Atualização de base de dados

Business Components: diferenças entre métodos

GeneXus™

Methods to operate on the database

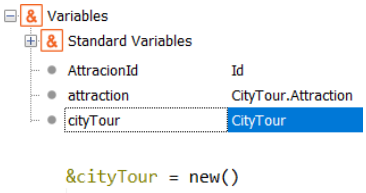
- Load(), Update(), Delete(), Insert()
- Save(), Asignation instead of Load, InsertOrUpdate()

```
&cityTour ...
...
&cityTour.Save()
```

```
&cityTour ...
...
&cityTour.Insert()
```

```
&cityTour ...
...
&cityTour.Update()
```

TrnMode.Insert



TrnMode.Udpate

```
&cityTour.Load(2)
&cityTour.Insert()
&cityTour.Update()
```

TrnMode.Delete

```
&cityTour.Delete()
```

Retomemos o final do vídeo, onde comparávamos os business components de 1 e 2 níveis e continuemos.

"Temos utilizado repetidamente os métodos: Load para carregar informações existentes em um BC, em combinação com os métodos Update() ou Delete(), para atualizar ou excluir, e o método Insert(), para inserir.

Mas podemos utilizar outras formas. Basta termos claras as diferenças e casos de uso.

Por exemplo, o método Save nos poupa de especificar a operação de que se trata. Se parece mais com o botão Confirm da transação. Vai depender do modo em que se encontra a variável, se tentará inserir ou atualizar: se encontra-se em modo Insert, tentará Inserir e se estiver em Update, atualizar.

Por exemplo, toda variável BC definida em um objeto, está em modo Insert, e o mesmo ocorre quando lhe destina novo espaço de memória com new().

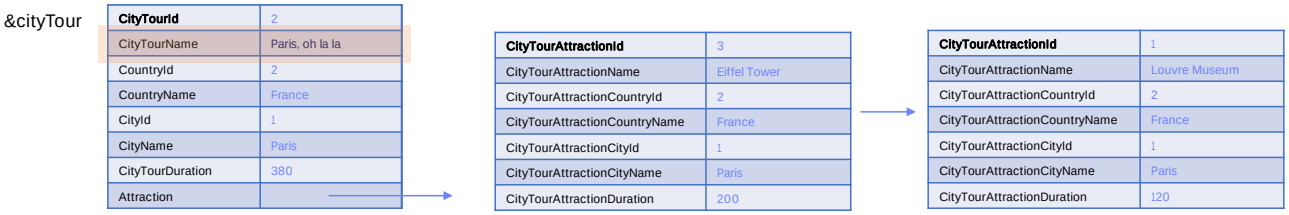
Se a variável for carregada com Load(), passa imediatamente a estar em modo Update.
Depois de realizada alguma operação sobre ela, se for bem-sucedida, estará em modo Update, se foi inserido ou modificado, e em modo Delete, se lhe foi aplicado o método Delete().

Então, para saber que operação tentou se realizar ao encontrar este Save, devemos conhecer o contexto desta variável. Em vez disso, com os métodos Insert e Update, isso não é necessário. Independentemente do modo, vai querer inserir ou atualizar as informações na base de dados.″

Methods to operate on the database

- Load(), Update(), Delete(), Insert()
- Save(), Asignation instead of Load, InsertOrUpdate()

```
&cityTour.Load(2)
&cityTour.CityTourName = "Paris, oh la la"
...
&cityTour.Save()
```



Com este exemplo ficará bastante claro.

Observemos que depois de realizar o Load, se existe um city tour 2, a variável carregará a informação da base de dados e ficará em modo Update, por isso, o Save tentará atualizar. Atualizará apenas os elementos que difiram, neste caso, apenas o CityTourName.

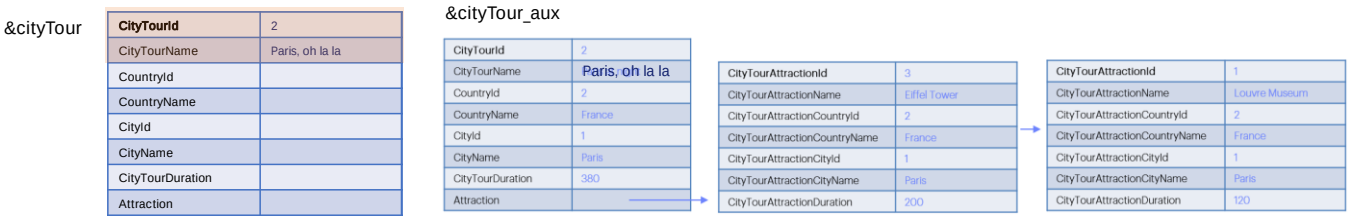
Methods to operate on the database

- Load(), Update(), Delete(), Insert()
- Save(), Asignation instead of Load, InsertOrUpdate()

```
&cityTour.Load(2)
&cityTour.CityTourName = "Paris, oh la la"
...
&cityTour.Save()
```

```
&cityTour.CityTourId = 2
&cityTour.CityTourName = "Paris, oh la la"
...
&cityTour.Save()
```

```
&cityTour.CityTourId = 2
&cityTour.CityTourName = "Paris, o
...
&cityTour.Update()
```



Mas o que aconteceria se, em vez de utilizar o método Load, diretamente atribuíssemos valor à chave primária?
Vai depender do modo em que se encontrava a variável. Se não estava em modo Update, por exemplo, porque só foi definida no objeto e isto é o primeiro que é feito com ela, então, o Save tentará inserir um city tour 2, o que falhará por chave primária duplicada.

Em vez disso, se utiliza-se o método Update no lugar do Save, não haverá problema. À primeira vista, poderia parecer que sim, porque a variável &cityTour terá somente dois elementos não vazios. Nem mesmo as duas linhas que sabemos que tem o cityTour 2, estão ali. Podemos então pensar que isto causará um dano na base de dados ao executar o Update.
No entanto, o Update é inteligente, e sabe que, como a variável está em modo Insert, isso significa que não tem carregados todos os dados (porque se tivesse executado um Load, a variável estaria em modo Update e não Insert), então, somente deve modificar da base de dados, os dados efetivamente atribuídos no BC, e nenhum outro.
Internamente carrega numa variável BC auxiliar os dados que, claro, fica em Update; modifica os que foram modificados na variável que está em modo Insert, neste caso, só o CityTourName, e faz um Save(), como a variável auxiliar está em Update por ter sido carregada... atualiza.

Em geral, não é boa a prática trabalhar deste modo, atribuindo diretamente a chave primária da variável BC, sem realizar um Load

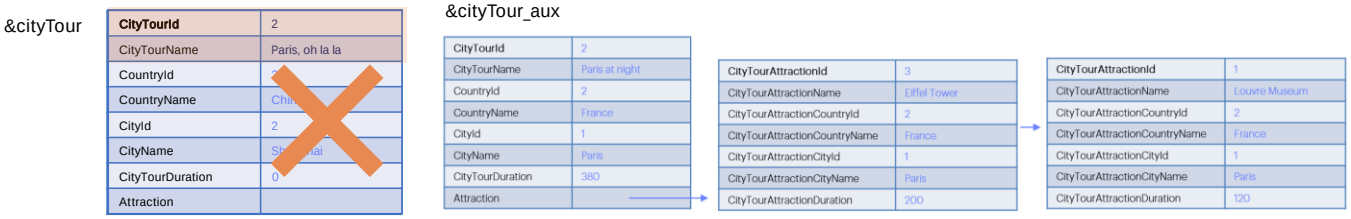
Methods to operate on the database

- Load(), Update(), Delete(), Insert()
- Save(), Asignation instead of Load, InsertOrUpdate()

```
&cityTour.Load(2)
&cityTour.CityTourName = "Paris, oh la la"
...
&cityTour.Save()
```

```
&cityTour.CityTourId = 2
&cityTour.CityTourName = "Paris, oh la la"
...
&cityTour.Save()
```

```
&cityTour.Load(5)
&cityTour.CityTourId = 2
&cityTour.CityTourName = "Paris, o
...
&cityTour.Update()
```



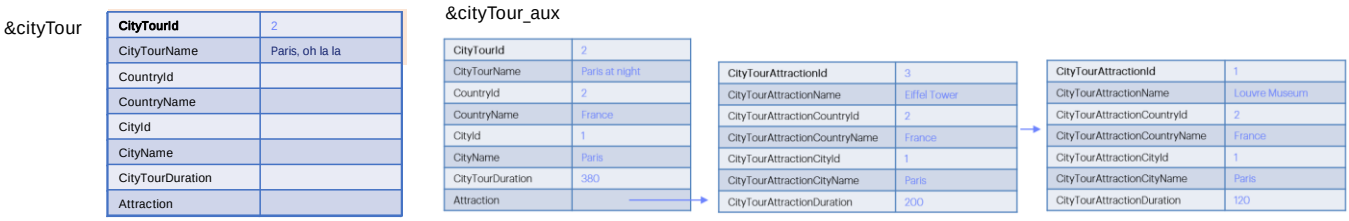
, uma vez que a variável se encontrava em modo Update porque, por exemplo, havia a utilizado antes (imaginemos que tinha a carregado com o city tour 5 que só tem cabeçalho), então, se não formos cuidadosos, podemos deixar-lhe valores lixo que provêm desse antes.

A atribuição é a única maneira quando o business component é carregado em um Data Provider, como veremos a seguir, aproveitando para estudar o método InsertOrUpdate.

Methods to operate on the database

- Load(), Update(), Delete(), Insert()
- Save(), Asignation instead of Load, InsertOrUpdate()

```
&cityTour = new()
&cityTour.CityTourId = 2
&cityTour.CityTourName = "Paris, o
...
&cityTour.Update()
```



A forte recomendação é que, se a atribuição vai ser utilizada, então antes se peça novo espaço de memória para a variável, de modo que esteja limpa e em Insert.

A atribuição é a única maneira quando o business component é carregado em um Data Provider, como veremos a seguir, aproveitando para estudar o método InsertOrUpdate.

Methods to operate on the database

- Load(), Update(), Delete(), Insert()
- Save(), Asignation instead of Load, InsertOrUpdate()

Insert()
Update()
Delete()
InsertOrUpdate()

FOR COLLECTION OF BCs TOO!

```
Parm( in: &cityTour )

If &cityTour.InsertOrUpdate()
   Commit
else
   Rollback
endif
```

```
If not &cityTour.Insert()
    for &message in &cityTour.GetMessages()
  if &message.Id = "DuplicatePrimaryKey"
      &cityTour.Update()
  endif
  endfor
endif
If &cityTour.Sucess
    Commit
else
    Rollback
```

Este método é utilizado quando não sabemos se existe ou não, na base de dados, a informação que estamos manipulando no BC.

Por exemplo, quando em um procedimento é recebida numa variável business component. Dentro do código não temos ideia de qual informação virá carregada ali. Sabemos que a variável estará em modo Insert, pois é nova neste objeto, assim seja parâmetro. Não recebe o modo em que estava no objeto chamador. Só recebe o valor de seus elementos.

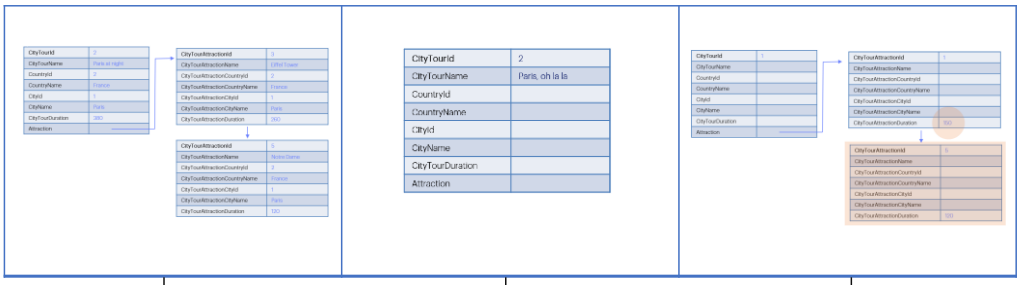
Então, se tentarmos fazer um Save(), só será bem-sucedido se a chave primária do BC não se encontra na base de dados. Do contrário, falhará por chave duplicada (uma vez que, como a variável estará inevitavelmente no modo Insert, o que tentará o Save é inserir). Portanto, se o objeto que chamou o procedimento, o fez com a intenção de que se realizasse uma atualização, estamos com problemas.

O método InsertOrUpdate nos evita de ter que programar as duas possibilidades. Ou seja, tentar inserir, e se isso falhar por chave duplicada, e não por outra coisa, então, tentar o Update. Isto é, nada mais, nem menos, do que faz o InsertOrUpdate().

Recordemos, além disso, que tanto o Insert, como o Update, o Delete e o InsertOrUpdate, podem ser utilizados sobre variáveis de tipo coleção de business components, que é o outro caso onde veremos que será muito útil este último método.

Methods to operate on the database

&cityTours




```
&cityTours = GetCityTours( ... )
If &cityTours.InsertOrUpdate()
   Commit
else
   Rollback
endif
```

Data Provider Source?

Aqui temos uma variável de tipo coleção de business components, que poderia estar carregada com base no que devolve um Data Provider, ou estar carregada por código, neste mesmo programa. Neste caso, o Data Provider devolverá uma coleção de itens de tipo do Business Component CityTour- e queremos impactar na base de dados, o realizado em cada business component da coleção.

Pode ser que o primeiro item da coleção corresponda à informação a ser inserida (neste caso um cabeçalho e suas duas linhas); no segundo item, à informação a ser modificada (por exemplo, somente o CityTourName do cabeçalho); e no último, também à informação a ser modificada (no exemplo, se modifica uma linha e outra é adicionada).

Neste caso, a operação não dependerá do modo de cada variável BC, porque esse modo será em todos os casos Insert, vê porquê? Por este motivo, não podemos percorrer a coleção e ir executando o Save (só funcionará bem para o primeiro elemento, o que se quer inserir, mas falhará para os demais, já que tentará inserir e encontrará chave duplicada).

A operação, portanto, deverá depender não do modo da variável, mas da existência ou não da chave primária na base de dados. Aqui o método adequado, portanto, outra vez é o InsertOrUpdate. Ao aplicá-lo à coleção, o aplicará a cada item. Para o primeiro, tentará Inserir e conseguirá. Para o segundo, tentará inserir, mas encontrará chave duplicada, então, tentará atualizar e conseguirá, e o mesmo para o último.

Podemos perguntar sobre o resultado da operação, se foi bem-sucedida para toda a coleção, então commitamos, e caso contrário, podemos fazer Rollback, se não desejamos que alguns registros sejam modificados e outros não.

Com base no que foi visto aqui, como declararia o Data Provider para que carregue os dados como os mostramos?

DP Source

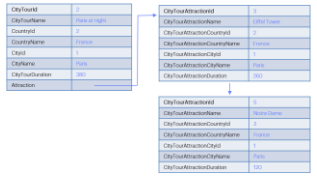
Output

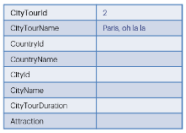
Infer Structure	No
Output	**CityTour**
Collection	**True**
Collection Name	GetCityTours

GetCityTours *

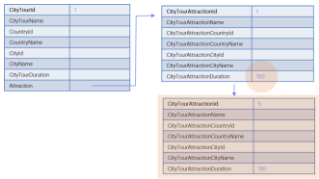
Source * | Rules | Variables

```
1  CityTour
2  {
3       CityTourId = 2
4       CityTourName = Paris at night
5       CountryId = find(CountryId, CountryName = "France")
6       CityId = find(CityId, CityName = "Paris")
7       Attraction
8       {
9            CityTourAttractionId = 3
10           CityTourAttractionDuration = 260
11      }
12      Attraction
13      {
14           CityTourAttractionId = 5
15           CityTourAttractionDuration = 120
16      }
17 }
18 CityTour
19 {
20      CityTourId = 2
21      CityTourName = "Paris, oh la la"
22 }
23 CityTour
24 {
25      CityTourId = 1
26
27      Attraction
28      {
29           CityTourAttractionId = 1
30           CityTourAttractionDuration = 150
31      }
32      Attraction
33      {
34           CityTourAttractionId = 5
35           CityTourAttractionDuration = 120
36      }
37 }
38
```





CityTourId	2
CityTourName	Paris, oh la la
CountryId	
CountryName	
CityId	
CityName	
CityTourDuration	
Attraction	



Aqui o mostramos, sem entrar em detalhes. O Data Provider devolve uma coleção de tipo de dados do Business Component CityTour. Coleção!

Estamos supondo que CityTourId não é autonumerado, e que o 2 não existe. Observemos que temos 3 grupos CityTour, para os três itens da coleção que serão devolvidos. O primeiro carrega todos os dados do cabeçalho e as duas linhas.

O segundo supõe que a operação de Insert sobre o primeiro já se realizou, portanto para o mesmo Citytour, o que vai tentar é modificar seu nome.

E o terceiro modifica o valor da linha de Id 1, e adicionar a linha de id 5 (aqui não se nota a diferença, porque os atributos armazenados do nível Attraction são esses dois unicamente. Mas se a tabela tivesse mais atributos, aqui apareceriam todos (ou todos aqueles aos quais queremos dar valor), enquanto aqui, somente os que queremos modificar.

training.genexus.com
wiki.genexus.com