

# Actualización de la base de datos

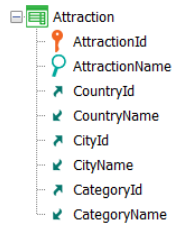
¿Qué sucede cuando una clave foránea acepta nulos?

**GeneXus**

# 1. Update through a form



Insert, Update, Delete



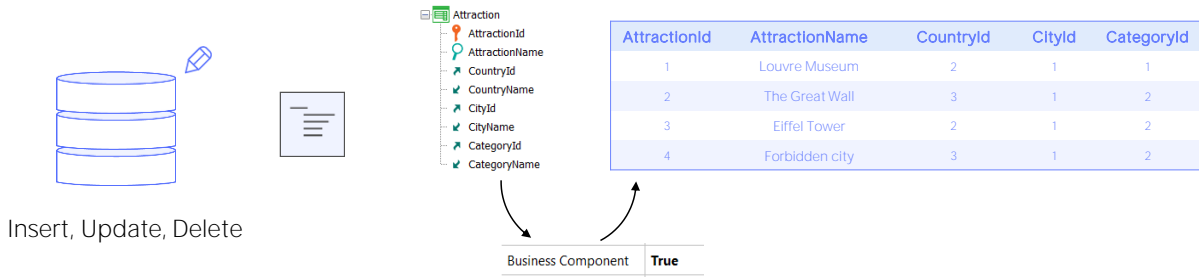
Attraction	
Id	2
Name	<input type="text" value="The Great Wall"/>
Country Id	<input type="text" value="3"/>
Country Name	China
City Id	<input type="text" value="1"/>
City Name	Beijing
Category Id	<input type="text" value="2"/>
Category Name	Monument

# 2. Update by code



Tenemos dos formas de actualizar la base de datos: hacerlo interactivamente a través del objeto transacción, utilizando su pantalla, o hacerlo por código.

## 1. Business Component: Insert(), Update(), Delete()



## 2. Procedure: New, For each, Delete

Para actualizar por código tenemos dos posibilidades:

Hacerlo utilizando el business component de la transacción, a través de sus métodos, o hacerlo exclusivamente dentro de un procedimiento, a través de los comandos New, For each con asignación directa de los atributos a ser modificados, y el comando Delete.



	Uniqueness controls	Referential Integrity controls	Rules/Events execution
Business Component	✓	✓	✓
Proc (new, for each, delete)	✓		

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2

Como sabemos, con ambas opciones se controla la unicidad de registros, es decir, nunca se permite dejar en la base de datos registros con clave primaria o candidata repetida. En el ejemplo, nunca se permitirán dos atracciones con el mismo identificador. Y si, por ejemplo, AttractionName fuera clave candidata, tampoco se permitirían dos atracciones con el mismo nombre.

Pero la gran diferencia entre actualizar vía Business Component y hacerlo vía comandos de actualización directa en un procedimiento es, no solo que únicamente los Business Components ejecutarán la lógica que proviene de las reglas y eventos de la transacción, sino que solo vía Business Components se controla la integridad referencial por programa. ¿Por qué decimos “por programa”?

The image shows two screenshots from the GeneXus IDE. The top screenshot displays the 'Structure' view for the 'Attraction' entity. The bottom screenshot displays the 'Structure' view for the 'Category' entity.

**Attraction Entity Structure:**

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

**Category Entity Structure:**




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

Si tenemos la transacción Attraction que registra las atracciones turísticas que siempre pertenecen a un país y ciudad y a alguna categoría (como monumento, museo, etc.), entonces claramente en la tabla asociada tendremos claves foráneas. En particular, el atributo CategoryId será clave foránea a la tabla Category.

El desarrollador GeneXus no tendrá que encargarse de programar en la transacción el control de integridad referencial para cada clave foránea porque ya estará incluido automáticamente en el programa generado. Es parte de la lógica de la transacción.

### Travel Agency

#### Attraction

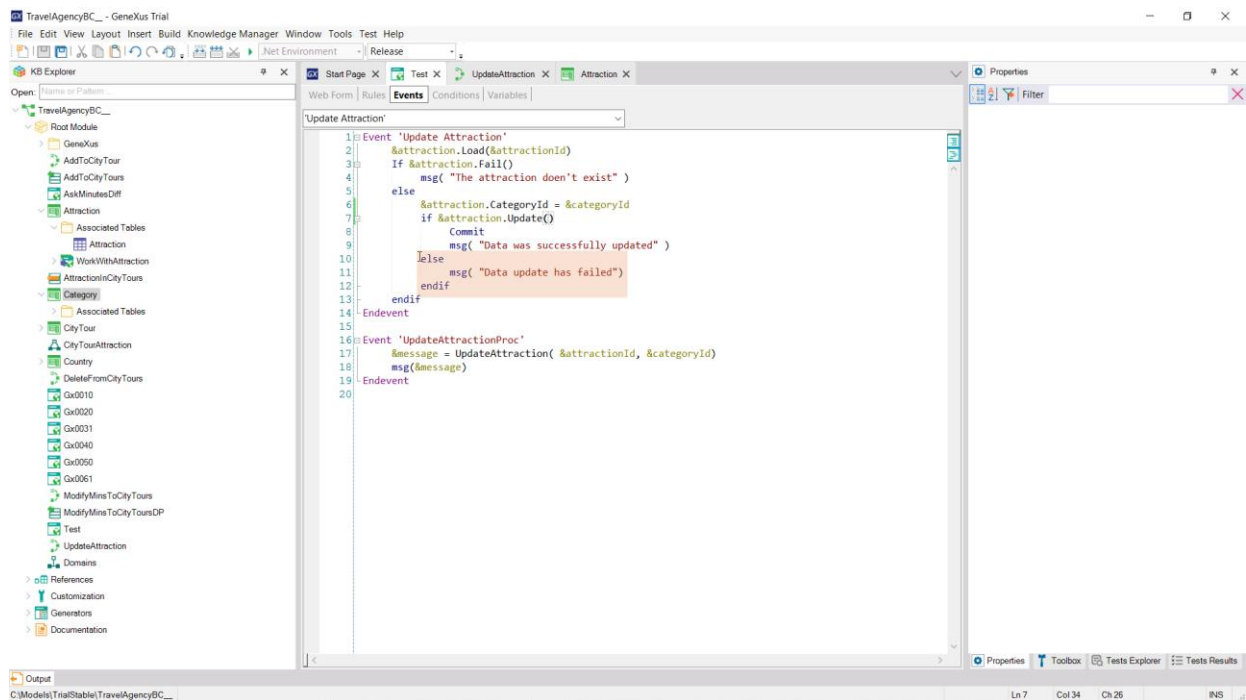
Id	2
Name	<input type="text" value="The Great Wall"/>
Country Id	<input type="text" value="3"/> 
Country Name	China
City Id	<input type="text" value="1"/> 
City Name	Beijing
Category Id	<input type="text" value="12"/>  <span style="color: red;">● No matching 'Category'.</span>
Category Name	

Así, cuando el usuario quiere asignarle a una atracción existente una categoría inexistente, la transacción nos informa que eso no es posible. Está haciendo el control de integridad referencial, antes de que lo haga la propia base de datos.

The image shows a web panel titled "Travel Agency". It contains two input fields: "Attraction id" (a white text box) and "Category id" (a blue dropdown menu). Below these fields are two buttons: "Update Attraction BC" on the left and "Update Attraction PROC" on the right. A mouse cursor is visible in the center of the panel.

Supongamos que implementamos un web panel que permite que el usuario ingrese un id de atracción y un id de categoría, para cambiarle a la atracción de ese id su categoría por esta otra.

Tenemos dos alternativas para hacerlo por código: una es a través del Business Component Attraction, y la otra será a través de un procedimiento.



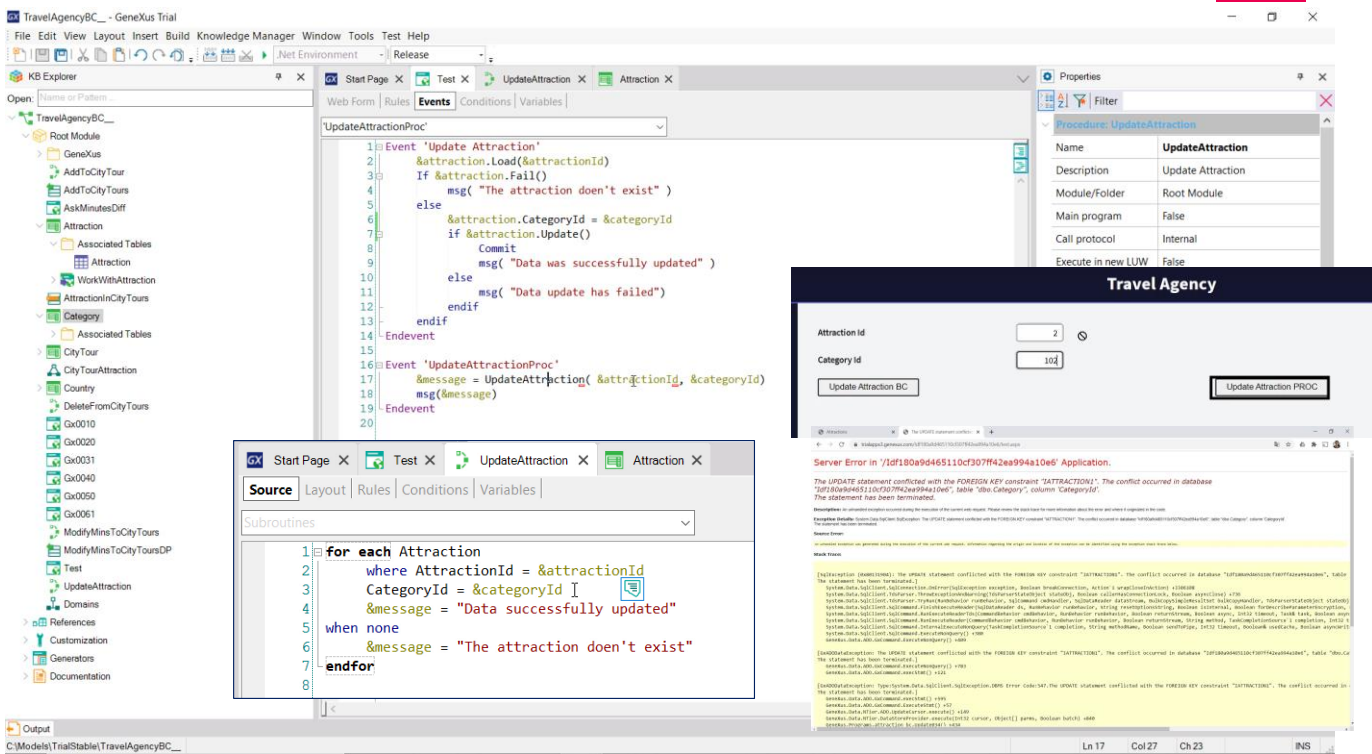
Veamos cómo implementáramos la primera.

Habiendo prendido la propiedad Business Component, definimos una variable de ese tipo, y en el evento asociado al botón la cargamos a partir de la variable &AttractionId que ingresa el usuario a través de la pantalla.

Si existe, es decir, no falla el Load, entonces le cambiamos la categoría por la ingresada por el usuario, e intentamos actualizar. Esto funcionará igual que la transacción, puesto que el código asociado a ese método Update chequeará la existencia de esa categoría para recién después enviar la orden de actualización a la base de datos. Y como ese código encontrará que no existe esa categoría, no llegará a hacer el intento de actualización en la base de datos; devolverá False y se ejecutará el “else”.

Probémoslo en ejecución.





En cambio, si esa misma actualización queremos hacerla vía procedimiento, es decir, como vemos aquí, a este procedimiento le enviamos identificador de atracción y de categoría, y en él se busca la atracción y se actualiza directamente la categoría.

Cuando ejecutemos con una categoría inexistente, ¡auch! Se dará una excepción de base de datos y el usuario tendrá que enfrentarse a esta pantalla tan poco amigable. Es que aquí el programa no está haciendo los controles para asegurar la integridad referencial. Pero sí los está realizando a la base de datos. Y por eso este error.

Podríamos capturar el error para hacer algo más amigable y que el programa no cancele abruptamente. Para ello contamos con la regla Error\_handler. Como parámetro debemos dar el nombre de una subrutina que debemos definir en el objeto y que será donde programaremos el comportamiento en caso de error de base de datos, preguntando primero por ese error. Puede ver detalles sobre esto en nuestro wiki.



The screenshot displays two windows from the GeneXus IDE. The 'Preferences' window on the left shows a tree view with 'Data Stores' expanded to 'Default (SQL Server)'. The 'Properties' window on the right shows the configuration for 'DataStore: SQL Server'. The 'Declare referential integrity' property is highlighted and set to 'Yes'.

DataStore: SQL Server	
Type	DataStore
Description	SQL Server
Access technology settings	
Creation/Reorganization information	
Database schema	
Primary key definition	Primary key
Declare referential integrity	Yes
Default tables storage area	
Default indices storage area	
Default temporary storage area	
Generate COMMENT ON statements	No
Database information	

También tenemos la posibilidad de eliminar las declaraciones de integridad referencial en la base de datos, para que ésta no realice los controles. No es muy aconsejable, pero para aquellos casos que lo necesiten, contamos con esta propiedad a nivel del Data Store.



If select CategoryId from  
Category where CategoryId =102  
...



AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	102
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2

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

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	null
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2

Por defecto, entonces, si no modificamos esa propiedad, la integridad referencial podrá o no ser controlada programáticamente, pero siempre lo va a ser por base de datos.




Sin embargo habíamos visto un caso en el que se permitía un valor de clave foránea inexistente. Es que estrictamente hablando, no se trataba de un valor.

Era cuando permitíamos que la clave foránea fuera nula. ¿Cómo le indicábamos a la base de datos que para una clave foránea íbamos a permitir esto?

En la estructura de la transacción teníamos la columna Nullable, que por defecto estaba en No, pero que podíamos pasar a Yes.

### Travel Agency

#### Attraction

Id	2
Name	<input type="text" value="The Great Wall"/>
Country Id	<input type="text" value="3"/> 
Country Name	China
City Id	<input type="text" value="1"/> 
City Name	Beijing
Category Id	<input type="text" value="0"/> 
Category Name	

Esto hacía que cuando ejecutábamos la transacción, y dejábamos vacío el identificador de categoría, no sólo el programa no controlara la integridad referencial, sino que tampoco lo hiciera la base de datos.

# Travel Agency

Attraction Id

Category Id

Update Attraction BC

Update Attraction PROC

```

Server Error in '/Idf180a9d465110cf307ff42ea994a10e6' Application.
The UPDATE statement conflicted with the FOREIGN KEY constraint "IATTRACTION1". The conflict occurred in database "Idf180a9d465110cf307ff42ea994a10e6", table "dbo.Category", column "CategoryId". The statement has been terminated.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: The UPDATE statement conflicted with the FOREIGN KEY constraint "IATTRACTION1". The conflict occurred in database "Idf180a9d465110cf307ff42ea994a10e6", table "dbo.Category", column "CategoryId". The statement has been terminated.

Source Error:
An unhandled exception was generated during the execution of the current web request. Information regarding the origin and location of the exception can be identified using the exception stack trace below.

Stack Trace:
[SqlException (0x80131904): The UPDATE statement conflicted with the FOREIGN KEY constraint "IATTRACTION1". The conflict occurred in database "Idf180a9d465110cf307ff42ea994a10e6", table "dbo.Category". The statement has been terminated.]
System.Data.SqlClient.SqlException.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction) +3385108
System.Data.SqlClient.SqlClient.Connection.OnError(SqlException exception, Boolean callerAsConnectionLock, Boolean asyncClose) +736
System.Data.SqlClient.SqlParser.ThrowExceptionAndWarn(SqlParserStateObject stateObj, Boolean callerAsConnectionLock, Boolean asyncClose) +736
System.Data.SqlClient.SqlParser.TryParse(ReadBehavior readBehavior, SqlCommand cacheHandler, SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopyResultSet, IdParserStateObject stateObj)
System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, ReadBehavior readBehavior, String resetOptionsString, Boolean isInternal, Boolean forDescribeParameterEncryption, SqlCommandRequest request, CancellationToken cancellationToken)
System.Data.SqlClient.SqlCommand.RunExecuteReaderTds(CommandBehavior cmdBehavior, ReadBehavior readBehavior, Boolean returnStream, Boolean async, Int32 timeout, Task<Task> task, Boolean async, CancellationToken cancellationToken, Boolean ignoreCancelKey, Boolean allowMultipleOpenResultSets)
System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, ReadBehavior readBehavior, Boolean returnStream, String method, TaskCompletionSource`1 completion, Int32 timeout, Int32 timeout, CancellationToken cancellationToken, Boolean ignoreCancelKey, Boolean allowMultipleOpenResultSets)
System.Data.SqlClient.SqlCommand.InternalExecuteNonQuery(TaskCompletionSource`1 completion, String methodName, Boolean sendToPipe, Int32 timeout, Boolean async, CancellationToken cancellationToken, Boolean ignoreCancelKey, Boolean allowMultipleOpenResultSets)
System.Data.SqlClient.SqlCommand.ExecuteNonQuery() +388
GeneXus.Data.ADO.GACCommand.ExecuteNonQuery() +409

[ADODataException: The UPDATE statement conflicted with the FOREIGN KEY constraint "IATTRACTION1". The conflict occurred in database "Idf180a9d465110cf307ff42ea994a10e6", table "dbo.Category". The statement has been terminated.]
GeneXus.Data.ADO.GACCommand.ExecuteNonQuery() +783
GeneXus.Data.ADO.GACCommand.ExecStat() +121

[ADODataException: Type: System.Data.SqlClient.SqlException, Dbms error code(s): 547. The UPDATE statement conflicted with the FOREIGN KEY constraint "IATTRACTION1". The conflict occurred in database "Idf180a9d465110cf307ff42ea994a10e6", table "dbo.Category". The statement has been terminated.]
GeneXus.Data.ADO.GACCommand.ExecStat() +395
GeneXus.Data.ADO.GACCommand.Executestat() +57
GeneXus.Data.NTier.ADO.UpdateCursor.execute() +149
GeneXus.Data.NTier.DataStoreProvider.execute(Int32 cursor, Object[] parms, Boolean batch) +848
GeneXus.Program.attraction_bc.UpdateAttraction() +418
  
```

Sin embargo, si intentamos hacer esto mismo, pero a través del Business component, ¡auch! Nos está dando básicamente la misma excepción de base de datos que antes, por falla de integridad referencial. ¿Qué está pasando?

¿Por qué usando la transacción pudimos pero usando su Business Component no?

empty ≠ null

Numeric  
CategoryId 0

Numeric  
CategoryId null

Es que el valor vacío y el nulo son dos cosas distintas, muy distintas.

El valor vacío es un valor. Dependiendo del tipo de datos, será uno de los valores posibles de ese tipo. Así, si el tipo es numérico, el valor vacío es el cero. Las bases de datos tienen especificados cuáles son los valores vacíos de acuerdo a cada tipo de datos. Pero el nulo, o null, estrictamente hablando no es un valor. De hecho es un NO VALOR. Decir que CategoryId es Null es lo mismo que decir que no está especificado su valor, que no se conoce. No es lo mismo que decir que es vacío.

empty ≠ null



For AttractionId = 2 → CategoryId = 0



AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2



For AttractionId = 2 → CategoryId = null



AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	null
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2



Por tanto, para las bases de datos vacío y nulo son dos cosas muy distintas. Si estamos implementando un programa y enviamos a la base de datos la orden de modificar la categoría de la atracción 2 para que sea 0 en lugar de 2, lo que obtendremos será ni más ni menos que la pantalla con al excepción de base de datos por falla de integridad referencial. En cambio, si le decimos que lo cambie por null, no habrá ningún problema, tal como sucedió cuando lo hicimos a través de la transacción.

Entonces la pregunta es: ¿por qué la transacción está pidiéndole a la base de datos que modifique por Null y en cambio el Business component está enviando el valor vacío, cero?

LevelAgencyBC\_ - GeneXus Trial

Edit View Layout Insert Build Knowledge Manager Window Tools Test Help

.Net Environment | Release

Structure Web Form Rules Events Variables Patterns

Attraction

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

Attraction

<ErrorViewer: ErrorViewer>

<Toolbar>

Id AttractionId

Name AttractionName

Address AttractionAddress

Country Id CountryId

Country Name CountryName

City Id CityId

City Name CityName

Category Id CategoryId

CategoryId

Category Name CategoryName

<FormButtons>

Properties

Filter

Attribute: CategoryId

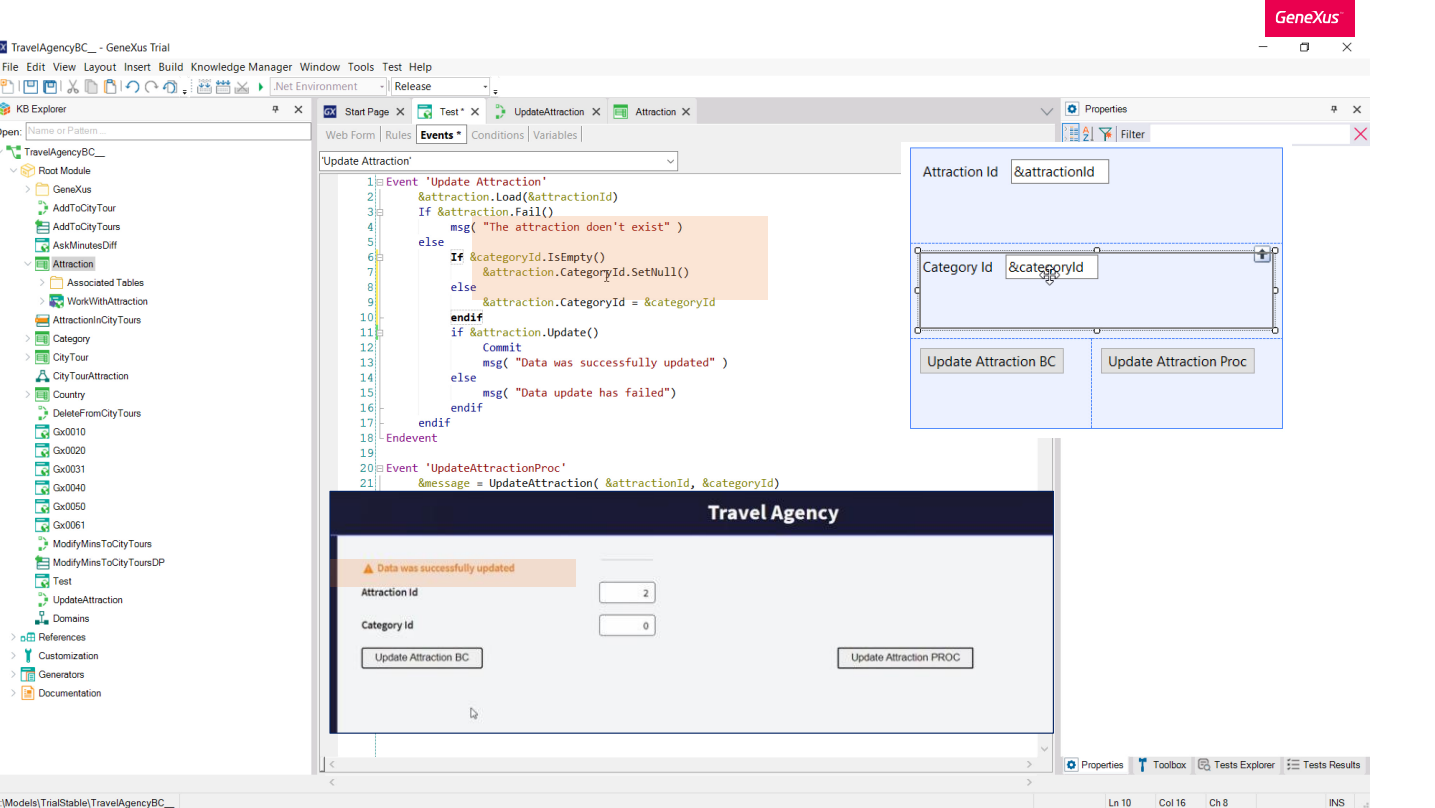
Name	CategoryId
Description	Category Id
Title	Category Id
Column title	Category Id
Contextual Title	Id
Formula	
Nulls in Forms	Empty as Null
Class	Attribute
Qualified Name	CategoryId

Type Definition

Supertype	
Based on	Id
Data Type	Numeric
Length	4
Decimals	0
Signed	False
Autonumber	True
Autonumber start	1
Autonumber step	1
Autonumber for rep	True
Initial value	
Validation	
Value range	
Validation Failed M	

Es que si observamos las propiedades del atributo clave foránea, CategoryId, vemos que hay una de nombre "Nulls in Forms" que tiene por defecto el valor "Empty as Null". Esto significará que si el atributo acepta nulos, como es el caso, entonces toda vez que se deje vacío el valor en el form, se considere que es Null cuando se envíe a la base de datos y no vacío.





Pero esto no vale cuando la actualización la hacemos a través del Business Component, que no tiene un form.

Por tanto si esta variable &CategoryId está vacía, este elemento quedará con valor vacío y no nulo, y por ello es que falla la integridad referencial al intentar la actualización. Tenemos que especificar el nulo si la variable está vacía.

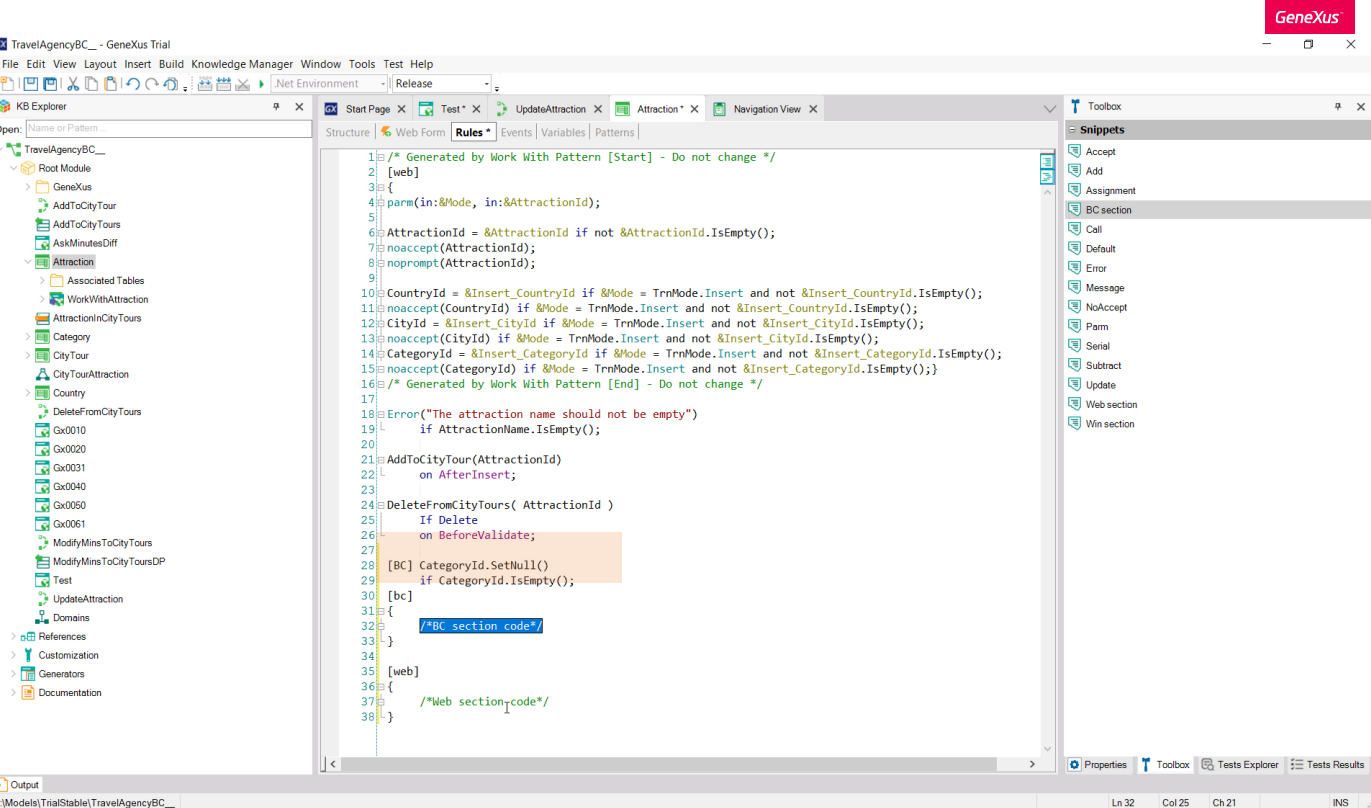
Tenemos dos formas de hacer esto. O lo hacemos a nivel local, solo aquí, o hacemos el mismo mapeo que consigue la propiedad del atributo, pero esta vez válido también en el Business Component, donde quiera que se use.

Para lo primero, alcanzará con especificar que si está vacía la variable CategoryId, al elemento del Business Component le configure el nulo. El método SetNull aplica a atributos que acepten nulos como a sus correspondientes en las variables Business Component, como es este caso.

Si ahora probamos, vemos que lo conseguimos.

La desventaja de esta alternativa es que si en otro lado queremos dejar nula esta clave foránea a través de una variable Business Component tendremos que hacer esto mismo.

La otra opción, más general, decíamos, es incorporar este SetNull como regla en la transacción. Aquí lo quitamos entonces...



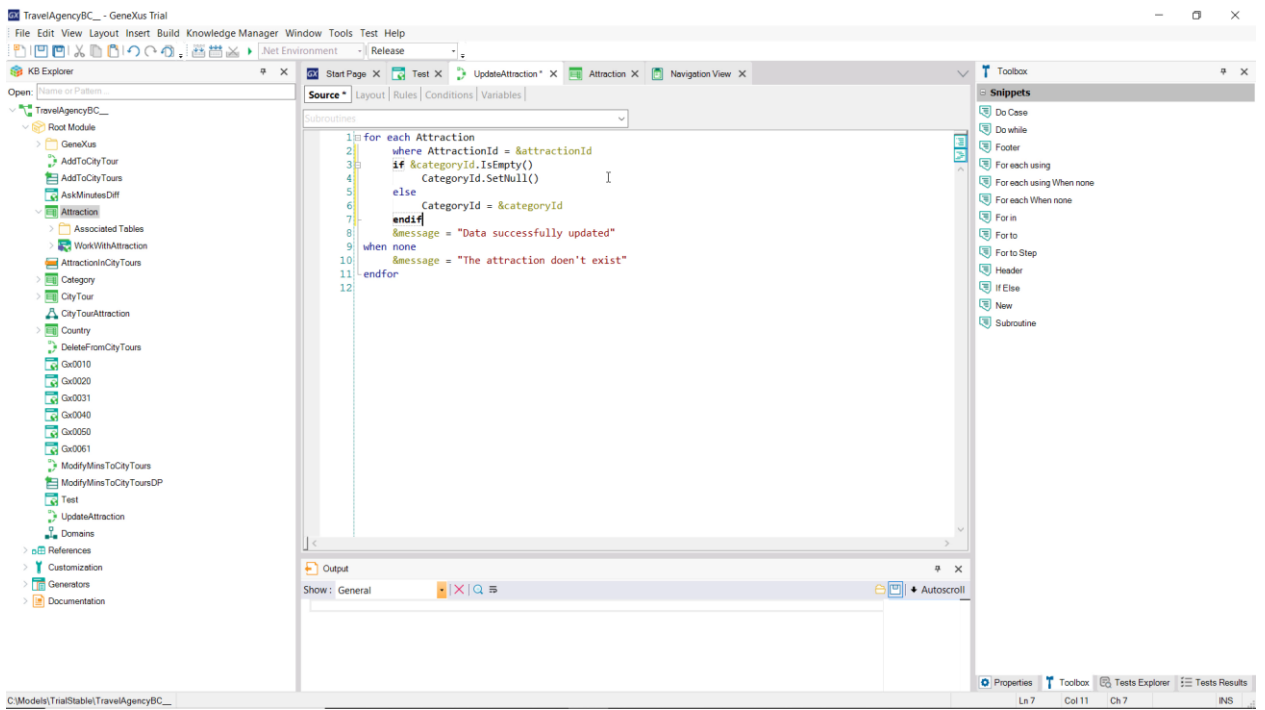
Y vamos a la transacción y especificamos que le asigne nulo al atributo cuando su valor se deje vacío. Esto mismo ya es lo que por defecto se realiza cuando se utiliza la pantalla de la transacción, gracias a esta propiedad, por lo que, estrictamente hablando, podemos pedir que esta regla solo se ejecute para el Business Component.

Siempre podemos indicar que una regla solo se ejecute cuando se trata de la transacción Web, o cuando se trata del Business Component, colocando esta marca.

De hecho, si queremos escribir un conjunto de reglas que solo apliquen a la transacción Web, o que solo apliquen al Business Component, lo hacemos así.

Probemos. Vemos que efectivamente es otra solución.





La solución aquí solo puede ser la local. Probemos.

Y vemos que lo conseguimos.

*GeneXus*<sup>TM</sup>

[training.genexus.com](http://training.genexus.com)  
[wiki.genexus.com](http://wiki.genexus.com)