Atualização da base de dados

Usando Business Component de um nível (REVISÃO)

GeneXus

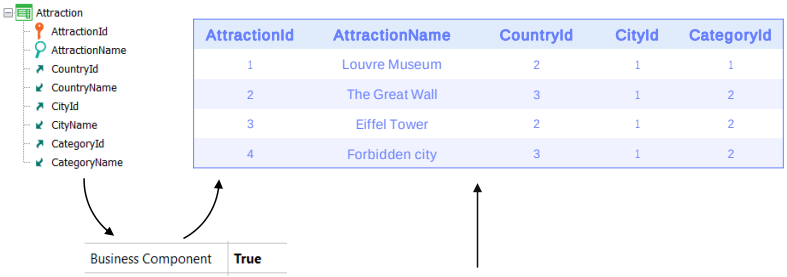
Database update

Save()

1. **Business Compontent**: Insert(), Update(), Delete()

InsertOrUpdate()

Insert, Update, Delete



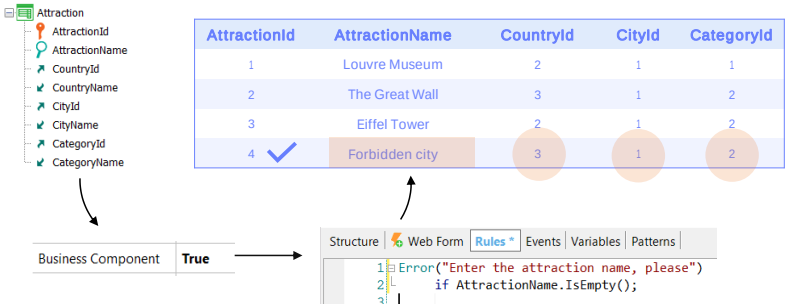
2. **Procedure**: New, For each, Delete

Para atualizar as informações da base de dados por código, tínhamos duas possibilidades:

Fazer isso utilizando o business component da transação, através de seus métodos Insert, Update ou Delete (ou em vez do Insert ou do Update, o Save, ou mesmo o InsertOrUpdate), 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 dentro de um for each para excluir o registro em que se esteja posicionado.

1. **Business Compontent**: Insert(), Update(), Delete()

Insert, Update, Delete



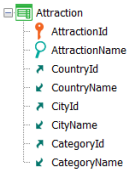
2. **Procedure**: New, For each, Delete

A grande diferença entre estas duas alternativas consistia em que, enquanto a primeira estava fortemente ligada à lógica da transação, uma vez que se disparavam as regras, incluindo o controle de duplicados e de integridade referencial...

Database update

1. **Business Compontent**: Insert(), Update(), Delete()

Insert, Update, Delete



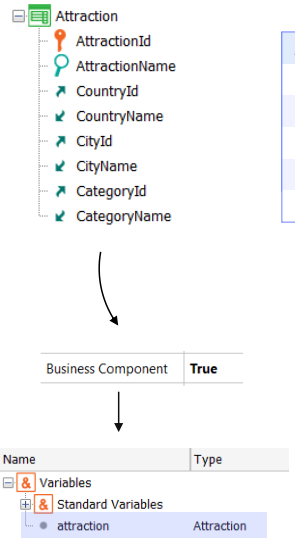
AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2
5		1	1	100

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

...na segunda, a atualização é independente da transação, portanto nenhuma regra será disparada, e o único controle que será realizado será o de duplicados: assim poderíamos atribuir uma categoria inexistente, que o programa não verificará, embora a base de dados o fará, e a execução será interrompida com uma tela de erro indesejável no navegador do usuário).

Vamos continuar nos aprofundando, então, na primeira alternativa, a dos business components.

BC: Insert



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
5	Christ the Redemmer	1	2	2

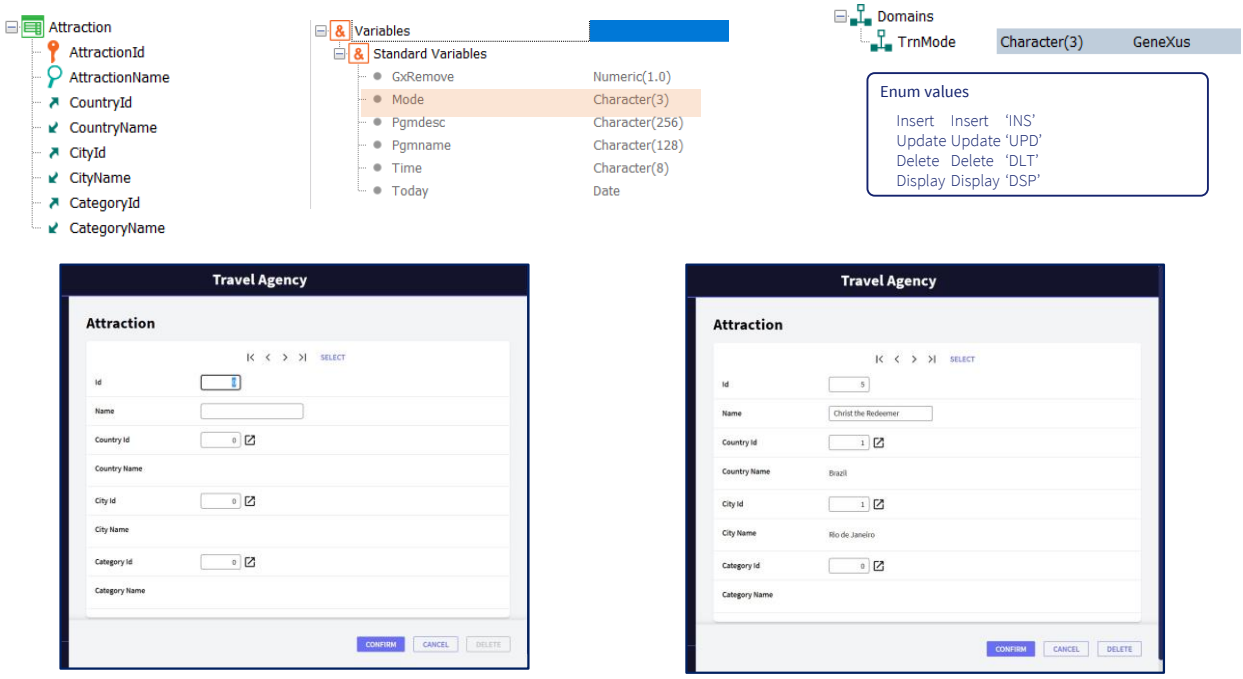
```
&attraction = new()

&attraction.AttractionId        = 5
&attraction.AttractionName  = "Christ the Redemmer"
&attraction.CountryId           = find( CountryId, CountryName = "Brazil" )
&attraction.CityId        = find( CityId, CityName = "Rio de Janeiro" )
&attraction.CategoryId          = find( CategoryId, CategoryName = "Monument" )

&attraction.Save()
if &attraction.Success()
   Commit
endif
```

Se quiséssemos inserir uma nova atração turística utilizando o business component, conseguíamos com:

- criar uma variável desse tipo,
- alocar um novo espaço de memória (não é imprescindível, mas é uma boa prática para garantir que, não importa o que aconteceu antes, a variável neste momento será completamente nova).
- atribuir valor a todos os elementos do Business Component correspondentes aos atributos da tabela:
    - se o identificador é autonumerado, não precisamos atribuir um valor a ele: a base de dados fará isso quando inserirmos,
    - se não atribuirmos valor a algum atributo da tabela, ele ficará vazio ou nulo. Em geral, isto não é um problema, exceto no caso em que o atributo seja chave estrangeira e não suporte nulos; ali, quando tentamos inserir dará um erro de falha de integridade referencial.
- E, finalmente, depois de carregar a estrutura do business component, resta apenas chamar o método Insert,
- Commitando se assim o quisermos quando a inserção for bem sucedida.

Isto é equivalente a utilizar o método Save, pois neste caso, como a variável business component está em modo Insert, tentará inserir (e não fazer um update).

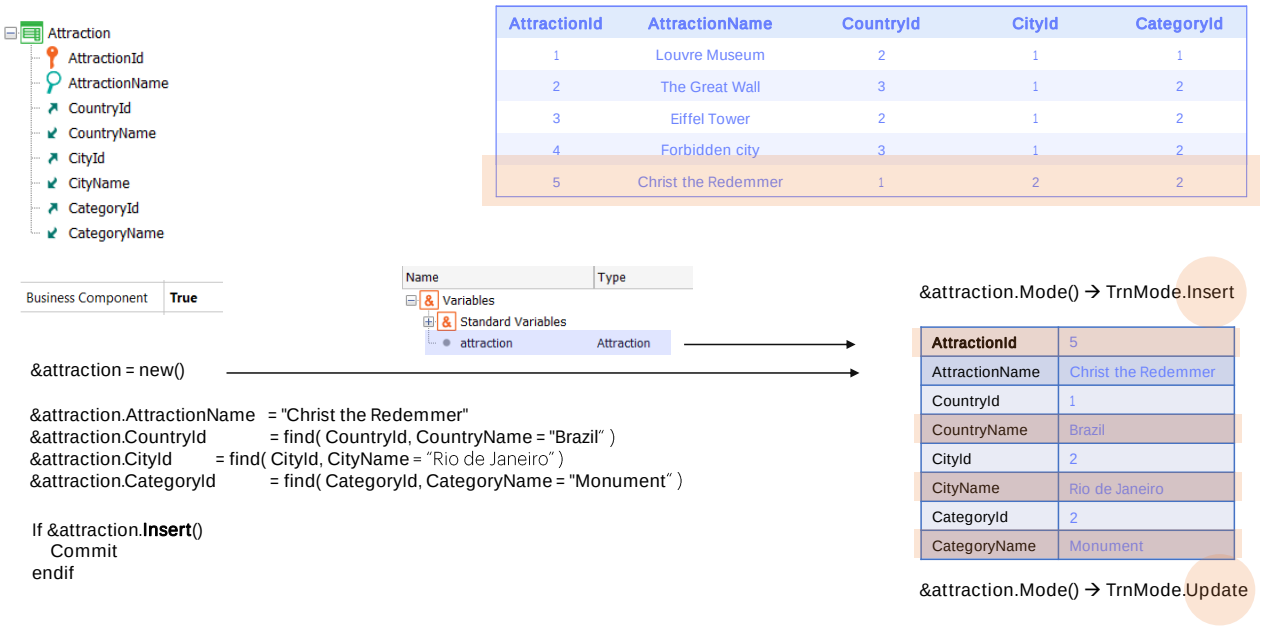
Mode



Lembremos que entre as variáveis padrão de toda transação, encontraremos a de nome Mode. Esta variável contém em todos os momentos o modo no qual se encontra sendo executada a transação.
Para lidar com seus valores em alto nível, o módulo GeneXus incorpora o domínio enumerado TrnMode, que pode assumir os 4 valores: Insert, Update, Delete e Display -que indica que apenas está sendo exibida a informação, mas nada será feito com ela-.

Quando a transação é aberta -se não especificamos que receba como parâmetros modo e identificador- o faz em modo Insert. Por isso os campos estão vazios. Quando saímos do identificador, pesquisa se há registro com o valor que deixamos, neste caso 0, e como não encontra, a transação fica em modo Insert. Se tivéssemos regras condicionadas com If Insert, seriam disparadas. E gravamos. Ao fazer isso nos informa que os dados foram inseridos, mas o que podemos ver também é que o formulário foi esvaziado novamente, o que significa que a transação voltou a ficar em modo Insert. Como veremos, isto não acontecerá quando inserirmos através do business component, que ficará em modo Update.
Se agora escolhemos um valor existente na base de dados para o identificador, por exemplo, o 5 que é o que acabamos de inserir, ao deixar o campo a transação traz seus valores carregados nos campos da tela e fica automaticamente em modo Update. Podemos modificar algo, por exemplo, remover a categoria, deixando-a vazia (supondo que aceitamos nulos nessa chave estrangeira) e ao confirmar nos informa que o registro foi atualizado com sucesso, mas também ficamos posicionados nesse mesmo registro, em modo Update. Aqui sim, o business component se comportará da mesma maneira.

Mode()



Vejamos o que acontece ao trabalhar com o business component.

Temos o método Mode() que nos permite consultar em qual modo se encontra. É só leitura.

Se definimos uma variável &attraction e consultamos antes de fazer qualquer coisa em que modo se encontra, veremos que está em modo Insert.

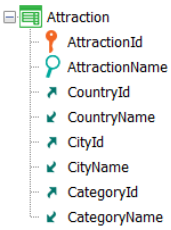
Toda vez que a reinicializemos com new ficará nesse modo.

Então inserimos os valores que queremos dar aos atributos da tabela associada (apenas àqueles que não queremos que fiquem vazios). Todo esse tempo, a variável business component permanecerá em modo Insert.

O que acontece quando se executa o método Insert?
Tentará realizar a inserção na base de dados, executando as regras correspondentes. Se o processo for bem sucedido, ficam carregados todos os elementos da variável business component com os valores correspondentes. Como AttractionId é autonumerado, obtemos o valor que foi atribuído a ele na base de dados, e os atributos que são inferidos na transação aqui também são carregados, de modo que poderíamos consultá-los.
E, por outro lado, o modo da variável business componente passa a ser Update.

BC: Update

Insert, Update, Delete



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

Business Component	True

```
&attraction = Load(2)
&attraction.CategoryId  = find( CategoryId, CategoryName = "Tourist site" )

&attraction.Save()
if &attraction.Success()
  Commit
endif
```

&attraction.Mode() → TrnMode.Update

AttractionId	2
AttractionName	The Great Wall
CountryId	3
CountryName	China
CityId	1
CityName	Beijing
CategoryId	3
CategoryName	Tourist site

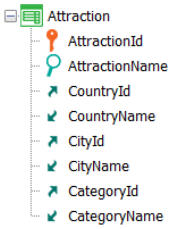
Por outro lado, se o que queríamos era modificar a atração turística de id 2, mudando a categoria para "Tourist site", conseguíamos com:

- carregar na variável Business component &attraction os valores do registro de chave primária 2 da tabela Attraction (fazíamos isso com o método Load, que automaticamente, se existisse esse registro, deixava a variável em modo Update),
- atribuir valor a todos os elementos do Business Component que queremos modificar (neste caso apenas CategoryId, porque os outros queremos que fiquem com o valor que tinham)
- E finalmente invocar o método Update para realizar essa atualização na base de dados,
- Commitando se a operação foi bem sucedida.

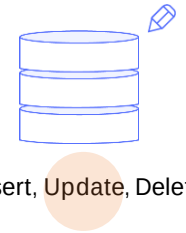
Uma vez realizado o Update, o elemento CategoryName, que é inferido na transação, fica na variável com o valor correto. A variável permanece em modo Update.

Aqui também é equivalente utilizar o método Save, pois neste caso, como a variável business component após o Load ficou em modo Update, o Save tentará atualizar e não inserir.

BC: Update



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



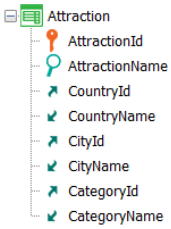
Insert, Update, Delete

```
For each Attraction
Where CountryName = "China" and CityName = "Beijing"
Where CategoryName = "Monument"

    &attraction.Load(AttractionId)
    &attraction.CategoryId = find( CategoryId, CategoryName = "Tourist site" )        st site" )

    If &attraction.Update()
       Commit
    endif

                endfor
```

Se o que queríamos era modificar a categoria de todas as atrações turísticas que, sendo de Beijing correspondiam a monumentos, para atribuir a elas agora a categoria "Tourist Site", então era possível colocando o código anterior dentro de um for each que seleciona apenas os registros desejados, e o Load é realizado para cada AttractionId desses registros.

BC: Update

Insert, Update, Delete



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

```
For each Attraction
Where CountryName = "China" and CityName = "Beijing"
Where CategoryName = "Monument"

   &attraction.Load(AttractionId)
   &attraction.CategoryId = find( CategoryId, CategoryName = "Tourist site" )

   If &attraction.Update()
      Commit
   endif

                endfor
```

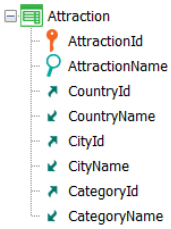
&attraction.Mode() → TrnMode.Update

AttractionId	2
AttractionName	The Great Wall
CountryId	3
CountryName	China
CityId	1
CityName	Beijing
CategoryId	3
CategoryName	Tourist site

Assim, primeiro é feito o Load do registro de Id 2, a variável fica em modo Update, então é modificado seu CategoryId, e ao executar o método Update o registro é atualizado na tabela e a variável fica com o CategoryName correspondente.

BC: Update

Insert, Update, Delete



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

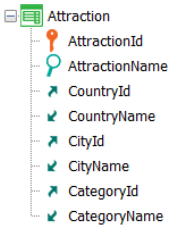
```
For each Attraction
Where CountryName = "China" and CityName = "Beijing"
Where CategoryName = "Monument"

    &attraction.Load(AttractionId)
    &attraction.CategoryId = find( CategoryId, CategoryName = "Tourist site" )

    If &attraction.Update()
       Commit
    endif

                endfor
```

&attraction.Mode() → TrnMode.Update

AttractionId	4
AttractionName	Forbidden city
CountryId	3
CountryName	China
CityId	1
CityName	Beijing
CategoryId	3
CategoryName	Tourist site

E então o mesmo para o próximo registro que atende às condições, ou seja, o de id4.

BC: Delete



AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2
5	Christ the Redemmer	1	2	2
5	Christ the Redemmer	1	2	2

Insert, Update, Delete

```
&attraction.Load(2)
&attraction.Delete()

If &attraction.Success()
   Commit
endif
```

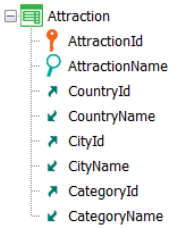
&attraction.Mode() → TrnMode.Update

AttractionId	2
AttractionName	The Great Wall
CountryId	3
CountryName	China
CityId	1
CityName	Beijing
CategoryId	2
CategoryName	Monument

&attraction.Mode() → TrnMode.Delete

Se em vez disso queríamos eliminar uma atração turística, por exemplo, a de identificador 2, primeiro tínhamos que carregá-la na variável Business Component, após o qual fica em modo Update e, em seguida, utilizar o método Delete, que a exclui da tabela e deixa a variável carregada com os dados, mas em modo Delete. Então consultamos por Success para que possamos dar a ordem de Commit.

BC: Delete



AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
3	Eiffel Tower	2	1	2
5	Christ the Redemmer	1	2	2
4	Forbidden city	3	1	2
5	Christ the Redemmer	1	2	2

Insert, Update, Delete

```
For each Attraction
Where CountryName = "China" and CityName = "Beijing"
Where CategoryName = "Monument"

        &attraction.Load(AttractionId)
        &attraction.Delete()

        If &attraction.Success()
           Commit
        endif

    endfor
```

Mas e se o que queríamos era eliminar todas as atrações do tipo monumento de Beijing? Outra vez localizaríamos uma a uma as atrações com o For each e as carregávamos com o load e, em seguida, utilizávamos o método Delete para removê-las.

GeneXus™

training.genexus.com
wiki.genexus.com