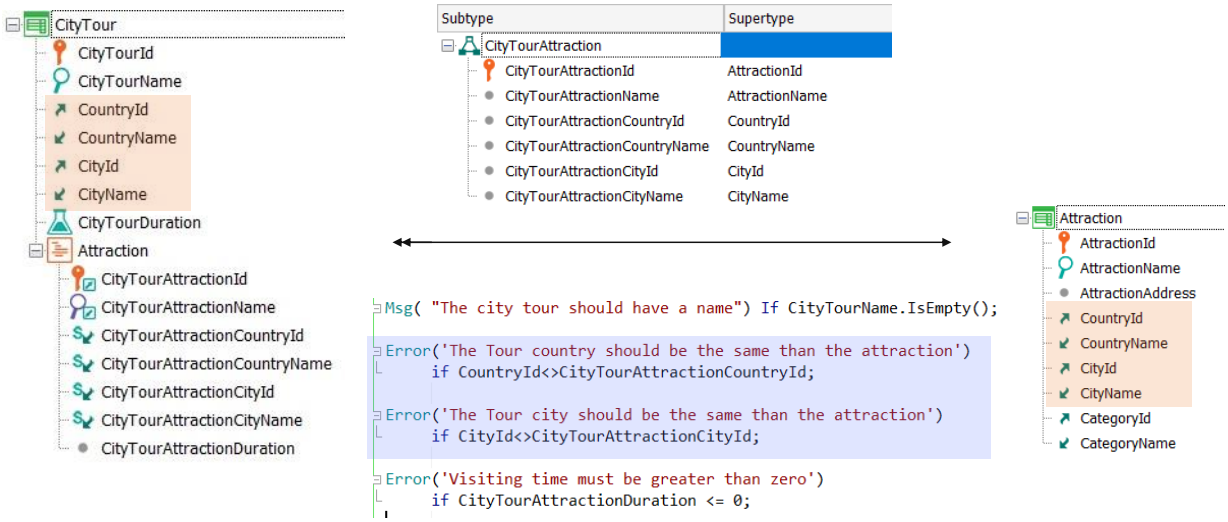
Atualização de base de dados

Usando Business Component de dois níveis

GeneXus

Two-level BC



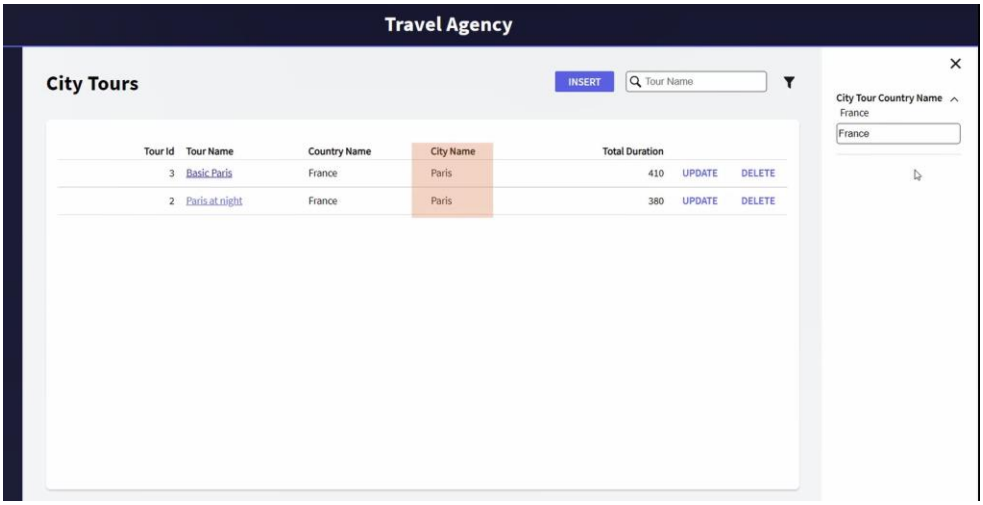
Suponhamos que criamos uma transação CityTour para representar os tours oferecidos aos clientes da agência de viagens para visitar as diferentes atrações turísticas de uma determinada cidade.

É uma transação de dois níveis: no primeiro, além do nome do tour, é especificado seu país e cidade, e o tempo estimado de duração do tour completo, que é a soma dos tempos estimados de visita de cada atração.

O subnível indica as atrações turísticas que o tour inclui, para as quais tivemos que definir um grupo de subtipos, já que precisamos especificar no primeiro nível o país e cidade do city tour e as atrações também têm um país e cidade, que deveremos controlar que coincidam com os do tour.
Observemos que CityTourAttracionId é subtipo de AttractionId, por isso é como se fosse o mesmo atributo, e, portanto, será chave estrangeira na tabela do segundo nível de CityTour.

Além disso, essa tabela contará com o atributo secundário CityTourAttractionDuration, para especificar quanto tempo é estimado que durará a visita a essa atração.

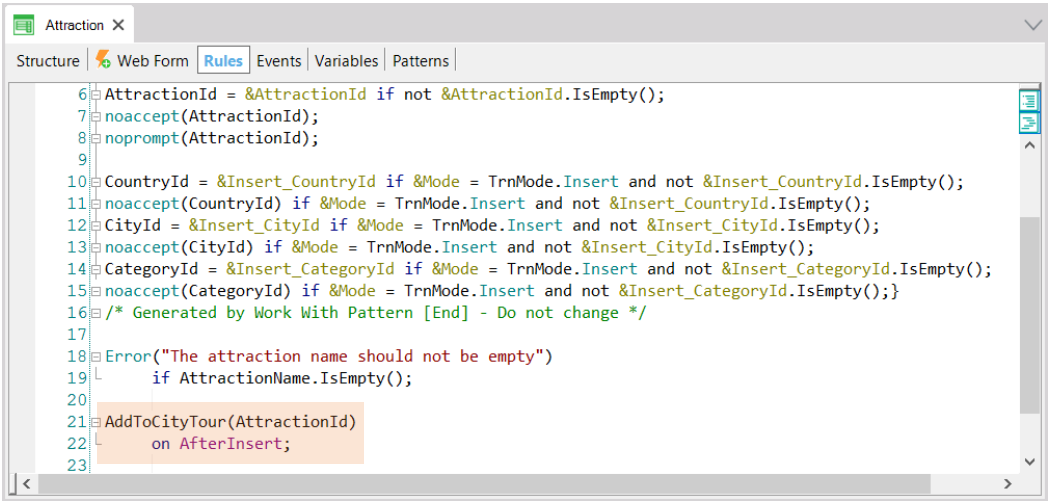
Insert many lines



Aplicamos o pattern work with para CityTour e também para a atrações, para trabalhar mais confortável. E carregamos alguns dados.
Por exemplo, vejamos que para Paris criamos apenas dois city tours. No primeiro temos para o Louvre e para a torre Eiffel, e no segundo apenas para a torre Eiffel.

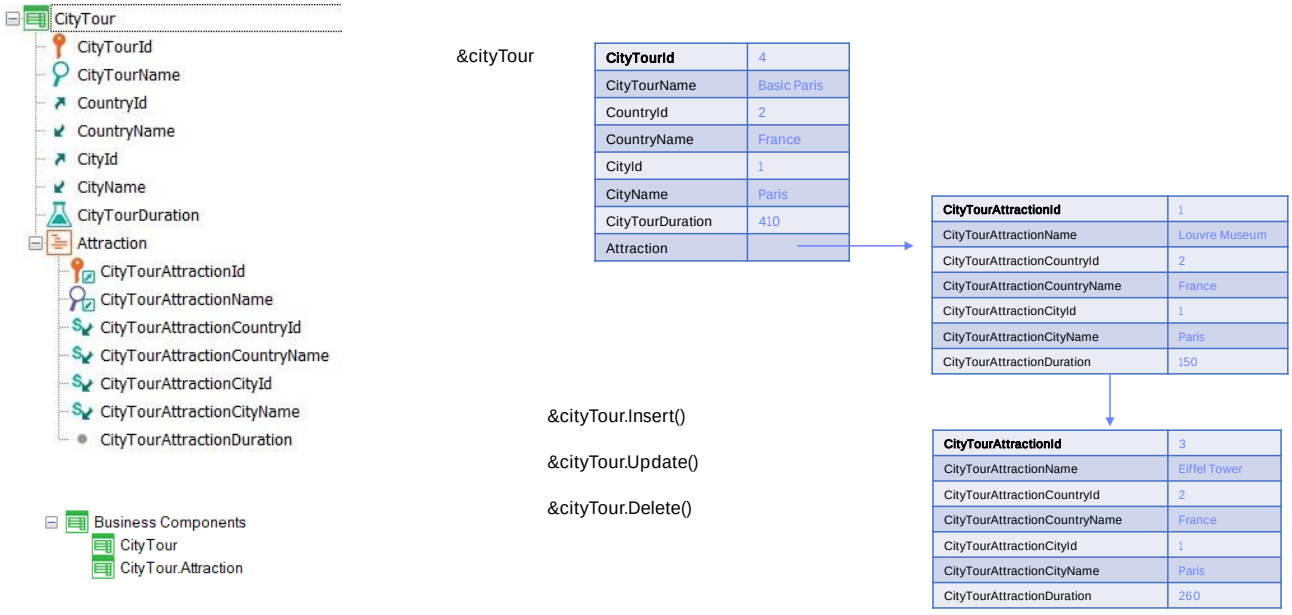
Suponhamos que se deseja que quando o usuário do backoffice estiver trabalhando com as atrações turísticas para inserir uma nova no sistema, -por exemplo, a catedral de Notre Dame- automaticamente, esta seja adicionada a todos os city tours correspondentes à cidade da atração –que neste caso é Paris-, com uma duração fixa para a visita de 120 minutos, que então poderá ser modificada. Como conseguimos este comportamento?

Insert many lines



Resumindo, assim que for inserida uma nova atração através desta transação, precisamos que automaticamente para todos os city tours que tenham seu mesmo país e cidade, seja inserida uma nova linha com essa atração, com tempo de 120 minutos.

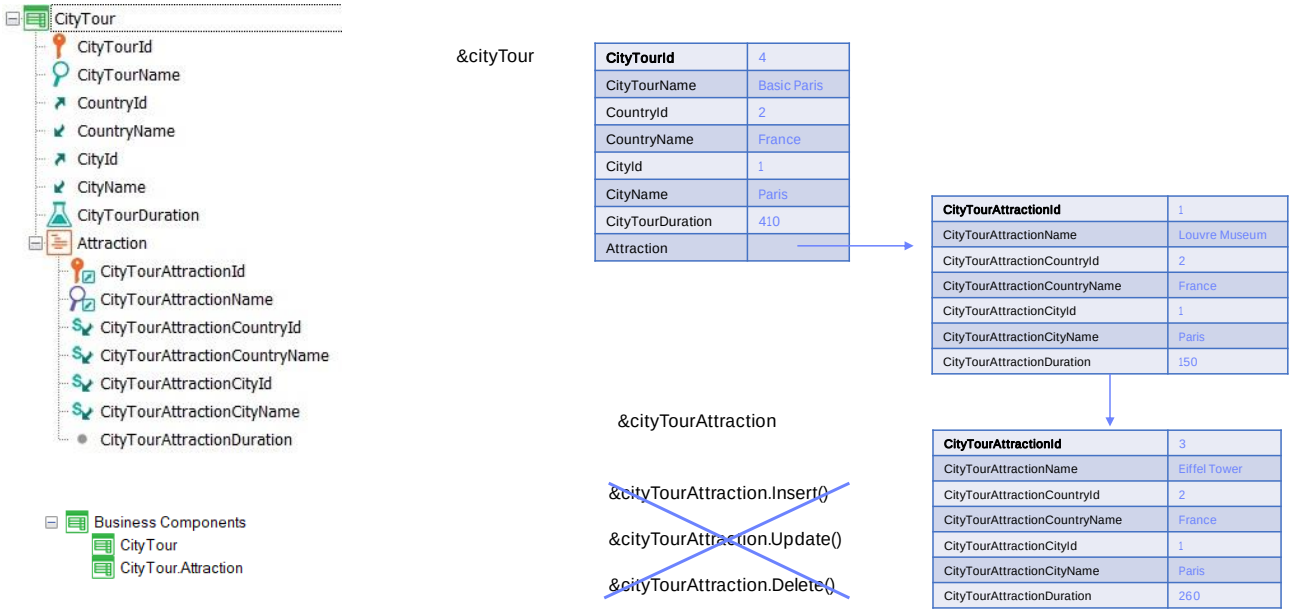
Então, chamaremos em AfterInsert, ou seja, logo após ter sido inserida na tabela Attraction a nova atração, um procedimento ao qual passaremos o identificador dessa atração inserida, e que será o responsável por inserir a linha para os city tours que encontre com o mesmo país e cidade.



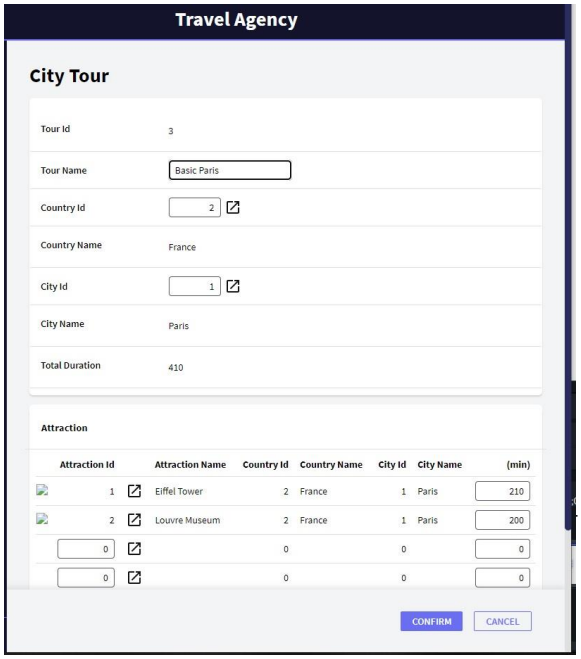
Ao ativar a propriedade Business Component da transação serão criados automaticamente na KB dois business components e não apenas um, como poderia se pensar.

O primeiro é o esperado, com uma estrutura como a que mostramos. Assim, se o city tour 4 tem duas linhas, a estrutura de uma variável do tipo de dados do business component CityTour será como a que aqui representamos, onde o último elemento, Attraction, é uma coleção de linhas. Cada linha corresponderá, por sua vez, a um dado estruturado. Com qual estrutura? A do business component CityTour.Attraction, que é a de cada linha da transação.

Ou seja, enquanto o business component correspondente à transação como um todo é aquele que nos permitirá realizar as operações de Insert, Update e Delete na base de dados, o que é criado para as linhas é para ser utilizado somente como estrutura de dados.

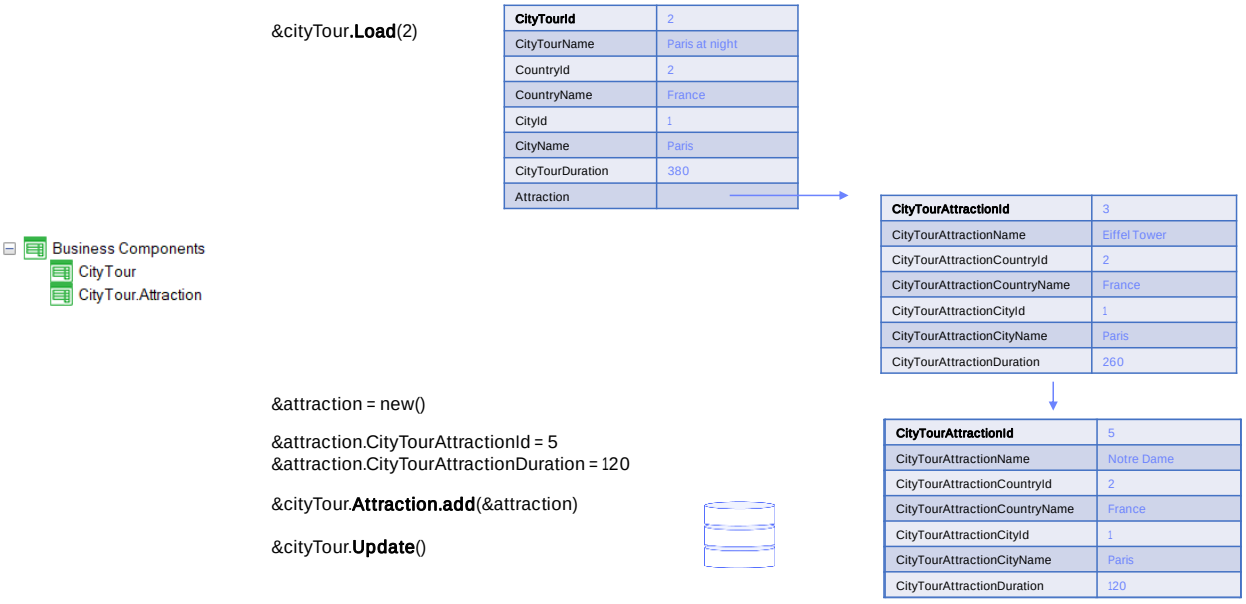


Não permitirá essas outras operações. Isto pode parecer confuso à primeira vista.



Mas, assim como quando queremos inserir uma nova linha através da tela da transação CityTour devemos primeiro instanciar o cabeçalho, depois inserir a linha e finalmente pressionar o Confirm global, o mesmo acontecerá com os business components. Para trabalhar com suas linhas, por exemplo adicionando uma, terá que trabalhar com o cabeçalho e suas linhas, adicionando a linha e, em seguida, fazendo a operação desejada sobre o todo, ou seja, na variável BC da transação.

Insert a line



Então, se queremos inserir uma nova atração para o city tour 2, precisaremos de uma variável &cityTour do tipo de dados do Business component da transação CityTour. Aplicamos a operação Load, após a qual obtemos a variável carregada com os dados da base de dados: neste caso, o cabeçalho e sua única linha.
Então, como precisaremos adicionar uma nova linha, definimos uma variável &attraction do tipo de dados do business component correspondente às linhas. Carregamos os valores de seus elementos e só temos que adicionar esta estrutura à coleção de linhas. Fazemos isto com o método add, aplicado nada mais nada menos do que à coleção Attraction.

Uma vez que temos as informações como queremos na variável &cityTour, realizamos sobre ela a operação de Update, porque o cityTour de Id 2 já existia, só queremos modificá-lo adicionando a linha. Esta operação será a que realmente irá inserir a linha na base de dados.

Insert many lines

```
parm(in:&AttracionId);
```

AddToCityTour

Source | Layout | Rules | Conditions | Variables

Subroutines

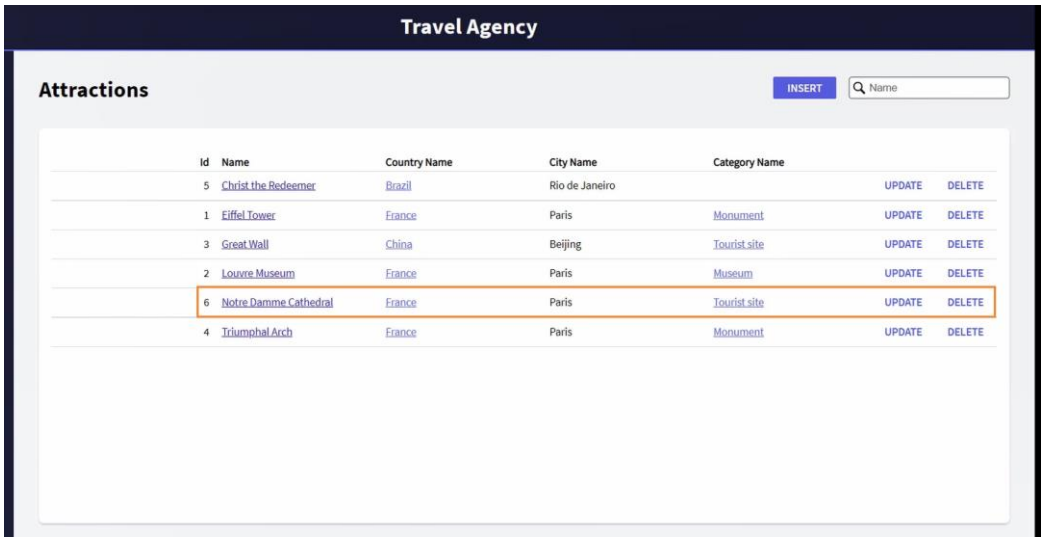
```
/*The attraction received by parameter will be inserted in all the City tours with the same
   country and city of the attraction */

for each CityTour
      where CountryId = find(CountryId, AttractionId = &AttracionId)
      where CityId = find(CityId, AttractionId = &AttracionId)

      &cityTour.Load(CityTourId)

      &attraction = new()
      &attraction.CityTourAttractionId = &AttracionId
      &attraction.CityTourAttractionDuration = 120

      //Add the attraction to the City Tour lines
      &cityTour.Attraction.Add(&attraction)

      If &cityTour.Update()
          commit
      endif
endfor
```

Se vamos verificar o procedimento que implementamos, já entenderemos tudo.
Recebemos o id da atração na variável &AttractionId.

Estamos iterando no for each por todos os city tours do país e cidade da atração recebida –aqui fica claro por que temos que chamar o procedimento depois que foi inserida a atração na tabela Attraction; caso contrário, não encontrará seu país e cidade para fazer este filtro–.

&cityTour é uma variável de tipo de dados do Business component CityTour. A carregamos a partir do id de city tour encontrado no for each. Só temos que adicionar a linha com a atração.
Para isso, definimos uma variável do tipo de dados do business component correspondente às linhas. Criamos um novo espaço de memória para esta variável, - aqui vemos todos os elementos que poderíamos atribuir - atribuímos os elementos armazenáveis: isto é, o id da atração e a duração da visita.
E então só nos resta adicionar essa linha à coleção de linhas do CityTour. Que aqui vemos que se chama Attraction.
Por último realizamos a operação de Update que devolverá True se foi bem-sucedida e, nesse caso, commitamos.
Em seguida, passa para a próxima iteração.

Delete many lines

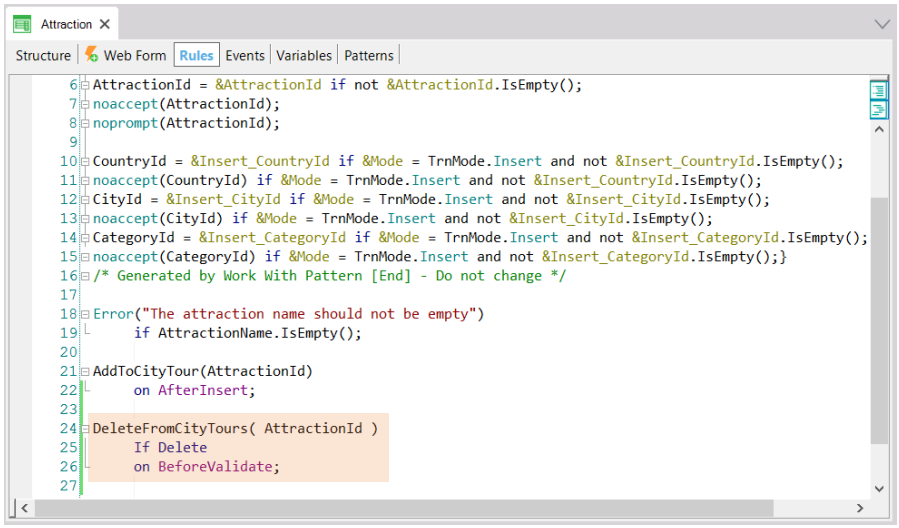


Agora vamos estudar como excluir linhas através do Business Component.

Vejamos que temos no Work with Attractions a catedral de Notre Dame que havíamos inserido anteriormente. Se vamos aos city tours, vemos que há dois de Paris que a contêm: este e este outro. O que queremos, então, agora, é que possamos eliminar a atração –normalmente isto não seria permitido pois será verificada a integridade referencial, portanto, que não existam city tours contendo esta atração, para permitir sua eliminação–. Então vamos querer que ao pressionar Delete, primeiro sejam removidas todas aquelas linhas de city tours nas quais se encontre a atração, para depois passar a eliminar, sim, a atração.

Confirmo, e agora vemos que foi removido deste city tour, e também do que tínhamos visto antes. A questão é como o implementamos.

Delete many lines

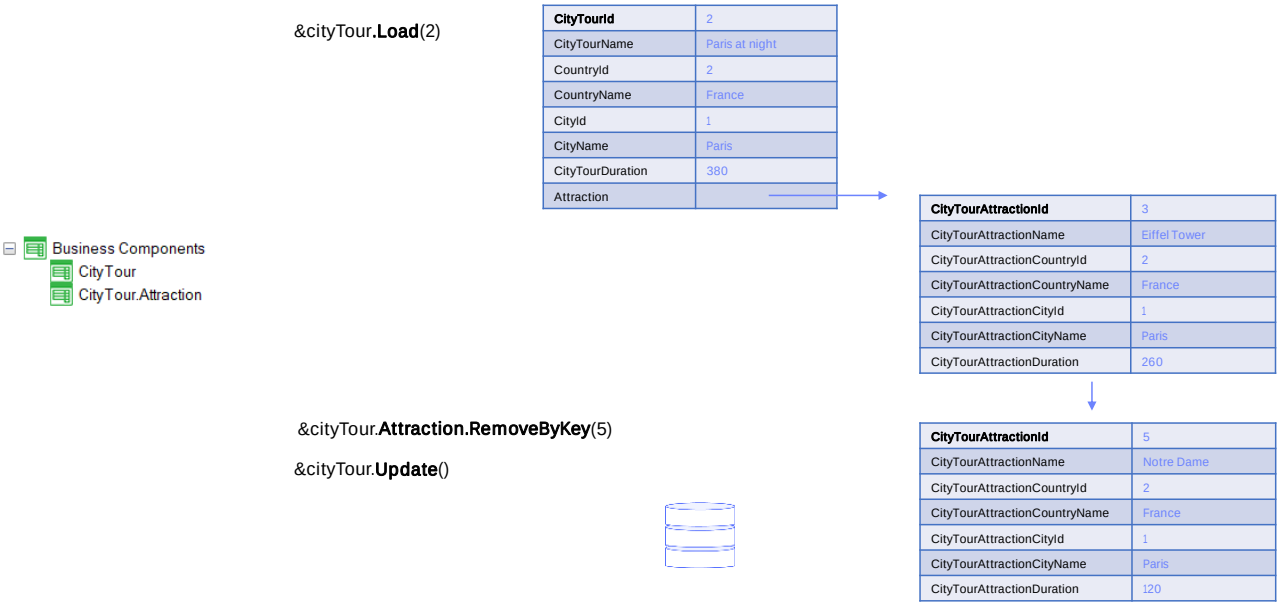


A partir da transação Attraction, quando estamos em modo Delete, ou seja, estamos querendo eliminar a atração e antes que os dados sejam validados, ou seja, antes de serem realizadas as verificações de integridade referencial, chamamos um procedimento, passando o identificador de atração; e esse procedimento é o que terá que cuidar de eliminar todas as linhas de city tours que a contenham.

Como fazemos isto?

Quando queremos excluir uma linha utilizando a transação, primeiro temos que carregar o city tour, que ficará em modo Update, em seguida, excluir a linha e finalmente pressionar o Confirm para que essa exclusão seja realmente realizada na base de dados. Aqui será análogo.

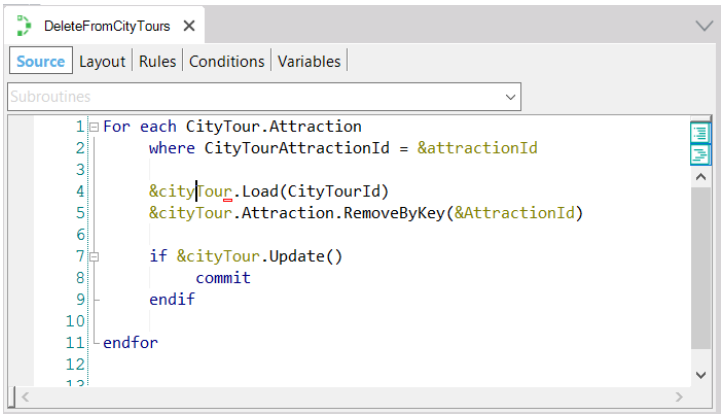
Delete a line

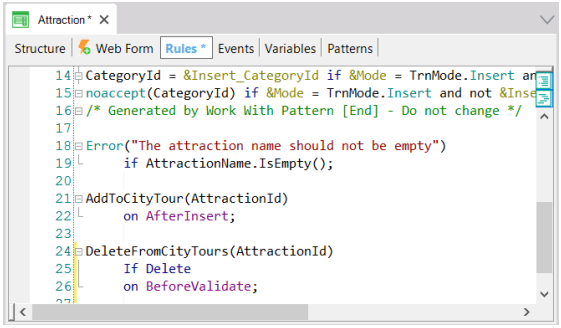


Se queremos remover a Catedral de Notre Dame do city tour 2, em uma variável do tipo de dados do Business Component CityTour, carregamos a estrutura utilizando o método Load e, em seguida, excluímos a linha da coleção Attraction. Como? Usando o método RemoveByKey. Como seu nome indica, este método procurará na coleção, aquele item que corresponda ao valor de chave indicado. No nosso caso, o de Id 5 que é da Catedral de Notre Dame. Esta exclusão foi realizada apenas em memória. Agora falta impactá-la na base de dados. Para fazer isso temos que atualizar o city tour e, portanto, utilizamos o método Update. Claro, também poderíamos ter utilizado o Save(). Tudo isto equivale a ter pressionado o botão Confirm na transação.

Delete many lines

```
parm( in: &AttractionId );
```

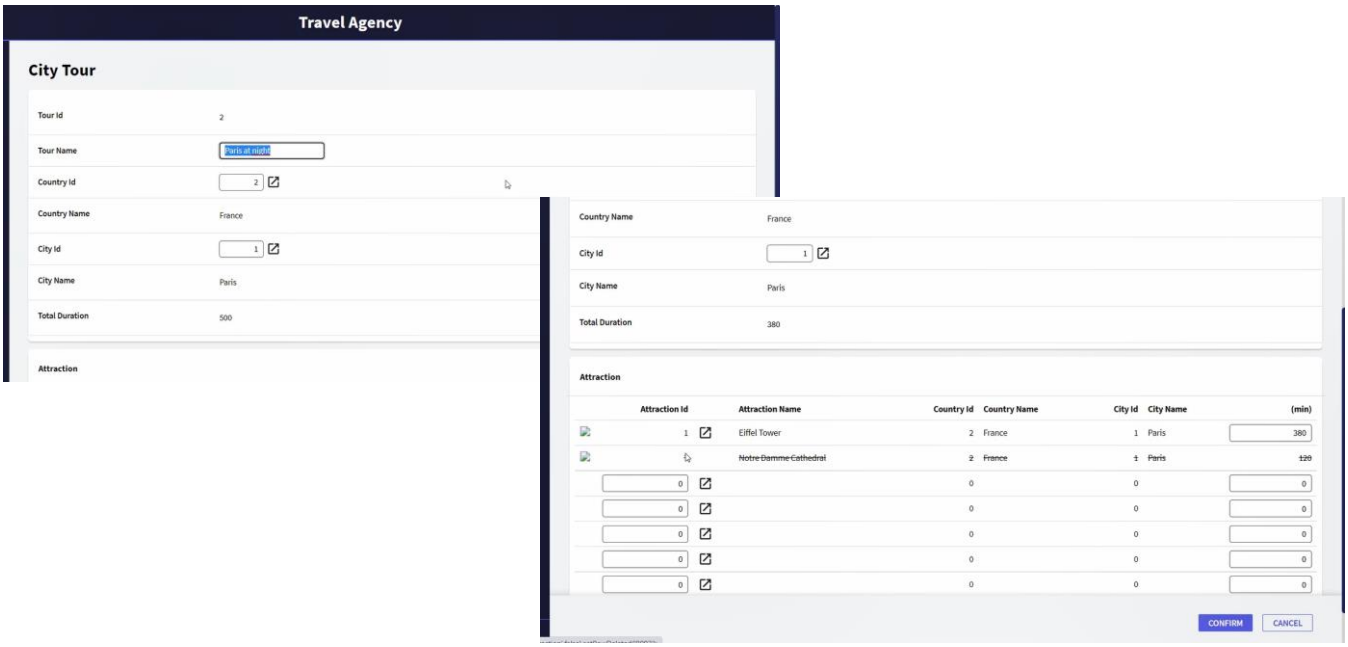




Se agora vamos ver como implementamos nosso procedimento no GeneXus: vemos que recebemos na variável &AttractionId, o identificador de atração que será removido. Fazemos um for each buscando nas linhas de city tour se há alguma que corresponda a esse attraction id. Se encontrarmos, então em uma variável city tour, do tipo de dados do Business Component correspondente à transação carregamos o CityTourId que corresponda a essa linha, e então o que fazemos é aplicar o método RemoveByKey à coleção de atrações desse city tour para remover a de identificador attraction id recebido por parâmetro.
Então fazemos o Update; se foi bem-sucedido commitamos.

Com isto temos certeza de que teremos removido todas as linhas de city tours que tiveram essa atração entre seus dados. Então, quando este procedimento termina sua execução retorna o controle aqui, é feita a validação da atração correspondente que agora não terá informações relacionadas e passa a excluir sem dar qualquer tipo de falha de integridade.

Update a line

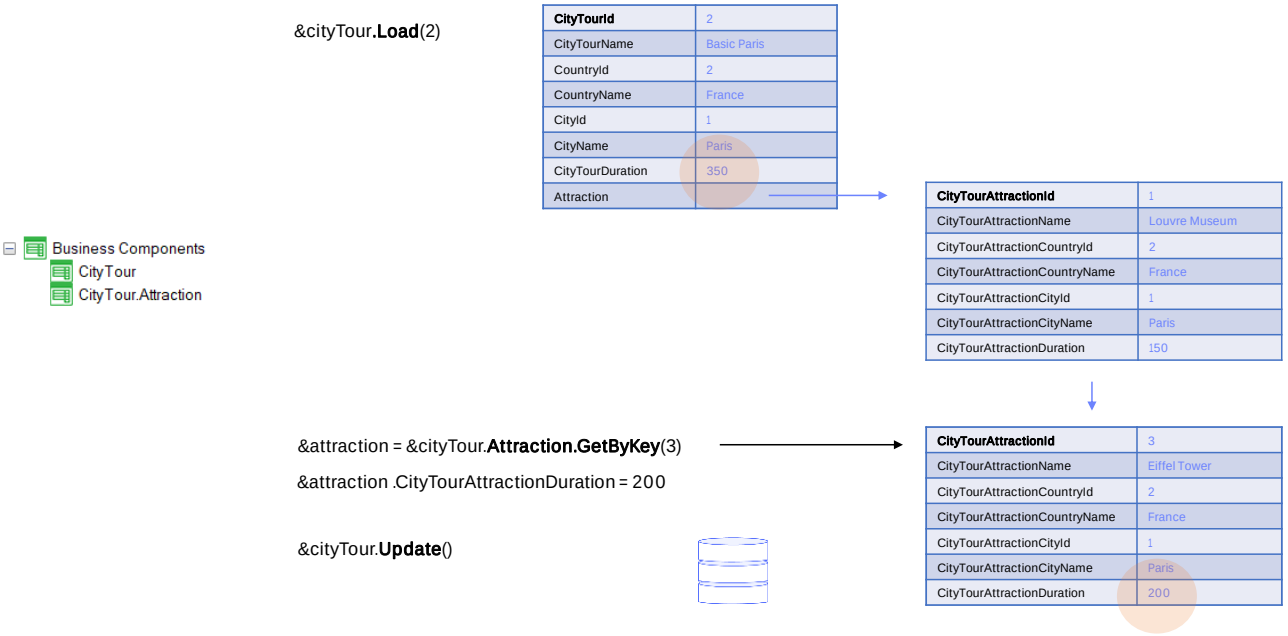


Nos resta estudar o último caso, que é como fazemos para atualizar linhas de uma transação, mas através do Business component.

Se fizéssemos isso através da transação editaríamos, neste caso o TourId 2. Suponhamos que nos interessa modificar a duração da visita à torre Eiffel, passando de 260 minutos para 200, e uma vez que fazemos isto o que nos resta é confirmar, para que a atualização ocorra.

Algo assim teremos que fazer através do business component: carregá-lo, modificar a linha e atualizar.

Update a line



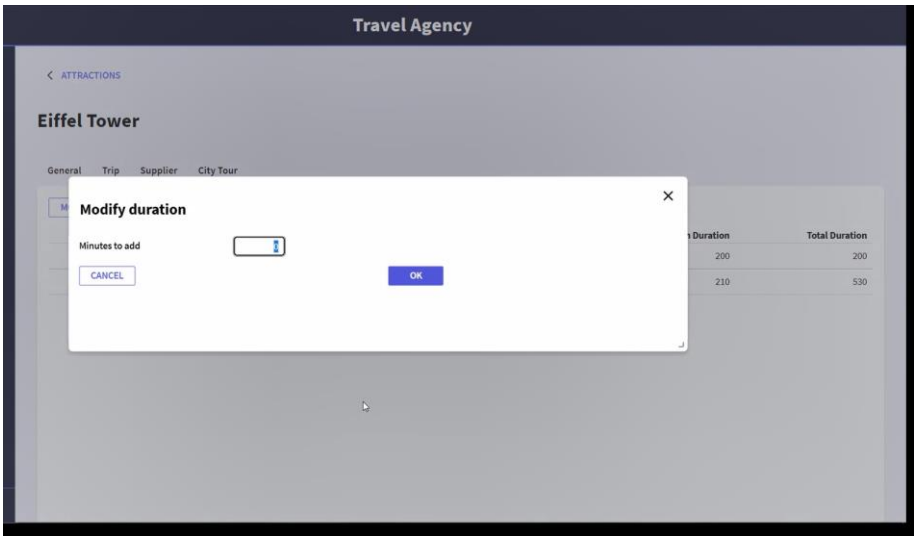
Carregamos então a variável &cityTour com o city tour de id 2.

Devemos acessar o item da coleção que corresponde à atração 3. Para isso, temos o método GetByKey, que aplicado à coleção Attraction, ou seja, à das linhas da transação, nos devolverá uma referência direta a essa posição de memória. Portanto, a atribuímos à variável &attraction do tipo de dados do business component correspondente às linhas. Desta forma, a variável &attraction não será uma cópia dessa linha, mas será exatamente essa linha.

Agora só precisamos modificar o elemento CityTourAttractionDuration dessa variável e isso terá um impacto direto na variável &cityTour.

Nos falta impactar isto na base de dados, e para isso pedimos que seja realizada a operação de Update do Business Component. Observemos que como resultado, além de atualizar na base de dados o registro correspondente a essa linha, a variável &cityTour também é atualizada, e o elemento CityTourDuration que correspondia a uma fórmula no cabeçalho da transação que somava os tempos de visita das linhas, também é atualizado na variável.

Update many lines



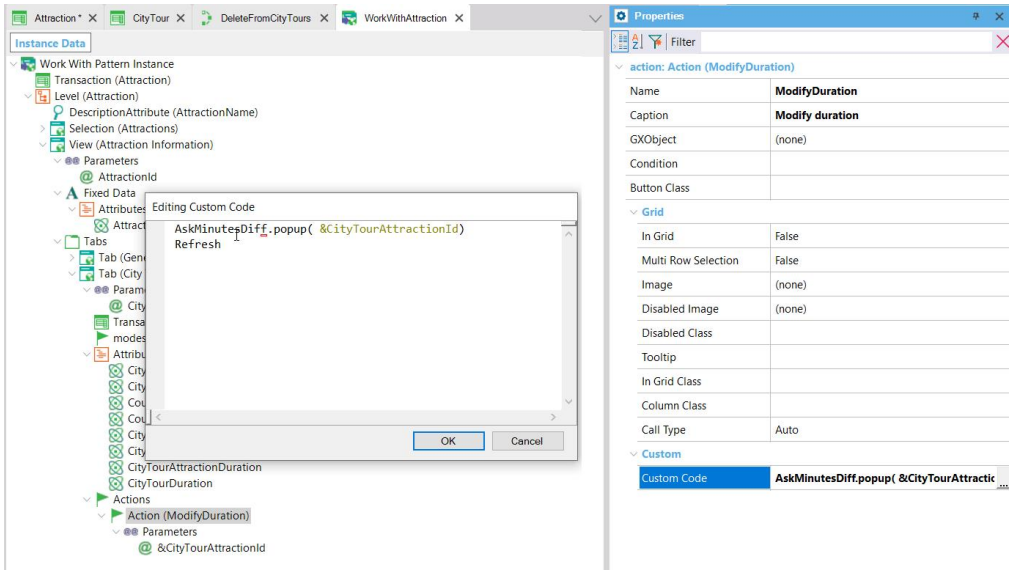
Agora vamos ver como colocamos em prática a modificação de linhas através de business component em nossa aplicação.

Estamos posicionados no Work with de Attractions. Se editamos uma determinada atração, por exemplo a torre Eiffel, e vamos até a aba que nos mostra todos os city tours nos quais essa atração se encontre, vemos que está nos mostrando, entre outras informações, a duração em minutos da visita a essa atração em cada um dos city tours.

Queríamos oferecer a partir daqui a possibilidade de modificar essa duração, somando ou subtraindo minutos em todos os city tours nos quais se encontre. Então, suponhamos que queremos adicionar 10 minutos em cada um. Apresentamos esta tela ao usuário onde indica que quer somar 10 minutos e vemos que efetivamente foi realizada essa adição.

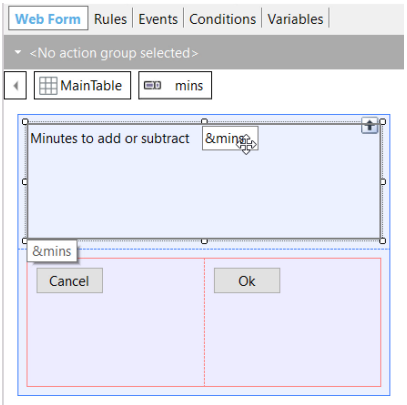
O que foi feito por trás? Foi acessado cada um desses dois city tours, se iniciou pelo primeiro, foi acessada a linha correspondente a esta atração, e foram adicionados 10 minutos ao tempo estimado. E foi feita a mesma coisa para esta outra. Ou seja, teve que ser modificada uma linha de cada um destes dois city tours.

Update many lines

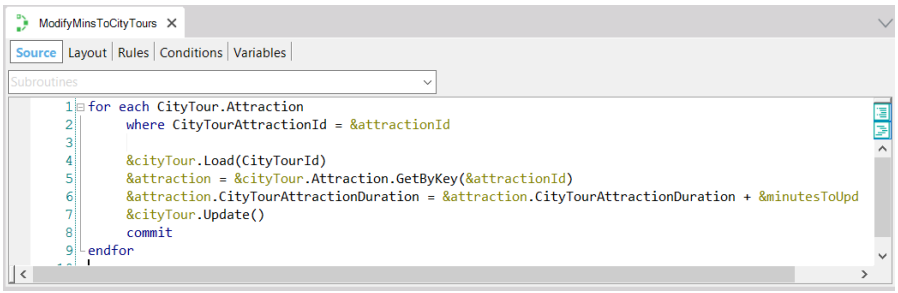


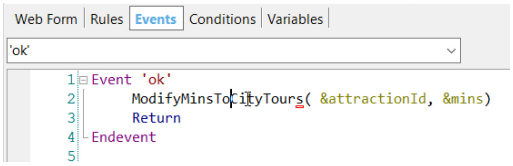
Se vamos ver como implementamos no GeneXus, vemos que no Work with Attraction adicionamos para a guia correspondente aos city tours da atração, uma ação, ModifyDuration, cujo código é a invocação ao web panel deste nome, lhe passando o identificador de atração, e quando esse web panel termina de executar é feito um refresh para mostrar no grid que víamos os dados atualizados.

Update many lines




```
parm( in: &attractionId, in: &minutesToUpd);
```





Vejamos esse web panel, que é o que pede ao usuário nesta variável a modificação em minutos e o que faz é invocar um procedimento que é o que efetivamente faz a modificação nos dados correspondentes. O procedimento precisa receber o id da atração e os minutos que queremos somar ou subtrair para essa atração nos diferentes city tours nos quais se encontre.

Então vamos visualizar como programamos esse procedimento.

Vemos que está recebendo nestas duas variáveis a informação, e o que faz é: em um for each percorre as atrações dos city tours, filtrando pela atração recebida por parâmetro. Então, para cada um dos registros que encontre o que faz é carregar na variável business component &cityTour o city tour em questão; é obtido na variável &attraction do tipo CityTour.Attraction –ou seja, o business component correspondente às linhas– o item da coleção Attraction que corresponde à chave &AtrractionId. Então o que é feito é modificar a duração: o elemento CityTourAttractionDuration para esse item, adicionando ao valor que tinha, a quantidade em minutos recebida por parâmetro. E então simplesmente é feito um Update, commitando.

Summary

Insert a line

```
&BC.Load(PKAttribute)

&lineBC = new()

&lineBC.PKLineAtt = …
&lineBC.LineAtt2 = …
…
&lineBC.LineAttn = …

&BC.Lines.Add(&lineBC)

&BC.Update()
```

Delete a line

```
&BC.Load(PKAttribute)

&BC.Lines.RemoveByKey(PKLineAtt)

&BC.Update()
```

Update a line

```
&BC.Load(PKAttribute)

&lineBC = &BC.Lines.GetByKey(PKLineAtt)
&lineBC.LineAtt2 = …
&lineBC.LineAttn = …

&BC.Update()
```

Aqui vemos um resumo do que foi visto sobre como inserir, excluir e modificar uma linha em um business component de dois níveis.

Em todos os casos a operação é Update, dado que supusemos que o cabeçalho já existia.

Sempre, então, como primeira coisa carregamos a variável business component sobre a qual queremos trabalhar. Para o caso da inserção de uma linha, é criada uma variável de tipo do business component das linhas; é reservado novo espaço de memória; são atribuídos valores a todos os elementos da linha que correspondam a atributos da tabela aos quais queremos dar valor –aqueles que não atribuímos ficarão nulos ou vazios– e então a linha é adicionada à coleção de linhas. Esta adição também é por referência, por isso devemos ter muito cuidado de fazer o new() para criar nova memória antes de trabalhar com cada linha.

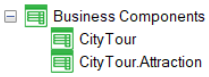
Para o caso da exclusão, se conta com o método RemoveByKey, que recebe por parâmetro o identificador da linha que se deseja excluir.

E, por último, a fim de poder modificar algum atributo ou atributos de uma linha, temos o método GetByKey da coleção de linhas, que também recebe por parâmetro o id da linha que deseja obter. Este método devolve uma referência ao business component correspondente a essa linha, por isso devemos atribuir o método a uma variável desse tipo. Então, ao modificar a variável já estaremos modificando esse item da coleção.

Insert a new header + lines

```
&cityTour = new()

&cityTour.CityTourId = 2
&cityTour.CityTourName = 'Paris at night'
&cityTour. CountryId = 2
&cityTour.CityId = 1
```

CityTourId	2
CityTourName	Paris at night
CountryId	2
CountryName	France
CityId	1
CityName	Paris
CityTourDuration	380
Attraction	




```
&attraction = new()
&attraction.CityTourAttractionId = find( AttractionId, AttractionName = "Eiffel Tower")
&attraction.CityTourAttractionDuration = 260

&cityTour.Attraction.add(&attraction)
```

CityTourAttractionId	3
CityTourAttractionName	Eiffel Tower
CityTourAttractionCountryId	2
CityTourAttractionCountryName	France
CityTourAttractionCityId	1
CityTourAttractionCityName	Paris
CityTourAttractionDuration	260

```
&attraction = new()
&attraction.CityTourAttractionId = find( AttractionId, AttractionName = "Notre Dame")
&attraction.CityTourAttractionDuration = 120

&cityTour.Attraction.add(&attraction)

&cityTour.Insert()
```

CityTourAttractionId	5
CityTourAttractionName	Notre Dame
CityTourAttractionCountryId	2
CityTourAttractionCountryName	France
CityTourAttractionCityId	1
CityTourAttractionCityName	Paris
CityTourAttractionDuration	120

Se queremos inserir um novo cabeçalho e suas linhas, por exemplo, o city tour 2, assumindo que não existia, começamos definindo a variável BC e alocando nova memória.

Em seguida, atribuímos valor para todos os elementos que correspondem a atributos do cabeçalho não inferidos nem fórmulas.

Em seguida, criamos um novo espaço de memória para uma variável &attraction do tipo do business component das linhas. Atribuímos os valores para os atributos não inferidos e adicionamos essa variável à coleção das linhas do business component global.

Fazemos o mesmo para inserir uma segunda linha. Observemos que aqui sim é obrigatório reservar novo espaço de memória, pois caso contrário, ao fazer as atribuições estaremos sobrescrevendo a linha anterior.
Adicionamos a nova variável à coleção Attractions do BC e por último executamos a operação de Insert, para que sejam inseridos cabeçalho e as duas linhas na base de dados.

Se não houve nenhum erro, a variável ficará carregada com todos os dados, os que inserimos e os inferidos e fórmula.

GeneXus™

training.genexus.com
wiki.genexus.com