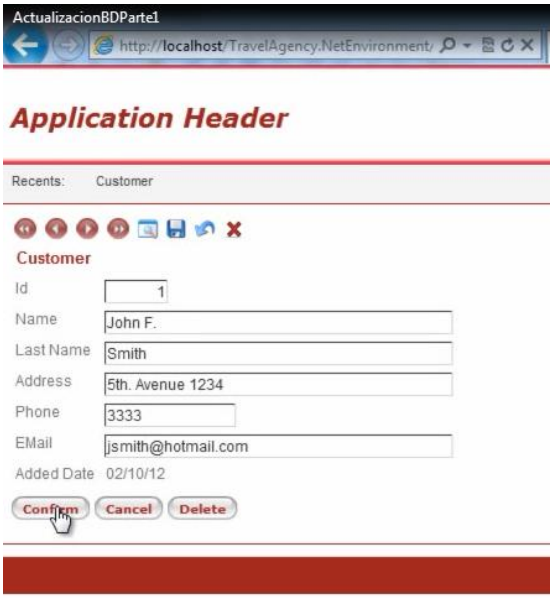
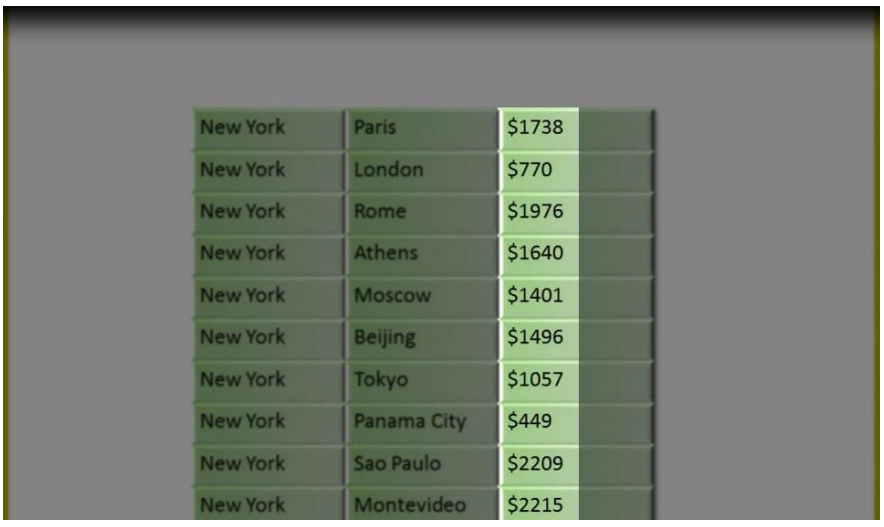
Curso GeneXus - Atualização do banco de dados - Business Components

Até o momento, vimos que as transações permitem aos usuários inserir, modificar e excluir dados nas tabelas do banco de dados do aplicativo.



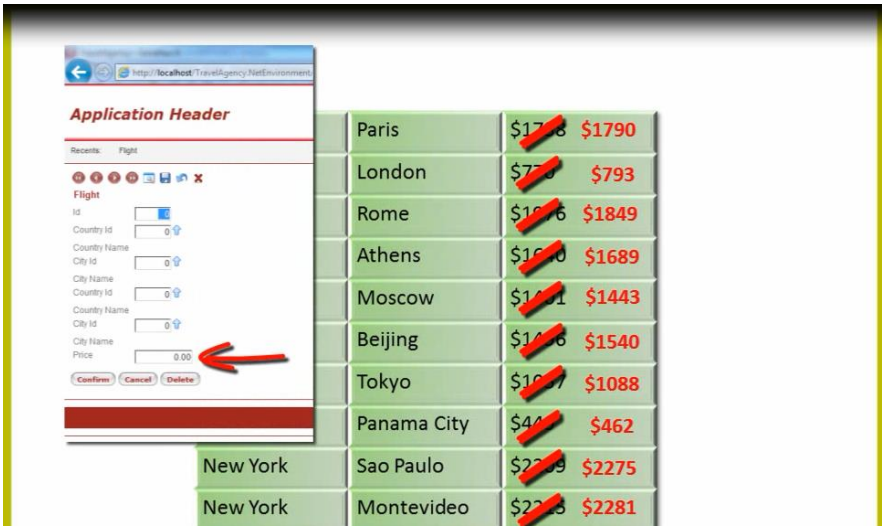
Além desse meio para atualizar o banco de dados, em certas ocasiões é preciso definir algum processo automático de atualização.

Imaginemos, por exemplo, uma agência de viagens que precise periodicamente, aumentar os preços de **todos os voos registrados**,



aplicando ao preços vigentes determinada porcentagem de aumento.

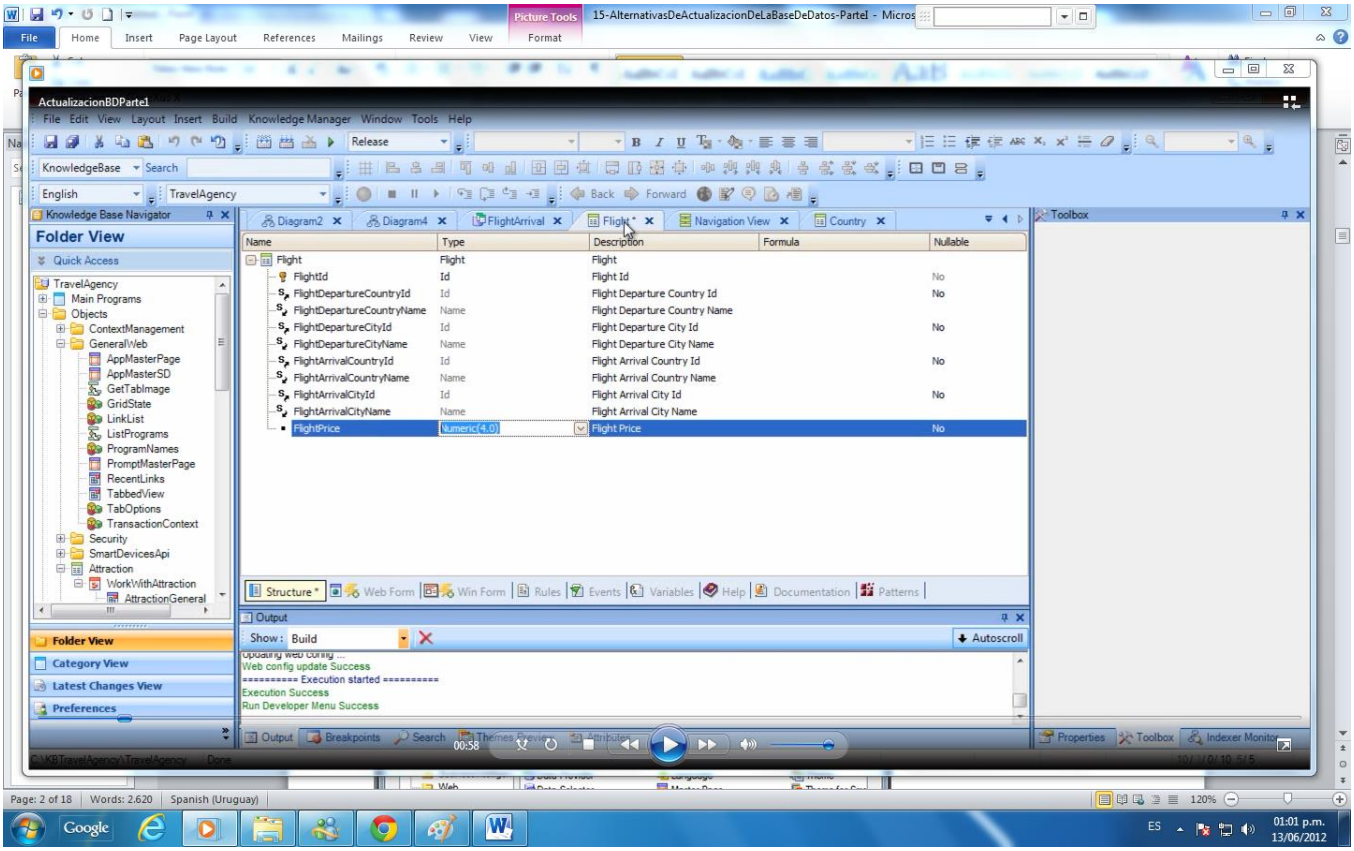
Seria muito entediante utilizar a tela de transação para editar manualmente cada preço das centenas de voos registrados,



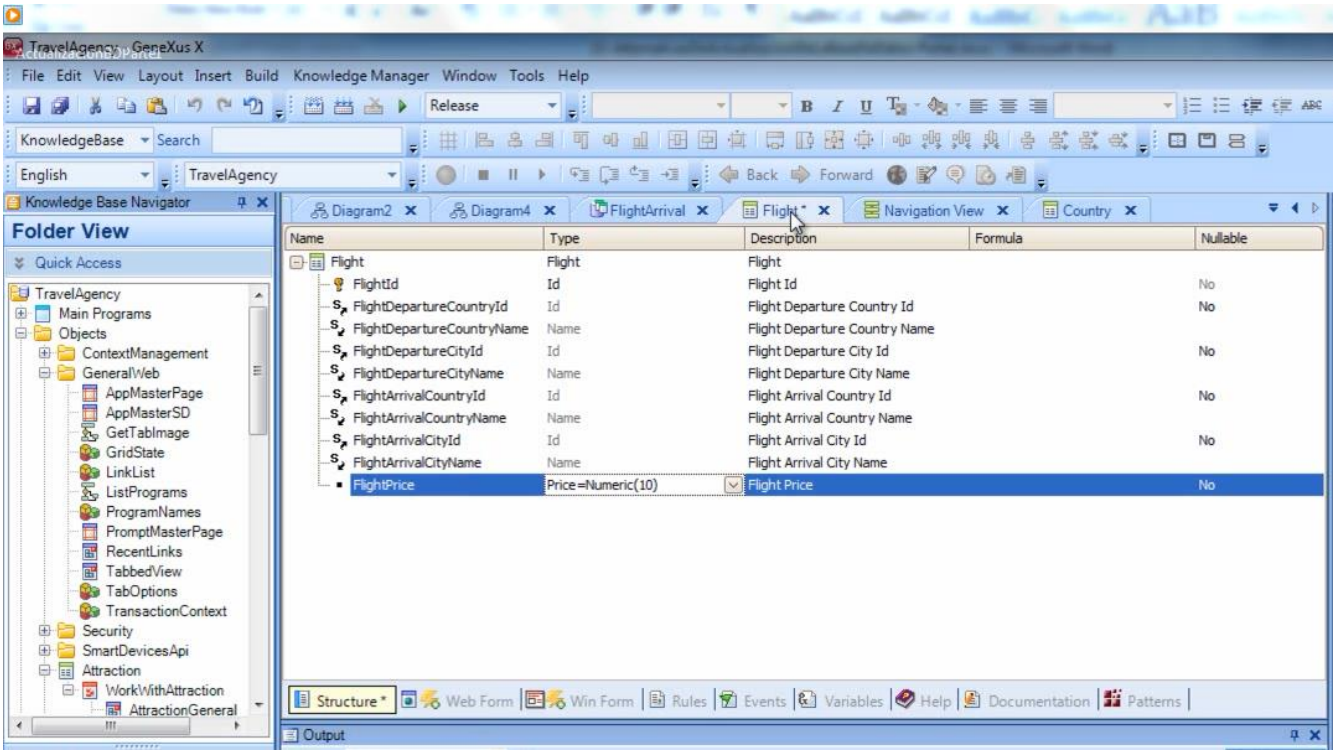
por isso nos interessa tanto que esse processo seja feito em massa.

Para resolver essa questão, o primeiro que faremos é acrescentar à transação Flight um atributo que armazene o preço de cada voo.

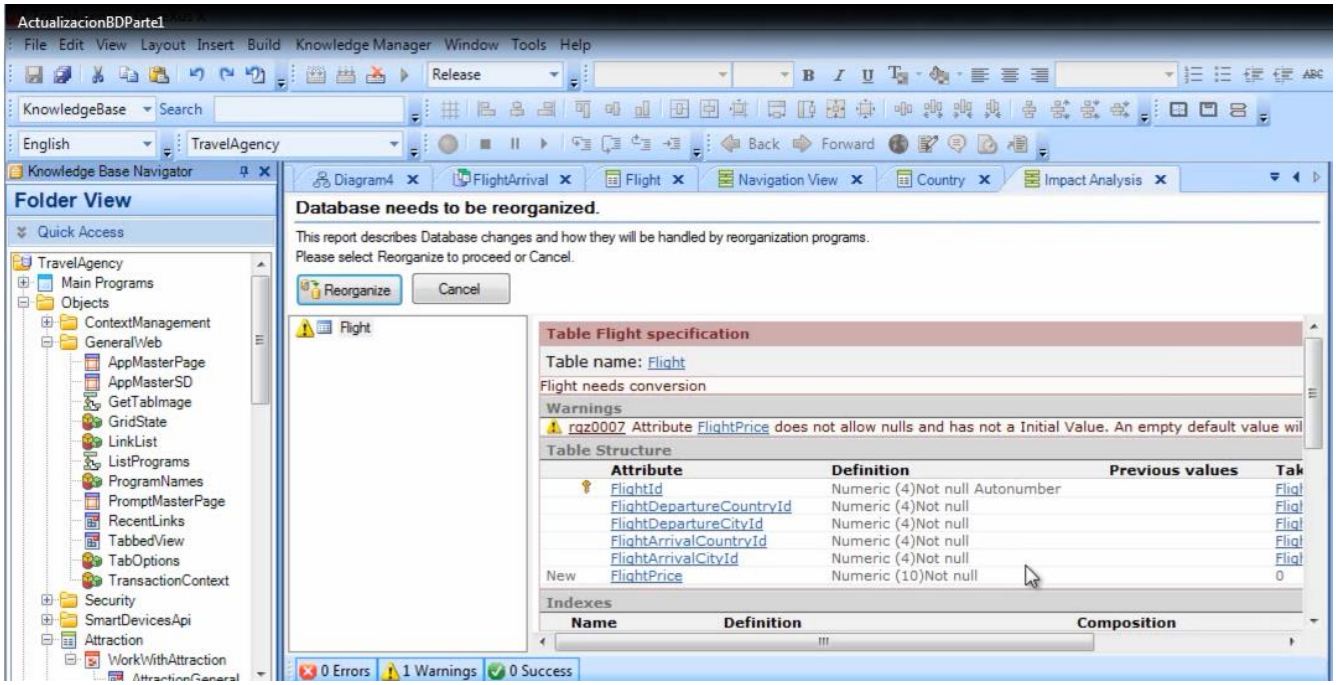
Criamos o atributo FlightPrice,



definimos um domínio chamado Price, como numérico igual a 10,

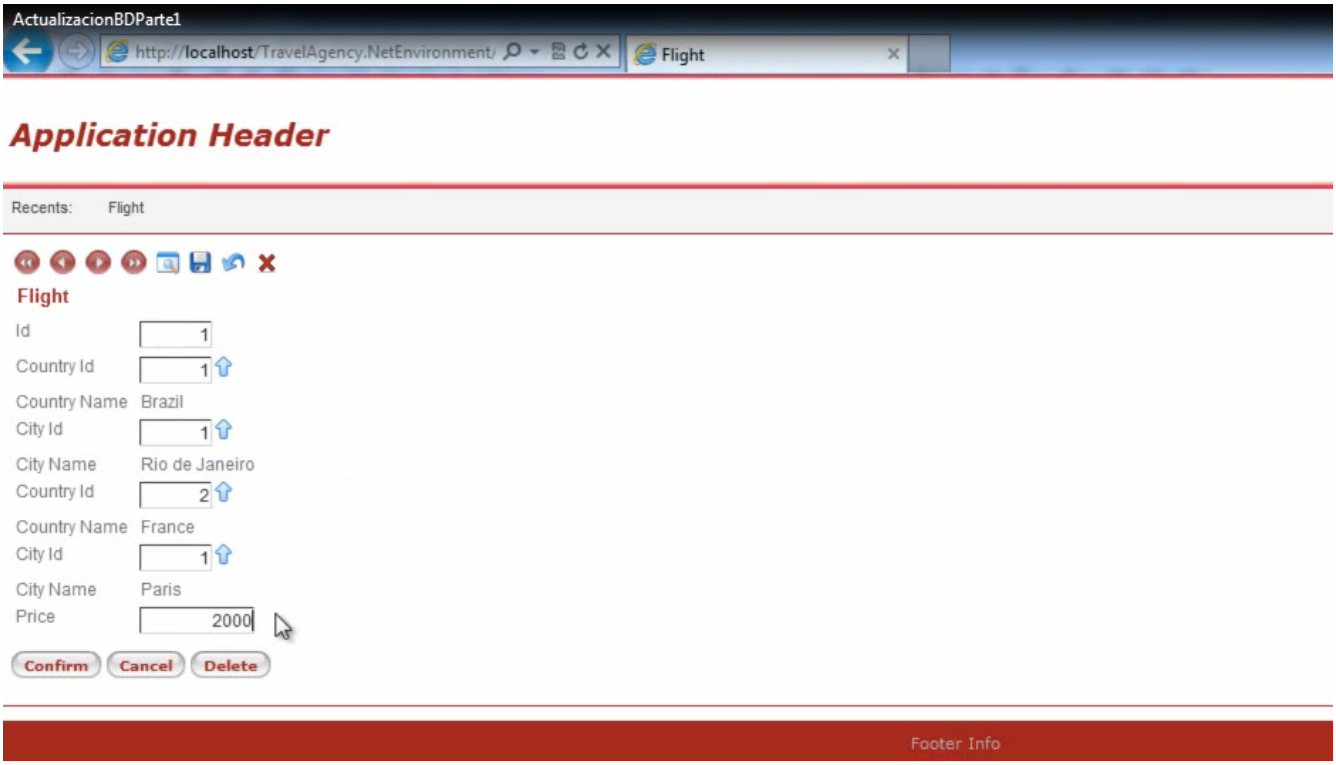


salvamos e pressionamos F5.
GeneXus avisa-nos que acrescentará esse campo à tabela FLIGHT,

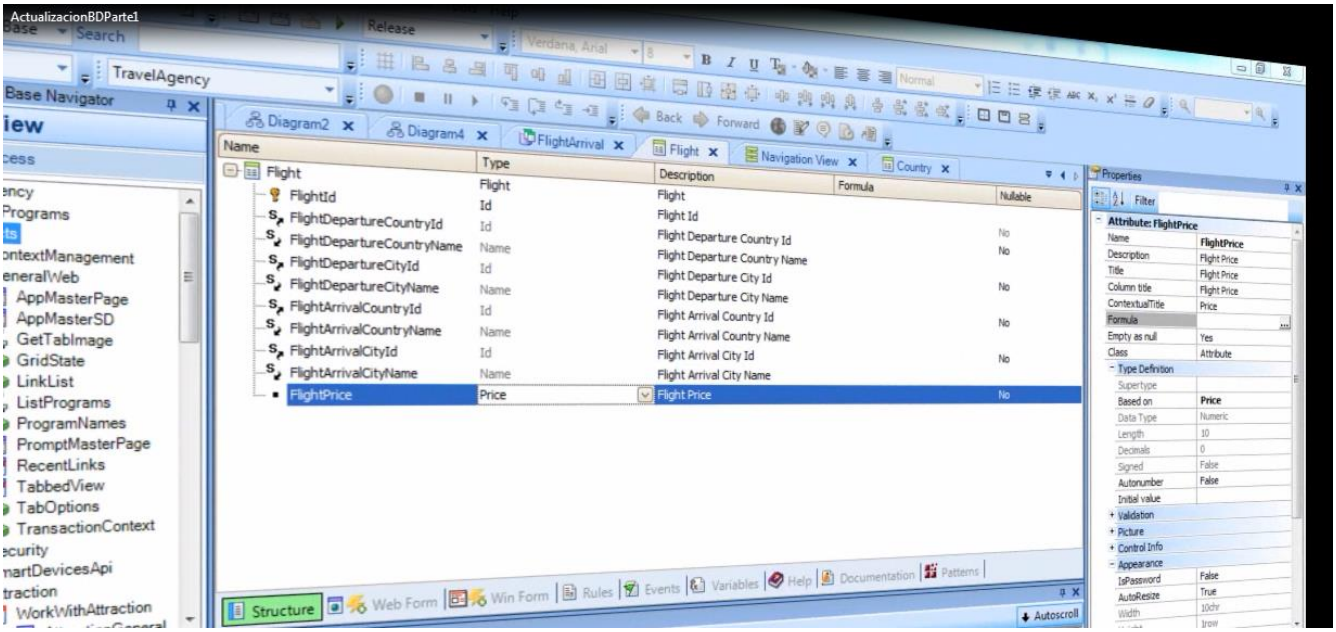


procedemos e seguimos com a geração dos programas.

Vamos inserir alguns preços para os voos que já tínhamos registrados,

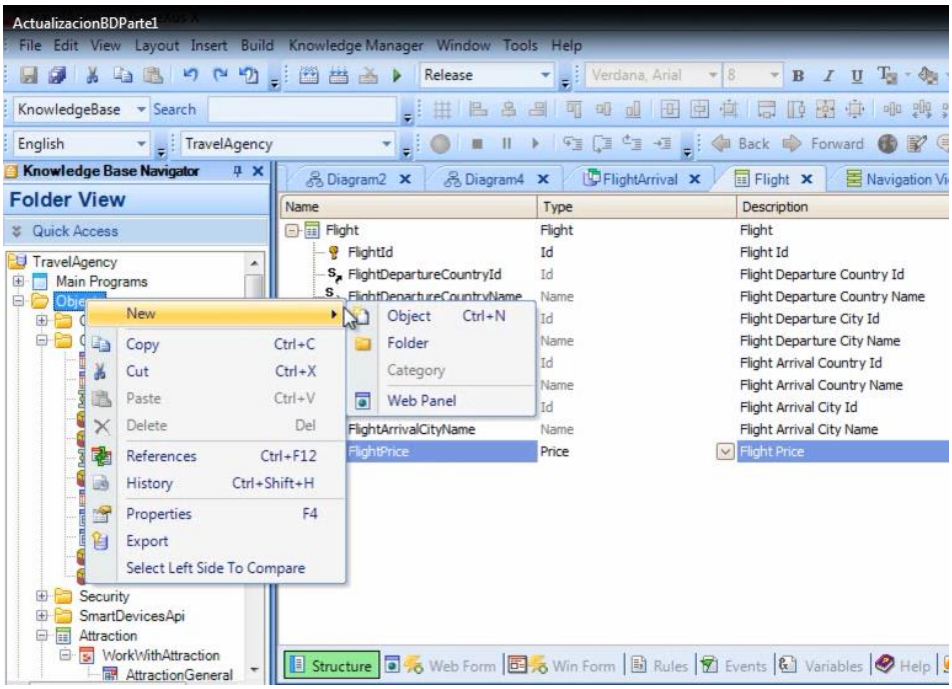


e então veremos como resolver o requisito de aumentar os preços de forma automática e massiva.



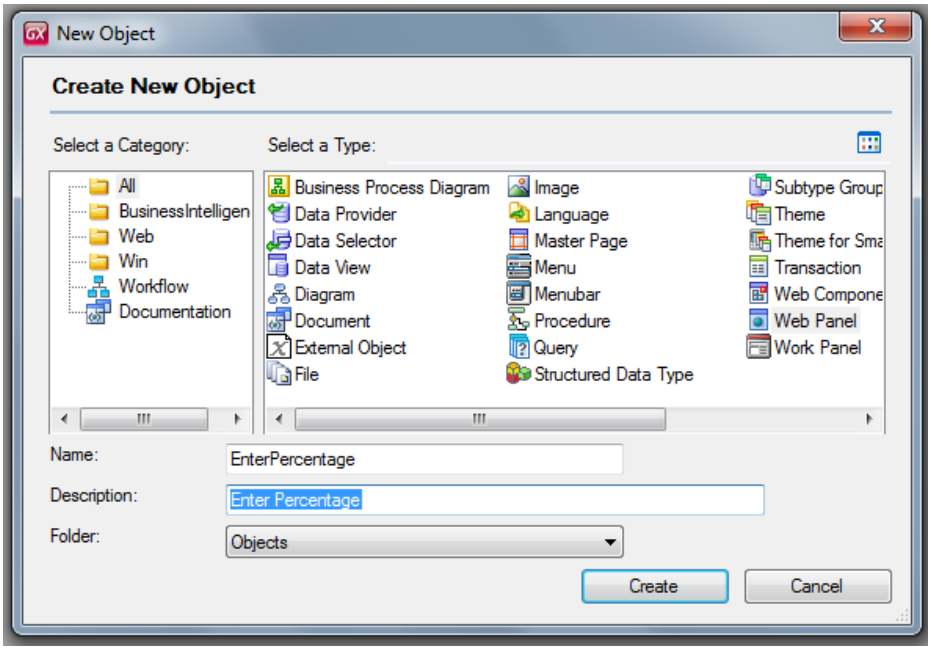
A primeira coisa a se fazer é solicitar aos usuários a porcentagem de aumento que se deseja aplicar aos preços dos voos. Para isso, criaremos um objeto de tipo web panel.

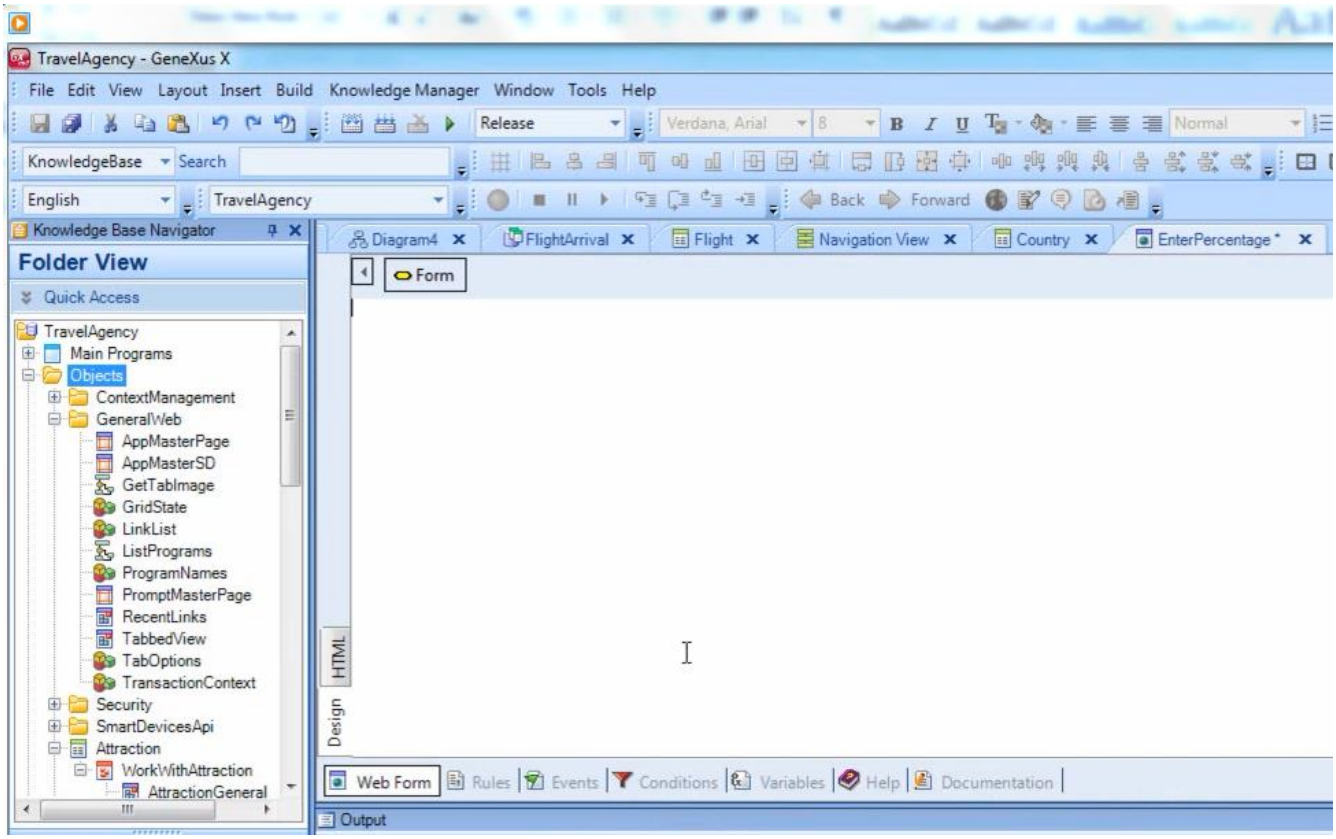
Selecionamos New/ Object



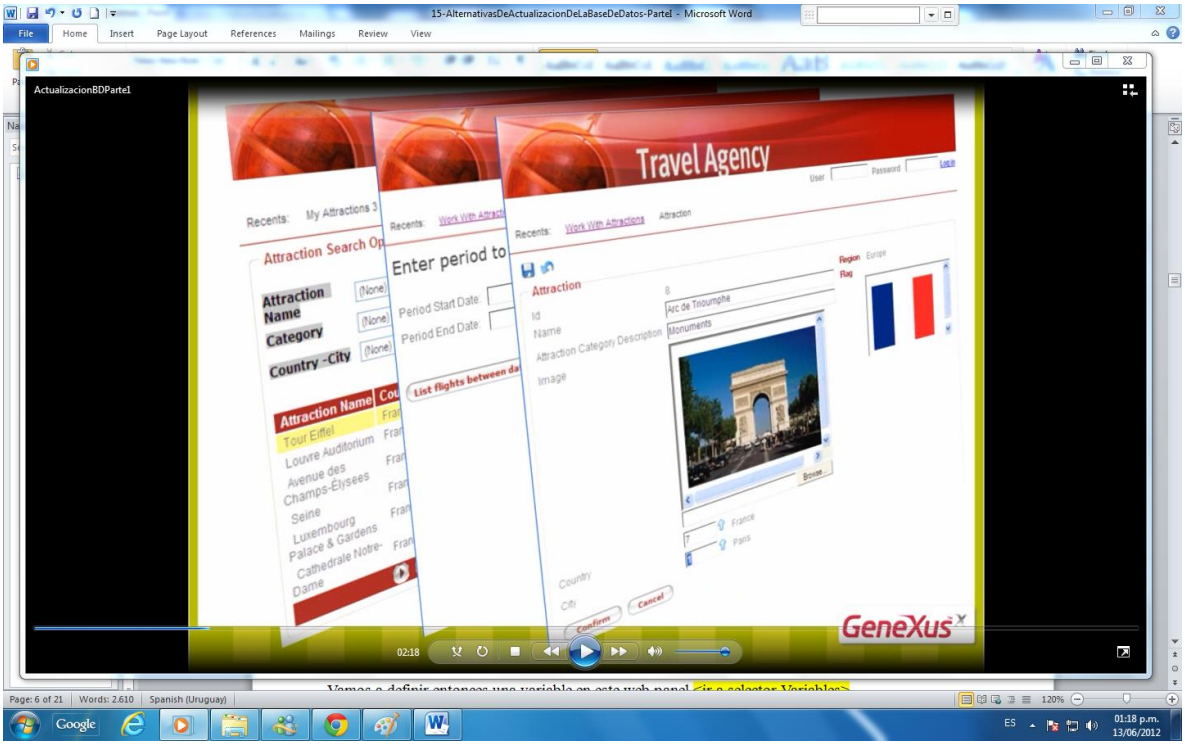
web panel

e o nomeamos EnterPercentage

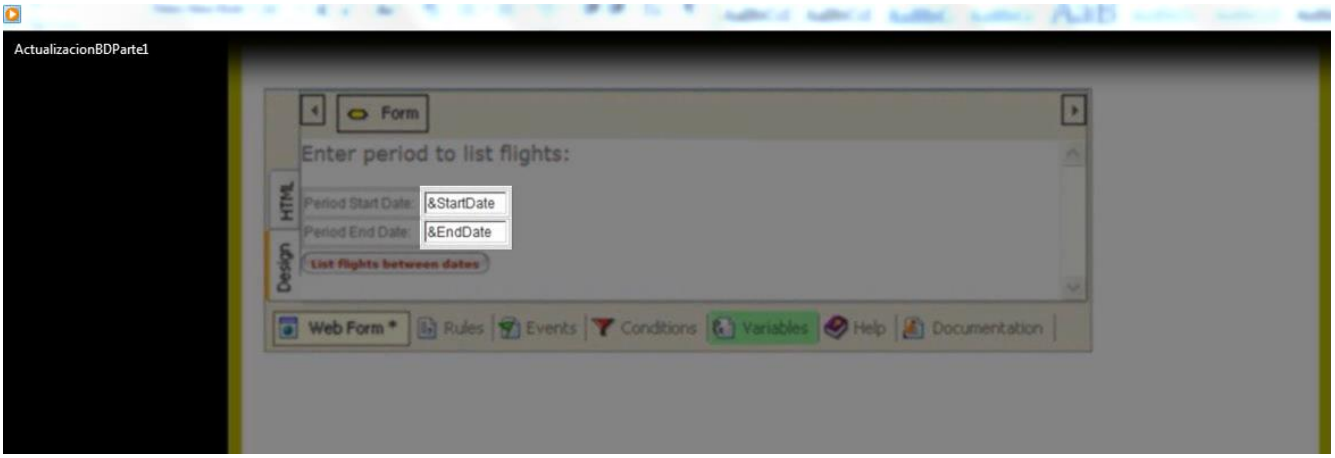




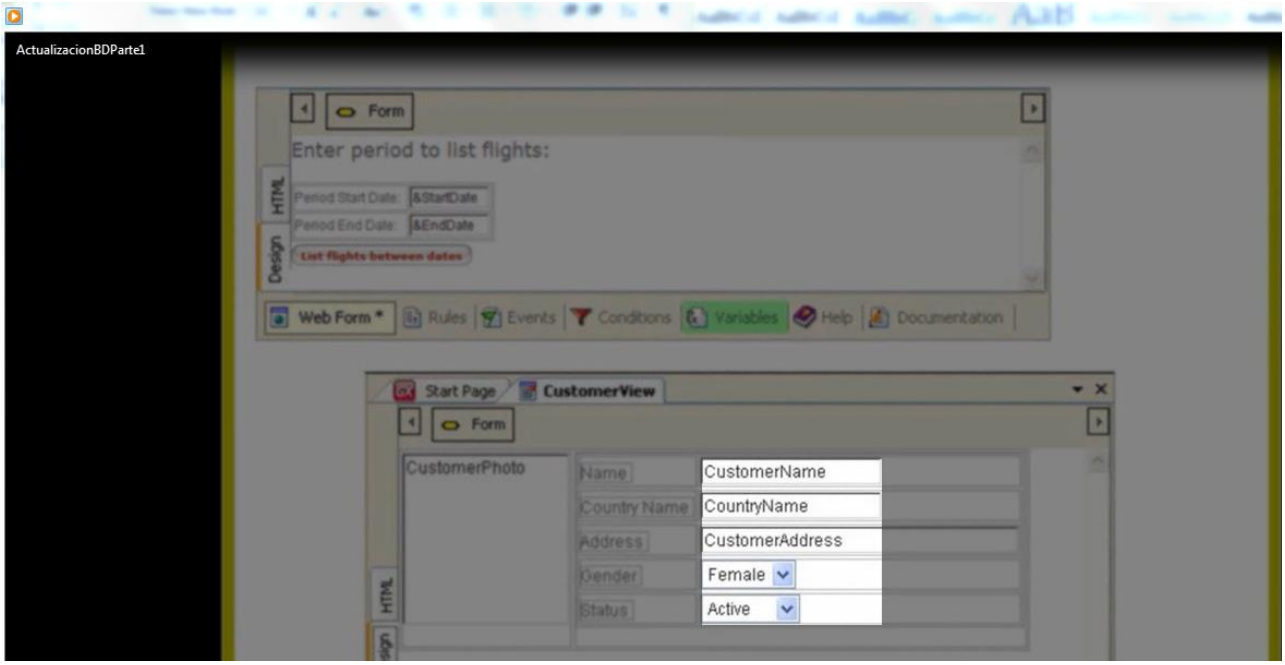
O objeto web panel é, basicamente, uma página web que nos permite resolver diversas funcionalidades, como solicitar dados ao usuário ou oferecer consultas funcionalmente completas.



Se incluirmos variáveis no form de uma web panel, elas terão um comportamento “de entrada de dados”, ou seja, o usuário poderá acrescentar valores.



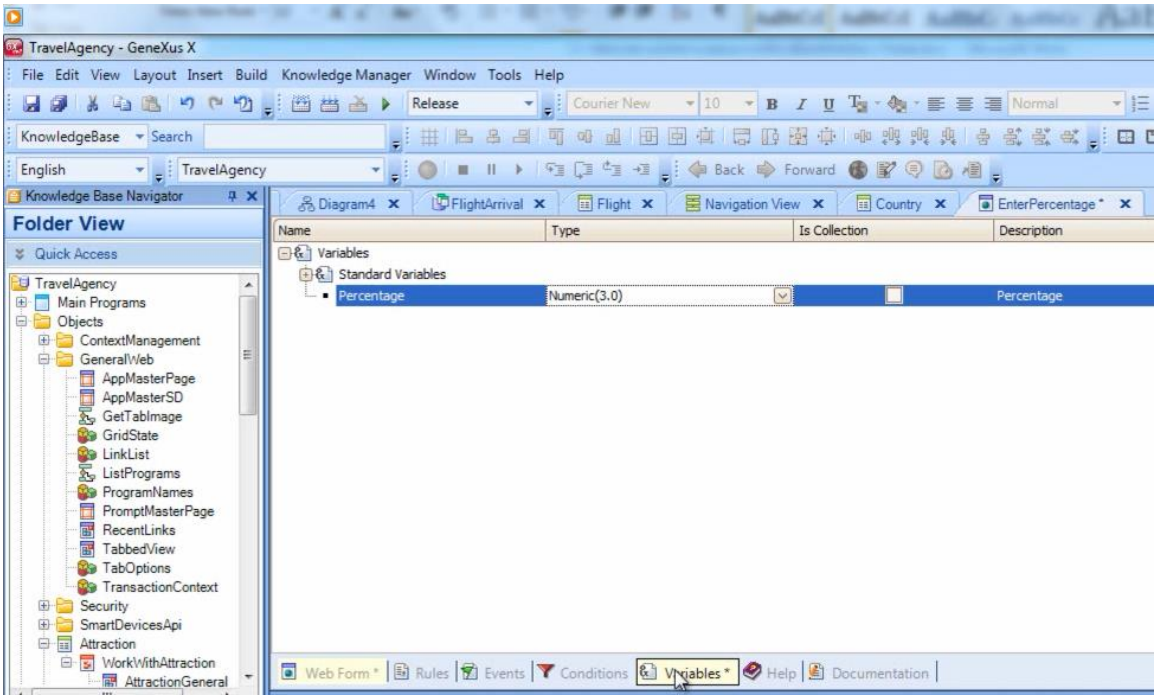
Se, em vez disso, incluímos atributos no form de uma web panel,



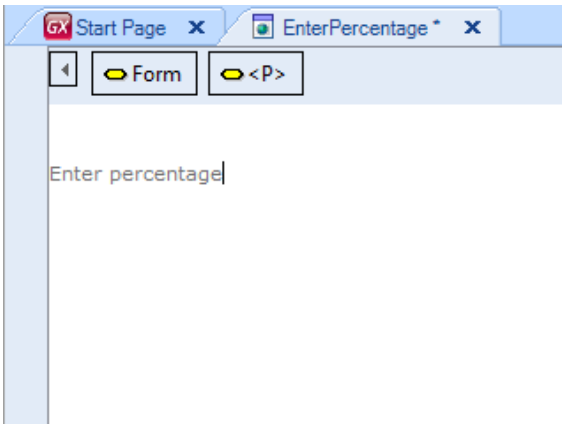
eles serão “de saída”, o que significa que não serão editáveis, apenas mostrarão os dados que estes atributos têm armazenados no banco de dados.

Vamos, então, definir uma variável nessa web panel.

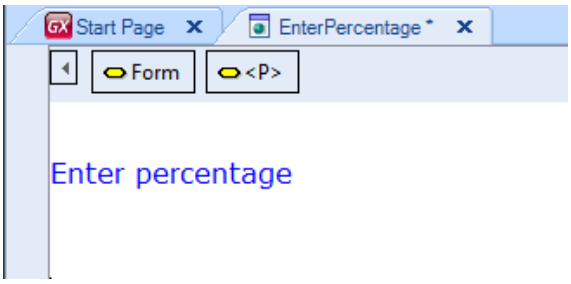
A variável será criada com o nome Percentage e a definimos do tipo Numeric(3)



Agora, voltamos ao form e digitamos diretamente dentro dele: “Enter percentage”.

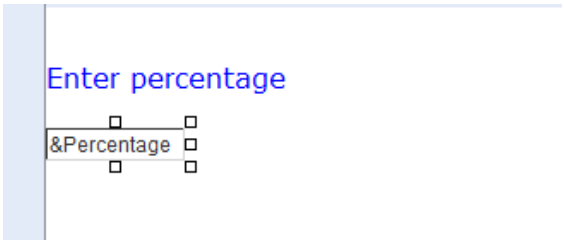


É possível mudar o formato desse texto

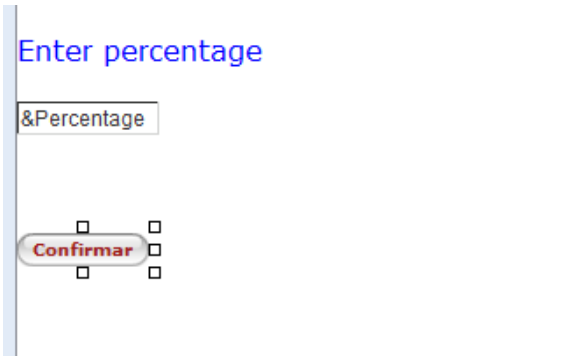


Pressionamos enter e embaixo do texto, inserimos a variável &Percentage

Selecionamos Insert/ Variable e escolhemos a variável &Percentage:

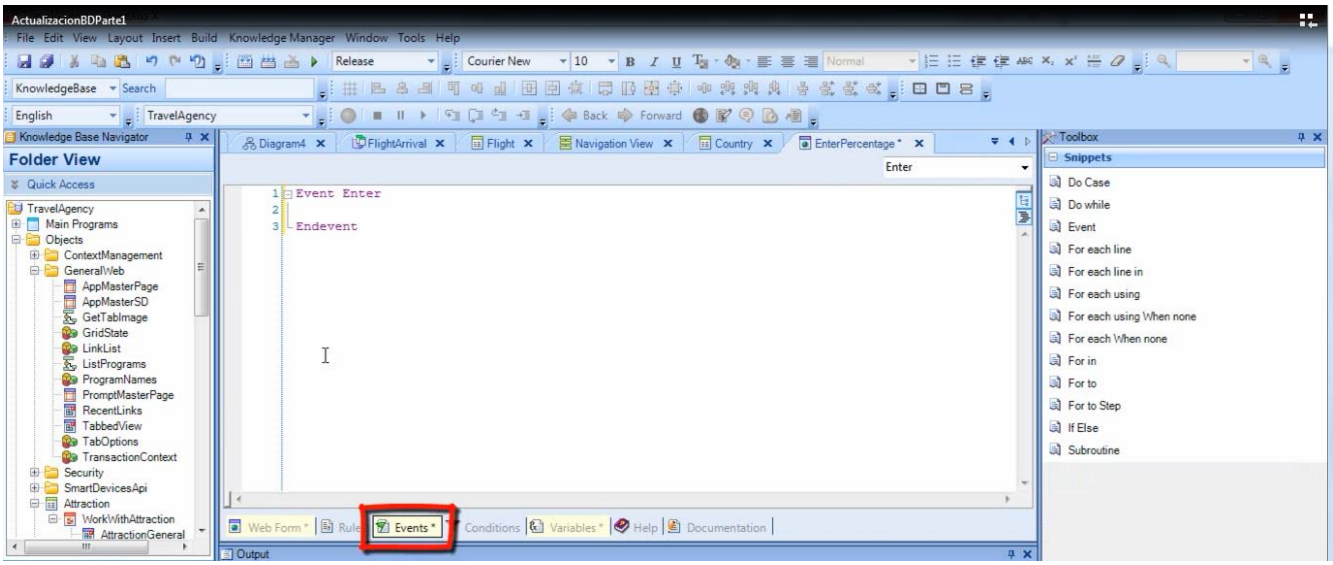


Agora, arrastamos um botão ao form a partir da toolboox [caixa de ferramentas]



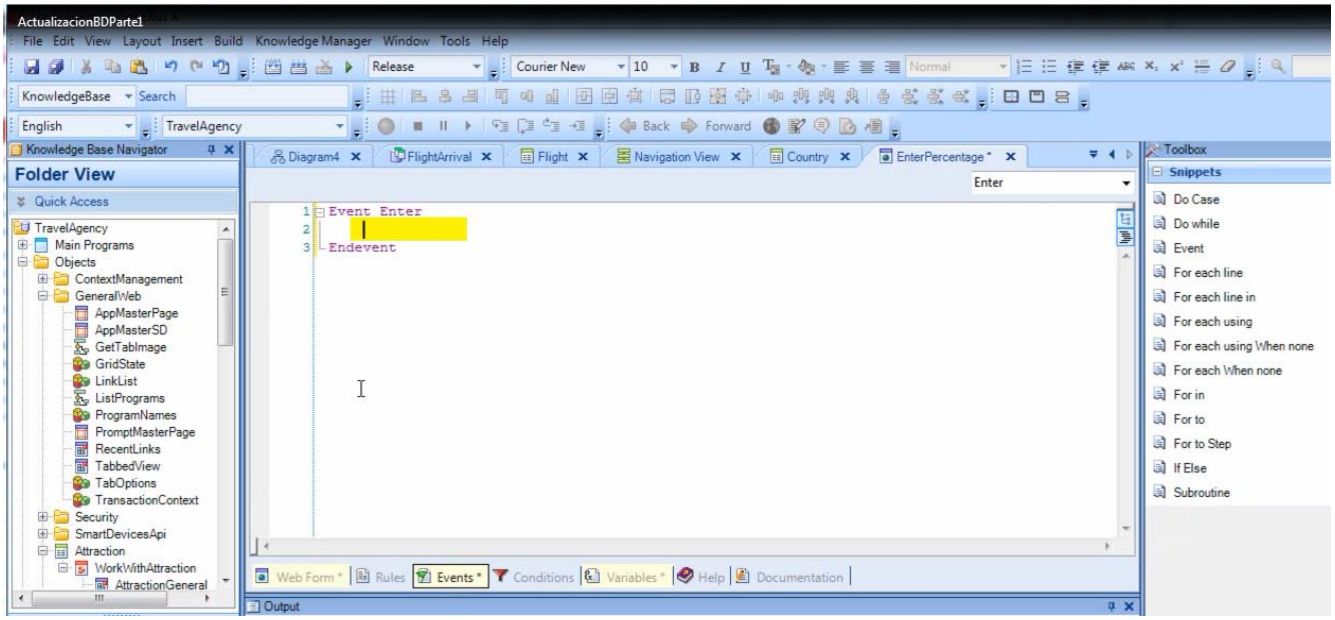
Pode-se perceber que a página ficou pronta para que o usuário insira a porcentagem de aumento que deseja e pressione o botão para executar um processo automático que resolve o aumento dos preços de todos os voos.

Se clicarmos duas vezes sobre o botão, somos levados à seção de eventos da web panel,



e, particularmente, o cursor posiciona-se dentro do evento associado ao botão: o evento Enter.

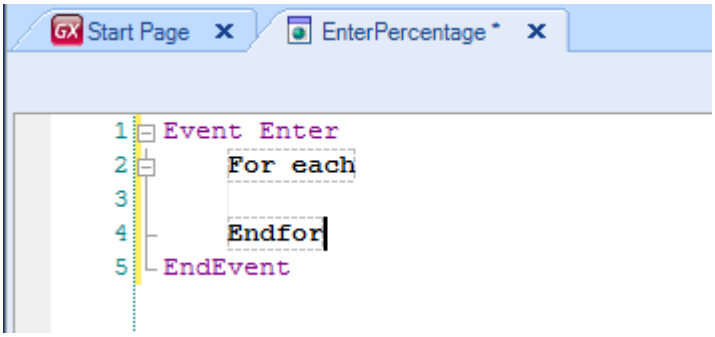
Aqui,



codificaremos as instruções que queremos executar quando o usuário clicar no botão.

Lembremos o que é preciso fazer: devemos navegar por todos os voos armazenados e, para cada um, atualizar o seu preço.

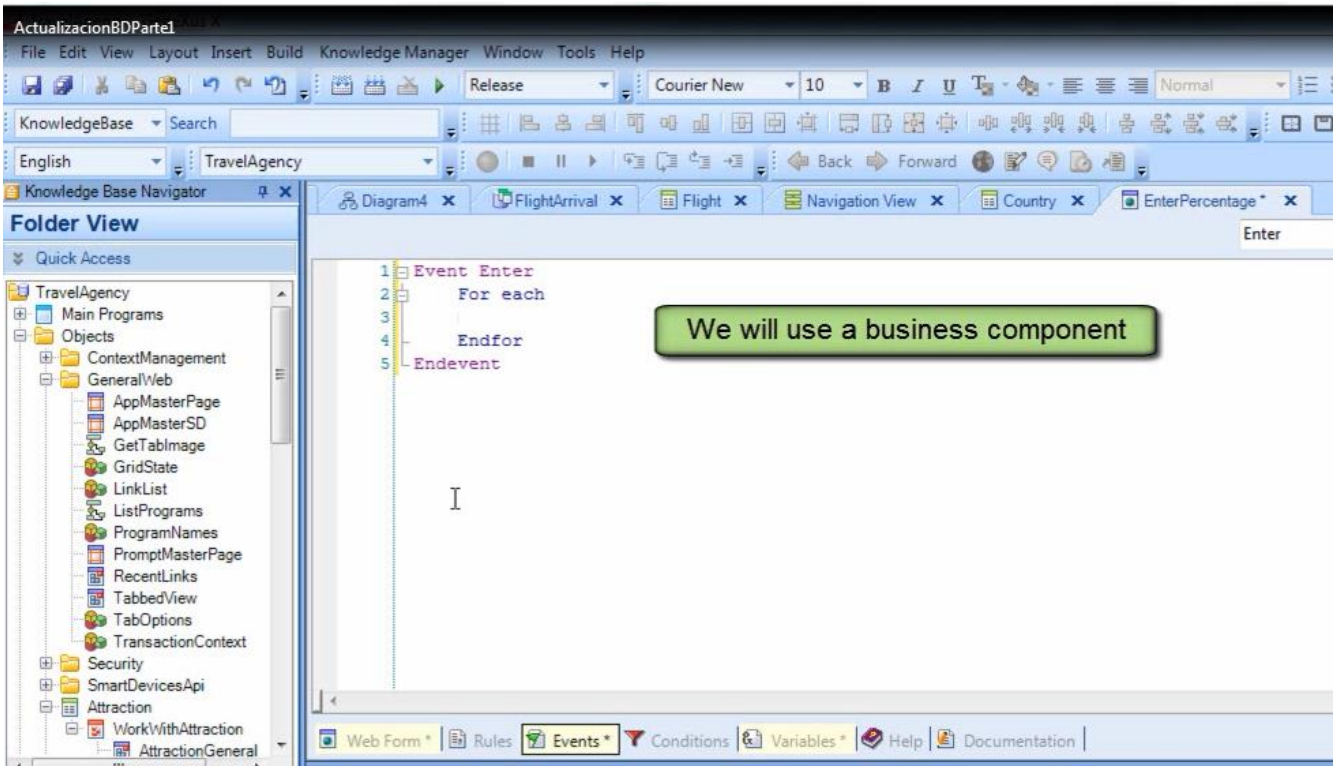
Escrevemos, então, um comando For Each, com seu correspondente Endfor, para navegar na tabela FLIGHT.



Um For each sempre tem que conter pelo menos um atributo em seu corpo para que GeneXus possa determinar a tabela base a ser navegada pelo For each.

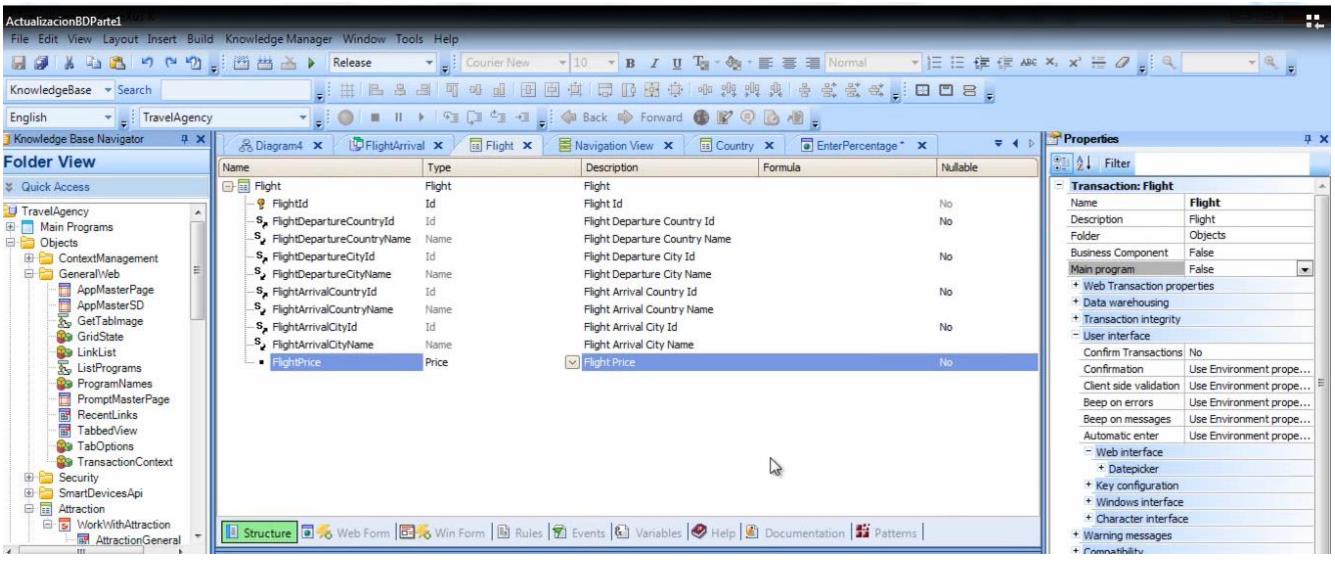
Que atributo podemos referenciar? E como fazemos para atualizar o preço de cada voo?

Para atualizar o banco de dados em uma web panel, temos apenas uma possibilidade... Precisamos empregar o conceito de business component.

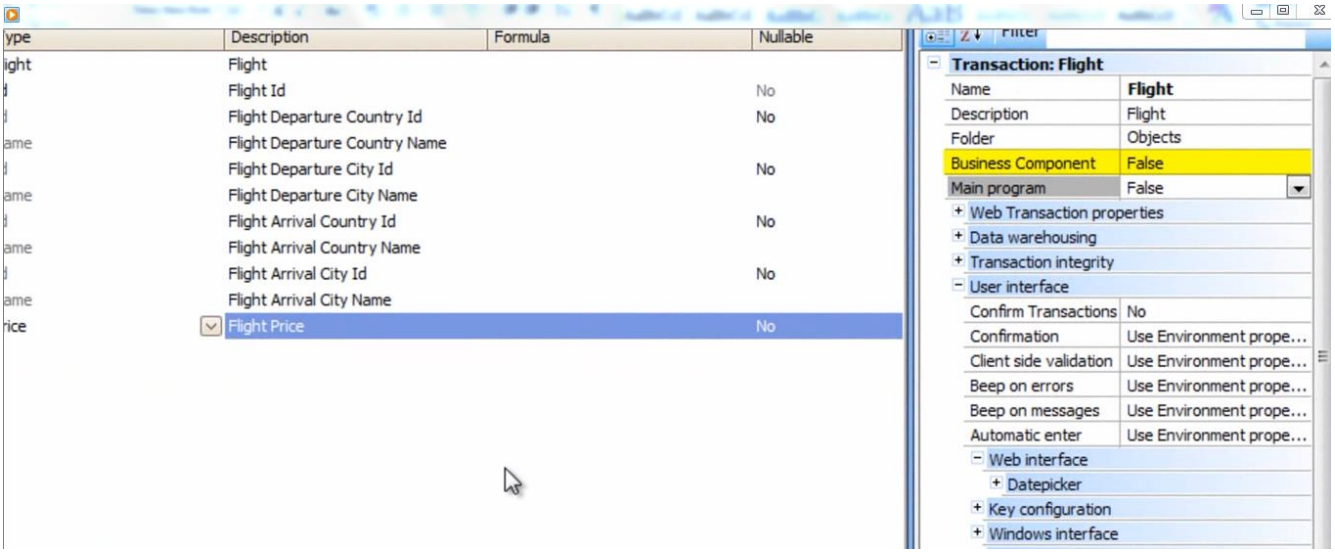


Vejamos o que é business component e como se usa.

Como queremos atualizar os dados de voos, vamos abrir a transação Flight e observar as propriedades da transação.



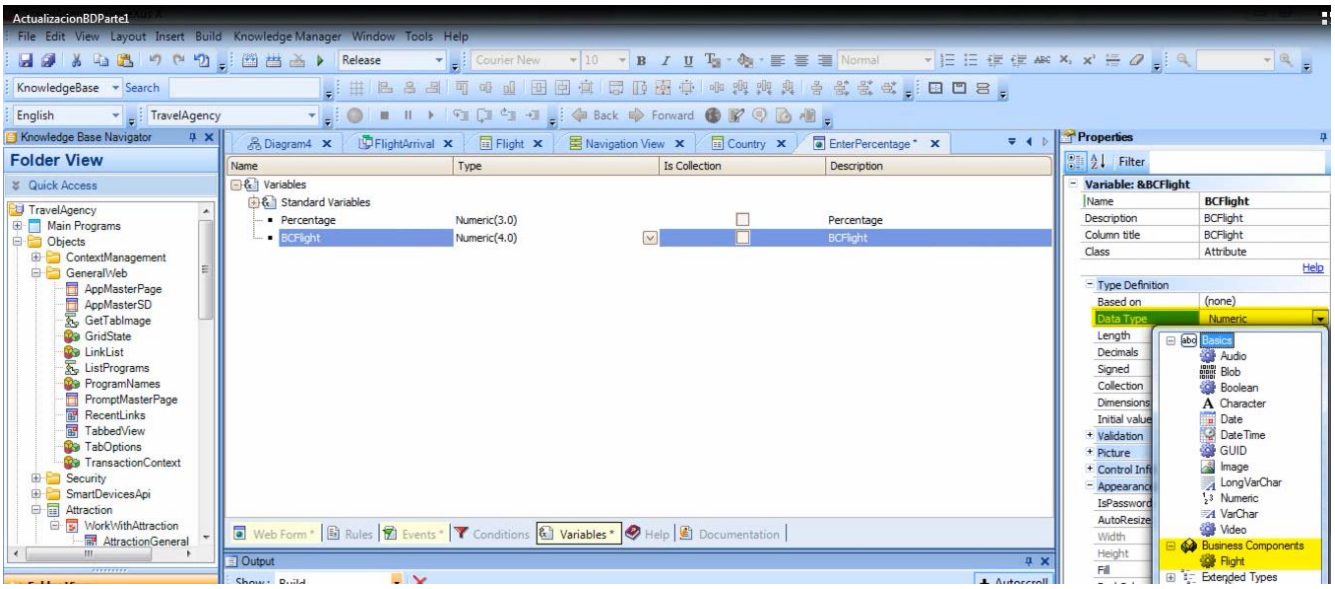
Podemos ver que há uma propriedade chamada Business Component,



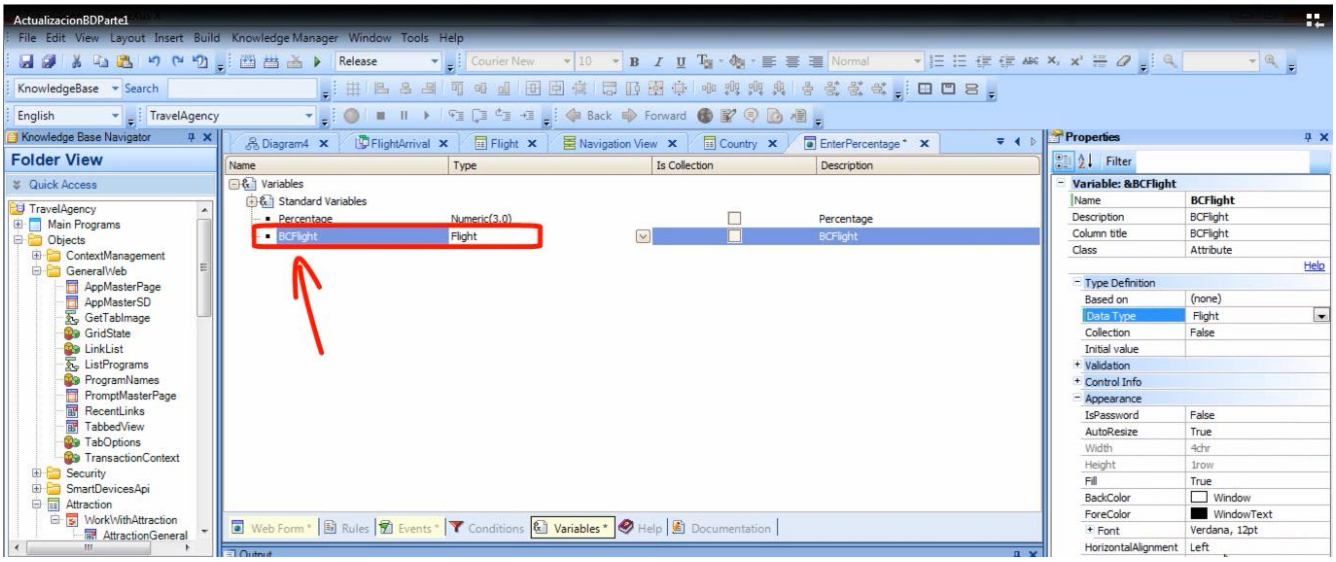
e a configuraremos com valor True:

Todas as transações têm a propriedade Business Component, ou seja, podemos configurar a propriedade Business Component de qualquer transação da base de conhecimento com valor True.

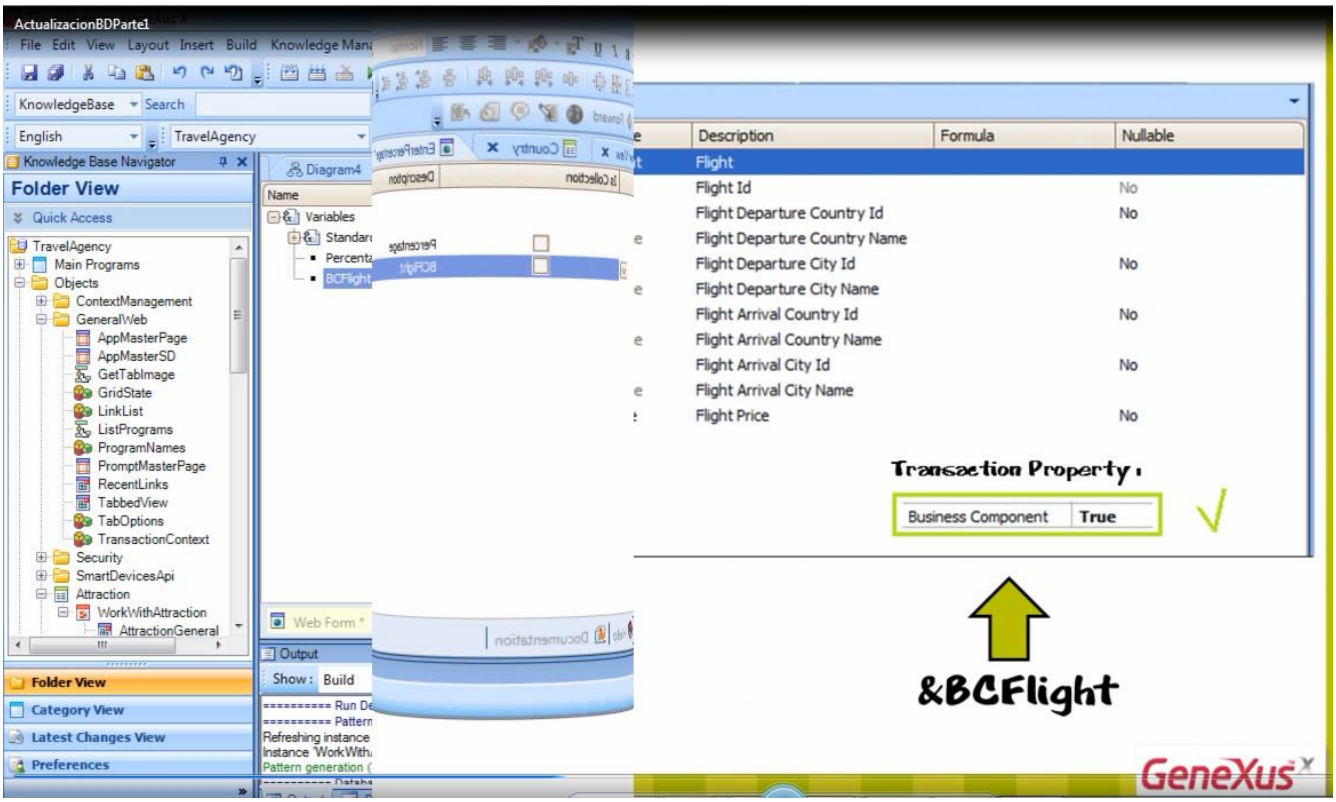
Quando configuramos a propriedade Business Component de uma transação com valor True, **em qualquer objeto da base de conhecimento podemos definir uma variável do tipo business component da transação definida como tal...**



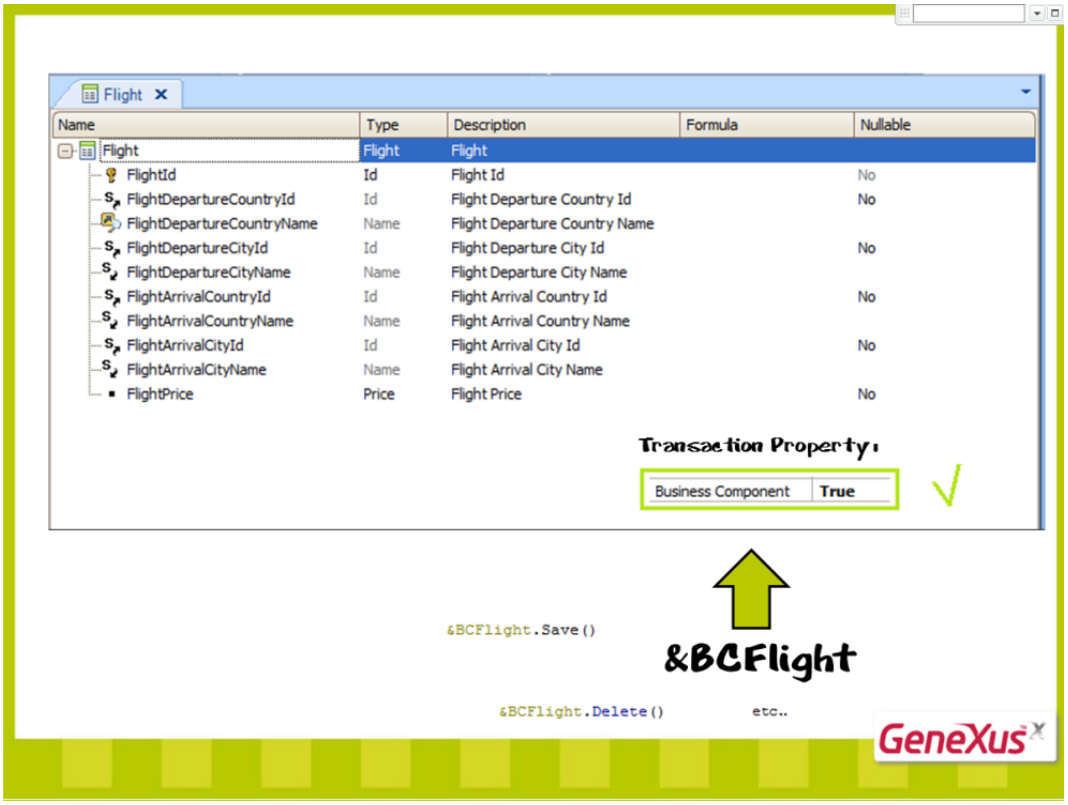
Empregando esta variável,



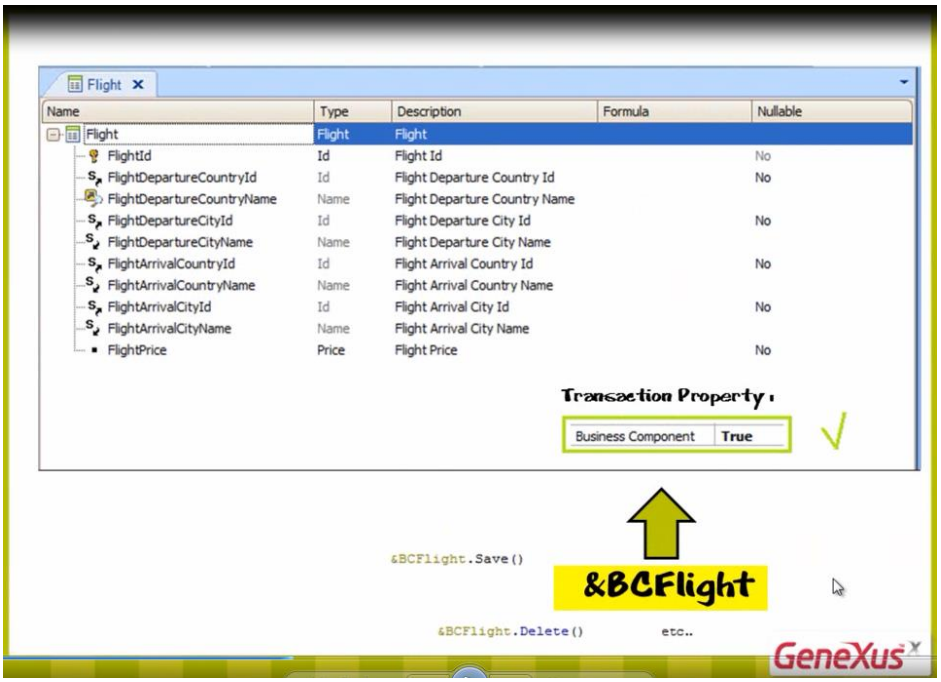
podemos executar a transação Flight <u>sem seu form</u>



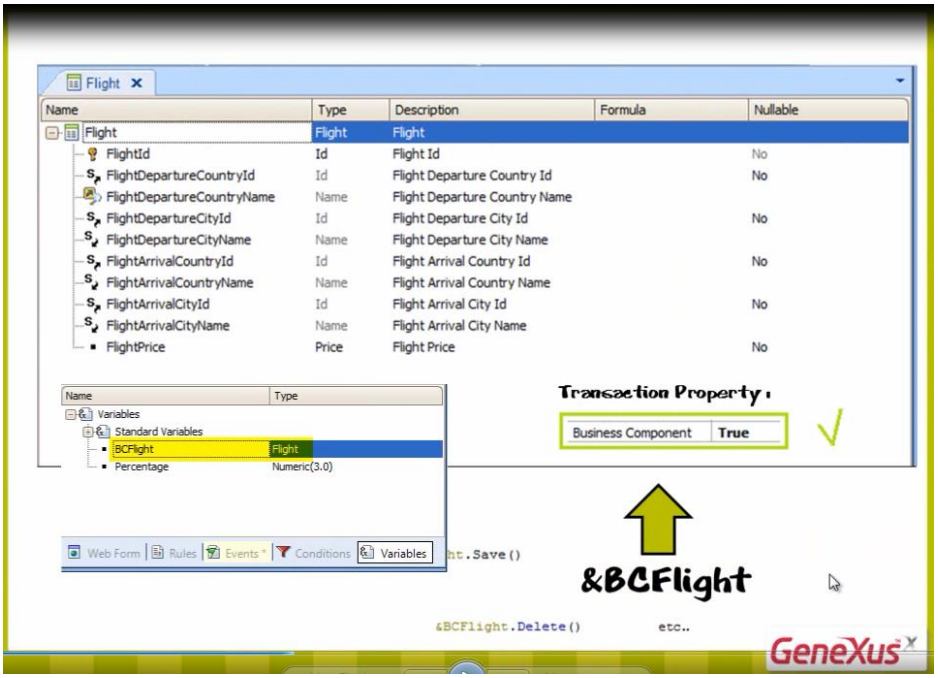
para realizar atualizações do banco de dados:



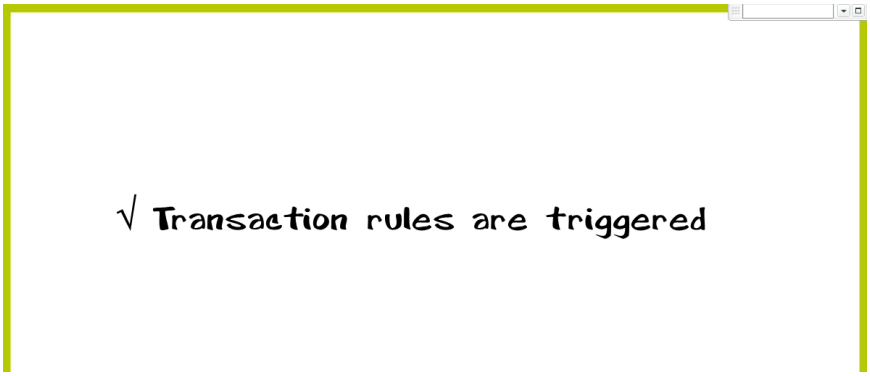
Além disso, quando utilizamos essa variável



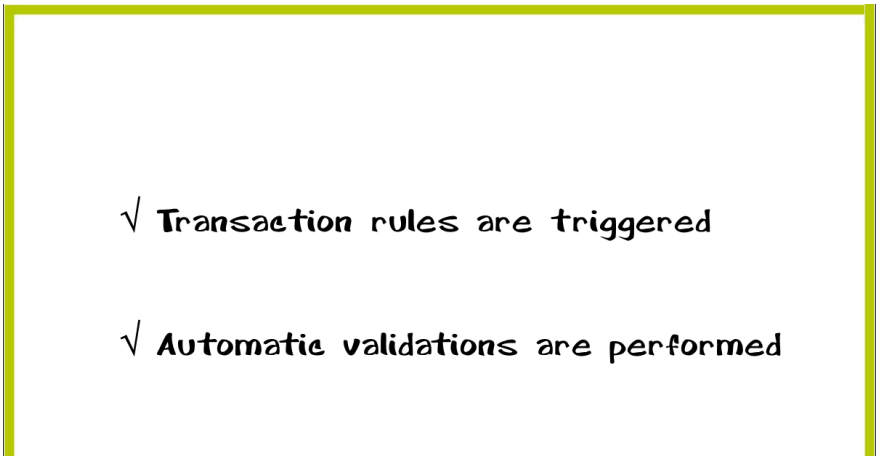
definida do tipo business component de uma transação



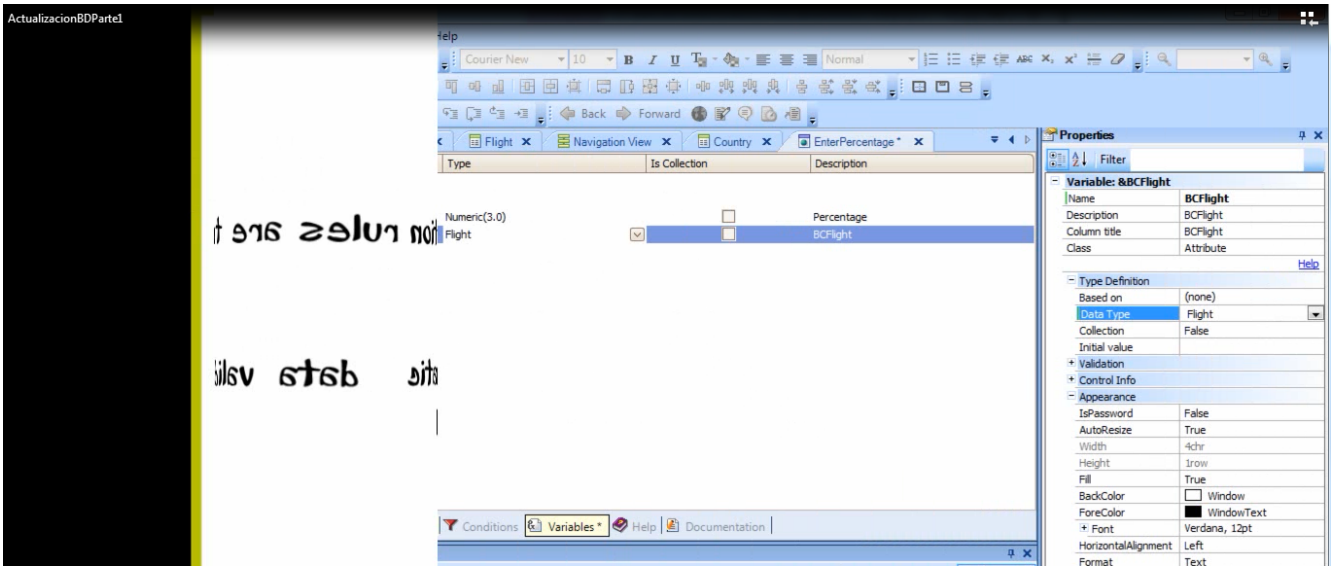
são disparadas as regras que tínhamos definido na transação



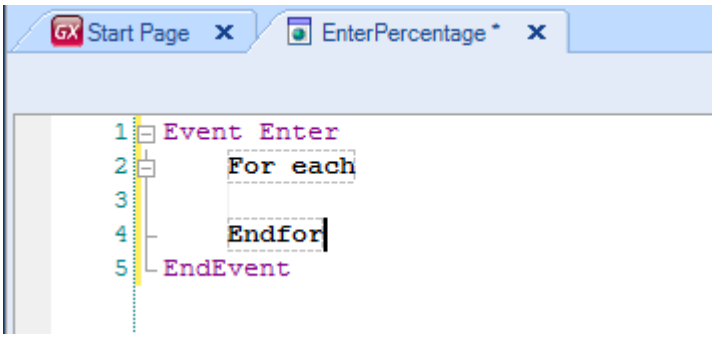
e também **são executados todos os controles automáticos que oferecem a transposição para validar que os dados armazenados sejam consistentes.**



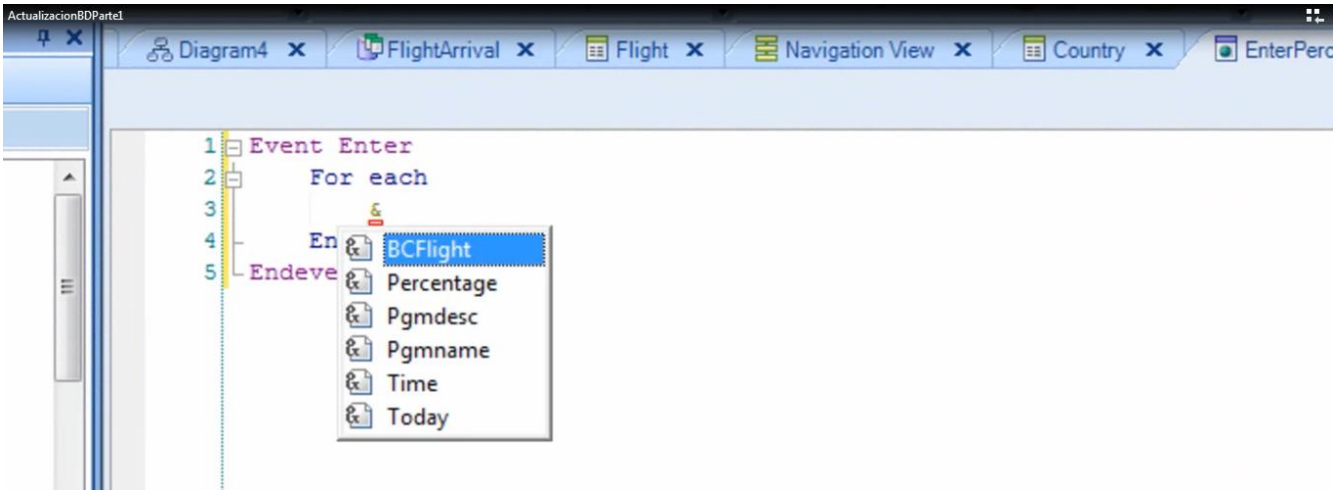
Agora, voltemos ao objeto que estávamos codificando para aprender a utilizar variável do tipo business components.



Na web panel “EnterPercentage”, dentro do corpo do For Each que estávamos codificando, vamos escrever a primeira instrução.

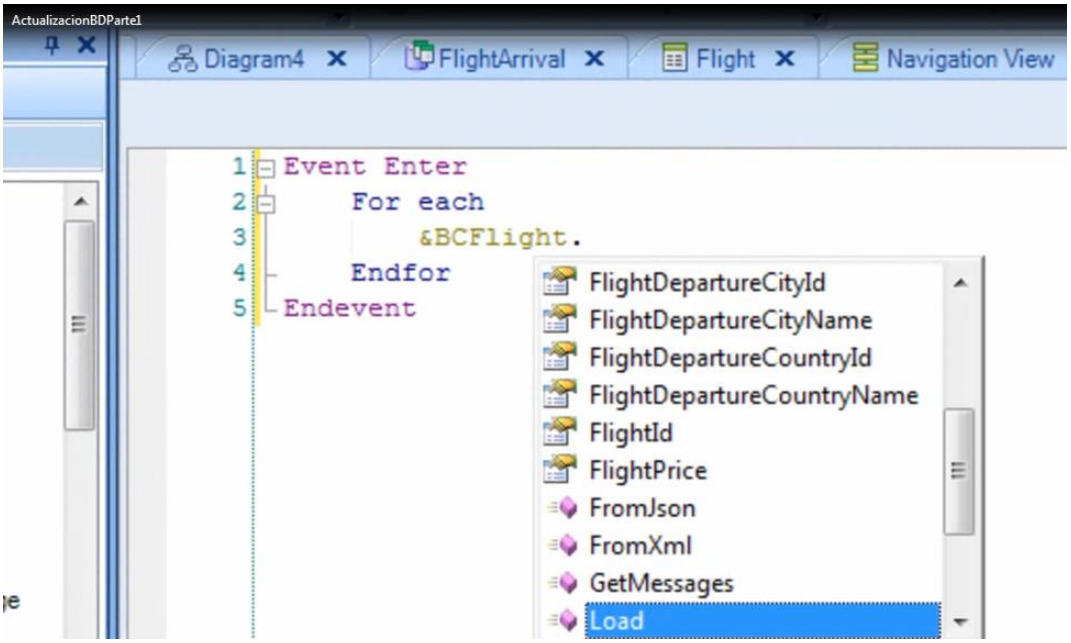


Escrevemos ampersand (&),

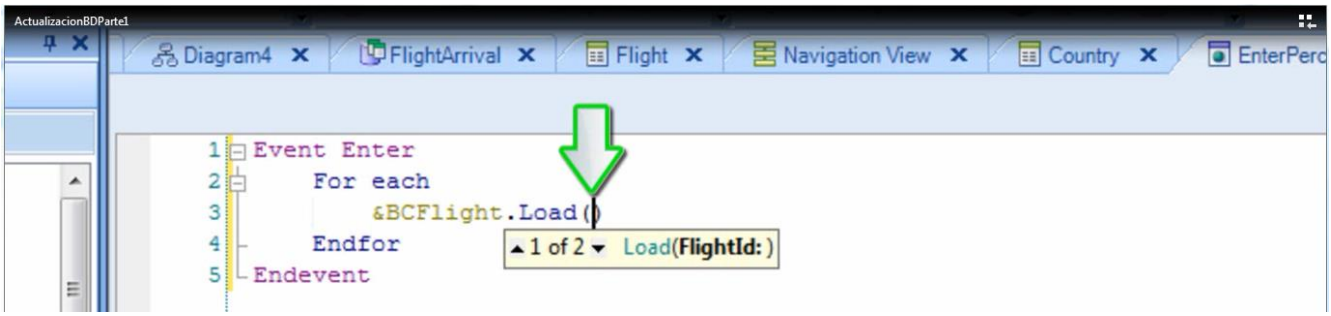


e selecionamos a variável &BCFlight que já definimos.

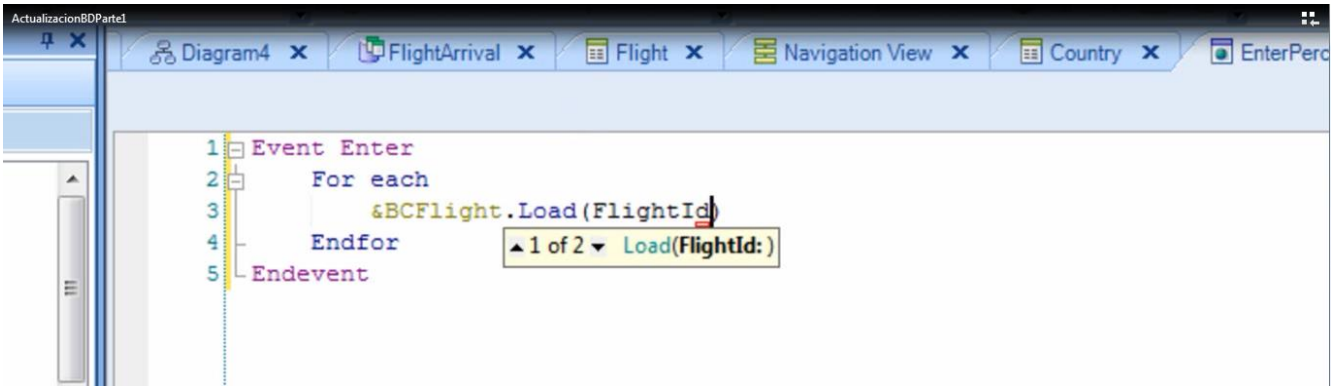
Agora, acrescentamos um ponto e escolhemos o método Load.



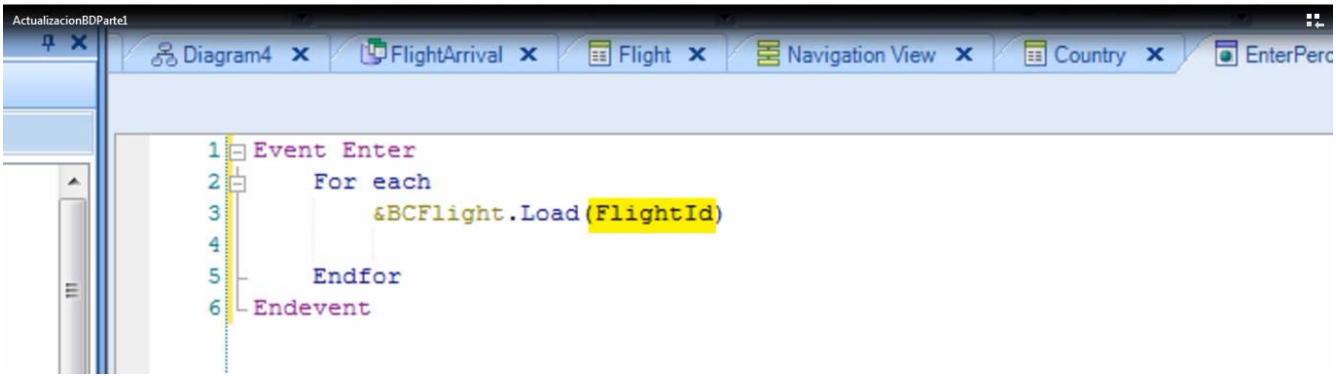
O método Load, como seu nome descreve, permite **carregar** na memória os dados correspondentes ao valor da chave primária que podemos indicar dentro dos parênteses.



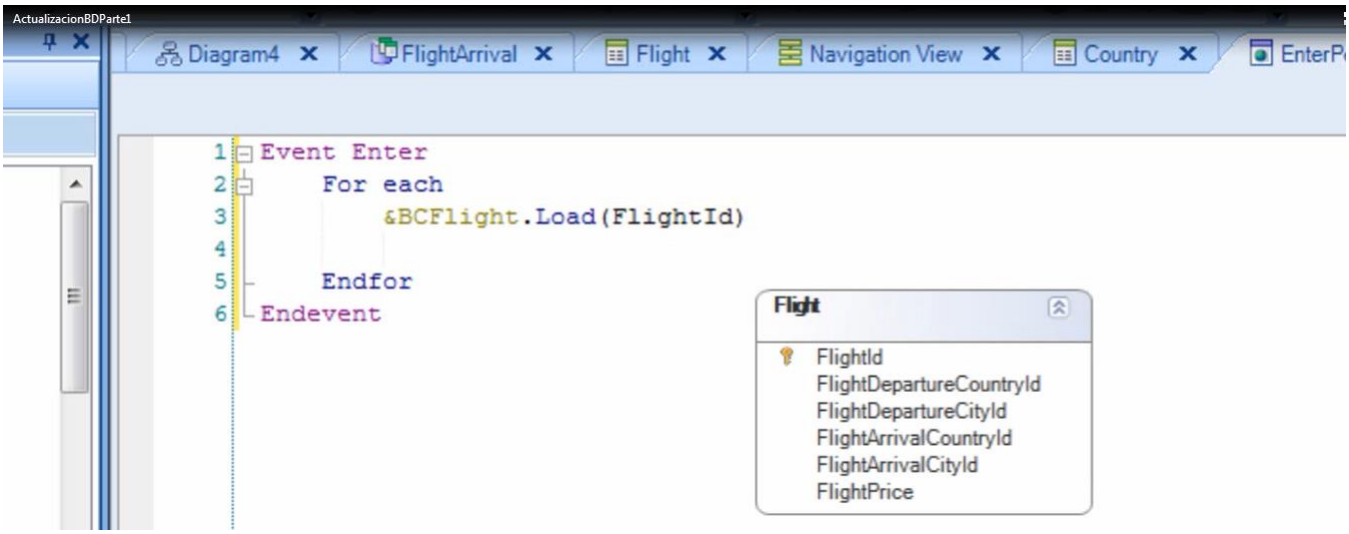
O valor da chave primária aí incluída deverá ser **um valor válido para chave primária da transação que é business component.**



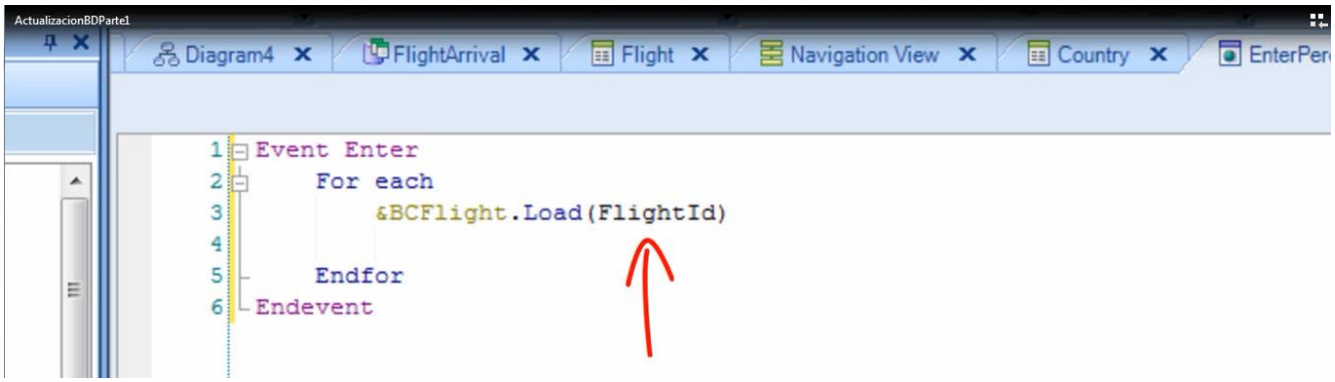
Neste caso, estamos carregando na memória um voo e temos o valor da chave primária no atributo FlightId,



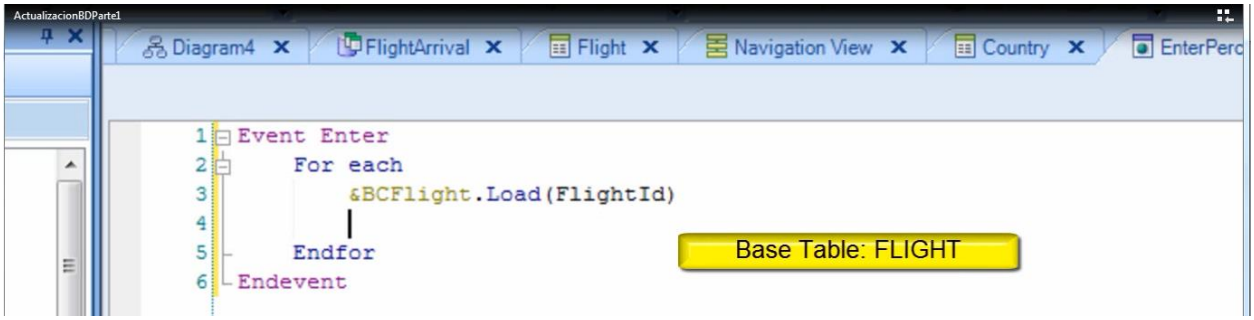
já que nosso objetivo é navegar pela tabela FLIGHT mediante o For each



e, para cada voo navegado, teremos os valores de seus atributos.
Como nosso objetivo é navegar pela tabela FLIGHT, observemos que este atributo FlighId

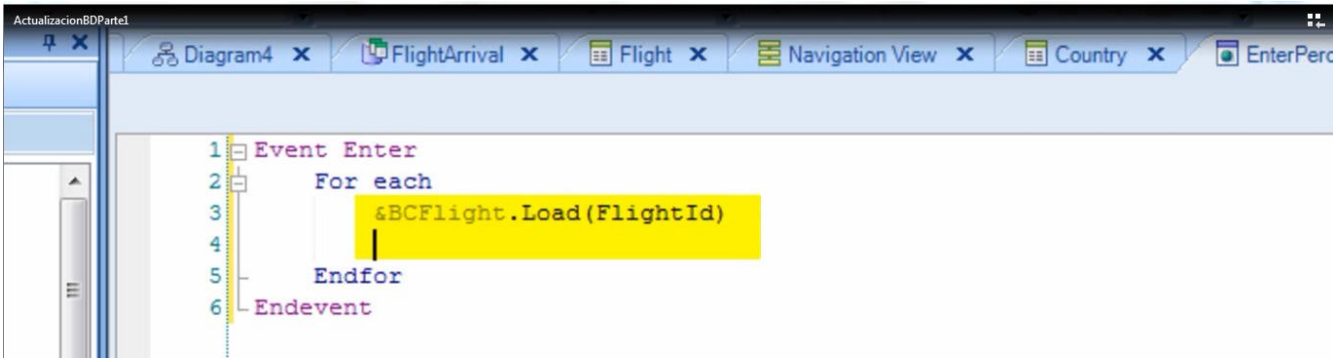


é, no momento, o único atributo presente no For Each e, portanto, faz com que a tabela base do For Each seja FLIGHT.

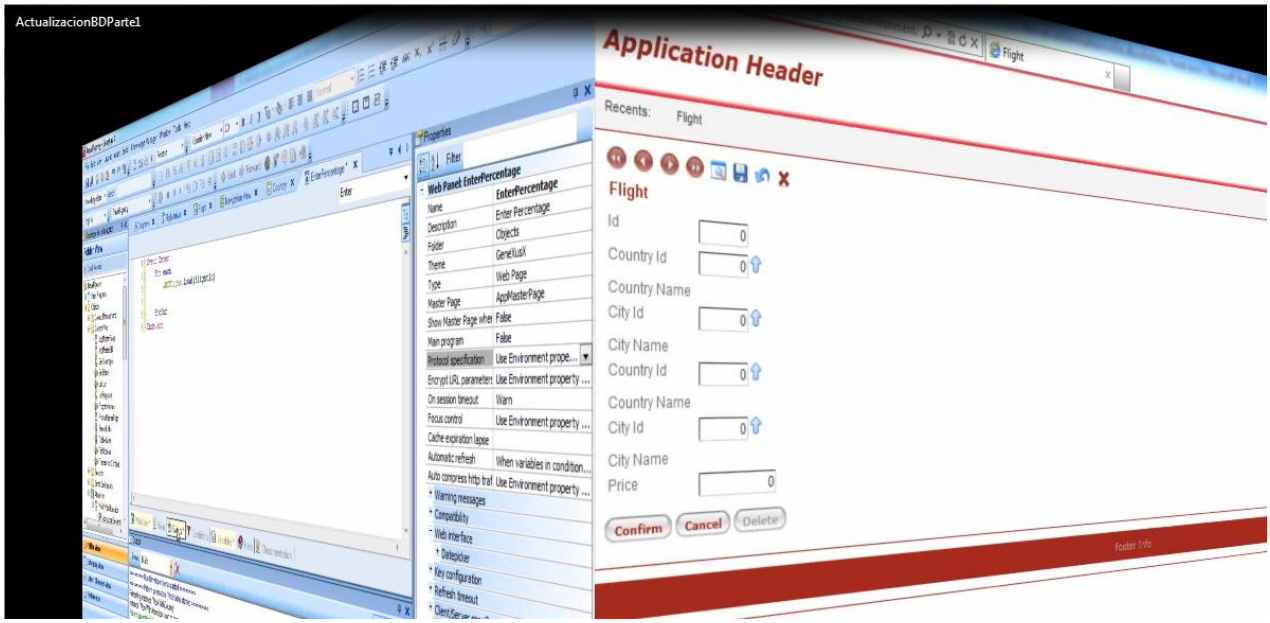


Assim, o For Each navegará **por toda a tabela FLIGHT**, uma vez que não definimos nenhum where com condições de filtro e, para cada registro navegado, serão executadas as instruções incluídas no corpo do For

Each.

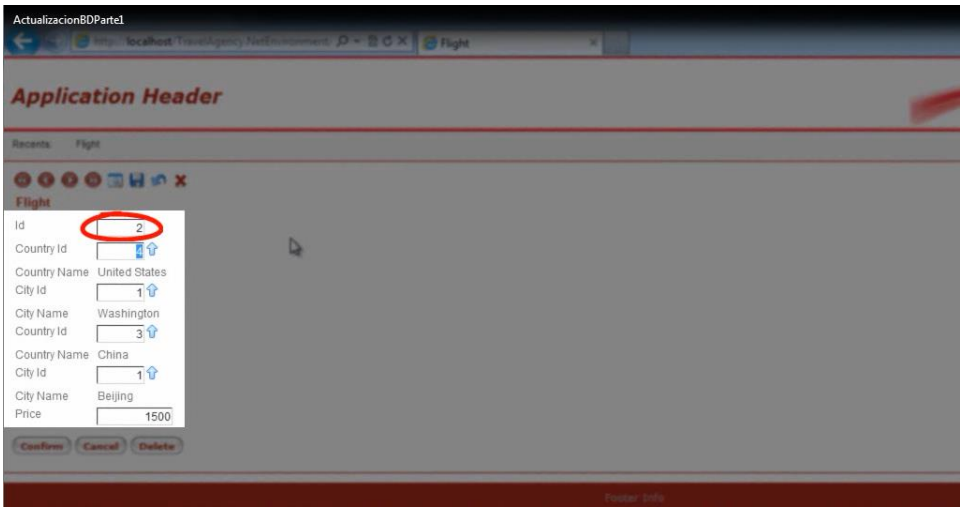


Então, a primeira sentença efetua algo equivalente ao que ocorre



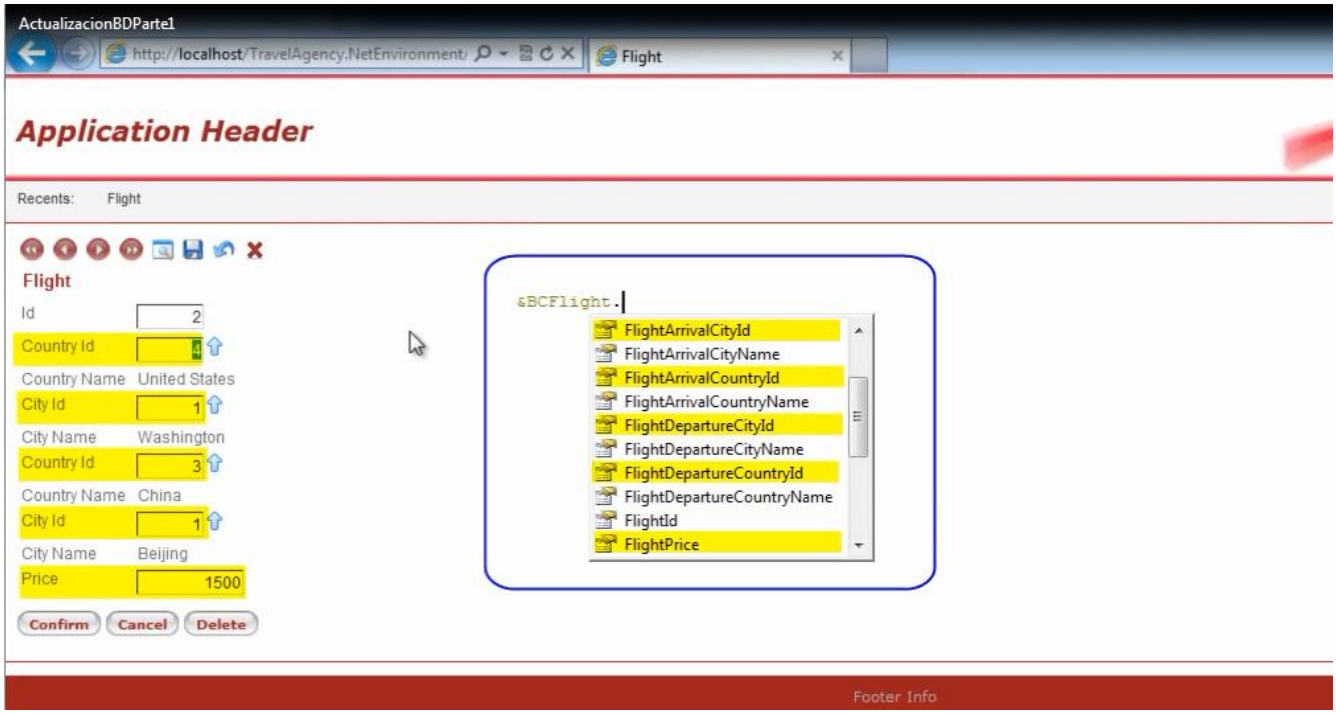
quando digitamos na transação um valor de identificador de voo e saímos do campo.

Carregam-se na memória todos os dados correspondentes a esse identificador.

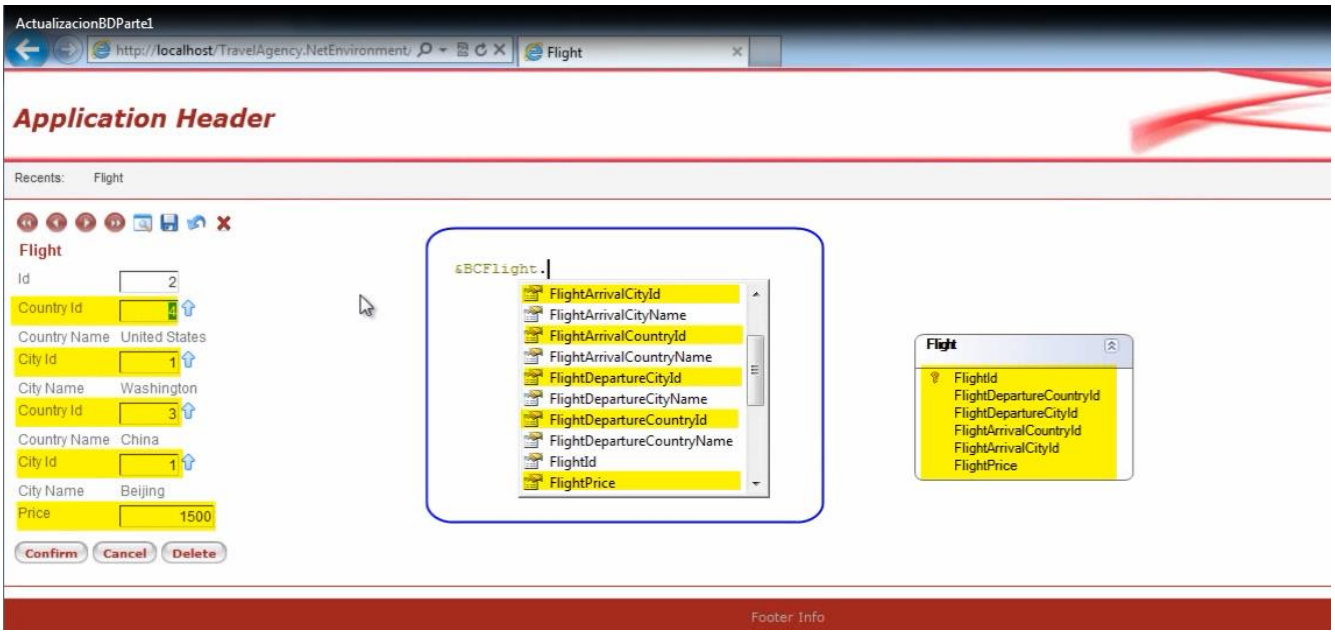


e os teremos disponíveis, neste caso, na variável &BCFlight.

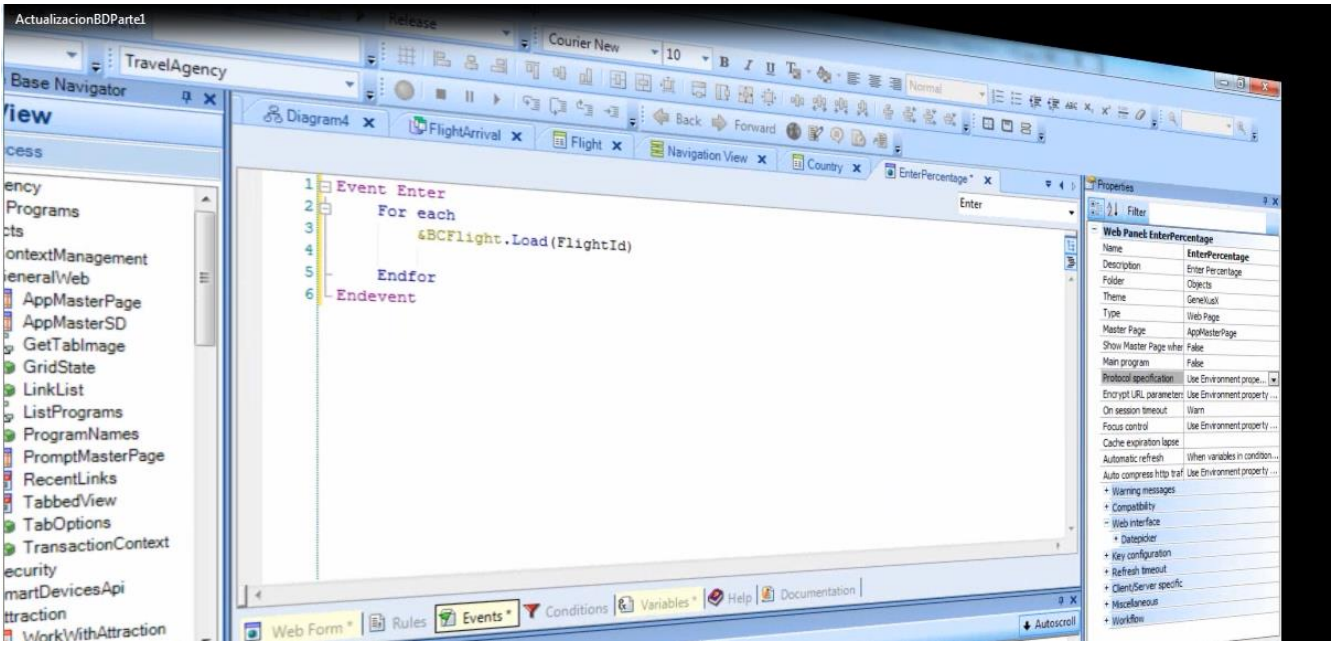
Podemos modificar <u>apenas</u> os mesmos valores de atributos que aparecem editáveis no form da transação



ou seja, os que se encontram na tabela base associada à transação; neste caso, FLIGHT.

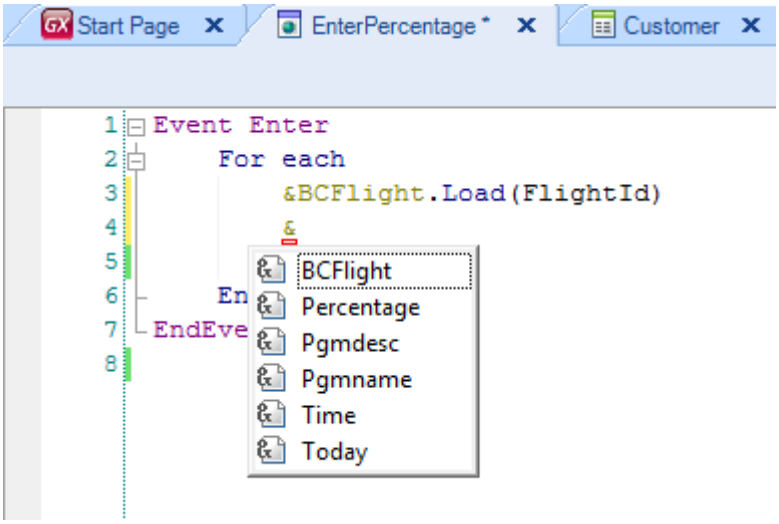


Voltemos, agora, ao código que estávamos escrevendo:

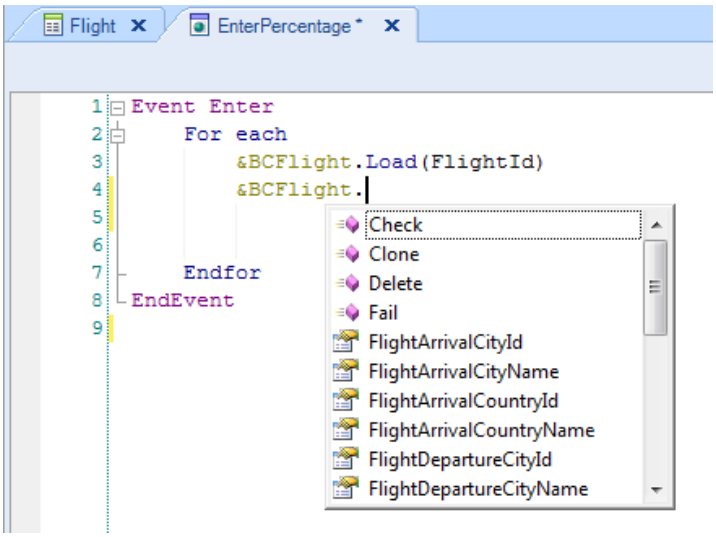


Como na primeira instrução do For each **carregaram-se os dados do voo navegando na memória**, agora nos resta apenas modificar o preço do voo e salvar as alterações.

Vamos escrever a segunda instrução dentro do For each. Digitamos ampersand (&)

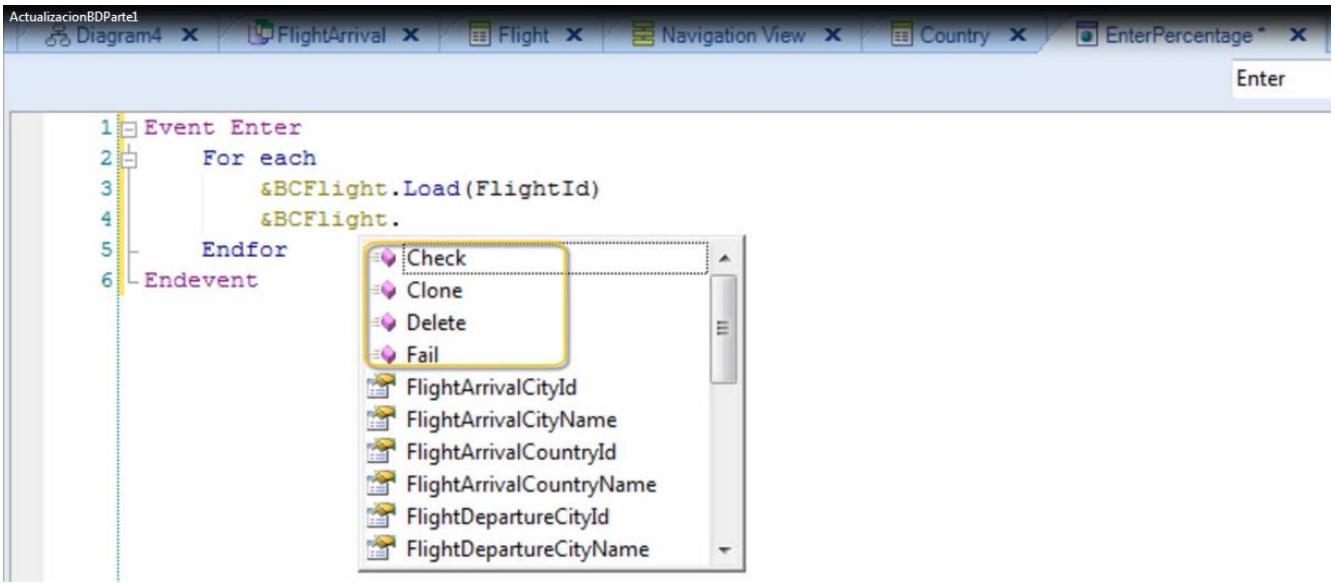


Selecionamos a variável &BCFlight, colocamos um ponto,

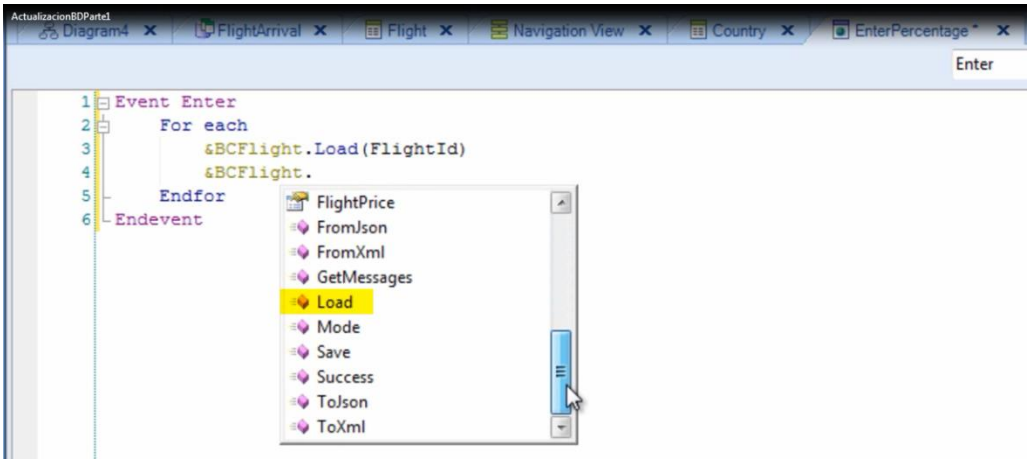


e vemos que temos dois tipos de elementos que podemos escolher.

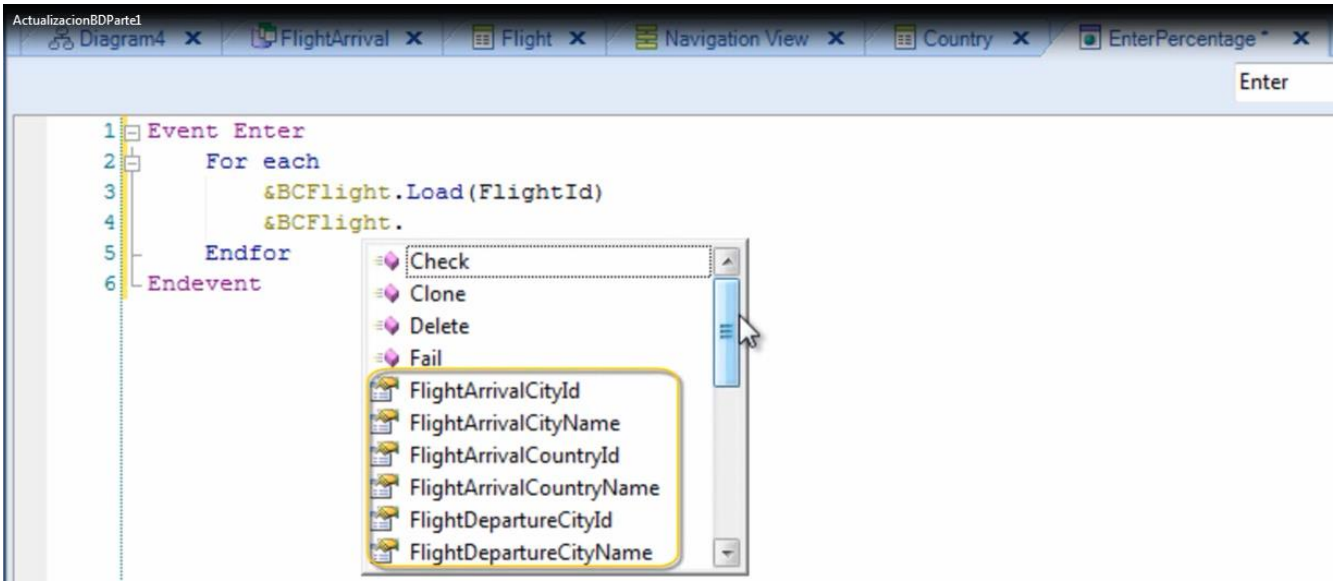
Os que têm um ícone cor violeta



são métodos para aplicar à variável, como o método Load que usamos e explicamos há pouco,



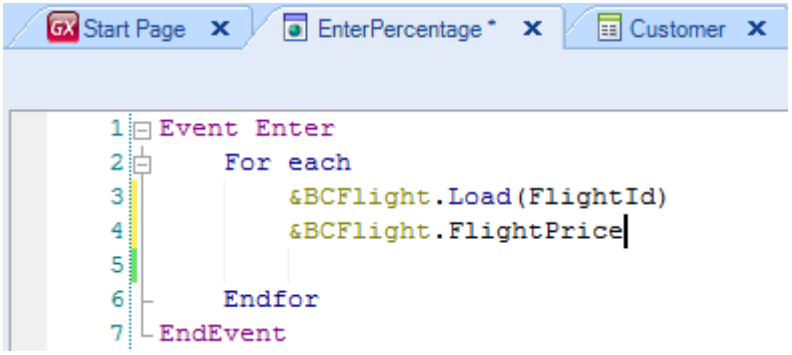
e os que têm o ícone com a mão



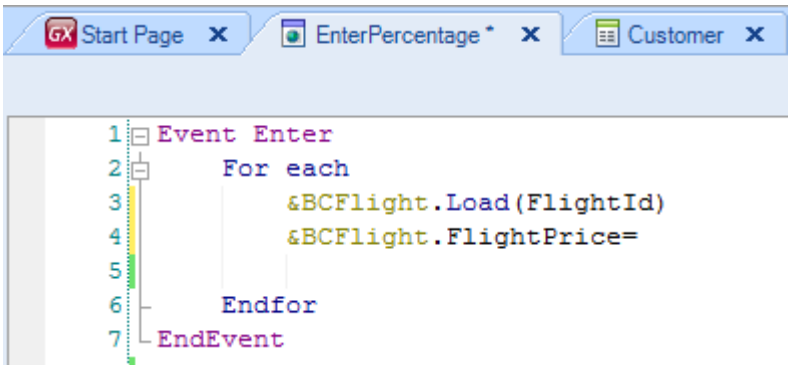
são atributos presentes na estrutura da transação que é business componente e de cujo tipo é a variável

&BCFlight.

Selecionamos o atributo FlightPrice



e para atribuir valor ao atributo, digitamos o sinal de igual



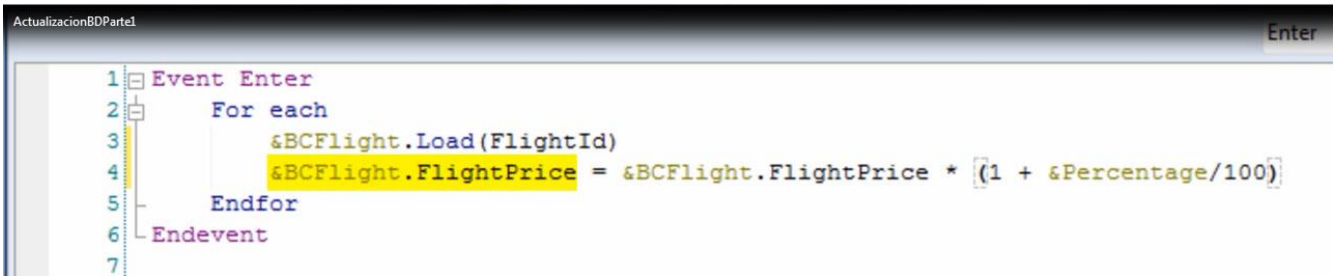
Agora, à direita do sinal, devemos definir o cálculo para obter o novo preço do voo, com a porcentagem do aumento aplicado.

Escreveremos o cálculo e, em seguida, o explicaremos

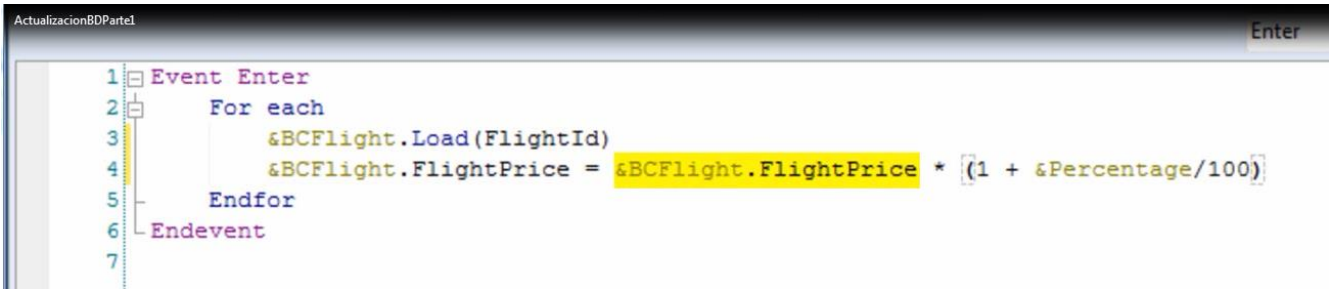
```
Event Enter
    For each
        &BCFlight.Load(FlightId)
        &BCFlight.FlightPrice= &BCFlight.FlightPrice * (1 + &Percentage/100)

    Endfor
EndEvent
```

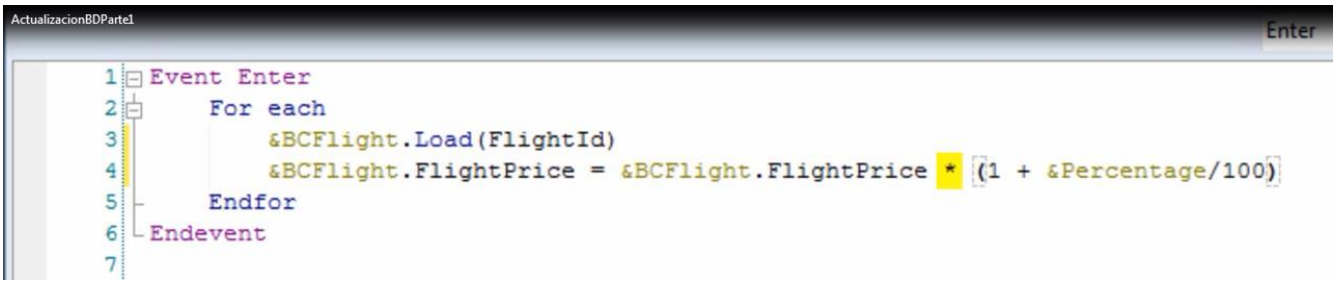
Essa instrução está atribuindo ao preço do voo que temos carregado na memória



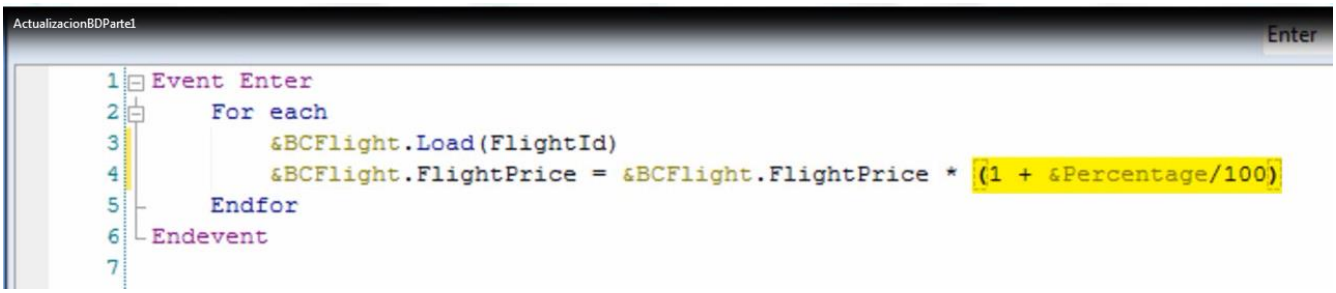
o mesmo valor **que tinha,**



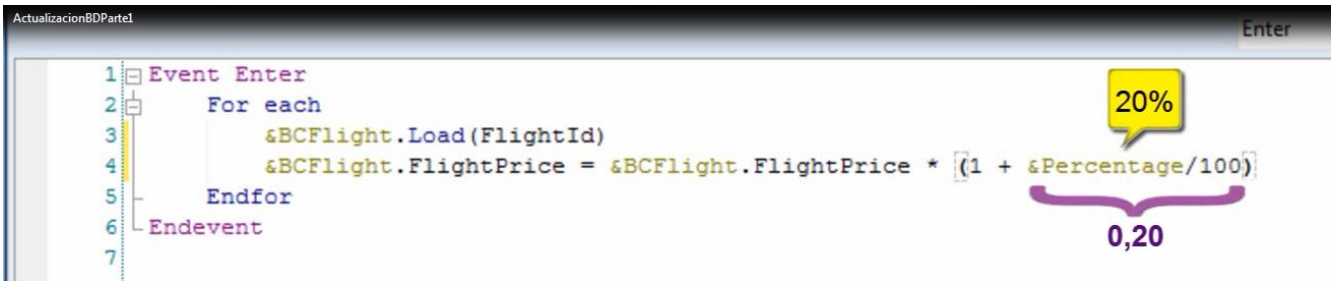
multiplicado pelo



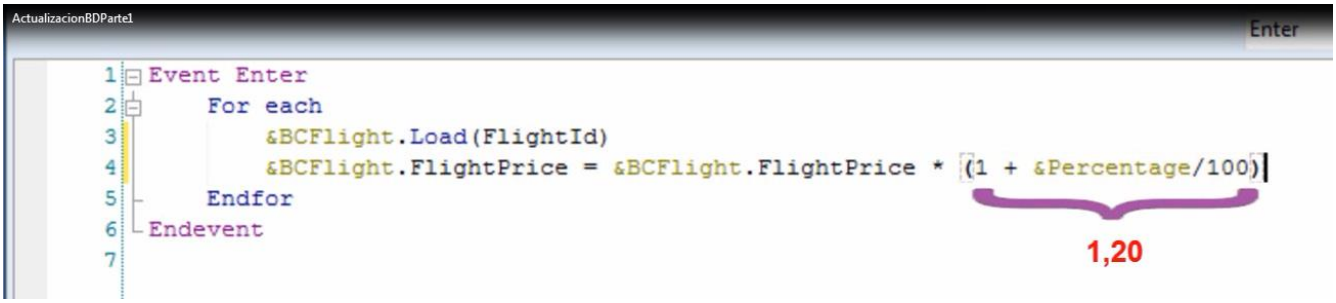
resultado da conta que está entre parênteses.



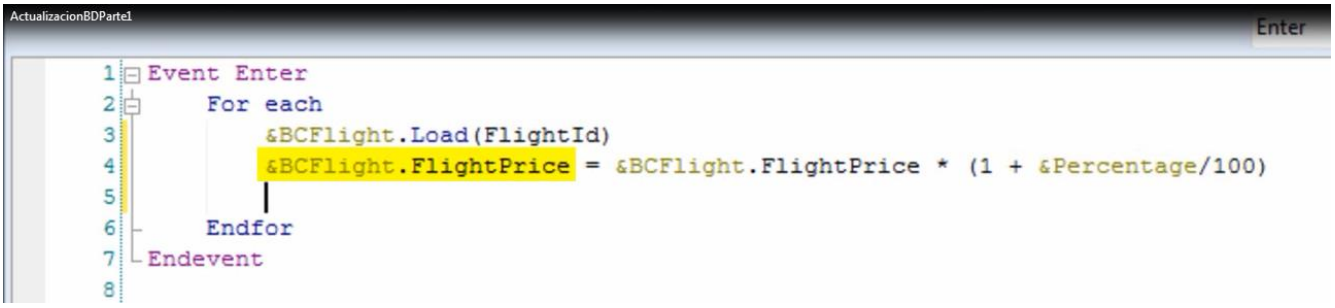
Este cálculo está somando 1 mais o valor que a variável &Percentage possui, dividido por 100... ou seja: se, por exemplo, a porcentagem de aumento que cadastraram na variável fosse 20,



ao multiplicar o preço atual por 1,20,

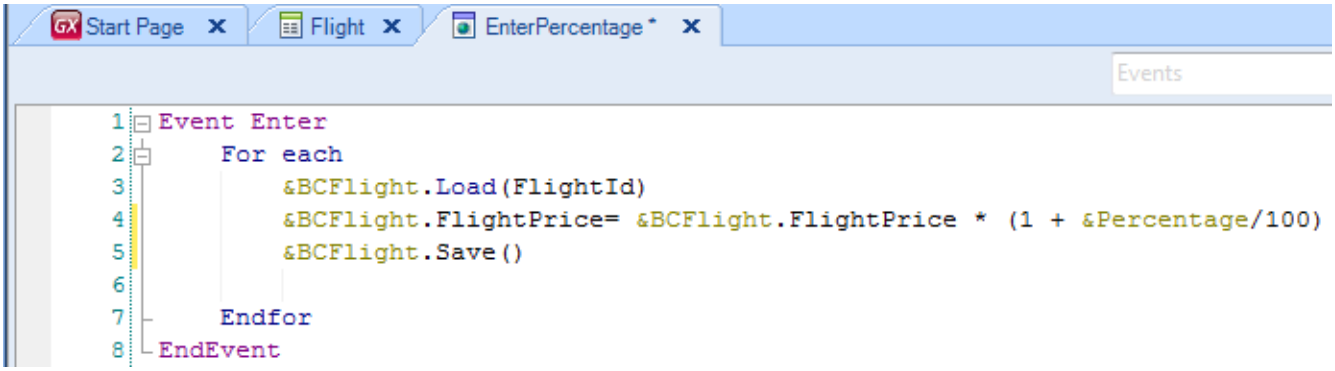


seria obtido o novo preço com 20% de aumento, o qual atribuímos o preço do voo.

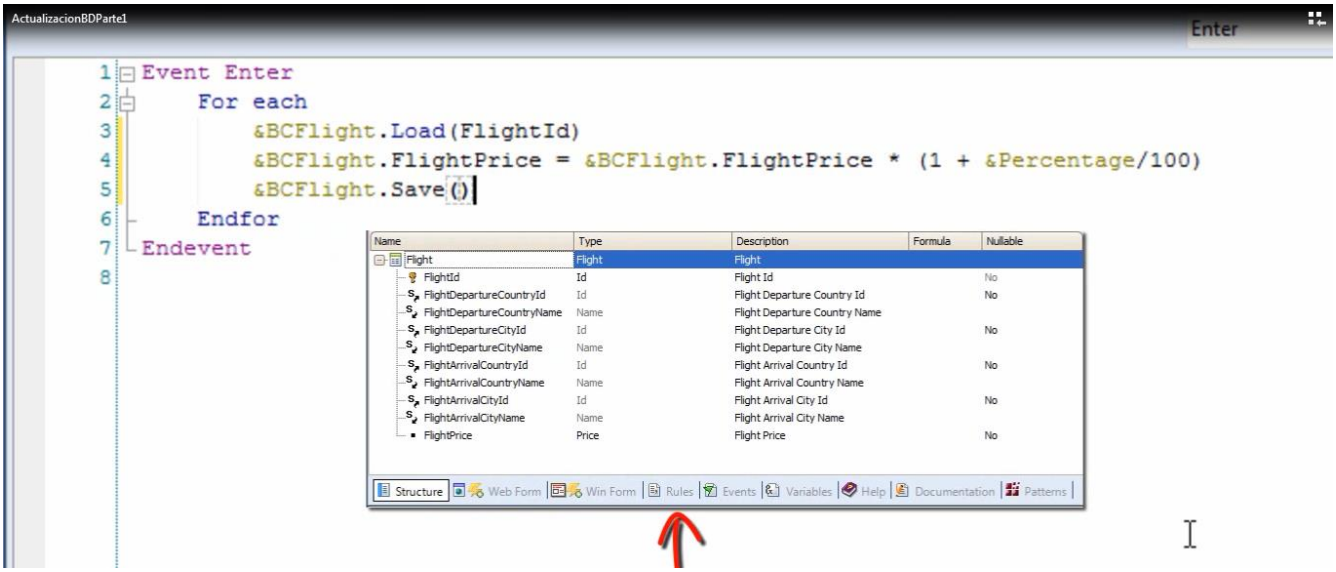


Agora devemos gravar fisicamente essa atualização. Escrevamos a instrução para fazê-lo.

&BCFlight, ponto... Selecionamos “save”... e vemos que os métodos sempre vão seguidos de parênteses.

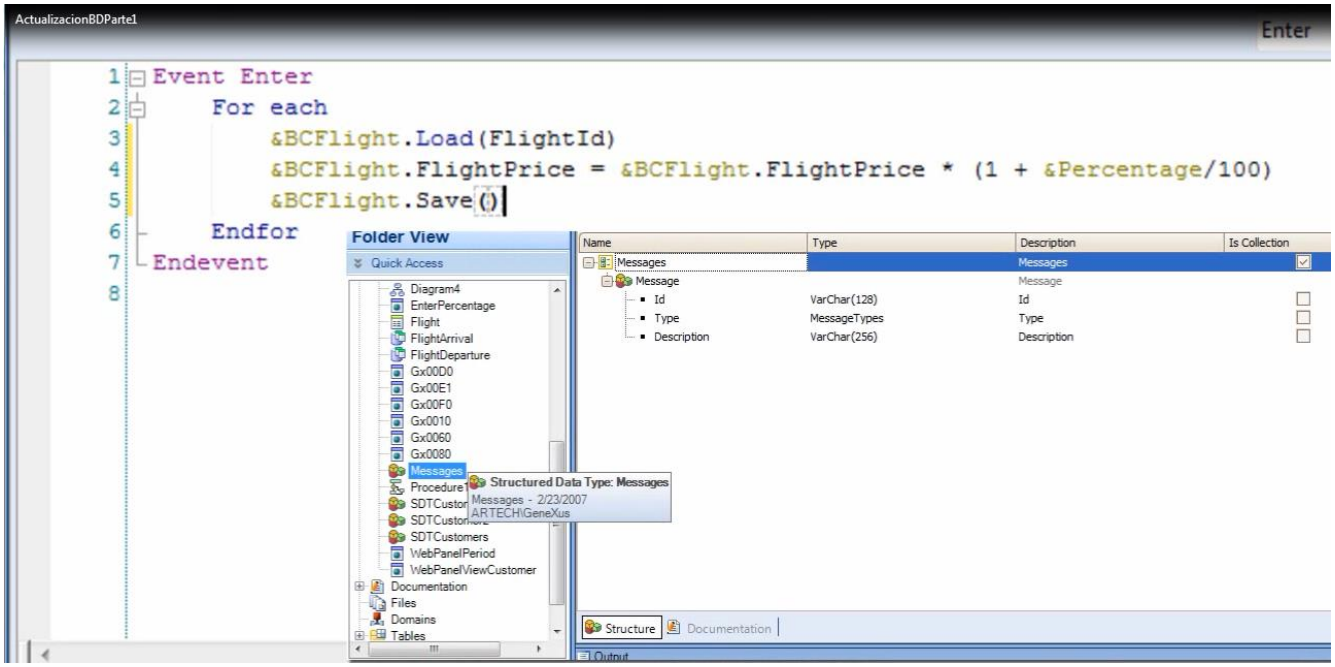


Ao executar o método save, serão disparadas regras definidas na transação Flight – com algumas exceções –

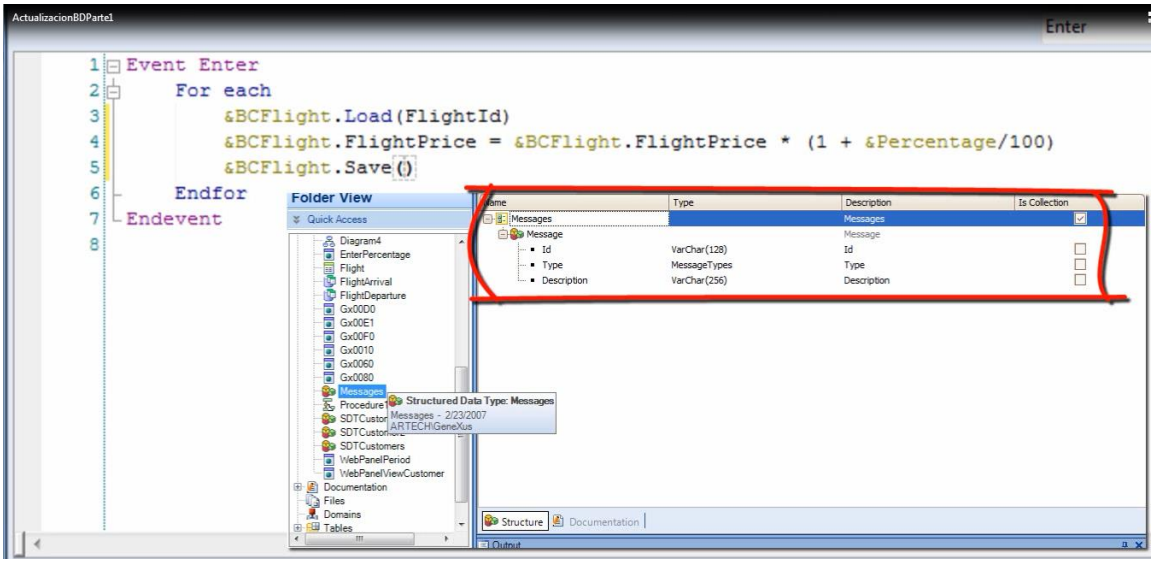


e são executadas as validações automáticas que a transação efetua.

Se ocorrerem erros (seja porque foram disparadas regras Error que tenhamos definido na transação ou por validações automáticas), os textos correspondentes ao que aconteceu ficarão carregados em uma coleção na memória

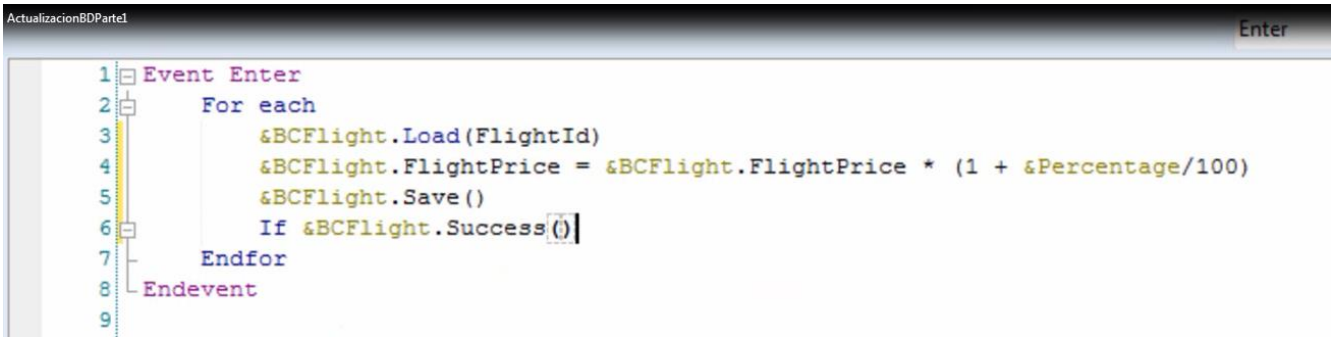


É possível recorrer e processar essa coleção.

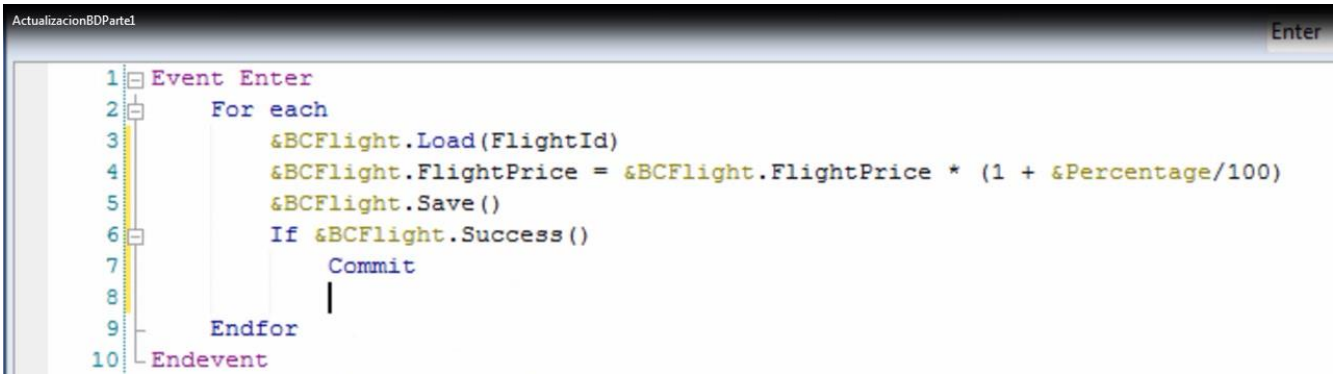


Não mostraremos isso nesse vídeo, mas em outro que ensina mais detalhes sobre business components.

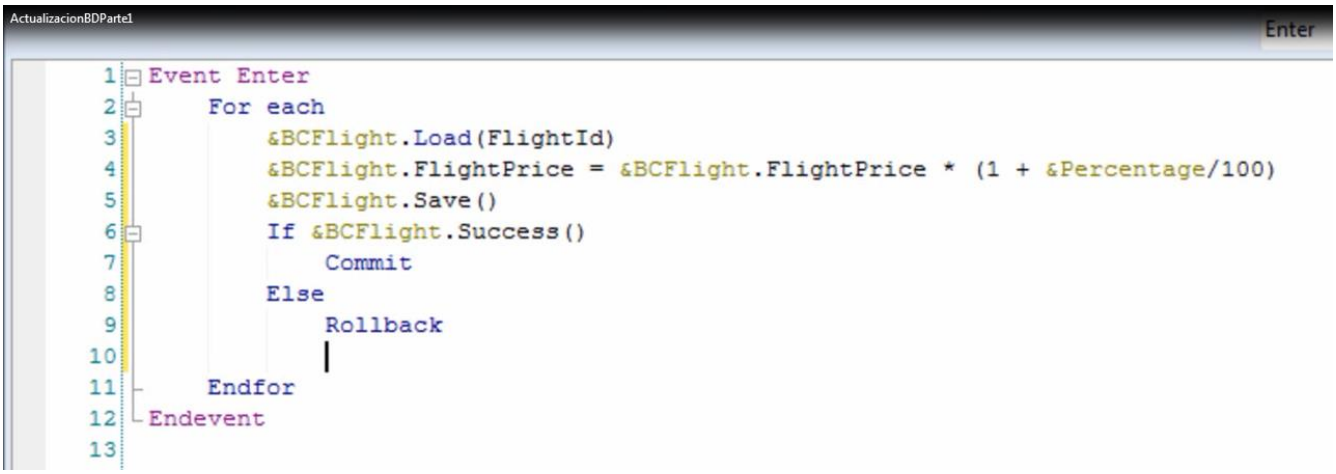
Agora, avaliaremos se a operação de gravação teve sucesso.



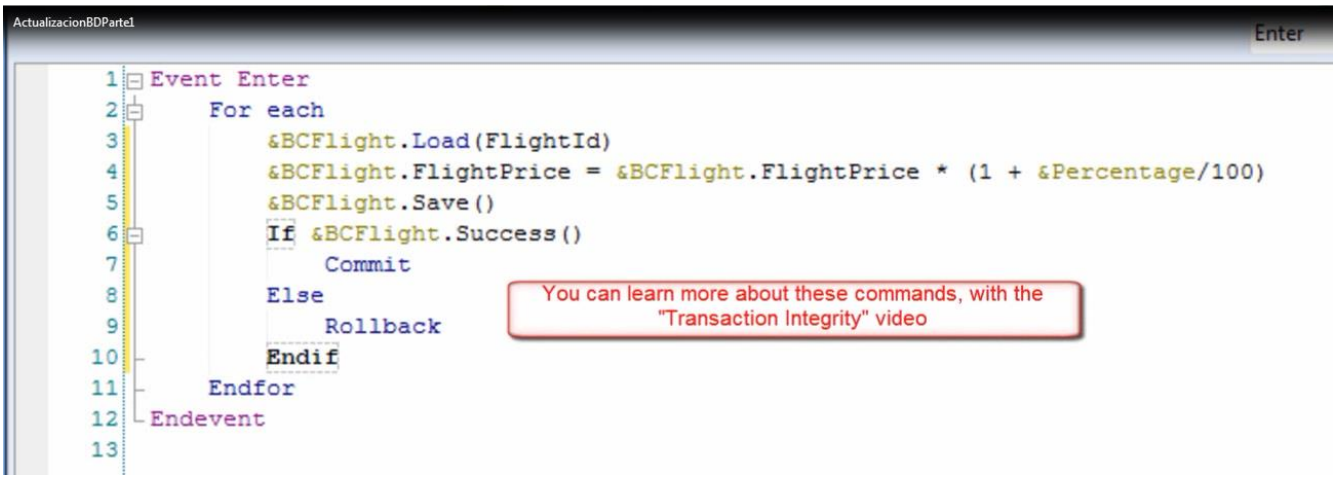
Caso tenha dado certo, escreveremos Commit



caso contrário, escreveremos Rollback



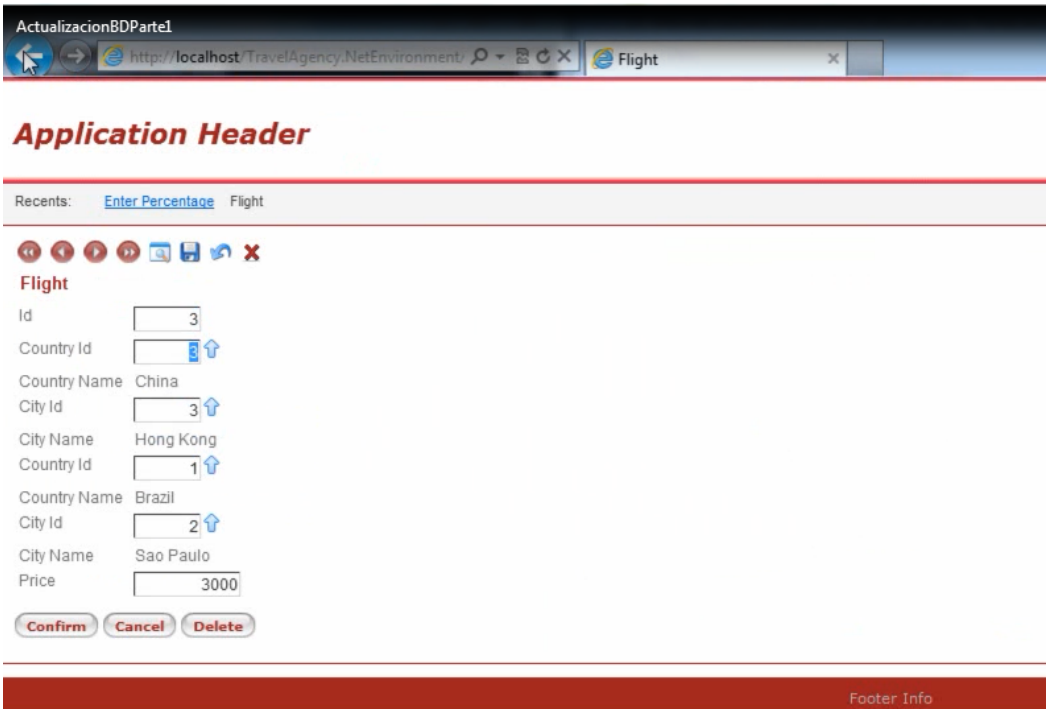
para desfazer as operações que tentamos efetuar no banco de dados posteriores ao último Commit.



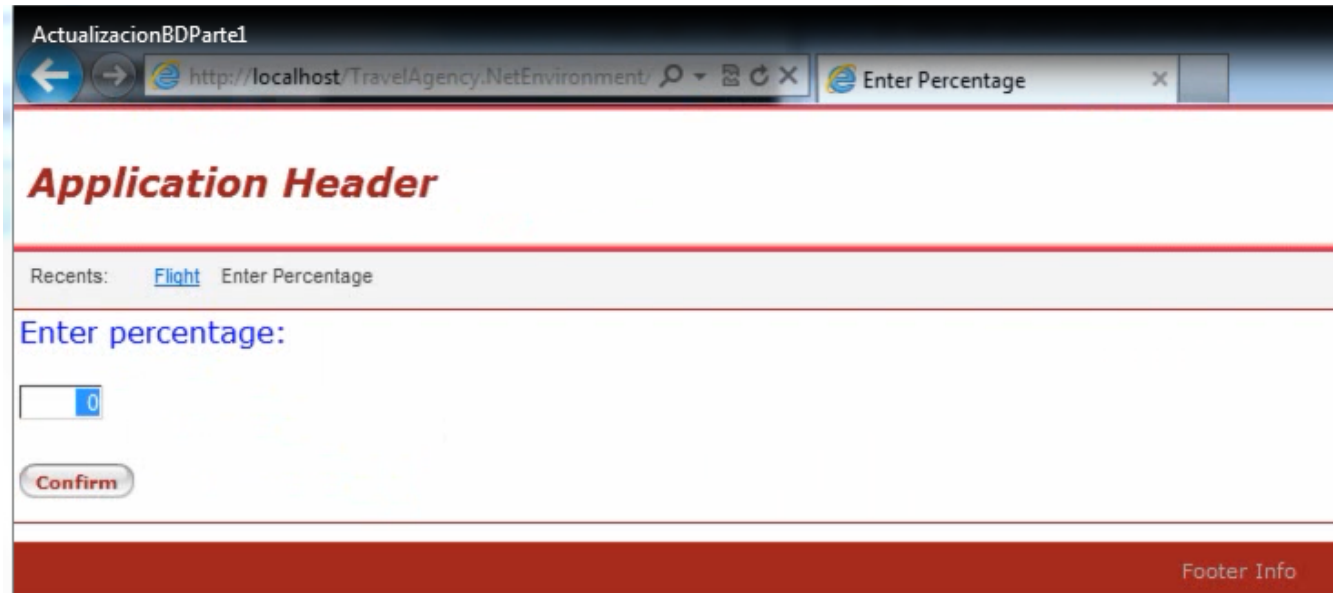
Vamos testar agora essa funcionalidade que acabamos de implementar.

Clicamos em F5...

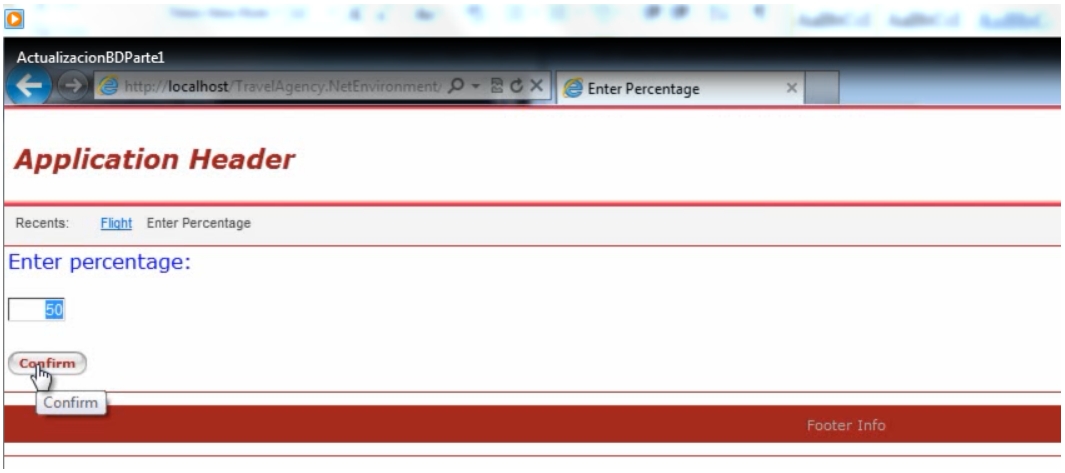
Executamos a transação Flight para revisar os preços de alguns voos…



E agora executamos a web panel “EnterPercentage”.

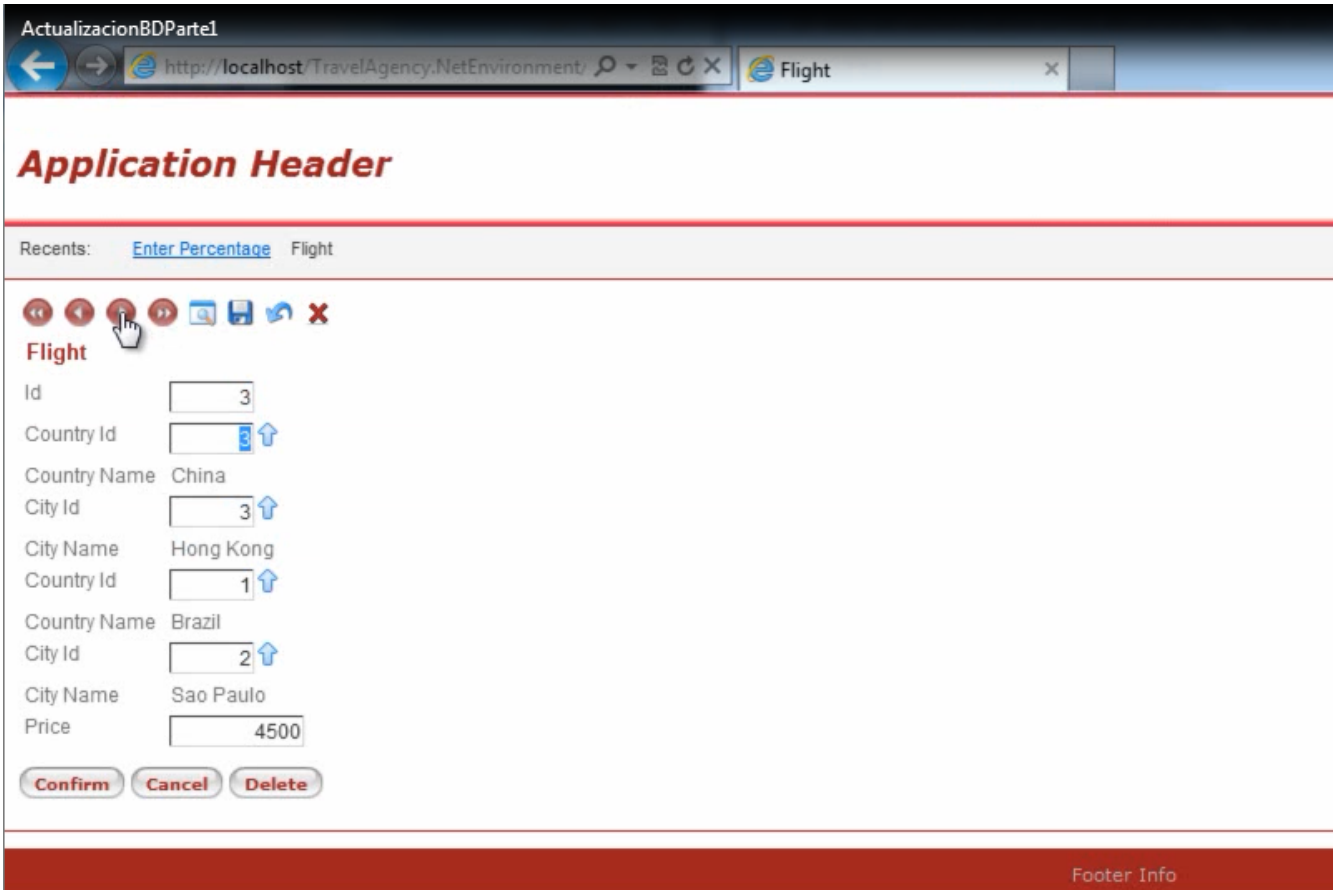


Inserimos uma porcentagem de aumento de 50% e clicamos no botão confirmar.

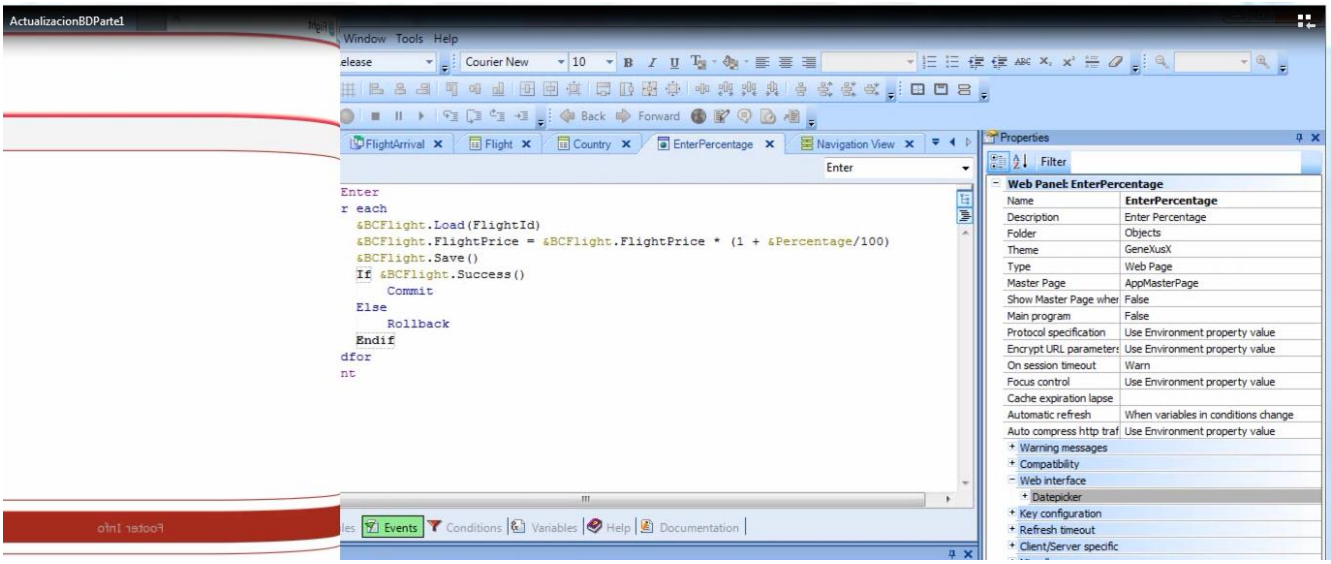


Não incluímos nenhuma mensagem que avise que o processo foi realizado e terminou, mas vamos confirmar executando a transação “Flight” novamente.

Se nos lembramos dos preços que havíamos cadastrado, podemos perceber que todos tiveram um aumento de 50%.

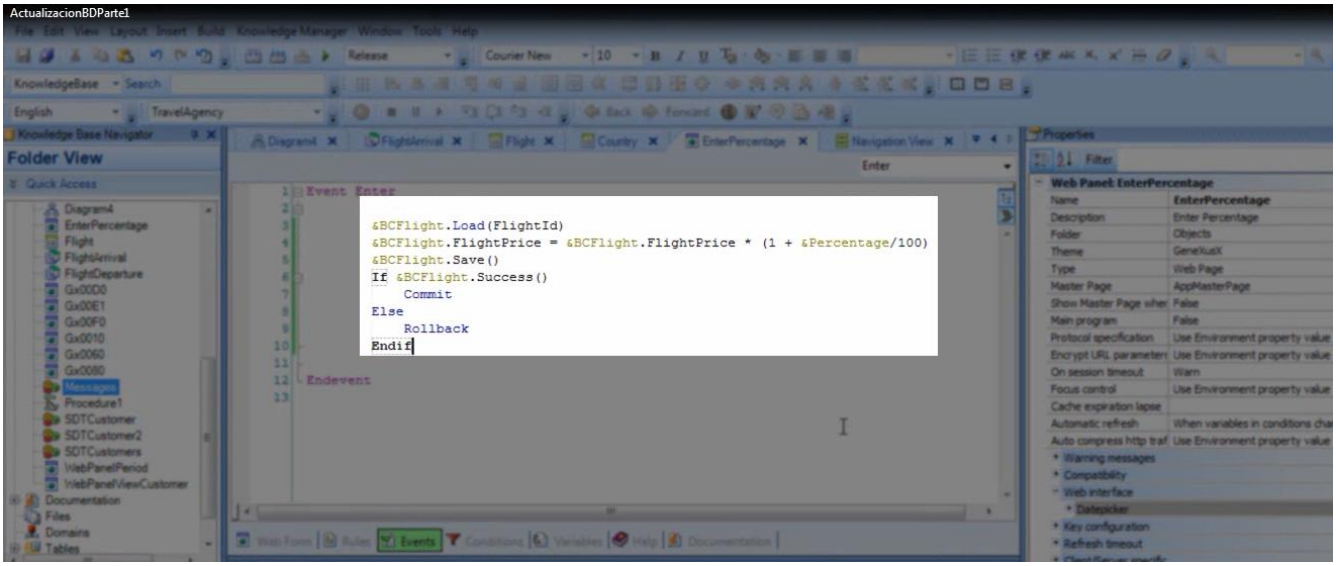


Voltemos, agora, à codificação que fizemos.

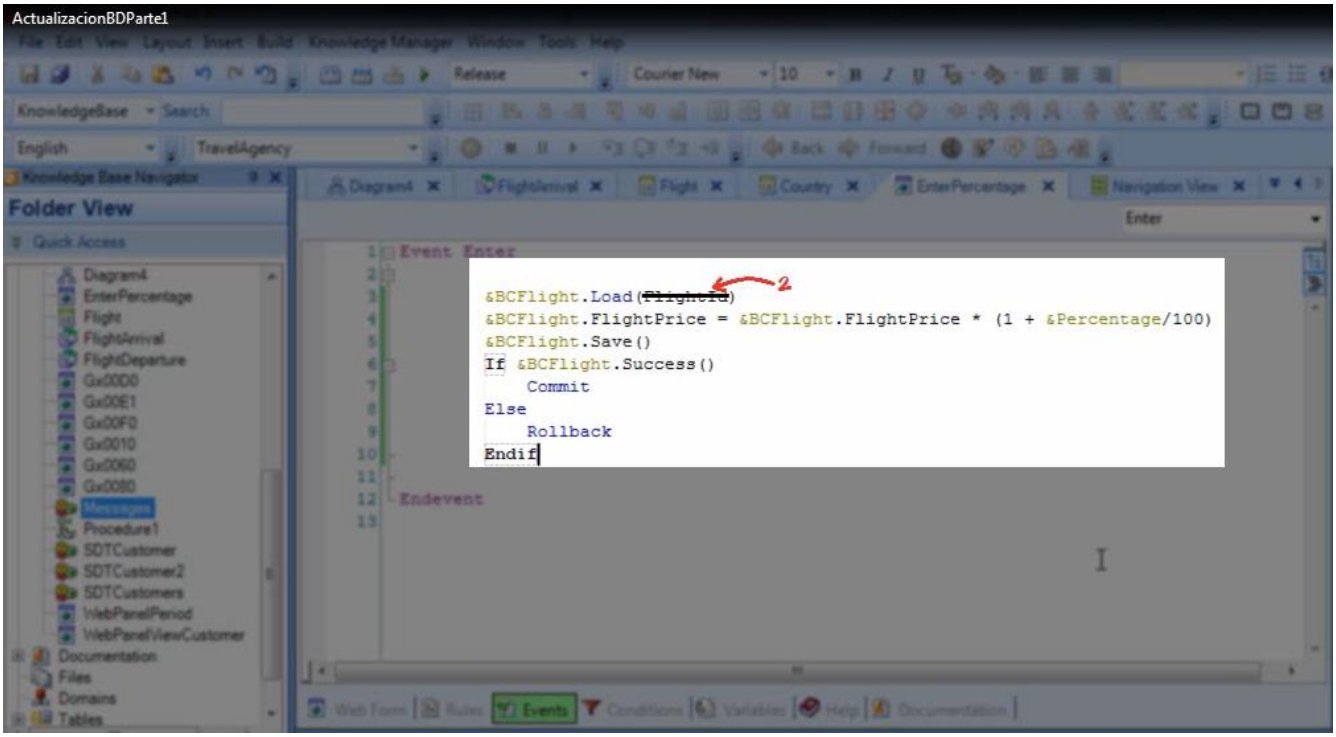


É importante esclarecer que podemos usar uma variável do tipo business component fora de um comando For Each.

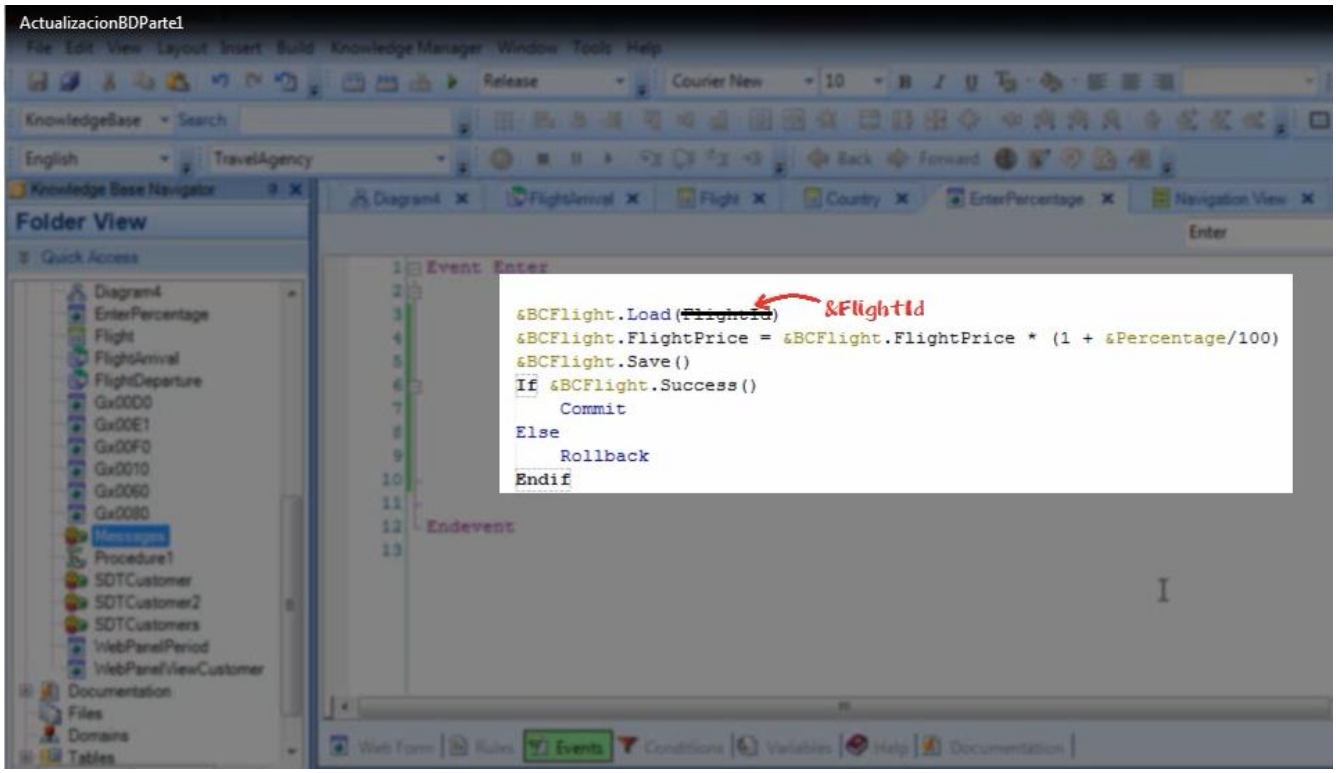
Por exemplo, todo este código



é válido fora do For each, com a única ressalva de que o valor do identificador de voo a ser carregado na memória teria que ser especificado como valor fixo; por exemplo assim:

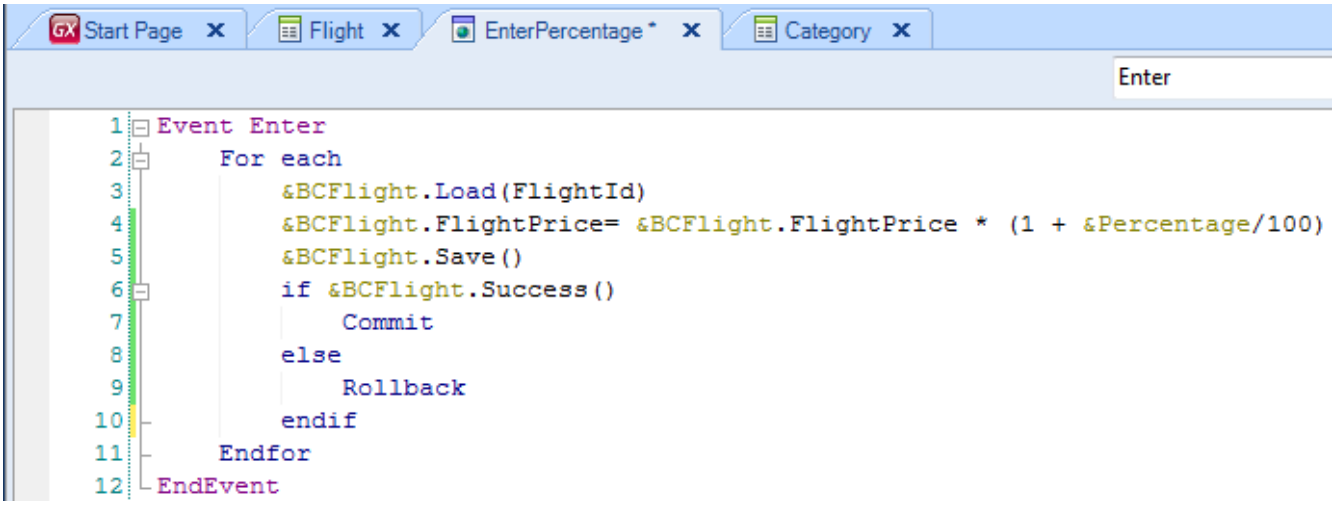


ou tê-lo carregado em uma variável e especificar a variável dentro do parênteses, assim:



Essa seria uma carga na memória pontual **de um voo**, e estaríamos atualizando o preço **desse voo**...

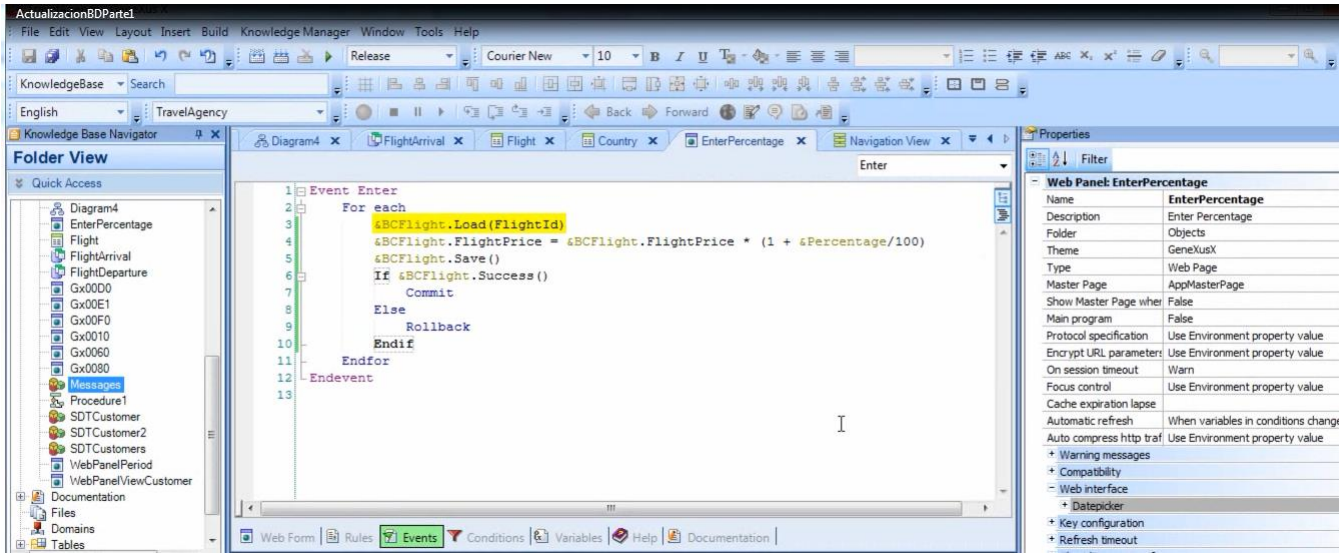
Não é uma atualização em massa, como a que fizemos dentro do For each.



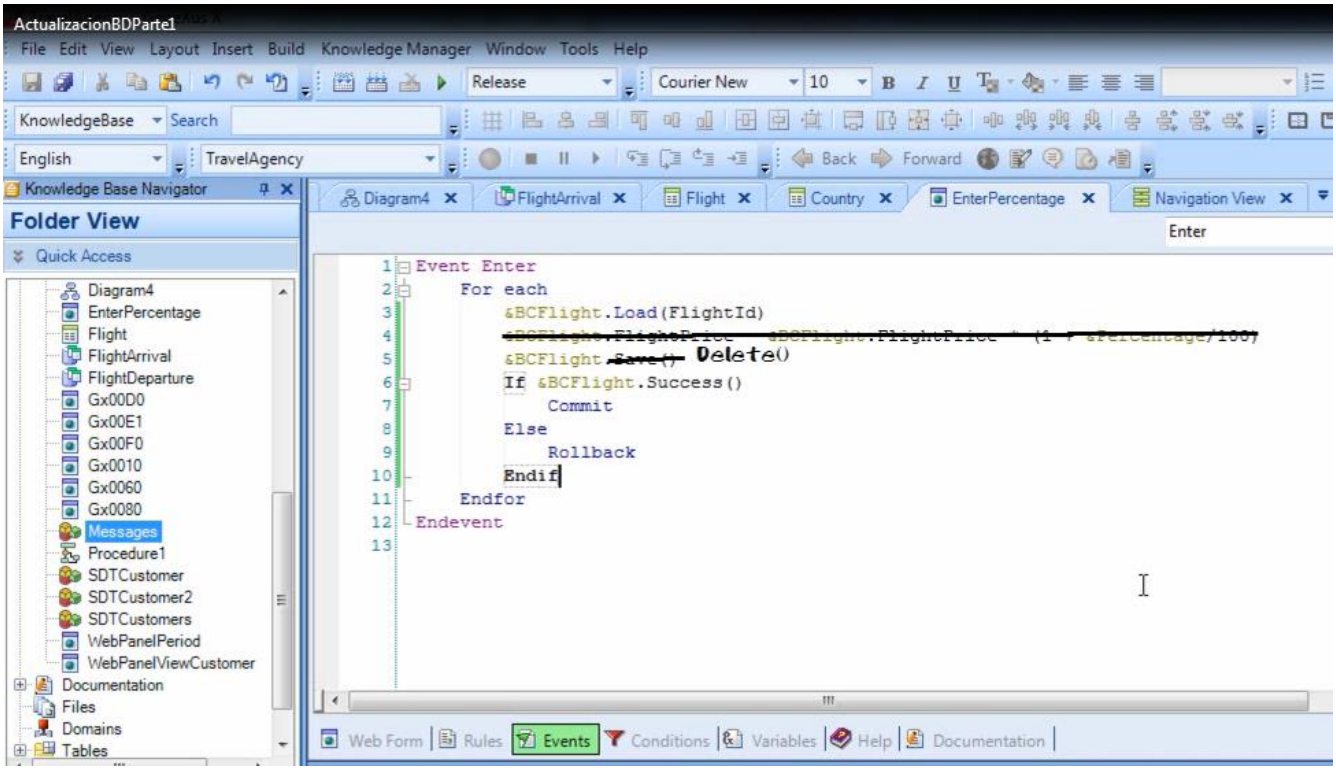
Agora, suponhamos que desejamos excluir os voos que temos registrados.

Para excluir no banco de dados usando business components, é preciso fazer quase o mesmo que fizemos para atualizar:

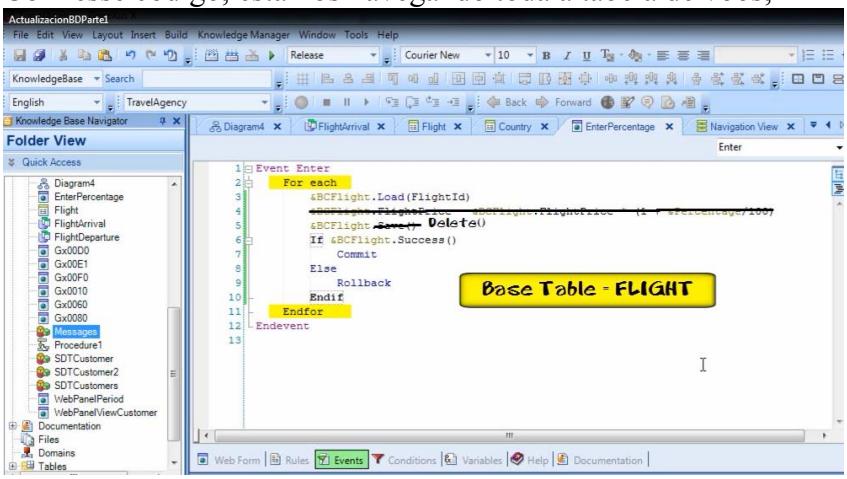
Primeiro, carregar na memória os dados associados a uma chave primária usando o método Load,



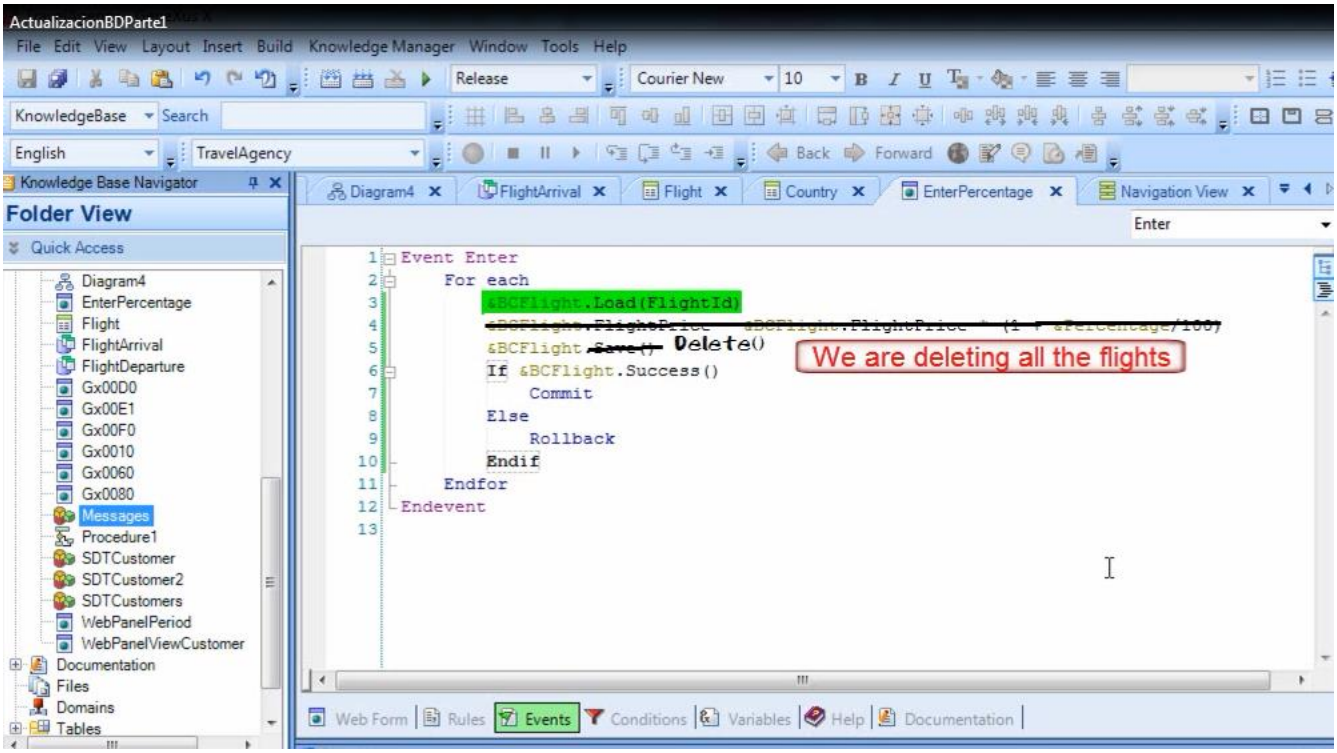
e depois executar o método Delete para efetuar a exclusão.



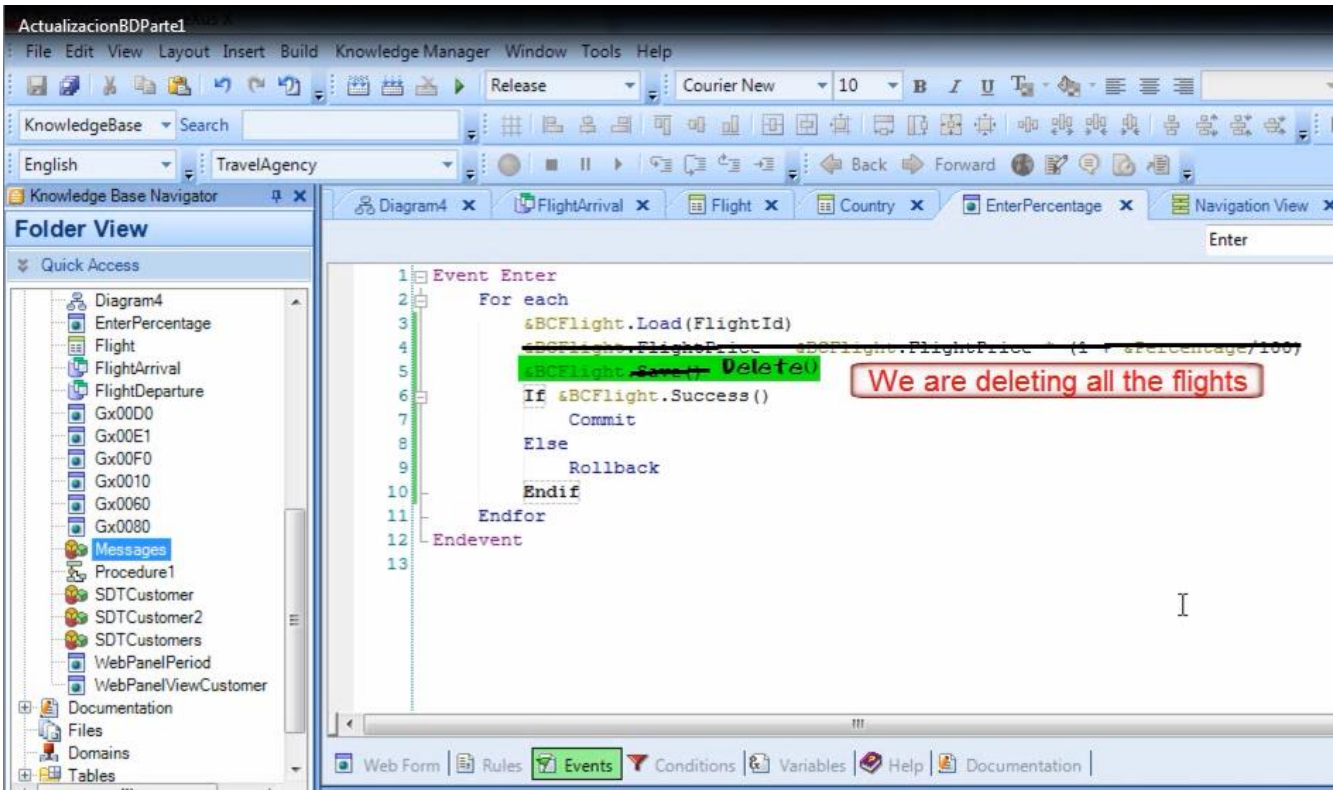
Com esse código, estamos navegando toda a tabela de voos,



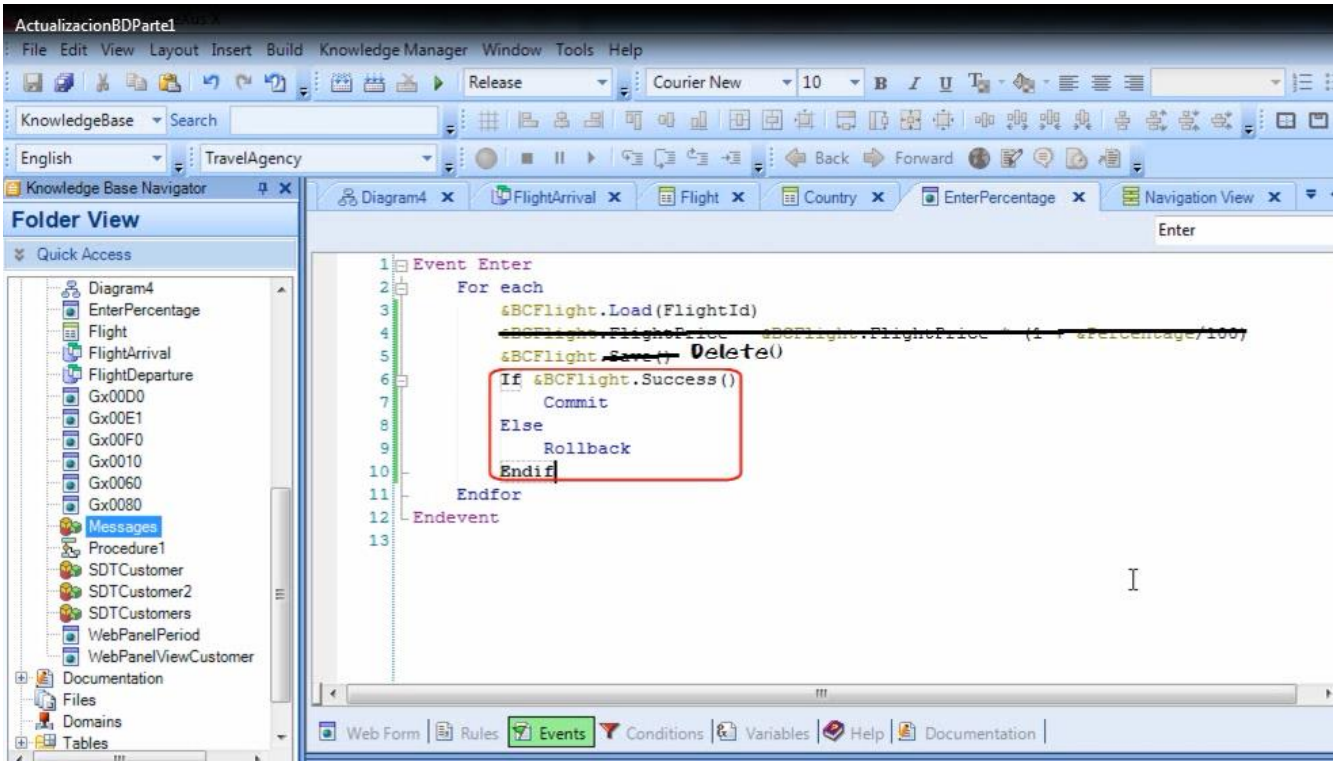
carregamos cada voo navegando na memória



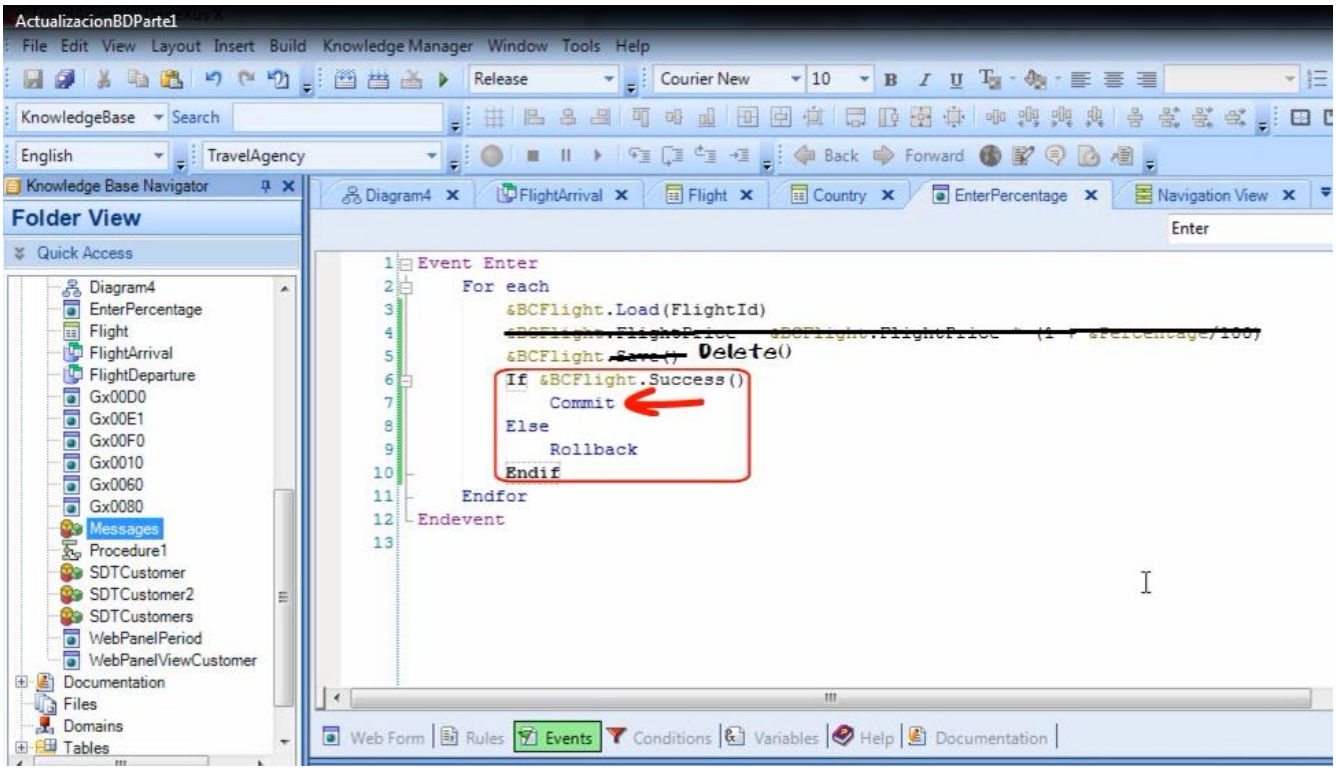
e excluimos o voo.



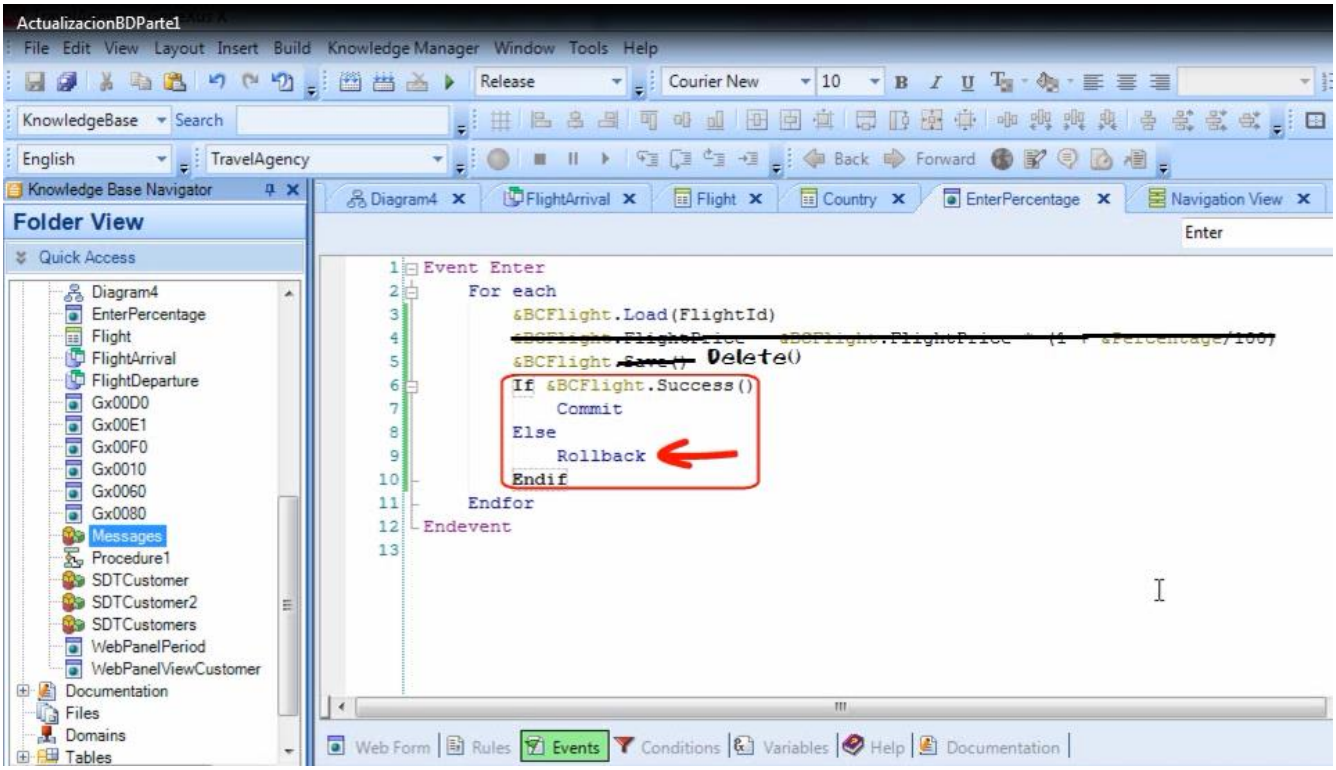
Esse último bloco de código



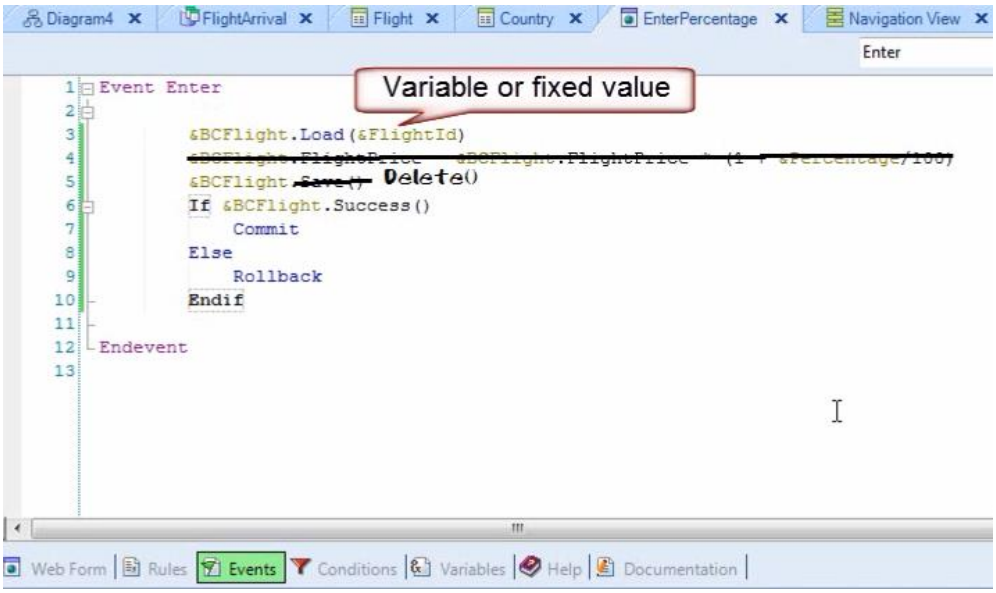
mantém-se igual, validando caso não tenham ocorrido erros ao tentar eliminar, então confirmamos a eliminação do banco de dados



caso contrario, não.

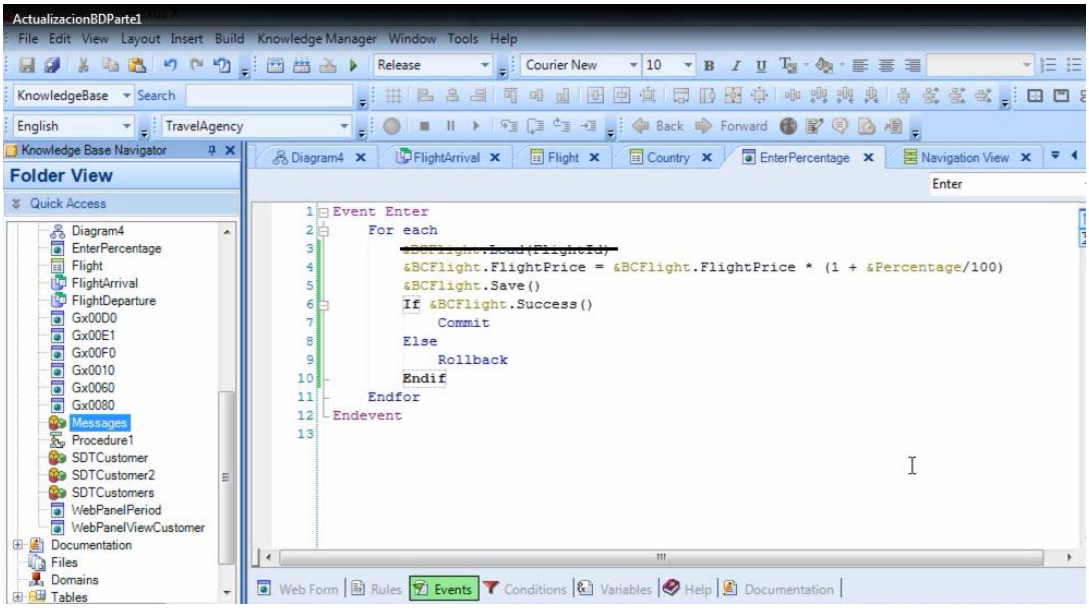


Como vimos para atualização, podemos querer excluir um voo pontualmente e, para isso, codificaríamos a carga na memória do voo específico a ser apagado, fora do comando For each



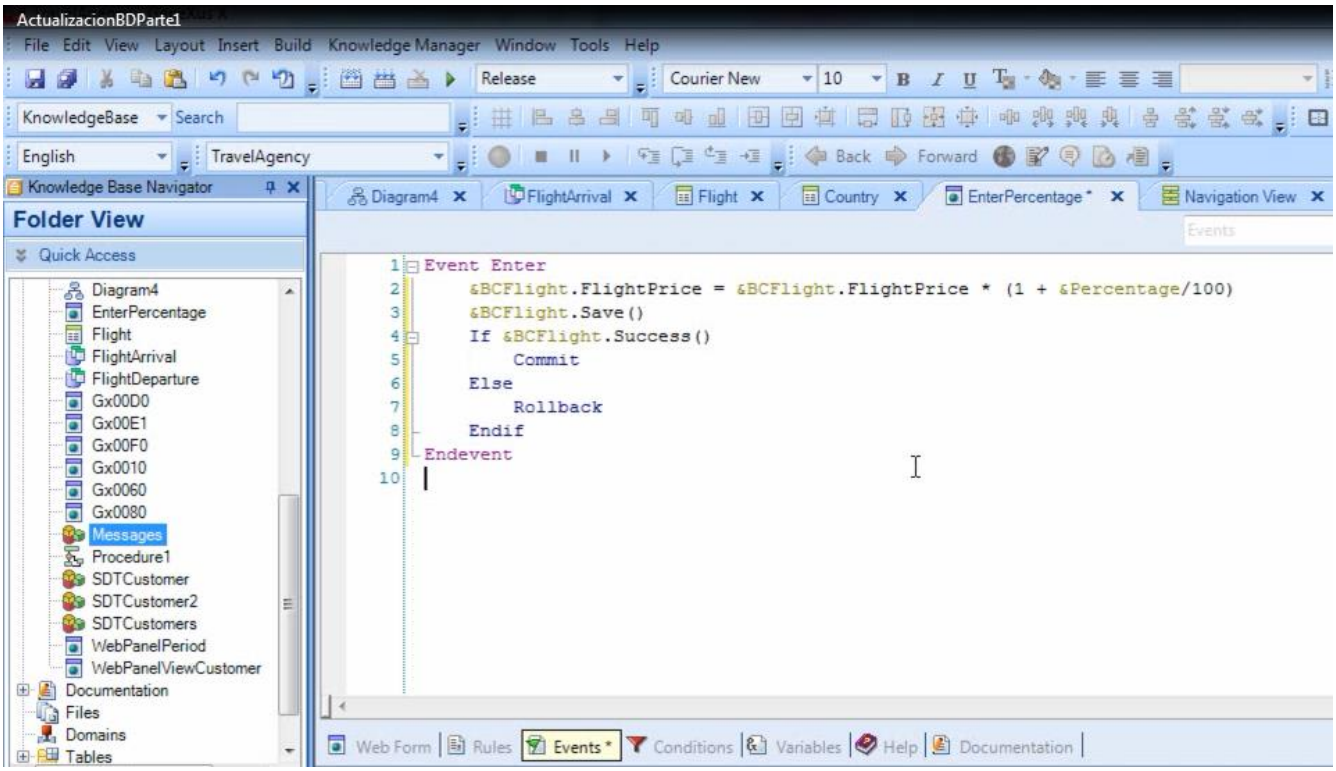
Resta-nos ver como inserir um voo, usando business componentes.

O procedimento é semelhante ao de atualização, com a única diferença **de que não é preciso efetuar o Load**

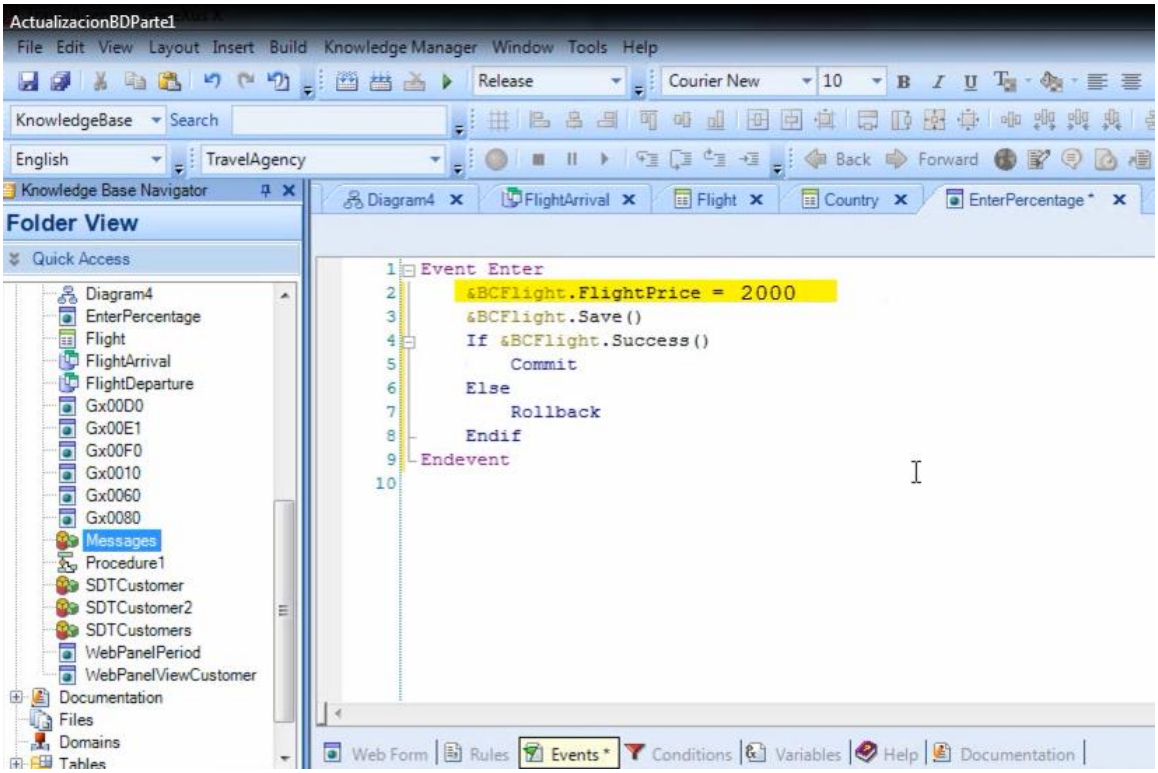


já que não vamos recuperar um voo, mas inserir um novo.

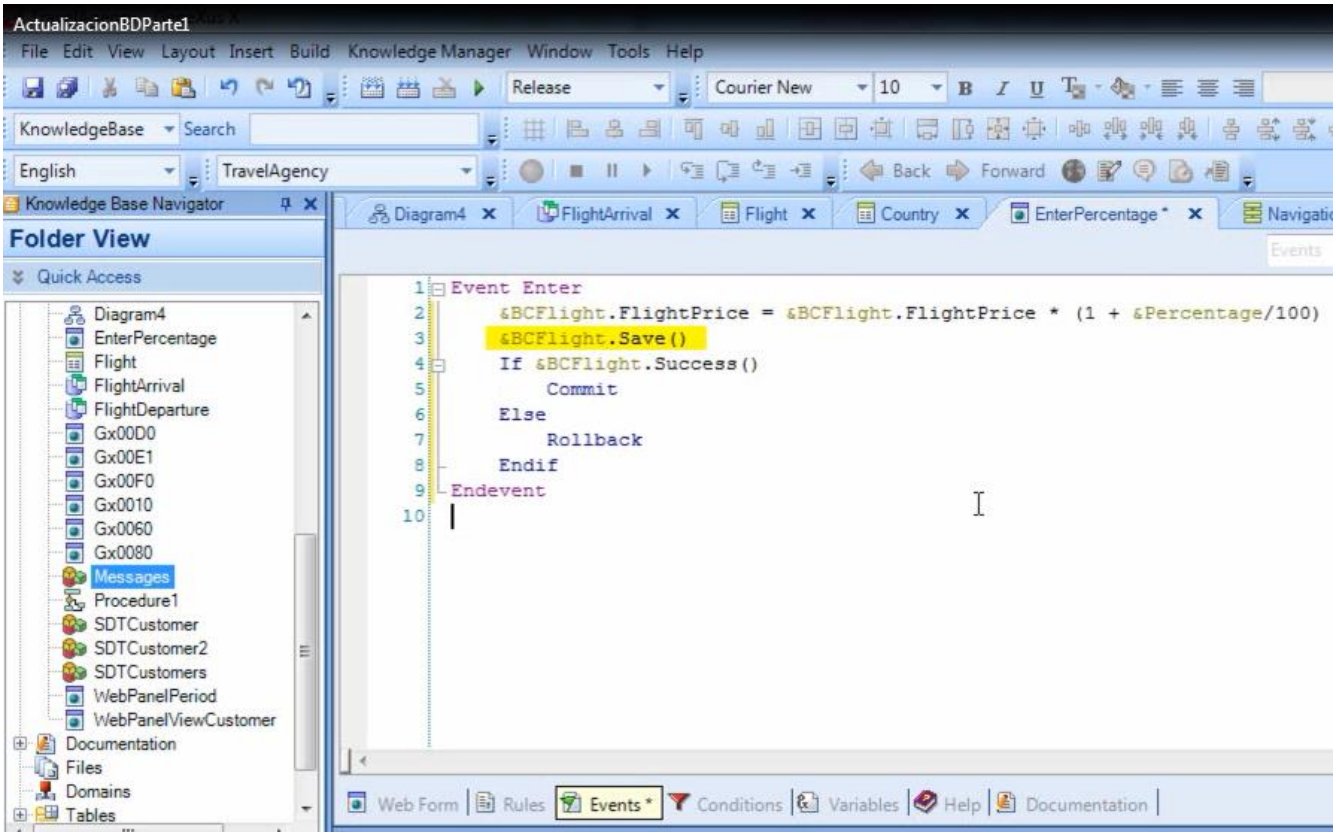
Apagamos o For each,/ endfor e a linha com método Load



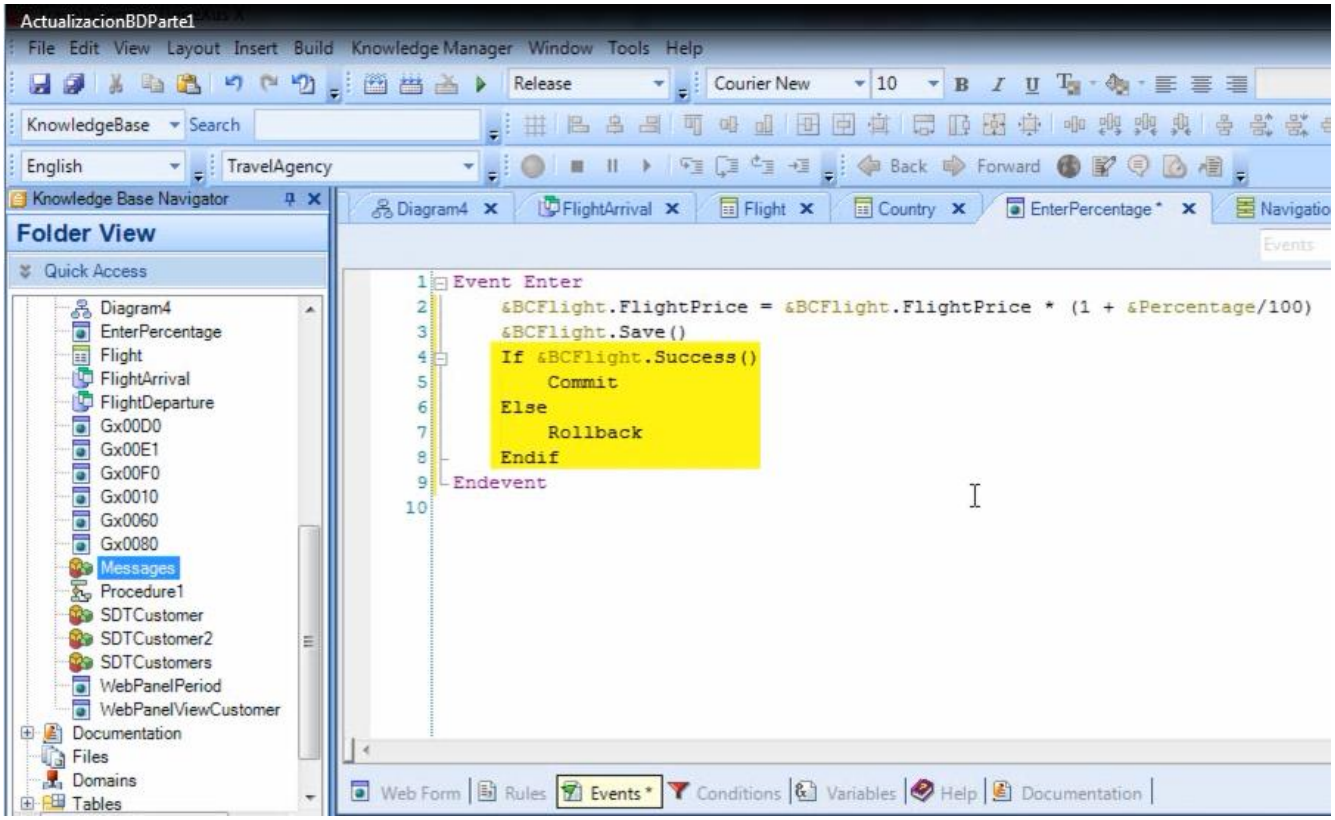
Seja um evento Enter de um objeto, ou o no contexto em que estamos... para inserir um registro empregando o conceito de business component, é preciso, basicamente, atribuir valores aos atributos que nos interessam



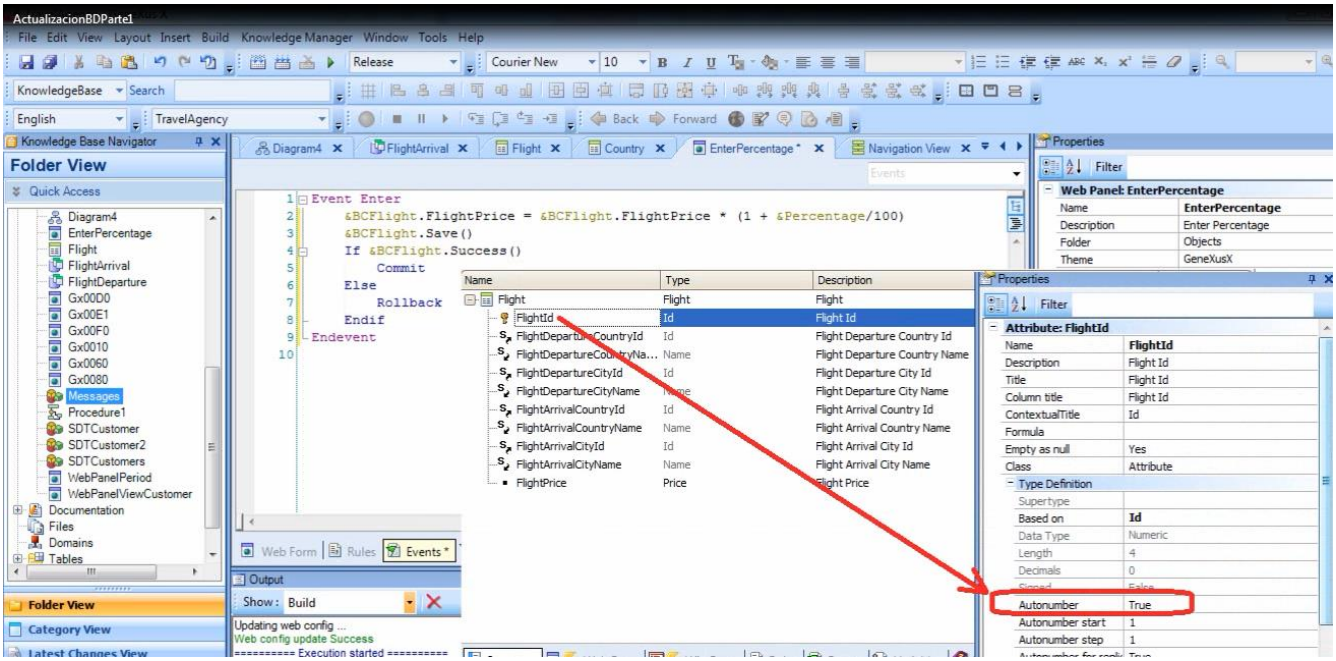
e logo depois, executar o método save para a variável



e executar o commit ou rollback no banco, dependo se a operação deu certo ou não, como já vimos.



Uma vez que FlghtId tem a propriedade autonumber com valor True



não é necessário atribuir valor ao atributo, de modo semelhante a quando operamos com o form da transação Flight.

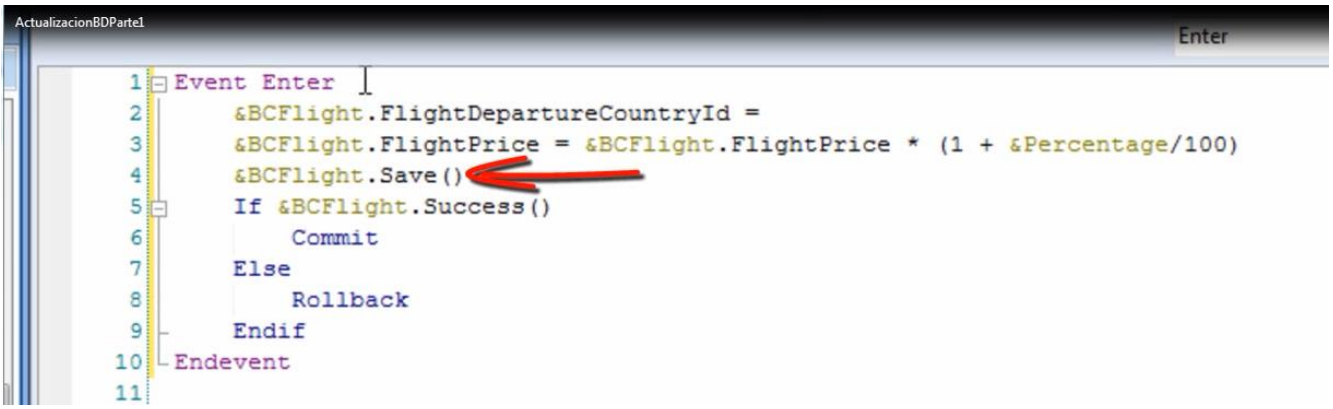
Se no novo voo que queremos inserir, queremos atribuir determinado país e cidade de origem e determinado país e cidade de destino... escrevemos como vemos fazendo &BCFlight, ponto, escolhemos FlightDepartureCountryId, e lhe atribuímos valor...

ActualizacionBDParte1 — Enter

```
Event Enter
    &BCFlight.FlightDepartureCountryId =
    &BCFlight.FlightPrice = &BCFlight.FlightPrice * (1 + &Percentage/100)
    &BCFlight.Save()
    If &BCFlight.Success()
        Commit
    Else
        Rollback
    Endif
Endevent
```

E assim a cada atributo que nos interessa carregar valores, até que chegue o momento de fazer save e commit ou rollback.

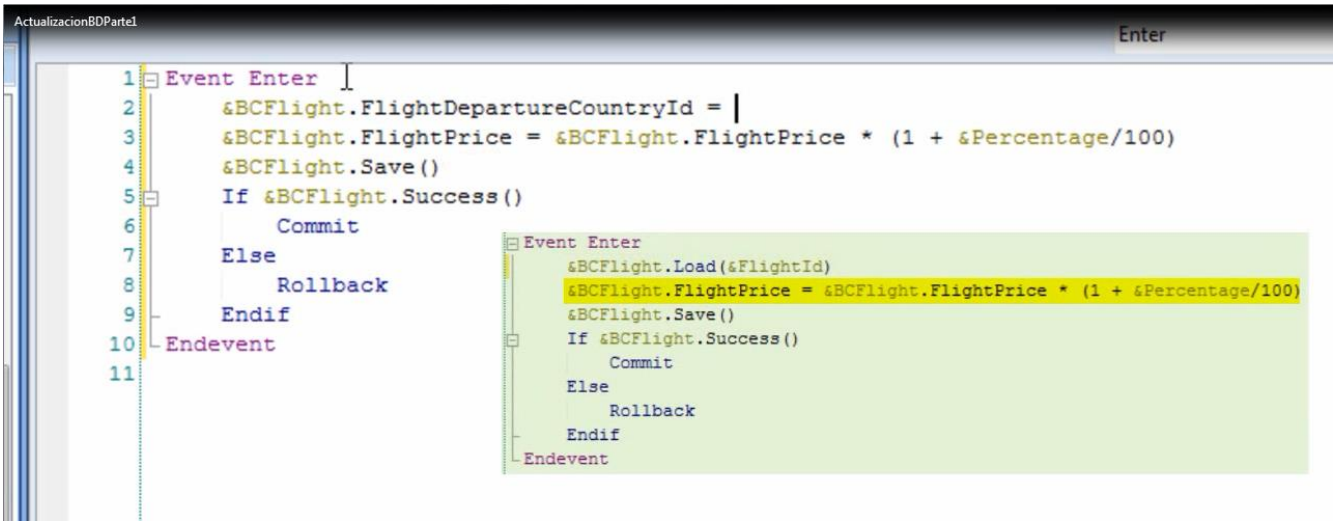
O método save, como vimos,



é usado para salvar tanto o acréscimo de um novo registro, como para salvar uma atualização.

Se primeiro carregamos na memória um registro,

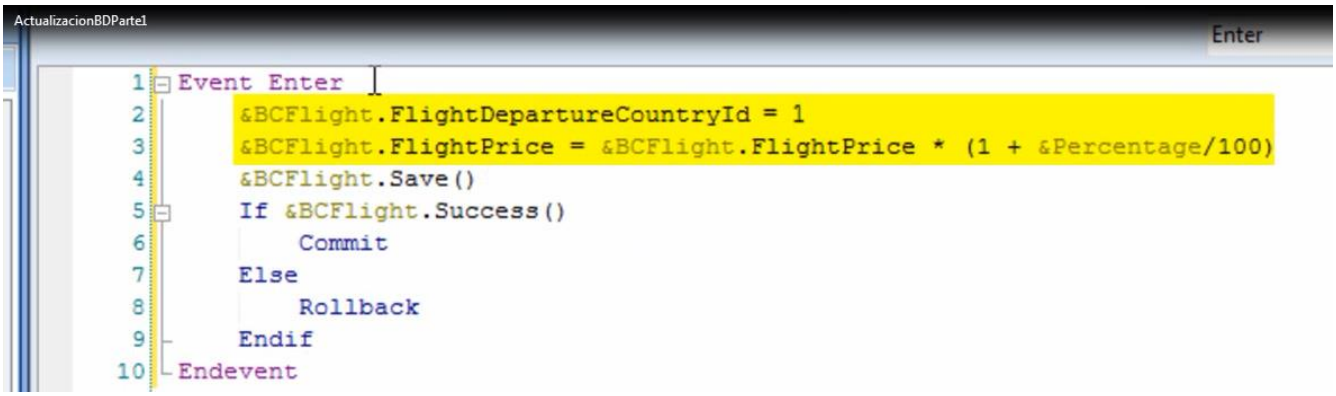
e atribuimos valores



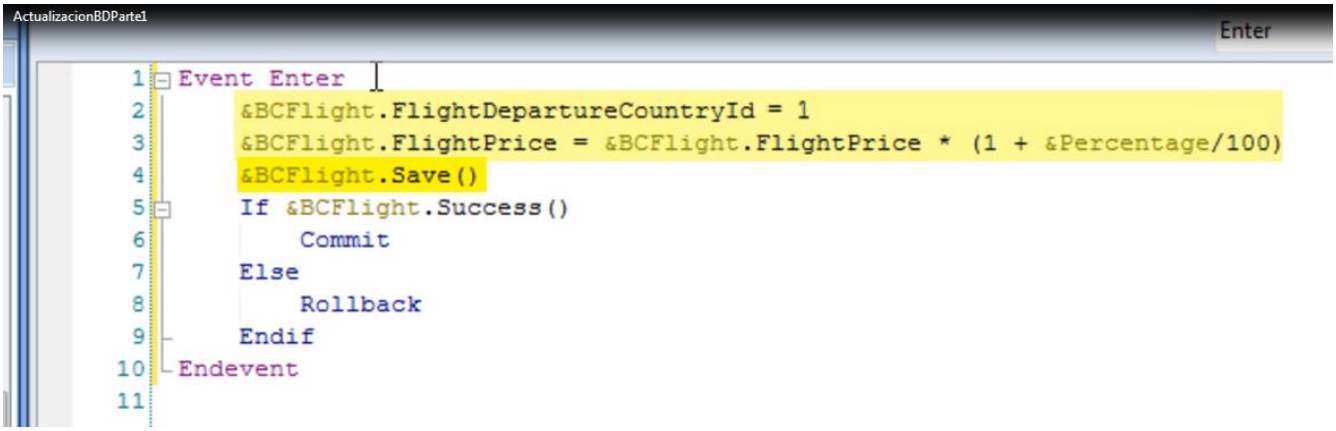
e fazemos Save…

GeneXus entende que é uma atualização

E se atribuímos valores ao atributos

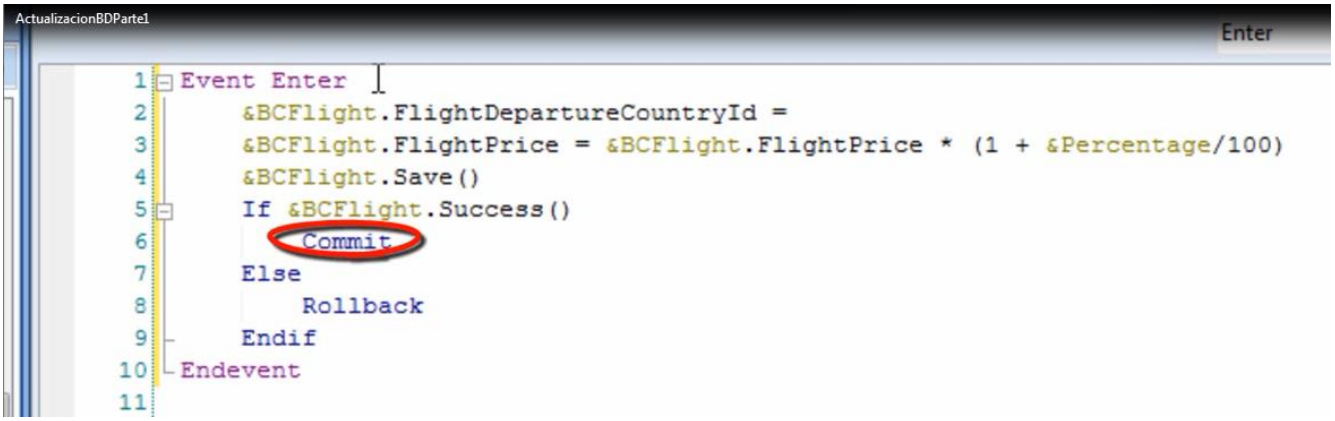


+ save



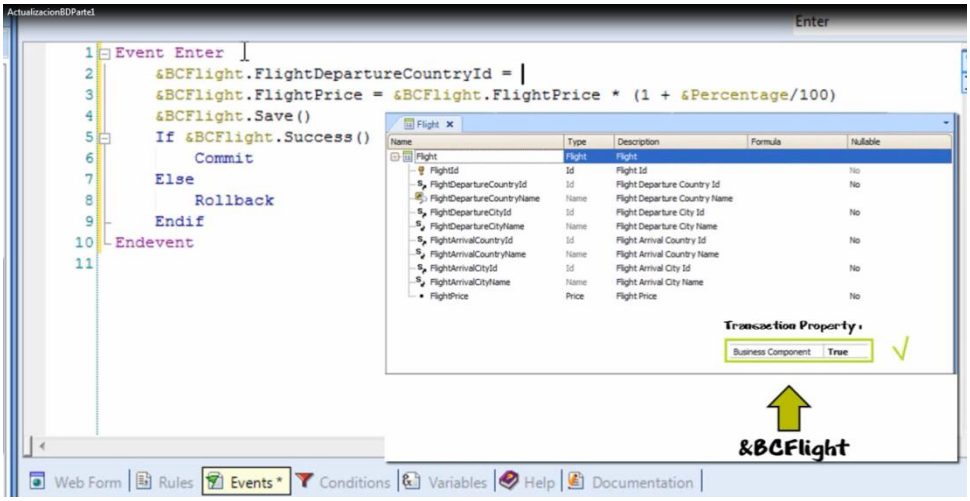
GeneXus entende que é uma inserção.

Não podemos nos esquecer do Commit.



Vimos, portanto, outra forma de atualizar o banco de dados, utilizando business componentes.

Como já dissemos, se uma transação é definida como business component



em qualquer objeto poderá ser definida uma variável desse tipo para atualizar o banco de dados a partir daí.

Nos falta ainda ver outra alternativa para atualizar o banco de dados com certos comandos. Veremos isso no próximo vídeo. Devemos saber que isso poderá ser usado somente em objetos procedures.