

Atualização de base de dados

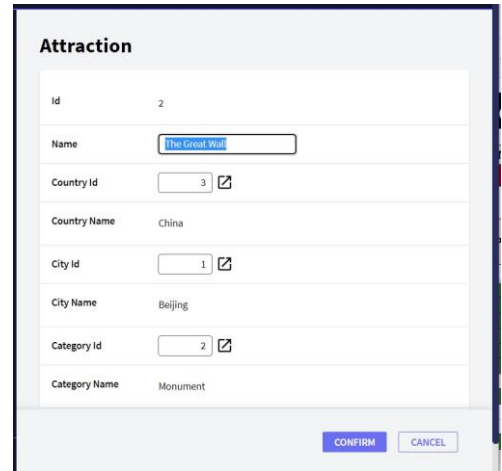
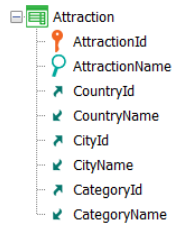
O que acontece quando uma chave estrangeira aceita nulos?

GeneXus




1. Update through a form



Insert, Update, Delete



Attraction

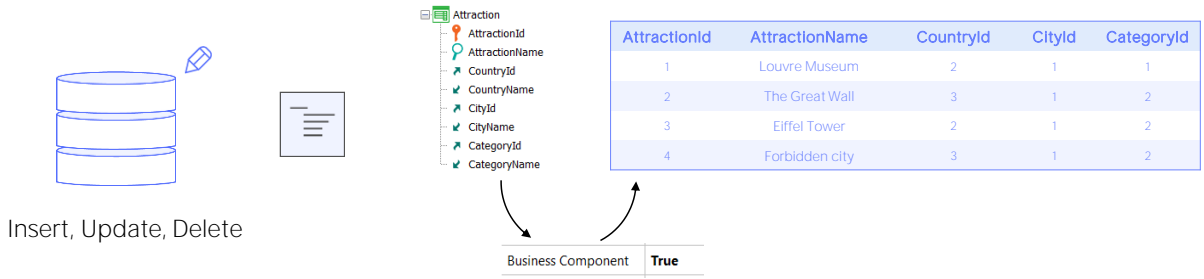
Id	2
Name	<input type="text" value="The Great Wall"/>
Country Id	<input type="text" value="3"/> 
Country Name	China
City Id	<input type="text" value="1"/> 
City Name	Beijing
Category Id	<input type="text" value="2"/> 
Category Name	Monument

2. Update by code



Temos duas formas de atualizar a base de dados: fazê-lo interativamente através do objeto transação, utilizando sua tela, ou fazê-lo por código.

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



2. Procedure: New, For each, Delete

Para atualizar por código tínhamos duas possibilidades:

Fazê-lo utilizando o business component da transação, através de seus métodos, ou fazê-lo exclusivamente dentro de um procedimento, através dos comandos New, For each com atribuição direta dos atributos a serem modificados, e o 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, com ambas as opções se controla a unicidade de registros, ou seja, nunca se permite deixar na base de dados registros com chave primária ou candidata repetida. No exemplo, nunca serão permitidas duas atrações com o mesmo identificador. E se, por exemplo, AttractionName fosse chave candidata, também não seriam permitidas duas atrações com o mesmo nome.

Mas a grande diferença entre atualizar via Business Component e fazê-lo via comandos de atualização direta em um procedimento é, não apenas que unicamente os Business Components executarão a lógica que provém das regras e eventos da transação, mas somente através da Business Components se controla a integridade referencial por programa. Por que dizemos “por programa”?

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

Attraction Table 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 Table Structure:




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

Se temos a transação Attraction que registra as atrações turísticas que sempre pertencem a um país e cidade e a alguma categoria (como monumento, museu, etc.), então, claramente na tabela associada teremos chaves estrangeiras. Em particular, o atributo CategoryId será chave estrangeira para a tabela Category.

O desenvolvedor GeneXus não terá que se encarregar de programar na transação o controle de integridade referencial para cada chave estrangeira, porque já estará incluído automaticamente no programa gerado. Faz parte da lógica da transação.

Travel Agency

Attraction

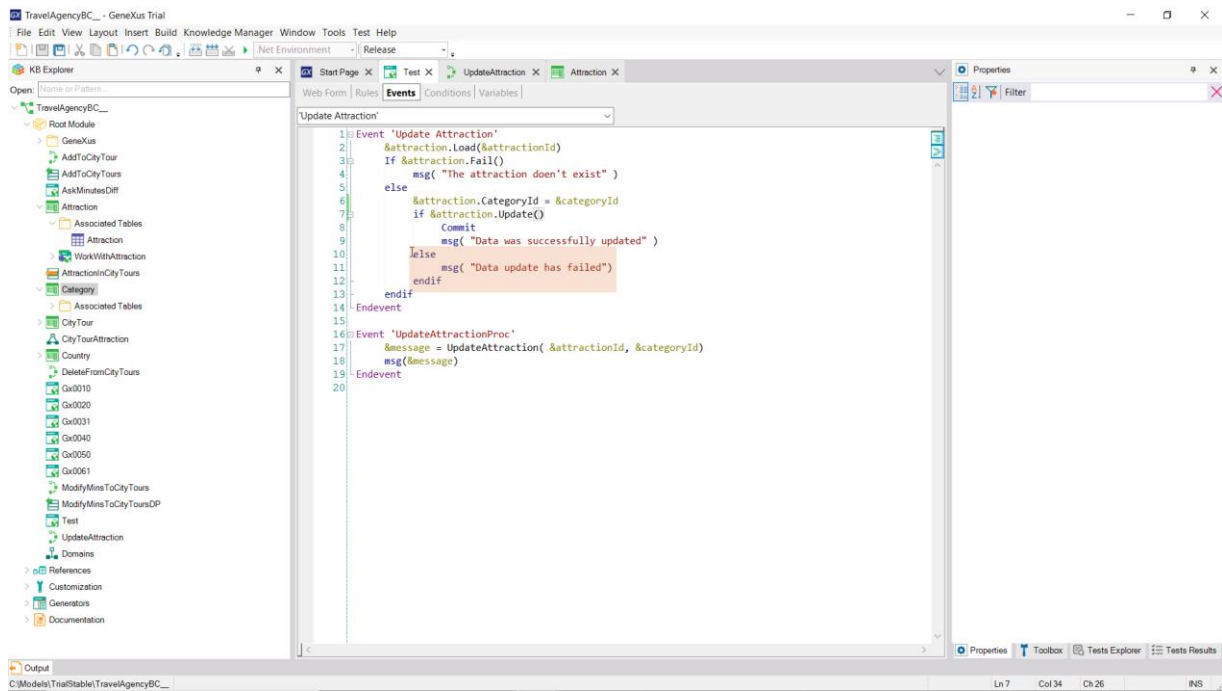
Id	2
Name	<input type="text" value="The Great Wall"/>
Country Id	<input type="text" value="3"/> 
Country Name	China
City Id	<input type="text" value="1"/> 
City Name	Beijing
Category Id	<input type="text" value="12"/>  ● No matching 'Category'.
Category Name	

Assim, quando o usuário quer atribuir a uma atração existente uma categoria inexistente, a transação nos informa que isso não é possível. Está fazendo o controle de integridade referencial, antes que a própria base de dados o faça.

The image shows a web interface titled "Travel Agency". It features two input fields: "Attraction id" and "Category id". The "Attraction id" field contains the value "1" and has a dropdown arrow on its right side. The "Category id" field contains the value "0". 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 page.

Suponhamos que implementamos um web painel que permite que o usuário insira um id de atração e um id de categoria, para mudar para a atração desse id, sua categoria por esta outra.

Temos duas alternativas para fazê-lo por código: uma é através do Business Component Attraction, e a outra será através de um procedimento.

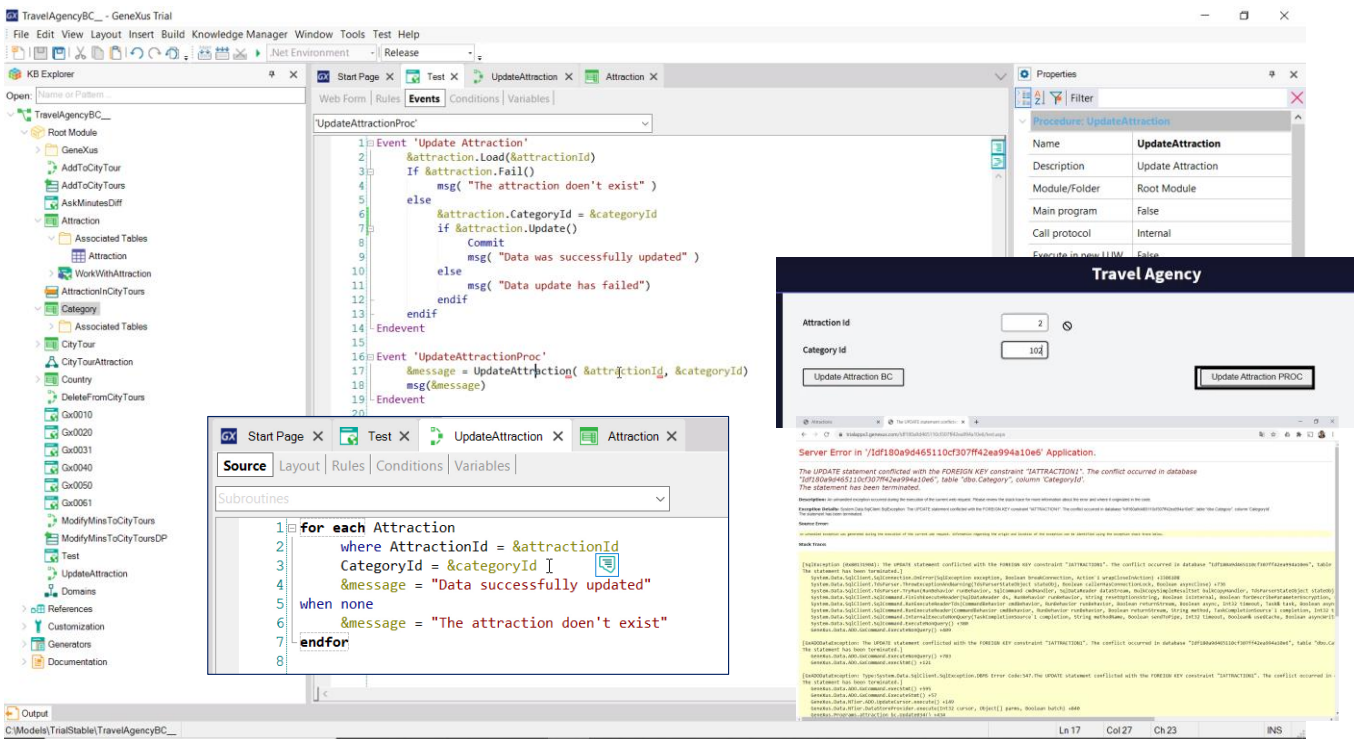


Vejamos como implementaríamos a primeira.

Tendo ativado a propriedade Business Component, definimos uma variável desse tipo, e no evento associado ao botão a carregamos a partir da variável &AttractionId que o usuário insere através da tela.

Se existe, ou seja, o Load não falha, então, trocamos a categoria pela ingressada pelo usuário, e tentamos atualizar. Isto funcionará como a transação, uma vez que o código associado a esse método Update irá verificar a existência dessa categoria para, em seguida, enviar a ordem de atualização para a base de dados. E como esse código descobrirá que não existe essa categoria, não chegará a fazer a tentativa de atualização na base de dados; devolverá False e executará o "else".

Testemos em execução.



Em contrapartida, se essa mesma atualização, quisermos fazê-la via procedimento, ou seja, como vemos aqui, a este procedimento enviamos identificador de atração e de categoria, e nele busca-se a atração e atualiza-se diretamente a categoria.

Quando executarmos com uma categoria inexistente, auch! Ocorrerá uma exceção de base de dados e o usuário terá que encarar esta tela tão pouco amigável. É que aqui o programa não está fazendo os controles para assegurar a integridade referencial. Mas sim, a base de dados está realizando-os. E por isso este erro.

Poderíamos capturar o erro para fazer algo mais amigável e que o programa não cancele abruptamente. Para isso temos a sub-regras com a regra Error_handler. Como parâmetro devemos dar o nome de uma sub-regras que devemos definir no objeto e que será onde programaremos o comportamento em caso de erro de base de dados, perguntando primeiro por esse erro. Você pode ver detalhes sobre isto em nossa wiki.



The screenshot displays the GeneXus IDE interface. On the left, the 'Preferences' window shows a tree view with 'TravelAgency' expanded to 'Data Stores', where 'Default (SQL Server)' is selected. On the right, the 'Properties' window shows the configuration for 'DataStore: SQL Server'. The 'Declare referential integrity' property is 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	

Também temos a possibilidade de eliminar as declarações de integridade referencial na base de dados, para que esta não realize os controlos. Não é muito aconselhável, mas para aqueles casos que necessitam, contamos com esta propriedade no nível do 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

Attraction X

Structure Web Form Rules Events Variables Patterns

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		No

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 padrão, então, se não modificamos essa propriedade, a integridade referencial poderá ou não ser controlada programaticamente, mas sempre será por base de dados.

No entanto, tínhamos visto um caso em que se permitia um valor de chave estrangeira inexistente. É que estritamente falando, não se tratava de um valor. Era quando permitíamos que a chave estrangeira fosse nula. Como indicávamos à base de dados que para uma chave estrangeira iríamos permitir isto?

Na estrutura da transação tínhamos a coluna Nullable, que por padrão estava em No, mas que podíamos passar para 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	

Isto fazia com que, quando executávamos a transação e deixávamos vazio o identificador de categoria, não só o programa não controlava a integridade referencial, como também não o fazia a base de dados.

Travel Agency

Attraction Id

2

Category Id

0

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", column "CategoryId". The statement has been terminated.]
System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction) +3306108
System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Boolean callerHasConnectionLock, Boolean asyncClose) +736
System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopySet, TdsParserStateObject stateObj)
System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, RunBehavior runBehavior, String resetOptionsString, Boolean isInternal, Boolean forDeserializeParameterEncryption, System.Data.SqlClient.SqlCommand.RunExecuteReaderTds(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, Timeout timeout, Int32 timeout, Task& task, Boolean async)
System.Data.SqlClient.SqlCommand.RunExecuteReaderTds(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, String method, TaskCompletionSource`1 completion, Int32 timeout, Int32 timeout, Task& task, Boolean async)
System.Data.SqlClient.SqlCommand.InternalExecuteNonQuery(TaskCompletionSource`1 completion, String methodName, Boolean sendToPipe, Int32 timeout, Boolean asyncWrite)
System.Data.SqlClient.SqlCommand.ExecuteNonQuery() +390
GeneXus.Data.ADO.GXCommand.ExecuteNonQuery() +409

[ADODataException: 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.]
GeneXus.Data.ADO.GXCommand.ExecuteNonQuery() +783
GeneXus.Data.ADO.GXCommand.ExecuteNonQuery() +321

[ADODataException: Type: System.Data.SqlClient.SqlException, DBMS Error Code: 1547, 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.]
GeneXus.Data.ADO.GXCommand.ExecuteNonQuery() +595
GeneXus.Data.ADO.GXCommand.ExecuteNonQuery() +57
GeneXus.Data.NTier.ADO.UpdateCursor.execute() +149
GeneXus.Data.NTier.DataStoreProvider.execute(Int32 cursor, Object[] params, Boolean batch) +840
GeneXus.Program_Attraction_Bc.UpdateAttraction() +834

```

No entanto, se tentarmos fazer o mesmo, mas através do Business Component, auch! Está nos dando basicamente a mesma exceção de base de dados que antes, por falha de integridade referencial. O que está acontecendo?

Por que usando a transação podíamos, mas usando seu Business Component não?

empty ≠ null

Numeric
CategoryId 0

Numeric
CategoryId null

É que o valor vazio e o nulo são duas coisas diferentes, muito diferentes.

O valor vazio é um valor. Dependendo do tipo de dados, será um dos valores possíveis desse tipo. Assim, se o tipo for numérico, o valor vazio é o zero. As bases de dados têm especificados quais são os valores vazios de acordo com cada tipo de dados. Mas o nulo, ou null, estritamente falando, não é um valor. De fato, é um NÃO VALOR. Dizer que CategoryId é Null, é o mesmo que dizer que seu valor não está especificado, que não é conhecido. Não é o mesmo que dizer que é vazio.

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



Portanto, para as bases de dados, vazio e nulo são duas coisas muito diferentes. Se estamos implementando um programa e enviamos à base de dados a ordem de modificar a categoria da atração 2 para que seja 0 em vez de 2, o que obteremos será nem mais nem menos que a tela com a exceção de base de dados por falha de integridade referencial. Em contrapartida, se lhe dissermos para o substituir por null, não haverá qualquer problema, tal como aconteceu quando o fizemos através da transação.

Então a pergunta é: por que a transação está pedindo à base de dados para modificar por Null e em vez disso o Business Component está enviando o valor vazio, zero?

The screenshot displays the GeneXus IDE interface. The 'Properties' window on the right is focused on the 'Attribute: CategoryId'. The 'Nulls in Forms' property is set to 'Empty as Null'. The 'Structure' window in the center shows the 'Attraction' entity with the following attributes:

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		

The 'Properties' window for 'Attribute: CategoryId' shows the following details:

- 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:

É que se observamos as propriedades do atributo chave estrangeira, CategoryId, vemos que há uma de nome “Nulls in Forms” que tem, por padrão, o valor “Empty as Null”. Isto significará que se o atributo aceitar nulos, como é o caso, então toda vez que se deixe vazio o valor no form, se considere que é Null quando se envie à base de dados, e não vazio.

The screenshot displays the GeneXus IDE interface. The central pane shows the 'UpdateAttraction' event code:

```

1 Event 'Update Attraction'
2   &attraction.Load(&attractionId)
3   If &attraction.Fail()
4     msg( "The attraction doesn't exist" )
5   else
6     If &categoryId.IsEmpty()
7       &attraction.CategoryId.SetNull()
8     else
9       &attraction.CategoryId = &categoryId
10    endif
11    if &attraction.Update()
12      Commit
13      msg( "Data was successfully updated" )
14    else
15      msg( "Data update has failed" )
16    endif
17  endif
18 Endevent
19
20 Event 'UpdateAttractionProc'
21   Message = UpdateAttraction( &attractionId, &categoryId)

```

Below the code, a Business Component form titled 'Travel Agency' is shown. It features a success message: 'Data was successfully updated'. The form contains two input fields: 'Attraction Id' with the value '2' and 'Category Id' with the value '0'. There are two buttons: 'Update Attraction BC' and 'Update Attraction PROC'.

Mas isto não vale quando fazemos a atualização através do Business component, que não tem um form.

Portanto, se esta variável &CategoryId está vazia, este elemento ficará com valor vazio e não nulo, e é por isso que a integridade referencial falha ao tentar a atualização. Temos que especificar o nulo se a variável estiver vazia.

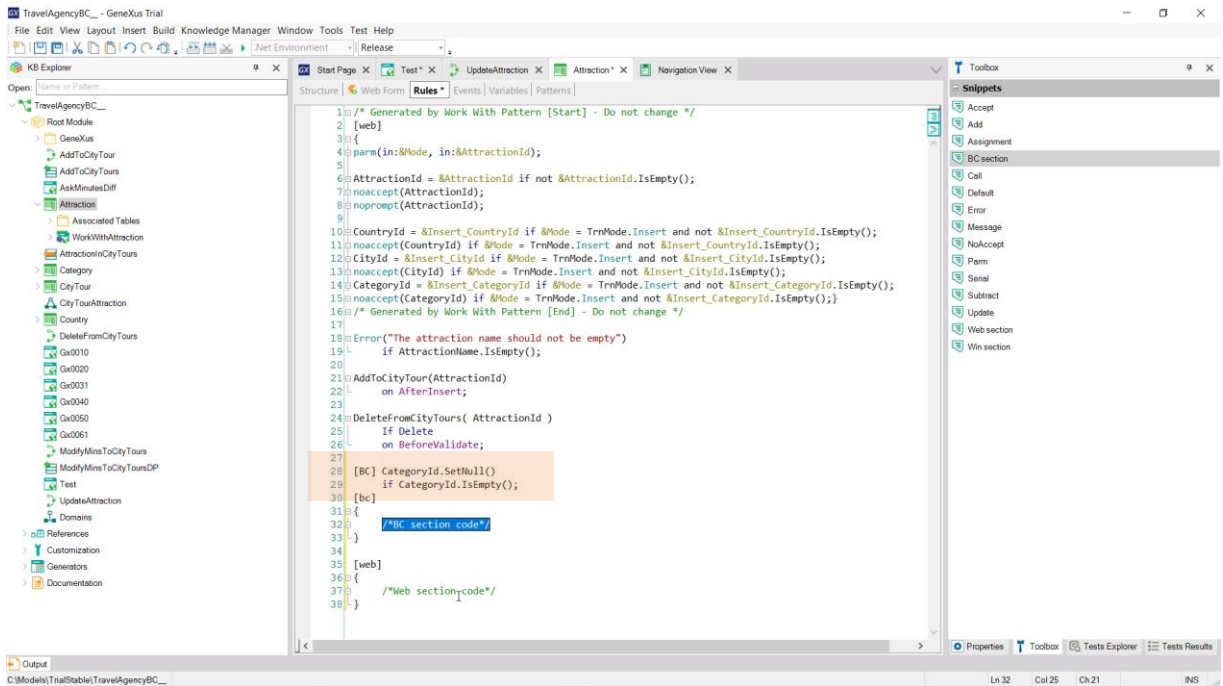
Temos duas maneiras de fazer isto. Ou o fazemos a nível local, só aqui, ou fazemos o mesmo mapeamento que obtém a propriedade do atributo, mas, desta vez, válido também no Business Component, onde quer que se use.

Para o primeiro, conseguirá ao especificar que se a variável CategoryId estiver vazia, ao elemento do Business Component lhe configure o nulo. O método SetNull aplica-se a atributos que aceitem nulos como a seus correspondentes nas variáveis Business Component, como é este caso.

Se testarmos agora, veremos que conseguimos.

A desvantagem desta alternativa é que, se em outro lado quisermos deixar nula esta chave estrangeira através de uma variável Business Component, teremos que fazer o mesmo.

A outra opção, mais geral, que dissemos, é incorporar este SetNull como regra na transação. Aqui o removemos então...

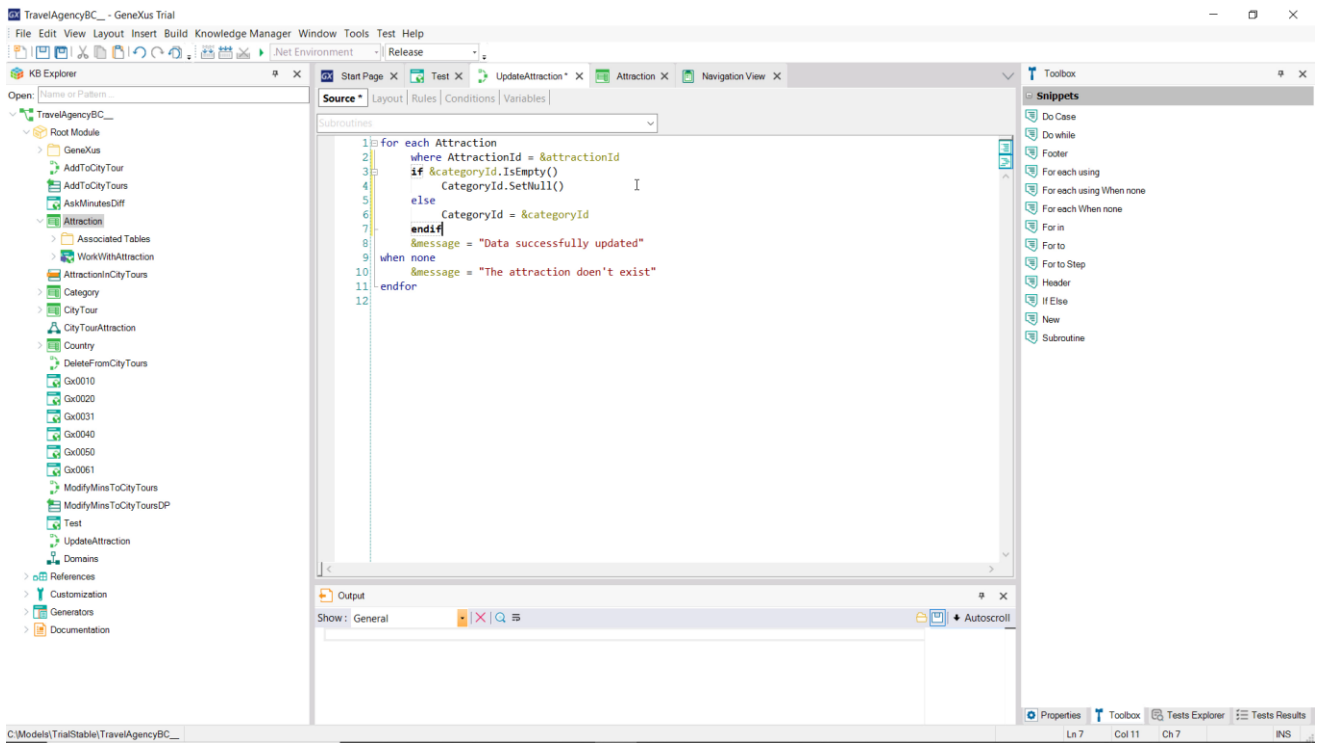


E vamos à transação e especificamos que atribua nulo ao atributo quando seu valor for deixado vazio. Isto mesmo já é o que por padrão se realiza quando se utiliza a tela da transação, graças a esta propriedade, por isso, estritamente falando, podemos pedir que esta regra só se execute para o Business Component.

Sempre podemos indicar que uma regra só se execute quando se trata da transação Web, ou quando se trata do Business Component, colocando esta marca.

Na verdade, se queremos escrever um conjunto de regras que só se aplicam à transação Web, ou que só se aplicam ao Business Component, o fazemos assim.

Testemos. Vemos que efetivamente é outra solução.



A solução aqui só pode ser a local. Testemos.

E vemos que conseguimos.

*GeneXus*TM

training.genexus.com
wiki.genexus.com