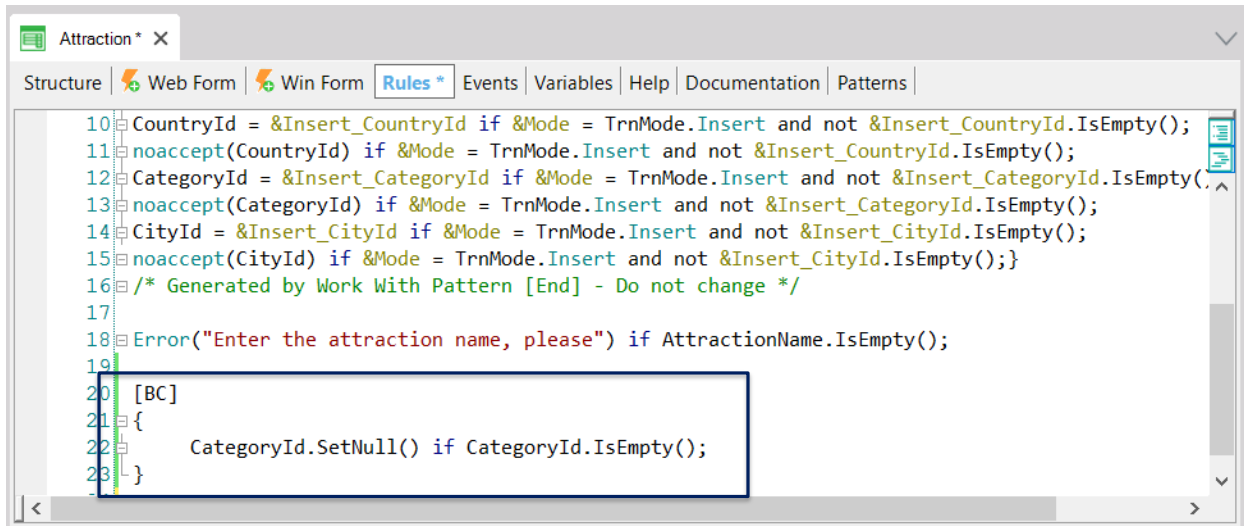
Sin embargo, si queremos hacer esto mismo pero a través del business component Attraction...

Para lo cual insertamos por ejemplo este botón en el web panel con el que veníamos trabajando...
...y en el evento vemos que al business component attraction le asignamos el nombre, el id del país, el id de ciudad e intentamos insertarlo... sin asignarle ningún valor al id de categoría.

Veamos qué ocurre cuando queremos ejecutar este código...

Nos dará un error de base de datos. ¿Por qué? Porque no insertó el valor null para ese atributo.

¿Por qué sucedió esto? porque cuando se ejecuta el business component no se está teniendo en cuenta el valor de esta propiedad, que está aplicando solamente cuando se trabaja con el atributo a través de la transacción web.

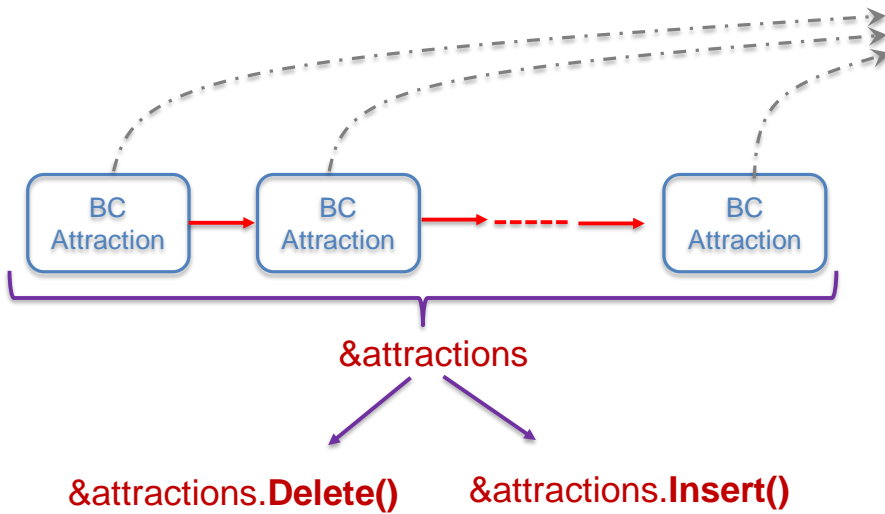
ADVERTENCIA

```
10 CountryId = &Insert_CountryId if &Mode = TrnMode.Insert and not &Insert_CountryId.IsEmpty();
11 noaccept(CountryId) if &Mode = TrnMode.Insert and not &Insert_CountryId.IsEmpty();
12 CategoryId = &Insert_CategoryId if &Mode = TrnMode.Insert and not &Insert_CategoryId.IsEmpty();
13 noaccept(CategoryId) if &Mode = TrnMode.Insert and not &Insert_CategoryId.IsEmpty();
14 CityId = &Insert_CityId if &Mode = TrnMode.Insert and not &Insert_CityId.IsEmpty();
15 noaccept(CityId) if &Mode = TrnMode.Insert and not &Insert_CityId.IsEmpty();}
16 /* Generated by Work With Pattern [End] - Do not change */
17
18 Error("Enter the attraction name, please") if AttractionName.IsEmpty();
19
20 [BC]
21 {
22     CategoryId.SetNull() if CategoryId.IsEmpty();
23 }
```

La manera de resolver esto es especificar una regla, que en este caso estamos condicionando a que solamente se ejecute para el caso del business component y no para la transacción web, y eso es lo que está indicado aquí entre corchetes: BC, business component; las reglas que se especifiquen dentro de las llaves van a ser incorporadas solamente dentro del business component... bueno lo que estamos diciendo aquí es que si CategoryId está vacío entonces se le asigne al atributo el valor nulo.

Probémoslo. Y ahora sí está funcionando tal como queríamos.

Operaciones sobre colección de BCs



BC Attraction

AttractionId
AttractionName
CountryId
CityId
CategoryId
AttractionPhoto

Load(), Save(), Delete()
 Insert(), Update()
 InsertOrUpdate()



Las operaciones de Insert, Update o Delete pueden realizarse también en forma masiva sobre una colección de BCs, que aún no sabemos cómo cargar. Es decir, si tuviéramos una variable colección de Business Components, por ejemplo si tuviéramos cargadas todas las atracciones en una variable &attractions donde cada ítem fuera del tipo Business Component Attraction, entonces podíamos eliminarlas todas haciendo &attractions.Delete() sin necesidad de recorrer la colección ítem por ítem, para eliminarlas en forma individual.

Veremos más adelante un ejemplo que agrega registros en forma masiva, con Insert.

GeneXus™

The power of doing.

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