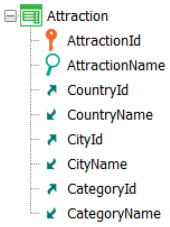
Atualização de base de dados

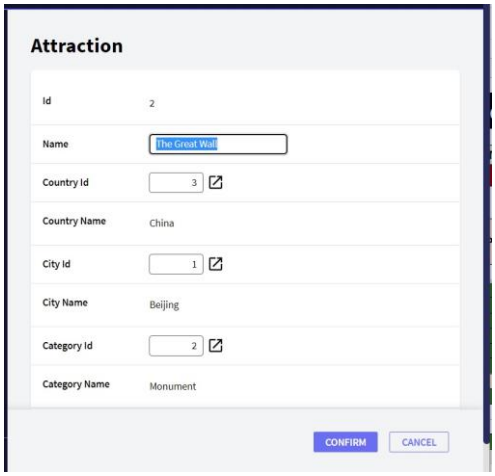
O que acontece quando uma chave estrangeira aceita nulos?

GeneXus™

1. Update through a form

Insert, Update, Delete





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 Compontent**: Insert(), Update(), Delete()

Insert, Update, 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

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

Para atualizar por código tínhamos duas possibilidades:

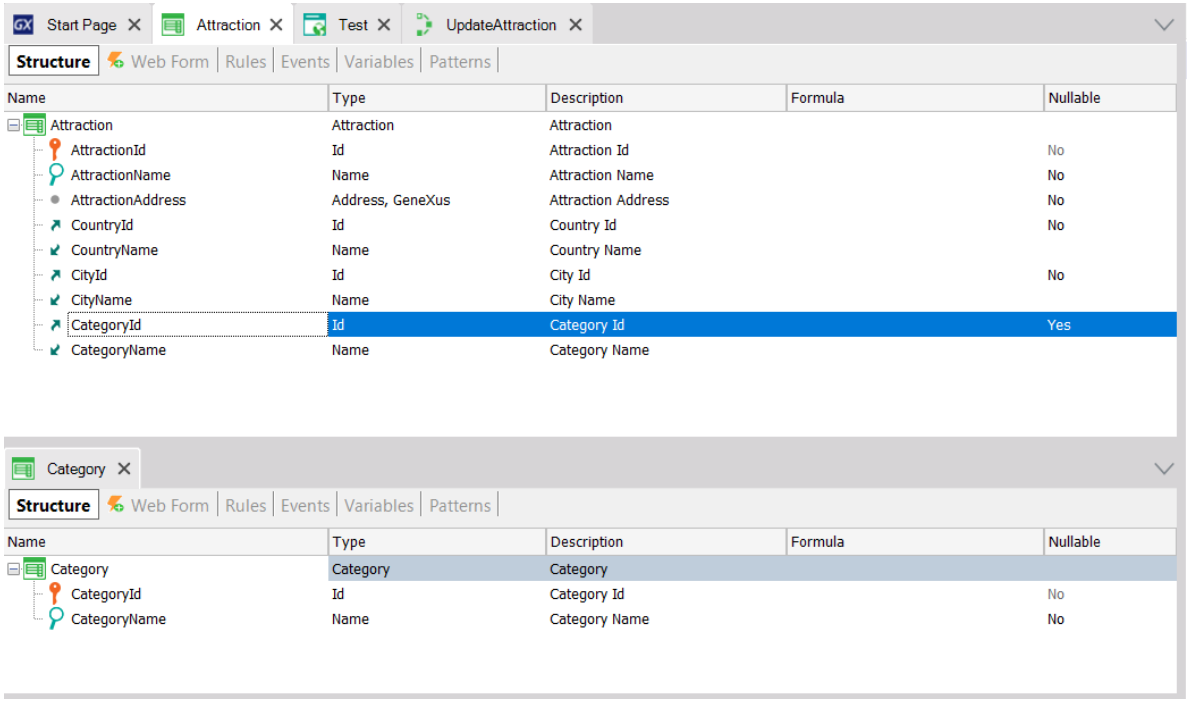
Fazê-lo utilizando o business component da transação, através de seus métodos, ou fazê-lo exclusivamente dentro de um procedimento, através dos comandos New, For each com atribuição direta dos atributos a serem modificados, e o comando Delete.

	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

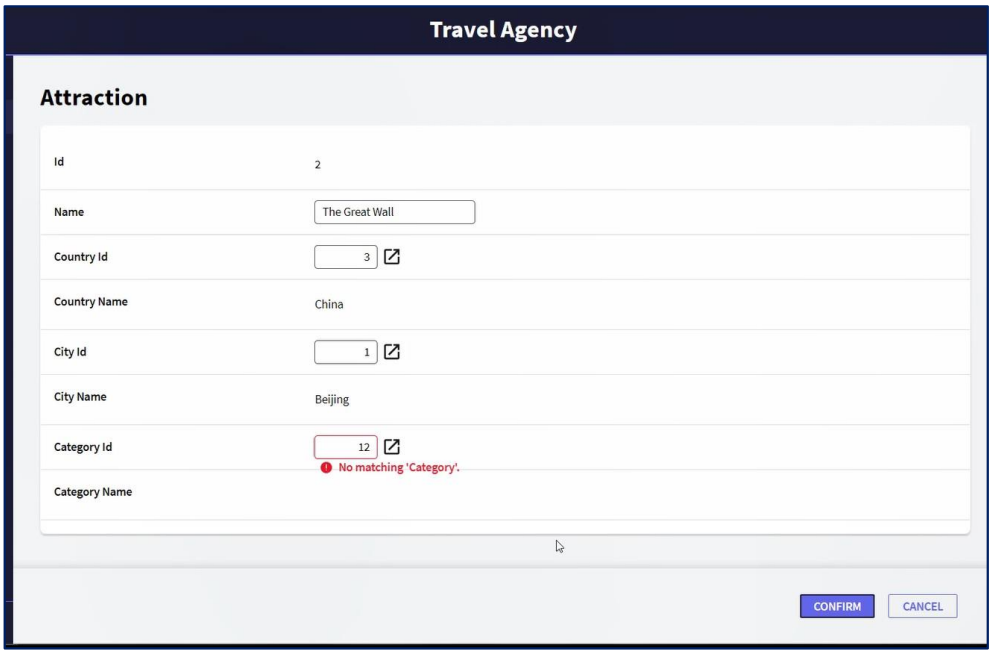
Como sabemos, com ambas as opções se controla a unicidade de registros, ou seja, nunca se permite deixar na base de dados registros com chave primária ou candidata repetida. No exemplo, nunca serão permitidas duas atrações com o mesmo identificador. E se, por exemplo, AttractionName fosse chave candidata, 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”?

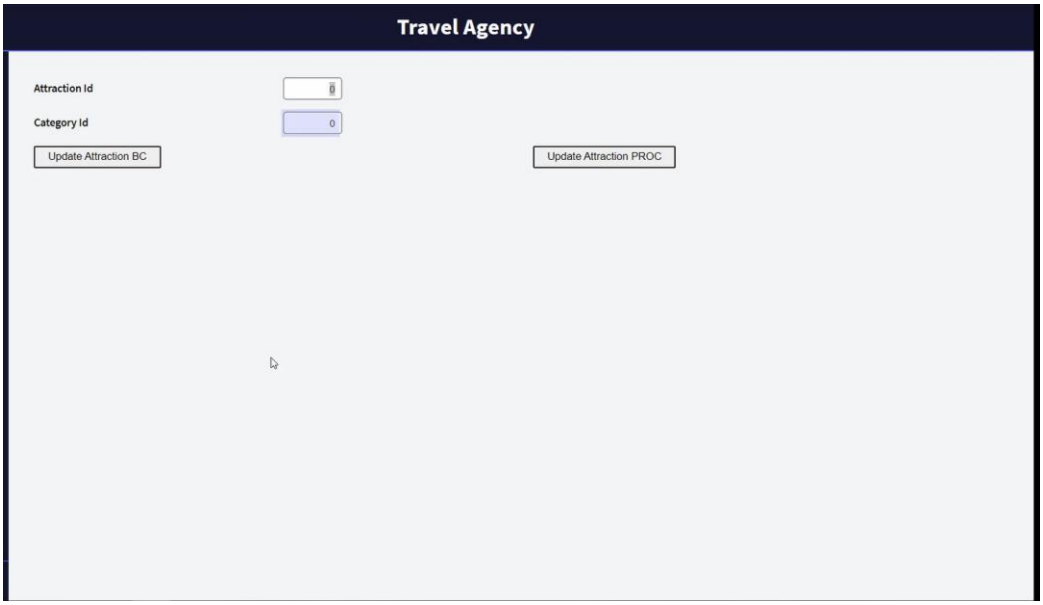


Se temos a transação Attraction que registra as atrações turísticas que sempre pertencem a um país e cidade e a alguma categoria (como monumento, museu, etc.), então, claramente na tabela associada teremos chaves estrangeiras. Em particular, o atributo CategoryId será chave estrangeira para a tabela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

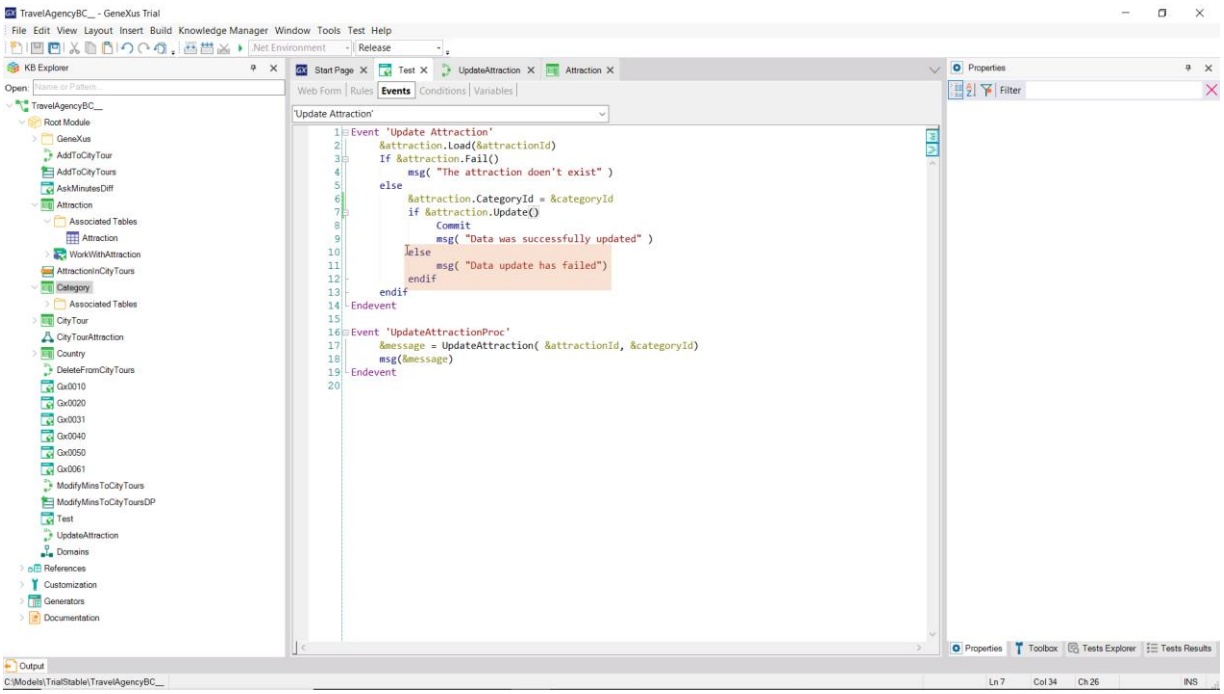


Assim, quando o usuário quer atribuir a uma atração existente uma categoria inexistente, a transação nos informa que isso não é possível. Está fazendo o controle de integridade referencial, antes que a própria base de dados o faça.



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.

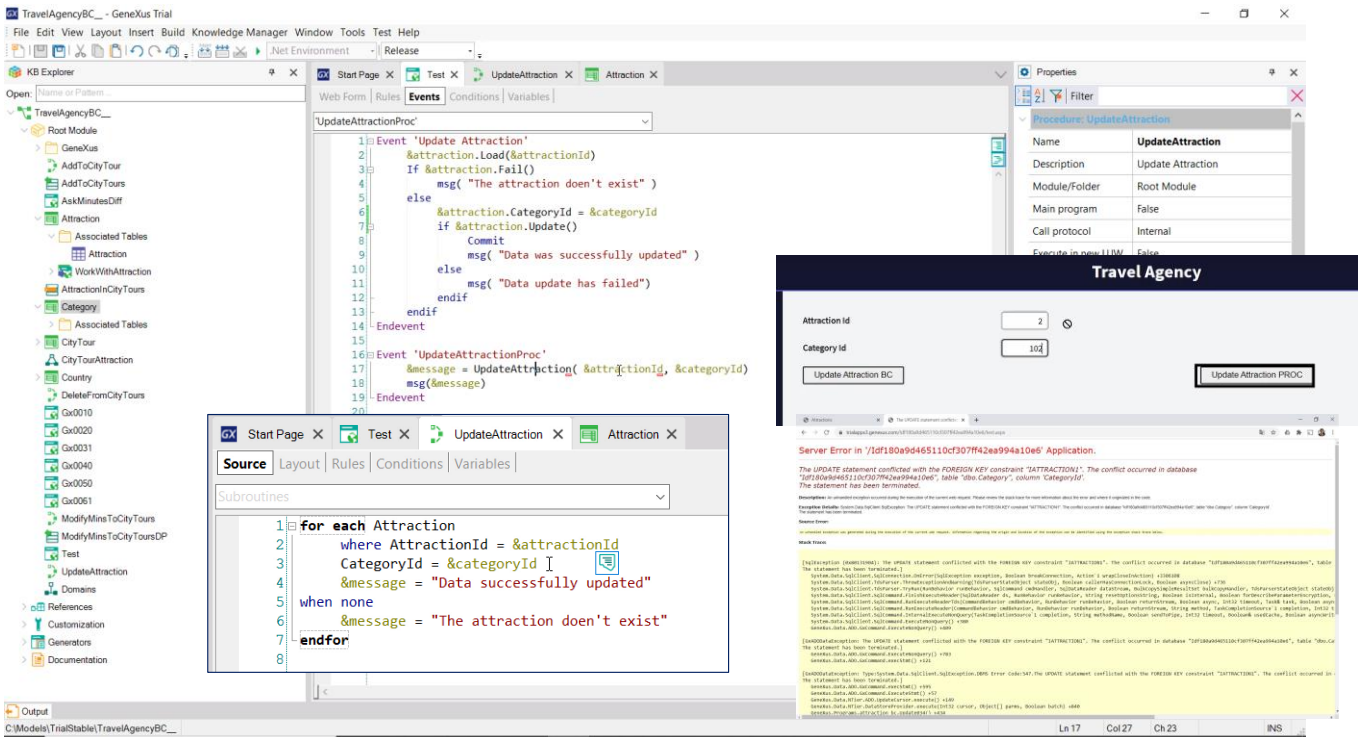


Vejamos como implementaríamos a primeira.

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".

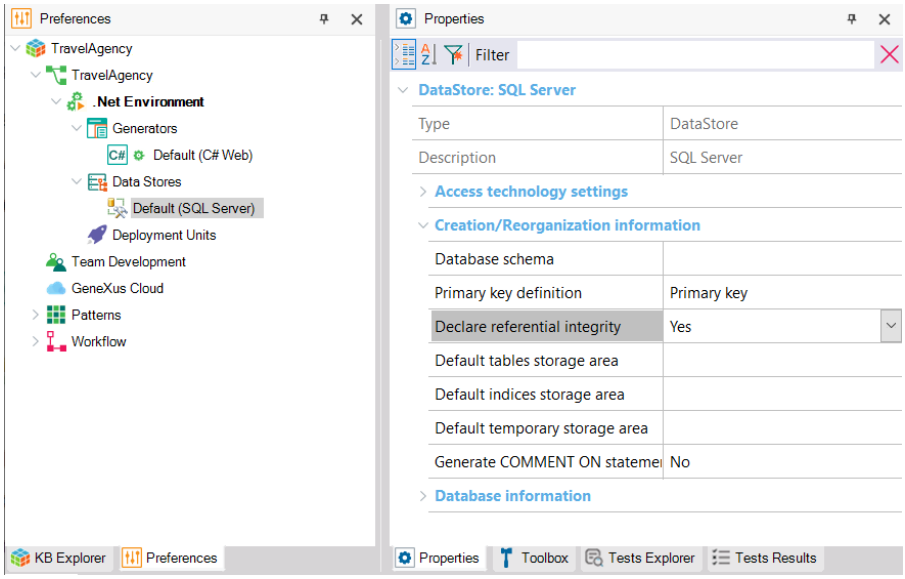
Testemos em execução.



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.

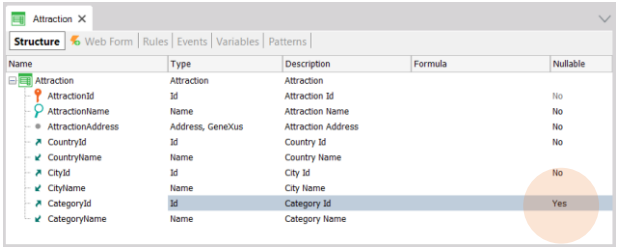
Poderíamos capturar o erro para fazer algo mais amigável e que o programa não cancele abruptamente. Para isso contamos com a regra Error_handler. Como parâmetro devemos dar o nome de uma sub-rotina que devemos definir no objeto e que será onde programaremos o comportamento em caso de erro de base de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



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

If select CateogryId 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

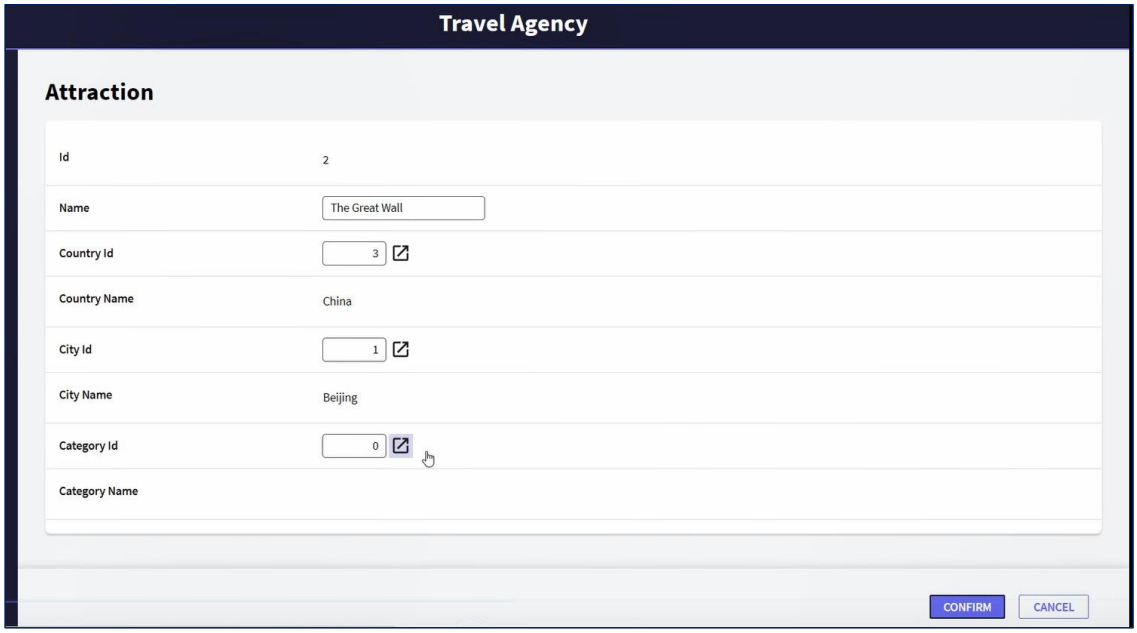


AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	null
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2

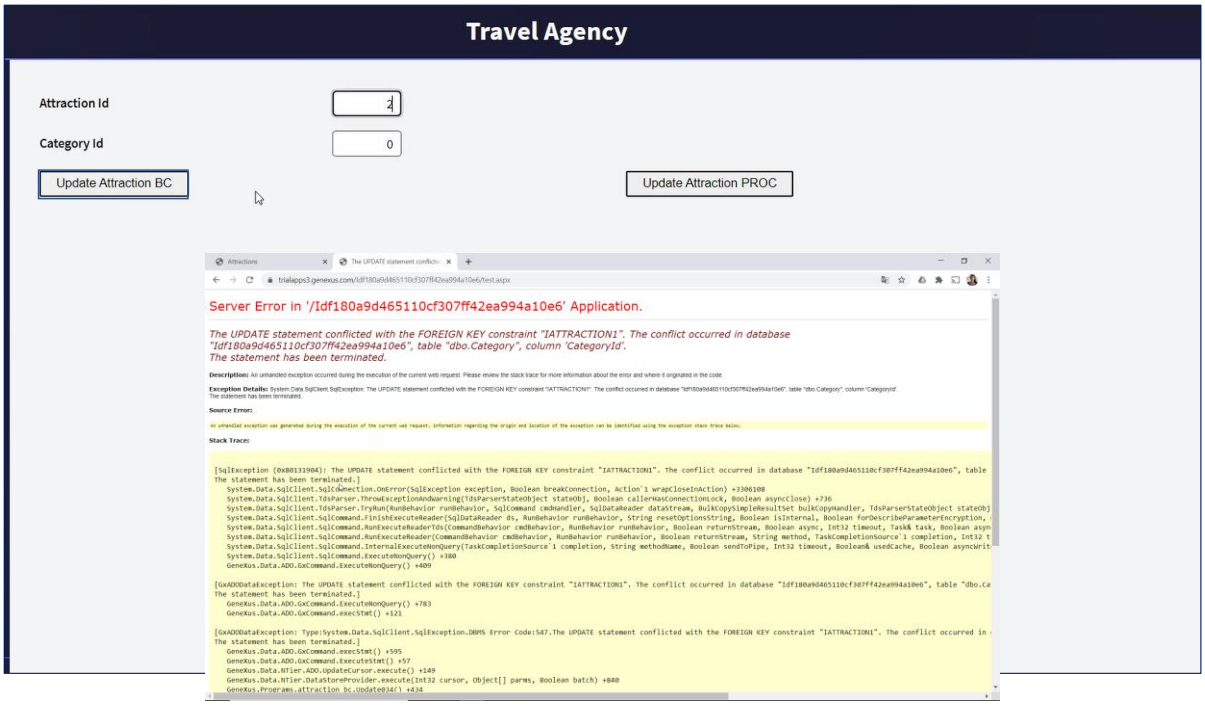
Por padrão, então, se não modificamos essa propriedade, a integridade referencial poderá ou não ser controlada programaticamente, mas sempre será por base de dados.

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.



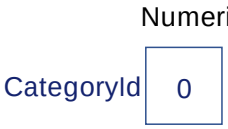
Isto fazia com que, quando executávamos a transação e deixávamos vazio o identificador de categoria, não só o programa não controlava a integridade referencial, como também não o fazia a base de dados.

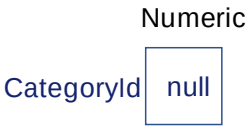


No entanto, se tentarmos fazer o mesmo, mas através do Business Component, auch! Está nos dando basicamente a mesma exceção de base de dados que antes, por falha de integridade referencial. O que está acontecendo?

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

empty ≠ null





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

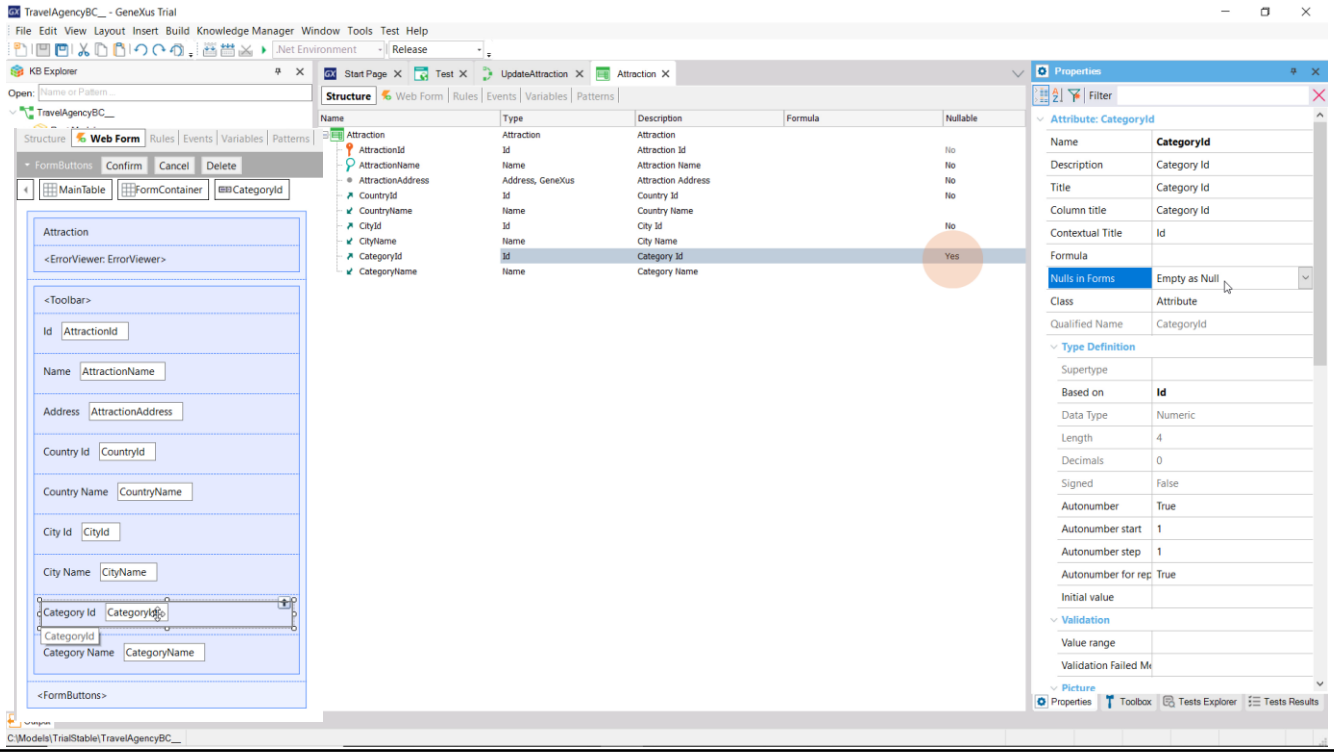
AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2

For AttractionId = 2 → CategoryId = null

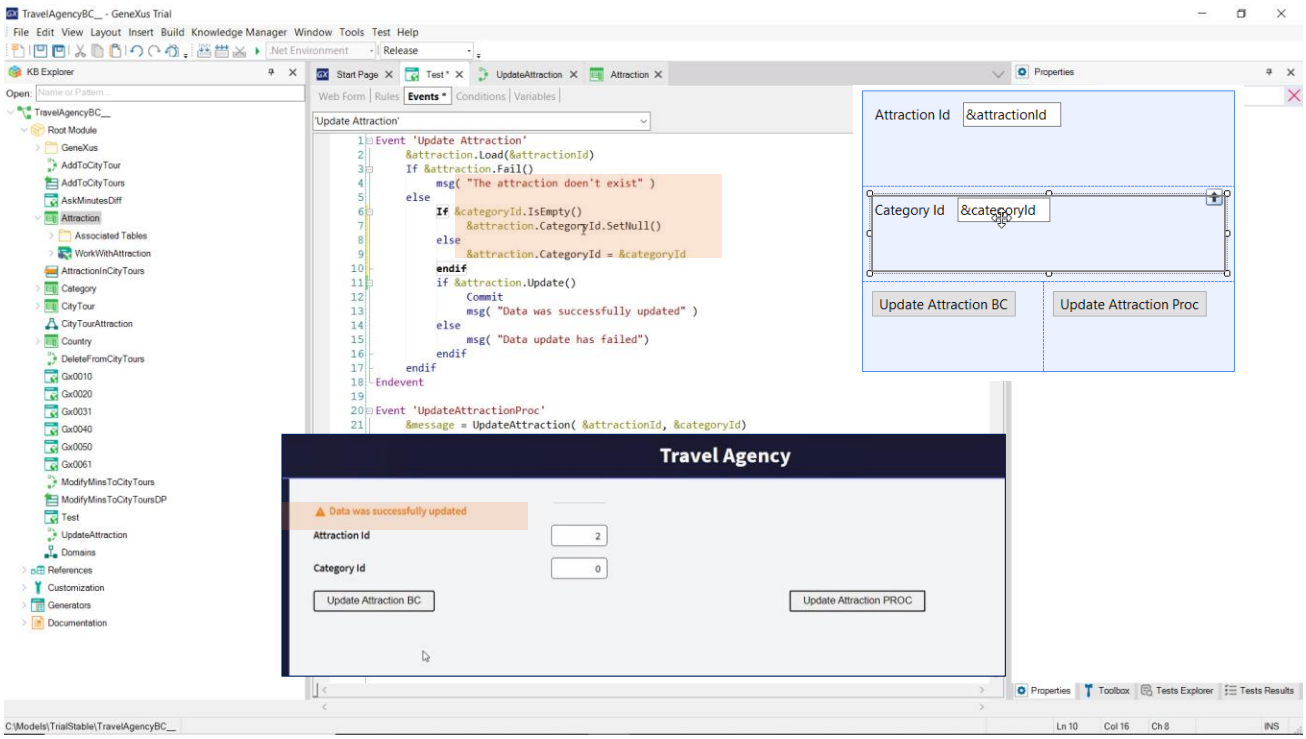
AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	null
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2

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?



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.



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

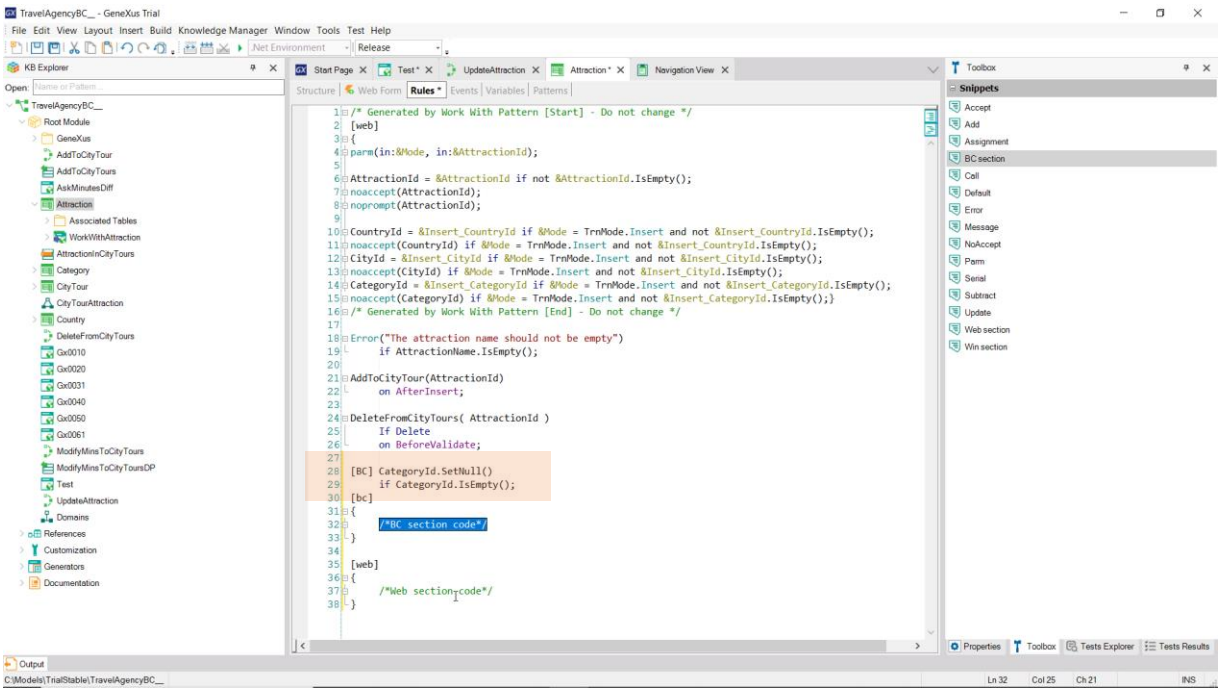
Portanto, se esta variável &CategoryId está vazia, este elemento ficará com valor vazio e não nulo, e é por isso que a integridade referencial falha ao tentar a atualização. Temos que especificar o nulo se a variável estiver vazia.

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.

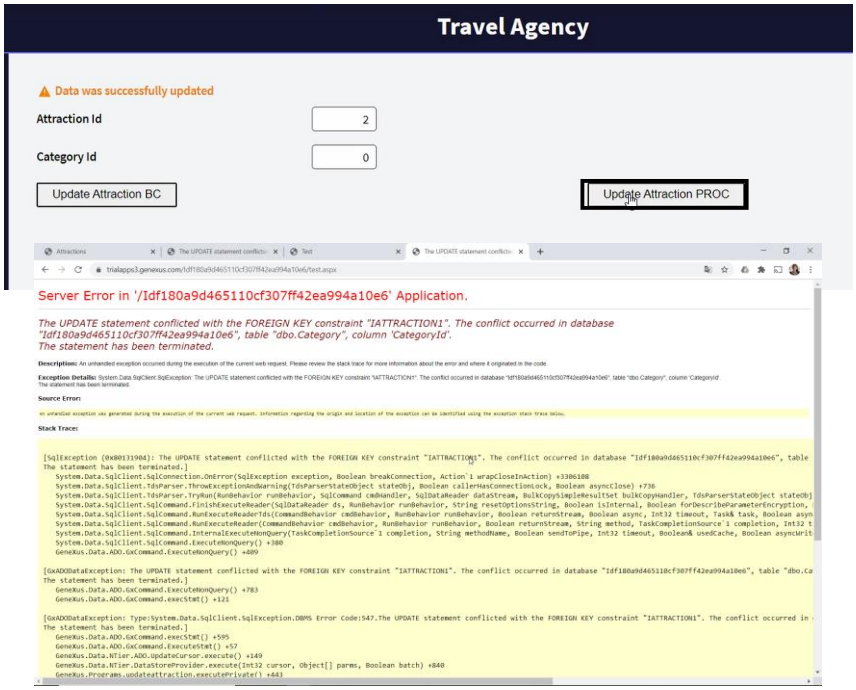
A desvantagem desta alternativa é que, se em outro lado quisermos deixar nula esta chave estrangeira através de uma variável Business Component, teremos que fazer o mesmo.
A outra opção, mais geral, que dissemos, é incorporar este SetNull como regra na transação. Aqui o removemos então...



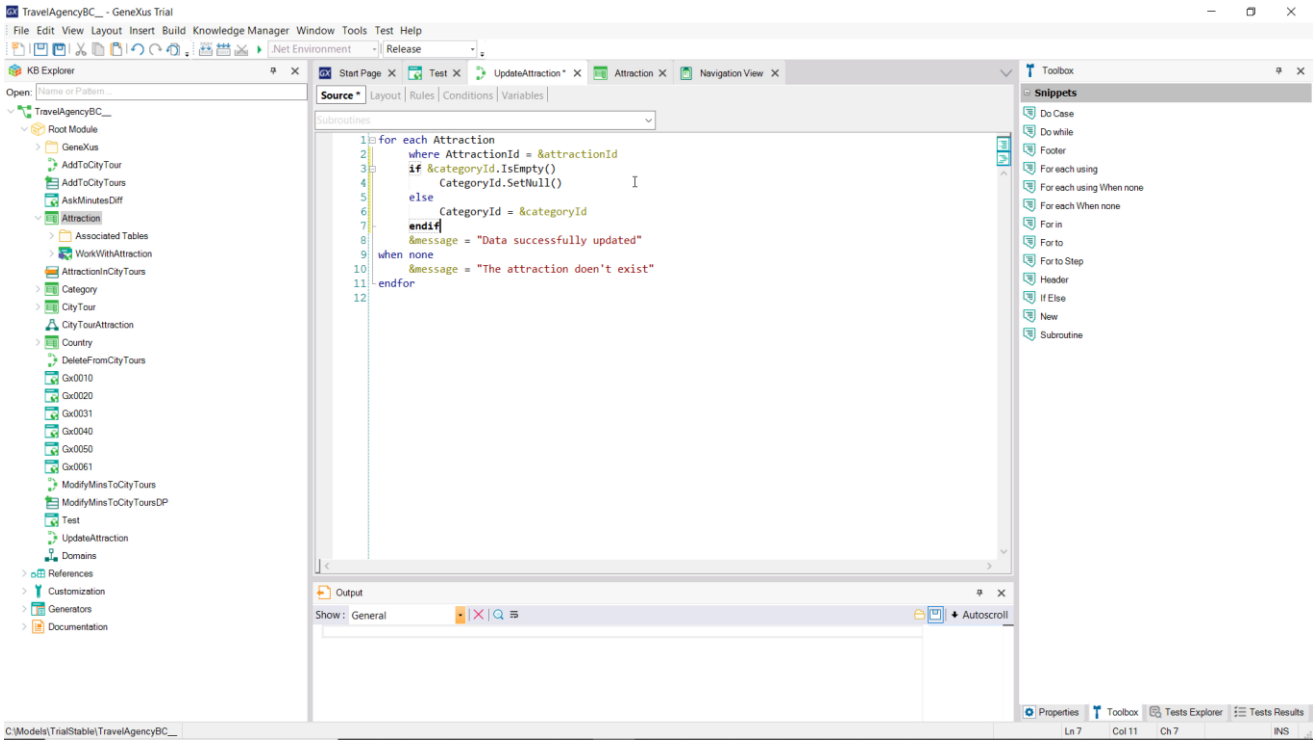
E vamos à transação e especificamos que atribua nulo ao atributo quando seu valor for deixado vazio. Isto mesmo já é o que por padrão se realiza quando se utiliza a tela da transação, graças a esta propriedade, por isso, estritamente falando, podemos pedir que esta regra só se execute para o Business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



Por último, o que acontece agora com a atualização através do procedimento? Temos exatamente o mesmo problema.



A solução aqui só pode ser a local. Testemos.

E vemos que conseguimos.

GeneXus™

training.genexus.com
wiki.genexus.com