Atualização da base de dados

Update com Business component. Nos bastidores

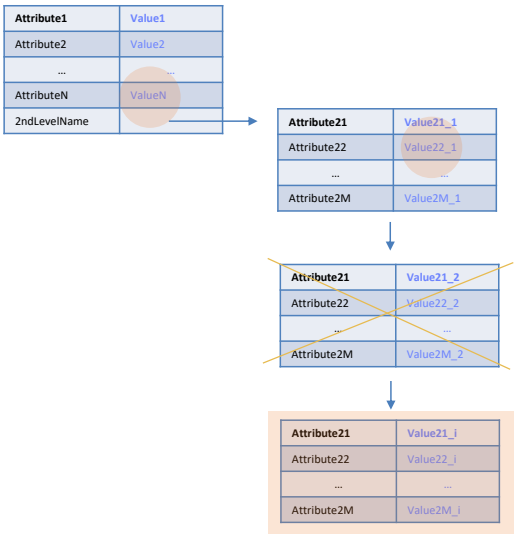
GeneXus™

Transaction

Attribute1*
Attribute2
…
AttributeN

2ndLevelName
{
Attribute21*
Attribute22
…
Attribute2M
}

&BC



&BC.Update()

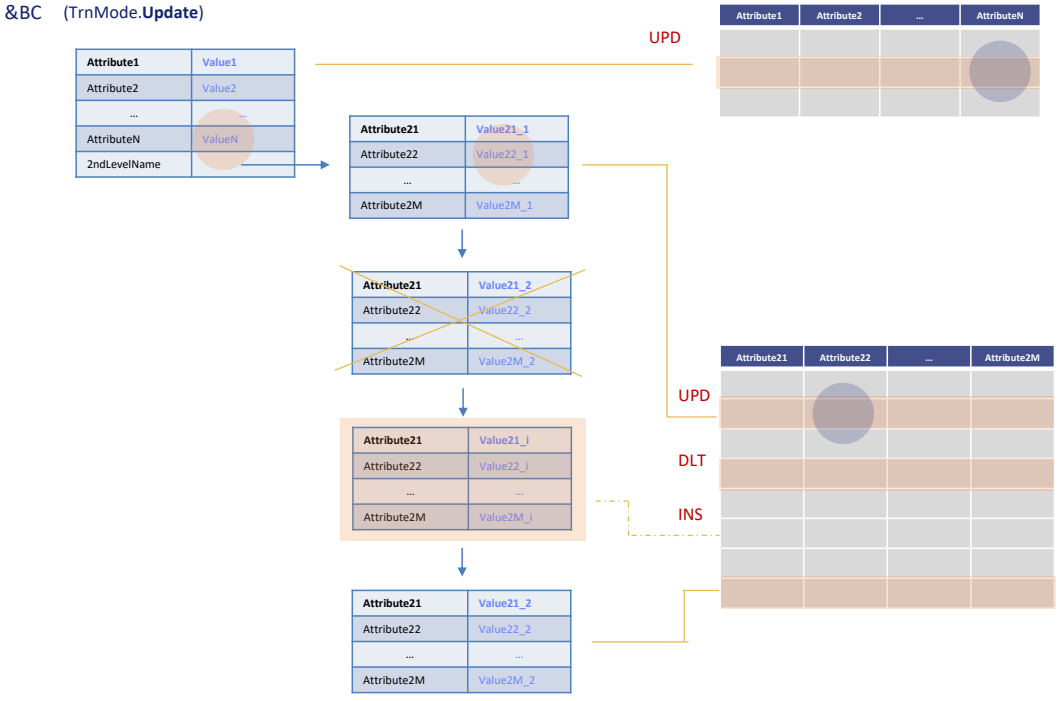
&BC.**Mode**()

TrnMode.Update

Suponhamos que temos uma transação de dois níveis, com a propriedade Business Component ativada e uma variável desse tipo de dados.

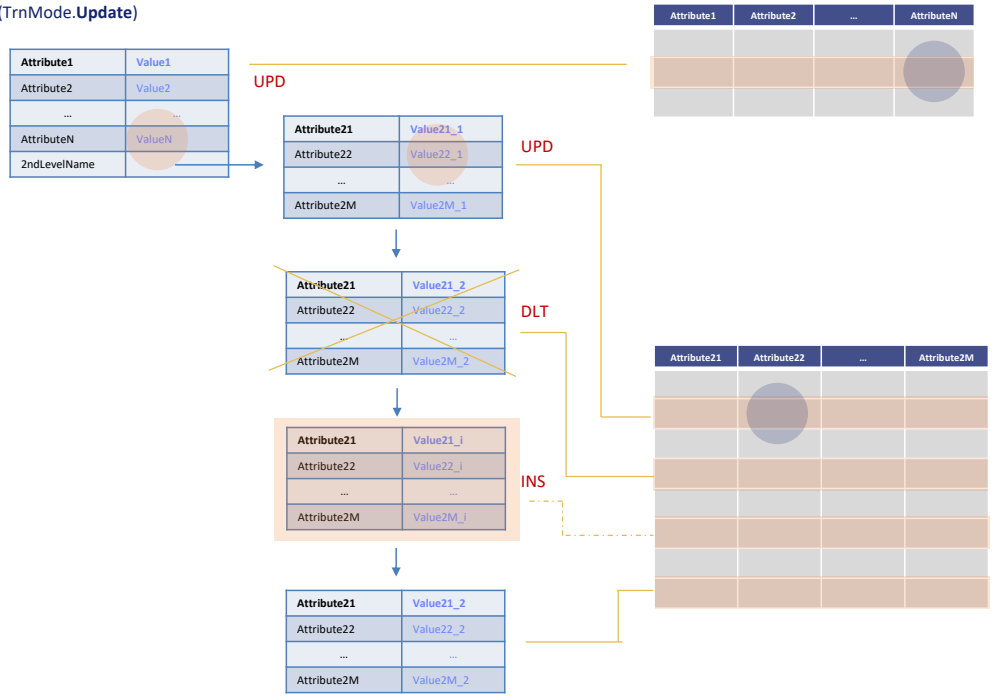
Vimos anteriormente que dependendo do modo da variável, o que será feito internamente ao ser executado o método Update.

Se a variável estava em modo Update, então era atualizado seu cabeçalho (independentemente de ter sido ou não modificada alguma de suas propriedades –ou seja, elementos, ou seja, atributos-) e se foi feito algo na coleção de linhas isso era refletido nos registros da base de dados. Vamos entrar aqui em um pouco mais de detalhes.

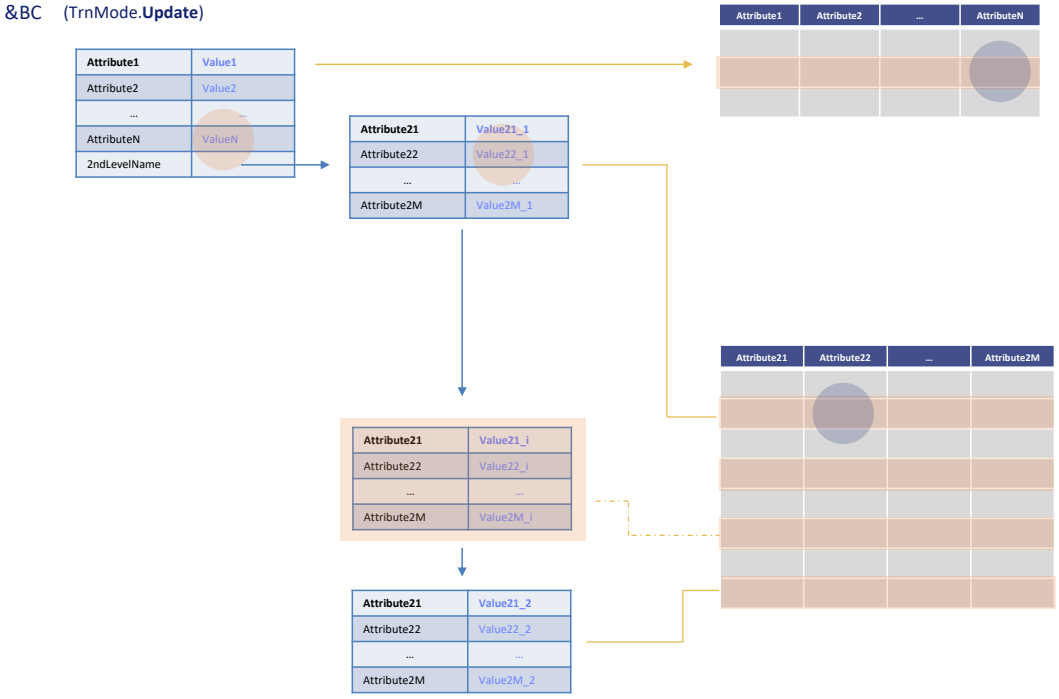


Como é operado realmente por trás? Vai para a base de dados em busca das diferenças entre os registros existentes e os dados contidos no BC no momento do Update?

&BC (TrnMode.**Update**)



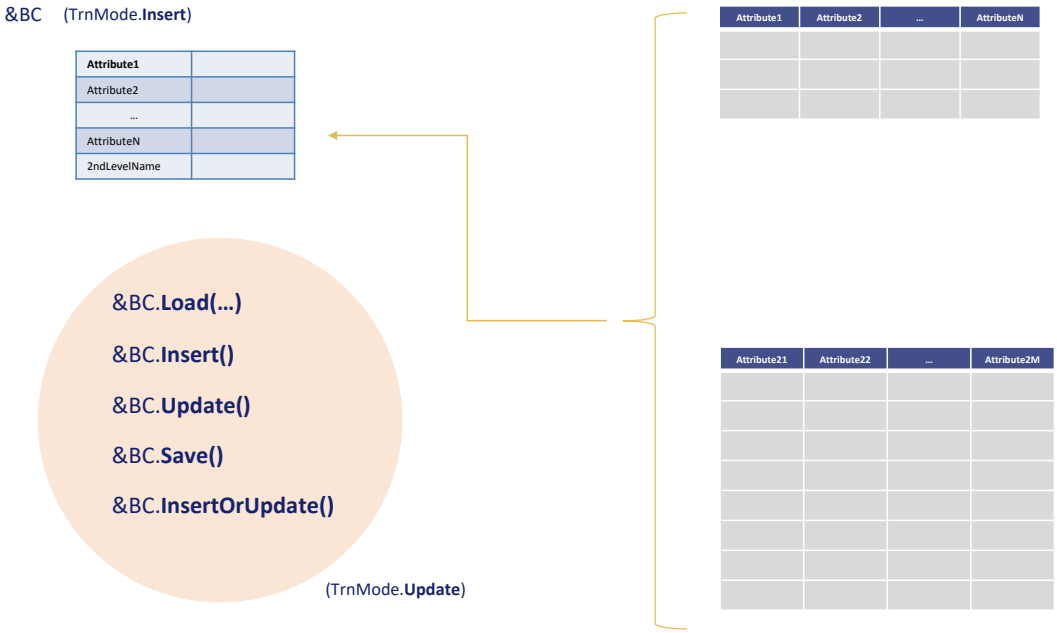
Ou é mantido na memória do BC tudo o que foi feito sobre ele de modo a fazer repercutir essas operações agora sobre a base de dados, sem verificar exatamente a consistência entre memória e base de dados?



Se a resposta fosse a primeira, então teria que acessar o registro na tabela do cabeçalho e comparar todos os campos, para ver se algum mudou e assim atualizar no registro, o campo particular ou os campos particulares que foram alterados. Ou diretamente sobrescrever o registro para economizar a comparação.

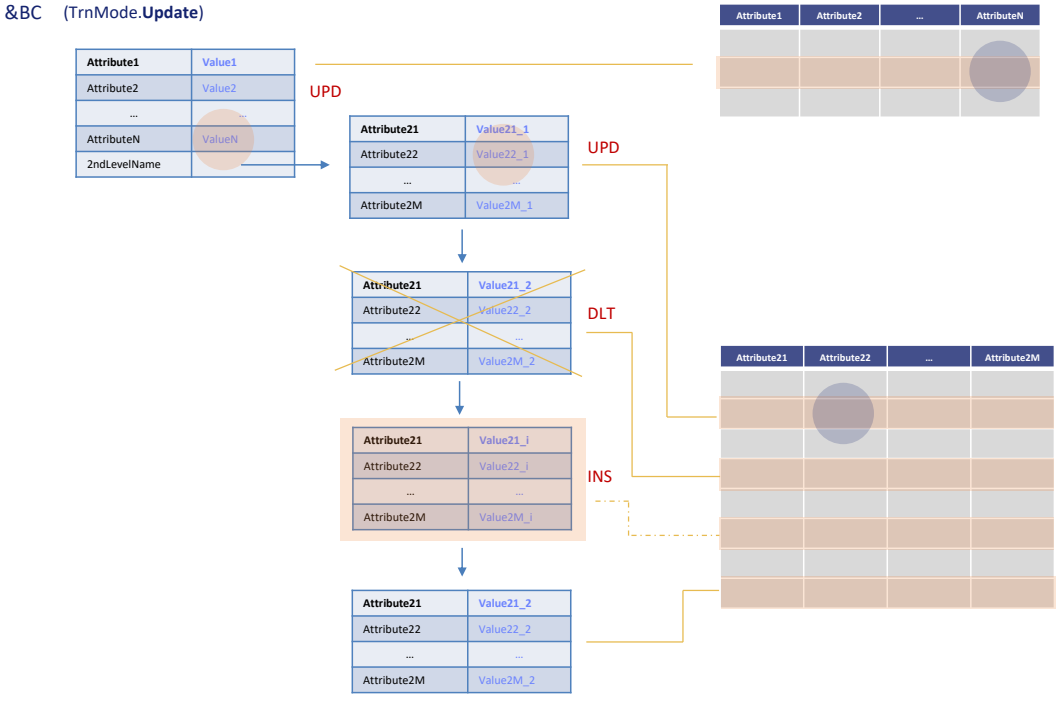
E então acessar todos os registros correspondentes às linhas, e se algum não estiver na coleção do BC, então excluí-lo da base de dados. E, em seguida, percorrer a coleção e, para cada item, ver se o registro existe e, em caso afirmativo, ver se algo mudou para modificá-lo ou sobrescrevê-lo diretamente. Se neste nada, então poderíamos deixá-lo como está. E então ficaríamos com este item que não tem um registro na base de dados, então ele é inserido.

Fazer toda esta comparação seria muito custoso. Não é o que será feito.



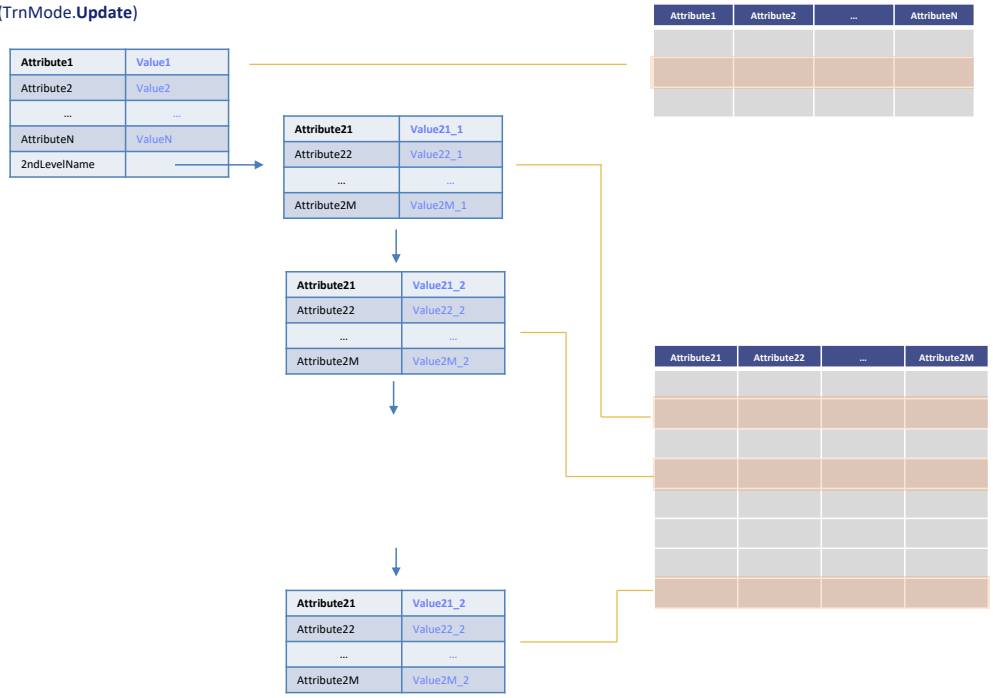
O que GeneXus realmente faz é o segundo. No início da execução do objeto onde está localizada toda variável BC, inicia em modo Insert, pois ao ser declarada, já é reservado espaço de memória vazio para ela.

Para que uma variável BC fique em modo Update, ela deve ter sido carregada da base de dados, seja com Load, seja com uma operação anterior (bem-sucedida) de Insert, Update, Save ou InsertOrUpdate. Só ali fica em modo Update.

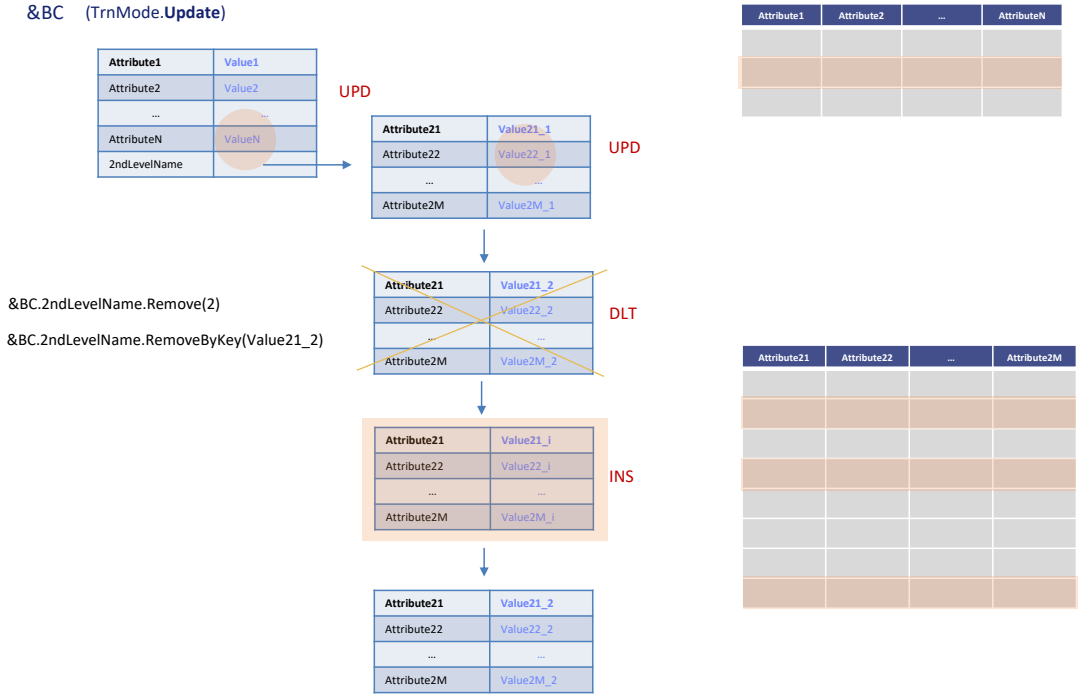


Portanto, a partir da última operação que deixou carregada a variável, ao manipulá-la por código, será registrada dentro de sua informação oculta o que está sendo feito com cabeçalho e com os itens: se atualizá-los, marcá-los para serem excluídos ou inseri-los. Para depois executar essas operações.

&BC (TrnMode.**Update**)



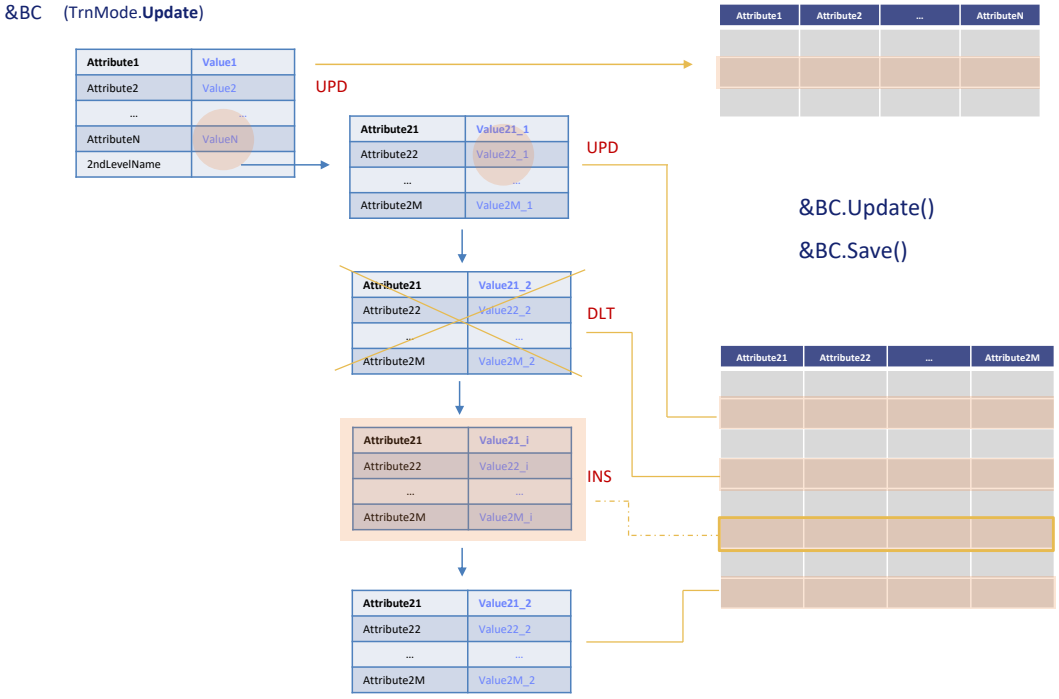
Vamos mais devagar. Suponhamos que foi feito um Load da variável &BC que a deixou em modo Update e com esta informação (que coincide com a da base de dados).



Em seguida, manipulamos por código a variável, fazendo, por exemplo, o seguinte:

- modificando este valor do cabeçalho,
- este deste item,
- removendo este outro item (seja com o método Remove da coleção, solicitando que remova o segundo item, ou o RemoveByKey, para o qual devemos passar o identificador). Ao fazer isto, o item não será realmente removido da coleção, mas **marcado** para ser eliminado, mas continuará estando ali, ou seja, será uma eliminação lógica),
- adicionando este outro,
- e deixando o último como estava.

Então, ficam assim marcados os elementos do BC: cabeçalho e primeiro item para serem atualizados, segundo item marcado para ser excluído, terceiro item marcado para ser inserido e o último em princípio não precisaria ter nenhuma marca, pois nada foi feito com ele. Voltaremos a isso depois.

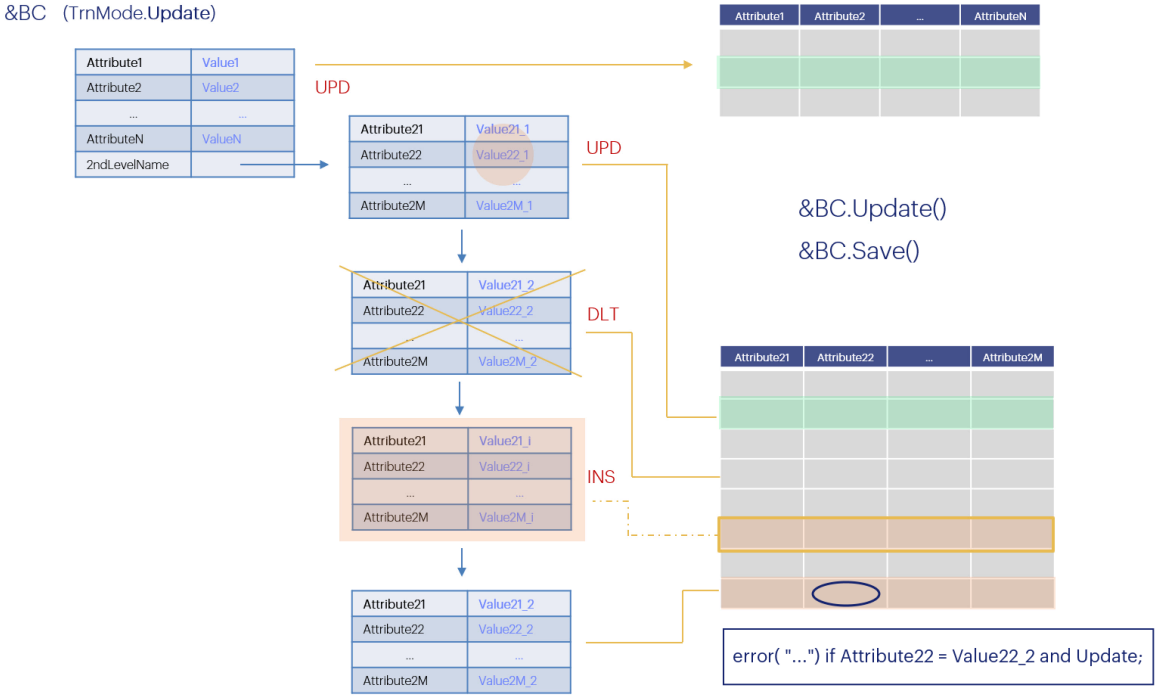


Damos a ordem de Update (ou Save, que neste caso é o mesmo porque a variável está em modo Update).

Tentará sobrescrever o cabeçalho, tenha ou não mudado algo nele (para o qual, claramente, primeiro deverá verificar se há um registro na tabela com essa chave primária, caso contrário já sabemos que falhará por PrimaryKeyNotFound). Isto é o mesmo que faz a transação quando é pressionado Confirm em modo Update. Atualizará o cabeçalho na base de dados mesmo que o usuário não tenha modificado nenhum de seus campos.

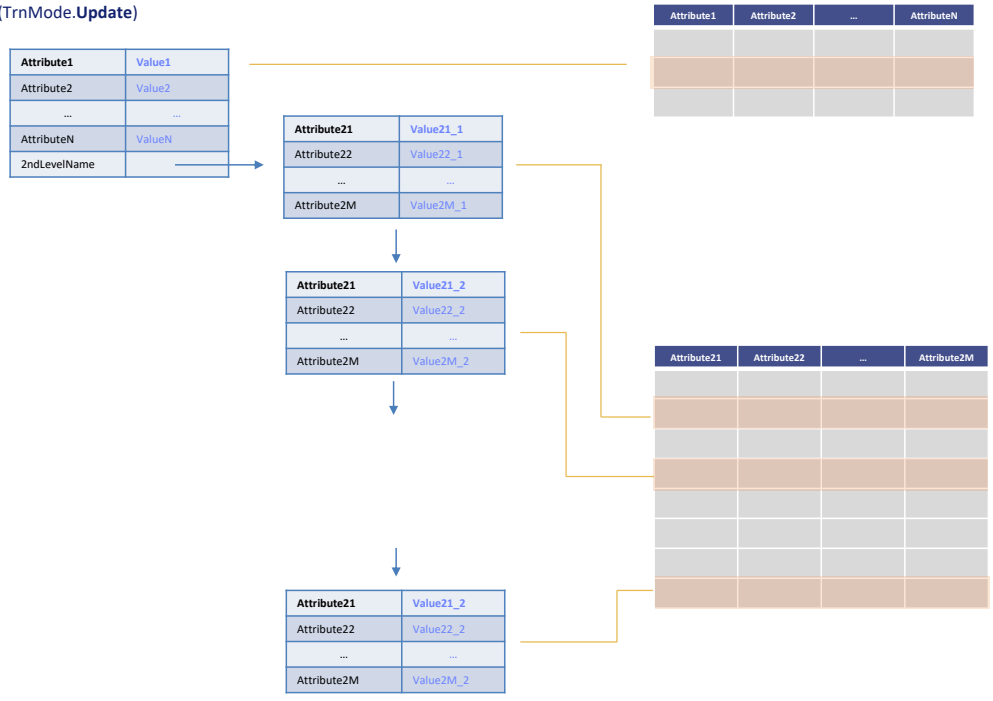
E então, para cada item da coleção:

- se estiver marcado como update, é substituído o registro inteiro,
- se estiver marcado como excluído, então é pesquisado e exclui o registro da tabela,
- se estiver marcado como novo na coleção, então é inserido na tabela,
- E se nada foi feito com ele... bem, aqui há uma diferença com a transação: ainda é sobrescrito. É como se estivesse em modo Update.

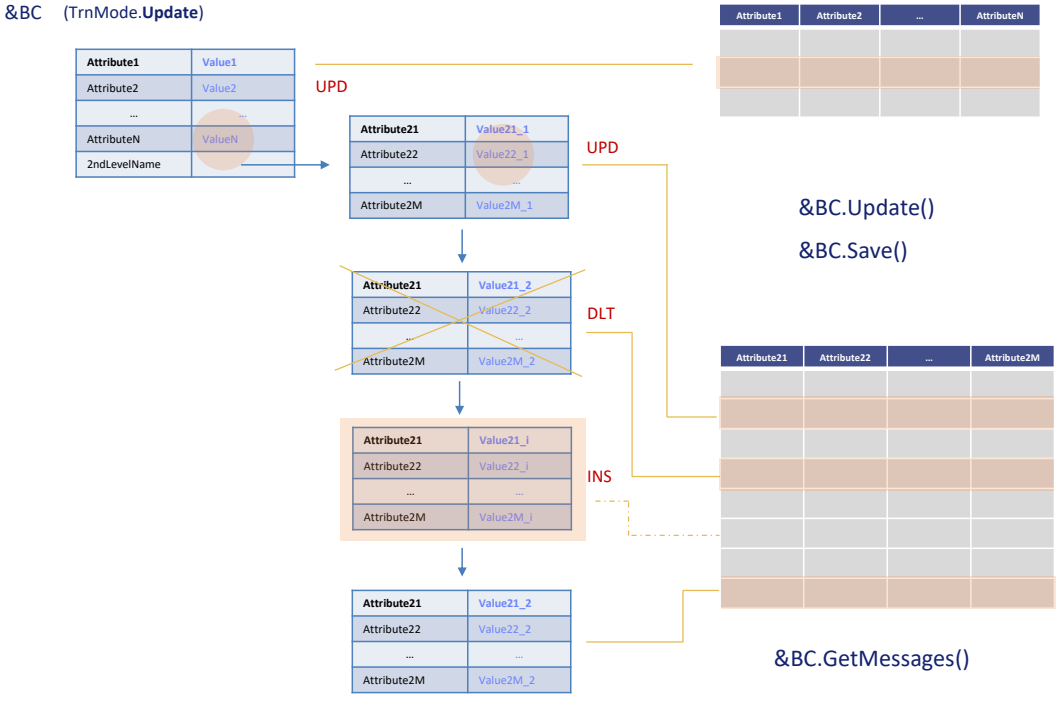


Um aviso: no momento em que este vídeo foi gravado, estava sendo analisado se deveria ou não ser modificado este comportamento para que o business component funcione da mesma forma que a transação. Na transação, se estiver em modo Update e nada foi feito em uma linha, essa linha não é processada. Portanto, nem sequer são avaliadas as regras para ela. Em vez disso, para o BC, como é processada a linha, já que é atualizada de qualquer maneira, serão avaliadas e disparadas suas regras. É importante estar ciente disso.

&BC (TrnMode.**Update**)

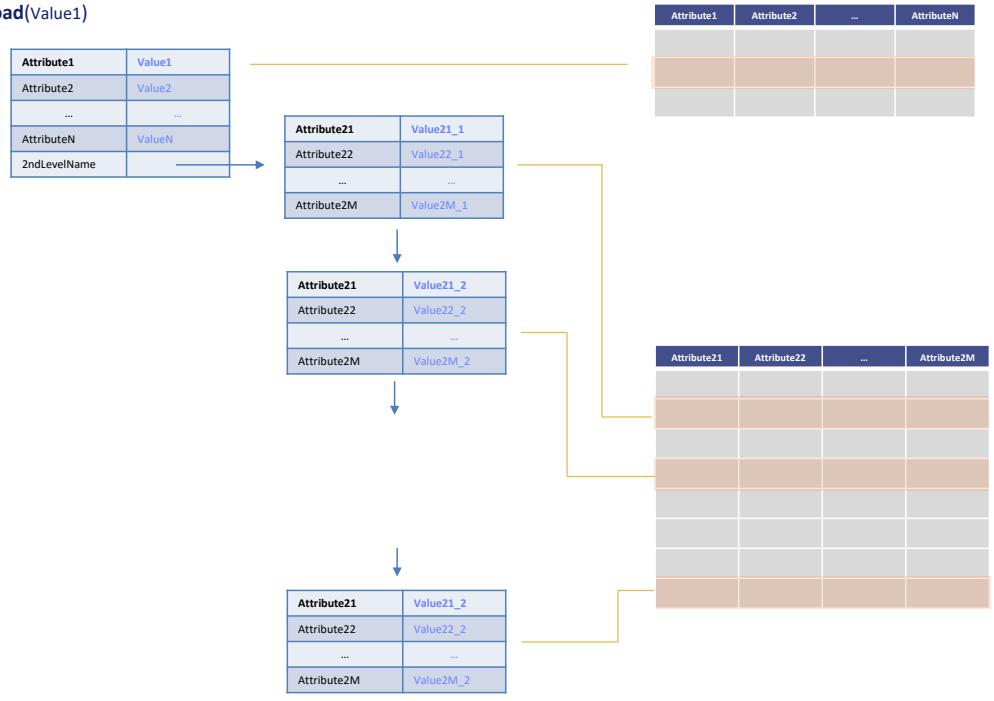


Voltemos ao nosso. Claro que esta solução se baseia em uma hipótese: que o conteúdo do BC quando foi carregado…

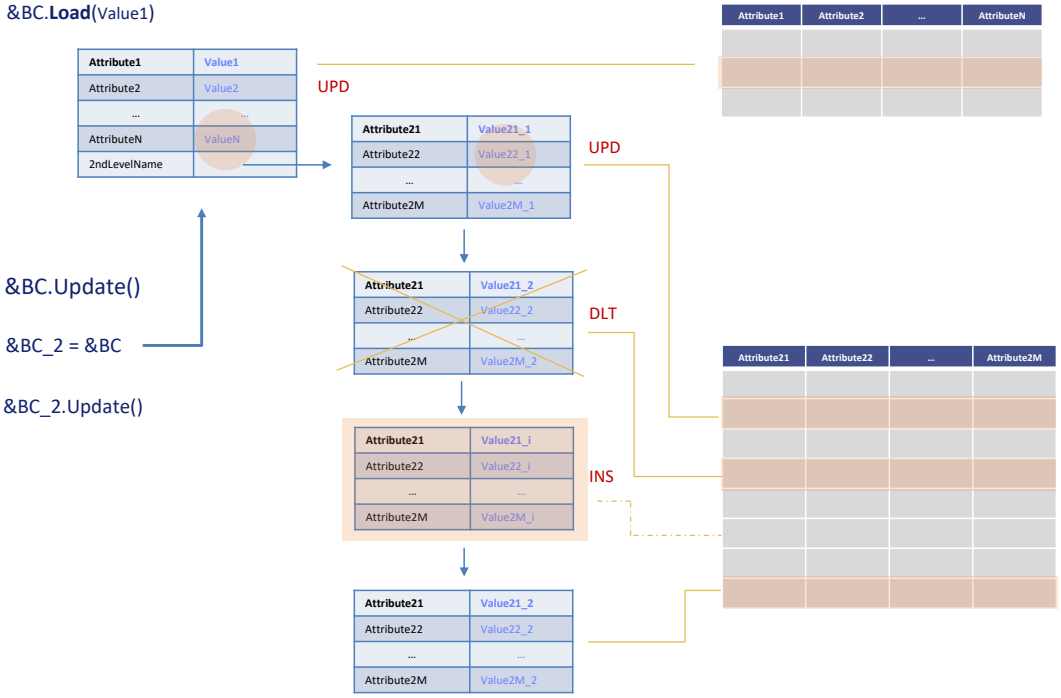


…antes de fazer as alterações por código, corresponde ao estado da base de dados imediatamente antes de executar a operação de Update ou Save. Deve-se ter isso em mente, pois se alguém alterou o estado da base de dados para estes registros nesse meio tempo, o resultado dependerá de tais alterações, por isso é conveniente revisar as mensagens produzidas após o Update ou Save.

&BC.**Load**(Value1)



E o que aconteceria se, por exemplo, carregássemos primeiro a variável &BC da base de dados…

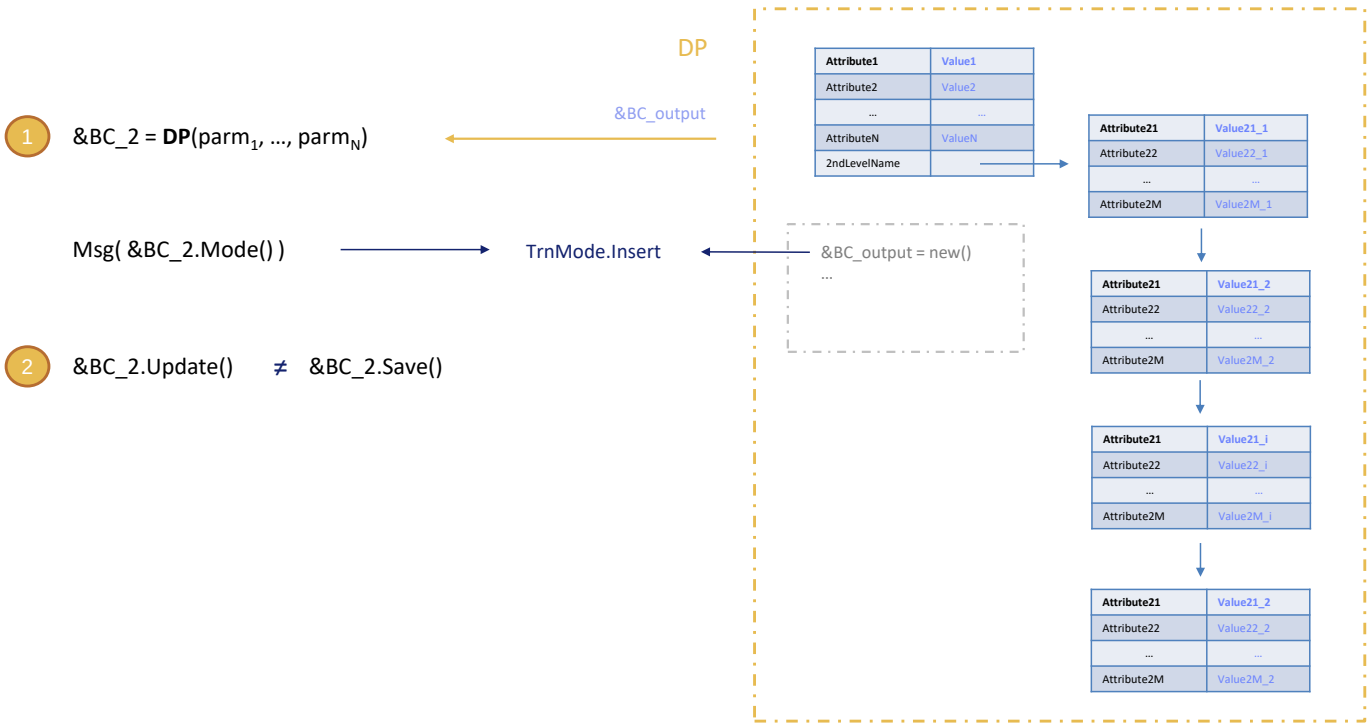


… a manipulássemos como fizemos aqui, e então ao invés de fazer um Update ou Save, a atribuíssemos a outra variável BC, por exemplo uma à qual tivéssemos aplicado um new antes da atribuição? E então aplicássemos o Update a esta nova variável.

Poderíamos acreditar que aqui a coisa fica mais complexa, pois a variável &BC_2 estará em modo Insert e não Update, por causa do new. E também poderíamos ser levados a acreditar que terá sido perdido o registro das operações que foram realizadas nos itens. Mas nada disto será verdade.

Embora a variável &BC_2 ficou em modo Insert após esta atribuição, esta outra jogará por terra aquilo. É que na verdade não apontará mais para o novo espaço de memória, mas passará a apontar para o da variável &BC. É que a atribuição não faz uma cópia do atribuído, mas sim aponta exatamente para esse mesmo local, como um ponteiro. Por esta razão, assumirá tanto o modo da variável &BC, quanto o registro das operações. E quando chegar a requisição de Update, será o mesmo que ter feito o Update da variável &BC.

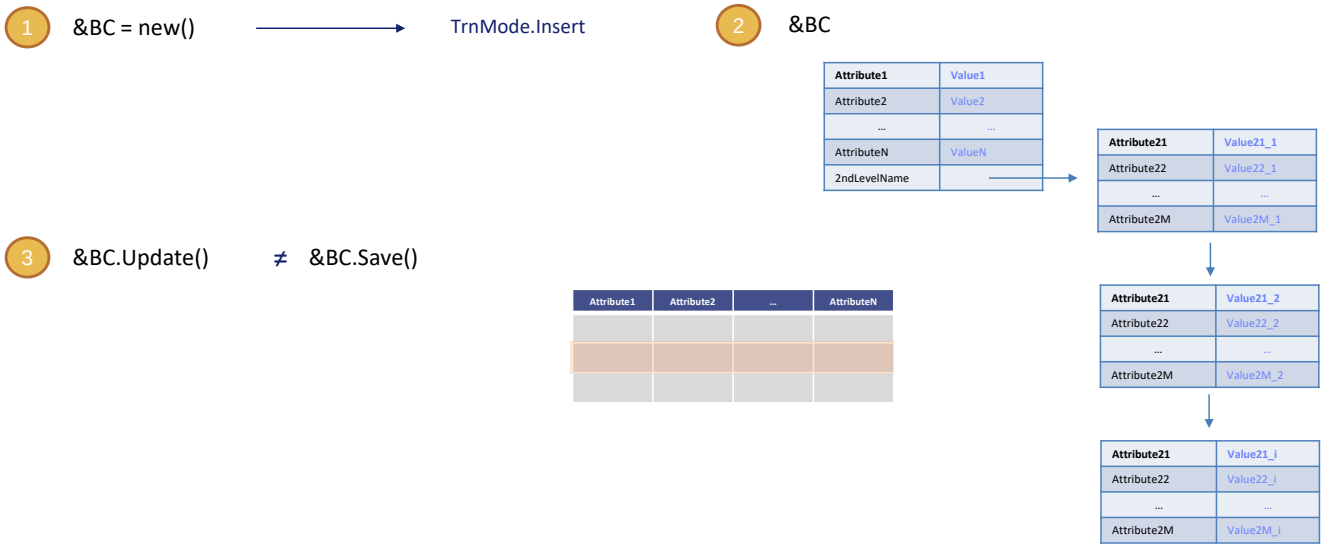
Se o modo de &BC_2 neste ponto for Insert, neste outro será o modo de &BC, que em nosso exemplo era Update.



Por que isso não funciona assim quando a uma variável BC atribuímos o resultado de um Data Provider?

Se perguntarmos aqui o modo da variável &BC_2 ficaremos surpresos ao descobrir que é Insert. É que a primeira coisa que o Data Provider faz internamente é um new, então a variável que vai retornar estará, inevitavelmente, em modo Insert.

Portanto, este caso é equivalente ao que ainda não analisamos aqui, mas nos vídeos do curso Advanced: o que acontece quando a variável está em modo Insert e é solicitada uma operação de Update. Aqui não dá no mesmo se a operação solicitada é a de Update e não a de Save.



É que, pensemos na de Save. Suponhamos que primeiro solicitamos memória, então a variável está em Insert, depois atribuímos valores às propriedades do cabeçalho e adicionamos 3 linhas, e depois executamos o Save. Tentará inserir o BC na base de dados, o que só poderá ser bem-sucedido se os valores indicados no BC para a chave primária não corresponderem a um registro existente na tabela da base de dados para o cabeçalho. Lembremos que o método Save sempre tentará realizar a operação que corresponde ao modo em que se encontra a variável. Neste caso, se o registro existir, falhará o Save().

Em vez disso, o que acontecerá com o Update quando o modo for Insert? Neste caso, como a variável está em modo Insert, pode-se supor que não foi carregada da base de dados. Em outras palavras, é cega para o registro da base de dados que o desenvolvedor está supondo que existe (supõe que existe, claramente porque caso contrário não solicitaria explicitamente um Update).

Resumindo, o desenvolvedor carregou a variável com valores que não precisam terem sido extraídos da base de dados. Isto significa, logicamente, que não realizou a operação de Load, pois nesse caso a variável estaria em Update se esse Load tivesse sido bem-sucedido.
Então é de se supor que para a chave primária foram atribuídos valores que o desenvolvedor supõe que correspondem a um registro existente, pois se não for assim, falhará a operação de Update.

Como funciona aqui o Update?

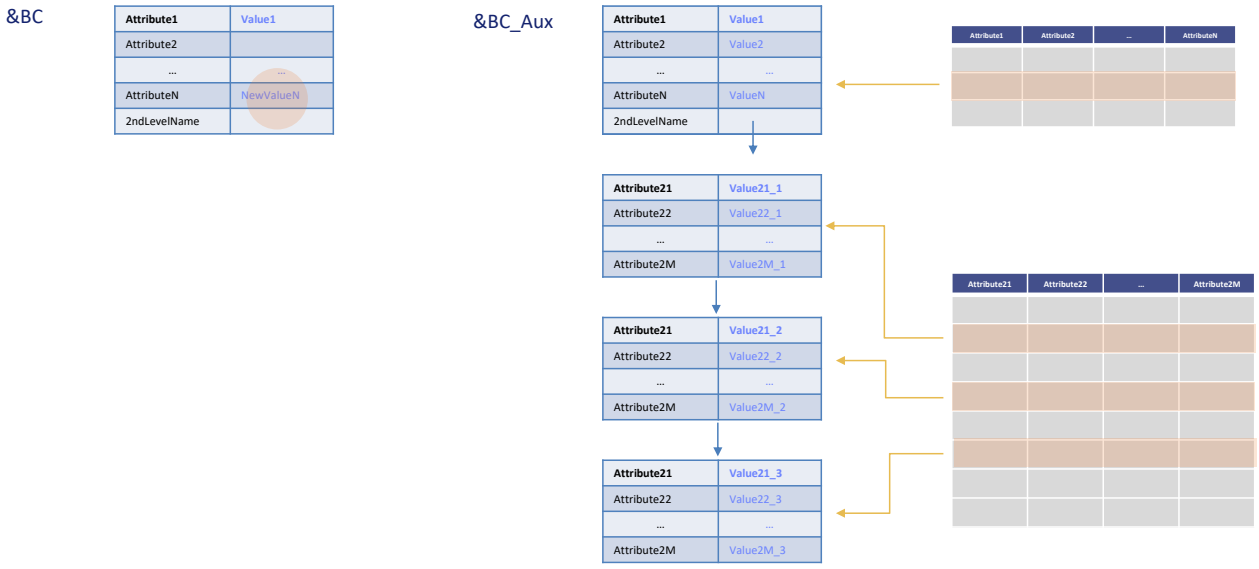
&BC

Attribute1	
Attribute2	
...	...
AttributeN	
2ndLevelName	

Attribute1	Attribute2	...	AttributeN

Attribute21	Attribute22	...	Attribute2M

Suponhamos que esta seja a variável sobre a qual fizemos um new ou que utilizamos pela primeira vez dentro do código, então estará em modo Insert.



&BC.Update()

Depois apenas atribuímos valor à propriedade que corresponde à chave primária, de forma a poder identificar o registro da base de dados que corresponderá ao cabeçalho. E damos valor à propriedade que queremos modificar, aquela que corresponde a este atributo. Todas as outras propriedades deixamos intocadas, ou seja, ficam vazias.

Então manipulamos a coleção de linhas, agora veremos o que podemos fazer, e então executamos o método Update.

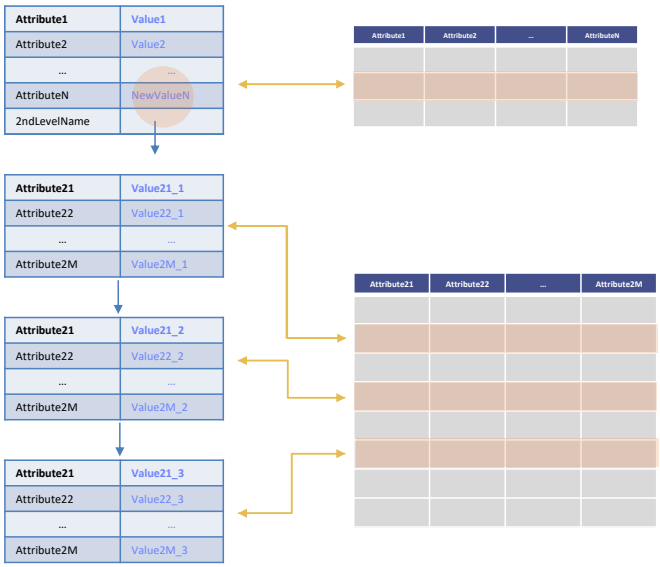
Internamente, o método detecta que a variável está em Insert, então criará uma variável auxiliar sobre a qual aplicará um Load com os valores da chave primária da variável do desenvolvedor. Portanto, se o registro existir, será carregada na variável auxiliar exatamente a informação da base de dados para esse identificador.

O que faz a seguir? No exemplo em que apenas foi modificada uma propriedade de cabeçalho…

&BC

Attribute1	Value1
Attribute2	
...	...
AttributeN	NewValueN
2ndLevelName	

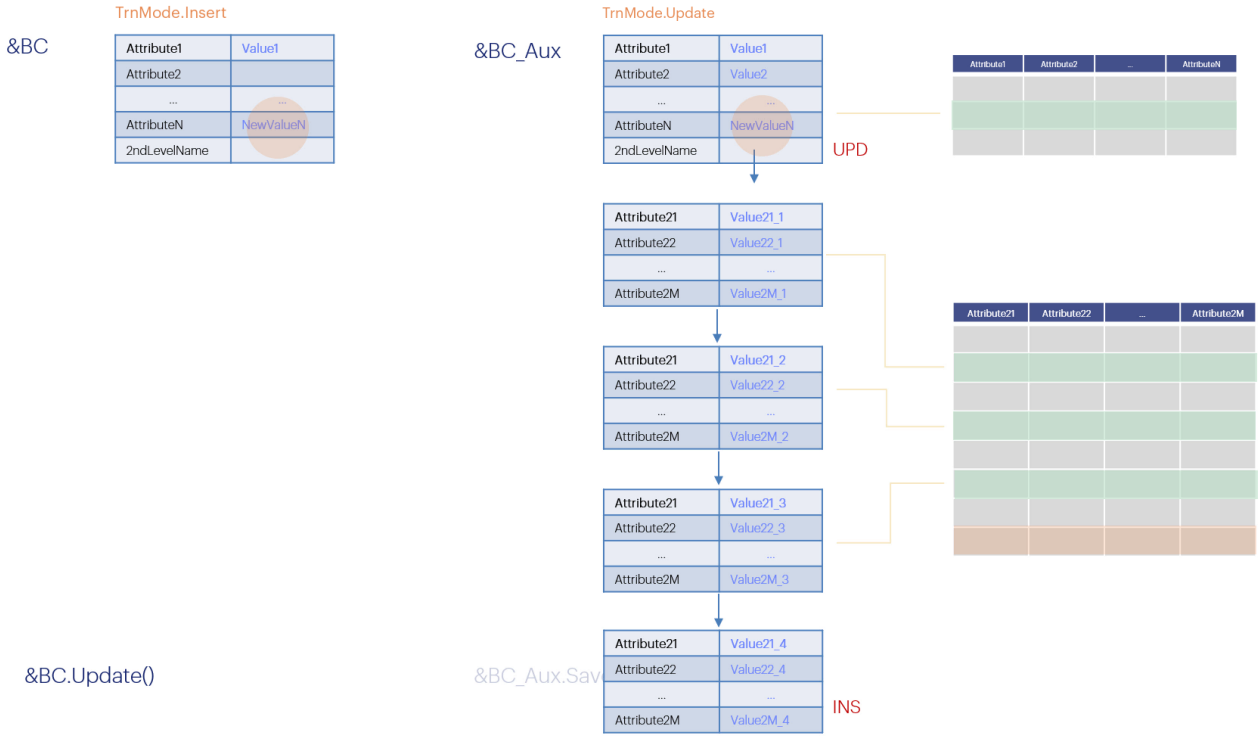
&BC_Aux



&BC.Update()

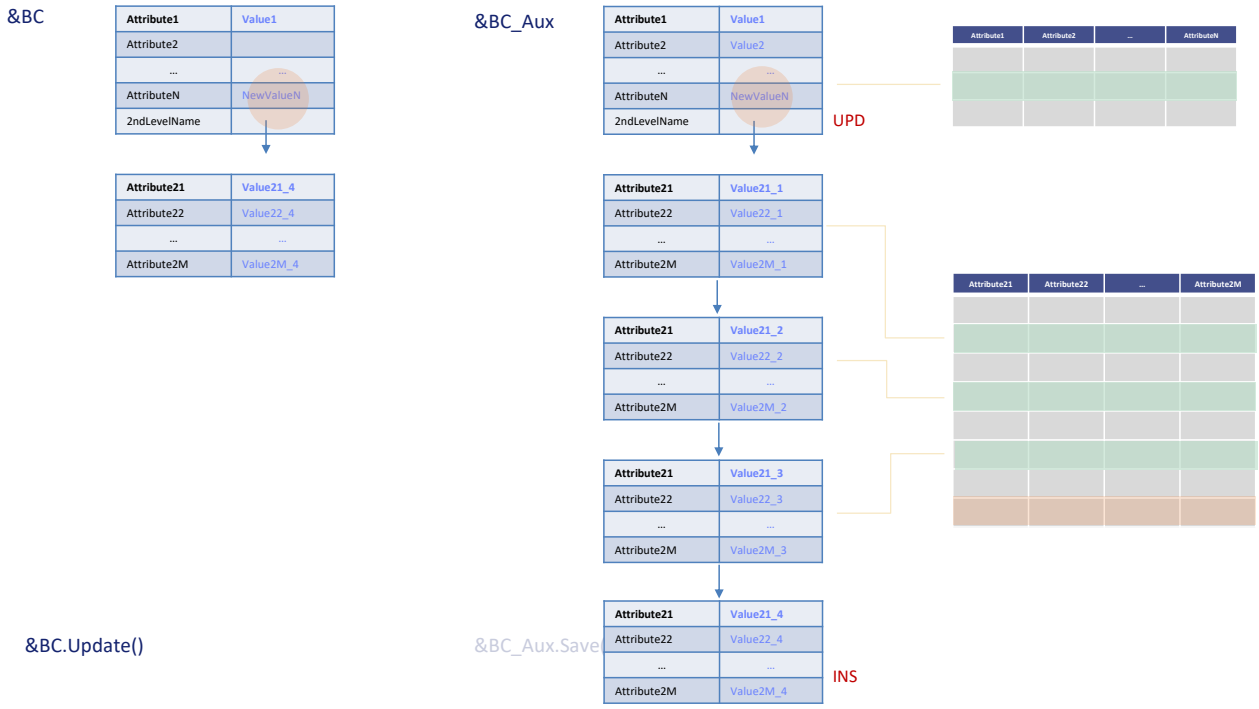
&BC_Aux.Save()

…copia esse valor para a mesma propriedade da variável auxiliar, sobrescrevendo-a. E sobre a variável BC auxiliar executa um Save, que é o mesmo que analisamos no início do vídeo, dado que a variável auxiliar está em modo Update.

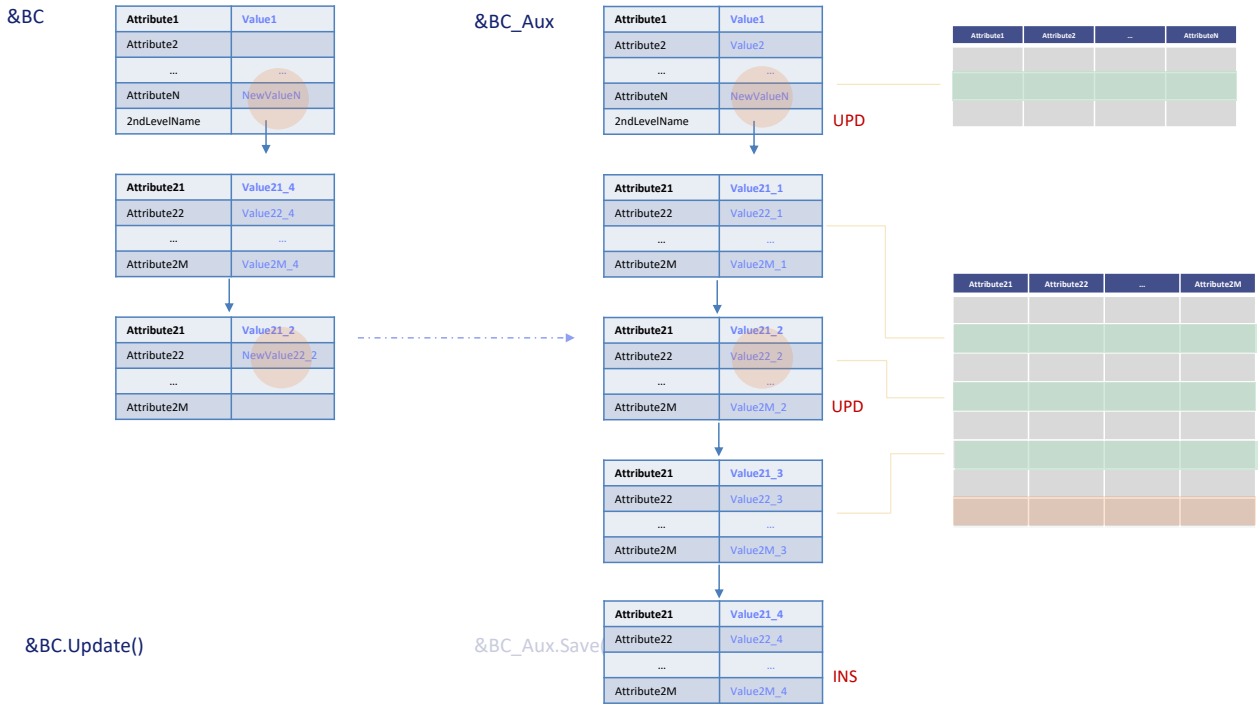


Agora vamos pensar na questão das linhas e fazer a explicação completa.

Suponhamos que o que queríamos era, além de modificar este atributo, adicionar um registro novo à tabela do segundo nível. Para conseguir que seja feito isto sobre a variável auxiliar que afetará o Insert deste registro na base de dados…

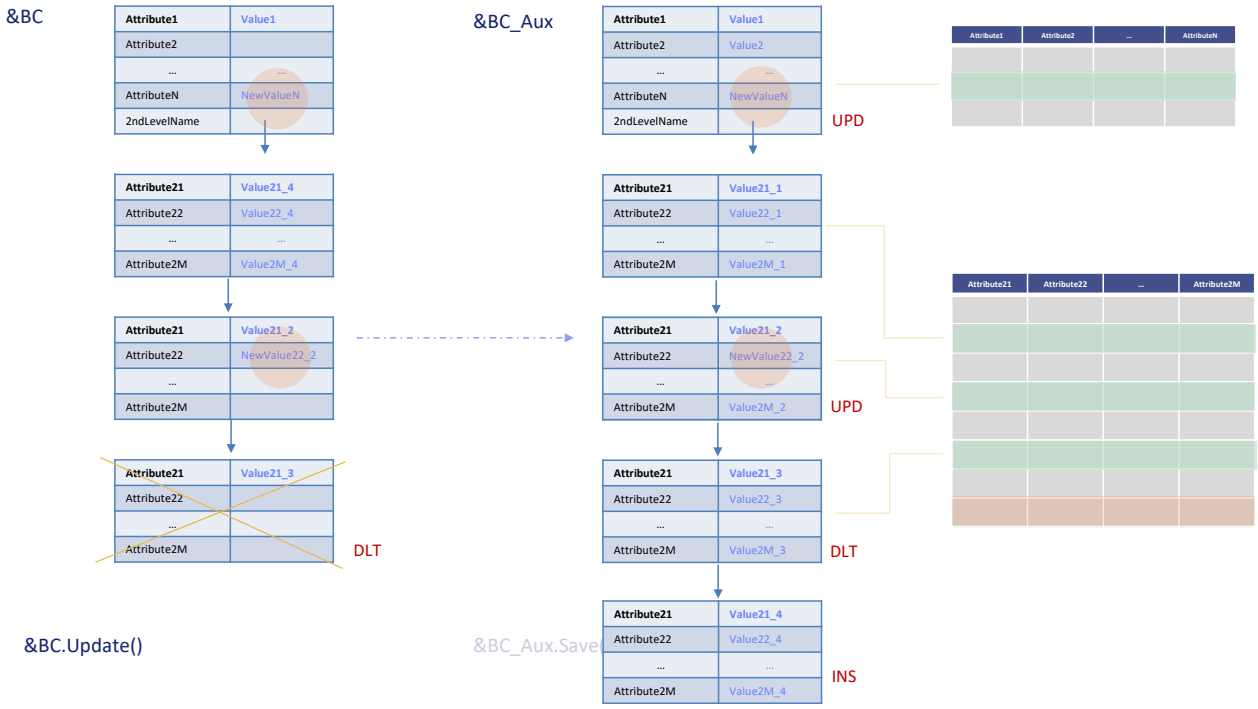


...adicionamos um item à coleção de nossa variável e atribuímos todos os seus valores. O que acontecerá por trás é que quando for avaliar este item para ver o que é feito com ele sobre a variável auxiliar, será pesquisado com o método GetByKey sobre a coleção de linhas se existe ou não uma com este valor. Se não for encontrada, então será adicionada.



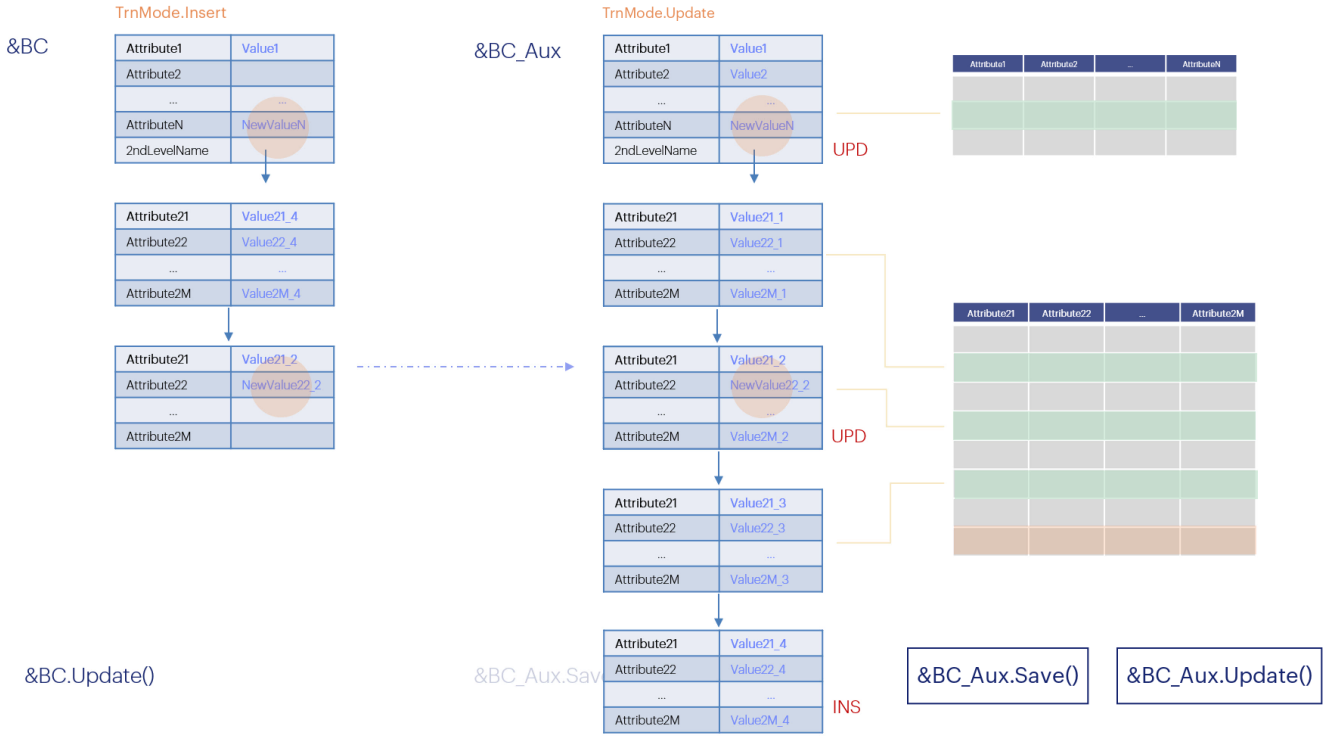
Então suponhamos que queremos atualizar um registro existente, por exemplo este valor, então adicionamos outro item, mas para o qual apenas atribuímos valor ao identificador e à propriedade que corresponde ao atributo que queremos modificar. Indicamos apenas o valor novo. Todas as demais propriedades deixamos intocadas, ou seja, vazias.

Ao analisar o que fazer com este item sobre a variável auxiliar, é feito um GetByKey, e como é encontrado item, é copiado apenas o que foi tocado, ou seja, esta propriedade.



E por último suponhamos que queremos eliminar um registro, então poderíamos supor que basta adicionar outro item, ao qual apenas atribuímos valor à propriedade que o identifica, e então executamos um Remove ou RemoveByKey **desse item**, de maneira que fique marcado para ser excluído.

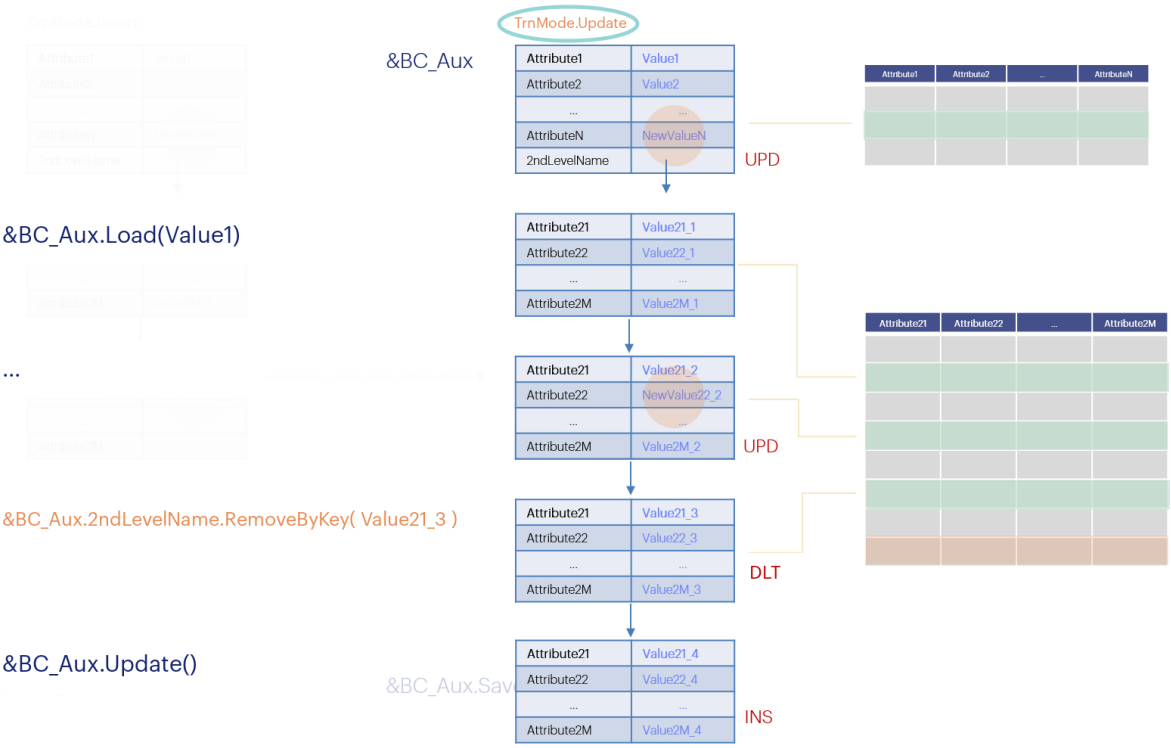
No entanto, isto não funcionará assim. Se a variável BC estiver em modo Insert, quando é excluído o item da coleção, ele não é deixado marcado para ser eliminado, mas diretamente é excluído. Isto significa que **não** podemos **excluir linhas** com variáveis BC em **modo Insert**. Necessariamente deverão estar em modo Update para poder fazer isto, pois somente nesse modo deixa marcado o item para ser eliminado.



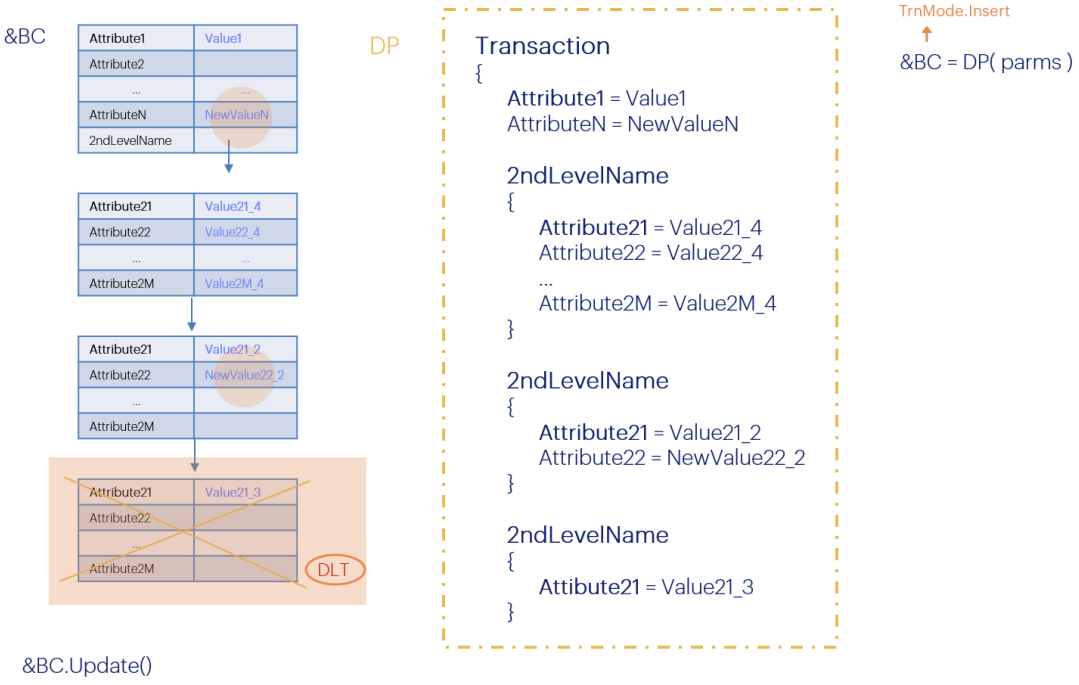
Em última análise, então, é utilizada uma variável em Insert somente quando é desejado modificar e/ou adicionar linhas, mas não excluir.

E o que o desenvolvedor faz sobre esta variável nova é indicar apenas os valores que deseja modificar do cabeçalho, e para cada linha que deseja modificar, seu identificador e valor ou valores a modificar; e para uma linha nova fornece todos os seus valores.

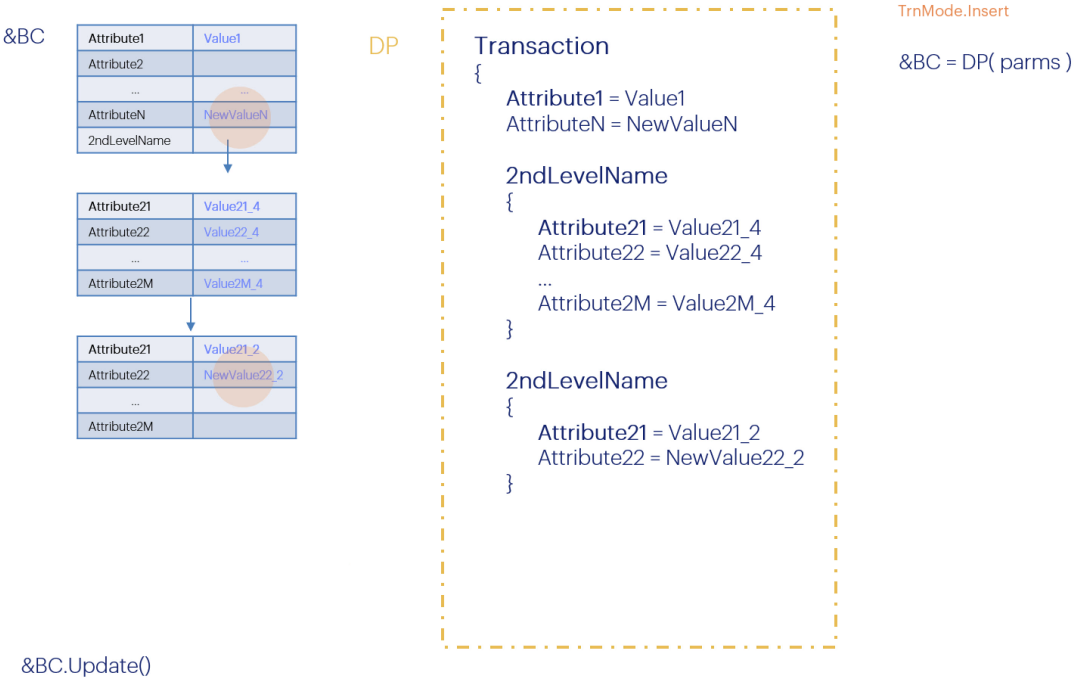
Isto terá as consequências que acabamos de analisar sobre a variável auxiliar, sobre a qual então será feito o Save(), que nela coincide com o Update.



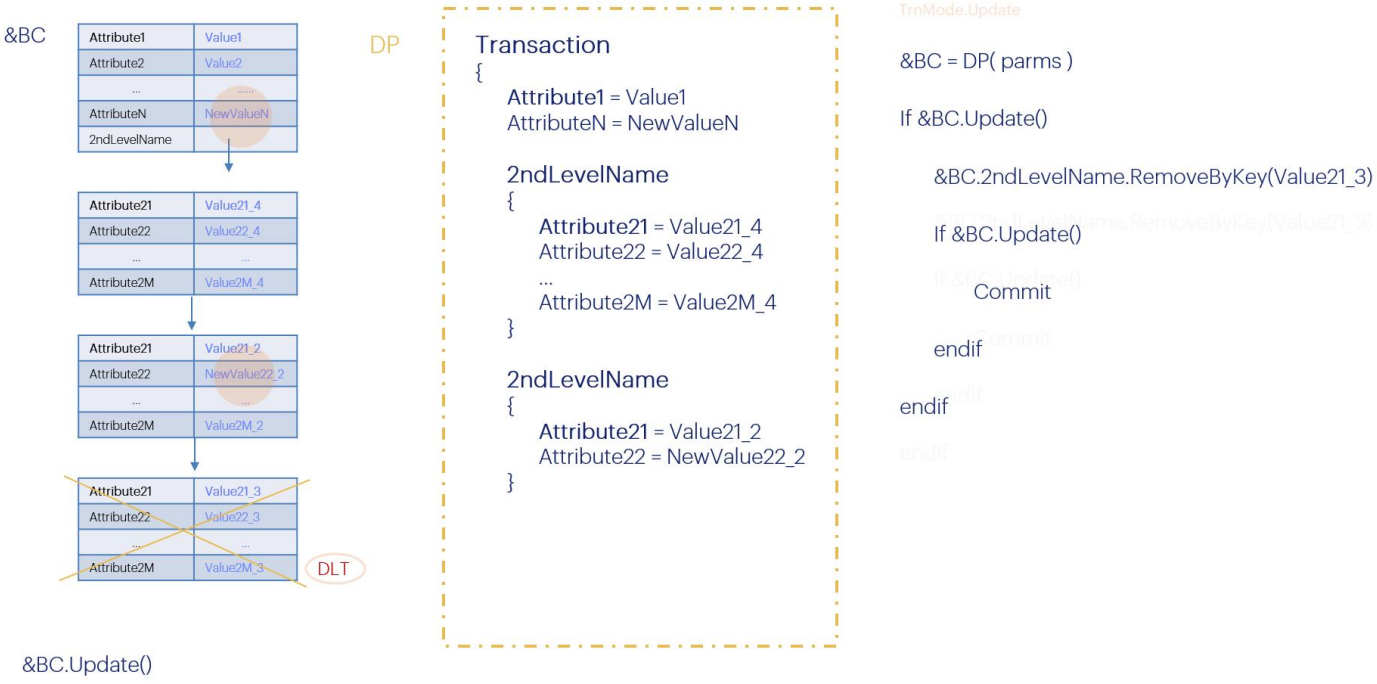
Pelo que vimos, então, se quiséssemos não apenas modificar dados, mas também eliminar linhas, será conveniente fazer um Load explícito para que a variável fique carregada da base de dados e ali fazer todas as operações, incluindo a exclusão de itens.



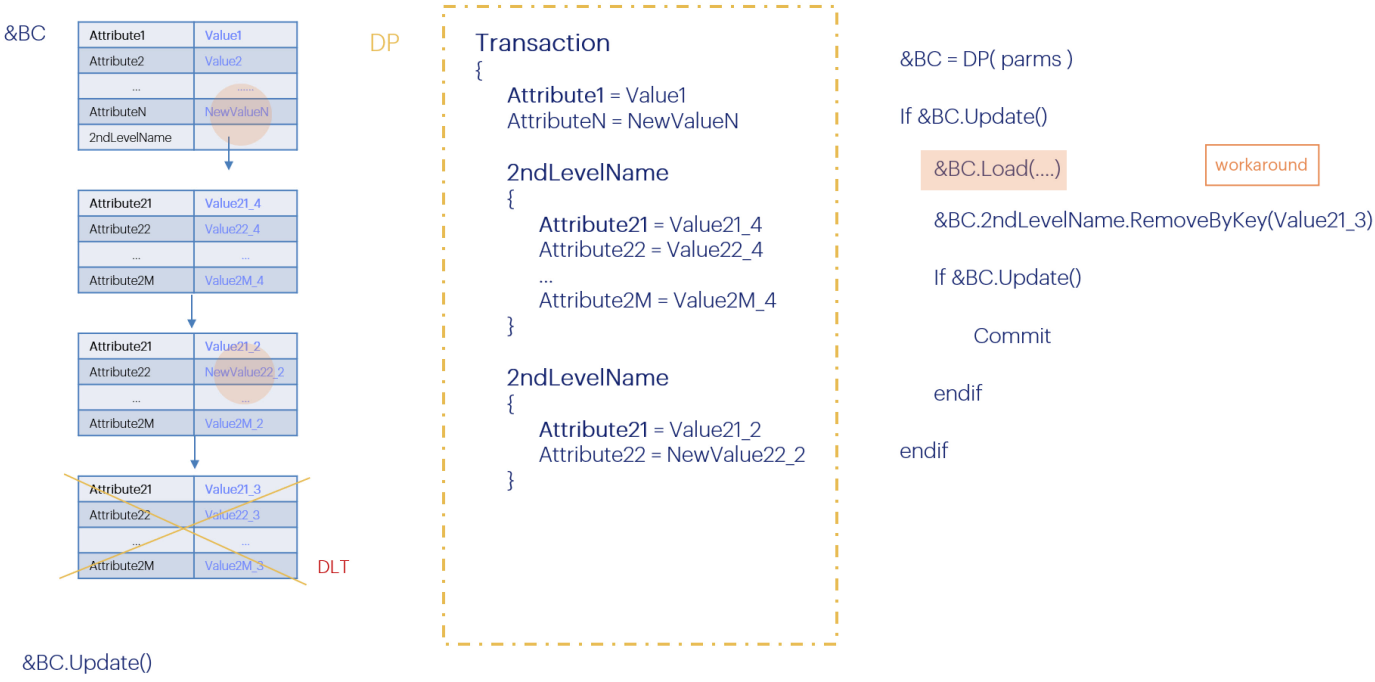
No entanto, se vamos utilizar um Data Provider para carregar a variável &BC, como nesse caso sabemos que estará inevitavelmente em modo Insert, como conseguimos eliminar esta linha?



Para obter o que buscamos, neste caso não deveríamos adicionar o item à variável, claramente, mas apenas deveríamos fazer as alterações relativas às modificações e novos itens…



…executar o Update e então, se for bem-sucedido, como a variável ficará em modo Update, ali sim podemos adicionar o Remove ou RemoveByKey pois lá após o Update sabemos que a variável ficará carregada com toda a informação da base de dados, incluindo a linha que queremos eliminar. Em seguida, o RemoveByKey a marcará para ser removida no próximo Update, que é o que deveremos executar para que esta ação seja realizada. Então podemos commitar.



Porém, na versão 17 de GeneXus existe um bug pelo qual se a variável &BC estava em modo Insert ao fazer o Update, mesmo que este tenha sucesso, não atualizará o conteúdo da variável, deixando-a como a base de dados e, portanto, com a linha que queremos eliminar carregada. Para a versão 18 estará solucionado, mas por enquanto, então, como workaround, poderíamos tornar explícita a carga.

Com isto concluímos a análise teórica detalhada sobre como se atualiza através do Business Component.

No próximo vídeo veremos tudo isto com um exemplo.

training.genexus.com
wiki.genexus.com
training.genexus.com/certifications