

Atualização do banco de dados usando Business Components

Como atualizar os dados como uma transação, porém via código, sem sua tela

GeneXus 16

Cenário

- Usamos as transações para INS, UPD, DLT, através de sua tela.

The image displays 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' (labeled 'Upd') and 'DELETE' (labeled 'Dlt') buttons.

- Contamos com outra forma de fazer isso: os Business Components (BCs)

Anteriormente, havíamos usado as transações para inserir, alterar ou excluir registros no banco de dados, mas sempre através de uma tela, com os botões e controles disponíveis para isso.

Agora veremos como podemos realizar as mesmas operações, mas usando comandos que poderão ser acionados por qualquer objeto GeneXus, utilizando o conceito de Business Components.

Novas requisições

- Nova categoria "Tourist site" para representar atrações muito visitadas, independente se do seu tipo (monumentos, museos, etc).
- Para todas as atrações de Beijing cuja categoria seja "Monument", altera-la para "Tourist site"

Vejamos um exemplo. Vamos supor que em nossa agência de viagens é identificado que algumas atrações são tão visitadas pelos turistas, que queremos classificá-las de outra maneira, para organizar excursões especialmente para estes lugares.

Por exemplo, em Beijing, dada a popularidade de seus monumentos, não queremos mais categorizar os monumentos como "Monument", mas como "Tourist site". Essa será uma nova categoria que criaremos para identificar as atrações que são muito concorridas, independente de como estavam classificadas anteriormente.

Vamos precisar criar essa categoria e passar todas as atrações de Beijing que estavam na categoria "Monument" para a nova categoria "Tourist site"

Acrescentamos uma regra Error

The image displays four screenshots from the GeneXus IDE, illustrating the process of adding an Error rule to two transactions: Category and Attraction.

Category Transaction Structure:

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

Category Transaction Rules:

```

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

Attraction Transaction Structure:

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

Attraction Transaction Rules:

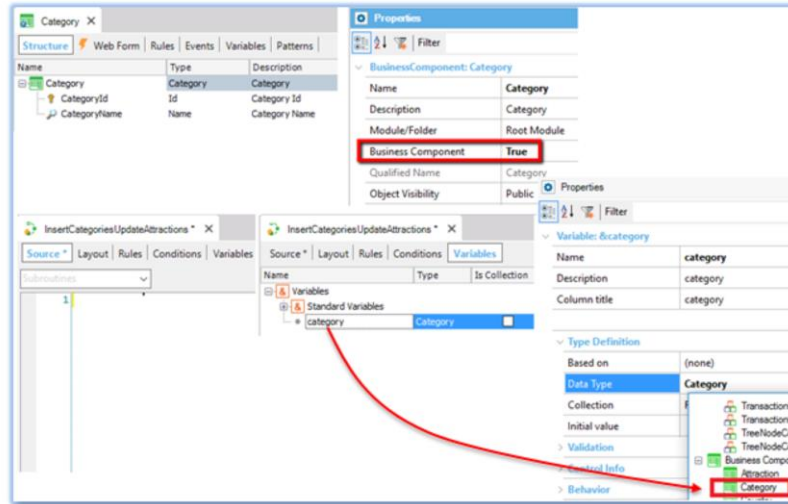
```

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

Para isso, deveremos inserir uma nova categoria na tabela CATEGORY e depois percorrer as atrações da tabela ATTRACTION, filtrando por Cidade = "Beijing" e por Categoria = "Monument", para alterar o código da categoria atual pelo código de "Tourist site".

Antes de continuar, vejam que foi acrescentada uma regra Error tanto na transação Category como na Attraction, para impedir que o nome fique vazio. Explicaremos mais à frente porque fizemos isso.

Business Component da transação Category



Para implementar a criação da nova categoria e a atualização das atrações, vamos criar um novo objeto procedimento que chamaremos de InsertCategoryUpdateAttractions.

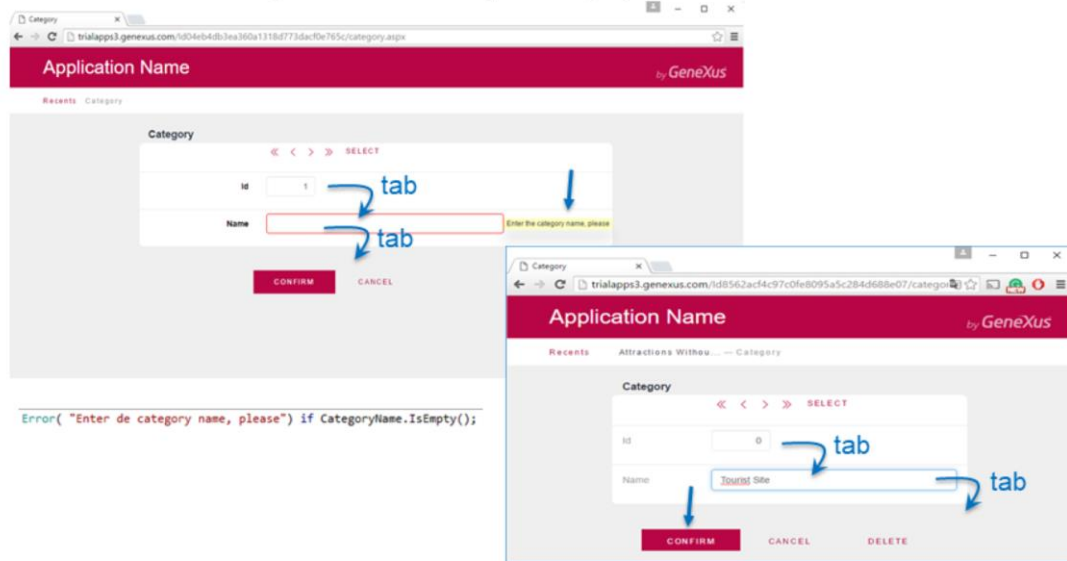
A primeira coisa que temos que fazer é criar a categoria "Tourist site" e, para isso, vamos utilizar um Business Component baseado na transação Category. Para criá-lo, abrimos a transação Category, e definimos o valor da propriedade **Business Component** como True.

Ao fazer isso, GeneXus criará uma nova estrutura a partir da transação Category, que terá funcionalidades similares às da transação.

Para poder usar esta nova estrutura, devemos criar uma variável do tipo Business Component, especificamente de Category. Então, vamos à aba de variáveis do procedimento, criamos uma variável chamada &Category e veremos que GeneXus assume automaticamente o tipo de dados Category. Este tipo de dados corresponde ao Business Component Category, criado a partir da transação Category.

Se formos até as propriedades da variável e clicarmos em Data Type, encontraremos, ao rolar as opções do combo, um grupo denominado Business Component. Expandindo as opções (+), encontraremos o Business Component Category, com um ícone igual ao de uma transação. Isso nos permite confirmar que GeneXus realmente criou um novo tipo de dados baseado na transação Category e veremos que mantém muitas características dela.

Como inserimos um registro com a transação Category



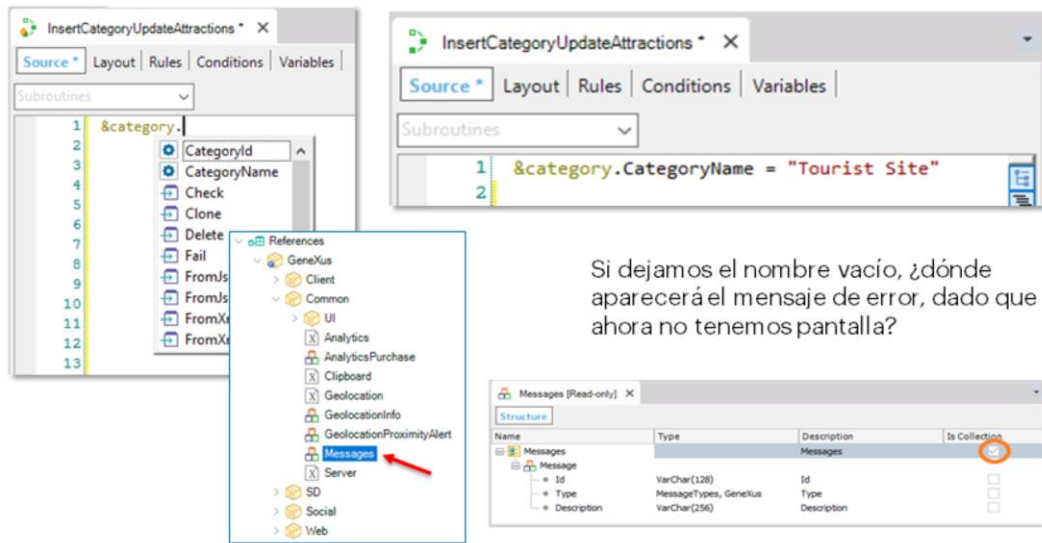
Lembrem-se que, se acessamos a transação Category, por default espera-se que o usuário insira um novo registro.

A primeira coisa que faríamos, como o cursor posicionado em CategoryId, seria informar o identificador da categoria, mas, como esse atributo é auto-numérico, não é necessário atribuir valor a ele. Damos TAB para sair do campo, deixando o mesmo vazio.

O próximo campo corresponde ao nome da Categoria. O que acontece se deixamos este campo vazio? Ao sairmos do campo, a regra Error que controla se o atributo CategoryName está vazio será disparada.

No nome da Categoria vamos escrever "Tourist site" e, por último, uma vez informados os valores que queremos, clicamos no botão Confirmar, para que os dados sejam salvos no banco de dados e, assim, a inserção do registro seja concluída.

Como inserimos um registro com o Business Component Category



Quando utilizamos um Business Component, a sequência de operações é exatamente a mesma. Para inserir um novo registro, o que faremos será informar valores aos campos da Categoria e confirmar as alterações.

Para isso, incluímos a variável `&category`, no Source, e pressionamos ponto (.). Vemos que aparecem vários métodos disponíveis (como Save, Delete, Check) além dos atributos `CategoryId` e `CategoryName`, para os quais podemos atribuir algum valor.

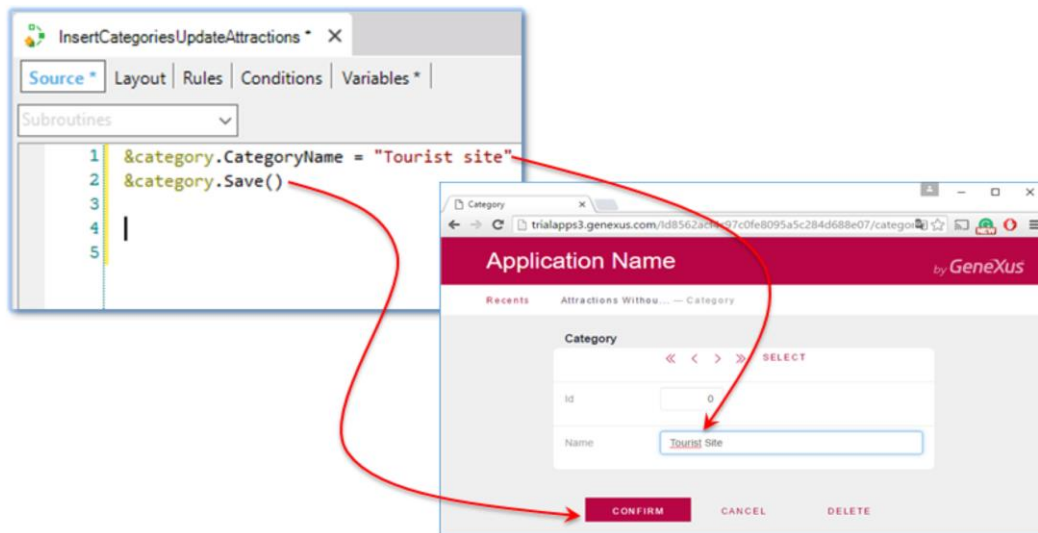
Como sabemos que o identificador de categoria é auto-numérico, não vamos atribuir valor a ele. Uma vez que não definimos o valor do identificador, o BC se comportará como a transação e entenderá que estamos realizando uma operação de **INSERÇÃO**. Em seguida, assim como fizemos na tela da transação, informamos o nome da Categoria.

O interessante de trabalhar com Business Components é que as regras definidas na transação que originou o BC também serão disparadas. Dessa forma, se, no nosso exemplo não tivéssemos informado valor para o nome da Categoria, teria sido disparada a regra Error que controla isso, exatamente como aconteceu quando usamos a tela da transação Category. A única diferença é que no Business Component, a regra será disparada somente quando pedirmos ao BC que grave as informações.

Precisamos deixar claro que nem todas as regras serão disparadas, já que, como estamos executando o Business Component via código, não poderá ser disparada nenhuma regra associada a algum objeto que contenha telas (por exemplo, uma outra transação ou um Web Panel), Também não será disparada a regra Parm que foi definida na transação.

Está claro, então, que, se deixamos a Categoria sem nome, a regra Error será disparada e o novo registro não será gravado. Porém, onde essa mensagem de erro irá aparecer, uma vez que agora não temos uma tela? Em toda Base de Conhecimento, GeneXus cria, por padrão, uma SDT chamada Messages que é uma coleção de itens, justamente para ajudar nesse tipo de situação.

Como inserimos um registro com o Business Component Category



Continuando com nosso exemplo, se estivéssemos trabalhando com a tela da transação, depois que informamos o nome da categoria, devemos pressionar o botão Confirmar. Com o Business Components, usamos o método Save():

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

Para cada variável Business Component (como &category), quando é realizada uma operação de Save, uma coleção de mensagens é carregada em memória, com todas as advertências ou erros resultantes dessa operação.

Temos maneiras de recuperar essa coleção e percorre-la, de forma muito simples, mas não veremos isso agora.

Como alterar a categoria da atração 2, "Muralla China"

Business Component vs Transação

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

```

Aqui estamos usando o valor definido para o id da categoria que criamos antes, com o BC

Até aqui já criamos a categoria "Tourist site". O que faremos agora é alterar todos os monumentos da China para essa nova categoria.

Vamos supor que temos que alterar somente a categoria da Muralla China, que tem o id= 2. Se fosse na transação, como teríamos que fazer? Abrimos a transação Attraction, informamos o valor 2 para o campo Id, saímos do campo, e GeneXus nos traz todos os dados da Muralla China.

Em seguida, alteramos os valores que nos interessam (neste caso, a categoria) e salvamos (pressionando Confirm).

Ao confirmar, é iniciado o processo de gravação. Nesse momento, são disparadas todas as regras que se aplicarem em alguma situação. Por exemplo, se tivéssemos apagado o nome da atração, a atualização teria falhado. Ainda, se tentássemos alterar o Id da categoria por um valor inexistente, como GeneXus controla a consistência das informações entre tabelas relacionadas (integridade referencial), aconteceria um erro e a modificação não seria realizada.

Este mesmo comportamento ocorrerá com o Business Component.

Para acessarmos uma determinada atração, utilizamos o comando Load e entre parenteses, colocamos o id da atração desejada, neste caso, a 2.

Em seguida, informamos o novo valor da categoria (que é o valor do id da categoria "Tourist site", que criamos anteriormente) e finalmente fazemos um Save().

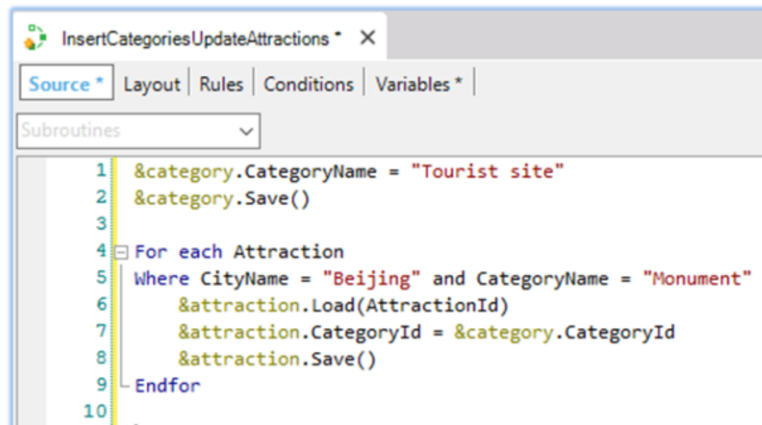
```
&Attraction.Load(2)
```

```
//Aqui estamos usando o valor gerado para o id da categoria que criamos antes com o BC.
&Attraction.CategoryId = &category.CategoryId
```

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

Ao utilizarmos o comando Load, o Save identificará que estamos querendo alterar o registro e não inserir.

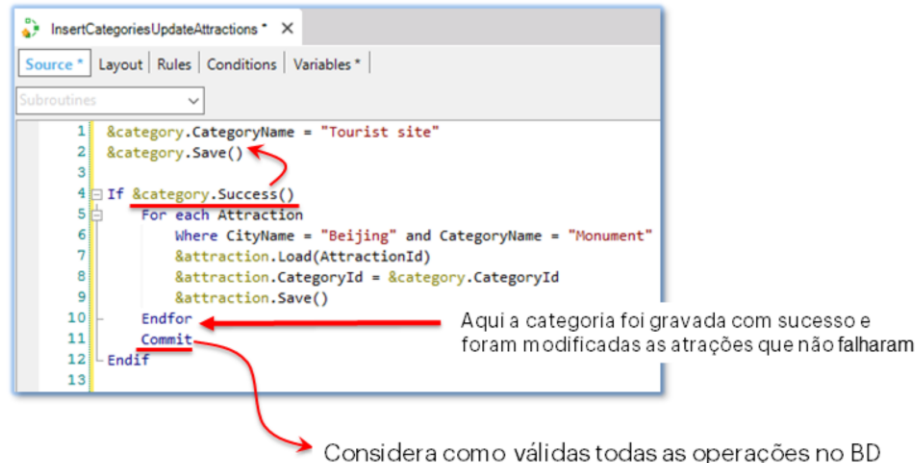
Como alterar a categoria de todas as atrações de Beijing que eram Monuments... com o Business Component Attraction



Porém nosso objetivo não é alterar somente a categoria da Muralla China, mas todas as atrações de Beijing que sejam monumentos. Para isso devemos percorrer todas essas atrações.

Como agora estamos percorrendo registros com um for each, faremos o Load do Id de cada atração percorrida.

Como saber se a gravação do Business Component teve sucesso e como considerar as operações como válidas no banco de dados



Como cada Save() pode ou não ter sucesso, dependendo das regras de negócio ou da integridade referencial, os registros também podem ter sido inseridos/modificados ou não no banco de dados.

Para sabermos se um comando Save() teve ou não sucesso, contamos com o método Success(), que nos devolve TRUE se foi bem sucedido e FALSE, caso contrário. O comando Save() pode falhar por causa do disparo de uma regra Error, ou porque houve uma queda do sistema no momento de salvar e o registro acabou sendo perdido. Ou ainda, porque houve alguma falha na integridade referencial dos dados.

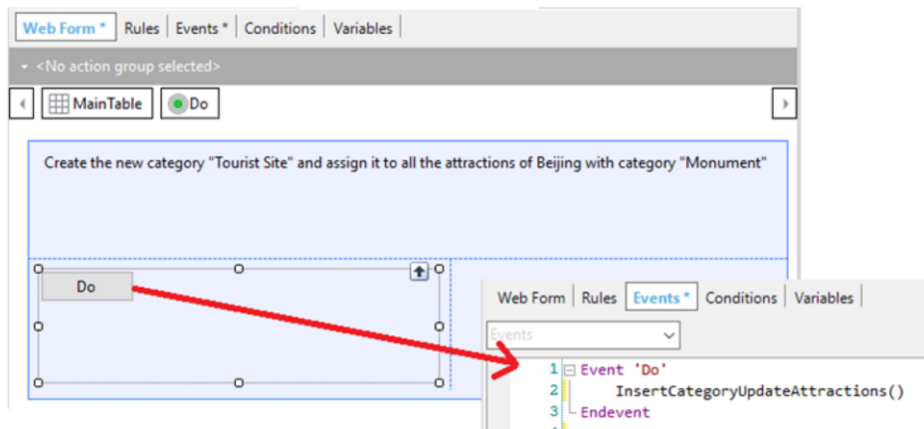
Se a inserção da nova categoria falhar, evidentemente o Save das atrações também falhará, porque essa categoria não existirá. Assim sendo, devemos condicionar a modificação das atrações ao resultado do método Success().

Só estaria faltando, então, informar ao banco de dados que os registros inseridos/alterados estão corretos e que devem ser gravados definitivamente, de tal forma que não se percam mesmo que ocorra uma queda do sistema. Para isso, usaremos o comando Commit.

Se chegamos até esse ponto do código, sabemos que a Categoria e as modificações nas atrações foram gravadas com sucesso (todas aquelas que não deram erro, por alguma regra de negócio, por exemplo). Agora necessitamos que o banco de dados entenda que todas as atualizações são válidas. Inserimos um registro e modificamos outros. Se houvesse uma falha do sistema (por exemplo, um corte de energia elétrica), os bancos de dados se recuperam e desfazem todas as operações que não chegaram a executar um Commit.

O comando **Commit** é um comando especial que nos permite indicar ao banco de dados que deve finalizar um bloco de operações programadas para serem executadas todas juntas (e em caso de falha, que todas se desfaçam). Quando esse comando é executado, o banco de dados salva as informações FISICAMENTE, de maneira definitiva. Depois do Commit, elas não se perdem mais, mesmo que haja uma queda do sistema ou falha de energia.

Chamando o procedimento de dentro de um Web Panel



Para executar nosso procedimento, precisamos que outro objeto GeneXus faça chamada a ele.

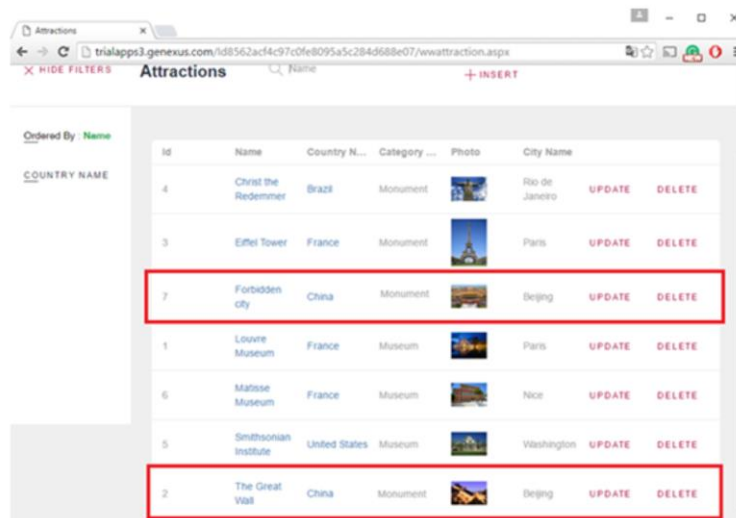
Podemos fazer isso de dentro de um objeto Web panel. Já havíamos visto que um Web Panel é uma tela que podemos desenhar de acordo com nossas necessidades e que servem para muitas coisas, como por exemplo, ver informações do banco de dados, construir a tela de Login do nosso sistema ou para cadastrar dados.








Em nosso caso, vamos criar um web panel com um botão. Quando pressionado, deve executar a chamada do procedimento que criamos anteriormente.

No evento associado ao botão, chamamos o procedimento `InsertCategoriesUpdateAttractions`, que não requer nenhum parâmetro.

ANTES

F5



Id	Name	Country N...	Category ...	Photo	City Name		
4	Christ the Redemmer	Brazil	Monument		Rio de Janeiro	UPDATE	DELETE
3	Effel 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

Executamos a aplicação. Podemos observar que as atrações da China tem a categoria Monument.

Quando vemos os detalhes da informação, confirmamos que ambas estão associadas a Beijing.

DEPOIS

The screenshot shows two browser windows. The left window, titled 'Categories And Attractions', has a 'Do' button. A red arrow points from this button to the 'Attractions' table in the right window. The right window, titled 'Attractions', displays a table with columns: Id, Name, Country Name, Category, Photo, City Name, and actions (UPDATE, DELETE). The table contains the following data:

Id	Name	Country Name	Category	Photo	City Name	UPDATE	DELETE
4	Christ the Redeemer	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

Se abrirmos o web panel CategoriesAndAttractions, e pressionarmos o botão Do, quando voltarmos às atrações, veremos que a atração Tourist Site foi criada e associada às duas atrações que eram de Beijing e que antes pertenciam à categoria Monument.

Vemos que, no caso da categoria, GeneXus deduziu que teria que fazer um Insert e em seguida, quando alteramos o valor da categoria da atração, identificou que se tratava de um Update, exatamente como faz uma transação. É por isso que no procedimento utilizamos o método Save(), que tem a inteligência de gravar os dados, sem se importar com qual operação está sendo realizada.

Métodos Insert() e Update() para especializar o 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

```

Diagram illustrating the code flow with annotations:

- A red arrow points from `&category.Insert()` to the `Where` clause in the `For each Attraction` loop.
- A red arrow points from `&attraction.Update()` to the `&attraction.AttractionId = AttractionId` line.

Para tentar um Insert e, se falhar, um Update: `&attraction.InsertOrUpdate()`

`InsertOrUpdate()` – Tenta um insert e, se não consegue, tenta um update

Vimos que o método `Save()` tem a inteligência de gravar a informação, sem se importar se trata-se de uma inclusão ou uma alteração dos dados.

Mas também contamos com os métodos `Insert()` e `Update()` que nos permitem indicar explicitamente a operação que desejamos realizar.

Vamos modificar o código para utilizar estes métodos, `Insert` para o caso das categorias e `Update` para as atrações.

Podemos notar que, no caso das atrações, não havia necessidade do `Load` e poderíamos ter escrito diretamente:

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

Já que declaramos explicitamente que queremos fazer um `Update` de atrações, GeneXus sabe que queremos atualizar um registro.

Além das operações `Insert()` e `Update()` para indicar que queremos que seja realizada uma inclusão ou uma modificação especificamente, contamos também com a operação `InsertOrUpdate()`, onde, ao atribuir novos valores aos atributos, GeneXus tentará fazer um `Insert`, e se não conseguir (porque, por exemplo, já existe um registro com essa chave), tentará realizar um `Update`.

Inserindo e atualizando o BD de um 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()
endfor

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

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

Outra coisa importante de ressaltar, como vimos, é que é possível incluir e alterar registros de duas tabelas distintas dentro do mesmo objeto (neste caso, um procedimento). Também poderíamos ter colocado este código diretamente no evento do web panel, porque as atualizações no banco de dados através de Business Components podem ser executadas por qualquer objeto (com algumas restrições no caso das transações).

Se quiséssemos desfazer o que fizemos, acrescentamos um botão “Undo” no web panel e nele, teríamos que apagar a categoria “Tourist site” recém-criada, alterar a categoria das atrações que havíamos mudado para “Tourist site” e volta-las para “Monuments”.

Como faríamos para apagar a categoria através da transação? Primeiro a selecionamos, informando seu ID. Em seguida pressionamos o botão **Delete**. Usando os business components será igual. Usaremos o método Delete().

Se não soubéssemos o ID, podemos busca-lo usando a fórmula Find. A fórmula Find é uma fórmula inline que nos permite encontrar o primeiro registro que atenda a condição indicada, e desse registro, devolve o valor do atributo especificado.

Se tentarmos excluir a categoria “Tourist Site” através da transação, acontecerá um erro, porque existem atrações que estão associadas a essa categoria.

O Business Component fará exatamente o mesmo, controlando a integridade referencial no momento em que executarmos o método Delete(). Não é possível apagar a categoria “Tourist Site” porque existem atrações associadas a ela.

Portanto, antes de excluir essa categoria, teremos que alterar a categoria das atrações vinculadas a ela, voltando-as para a categoria que tinham anteriormente: “Monument”.

Como quase sempre não nos lembramos dos valores dos identificadores mas sim dos nomes, é mais seguro, em vez de confiar em nossa memória, fazermos uma busca no banco de dados, filtrando por nome. Assim, em nosso caso, para recuperar o valor do Id da categoria Monument utilizamos a fórmula Find.

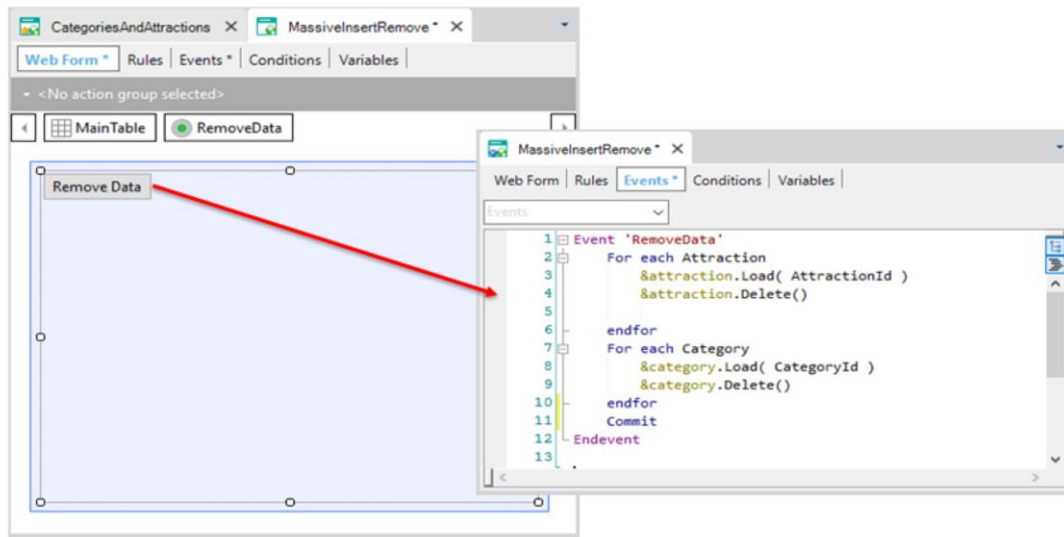
Para cada atração de Beijing com a categoria “Tourist Site” carregamos todos os dados na variável Business Component &Attraction, modificamos unicamente o valor de CategoryId,

atribuindo-lhe o Id correspondente a “Monument” e atualizamos.

Agora sim podemos eliminar a categoria “Tourist Site”. E, por último, não podemos esquecer do Commit, para confirmar que as exclusões sejam efetivadas definitivamente no banco de dados, e não aconteça que, devido a uma queda do sistema, os registros que haviam sido excluídos voltem a existir graças ao rollback que os bancos de dados realizam quando se recuperam.

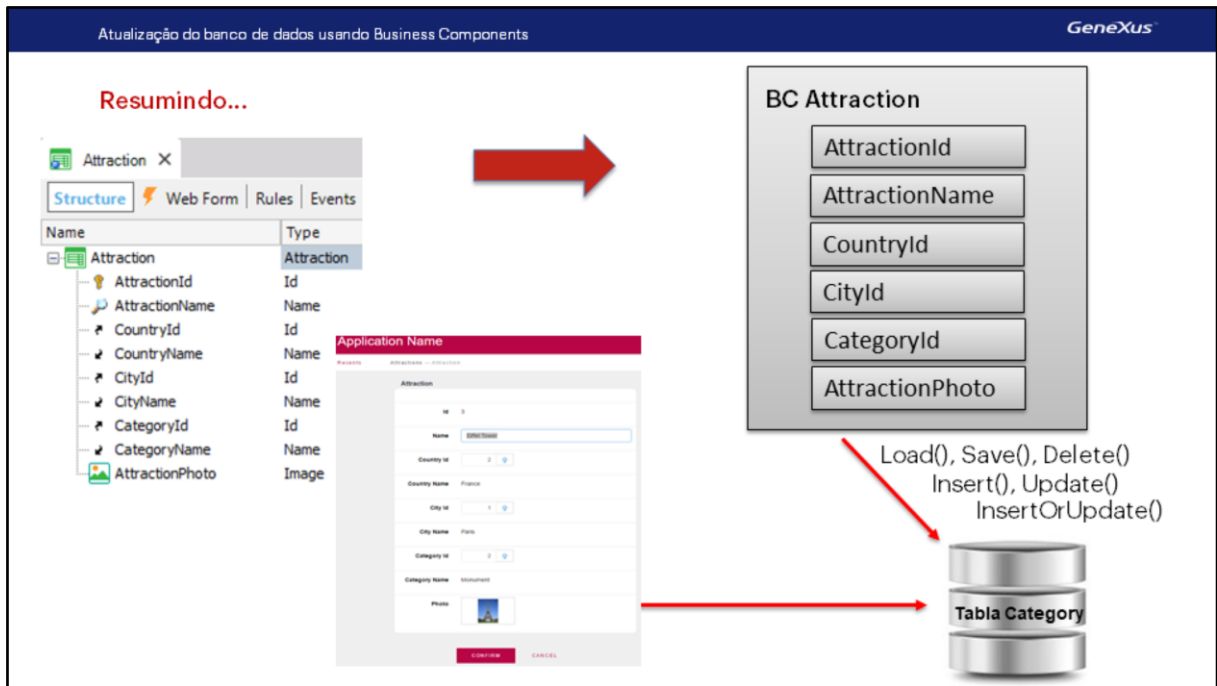
Se alguma das atrações falhar em sua atualização, o Delete de &Category também falhará. Assim, podemos criar uma condição que identifique se a eliminação da Categoria teve sucesso, antes de fazer o Commit. Caso contrário, não queremos que nenhuma modificação no banco de dados seja efetuada. Fazemos isso disparando um Rollback, ou seja, desfazemos tudo o que havia sido feito.

Inserção e exclusão em massa



Notem que se quiséssemos excluir todas as atrações em vez de alterar suas categorias... deveríamos eliminar todos estes registros da tabela ATTRACTION. Isso pode ser feito através de um For each com Delete. Para isso, vamos criar um Web panel chamado "MassiveInsertRemove com um botão "Remove data".

É muito importante a ordem em que as exclusões serão executadas, já que, como sabemos, os Business Components também controlam a integridade referencial. Portanto, não podemos excluir primeiro as categorias.

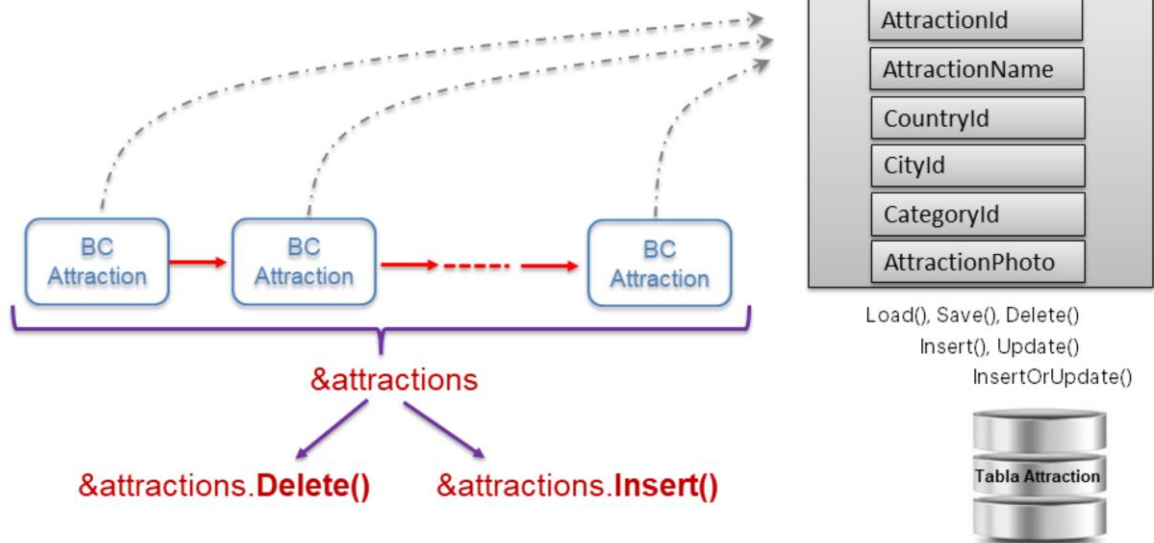


Resumindo o que vimos até agora, aprendemos que o Business Component é uma espécie de tipo de dados especial, gerado a partir de uma transação, como se fosse um espelho dela, ou seja, permite fazer tudo o que a transação faz, com exceção daquilo que se refere a elementos de tela.

É por isso que, depois que definimos uma transação como Business Component, podemos declarar uma variável desse tipo de dados, que se comportará como um SDT, com uma estrutura igual à da transação. E, através dos métodos Load, Save e Delete (que correspondem às ações de Insert, Update e Delete), poderemos realizar as mesmas operações que faríamos nas transações. Com isso, temos a certeza de que as regras de negócio serão disparadas como se estivéssemos executando a transação. Quais regras de negócio? As que não tem relação com elementos de tela, nem fazem chamada a objetos que possuem telas (Web Panels, por exemplo). É por isso que dizemos que executar um Business Component é como executar a transação de forma "silenciosa".

Para uma transação de um só nível como Attraction, a estrutura do Business Component irá conter um elemento para cada atributo físico da tabela associada, mais os atributos como CategoryName, o CountryName o CityName que aparecem na estrutura da transação como inferidos. Eles tem a mesma função que nas transações: poder inferir valores quando executamos o Load.

Operações sobre coleção de BCs



As operações de Insert, Update ou Delete podem ser realizadas também de forma massiva através de uma coleção de BCs, que ainda não sabemos como carregar, mas é como se tivéssemos uma variável do tipo coleção de Business Components. Por exemplo, si tivéssemos todas as atrações carregadas em uma variável &attractions, onde cada item fosse do tipo Business Component Attraction, então poderíamos elimina-as todas de uma só vez fazendo &attractions.Delete() sem necessidade de percorrer a coleção ítem a ítem, para elimina-las de forma individual.

Veremos mais adiante um exemplo que insere registros de forma massiva, com Insert.



Videos

training.genexus.com

Documentation

wiki.genexus.com

Certifications

training.genexus.com/certifications