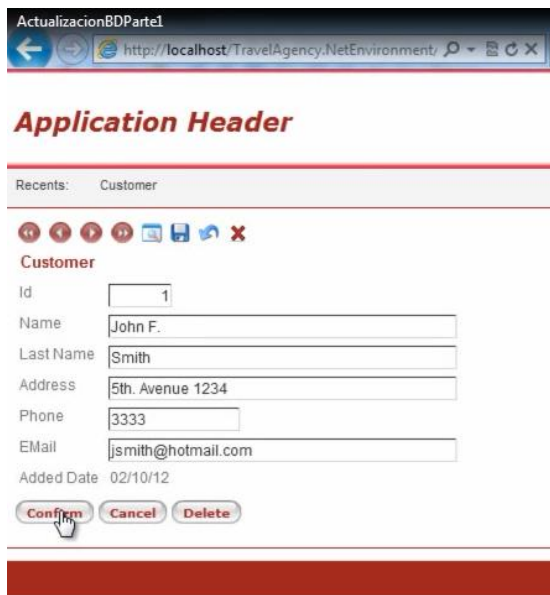


Actualización de la Base de Datos - Business Components

Hasta el momento hemos visto que las transacciones permiten a los usuarios, ingresar, modificar y eliminar datos, en las tablas de la base de datos de la aplicación.



The screenshot shows a web browser window with the address `http://localhost/TravelAgency.NetEnvironment/`. The page has a red header with the text "Application Header". Below the header, there is a "Recents:" section with the text "Customer". A toolbar with various icons is visible. The main form is titled "Customer" and contains the following fields:

- Id: 1
- Name: John F.
- Last Name: Smith
- Address: 5th. Avenue 1234
- Phone: 3333
- E-Mail: jsmith@hotmail.com
- Added Date: 02/10/12

At the bottom of the form, there are three buttons: "Confirm", "Cancel", and "Delete". A mouse cursor is pointing at the "Confirm" button.

Además de este medio para ir actualizando la base de datos, en ocasiones resulta necesario definir algún proceso de actualización automático.



The screenshot shows a web browser window with the address `http://localhost/TravelAgency.NetEnvironment/`. The page displays a table of flight prices. The table has three columns: "From", "To", and "Price". The data is as follows:

New York	Paris	\$1738
New York	London	\$770
New York	Rome	\$1976
New York	Athens	\$1640
New York	Moscow	\$1401
New York	Beijing	\$1496
New York	Tokyo	\$1057
New York	Panama City	\$449
New York	Sao Paulo	\$2209
New York	Montevideo	\$2215

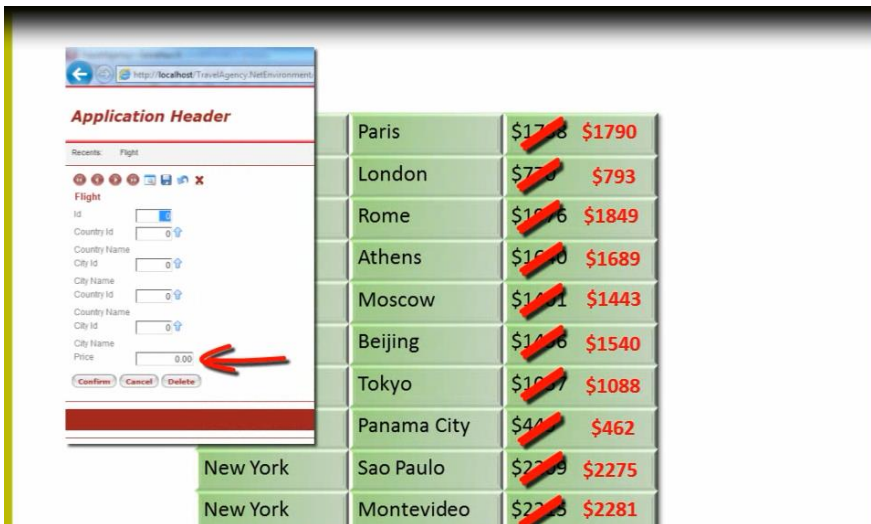
A modo de ejemplo, la agencia de viajes podría necesitar cada determinado tiempo, aumentar los precios de **todos los vuelos registrados**,

New York	Paris	\$1738	
New York	London	\$770	
New York	Rome	\$1976	
New York	Athens	\$1640	
New York	Moscow	\$1401	
New York	Beijing	\$1496	
New York	Tokyo	\$1057	
New York	Panama City	\$449	
New York	Sao Paulo	\$2209	
New York	Montevideo	\$2215	

aplicándole a los precios vigentes, determinado porcentaje de aumento.

New York	Paris	\$1738	\$1790
New York	London	\$770	\$793
New York	Rome	\$1976	\$1849
New York	Athens	\$1640	\$1689
New York	Moscow	\$1401	\$1443
New York	Beijing	\$1496	\$1540
New York	Tokyo	\$1057	\$1088
New York	Panama City	\$449	\$462
New York	Sao Paulo	\$2209	\$2275
New York	Montevideo	\$2215	\$2281

Resultaría muy tedioso utilizar la pantalla de la transacción para editar manualmente cada precio de los cientos de vuelos registrados,

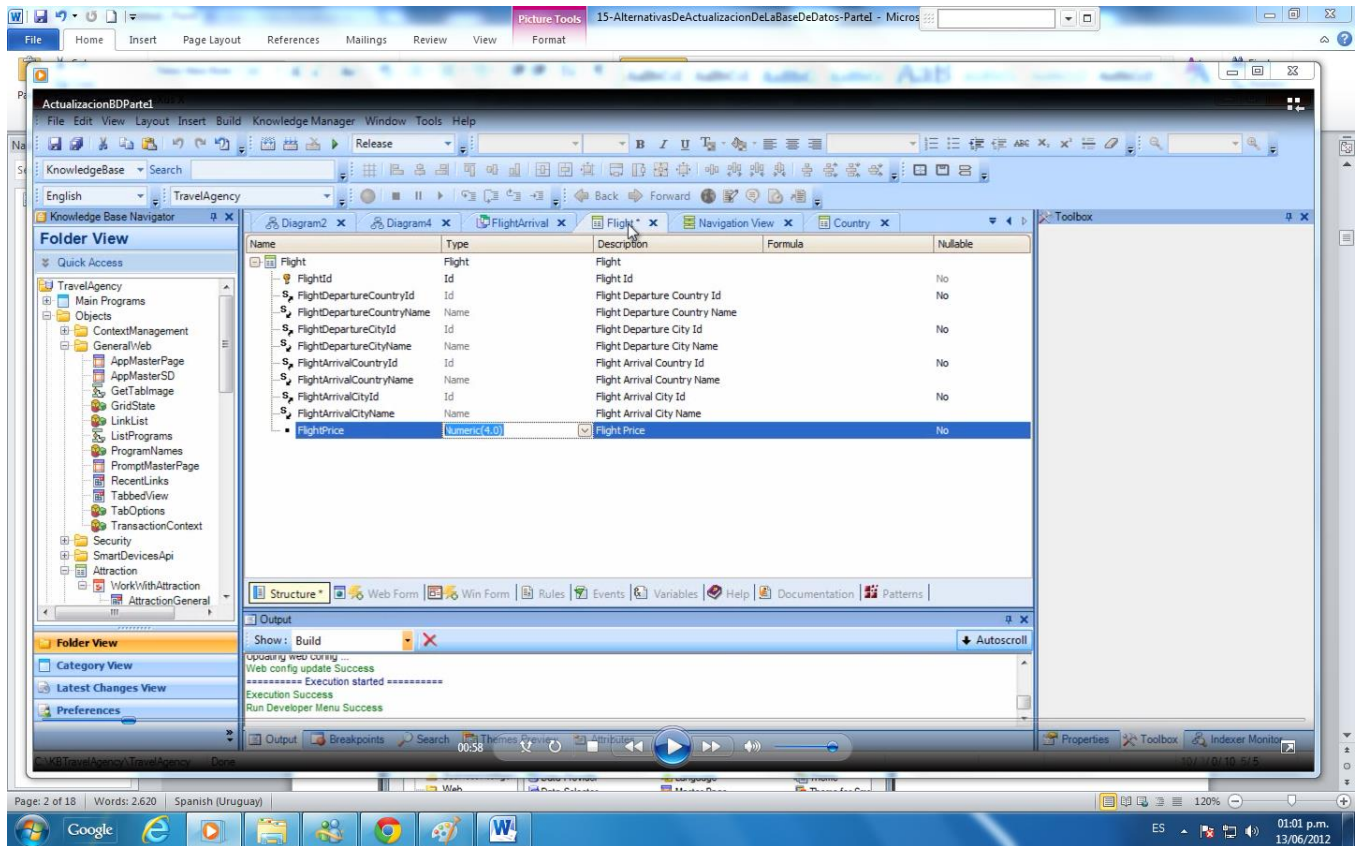


por lo que nos interesa que esto lo haga un proceso masivo.

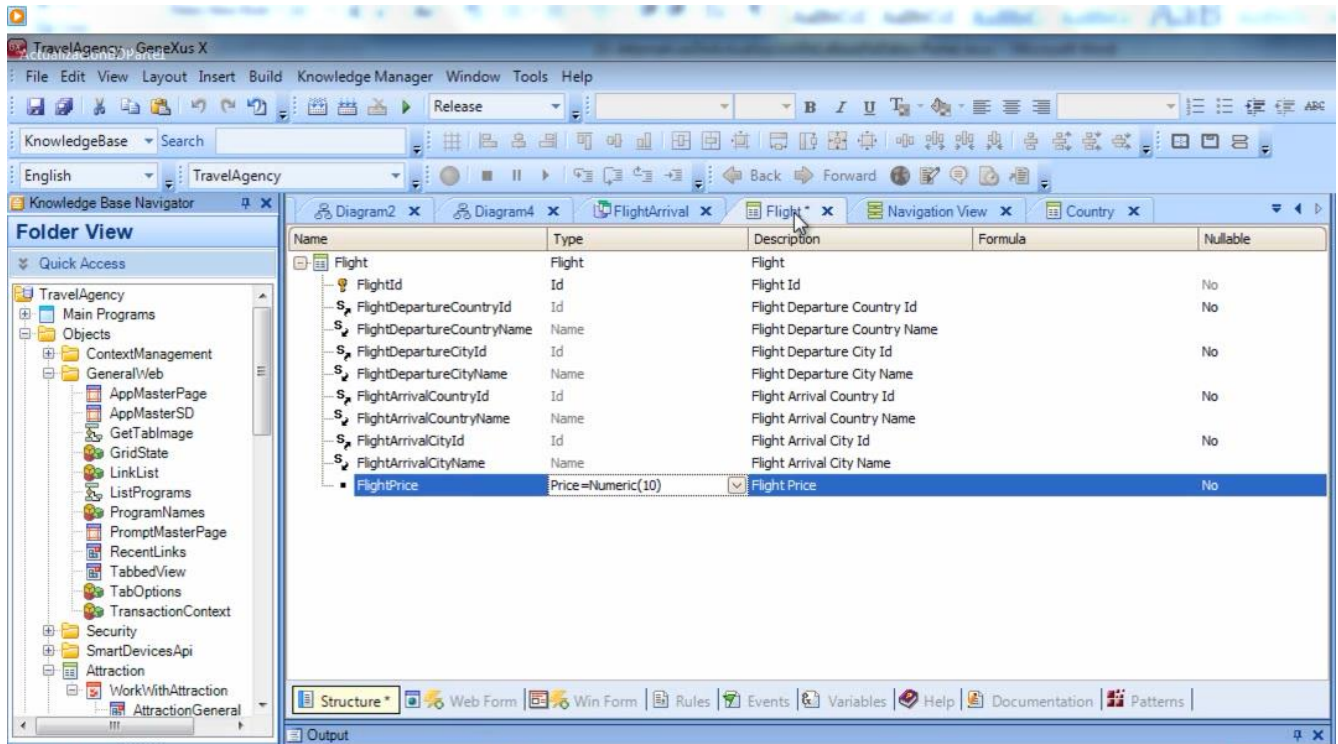


Para resolver este ejemplo que planteamos, lo primero que vamos a hacer es agregar a la transacción Flight, un atributo que almacene el precio de cada vuelo.

Creamos el atributo FlightPrice,

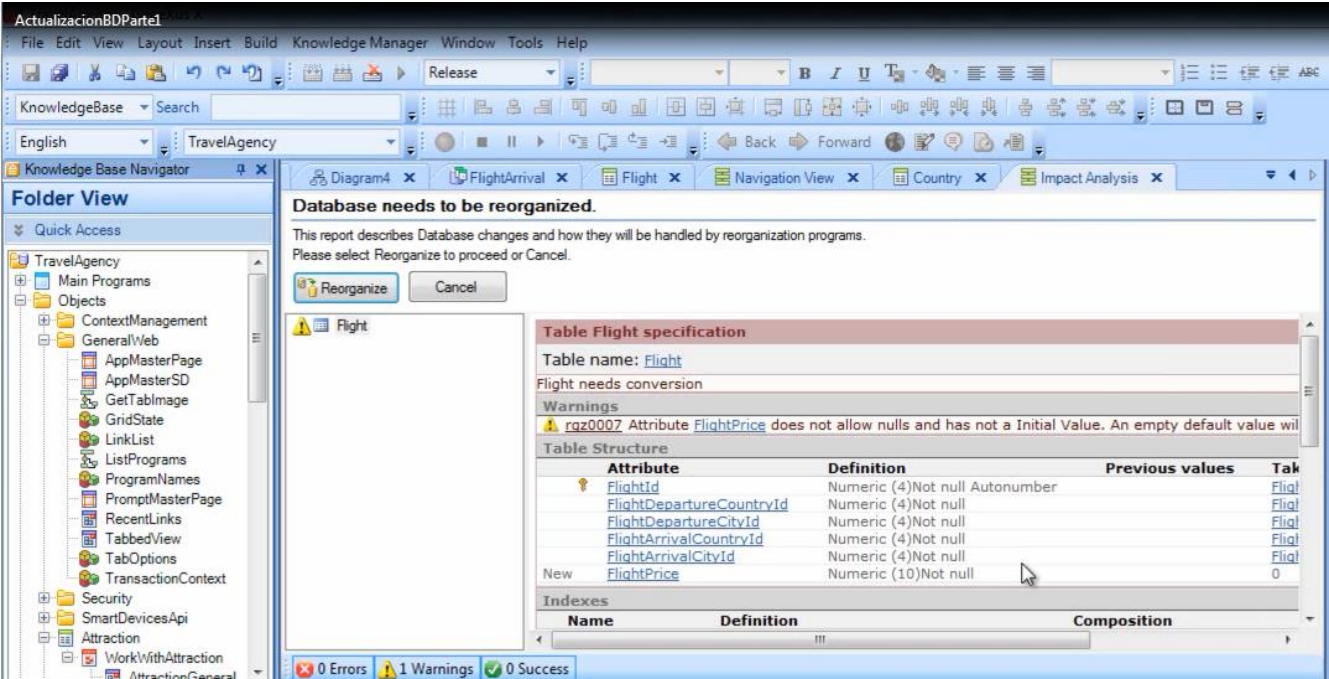


definimos un dominio de nombre Price, como numérico de 10



salvamos y presionamos F5.

GeneXus nos avisa que agregará este campo en la tabla FLIGHT,



procedemos y a continuación sigue la generación de los programas.

Vamos a ingresar algunos precios para los vuelos que teníamos registrados

ActualizacionBDParte1

http://localhost/TravelAgency.NetEnvironment/ Flight

Application Header

Recents: Flight

Flight

Id: 1

Country Id: 1

Country Name: Brazil

City Id: 1

City Name: Rio de Janeiro

Country Id: 2

Country Name: France

City Id: 1

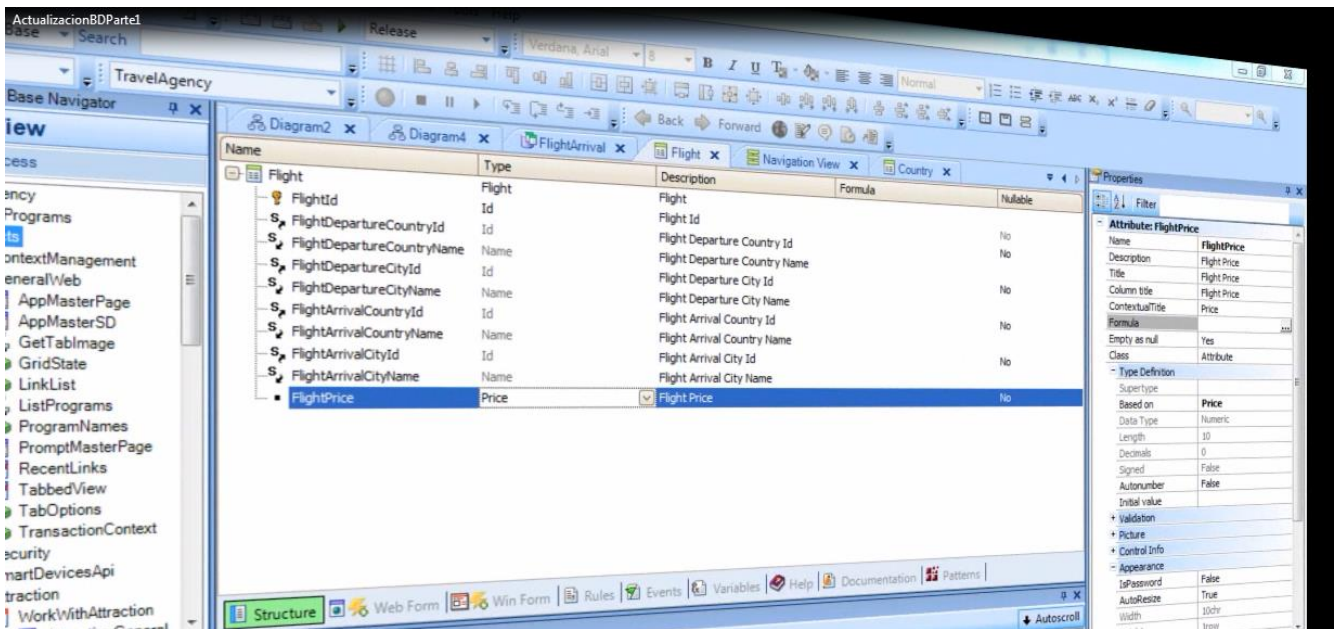
City Name: Paris

Price: 2000

Confirm Cancel Delete

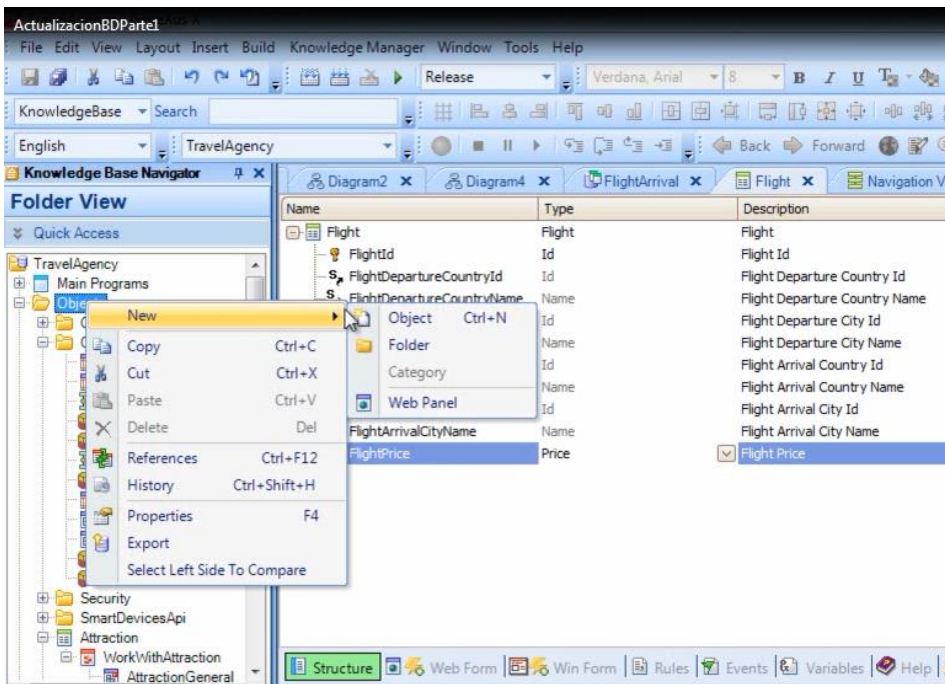
Footer Info

Y ahora sí, veremos cómo resolver el requisito de aumentar los precios de forma automática y masiva.



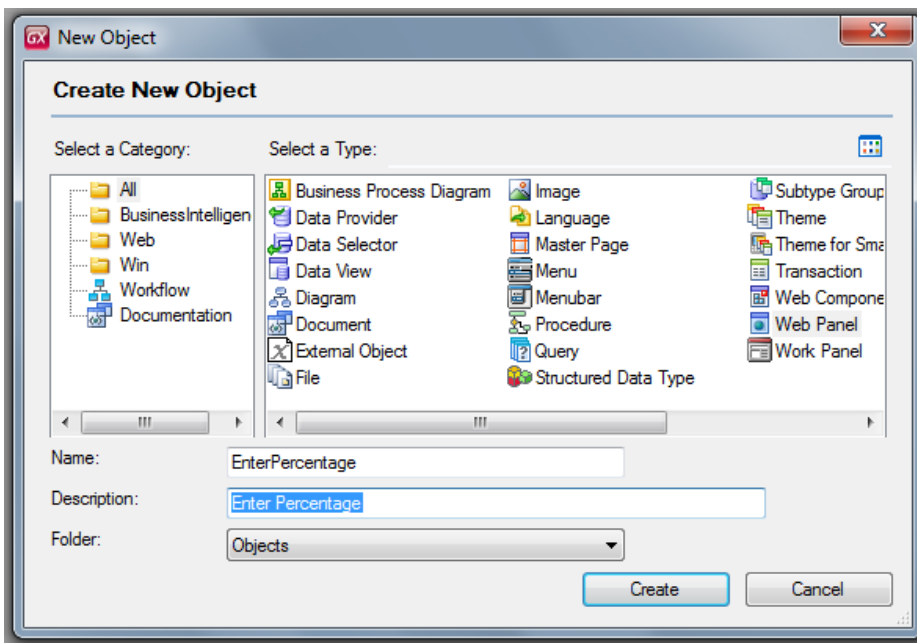
Lo primero que debemos hacer, es solicitarle al usuario el porcentaje de aumento que desea aplicar a los precios de los vuelos. Para eso, vamos a crear un objeto de tipo web panel.

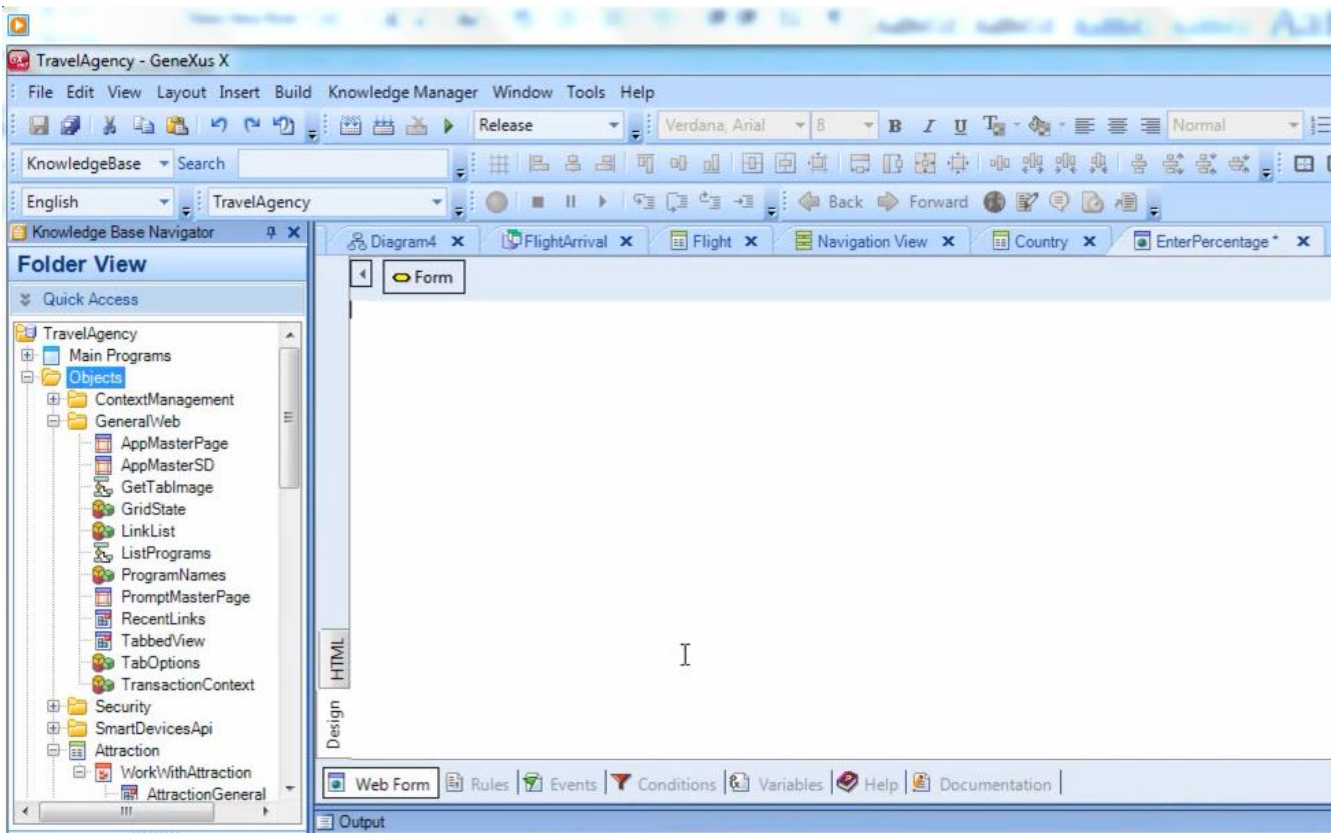
Seleccionamos new / object



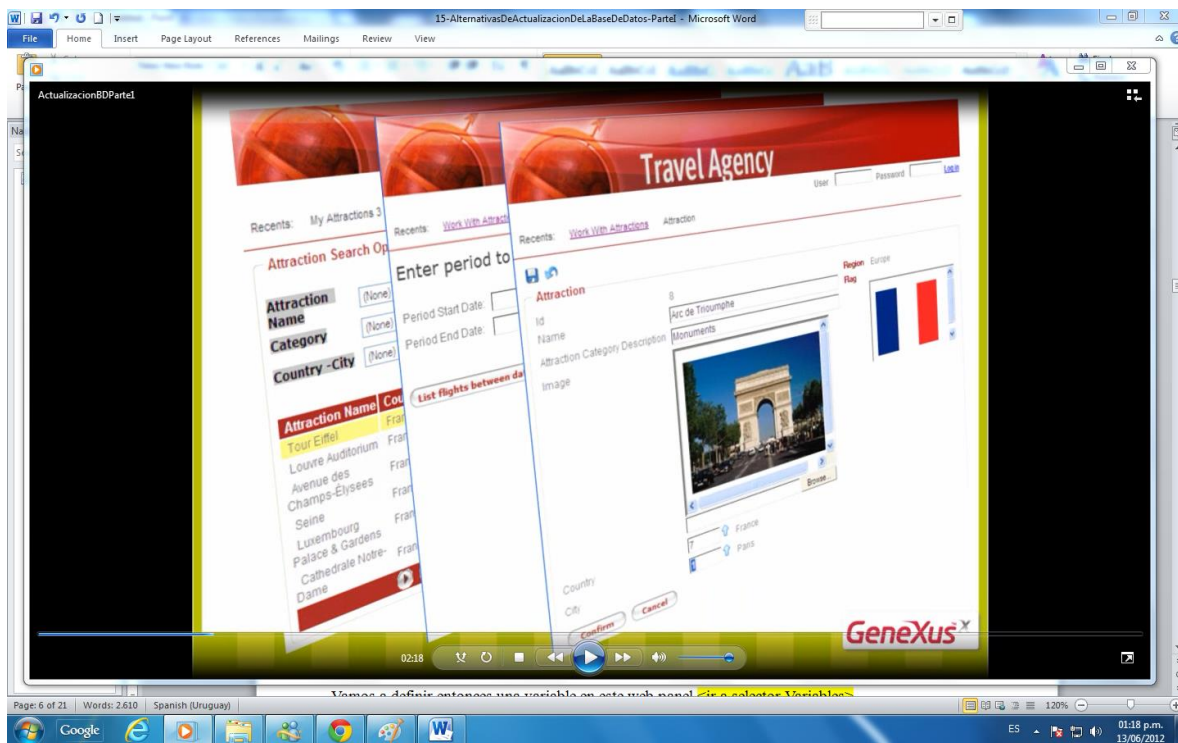
web panel

y le damos el nombre "EnterPercentage"

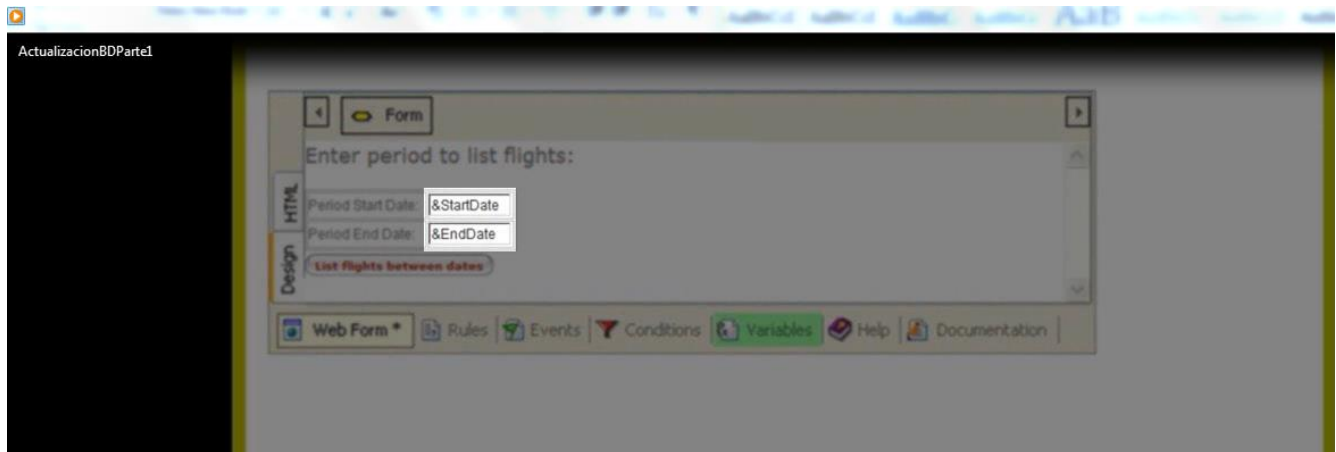




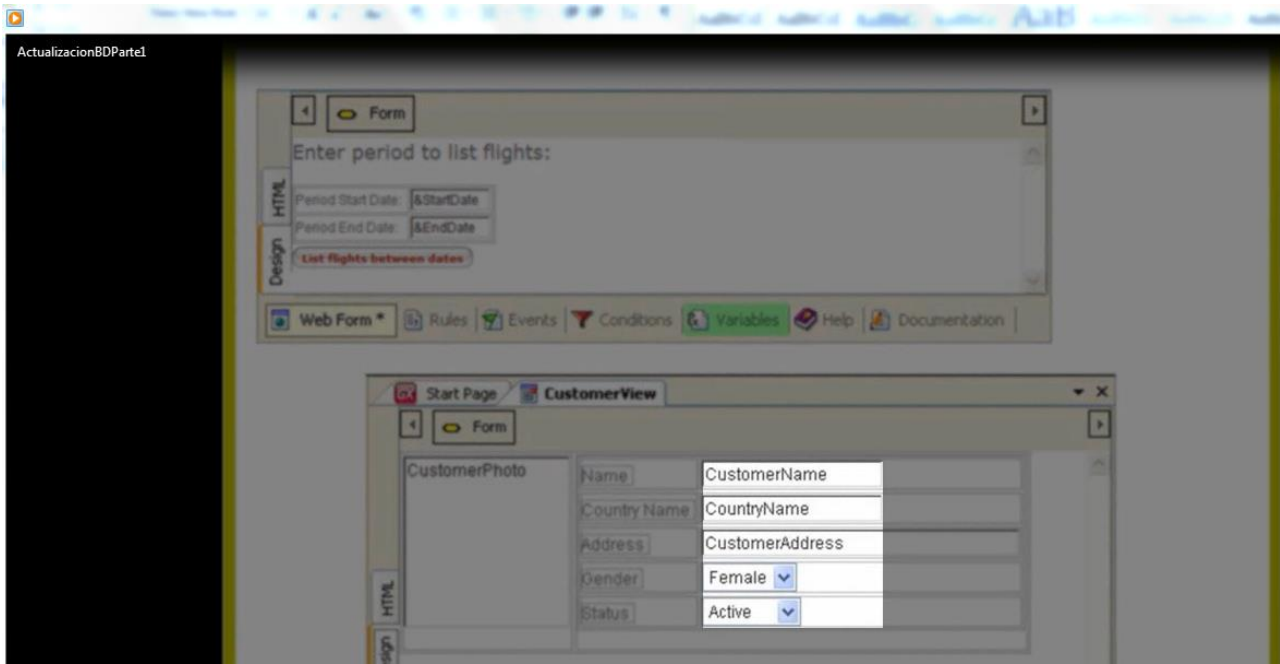
El objeto web panel, es básicamente una página web que nos permite resolver variadas funcionalidades, como solicitarle datos al usuario, o también brindarle consultas muy completas funcionalmente.



Si incluimos variables en el form de un web panel, las mismas tendrán comportamiento “de entrada de datos”, es decir que el usuario podrá ingresarles valor.



Si en cambio incluimos atributos en el form de un web panel,

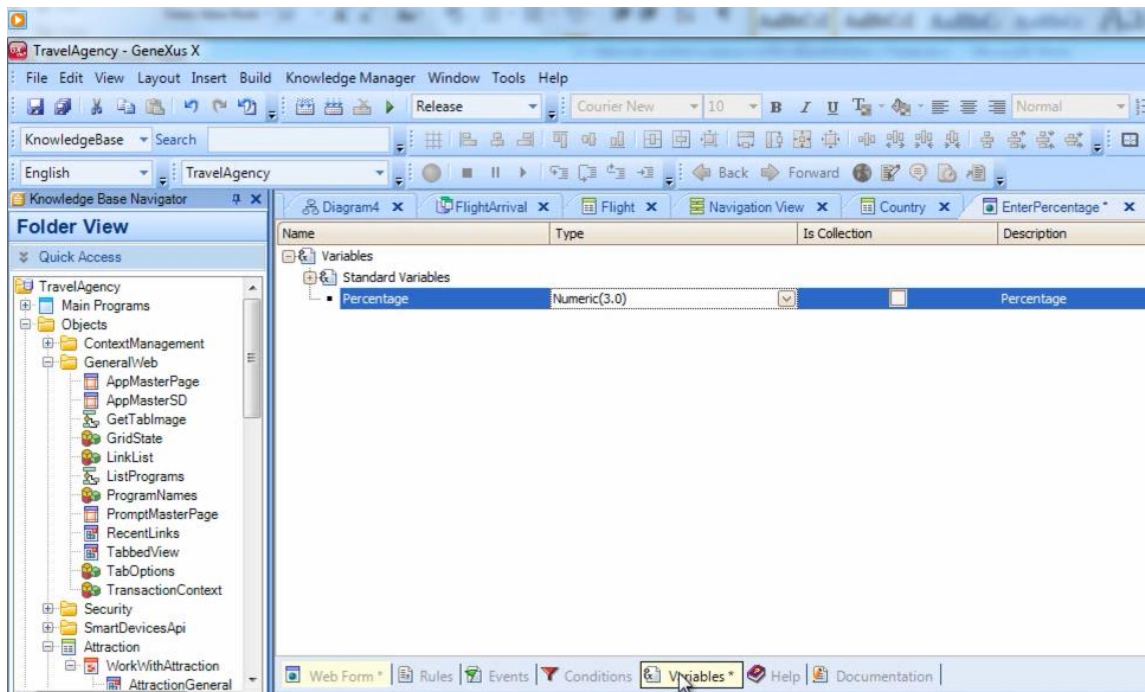


los mismos serán “de salida”, o sea que no serán editables sino que solamente se mostrarán los datos que dichos atributos tengan almacenados en la base de datos.

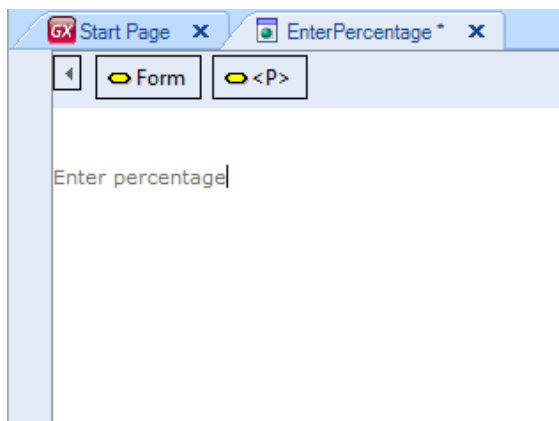


Vamos a definir entonces una variable en este web panel.

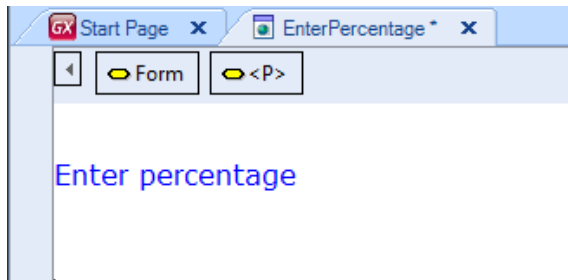
La creamos con el nombre “Percentage” y la definimos del tipo Numeric(3):



Ahora volvemos al form y digitamos directamente dentro del mismo: “Enter percentage”

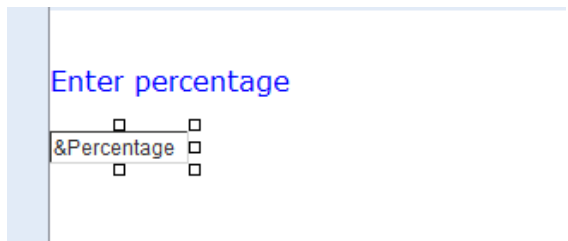


Podemos cambiar el formato de este texto

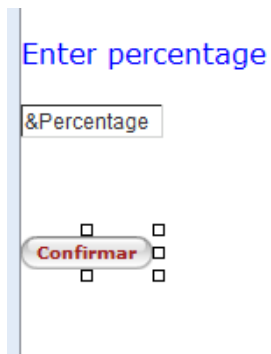


Presionamos enter y debajo del texto insertaremos la variable &percentage

Seleccionamos Insert / Variable , elegimos a la variable &percentage:

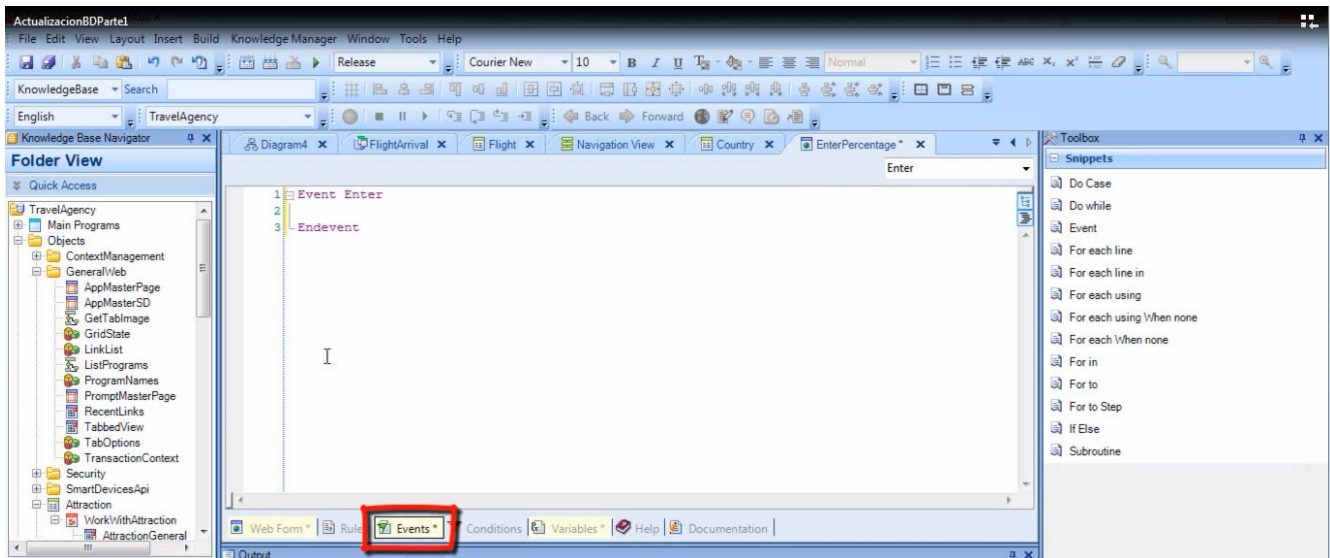


Y ahora arrastramos un botón al form desde la toolbox



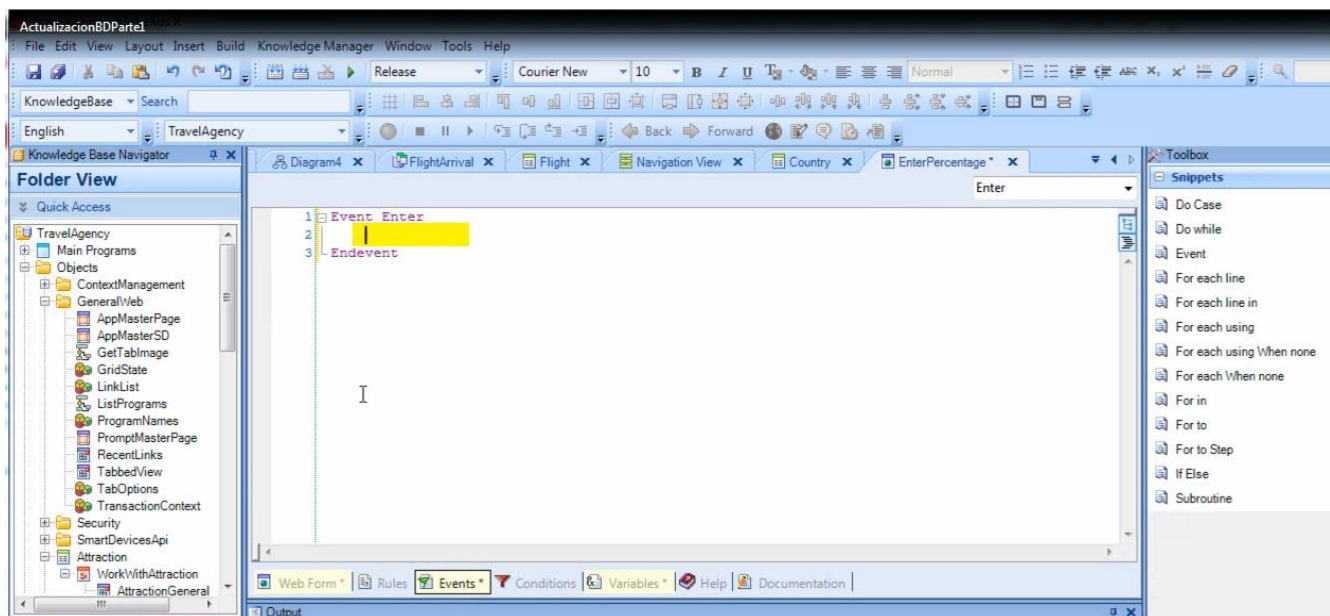
Como se puede percibir, la página nos ha quedado pronta para que el usuario ingrese el porcentaje de aumento que desee y presione el botón para ejecutar un proceso automático que resuelva el aumento de los precios de todos los vuelos.

Ahora bien, si hacemos doble clic en el botón, vemos que nos lleva a la sección de eventos del web panel



y en particular el cursor se posiciona dentro del evento asociado al botón: el evento Enter.

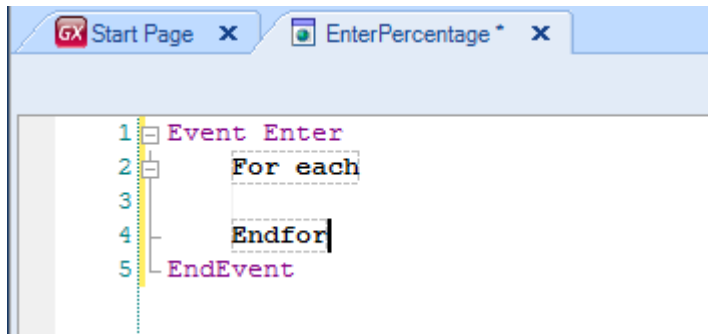
Aquí,



codificaremos las instrucciones que queremos ejecutar cuando el usuario presione el botón

Recordemos qué es lo que tenemos que hacer: debemos navegar todos los vuelos almacenados y para cada uno actualizar su precio.

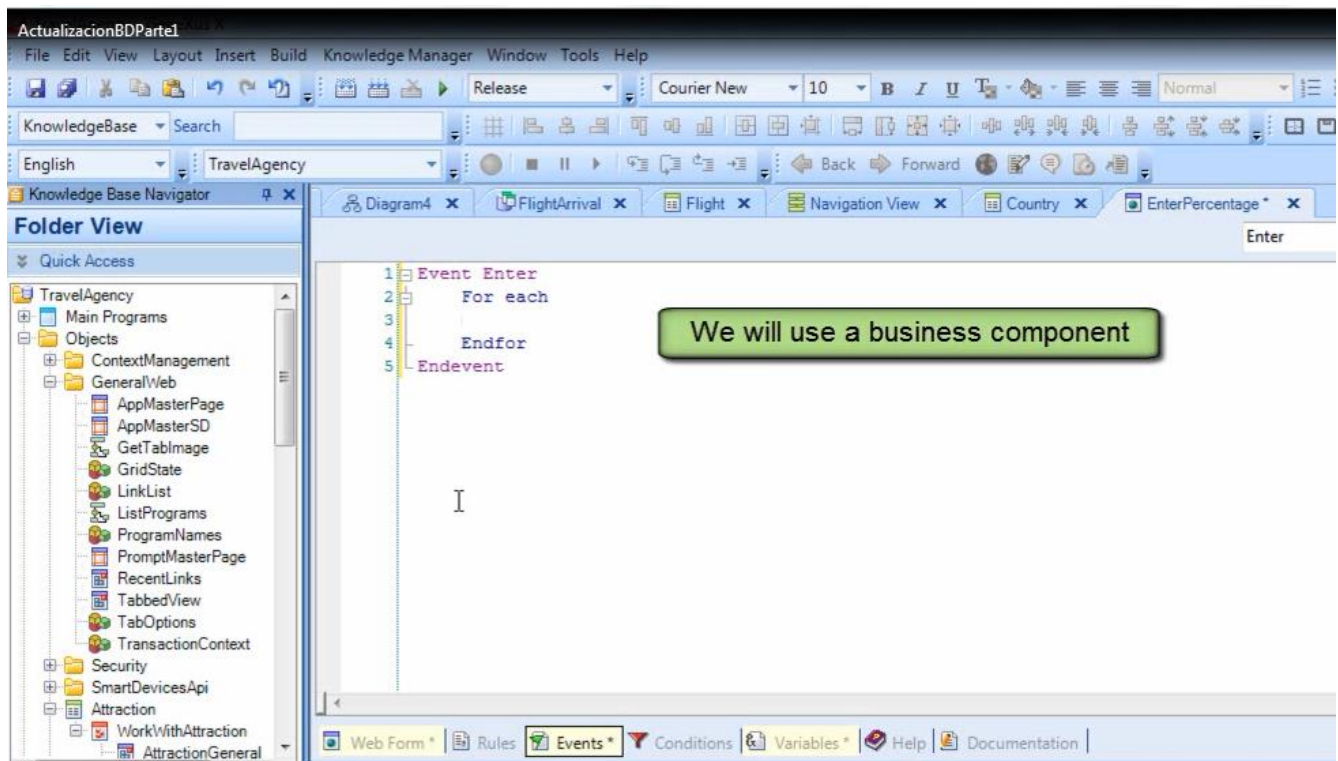
Vamos a escribir entonces, un comando For each con su correspondiente Endfor, para navegar la tabla FLIGHT



Un For each siempre tiene que contener al menos 1 atributo en su cuerpo, para que GeneXus pueda determinar la tabla base a ser navegada por el For each.

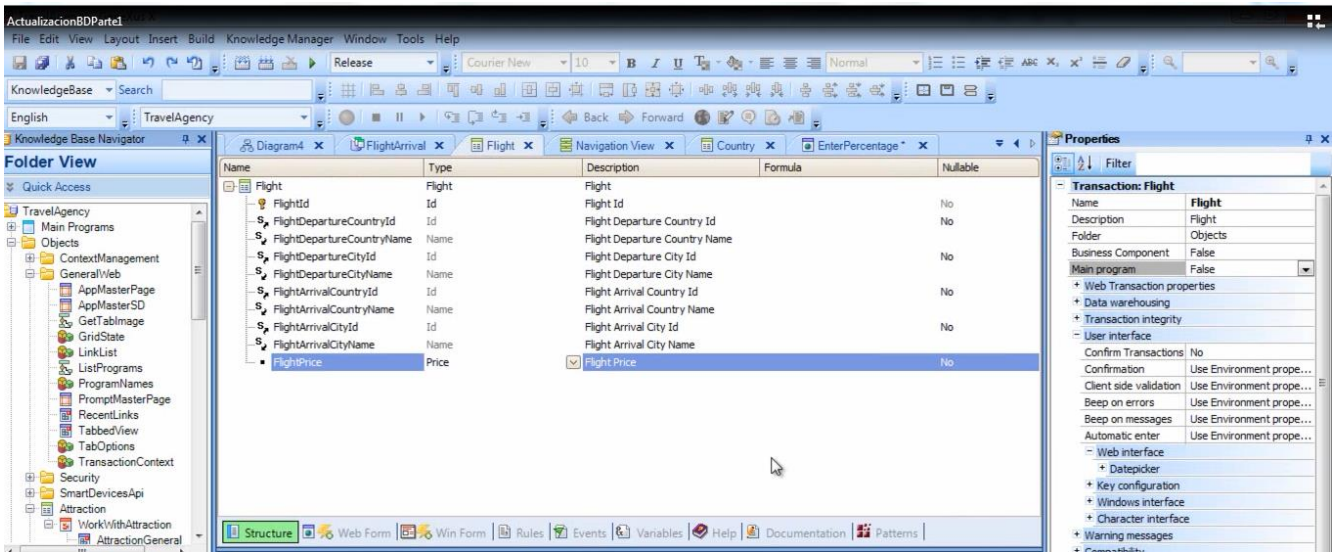
¿Qué atributo podremos referenciar? ¿y cómo hacemos para actualizar el precio de cada vuelo?

Para actualizar la base de datos en un web panel, solamente contamos con una posibilidad... y es empleando el concepto de business component

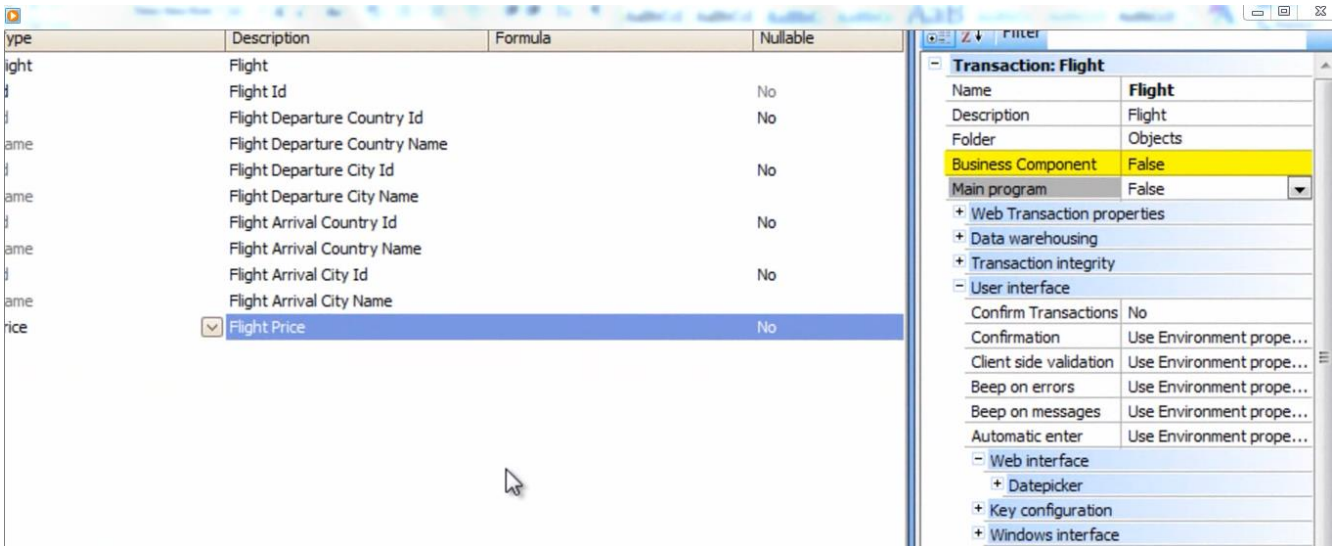


Veamos de qué se trata un business component y cómo se usa.

Dado que queremos actualizar datos de vuelos, vamos a abrir la transacción Flight y observemos las propiedades de la transacción.



Vemos que hay una propiedad de nombre Business Component

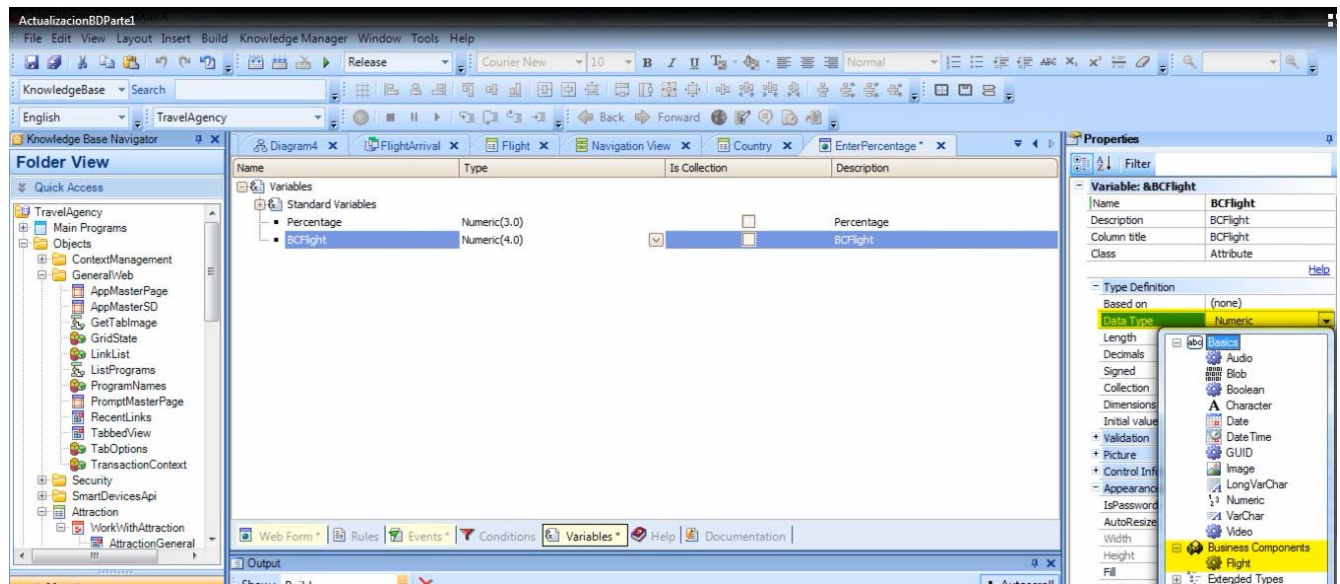


y la configuraremos con valor True:

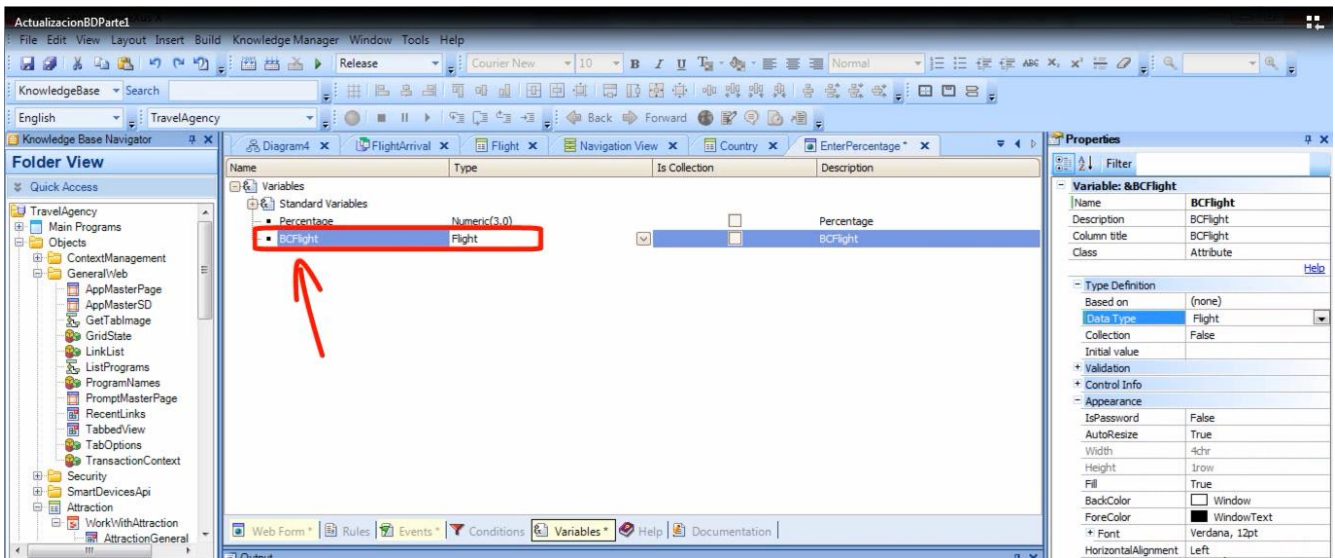


Todas las transacciones tienen la propiedad Business Component. O sea que a cualquier transacción de la base de conocimiento le podemos configurar su propiedad Business Component con valor True.

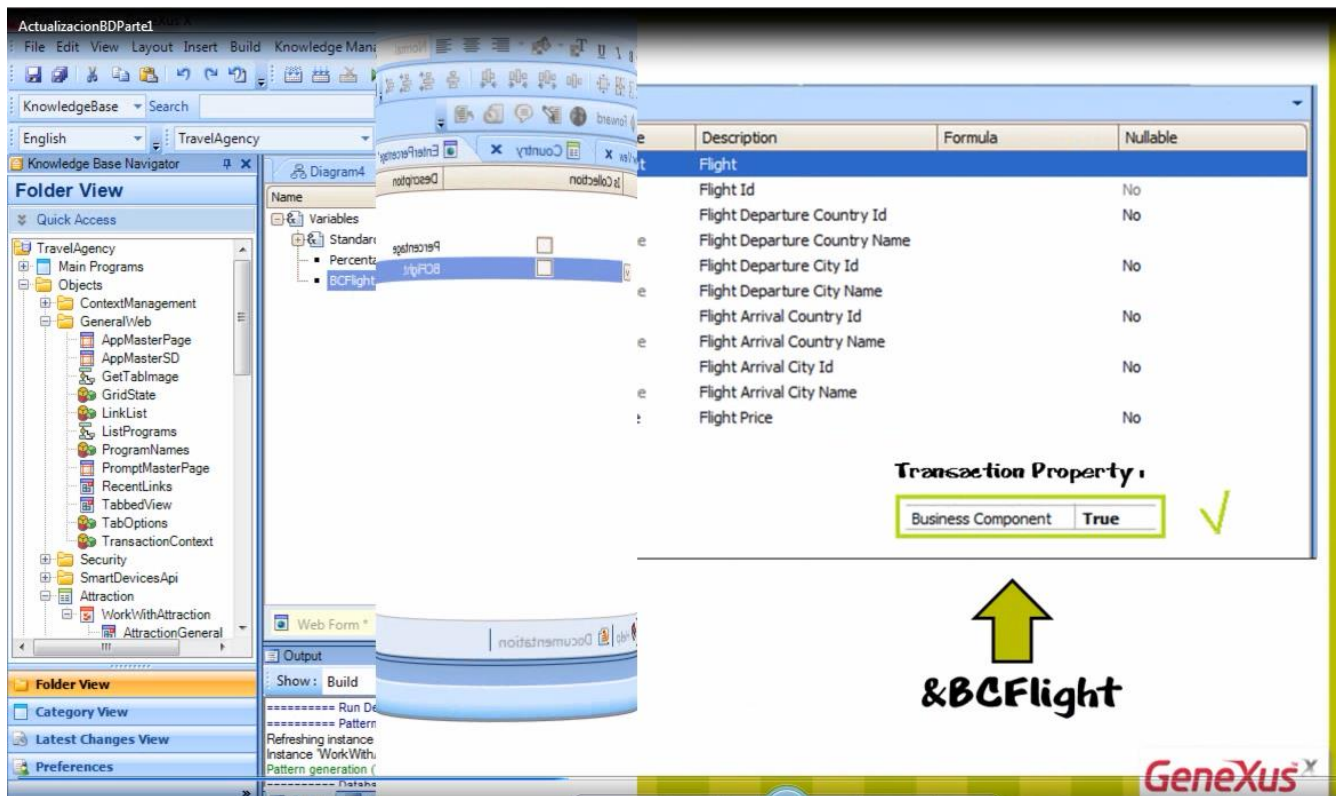
Cuando a una transacción le configuramos su propiedad Business Component con valor True, en cualquier objeto de la base de conocimiento, podremos definir una variable del tipo business component de la transacción definida como tal...



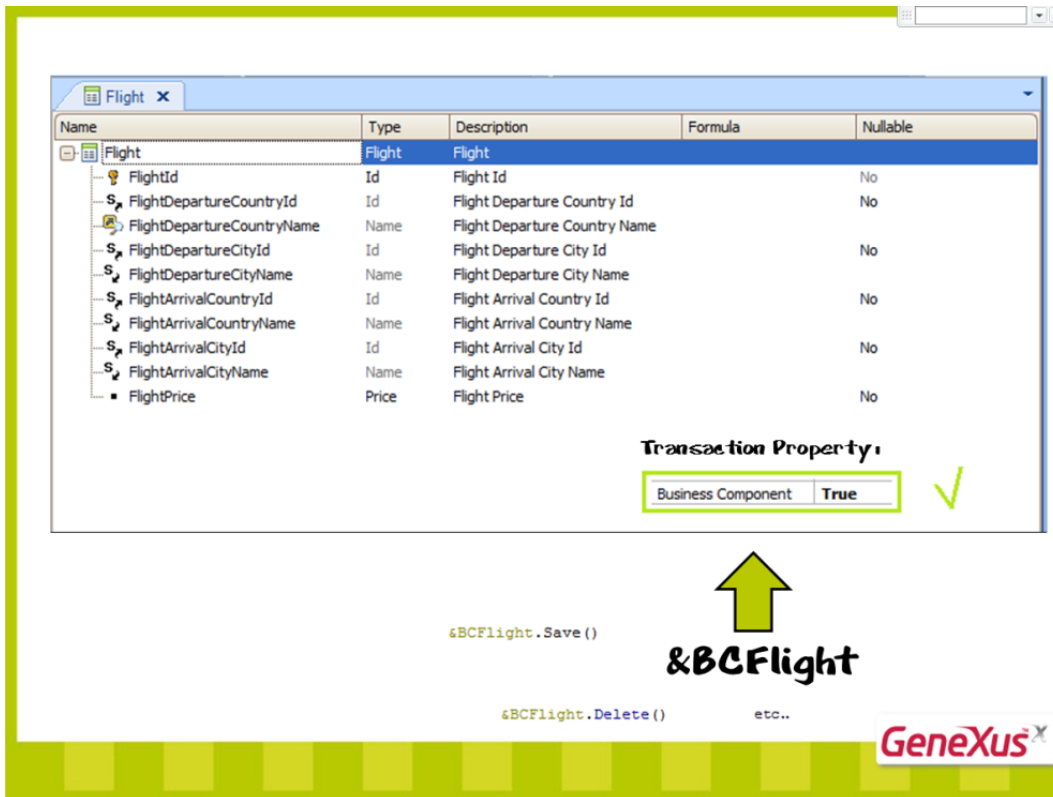
Empleando esta variable



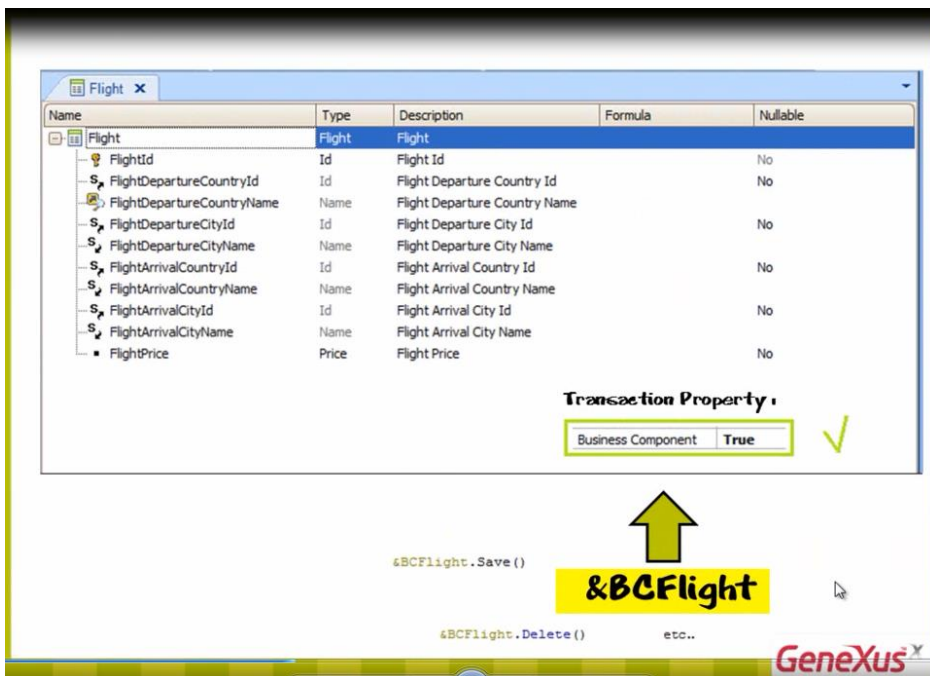
podremos ejecutar a la transacción Flight sin su form



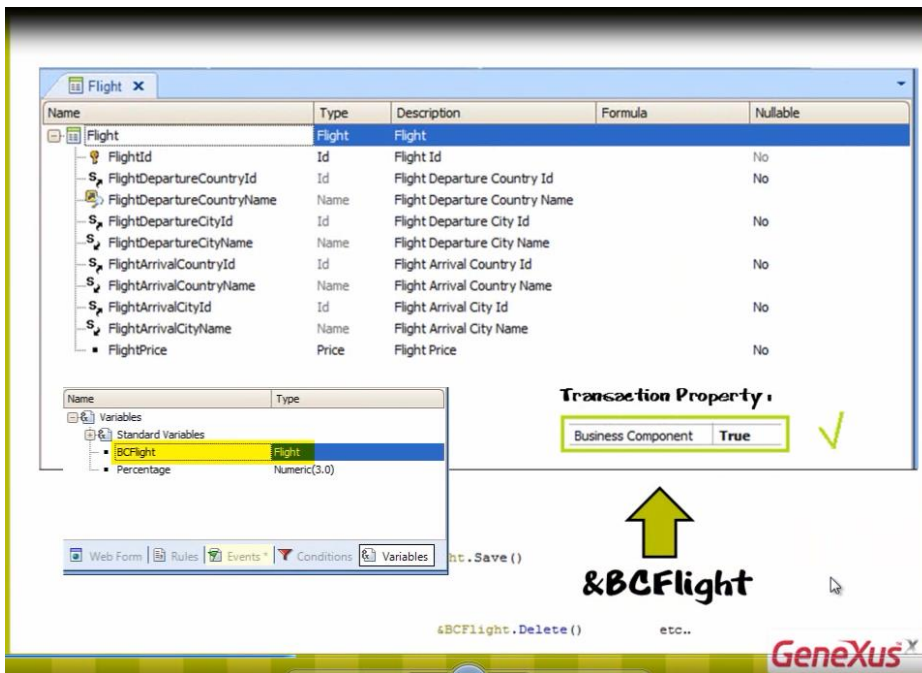
para realizar actualizaciones de la base de datos:



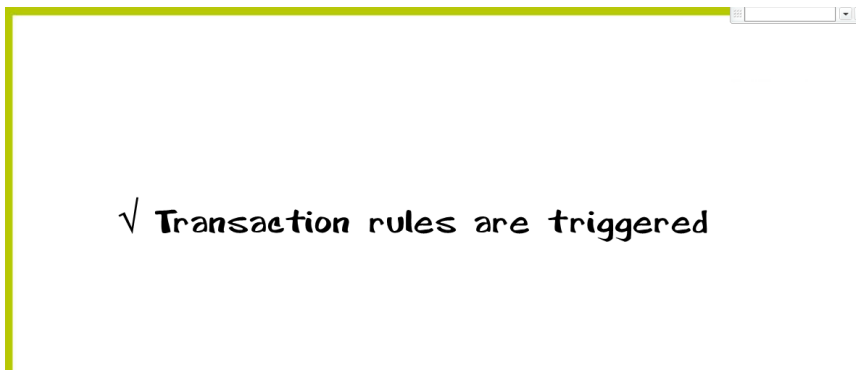
Además, cuando utilizemos esta variable



definida del tipo business component de una transacción



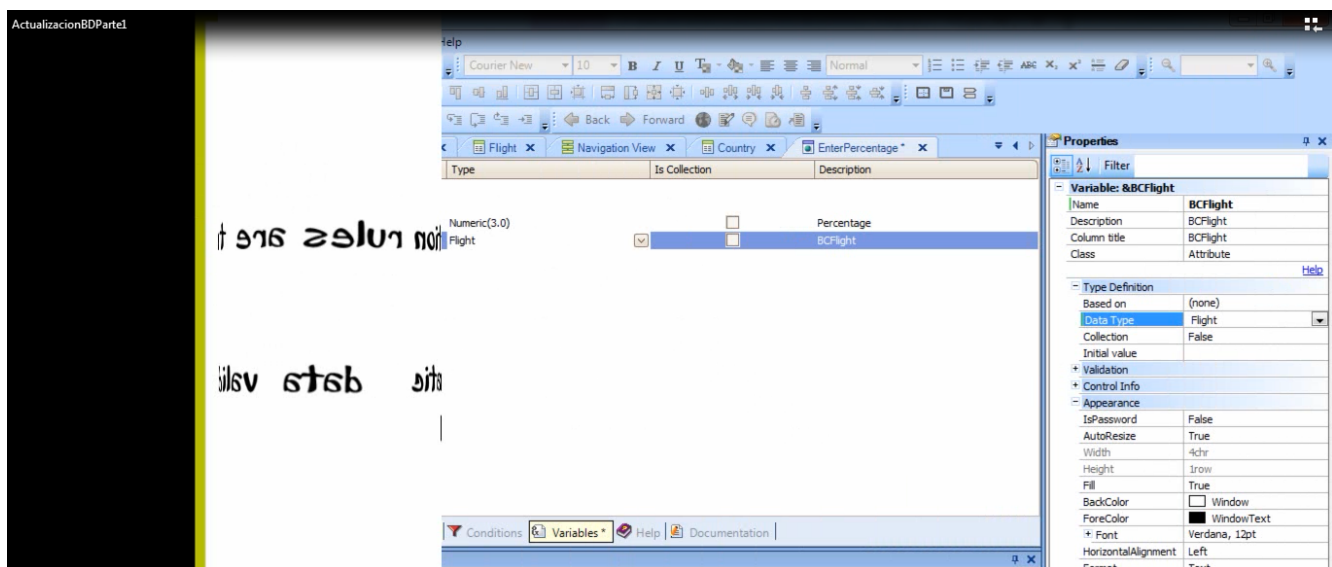
se dispararán las reglas que hayamos definido en la transacción



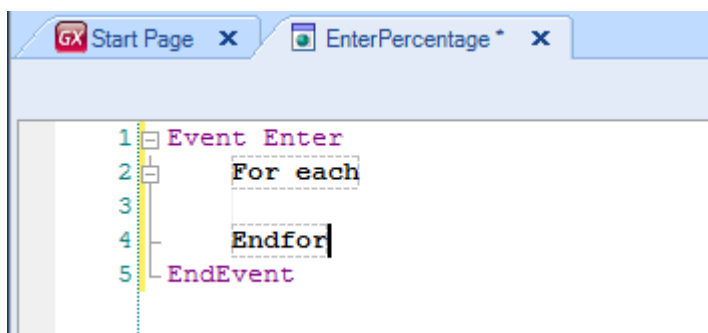
y también se ejecutarán todos los controles automáticos que ofrece la transacción para validar que los datos almacenados sean consistentes

- ✓ Transaction rules are triggered
- ✓ Automatic validations are performed

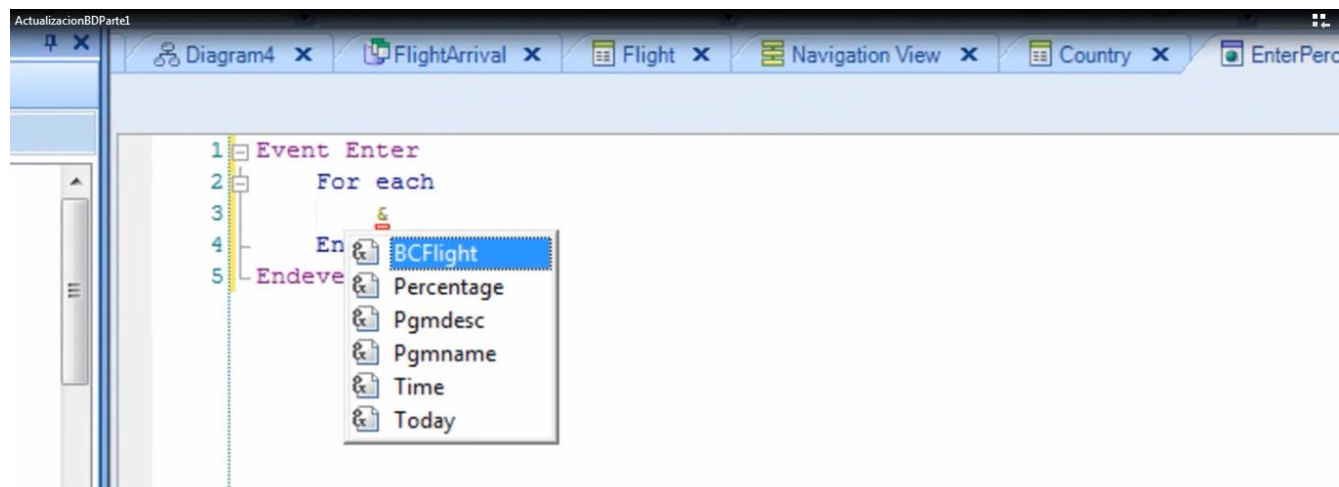
Ahora bien, volvamos al objeto que estabamos codificando, para aprender a utilizar variables del tipo business components



En el web panel “EnterPercentage”, dentro del cuerpo del For each que estbamos codificando, vamos a escribir la primer instruccin.

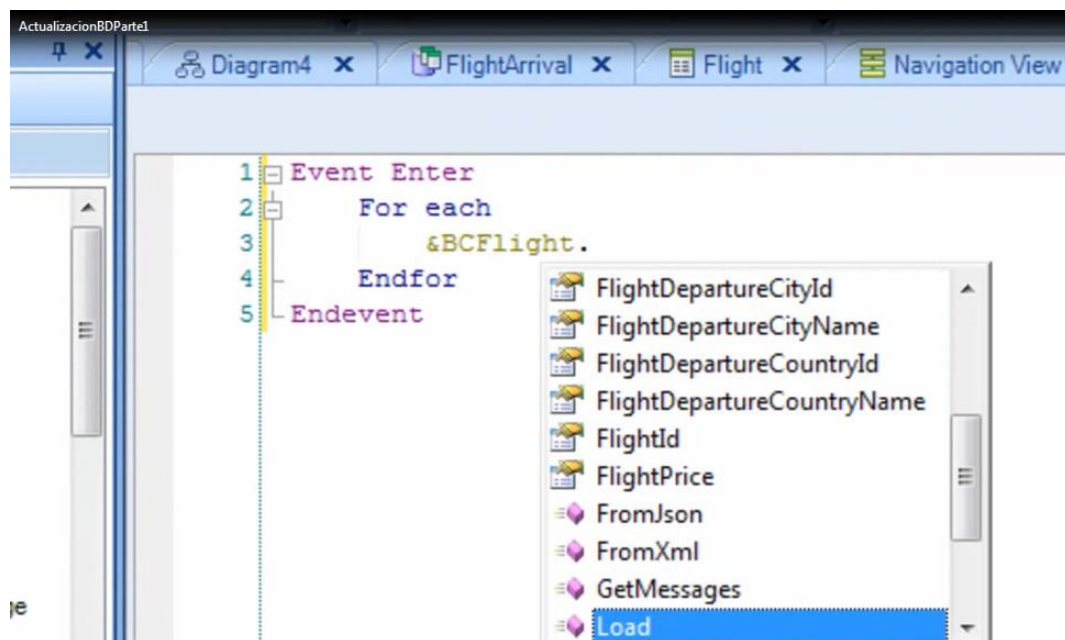


Escribimos ampersand

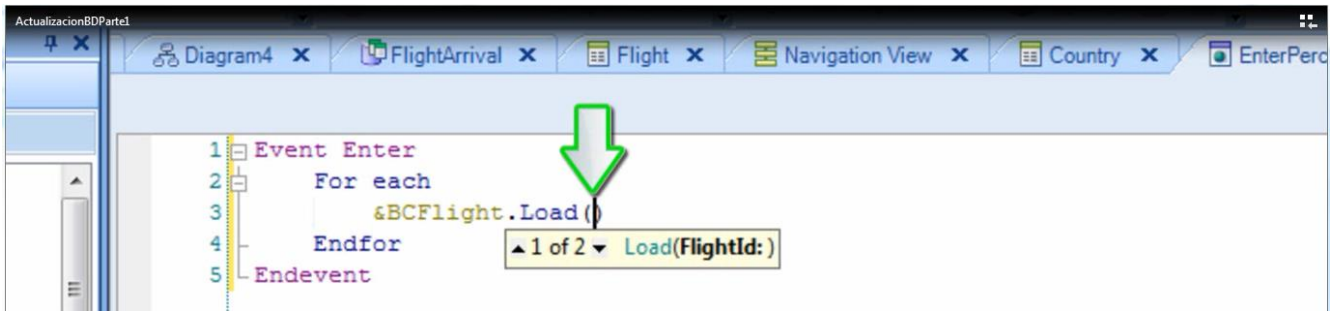


y seleccionamos la variable &BCFlight que hemos definido.

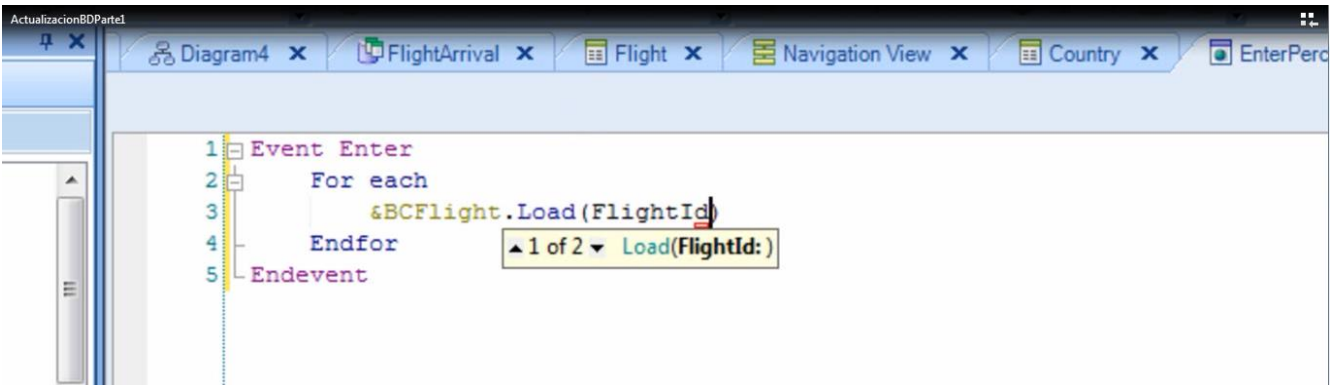
Ahora digitamos punto y elegimos el método Load.



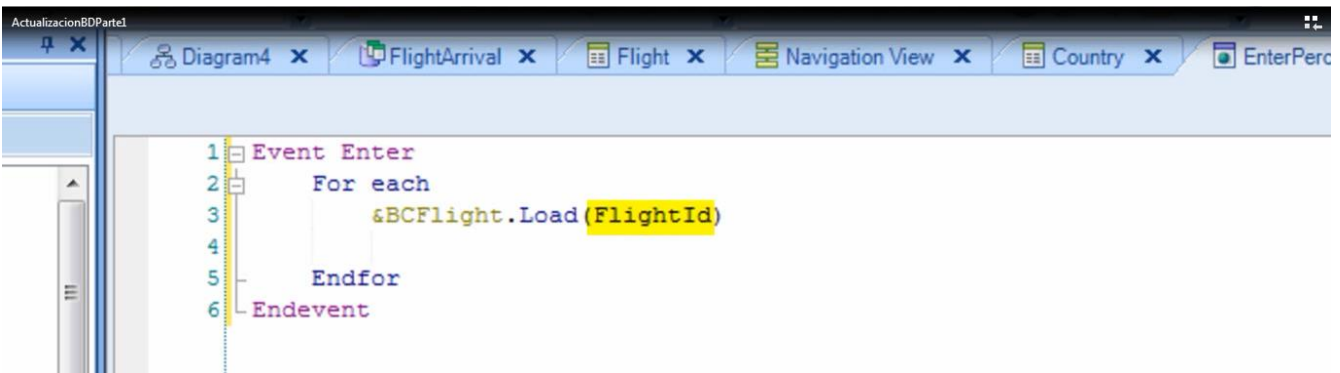
El método Load, tal como su nombre lo describe, permite **cargar** en memoria, los datos correspondientes al valor de llave primaria que indiquemos dentro del paréntesis.



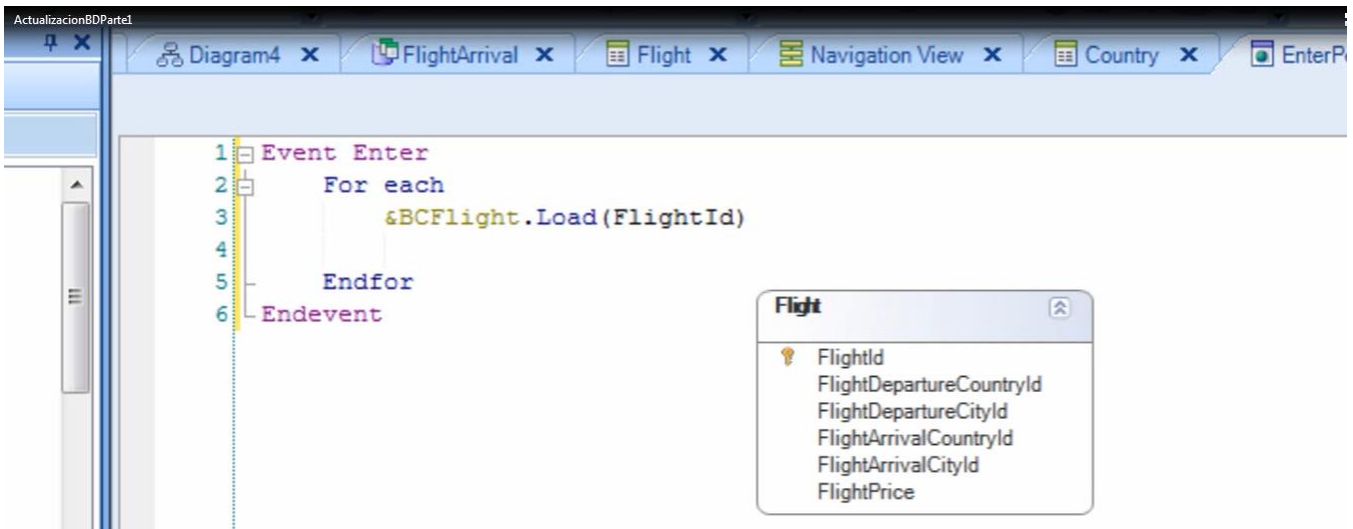
El valor de llave primaria que se incluya dentro del mismo, deberá ser **un valor válido para la llave primaria de la transacción que es business component**.



En este caso, estamos cargando en memoria un vuelo, y el valor de la llave primaria lo tenemos en el atributo FlightId

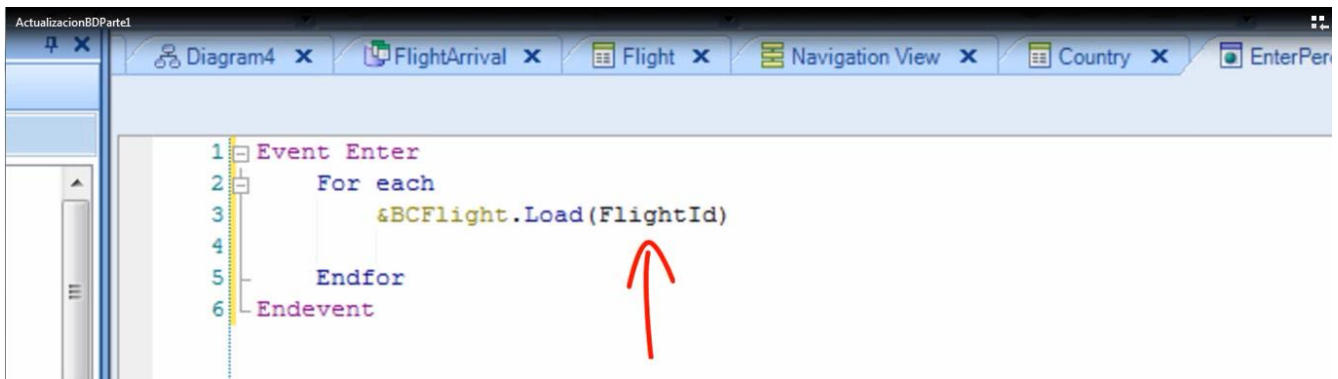


ya que nuestro objetivo es navegar la tabla FLIGHT mediante el For each

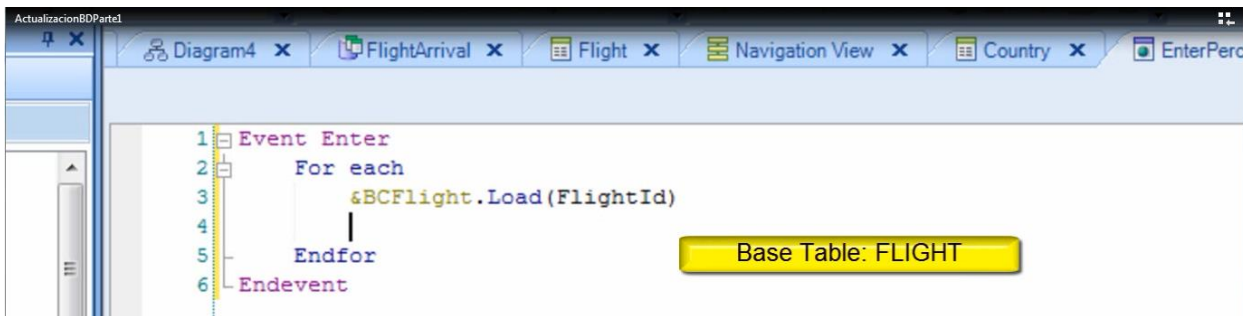


y para cada vuelo navegado, tendremos los valores de sus atributos.

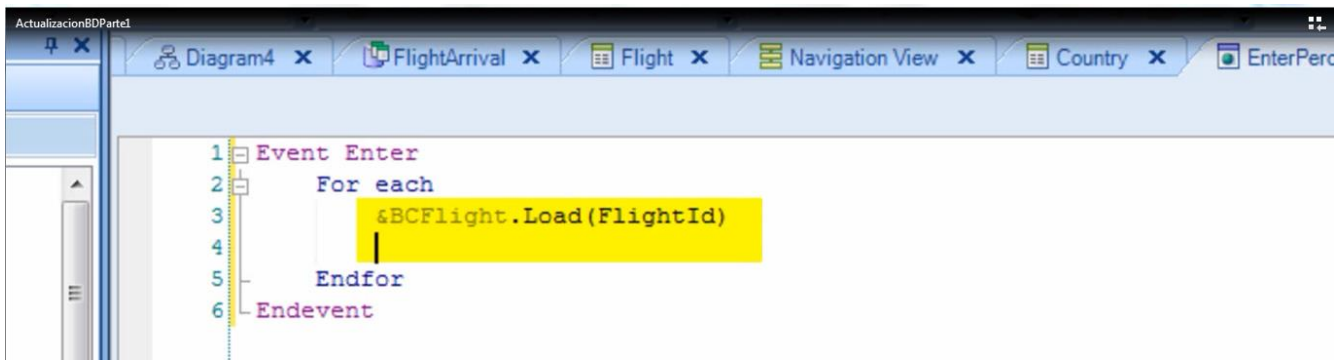
A propósito de que nuestro objetivo es navegar la tabla FLIGHT, observemos que este atributo FlightId



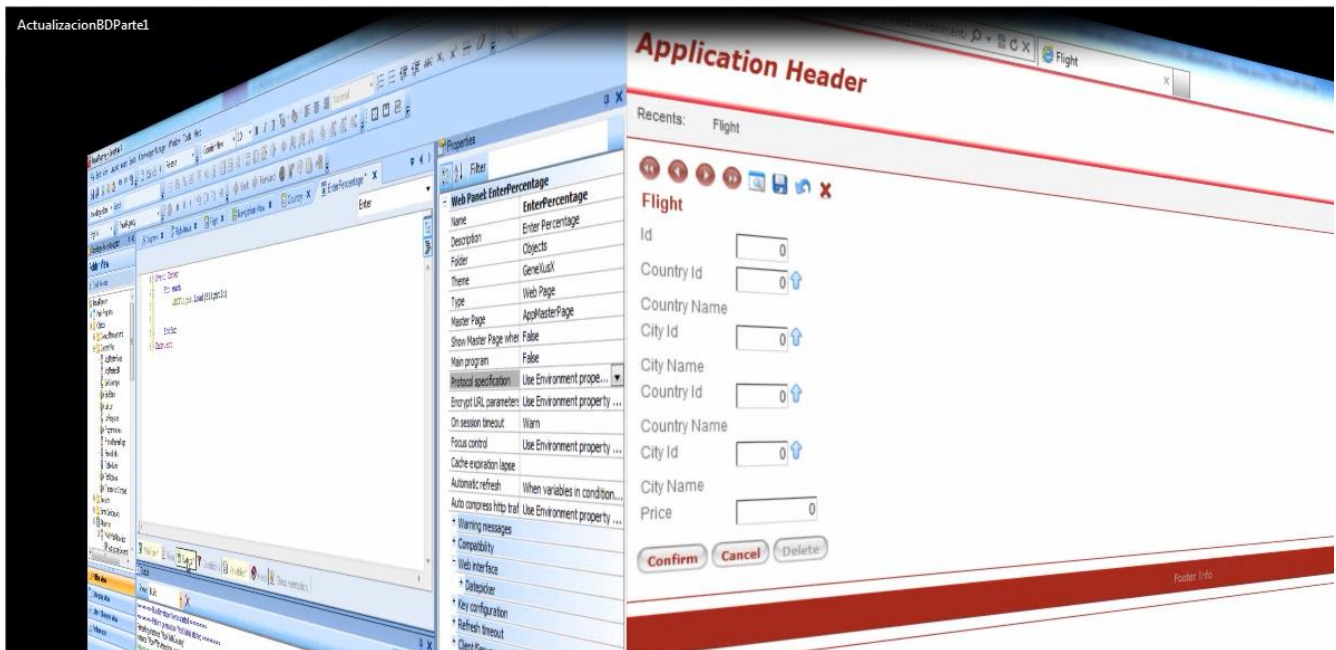
es por el momento el único atributo presente en el For each, y por lo tanto, hace que la tabla base del For each sea FLIGHT



De modo que el For each navegará **toda la tabla FLIGHT**, puesto que no hemos definido ningún where con condiciones de filtro, y para cada registro navegado, se ejecutarán las instrucciones incluidas en el cuerpo del For each

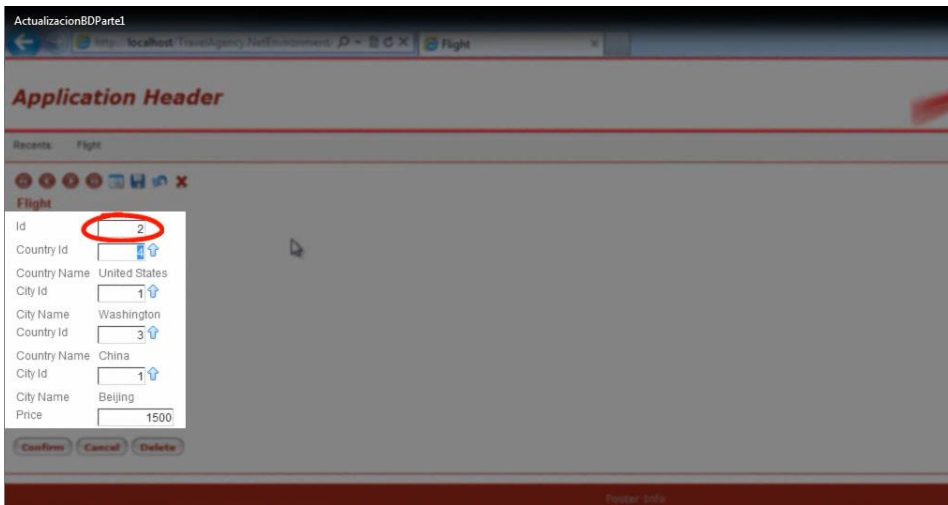


La primer sentencia entonces, efectúa algo equivalente a lo que sucede



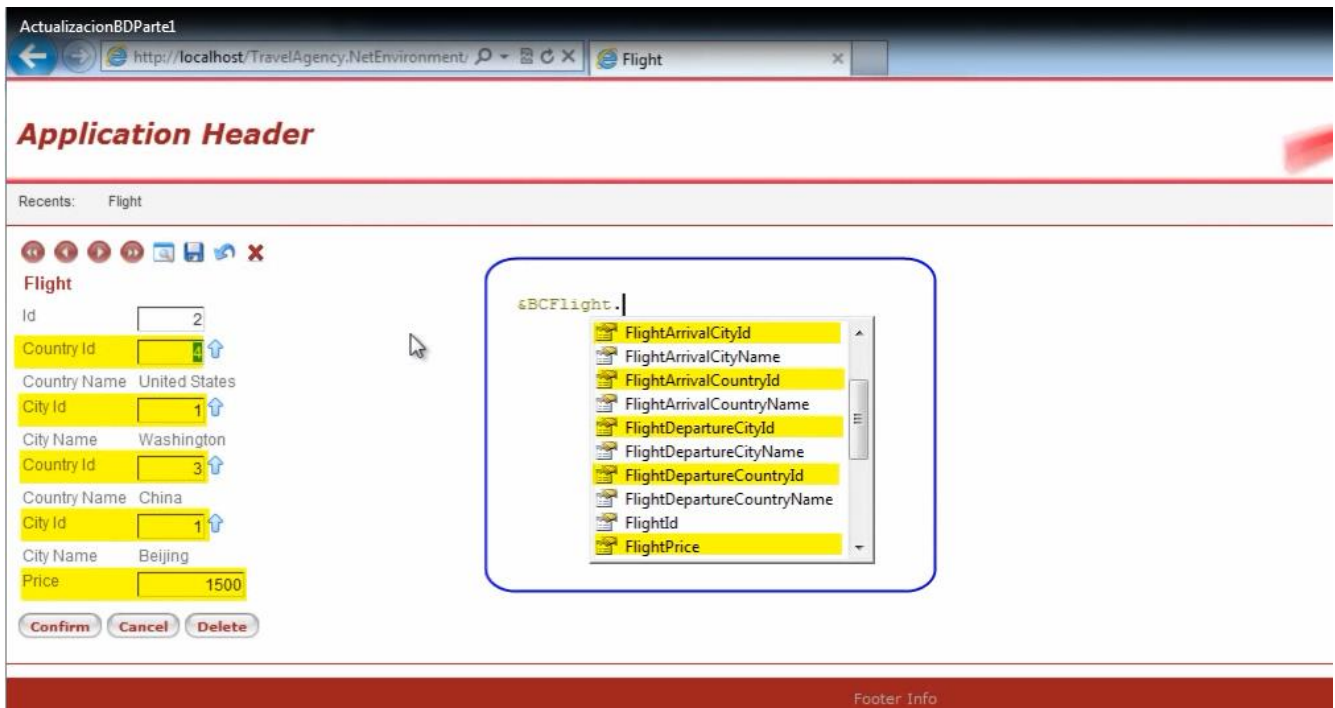
cuando digitamos en la transacción un valor de identificador de vuelo y salimos del campo.

Se cargan en memoria todos los datos correspondientes a ese identificador

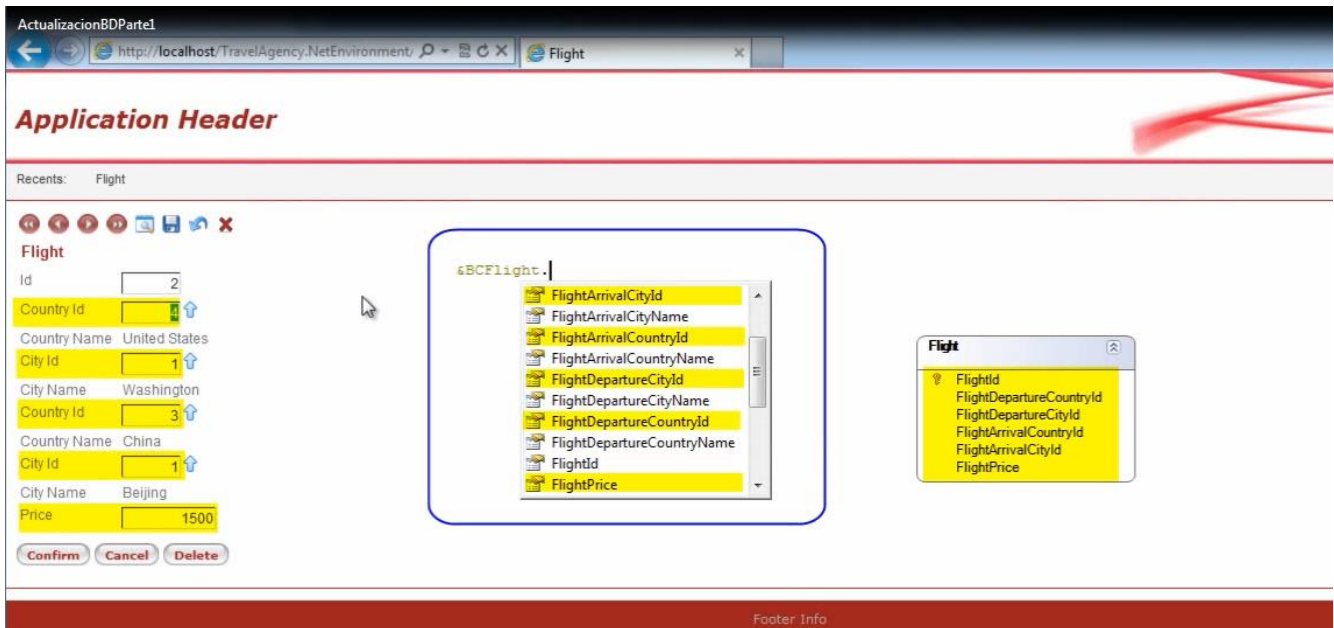


y los tendremos disponibles en este caso, en la variable &BCFlight.

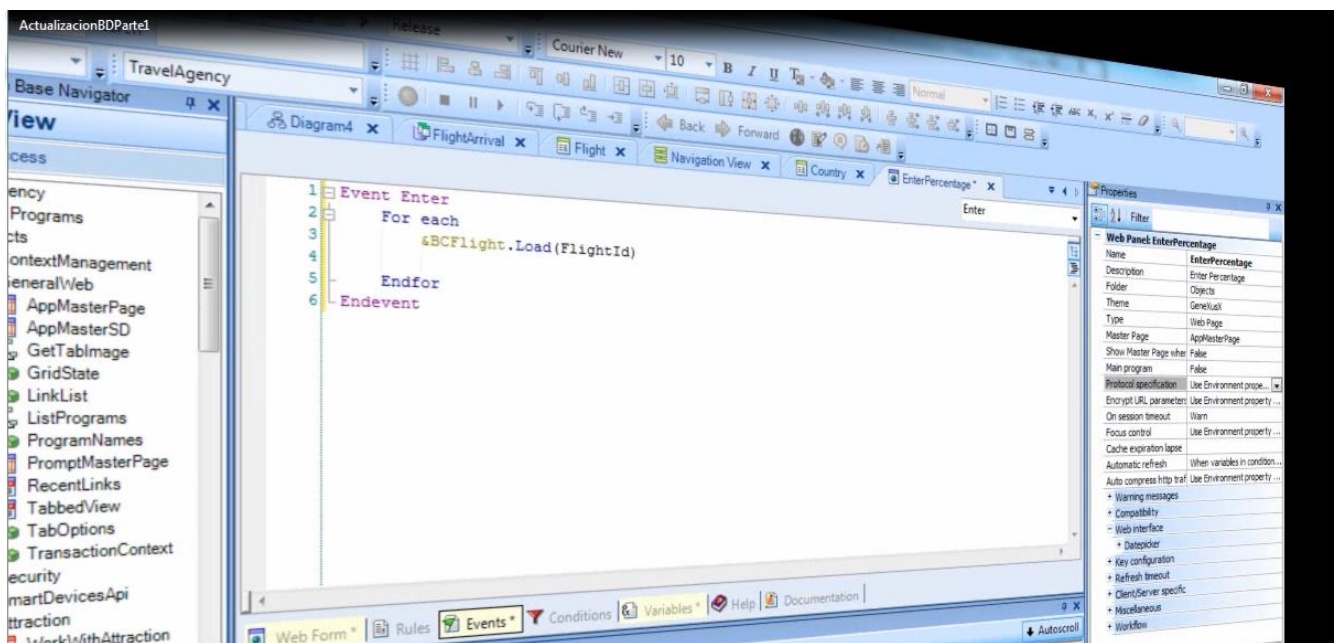
Podremos modificar solamente los mismos valores de atributos que aparecen editables en el form de la transacción



es decir, los que se encuentran en la tabla base asociada a la transacción, en este caso, FLIGHT

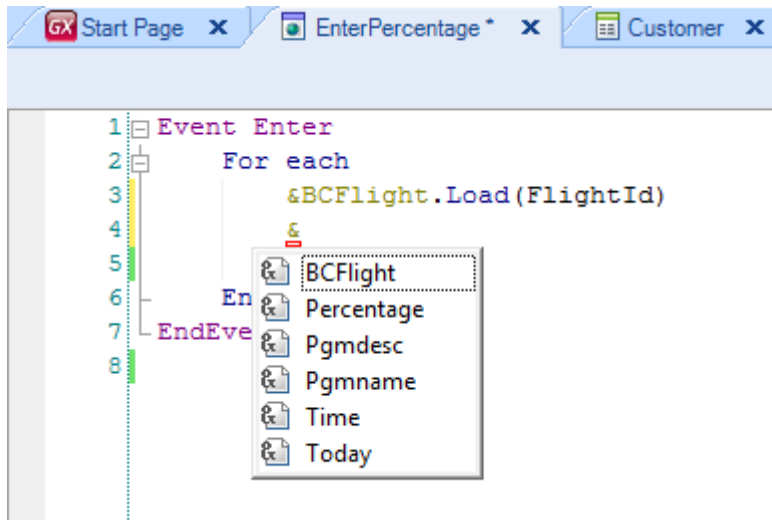


Volvamos ahora al código que estábamos escribiendo

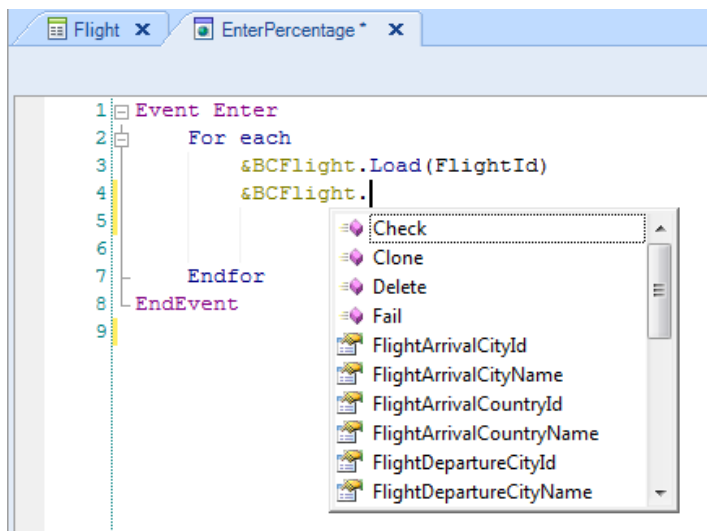


Como la primer instrucción del For each, **cargó los datos del vuelo navegado en memoria**, ahora solamente nos resta modificar el precio del vuelo y grabar los cambios.

Vamos a escribir la 2da instrucción dentro del For each. Digitamos ampersand

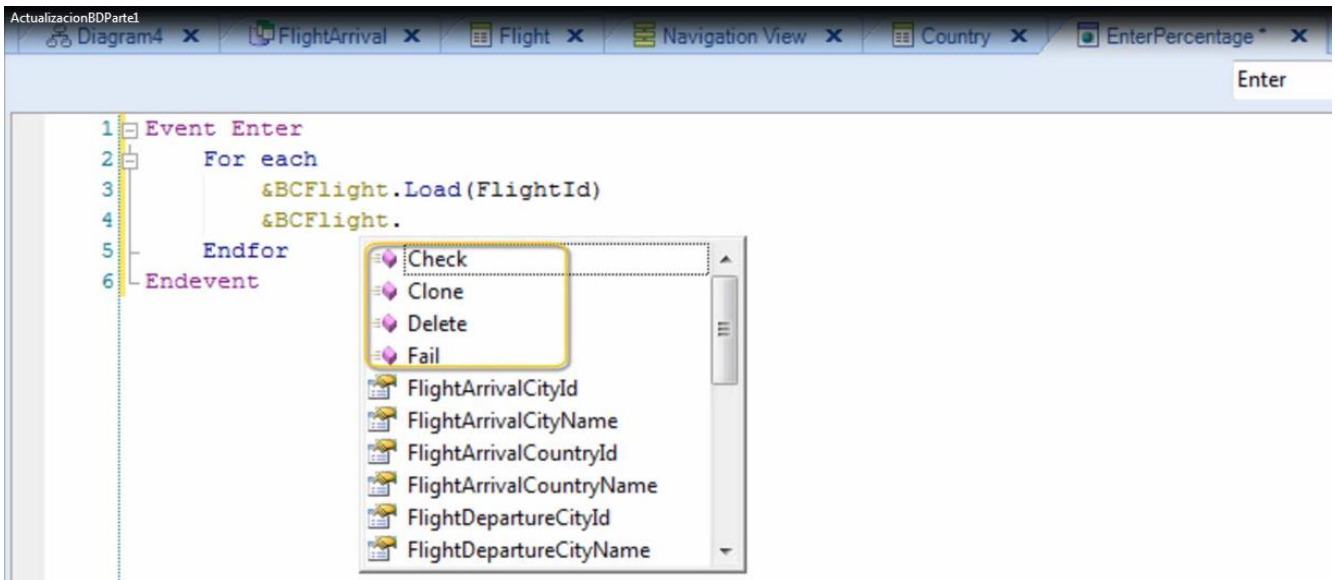


Seleccionamos la variable &BCFlight, digitamos punto,

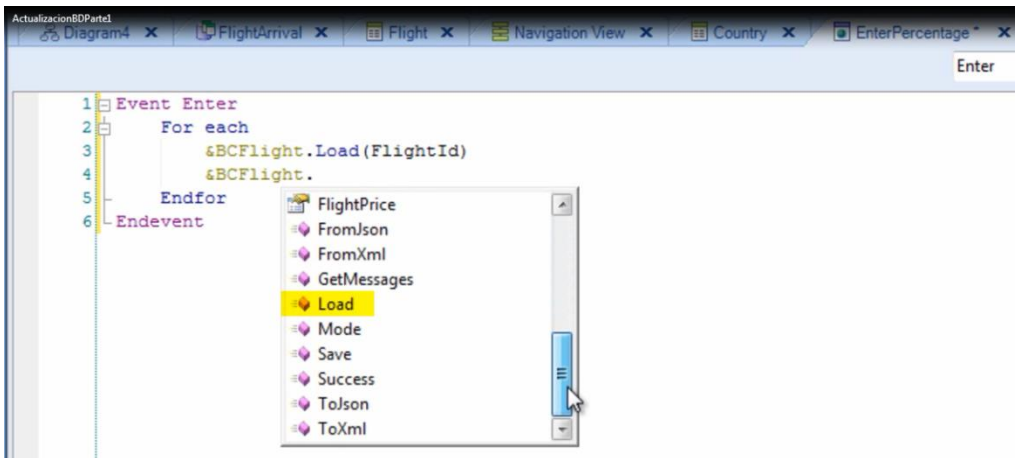


y observemos que tenemos 2 tipo se elementos que podemos elegir.

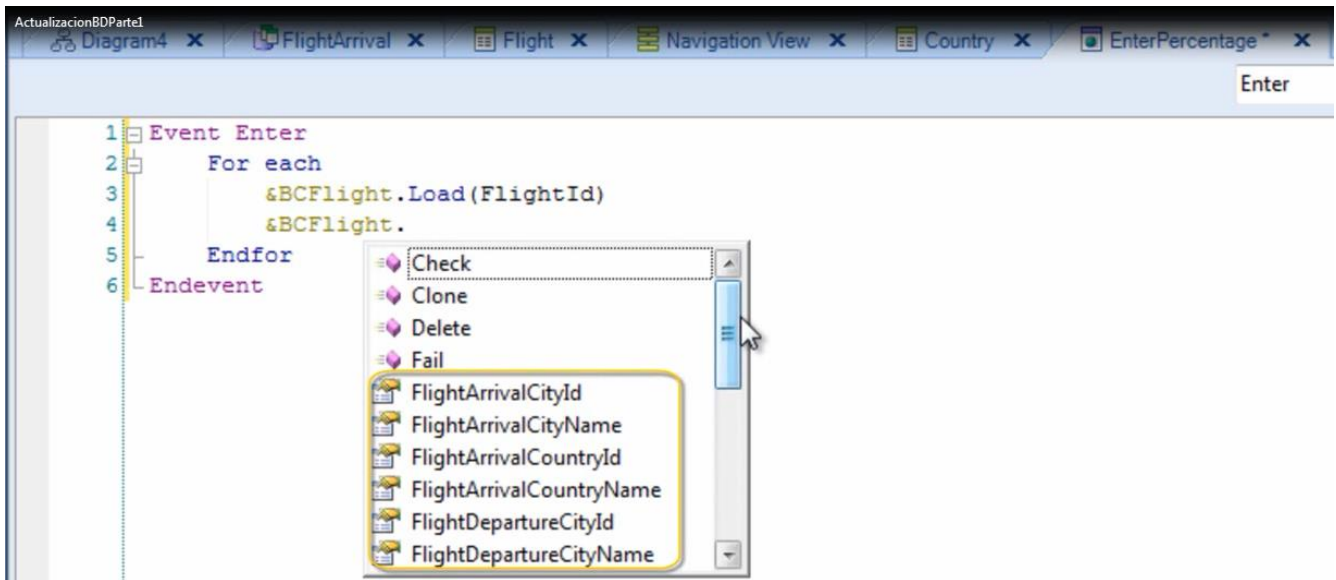
Los que tienen un ícono color violeta



son métodos para aplicar a la variable, como el método Load que recién hemos usado y explicado

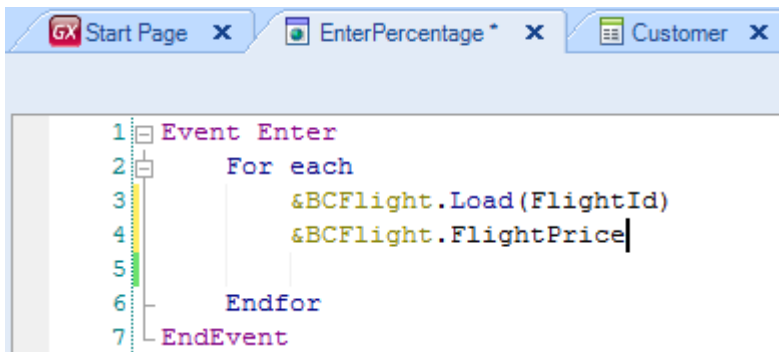


y los que tienen el ícono con la mano

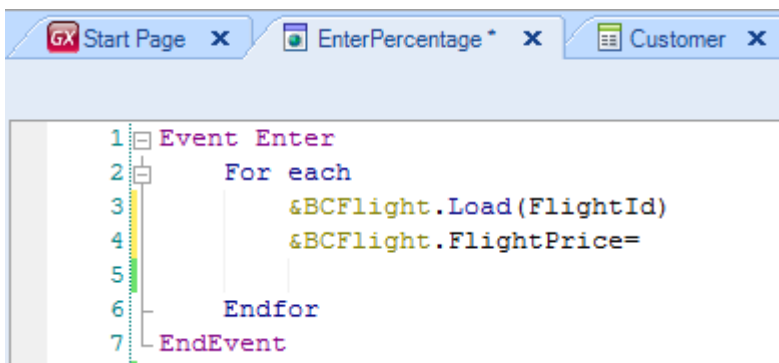


son los atributos presentes en la estructura de la transacción que es business component y de cuyo tipo es la variable &BCFlight

Seleccionamos el atributo FlightPrice



y para asignarle valor al atributo, digitamos signo de igual

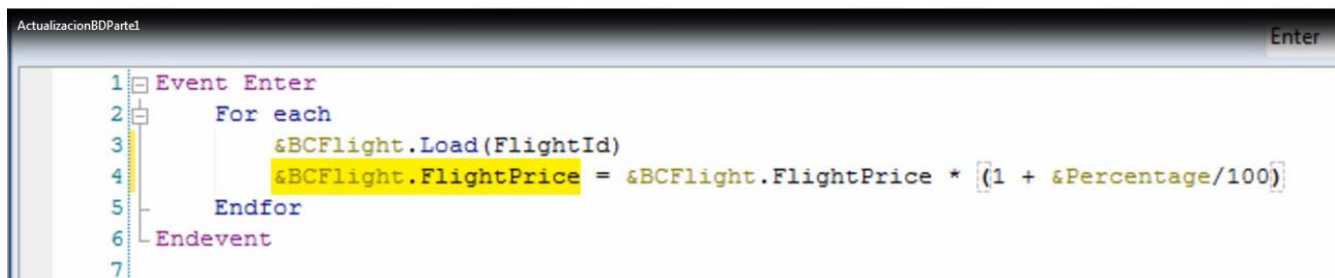


Ahora a la derecha del signo, debemos definir el cálculo para obtener el nuevo precio del vuelo, con el porcentaje de aumento aplicado.

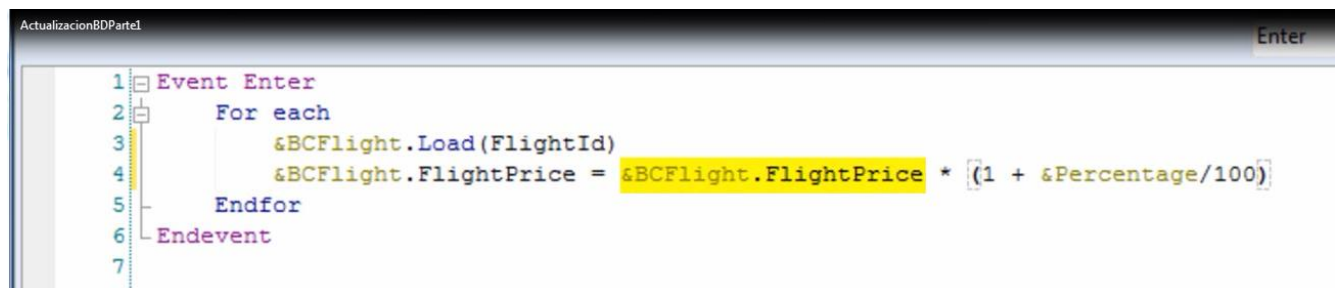
Vamos a escribir el cálculo y en seguida lo explicaremos

```
1 Event Enter
2   For each
3     &BCFlight.Load(FlightId)
4     &BCFlight.FlightPrice= &BCFlight.FlightPrice * (1 + &Percentage/100)
5   Endfor
6 EndEvent
```

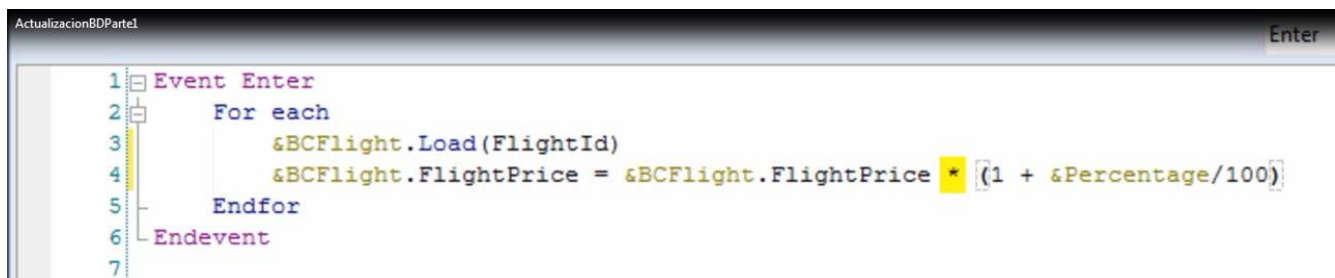
Esta instrucción, le está asignando al precio del vuelo que tenemos cargado en memoria



el mismo valor **que tenía**



multiplicado por



el resultado de la cuenta que está entre paréntesis

```
ActualizacionBDParte1
Enter
1 Event Enter
2   For each
3     &BCFlight.Load(FlightId)
4     &BCFlight.FlightPrice = &BCFlight.FlightPrice * (1 + &Percentage/100)
5   Endfor
6 Endevent
7
```

Este cálculo está sumando 1 más el valor que tiene la variable &Percentage dividido 100... o sea, que si por ejemplo, el porcentaje de aumento que ingresaron en la variable fuera 20

```
ActualizacionBDParte1
Enter
1 Event Enter
2   For each
3     &BCFlight.Load(FlightId)
4     &BCFlight.FlightPrice = &BCFlight.FlightPrice * [(1 + &Percentage/100)]
5   Endfor
6 Endevent
7
```

20%

0,20

al multiplicar el precio actual * 1,20

```
ActualizacionBDParte1
Enter
1 Event Enter
2   For each
3     &BCFlight.Load(FlightId)
4     &BCFlight.FlightPrice = &BCFlight.FlightPrice * [(1 + &Percentage/100)]
5   Endfor
6 Endevent
7
```

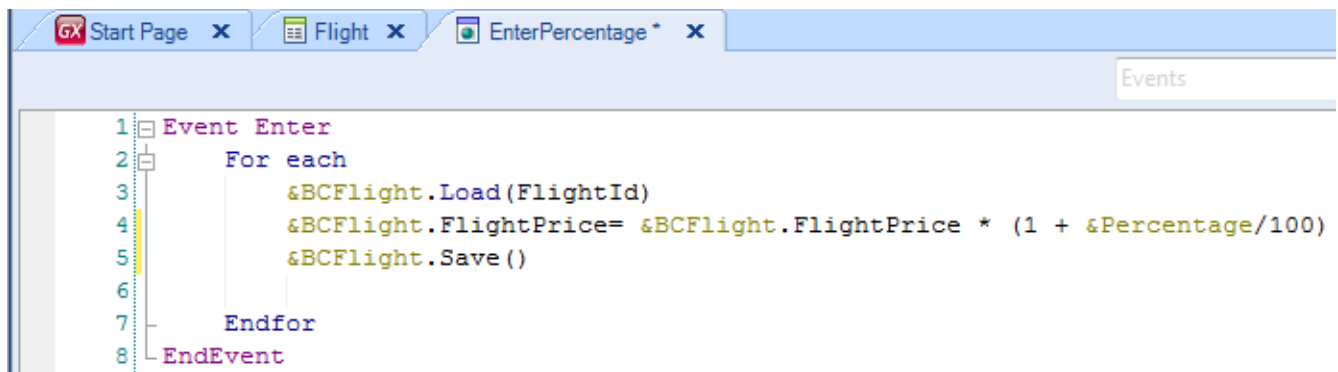
1,20

se obtiene el nuevo precio con el 20% de aumento, el cual asignamos al precio del vuelo

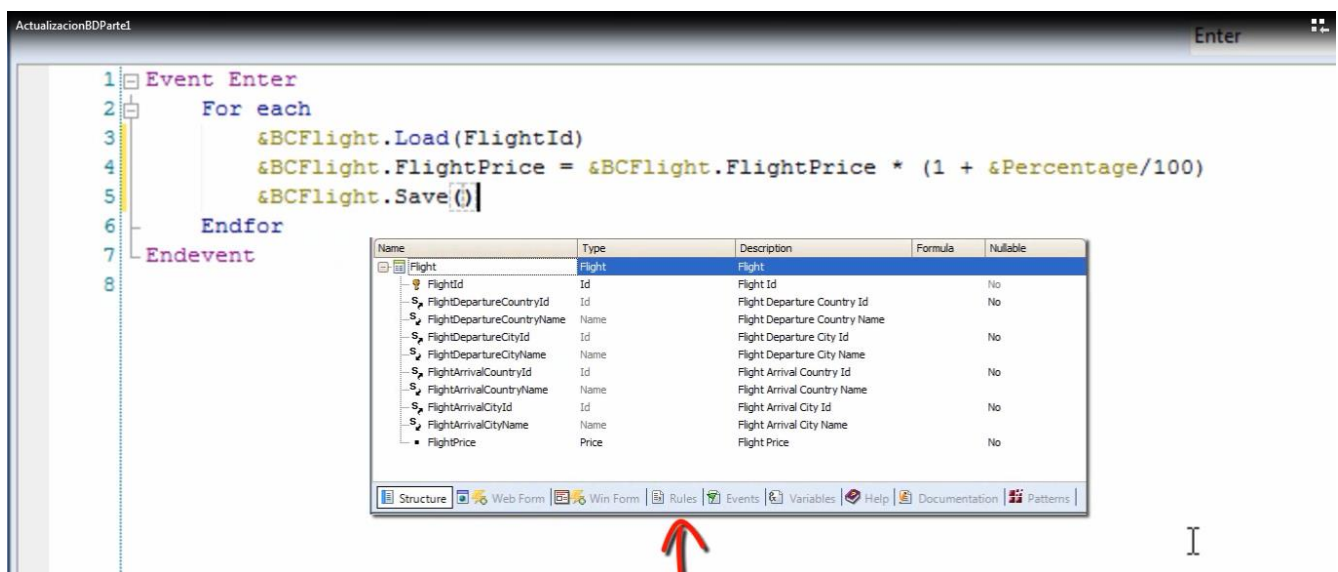
```
ActualizacionBDParte1
Enter
1 Event Enter
2   For each
3     &BCFlight.Load(FlightId)
4     &BCFlight.FlightPrice = &BCFlight.FlightPrice * (1 + &Percentage/100)
5   Endfor
6 Endevent
7
```

Ahora debemos grabar físicamente esta actualización. Escribamos la instrucción para hacerlo.

&BCFlight, punto... elegimos save... y observemos que los métodos siempre van seguidos de paréntesis.

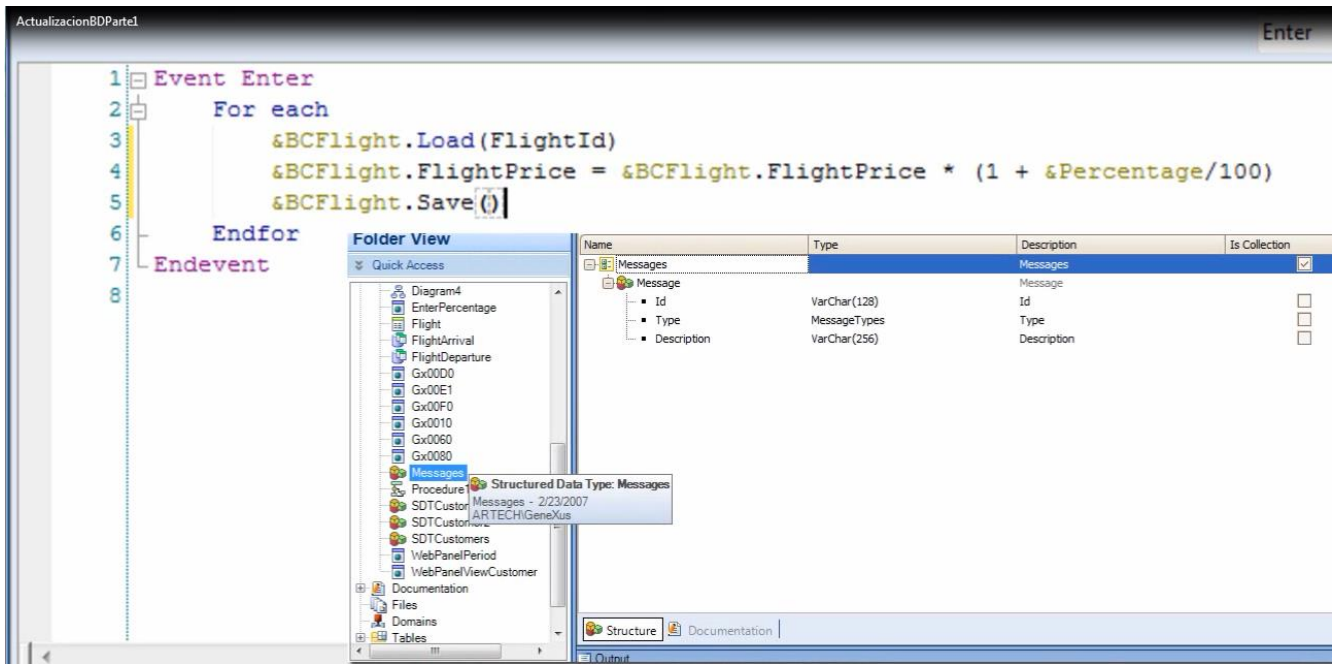


Cuando se ejecute el método save, se dispararán las reglas definidas en la transacción Flight –con algunas excepciones-

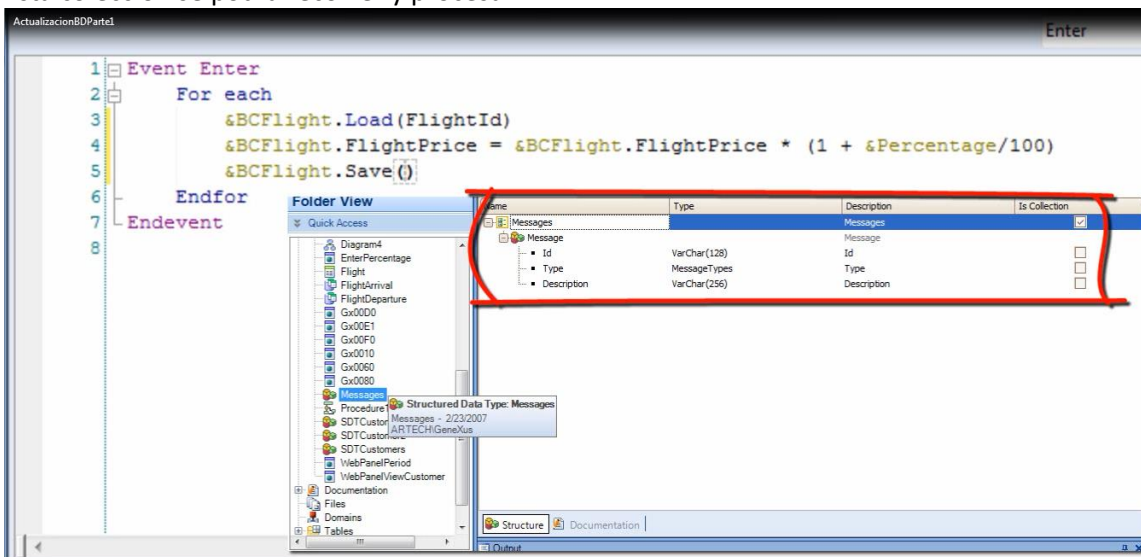


y se ejecutarán las validaciones automáticas que efectúa la transacción.

Si ocurren errores (ya sea porque se han disparado reglas Error que hayamos definido en la transacción o por validaciones automáticas), los textos correspondientes a lo que sucedió, quedarán cargados en una colección en memoria.



Esta colección se podrá recorrer y procesar.



No lo mostraremos en este video, sino en otro que enseña más detalles sobre business components.

Ahora evaluaremos si la operación de grabación fue exitosa

```
ActualizacionBDParte1 Enter
1 Event Enter
2 For each
3     &BCFlight.Load(FlightId)
4     &BCFlight.FlightPrice = &BCFlight.FlightPrice * (1 + &Percentage/100)
5     &BCFlight.Save()
6     If &BCFlight.Success()
7     Endfor
8 Endevent
9
```

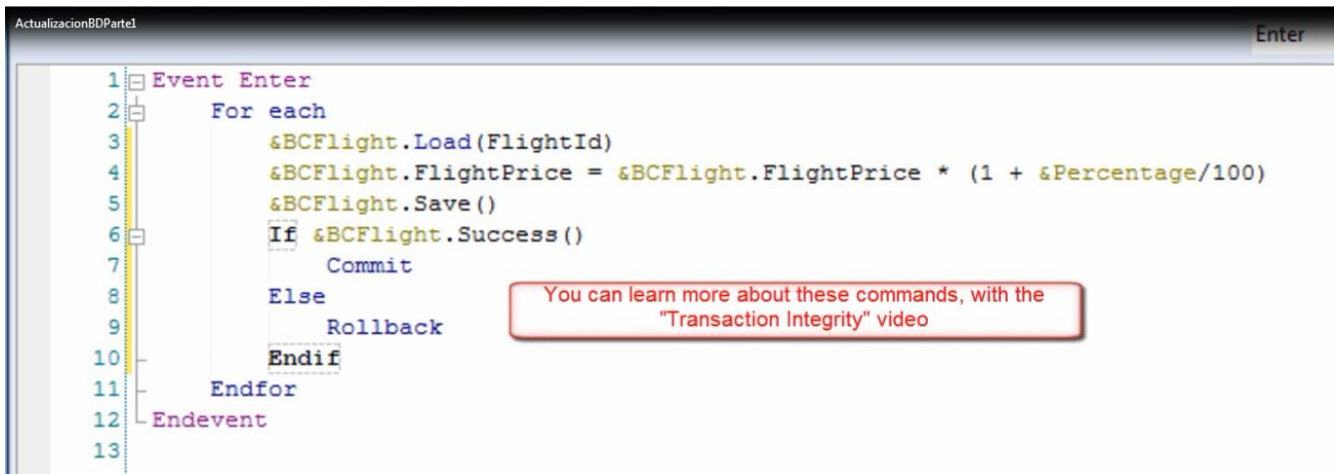
En caso de éxito, escribiremos Commit

```
ActualizacionBDParte1 Enter
1 Event Enter
2 For each
3     &BCFlight.Load(FlightId)
4     &BCFlight.FlightPrice = &BCFlight.FlightPrice * (1 + &Percentage/100)
5     &BCFlight.Save()
6     If &BCFlight.Success()
7         Commit
8     Endfor
9 Endevent
10
```

y en caso contrario, escribiremos Rollback

```
ActualizacionBDParte1 Enter
1 Event Enter
2 For each
3     &BCFlight.Load(FlightId)
4     &BCFlight.FlightPrice = &BCFlight.FlightPrice * (1 + &Percentage/100)
5     &BCFlight.Save()
6     If &BCFlight.Success()
7         Commit
8     Else
9         Rollback
10    Endfor
11 Endevent
12
13
```

para deshacer las operaciones que intentamos efectuar a la base de datos posteriores al último commit

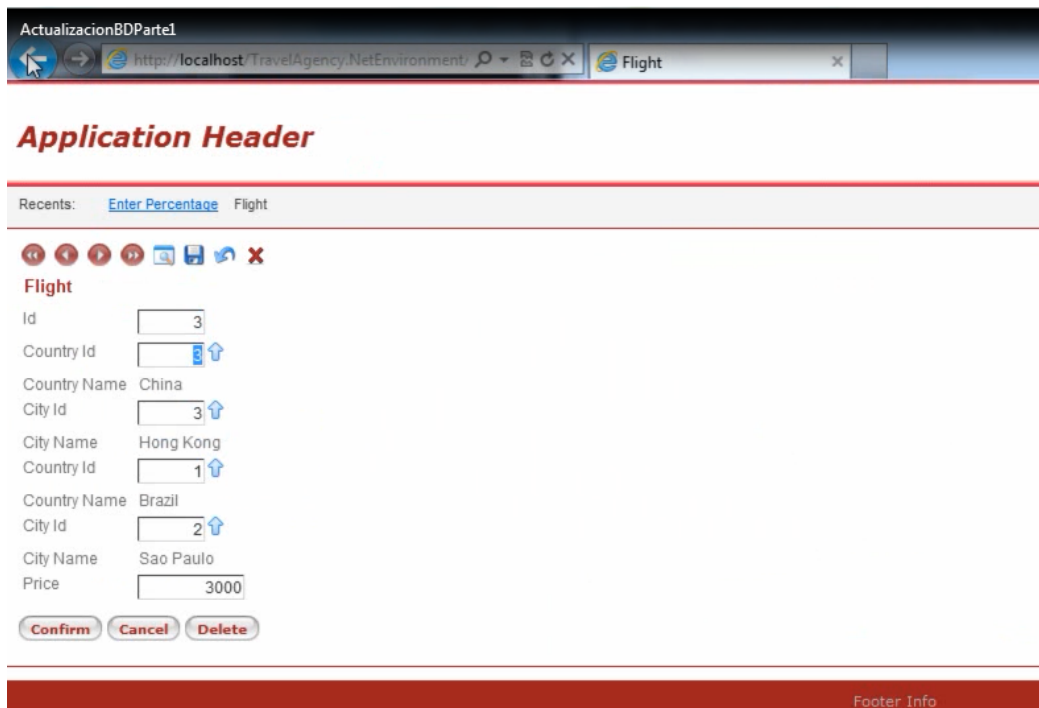


Vamos a probar ahora esta funcionalidad que hemos implementado.

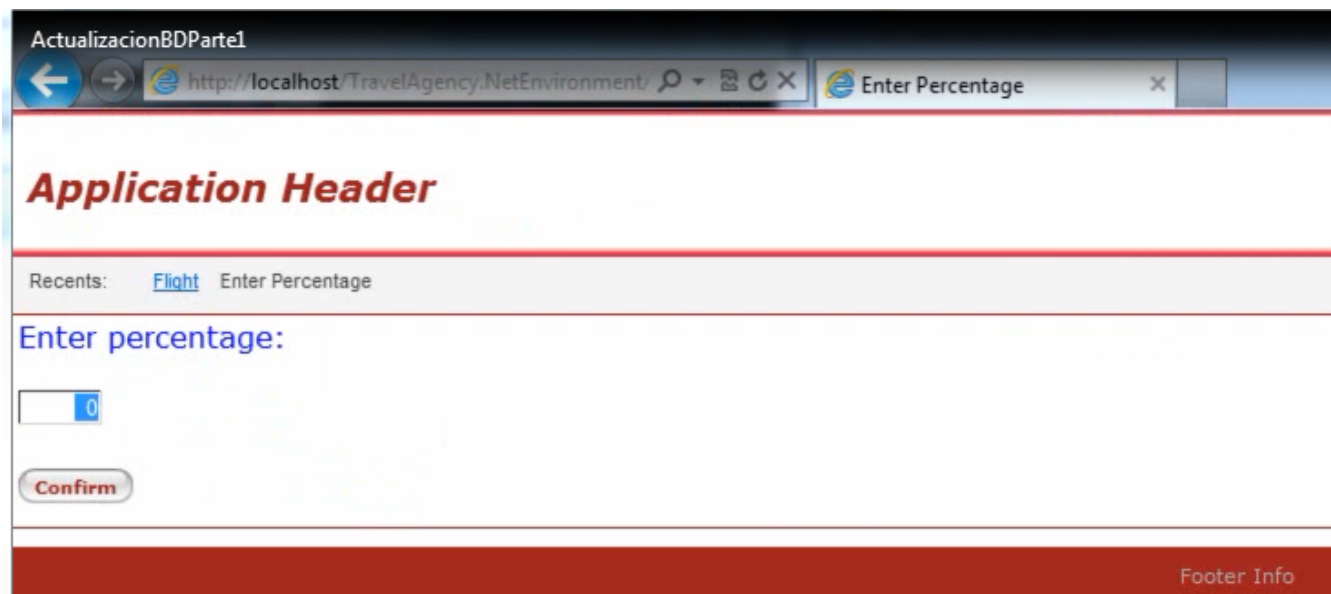
Presionamos F5...



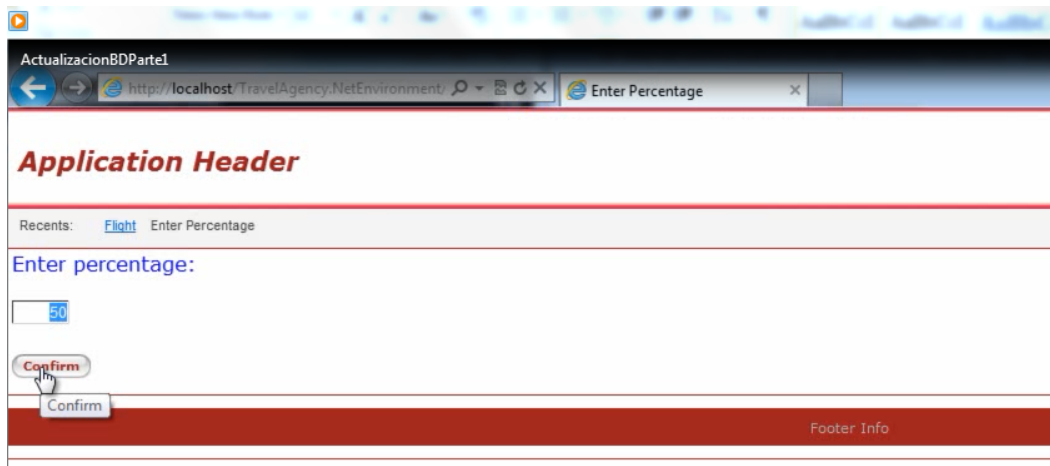
Ejecutemos la transacción Flight para revisar los precios de algunos vuelos...



Y ahora ejecutemos el web panel “EnterPercentage”.

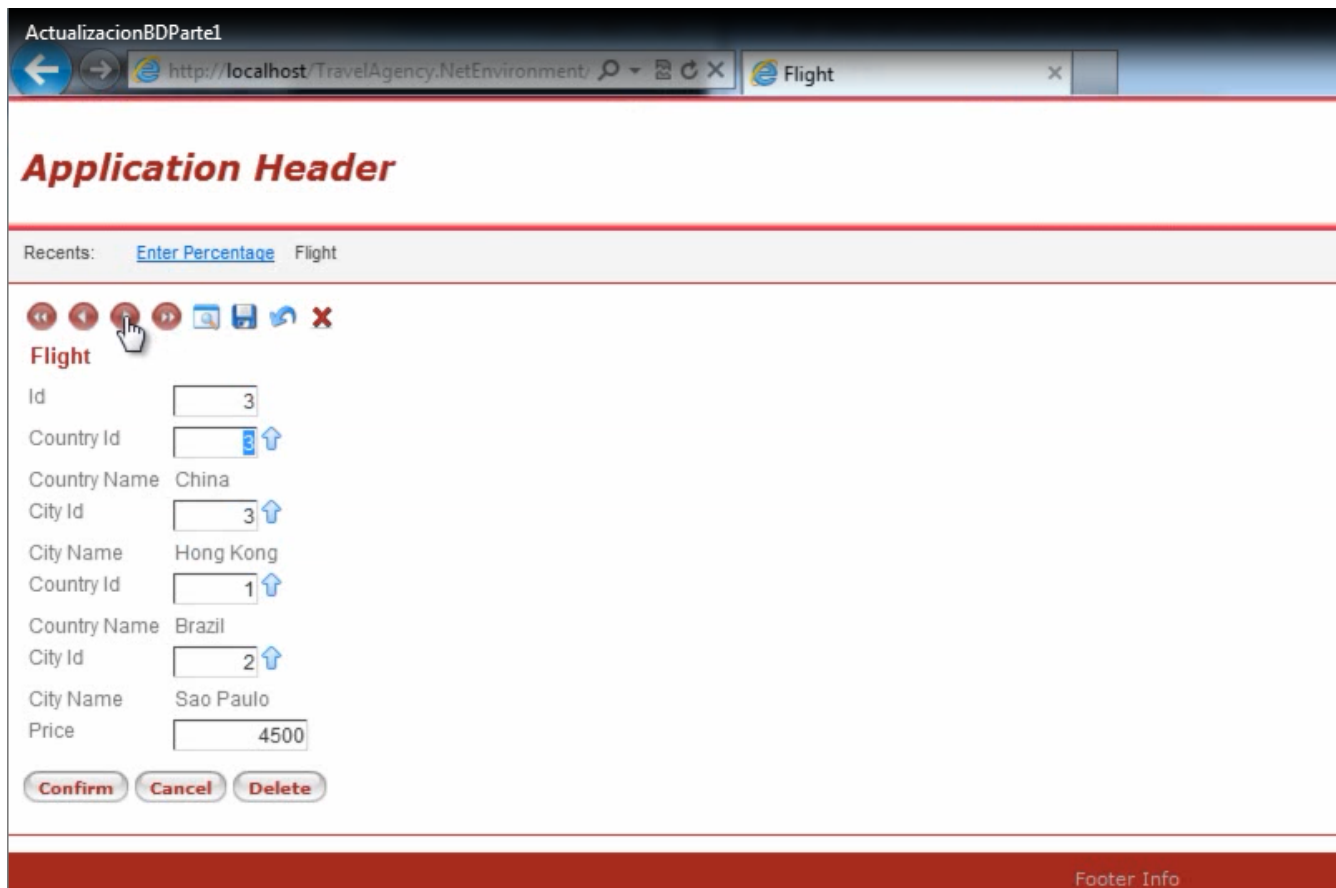


Ingrese un porcentaje de aumento del 50 % y presionemos el botón confirmar

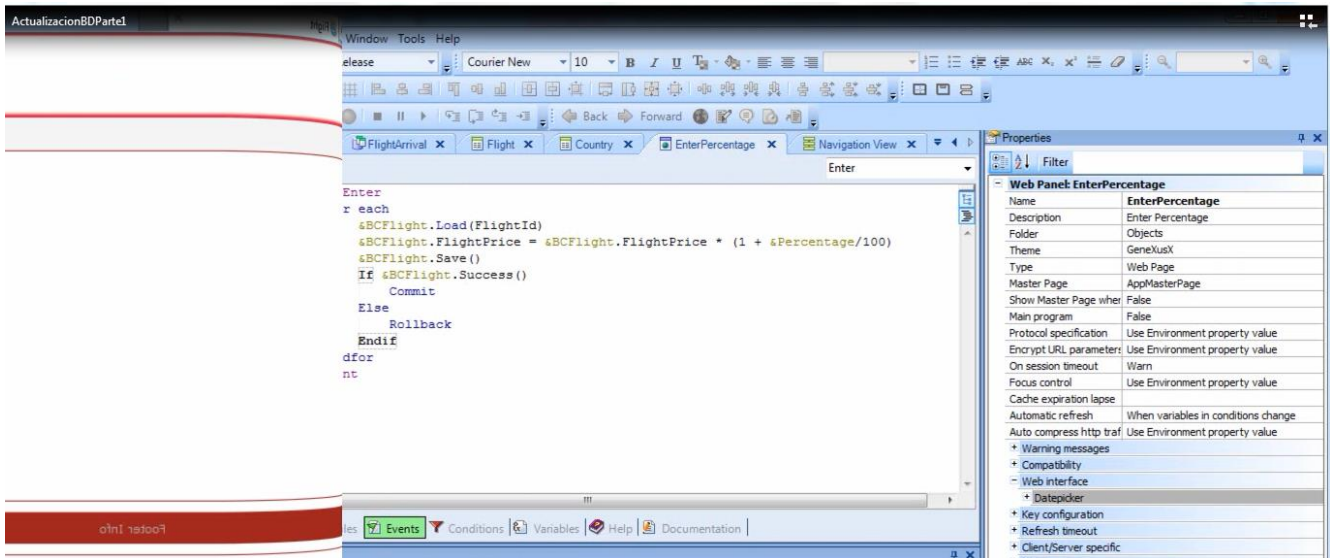


No hemos incluido ningún mensaje que avise que el proceso se realizó y terminó, pero vamos a corroborarlo, ejecutando la transacción “Flight” nuevamente.

Si recordamos los precios que habíamos ingresado, podemos notar que todos se han aumentado en un 50%.

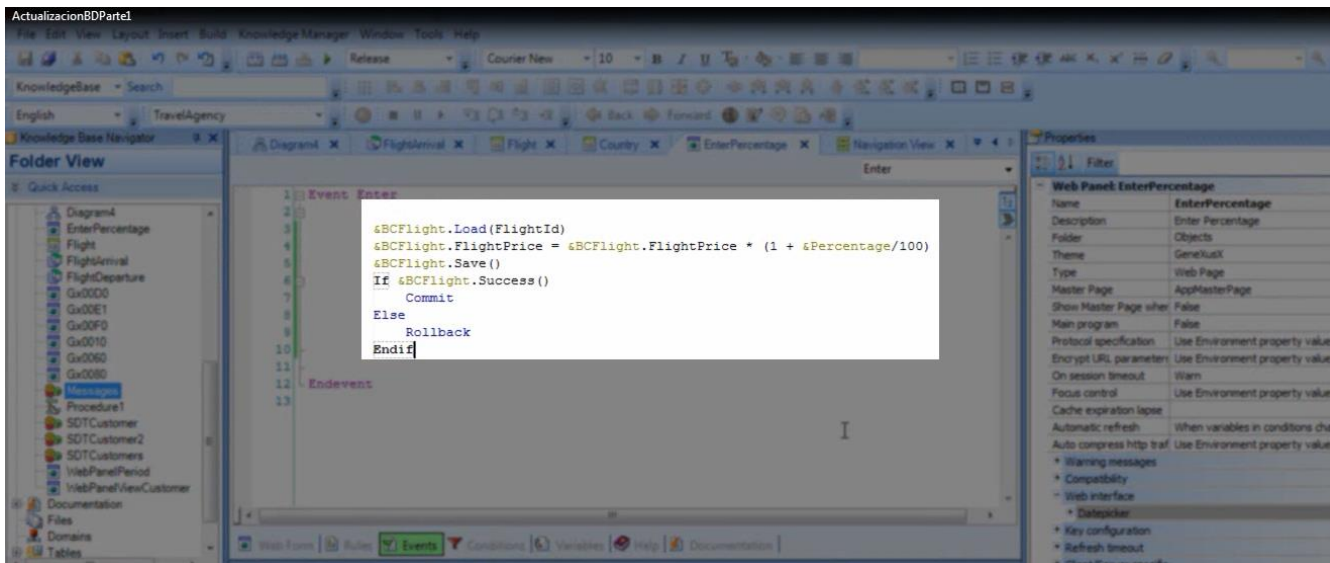


Ahora volvamos a la codificación que hicimos

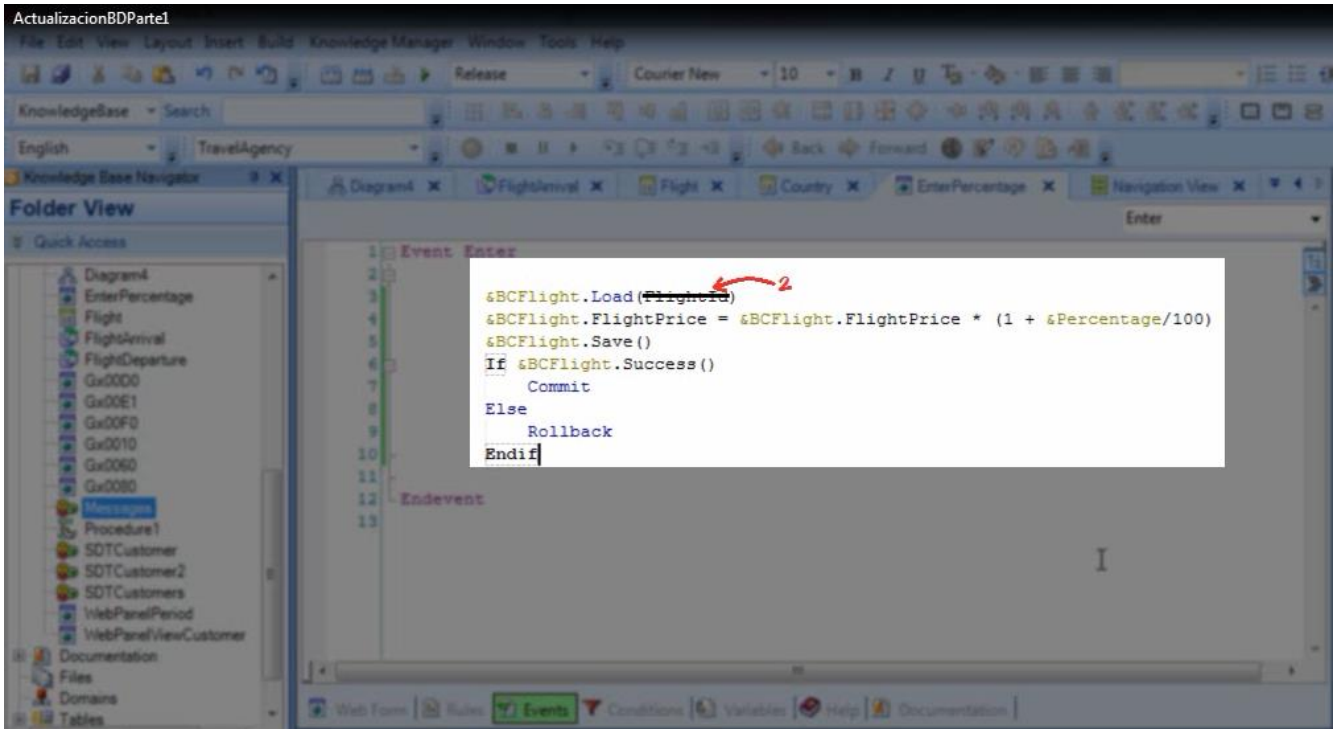


Es importante aclarar que podemos usar una variable del tipo business component fuera de un comando For each.

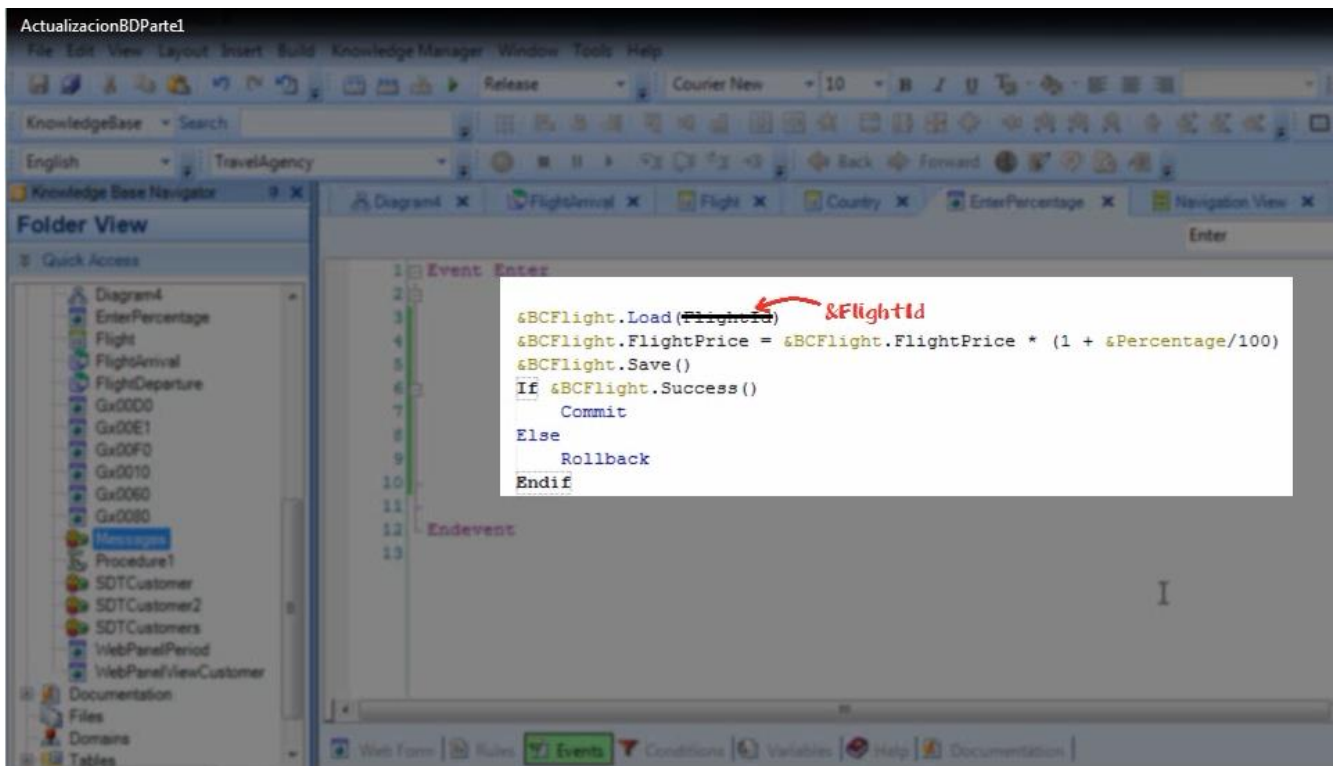
Por ejemplo, todo este código



es válido fuera del For each, con la única salvedad que el valor del identificador de vuelo a ser cargado en memoria, tendría que especificarse como valor fijo, por ejemplo así

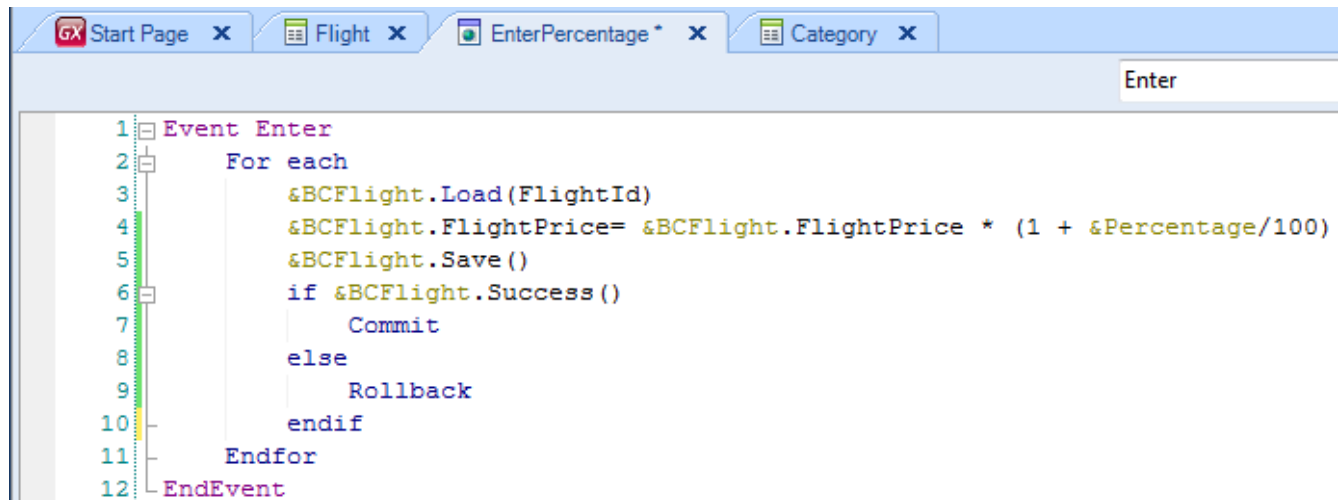


o tenerlo cargado en una variable y especificar la variable dentro del paréntesis, así:



Esta sería una carga en memoria puntual **de 1 vuelo** y estaríamos actualizando el precio **de ese vuelo...**

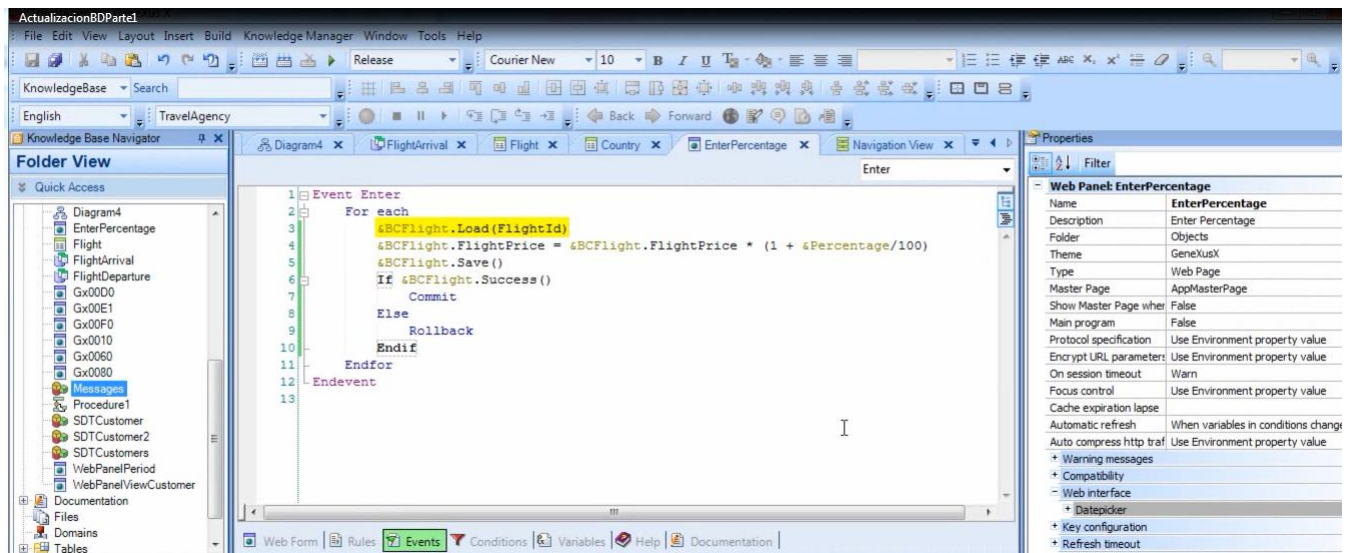
No es una actualización masiva, como la que hicimos dentro del For each.



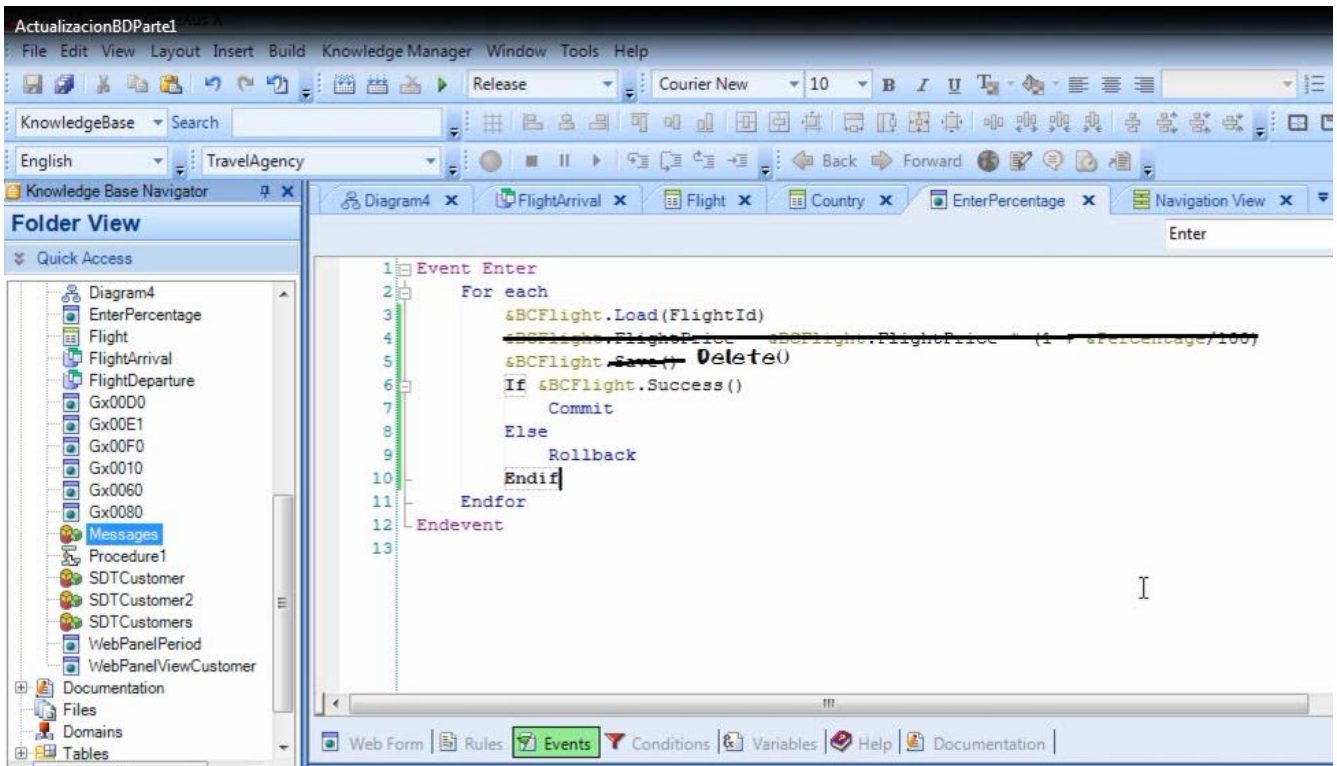
Ahora, supongamos que deseamos eliminar los vuelos que tenemos registrados.

Para eliminar en la base de datos usando business components, hay que hacer casi lo mismo que hicimos para actualizar:

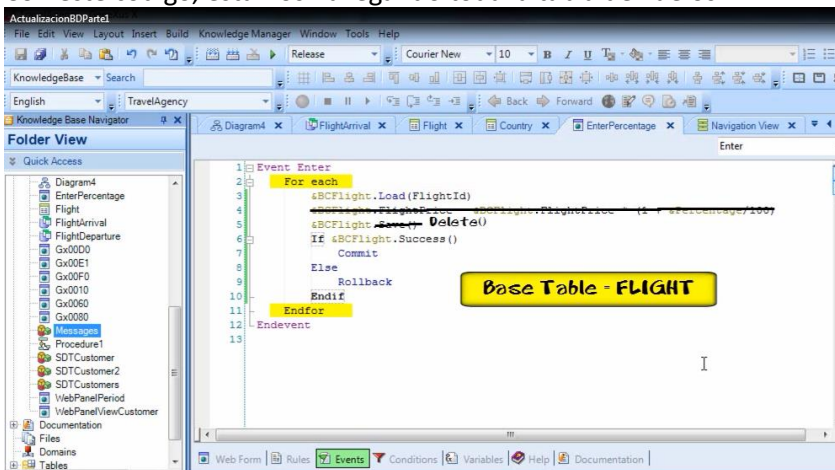
Primero cargar en memoria los datos asociados a una llave primaria usando el método Load



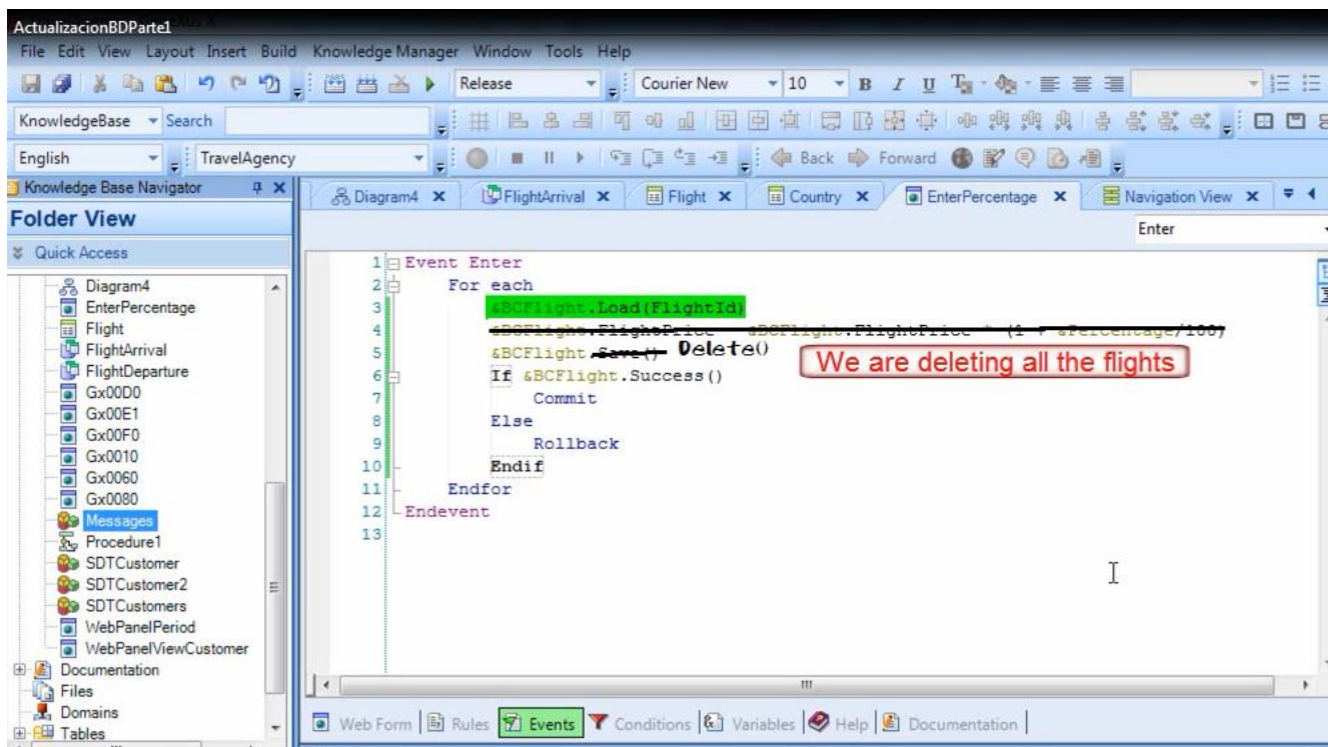
y después ejecutar el método Delete para efectuar la eliminación



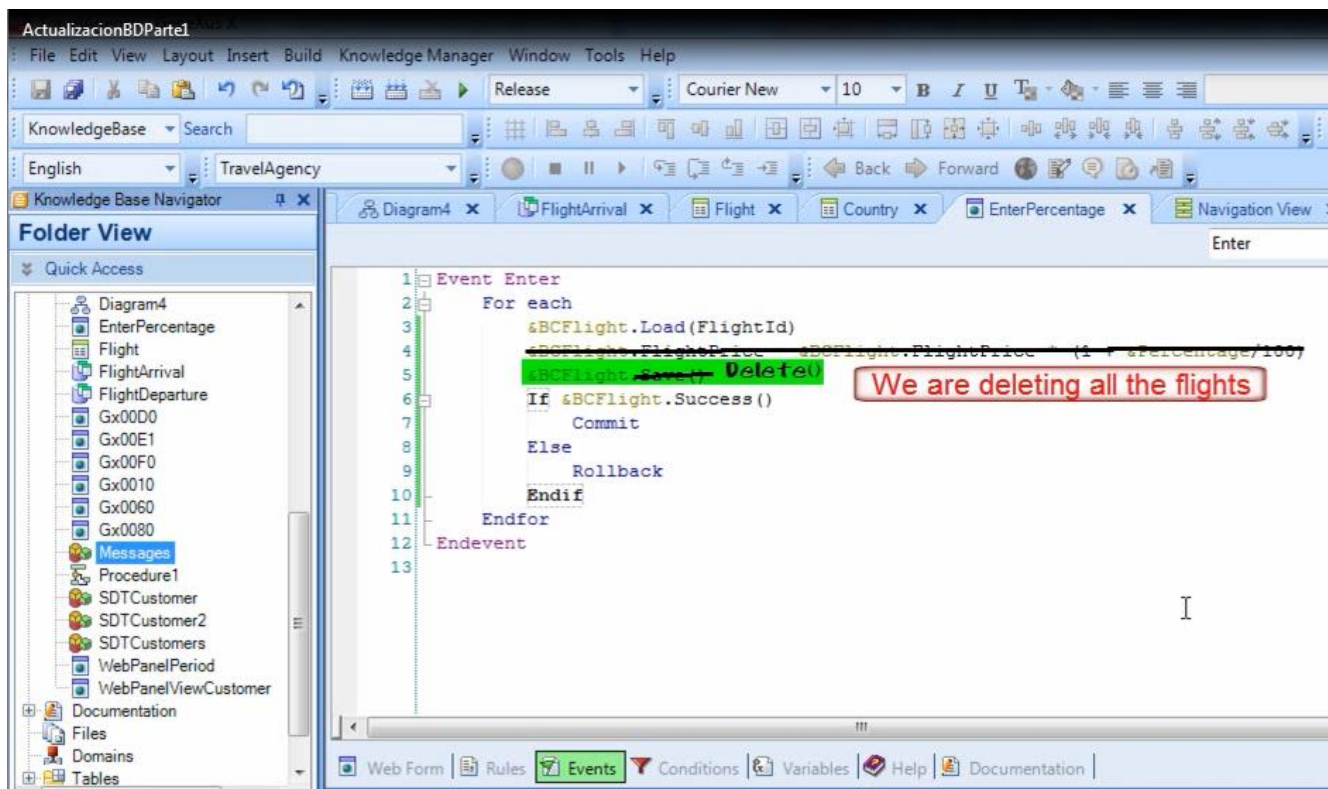
Con este código, estamos navegando toda la tabla de vuelos



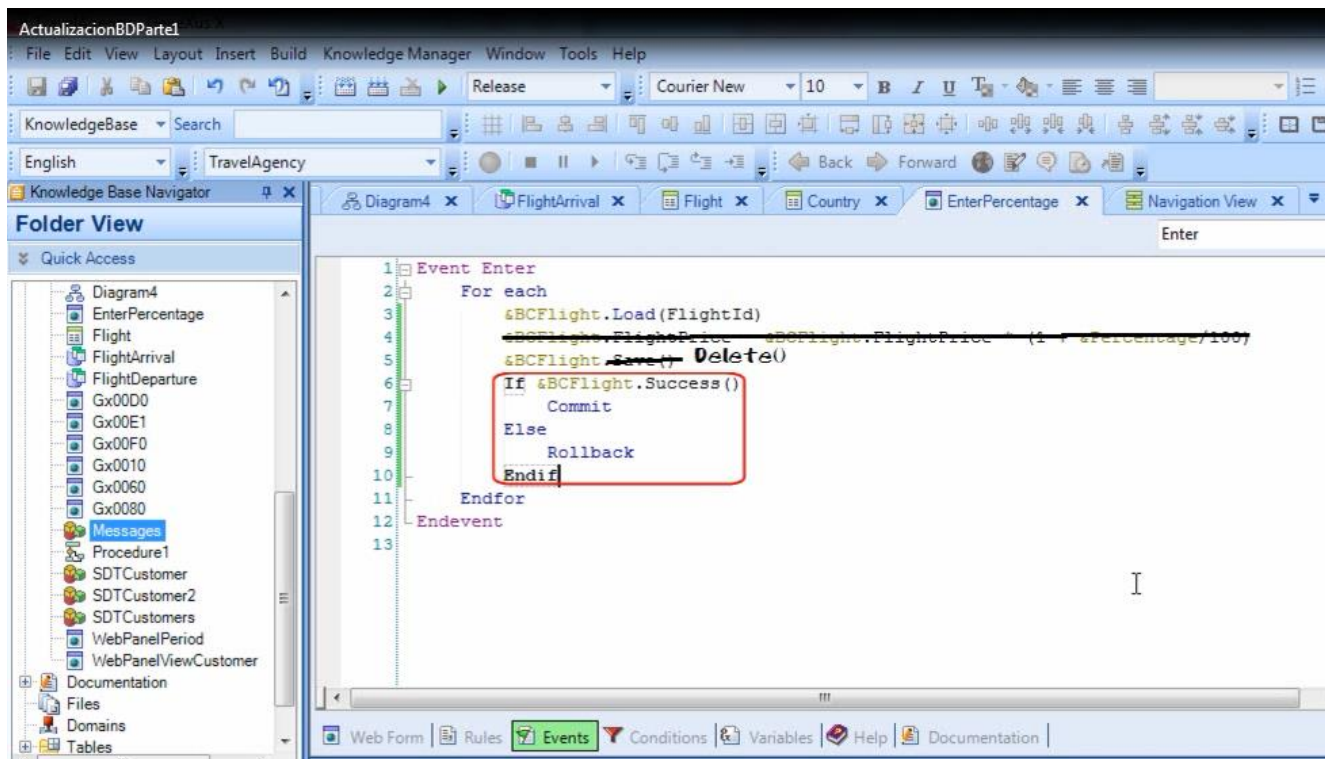
cargamos cada vuelo navegado en memoria



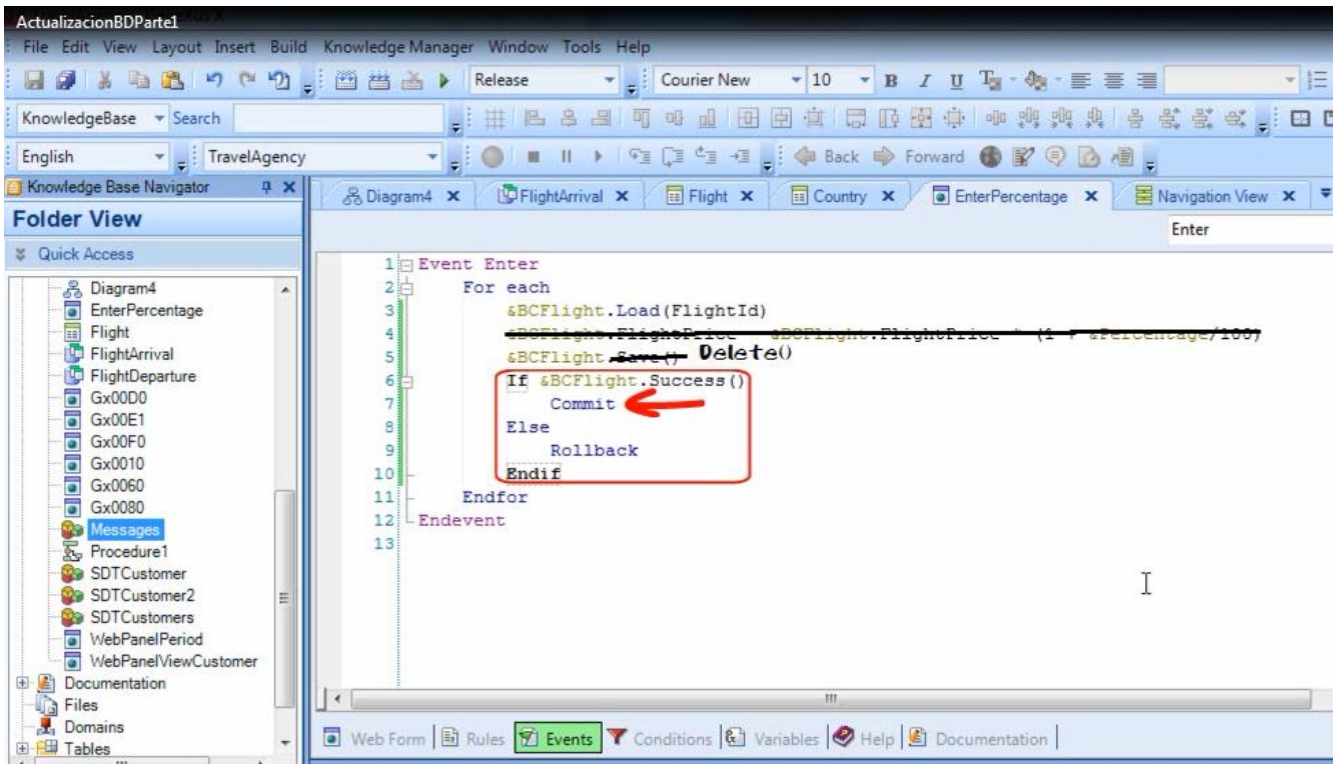
y eliminamos el vuelo



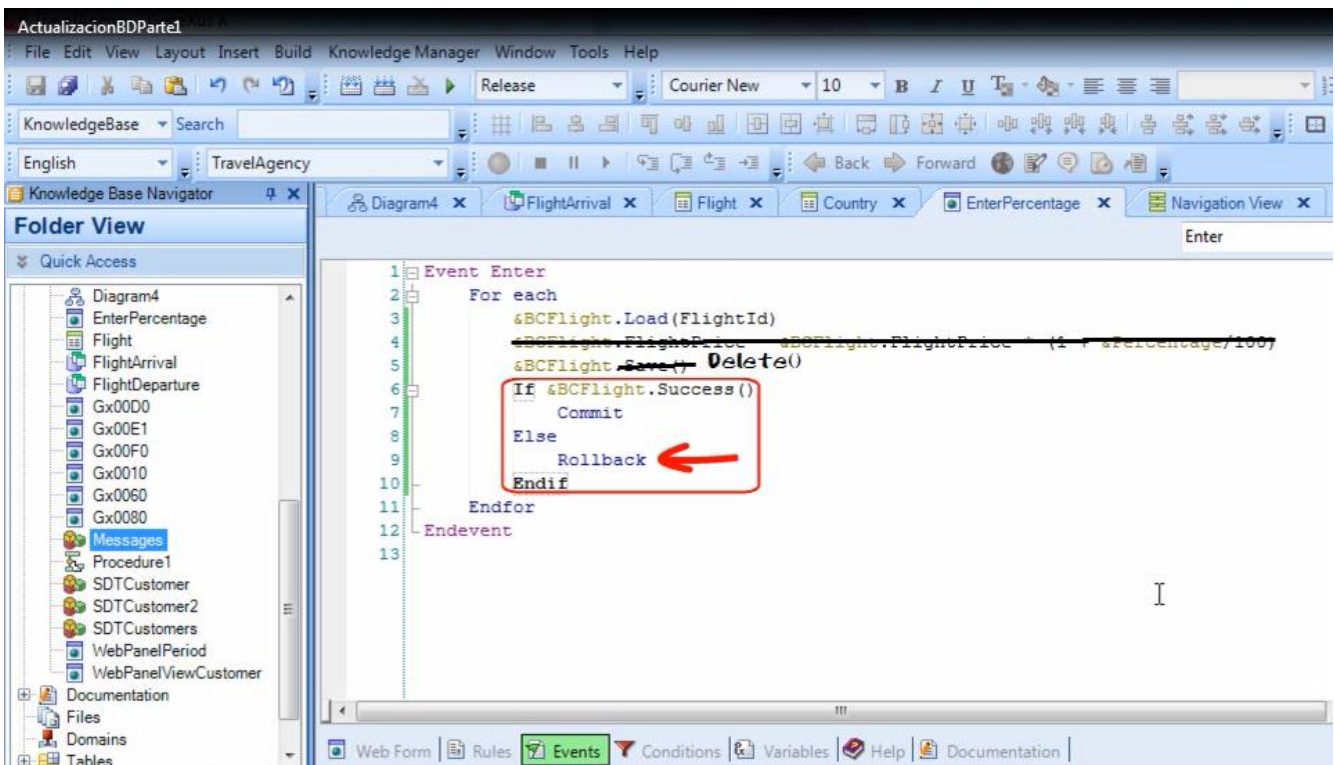
Este último bloque de código



se mantiene igual, validando si no ocurrieron errores al intentar eliminar, en cuyo caso confirmamos la eliminación de la base de datos

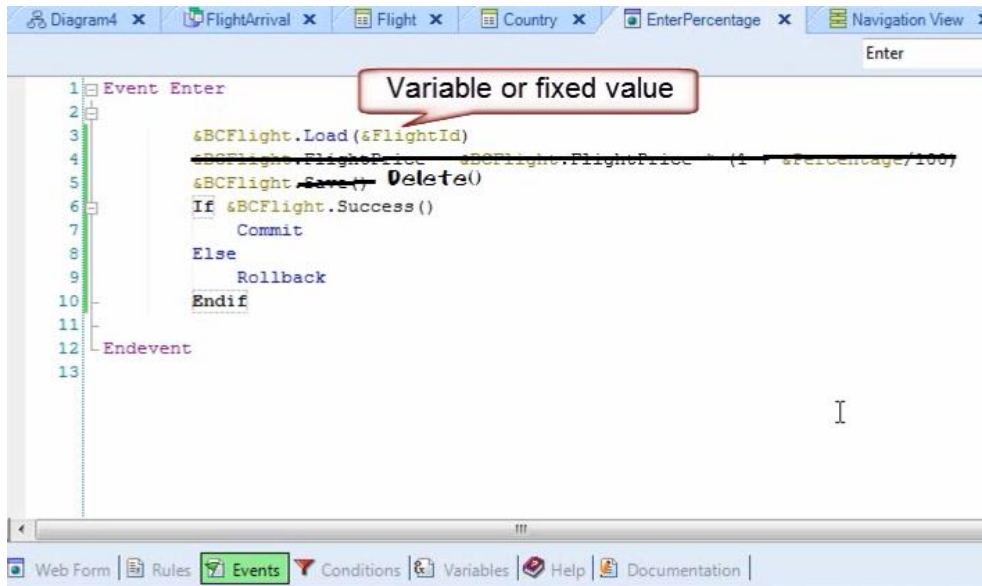


y en caso contrario, no



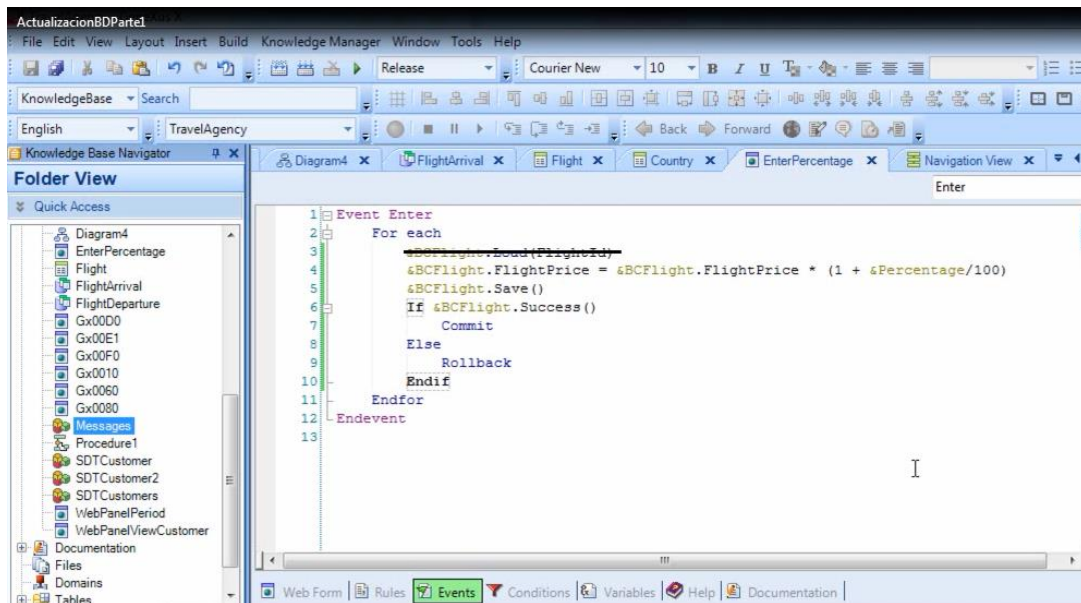
Al igual que como vimos para la actualización, podemos querer eliminar un vuelo puntualmente y para eso

codificaríamos la carga en memoria del vuelo específico y eliminación, fuera del comando For each



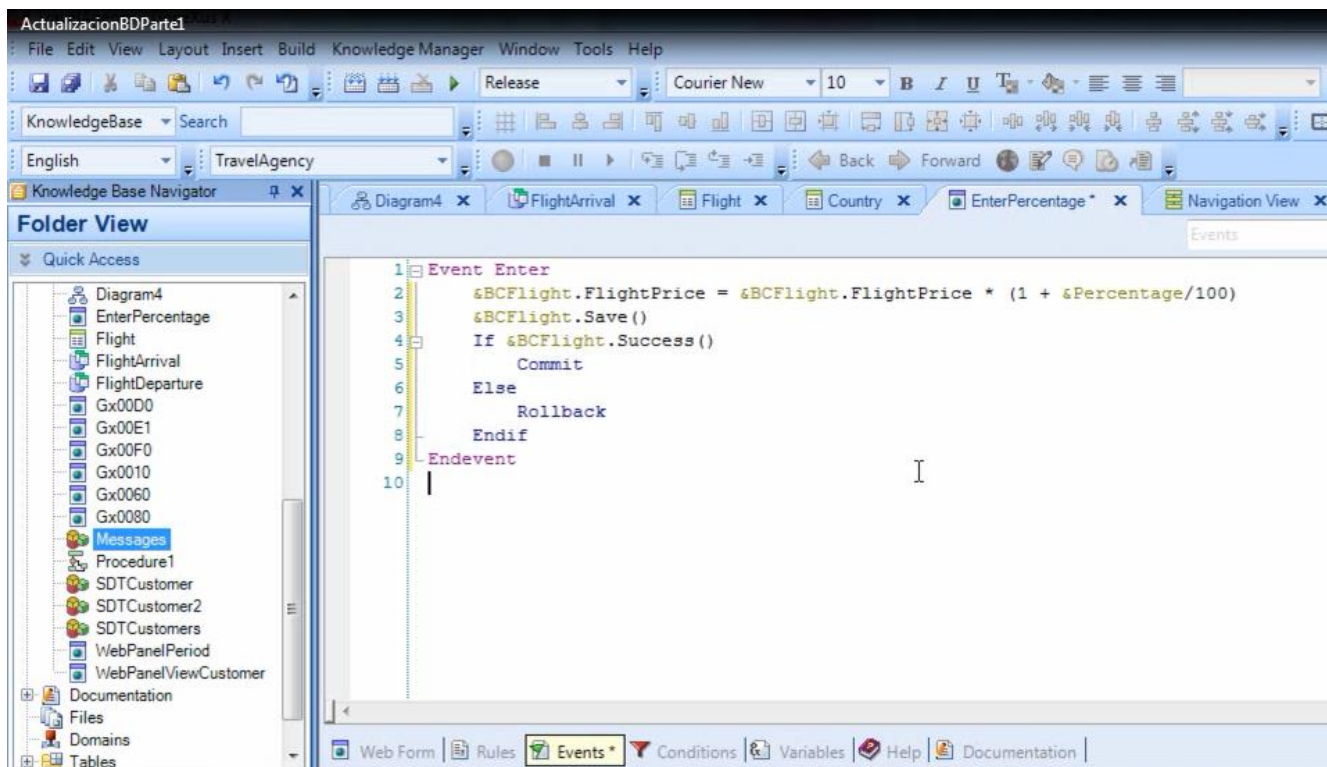
Nos resta ver como insertar un vuelo, usando business components.

Es igual a cuando actualizamos, con la única diferencia **de que no hay que efectuar Load**

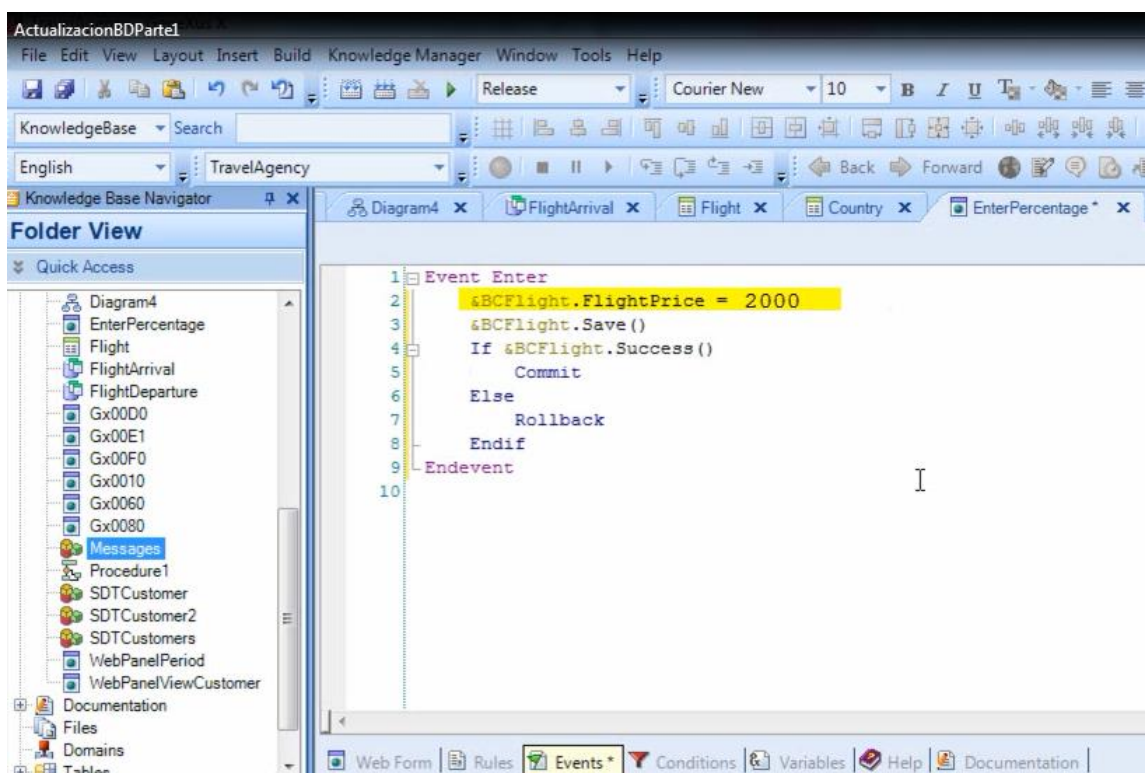


ya que no vamos a recuperar un vuelo, sino insertar uno nuevo.

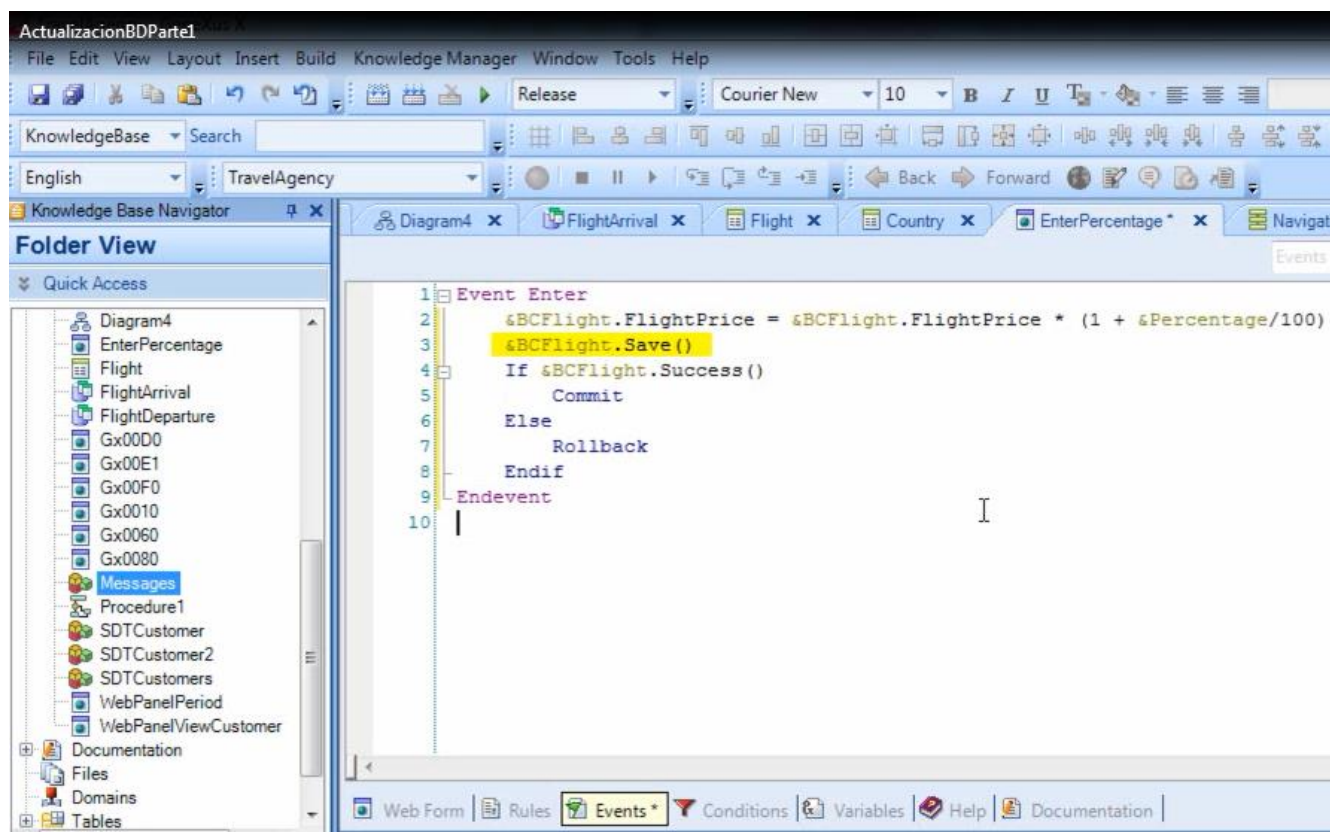
Quitemos el For each, endfor y la línea con el método Load



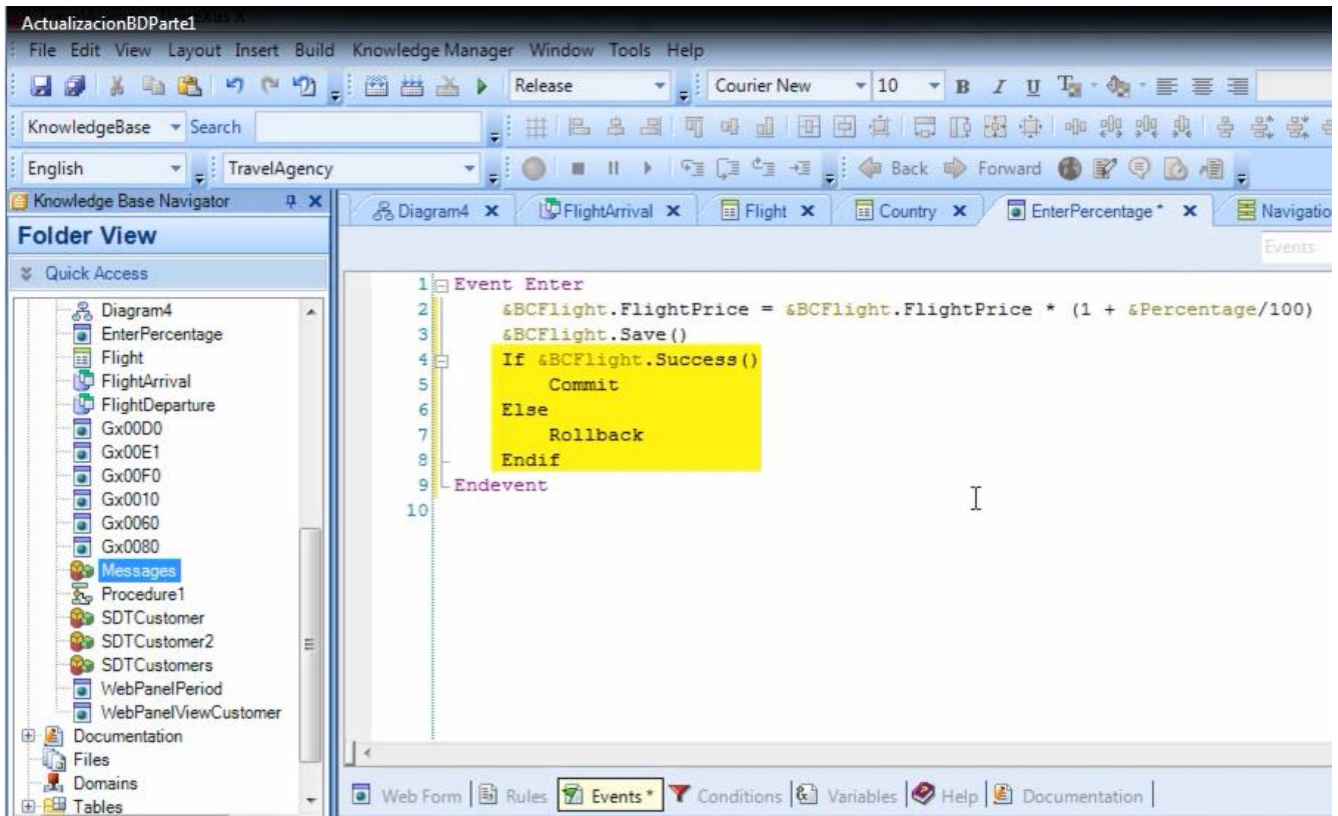
Así sea en un evento Enter de un objeto, o en el contexto que estemos... para insertar un registro empleando el concepto de business component, básicamente, hay que asignar valores a los atributos que nos interese



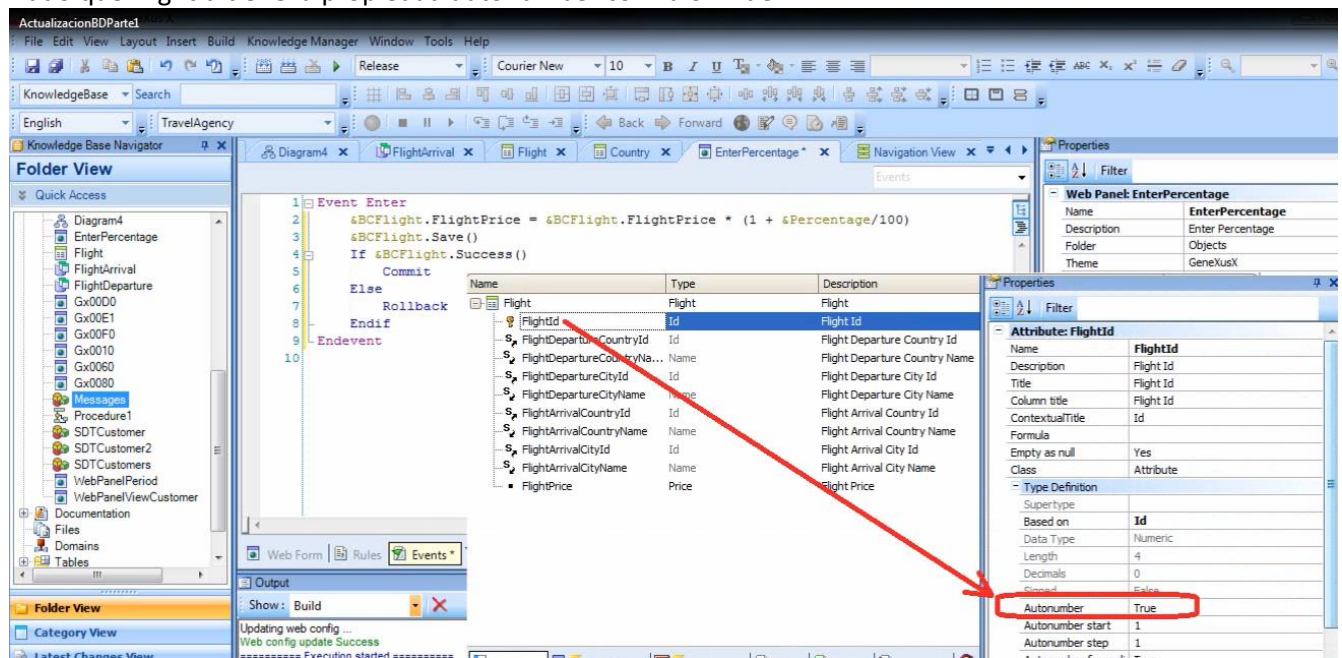
luego ejecutar el método save para la variable



y ejecutar el commit o rollback, en base al éxito o no de la operación, como venimos viendo

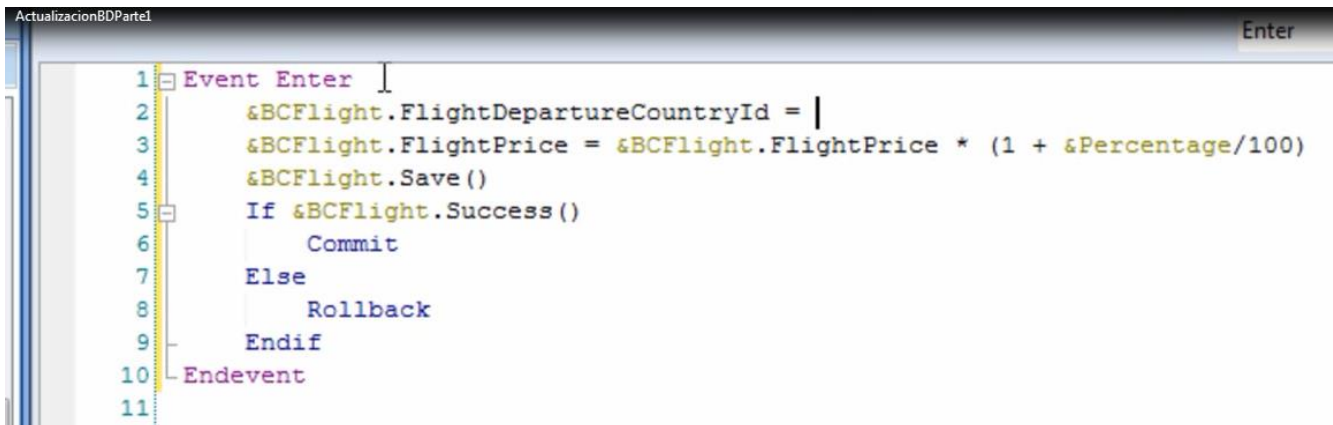


Dado que FlightId tiene la propiedad autonumber con valor True



no hay que asignarle valor a dicho atributo, del mismo modo que cuando operamos con el form de la transacción Flight.

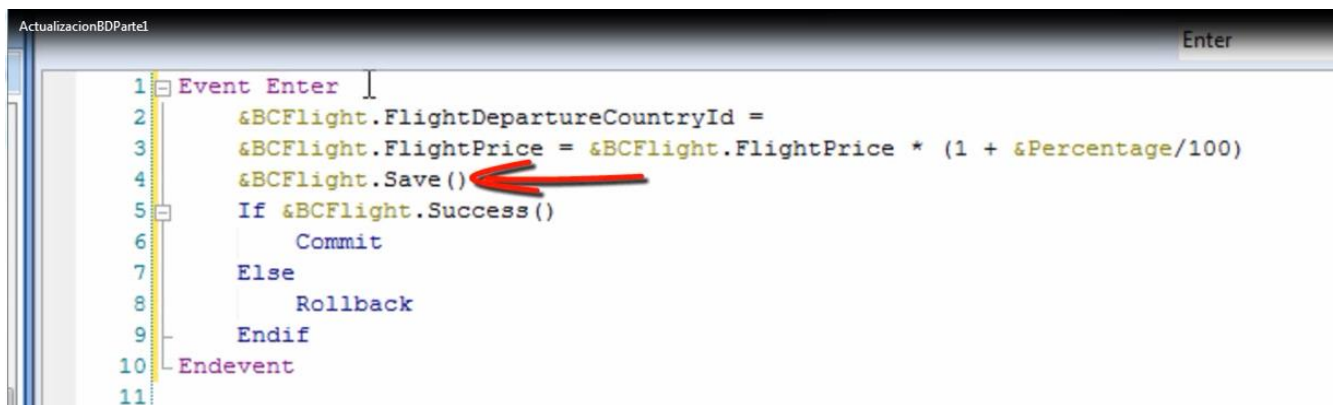
A su vez, si al nuevo vuelo que queremos insertar, le queremos asignar determinado país y ciudad de origen y determinado país y ciudad de destino.... escribimos como venimos haciendo &BCFlight, punto, elegimos FlightDepartureCountryId, le asignamos valor...



```
1 Event Enter |
2   &BCFlight.FlightDepartureCountryId = |
3   &BCFlight.FlightPrice = &BCFlight.FlightPrice * (1 + &Percentage/100)
4   &BCFlight.Save()
5   If &BCFlight.Success()
6     Commit
7   Else
8     Rollback
9   Endif
10 Endevent
11
```

y así a cada atributo que nos interese cargarle valores, hasta que llegue el momento de hacer save y commit o rollback.

El método save, como vemos,



```
1 Event Enter |
2   &BCFlight.FlightDepartureCountryId =
3   &BCFlight.FlightPrice = &BCFlight.FlightPrice * (1 + &Percentage/100)
4   &BCFlight.Save()
5   If &BCFlight.Success()
6     Commit
7   Else
8     Rollback
9   Endif
10 Endevent
11
```

se usa para grabar tanto una inserción de un nuevo registro, como para grabar una actualización.

Si primero cargamos en memoria un registro,

```
ActualizacionBDParte1 Enter
1 Event Enter |
2   &BCFlight.FlightDepartureCountryId =
3   &BCFlight.FlightPrice = &BCFlight.FlightPrice * (1 + &Percentage/100)
4   &BCFlight.Save()
5   If &BCFlight.Success()
6     Commit
7   Else
8     Rollback
9   Endif
10 Endevent
11
```

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

le asignamos valores

```
ActualizacionBDParte1 Enter
1 Event Enter |
2   &BCFlight.FlightDepartureCountryId = |
3   &BCFlight.FlightPrice = &BCFlight.FlightPrice * (1 + &Percentage/100)
4   &BCFlight.Save()
5   If &BCFlight.Success()
6     Commit
7   Else
8     Rollback
9   Endif
10 Endevent
11
```

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

y hacemos Save...

```
ActualizacionBDParte1 Enter
1 Event Enter
2   &BCFlight.FlightDepartureCountryId =
3   &BCFlight.FlightPrice = &BCFlight.FlightPrice * (1 + &Percentage/100)
4   &BCFlight.Save()
5   If &BCFlight.Success()
6     Commit
7   Else
8     Rollback
9   Endif
10 Endevent
11
```

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

GeneXus entiende que es una actualización.

Y si, asignamos valores a los atributos

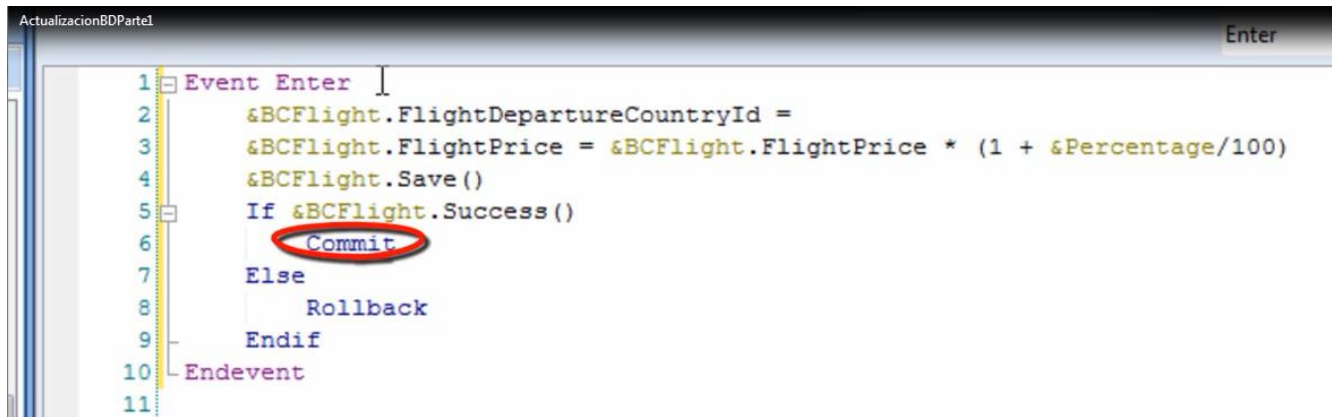
```
ActualizacionBDParte1 Enter
1 Event Enter
2   &BCFlight.FlightDepartureCountryId = 1
3   &BCFlight.FlightPrice = &BCFlight.FlightPrice * (1 + &Percentage/100)
4   &BCFlight.Save()
5   If &BCFlight.Success()
6     Commit
7   Else
8     Rollback
9   Endif
10 Endevent
```

+ save

```
ActualizacionBDParte1 Enter
1 Event Enter
2   &BCFlight.FlightDepartureCountryId = 1
3   &BCFlight.FlightPrice = &BCFlight.FlightPrice * (1 + &Percentage/100)
4   &BCFlight.Save()
5   If &BCFlight.Success()
6     Commit
7   Else
8     Rollback
9   Endif
10 Endevent
11
```

y GeneXus entiende que es una inserción.

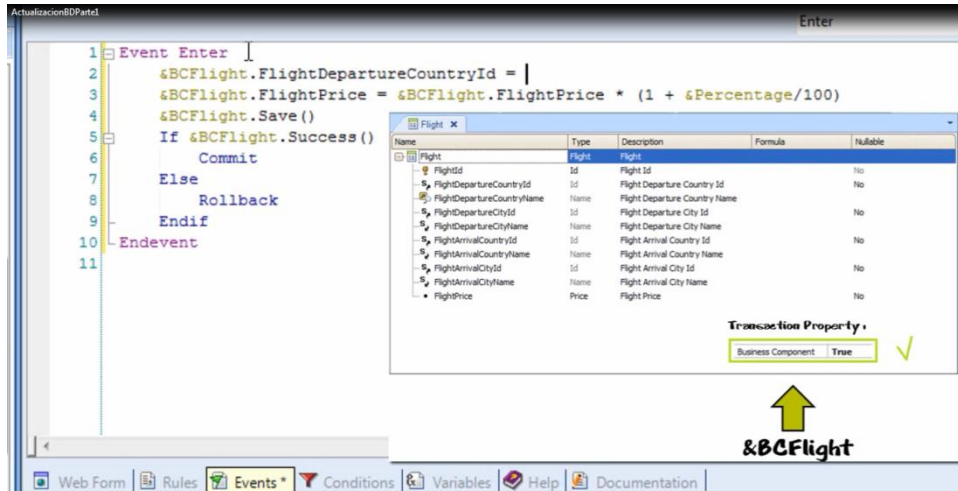
No podemos olvidar el commit.



```
1 Event Enter |
2   &BCFlight.FlightDepartureCountryId =
3   &BCFlight.FlightPrice = &BCFlight.FlightPrice * (1 + &Percentage/100)
4   &BCFlight.Save()
5   If &BCFlight.Success()
6     Commit
7   Else
8     Rollback
9   Endif
10 Endevent
11
```

Así hemos visto otra forma de actualizar la base de datos, en particular usando business components.

Como ya hemos dicho, si una transacción es definida como business component



Name	Type	Description	Formula	Nullable
Flight	Flight	Flight		
FlightId	Id	Flight Id		No
FlightDepartureCountryId	Id	Flight Departure Country Id		No
FlightDepartureCountryName	Name	Flight Departure Country Name		No
FlightDepartureCityId	Id	Flight Departure City Id		No
FlightDepartureCityName	Name	Flight Departure City Name		No
FlightArrivalCountryId	Id	Flight Arrival Country Id		No
FlightArrivalCountryName	Name	Flight Arrival Country Name		No
FlightArrivalCityId	Id	Flight Arrival City Id		No
FlightArrivalCityName	Name	Flight Arrival City Name		No
FlightPrice	Price	Flight Price		No

Transaction Property:
Business Component: True ✓

↑
&BCFlight

en cualquier objeto se podrá definir una variable de dicho tipo, para actualizar la base de datos desde allí.

Nos queda ver una alternativa más para actualizar la base de datos con ciertos comandos. Lo veremos en el próximo video y tenemos que saber que eso solamente se podrá usar en objetos procedimientos.