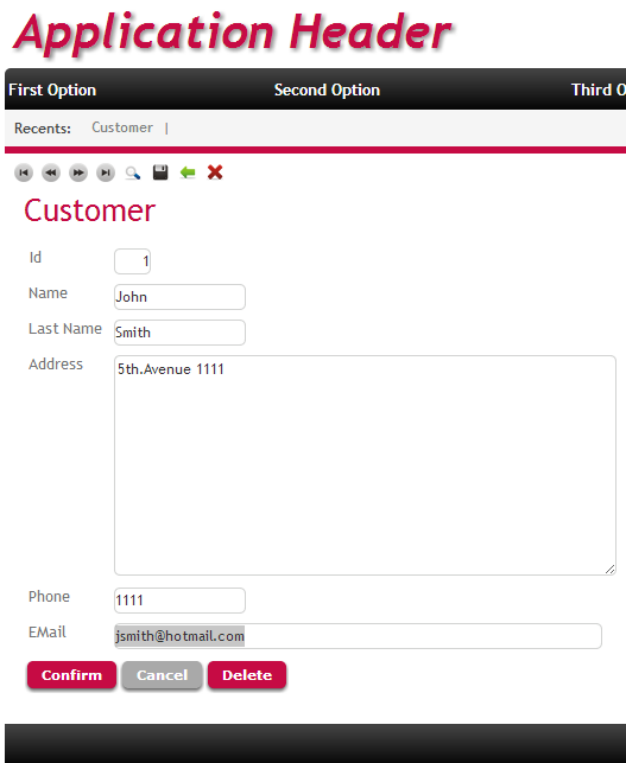
Actualización de la Base de Datos - Business Components

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.

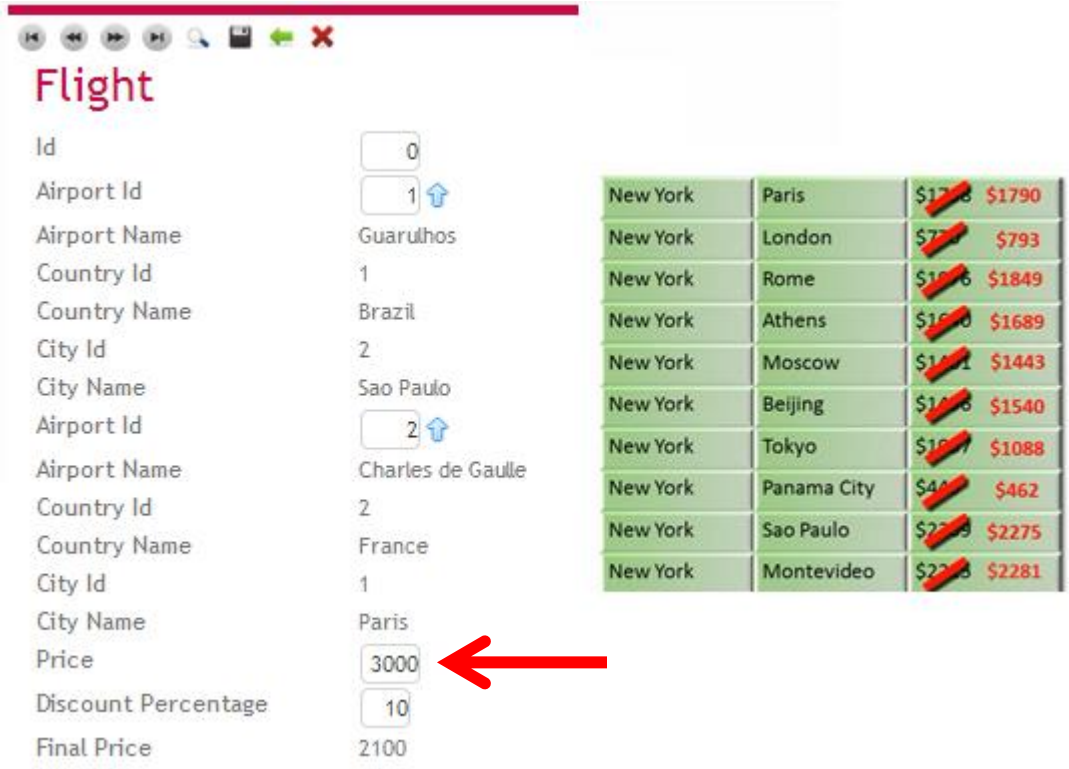


Además de este medio para ir actualizando la base de datos, en ocasiones resulta necesario definir algún proceso de actualización automático.
A modo de ejemplo, la agencia de viajes podría necesitar cada determinado tiempo, aumentar los precios de **todos los vuelos registrados**,

aplicándole a los precios vigentes, determinado porcentaje de aumento.

New York	Paris	[illegible]	$1790
New York	London	[illegible]	$793
New York	Rome	[illegible]	$1849
New York	Athens	[illegible]	$1689
New York	Moscow	[illegible]	$1443
New York	Beijing	[illegible]	$1540
New York	Tokyo	[illegible]	$1088
New York	Panama City	[illegible]	$462
New York	Sao Paulo	[illegible]	$2275
New York	Montevideo	[illegible]	$2281

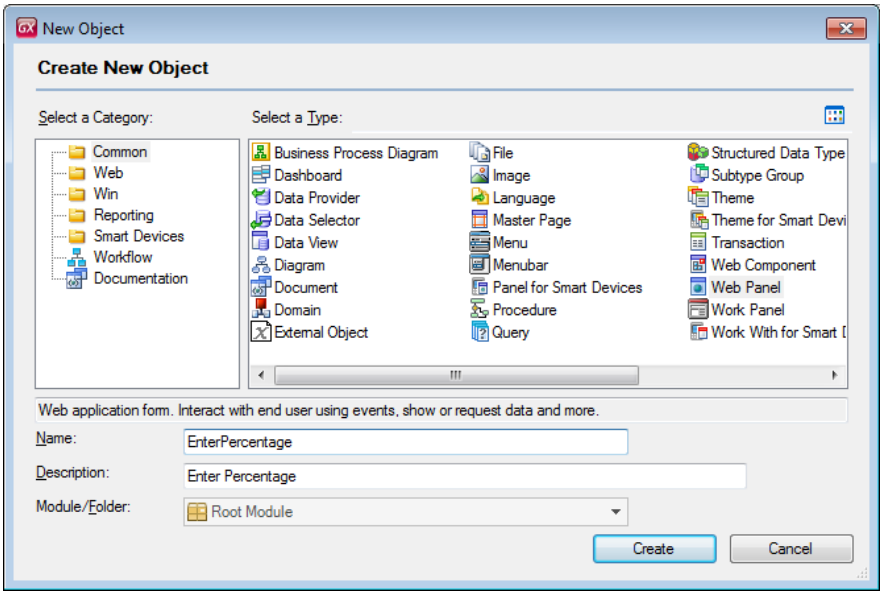
Resultaría muy tedioso utilizar la pantalla de la transacción para editar manualmente cada precio de los cientos de vuelos registrados,

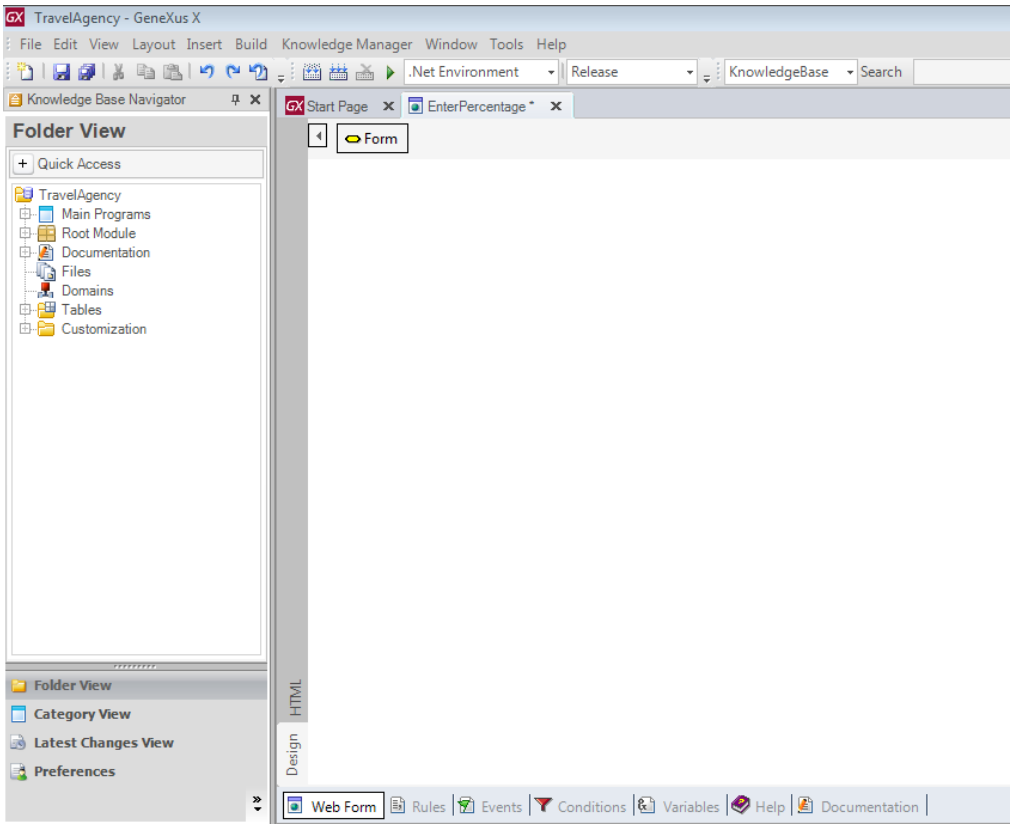


por lo que nos interesa que esto lo haga un proceso masivo.

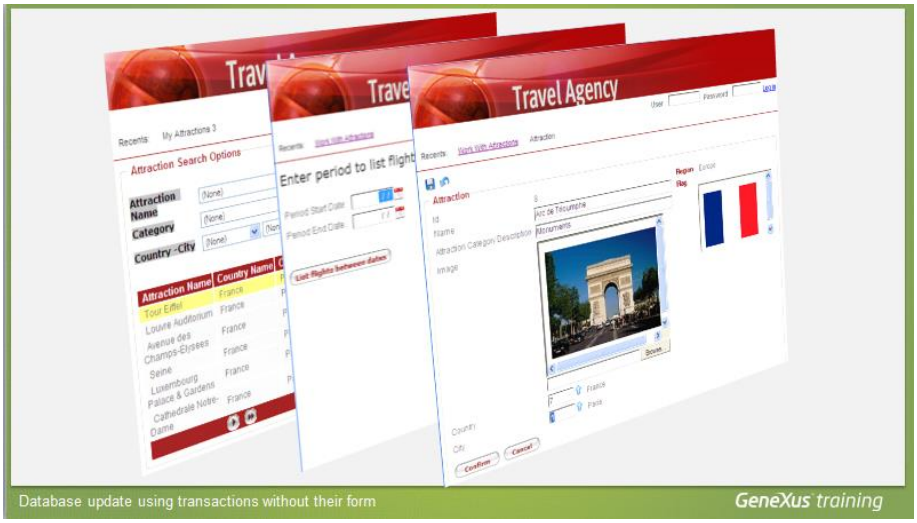
Para resolver este ejemplo que planteamos, lo primero que debemos hacer es solicitarle al usuario el porcentaje de aumento que desea aplicarle a los precios de los vuelos. Para eso, vamos a crear un objeto de tipo web panel.

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.



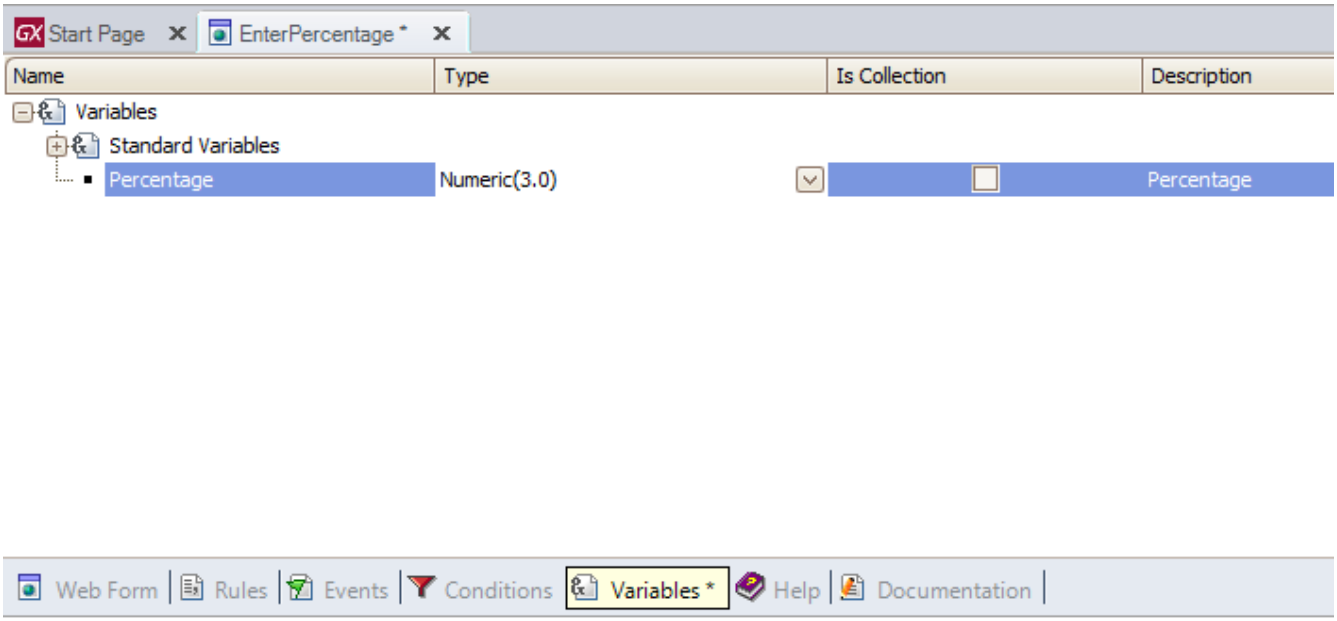
Si incluimos variables en el form de un web panel, las mismas tendrán comportamiento “de entrada de datos”, es decir que el usuario podrá ingresarles valor.

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

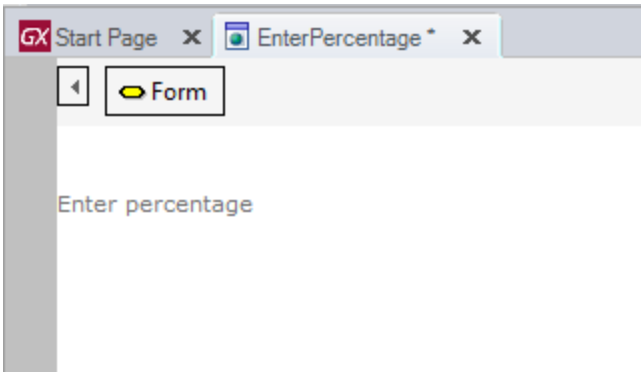
los mismos serán “de salida”, o sea que no serán editables sino que solamente se mostrarán los datos que dichos atributos tengan almacenados en la base de datos.

Vamos a definir entonces una variable en este web panel.

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

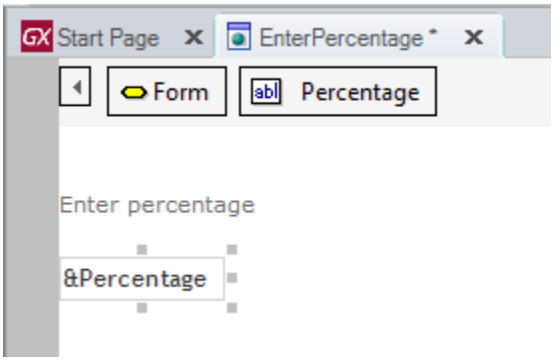


Ahora volvemos al form y digitamos directamente dentro del mismo: “Enter percentage”

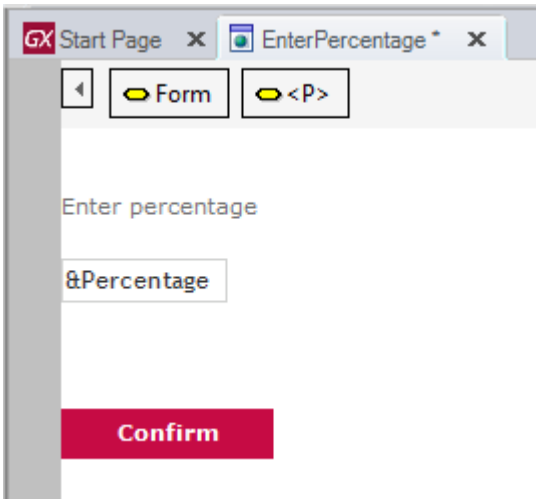


Presionamos enter y debajo del texto insertaremos la variable &percentage

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

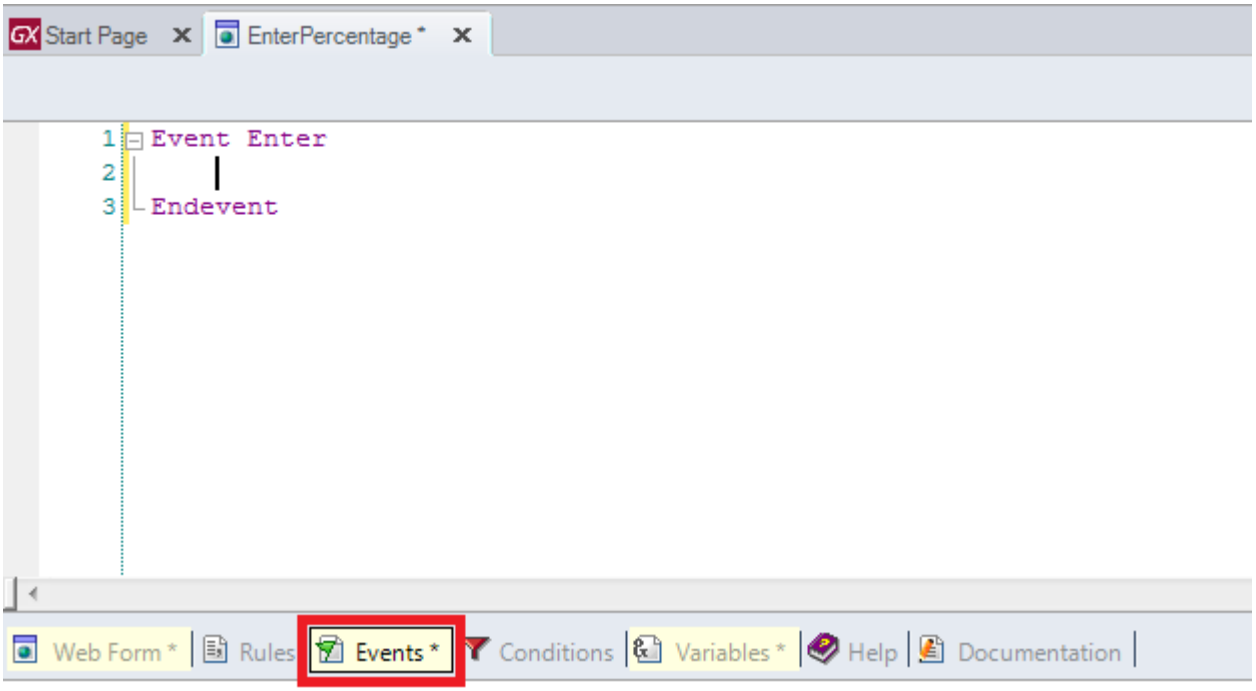


Y ahora arrastramos un botón al form desde la toolbox



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

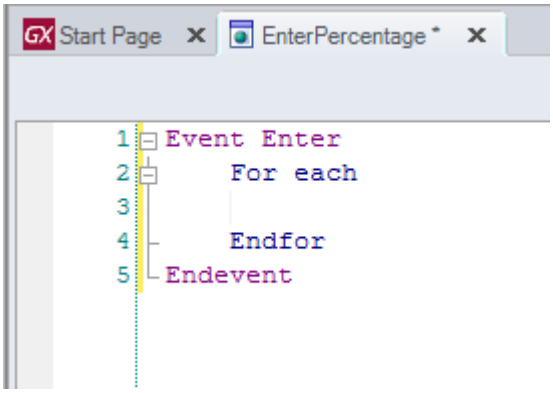


y en particular el cursor se posiciona dentro del evento asociado al botón: **el evento Enter**.

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.

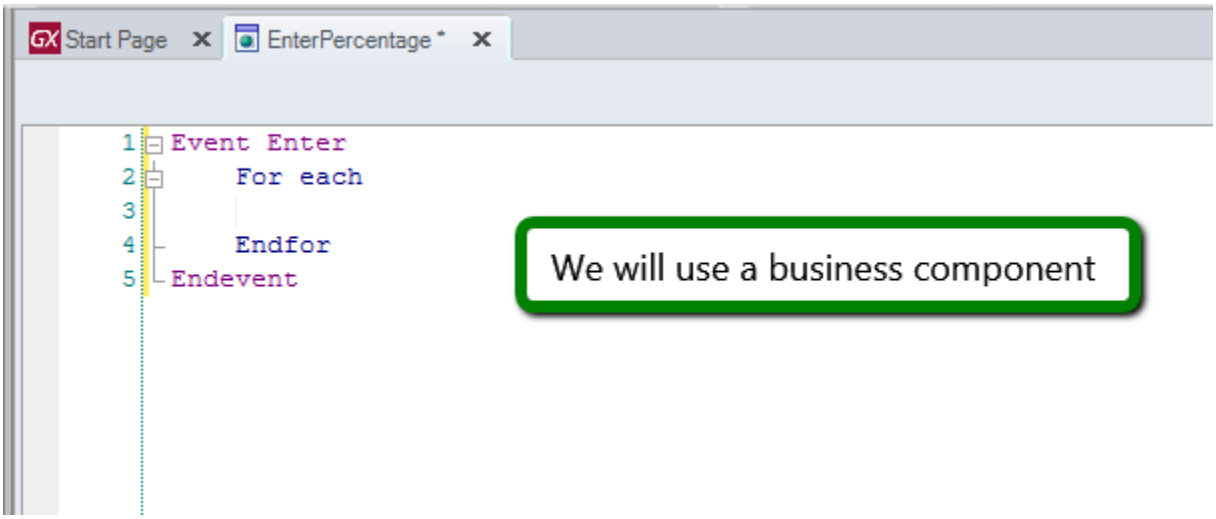
Vamos a escribir entonces, un comando For each con su correspondiente Endfor, para navegar la tabla FLIGHT



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.

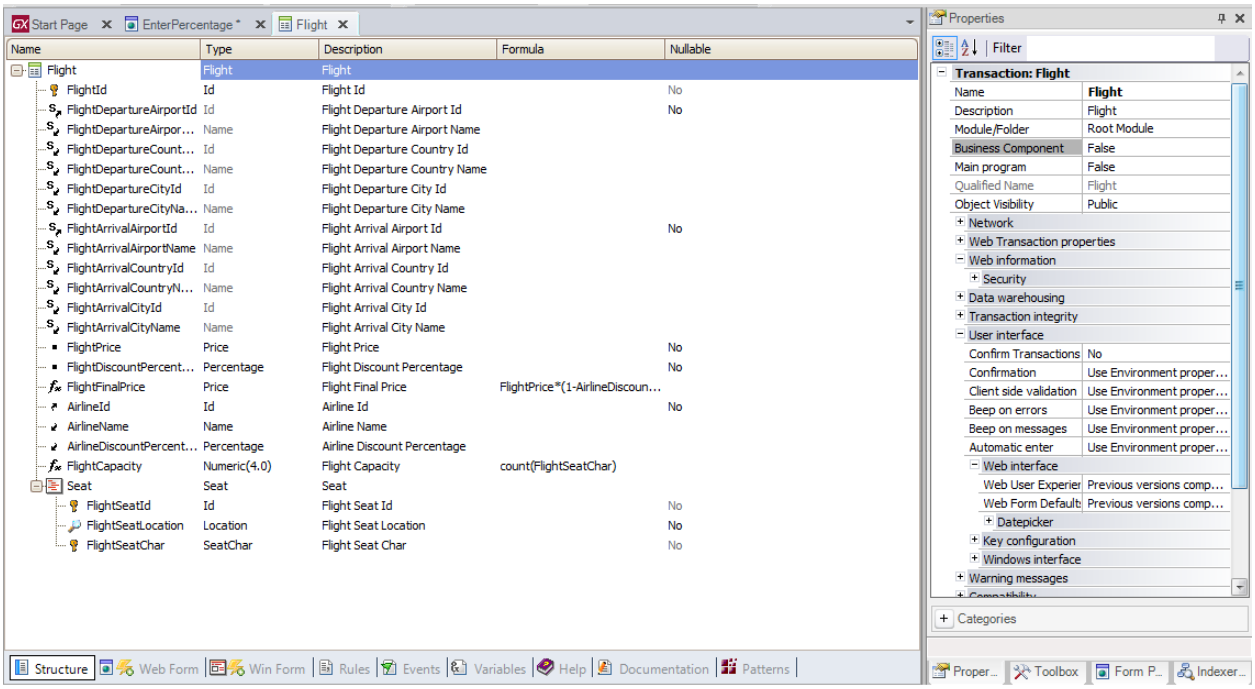
¿Qué atributo podremos referenciar? ¿y cómo hacemos para actualizar el precio de cada vuelo?

<u>Para actualizar la base de datos en un web panel, solamente contamos con una posibilidad</u>… y es empleando el concepto de business component

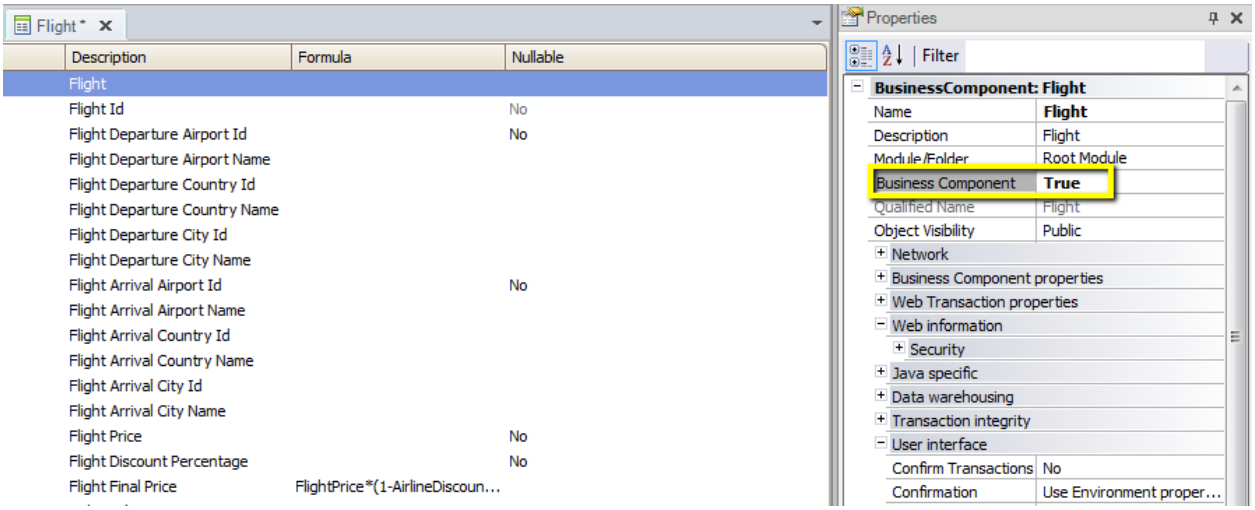


Veamos de qué se trata un business component y cómo se usa.

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



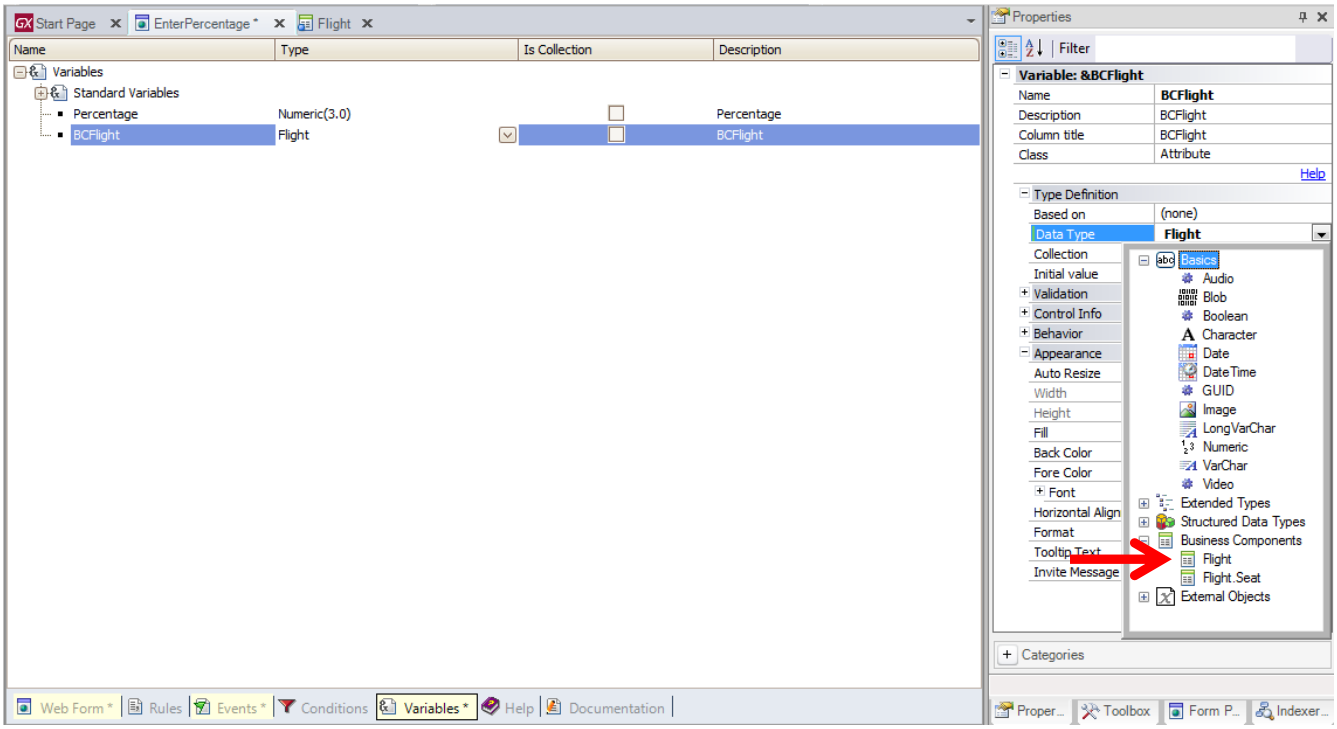
Vemos que hay una propiedad de nombre Business Component



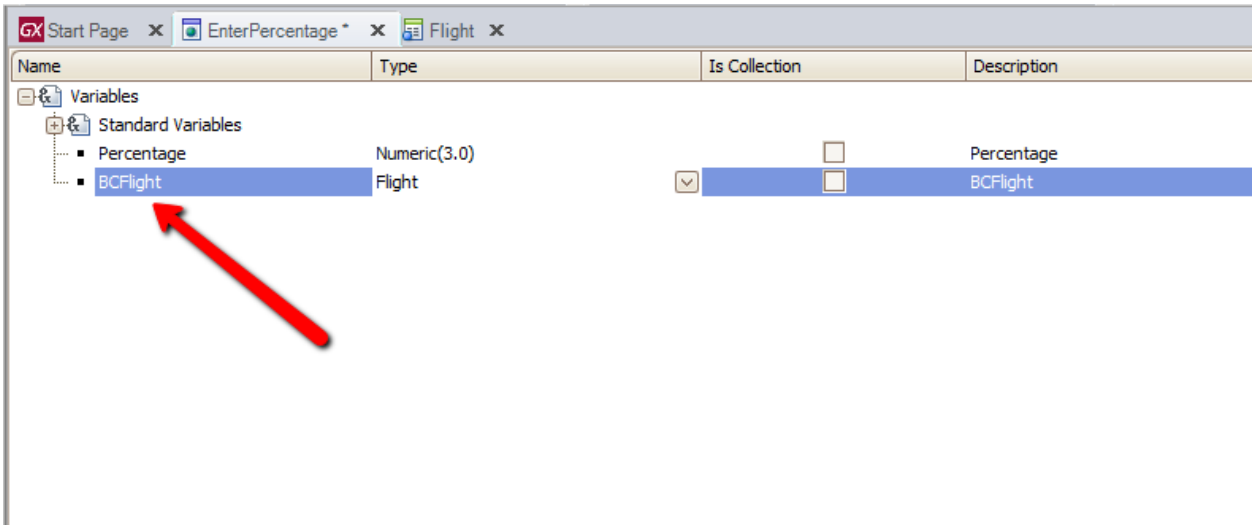
y la configuraremos con valor True:

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.

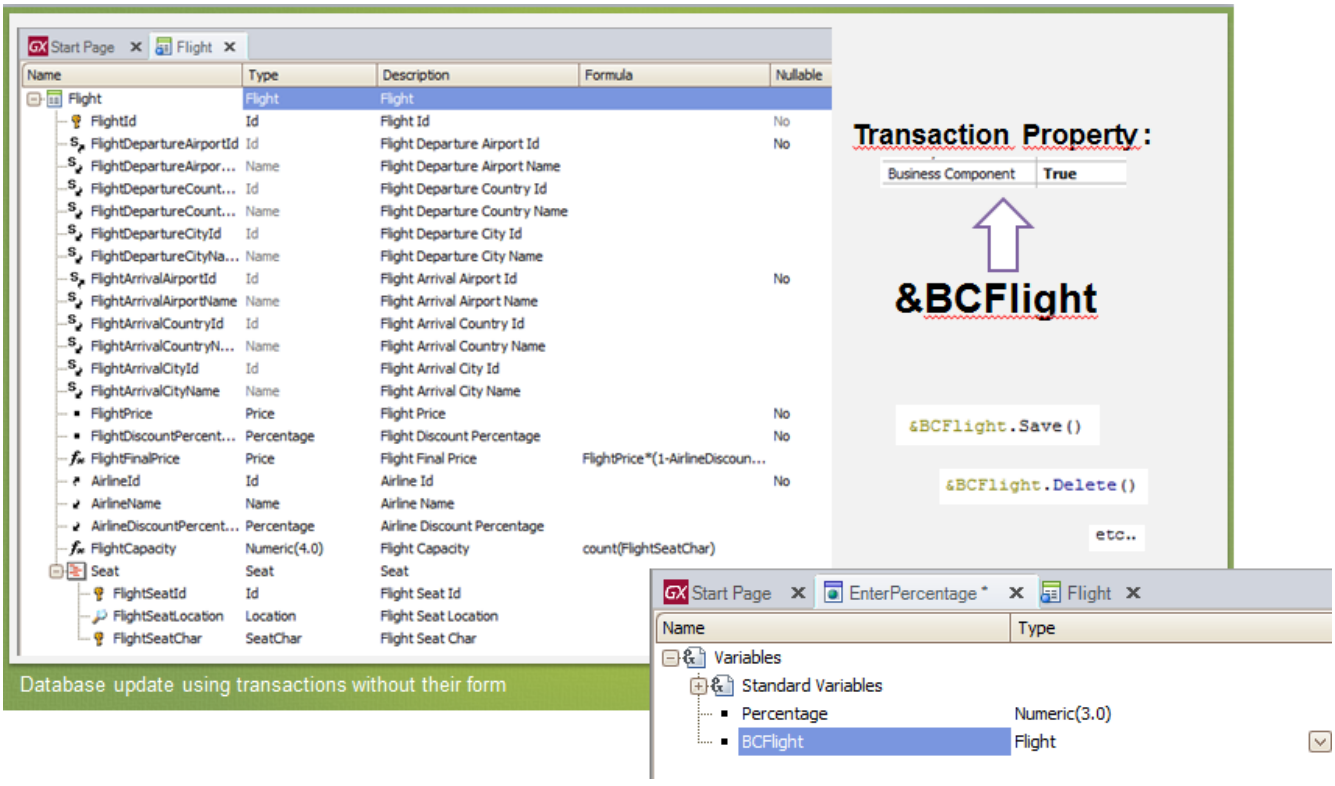
Cuando a una transacción le configuramos su propiedad Business Component con valor True, **en cualquier objeto de la base de conocimiento, podremos definir una variable del tipo business component de la transacción definida como tal…**



Empleando esta variable

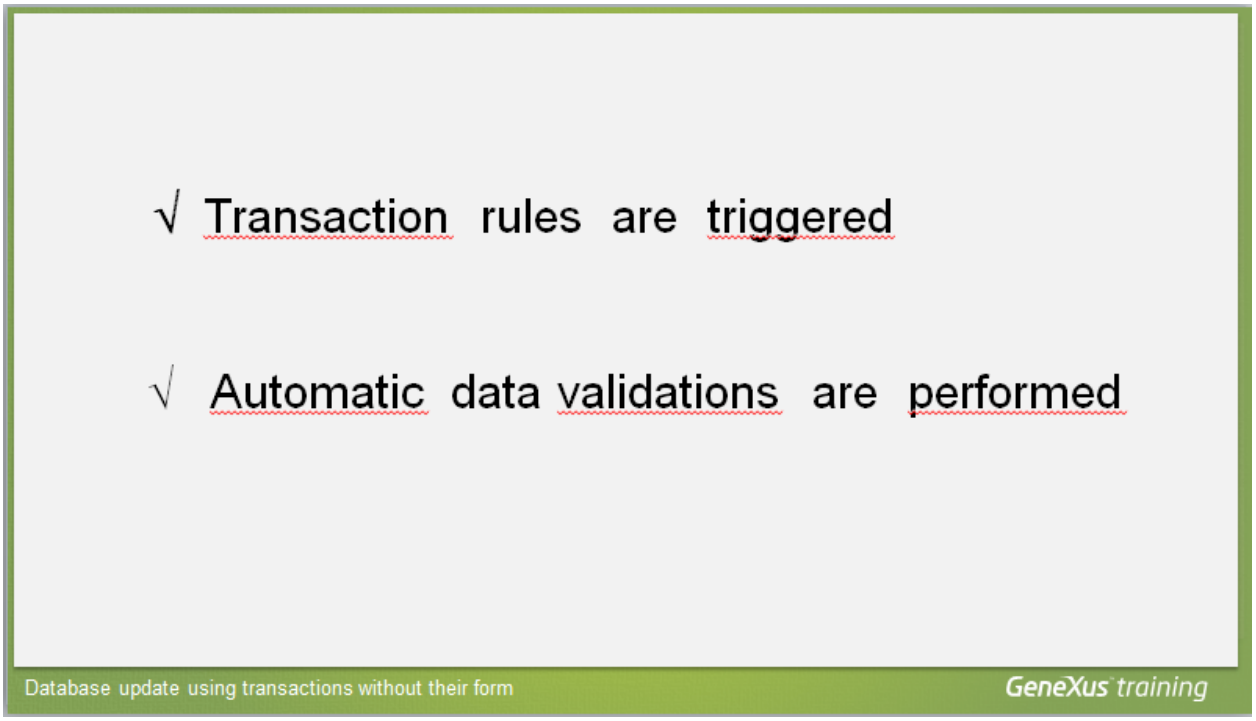


podremos ejecutar a la transacción Flight <u>sin su form</u> para realizar actualizaciones de la base de datos:



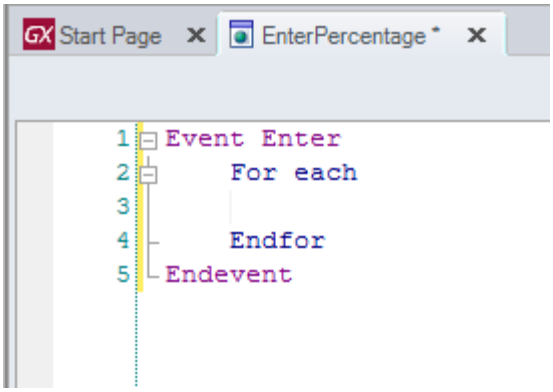
Además, cuando utilicemos esta variable definida del tipo business component de una transacción,

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

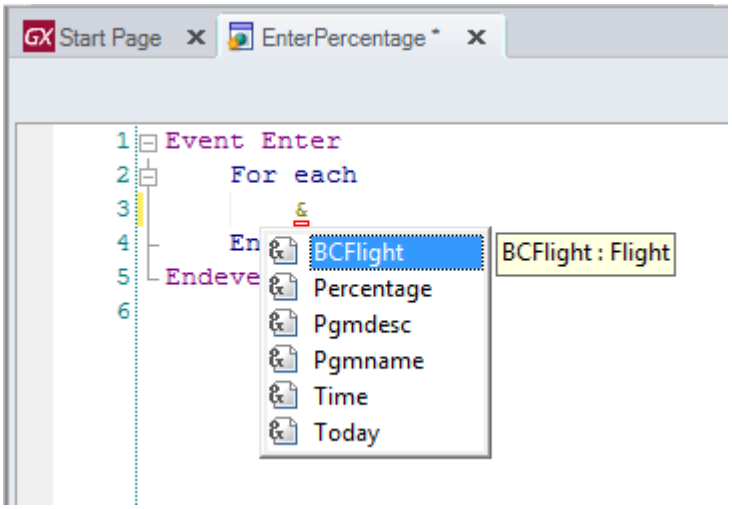


Ahora bien, volvamos al objeto que estabamos codificando, para aprender a utilizar variables del tipo business components-

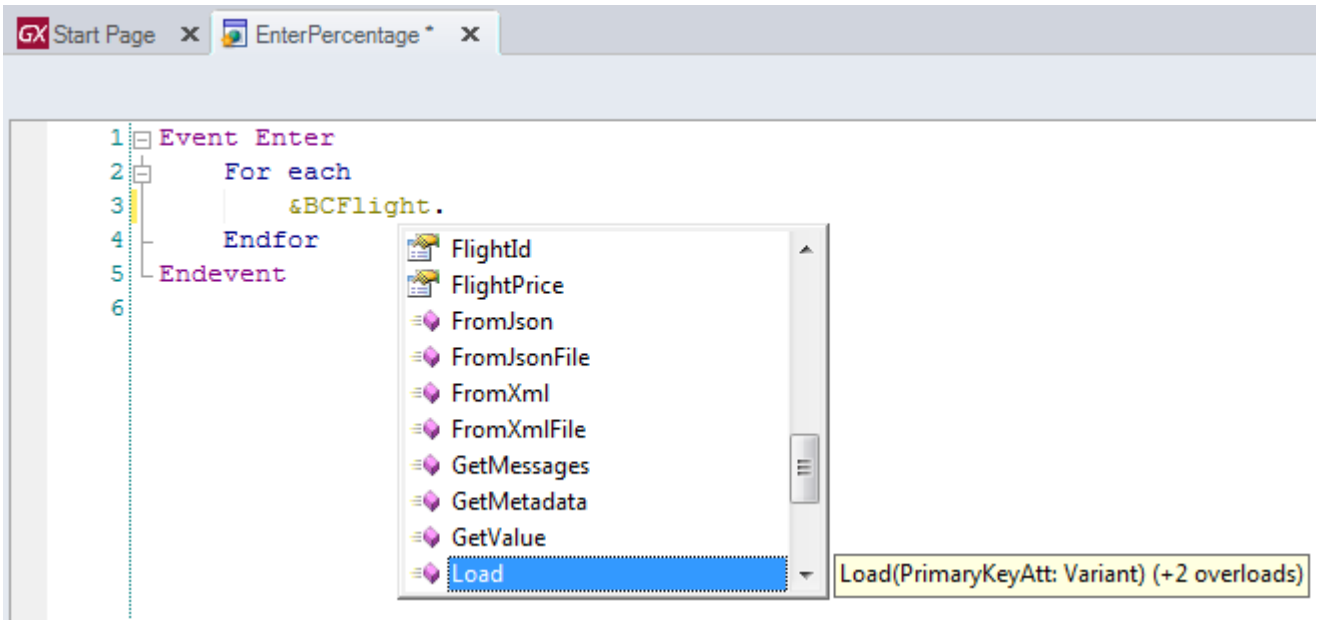
En el web panel “EnterPercentage”, dentro del cuerpo del For each que estábamos codificando, vamos a escribir la primer instrucción.



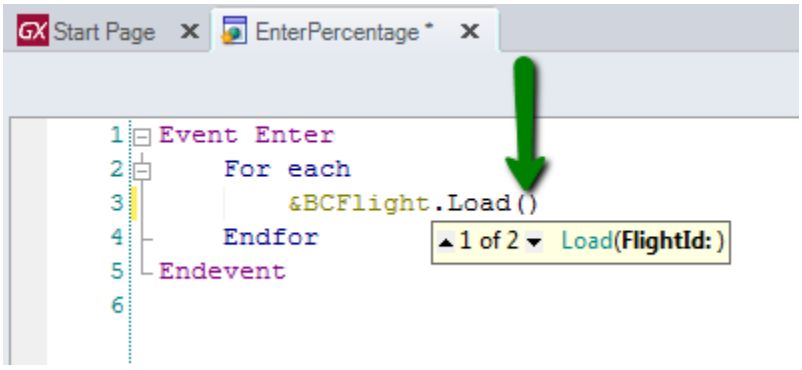
Escribimos ampersand



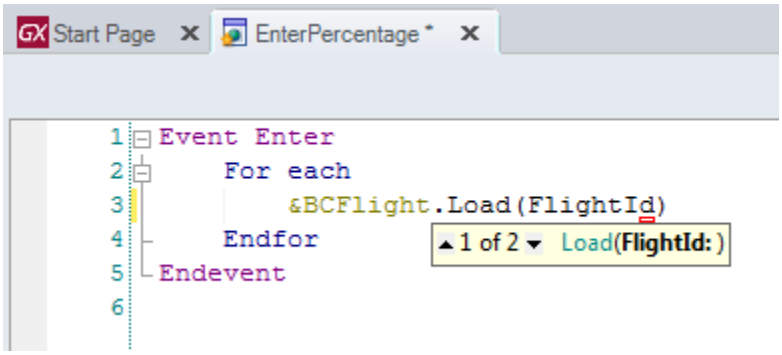
y seleccionamos la variable &BCFlight que hemos definido. Ahora digitamos punto y elegimos el método Load.



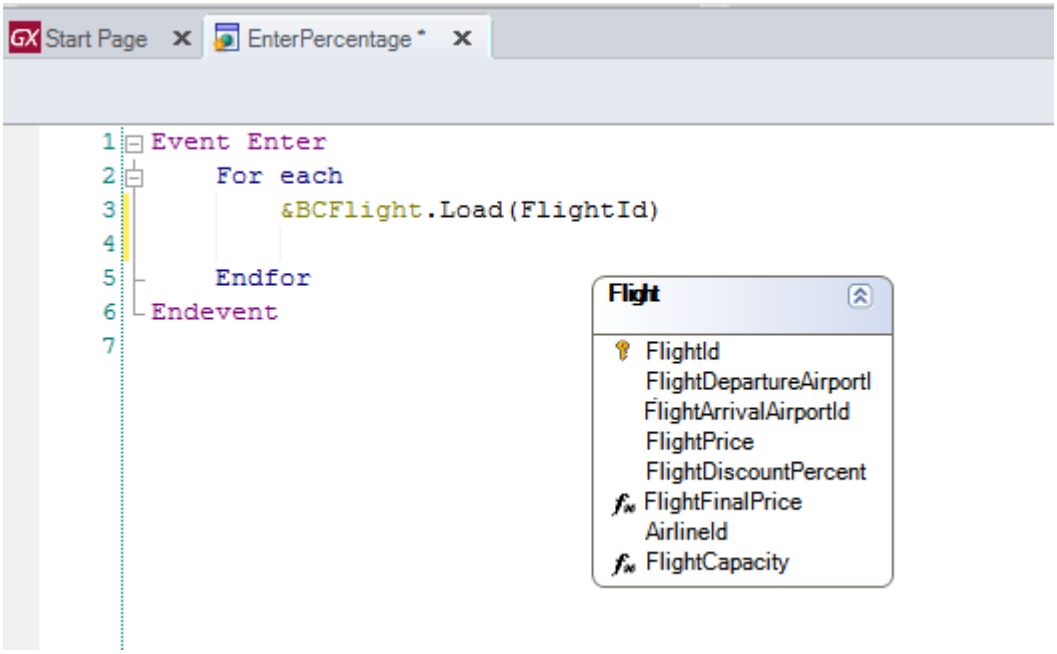
El método Load, tal como su nombre lo describe, permite **cargar** en memoria, los datos correspondientes al valor de llave primaria que indiquemos dentro del paréntesis.



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.**

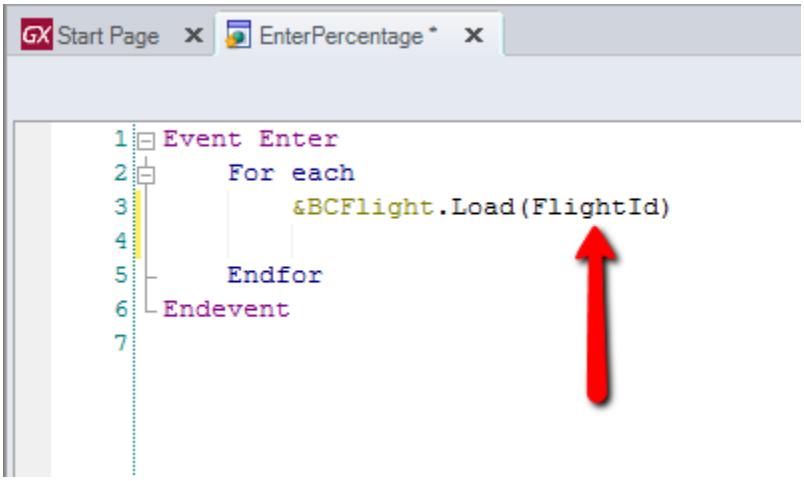


En este caso, estamos cargando en memoria un vuelo, y el valor de la llave primaria lo tenemos en el atributo FlightId ya que nuestro objetivo es navegar la tabla FLIGHT mediante el For each

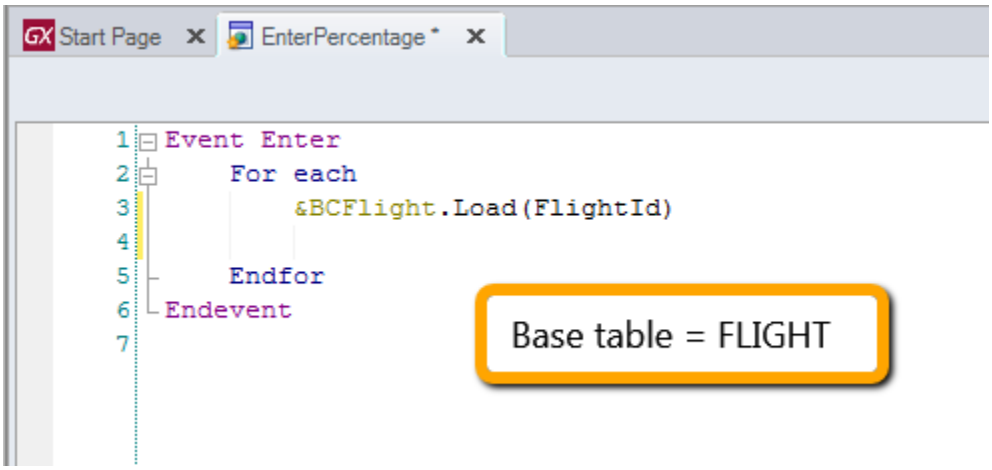


y para cada vuelo navegado, tendremos los valores de sus atributos.

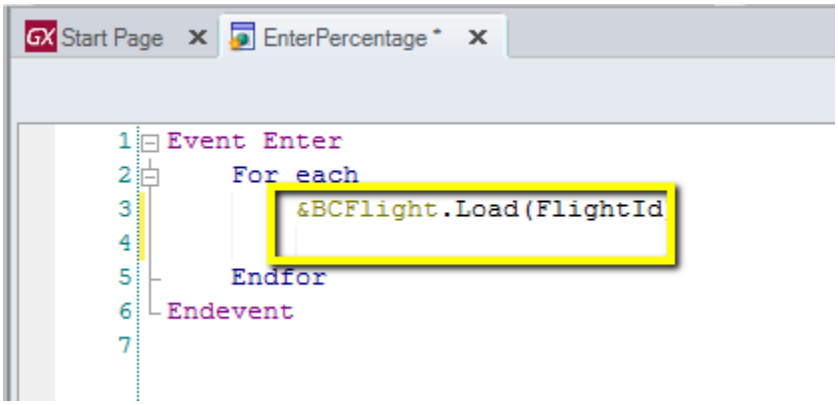
A propósito de que nuestro objetivo es navegar la tabla FLIGHT, observemos que este atributo FlightId



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

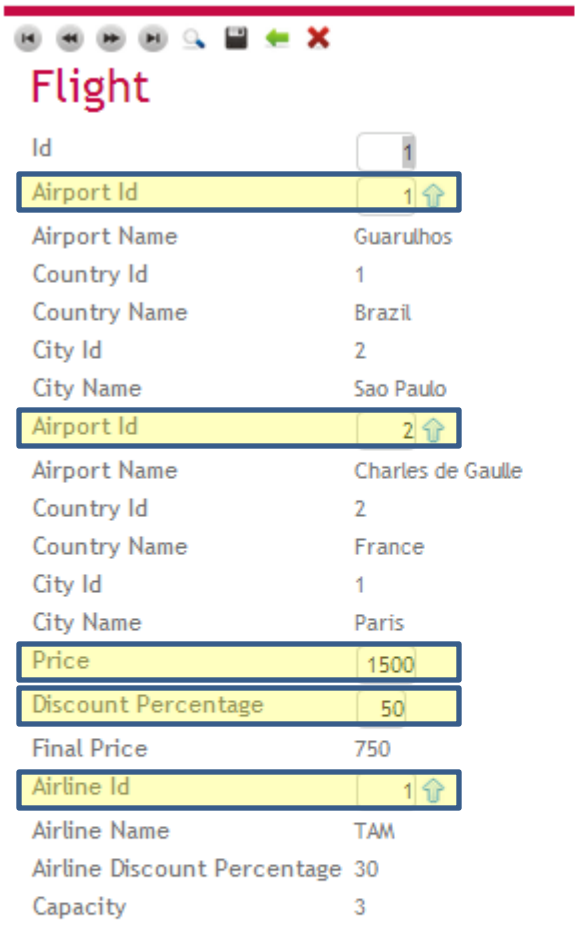


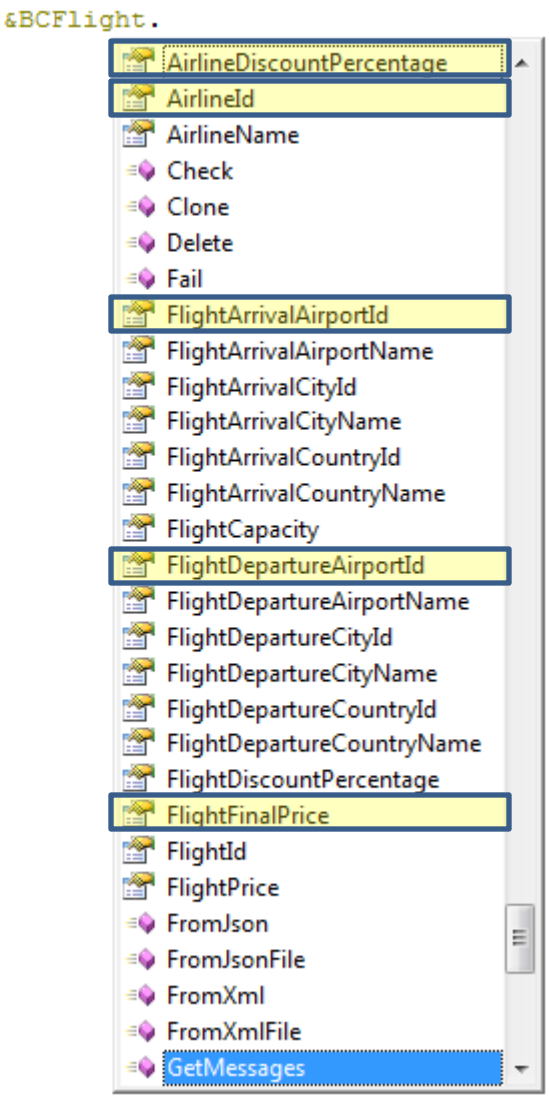
La primer sentencia entonces, efectúa algo equivalente a lo que sucede cuando digitamos en la transacción un valor de identificador de vuelo y salimos del campo.

Se cargan en memoria todos los datos correspondientes a ese identificador

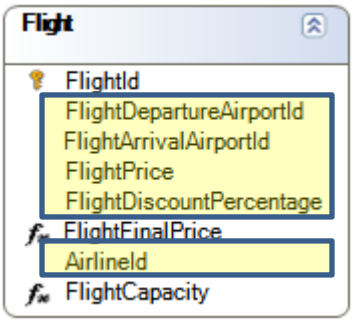
y los tendremos disponibles en este caso, en la variable &BCFlight.

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





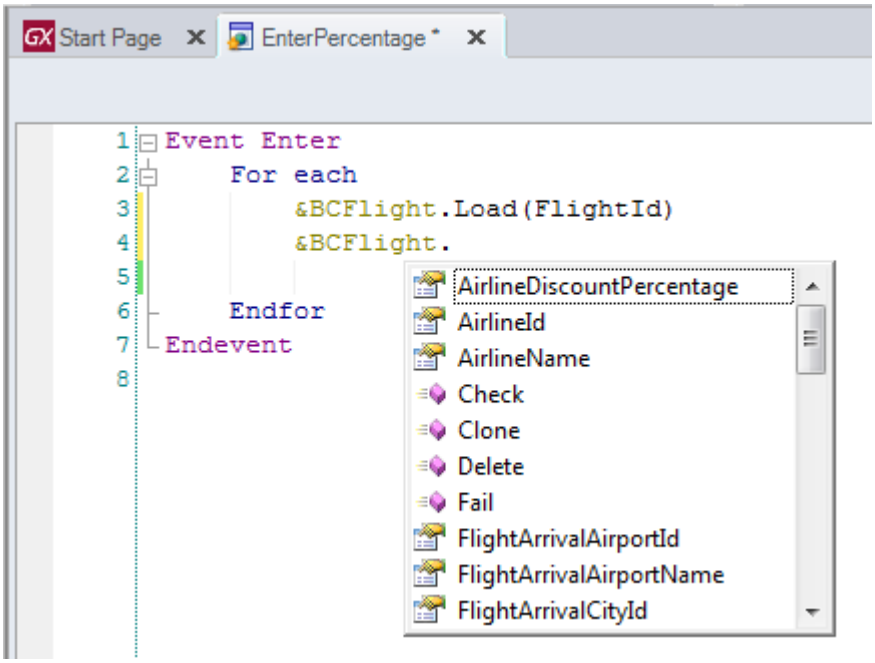
es decir, los que se encuentran en la tabla base asociada a la transacción, en este caso, FLIGHT



Volvamos ahora al código que estábamos escribiendo.

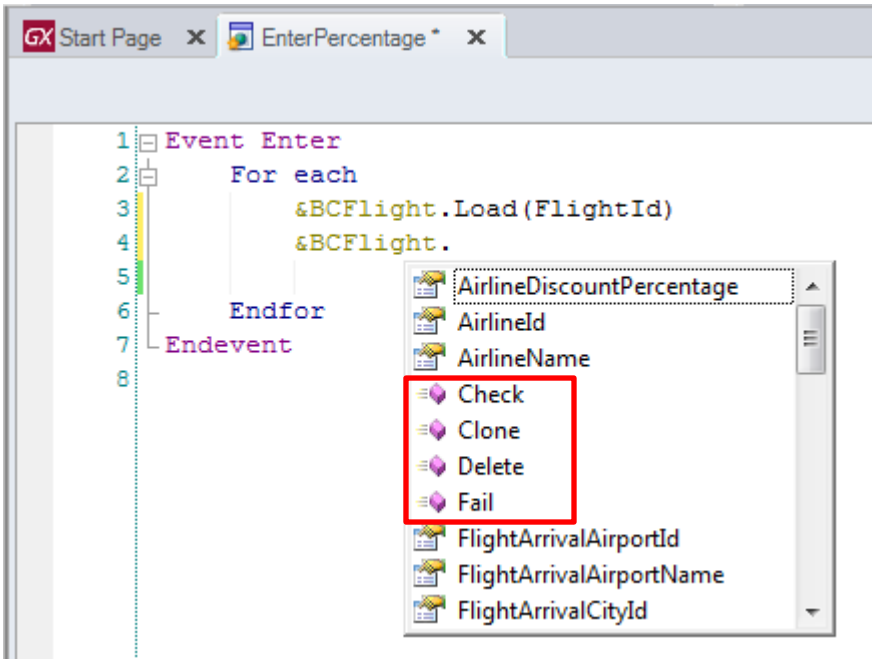
Como la primer instrucción del For each, **cargó los datos del vuelo navegado en memoria,** ahora solamente nos resta modificar el precio del vuelo y grabar los cambios.

Vamos a escribir la 2da instrucción dentro del For each. Digitamos ampersand, seleccionamos la variable &BCFlight, digitamos punto,

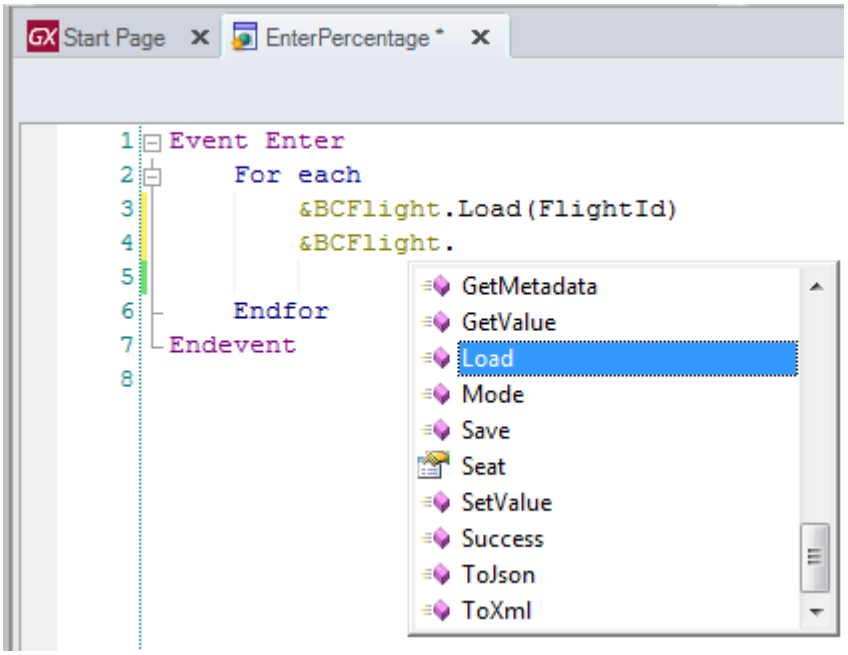


y observemos que tenemos 2 tipo se elementos que podemos elegir.

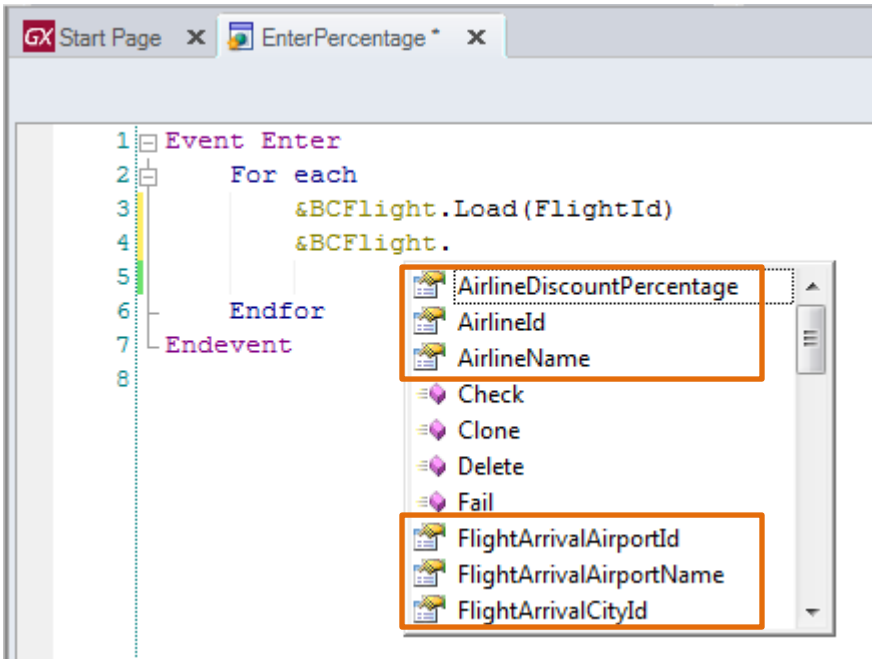
Los que tienen un ícono color violeta



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

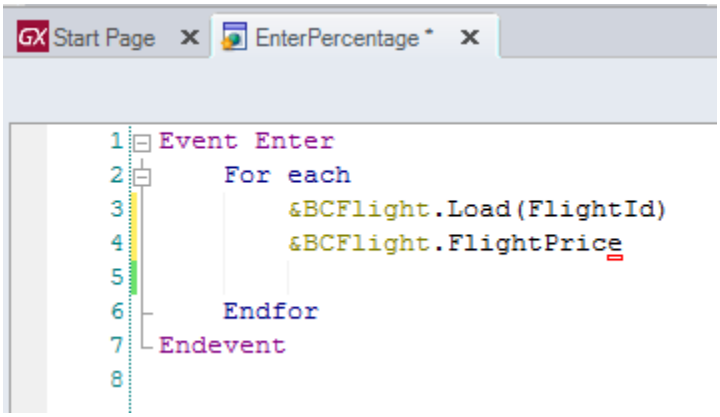


y los que tienen el ícono con la mano

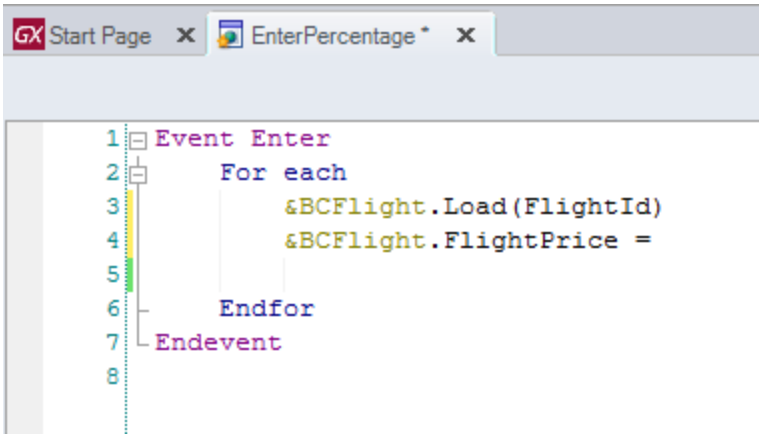


son los atributos presentes en la estructura de la transacción que es business component y de cuyo tipo es la variable &BCFlight

Seleccionamos el atributo FlightPrice

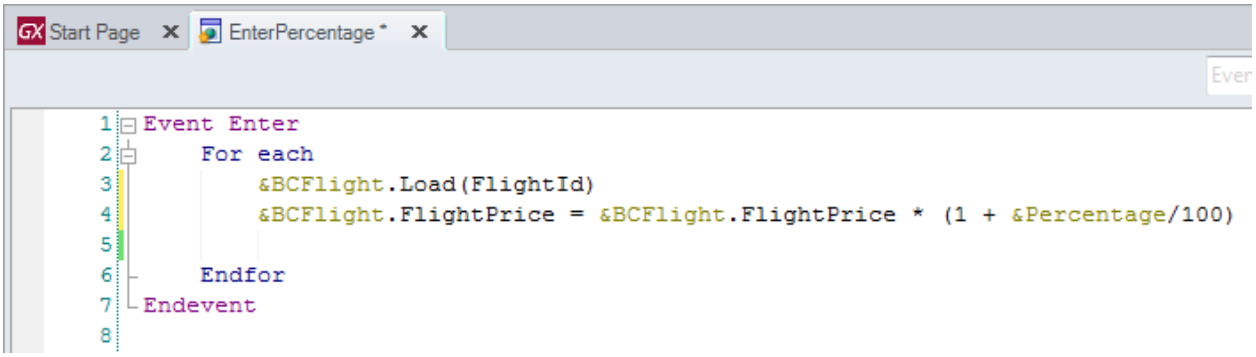


y para asignarle valor al atributo, digitamos signo de igual

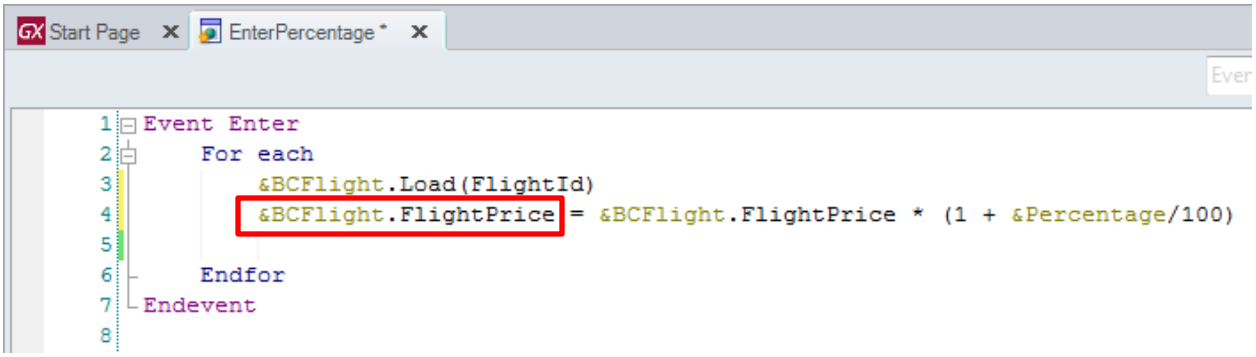


Ahora a la derecha del signo, debemos definir el cálculo para obtener el nuevo precio del vuelo, con el porcentaje de aumento aplicado.

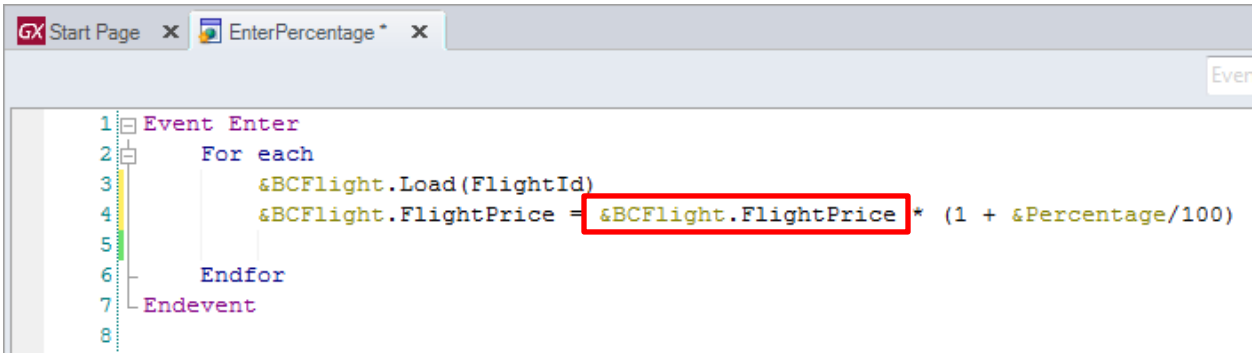
Vamos a escribir el cálculo y en seguida lo explicaremos



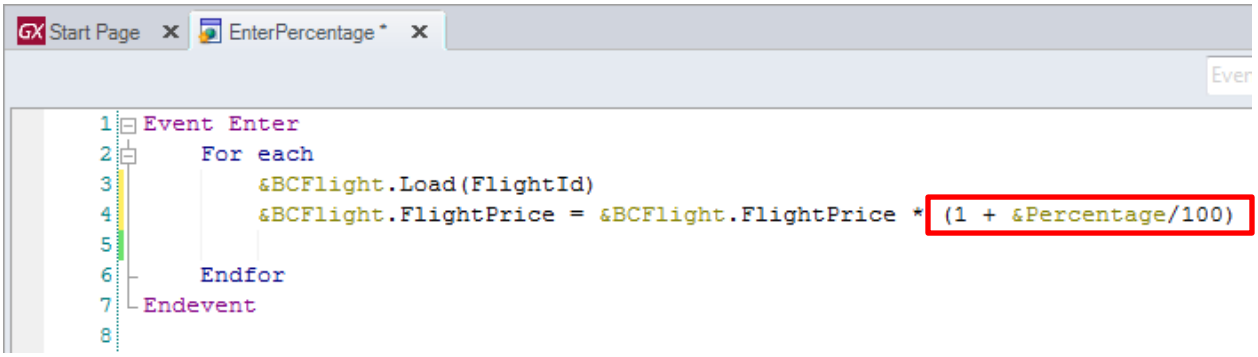
Esta instrucción, le está asignando al precio del vuelo que tenemos cargado en memoria



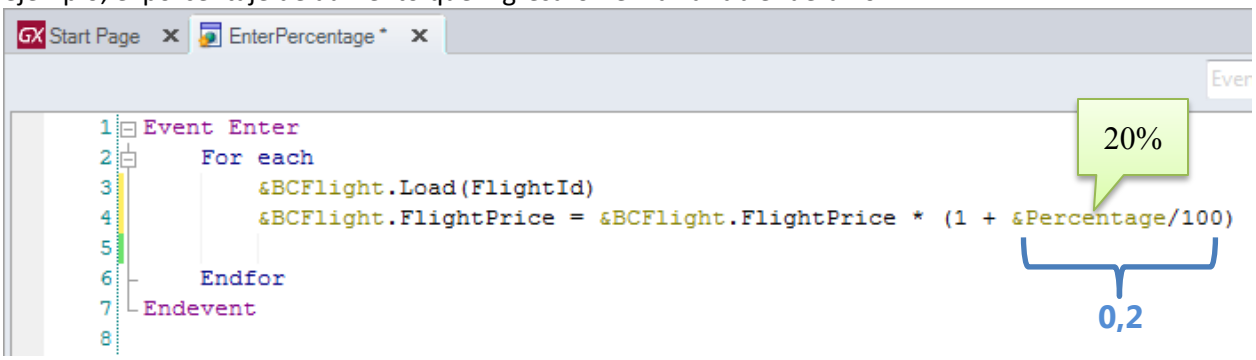
el mismo valor **que tenía**



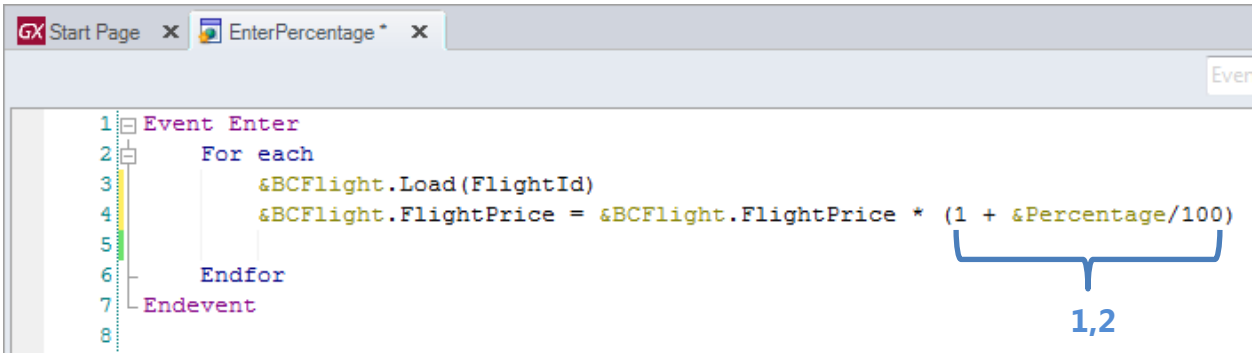
multiplicado por el resultado de la cuenta que está entre paréntesis



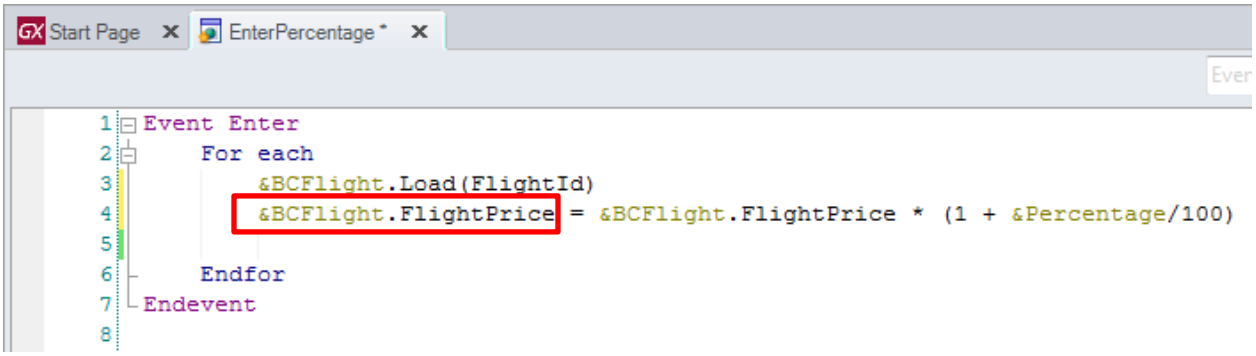
Este cálculo está sumando 1 más el valor que tiene la variable &Percentage dividido 100… o sea, que si por ejemplo, el porcentaje de aumento que ingresaron en la variable fuera 20



al multiplicar el precio actual * 1,20

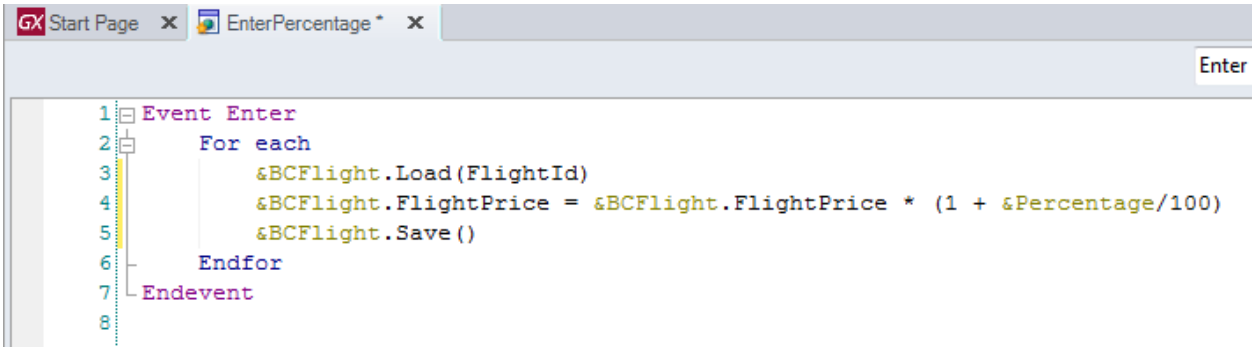


se obtiene el nuevo precio con el 20% de aumento, el cual asignamos al precio del vuelo



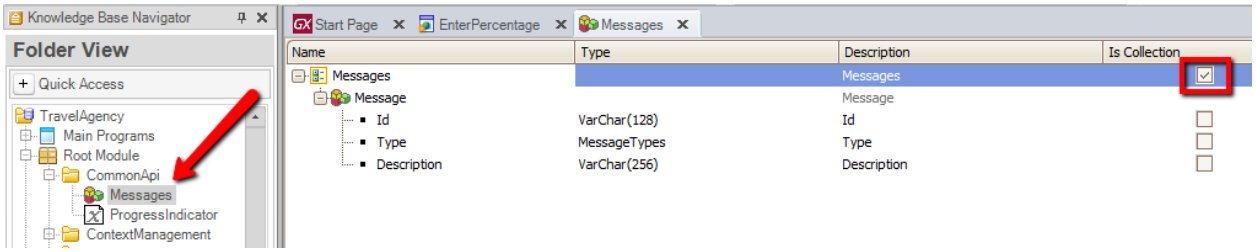
Ahora debemos grabar físicamente esta actualización. Escribamos la instrucción para hacerlo.

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



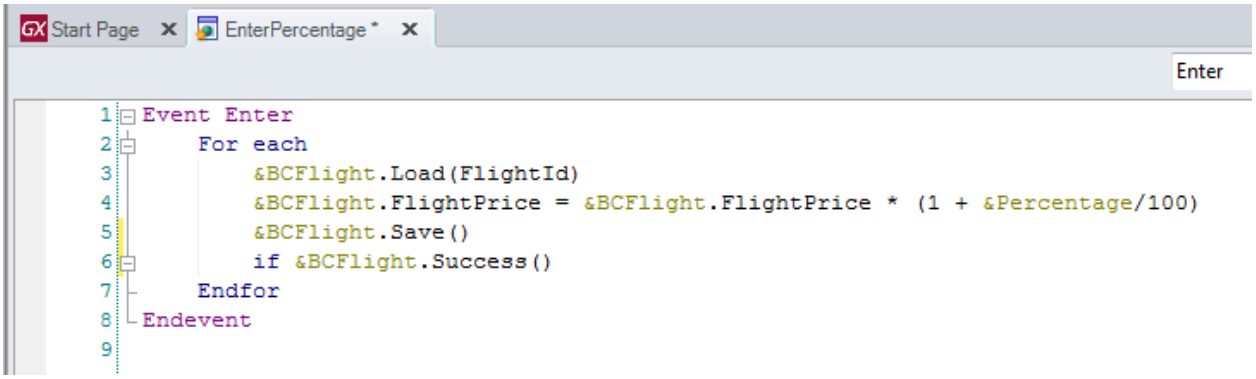
Cuando se ejecute el método save, se dispararán las reglas definidas en la transacción Flight -con algunas excepciones- y se ejecutarán las validaciones automáticas que efectúa la transacción.

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.

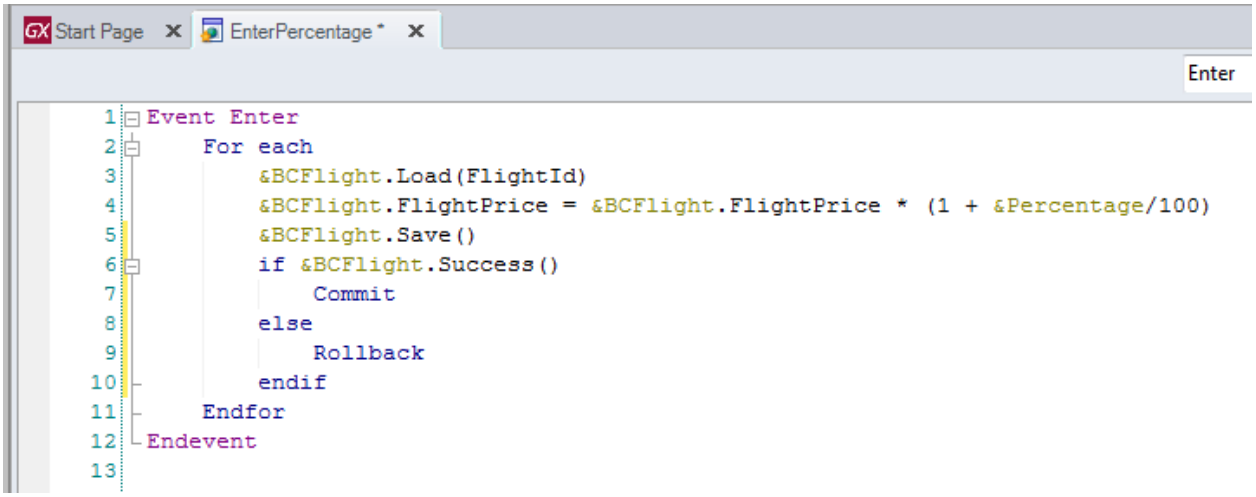


Esta colección se podrá recorrer y procesar. No lo mostraremos en este video, sino en otro que enseña más detalles sobre business components.

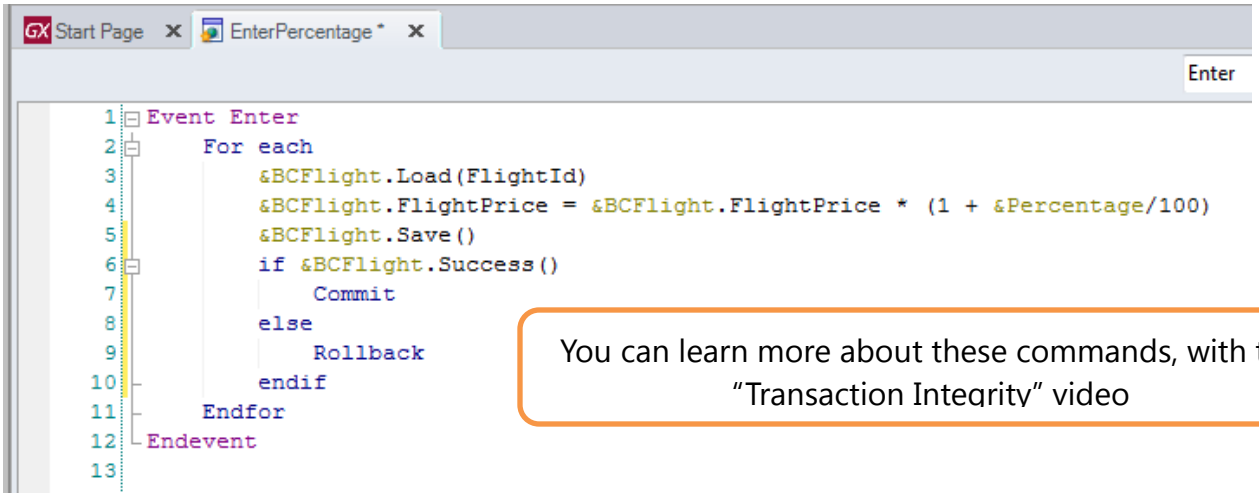
Ahora evaluaremos si la operación de grabación fue exitosa



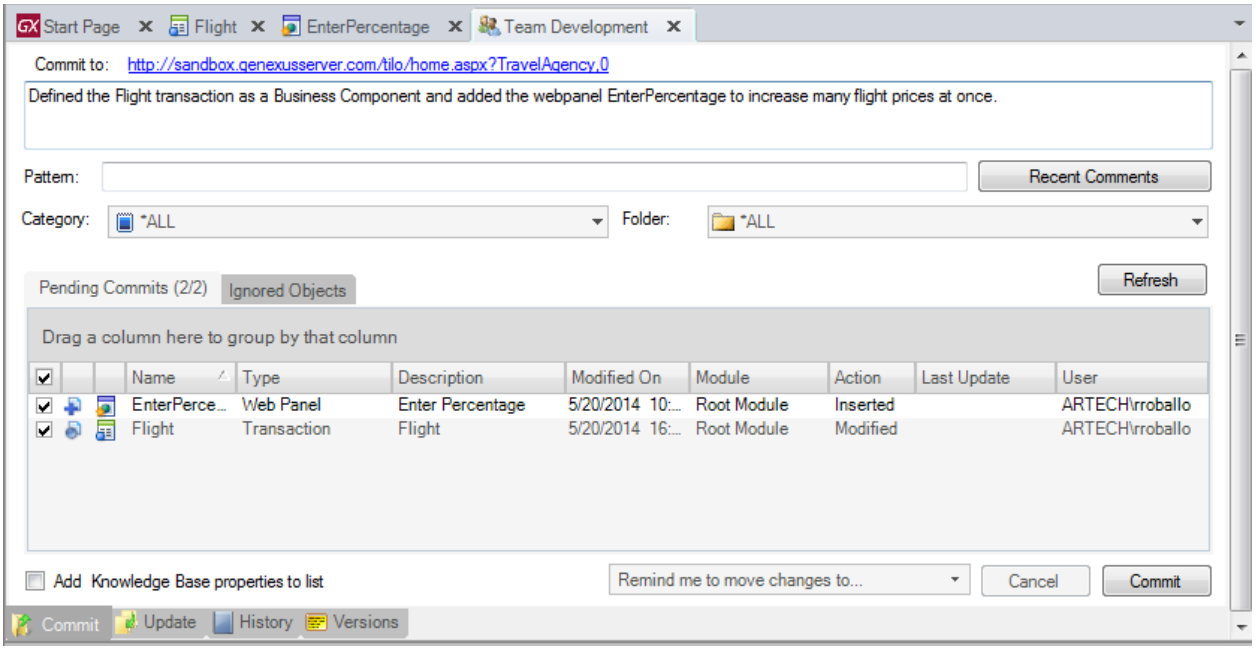
En caso de éxito, escribiremos Commit



y en caso contrario, escribiremos Rollback para deshacer las operaciones que intentamos efectuar a la base de datos posteriores al último commit.

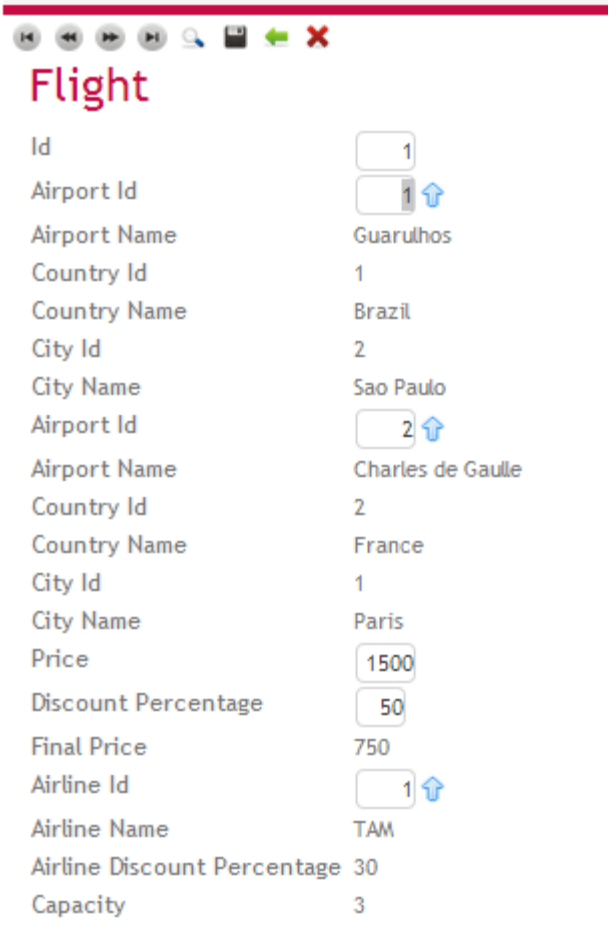


Vamos a probar ahora esta funcionalidad que hemos implementado. Antes salvemos los cambios hasta aquí, en GXserver.

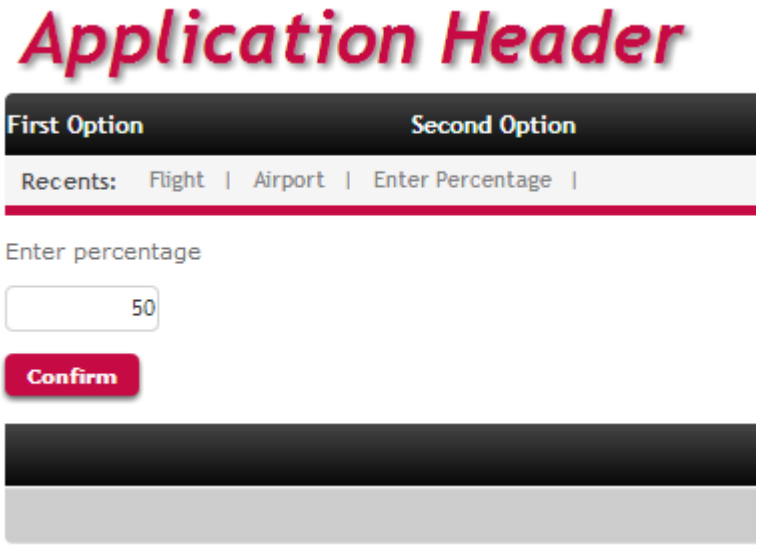


Presionamos F5…

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



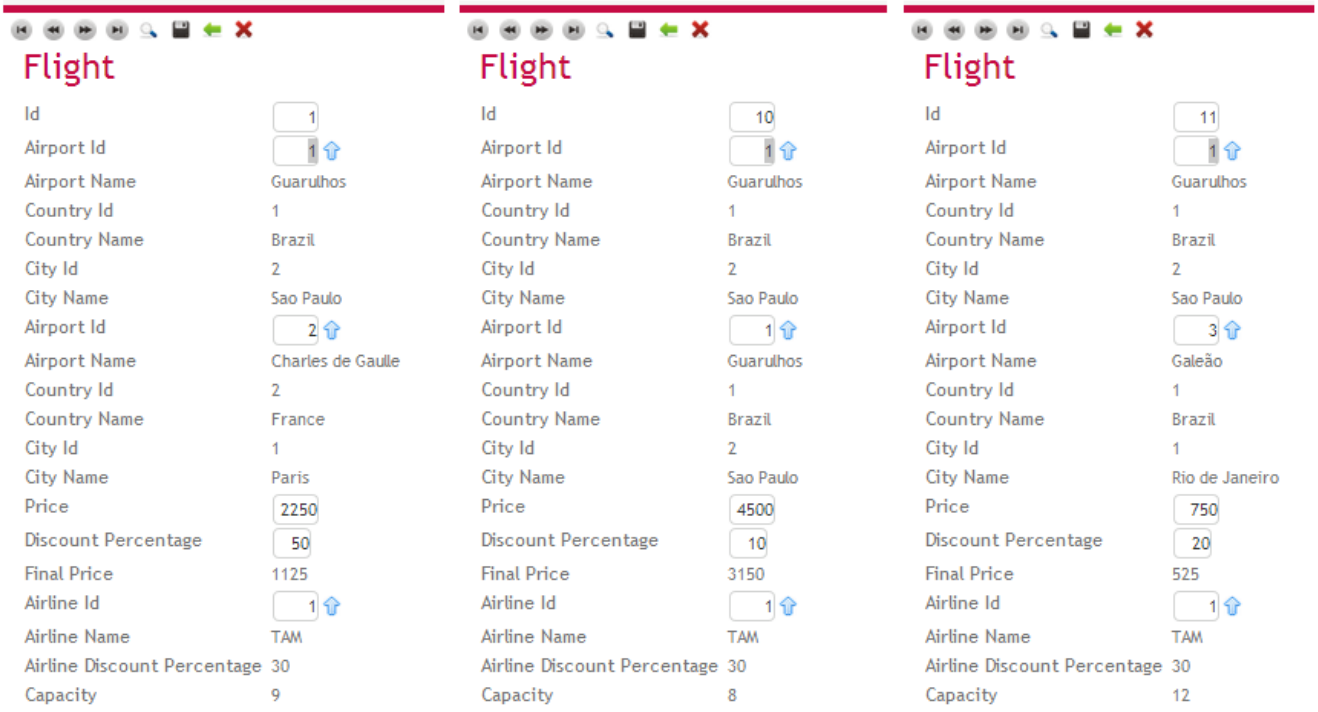
Y ahora ejecutemos el web panel “EnterPercentage”.



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

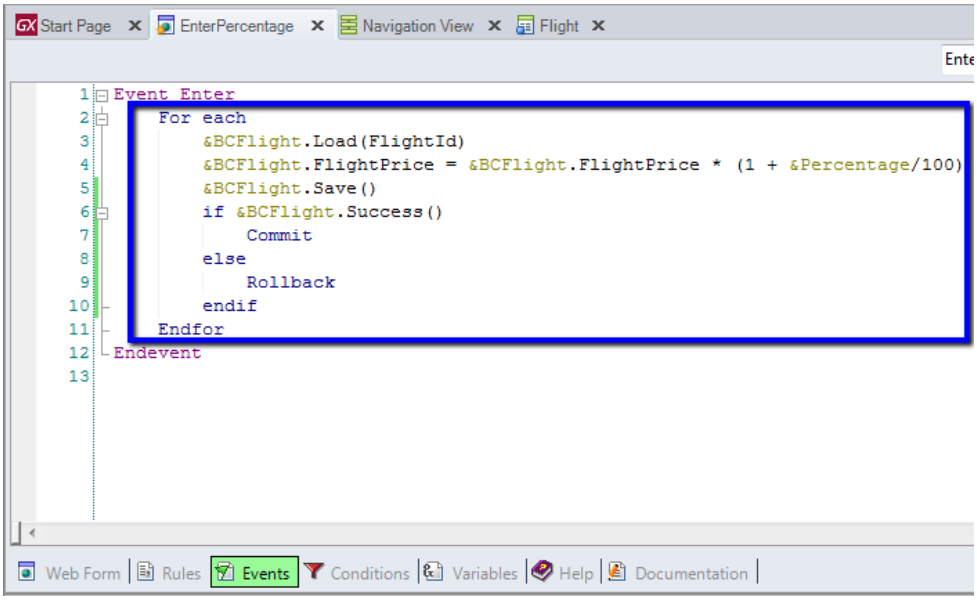
No hemos incluído ningún mensaje que avise que el proceso se realizó y terminó, pero vamos a corroborarlo, ejecutando la transacción “Flight” nuevamente.

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

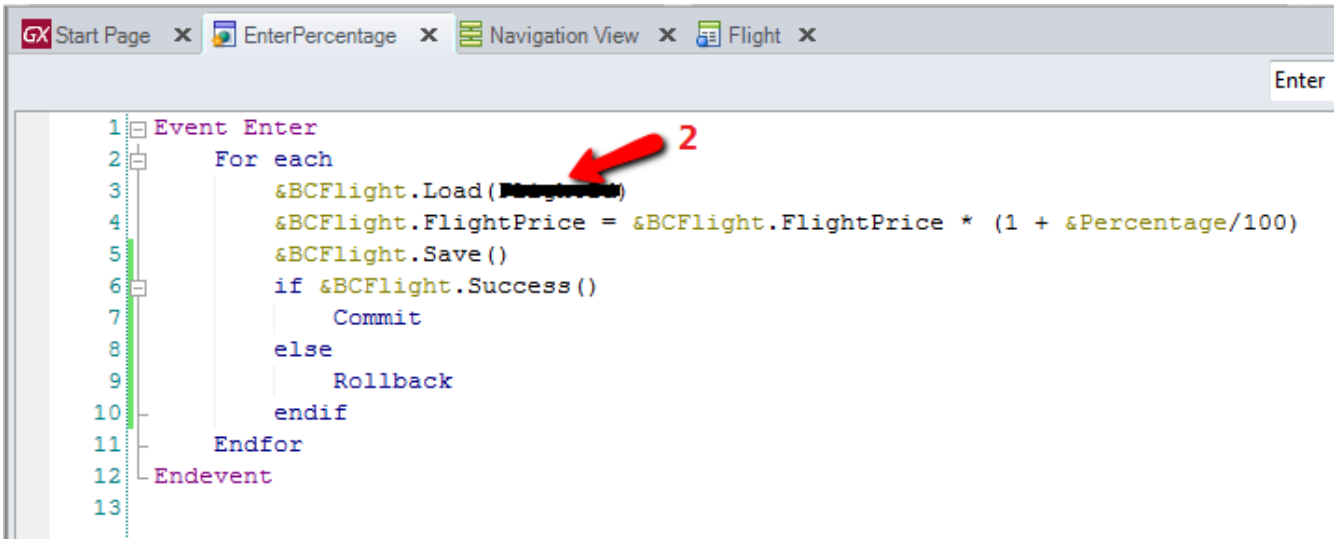


Ahora volvamos a la codificación que hicimos. Es importante aclarar que podemos usar una variable del tipo business component fuera de un comando For each.

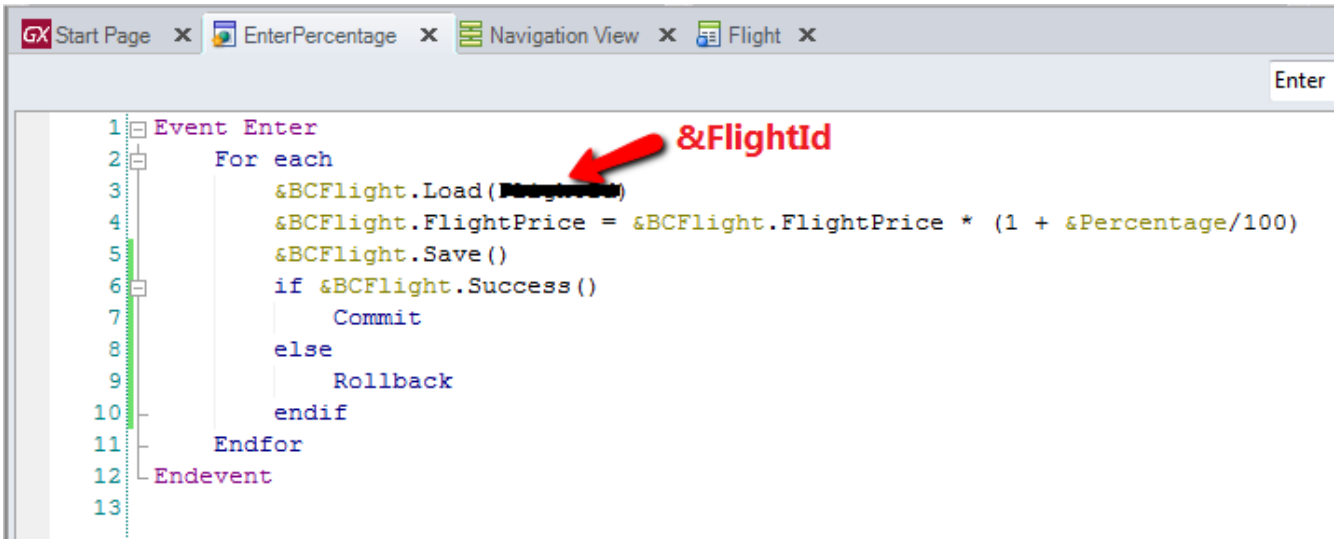
Por ejemplo, todo este código:



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.

```
Event Enter
    For each
        &BCFlight.Load(FlightId)
        &BCFlight.FlightPrice = &BCFlight.FlightPrice * (1 + &Percentage/100)
        &BCFlight.Save()
        if &BCFlight.Success()
            Commit
        else
            Rollback
        endif
    Endfor
Endevent
```

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

```
Event Enter
    For each
        &BCFlight.Load(FlightId)
        &BCFlight.FlightPrice = &BCFlight.FlightPrice * (1 + &Percentage/100)
        &BCFlight.Save()
        if &BCFlight.Success()
            Commit
        else
            Rollback
        endif
    Endfor
Endevent
```

y después ejecutar el método Delete para efectuar la eliminación

```
Event Enter
    For each
        &BCFlight.Load(FlightId)
        &BCFlight.Delete()
        if &BCFlight.Success()
            Commit
        else
            Rollback
        endif
    Endfor
Endevent
```

Con este código, estamos navegando toda la tabla de vuelos

```
Event Enter
    For each
        &BCFlight.Load(FlightId)
        &BCFlight.Delete()
        if &BCFlight.Success()
            Commit
        else
            Rollback
        endif
    Endfor
Endevent
```

Tabla Base: FLIGHT

cargamos cada vuelo navegado en memoria

```
Event Enter
    For each
        &BCFlight.Load(FlightId)
        &BCFlight.Delete()
        if &BCFlight.Success()
            Commit
        else
            Rollback
        endif
    Endfor
Endevent
```

We are deleting all the flights

y eliminamos el vuelo.

Este último bloque de código

```
Event Enter
    For each
        &BCFlight.Load(FlightId)
        &BCFlight.Delete()
        if &BCFlight.Success()
            Commit
        else
            Rollback
        endif
    Endfor
Endevent
```

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

```
Event Enter
    For each
        &BCFlight.Load(FlightId)
        &BCFlight.Delete()
        if &BCFlight.Success()
            Commit
        else
            Rollback
        endif
    Endfor
Endevent
```

y en caso contrario, no

```
Event Enter
    For each
        &BCFlight.Load(FlightId)
        &BCFlight.Delete()
        if &BCFlight.Success()
            Commit
        else
            Rollback
        endif
    Endfor
Endevent
```

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

```
Event Enter
    For each
        &BCFlight.Load(&FlightId)
        &BCFlight.Delete()
        if &BCFlight.Success()
            Commit
        else
            Rollback
        endif
    Endfor
Endevent
```

Variable or fixed value

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

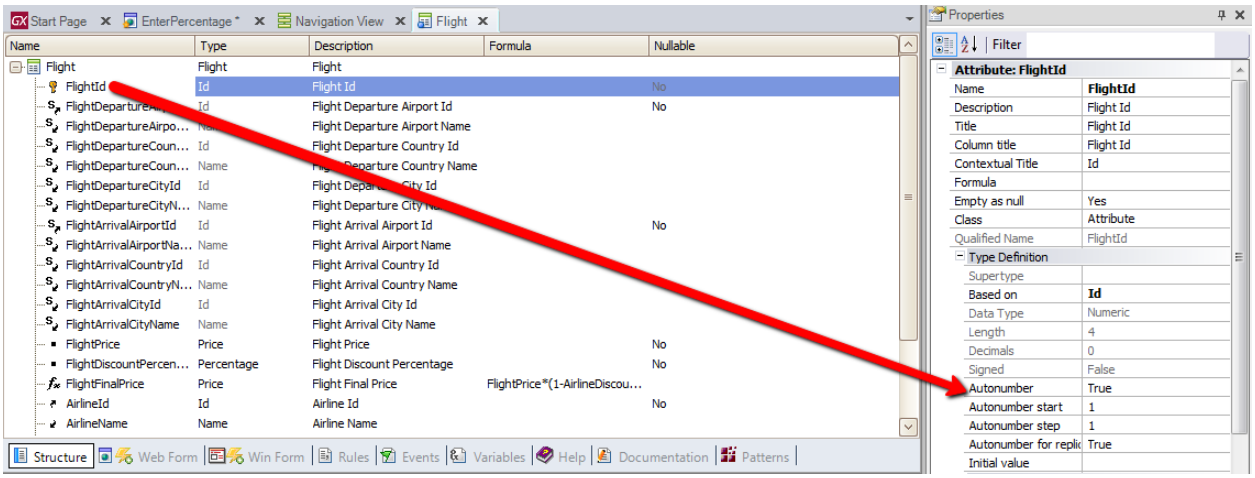
```
Event Enter
    For each
        &BCFlight.Load(FlightId)
        &BCFlight.FlightPrice = &BCFlight.FlightPrice * (1 + &Percentage/100)
        &BCFlight.Save()
        if &BCFlight.Success()
            Commit
        else
            Rollback
        endif
    Endfor
Endevent
```

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

```
Event Enter
        &BCFlight.FlightPrice = 2000
        &BCFlight.Save()
        if &BCFlight.Success()
            Commit
        else
            Rollback
        endif
Endevent
```

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 aeropuerto de origen y determinado aeropuerto de destino…. escribimos como venimos haciendo &BCFlight, punto, elegimos FlightDepartureAirportId, le asignamos valor…

```
Event Enter
        &BCFlight.FlightDepartureAirportId =
        &BCFlight.FlightPrice = 2000
        &BCFlight.Save()
        if &BCFlight.Success()
            Commit
        else
            Rollback
        endif
Endevent
```

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,

```
Event Enter
        &BCFlight.FlightDepartureAirportId =
        &BCFlight.FlightPrice = 2000
        &BCFlight.Save()
        if &BCFlight.Success()
            Commit
        else
            Rollback
        endif
Endevent
```

se usa para grabar tanto una inserción de un nuevo registro, como para grabar una actualización.
Si primero cargamos en memoria un registro,

```
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

```
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…

```
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, más Save(),

```
Event Enter
        &BCFlight.FlightDepartureAirportId = 1
        &BCFlight.FlightPrice = &BCFlight.FlightPrice * (1 + &Percentage/100)
        &BCFlight.Save()
        if &BCFlight.Success()
            Commit
        else
            Rollback
        endif
    Endfor
Endevent
```

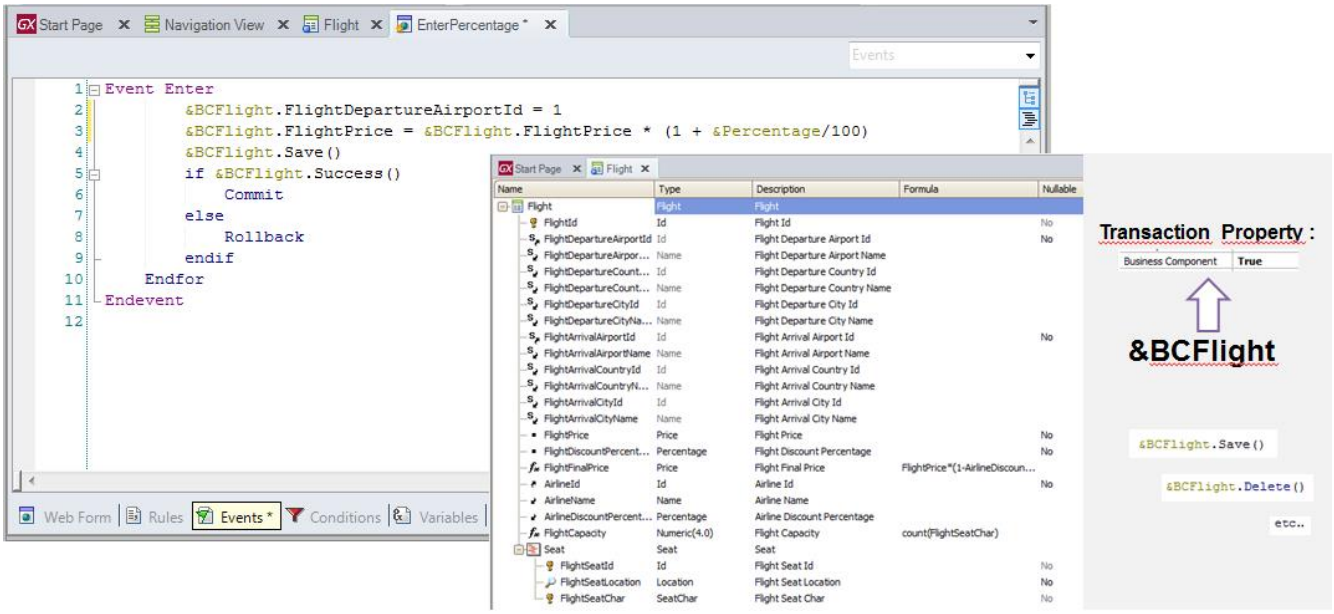
GeneXus entiende que es una inserción.

No podemos olvidar el commit.

```
Event Enter
        &BCFlight.FlightDepartureAirportId = 1
        &BCFlight.FlightPrice = &BCFlight.FlightPrice * (1 + &Percentage/100)
        &BCFlight.Save()
        if &BCFlight.Success()
            Commit
        else
            Rollback
        endif
    Endfor
Endevent
```

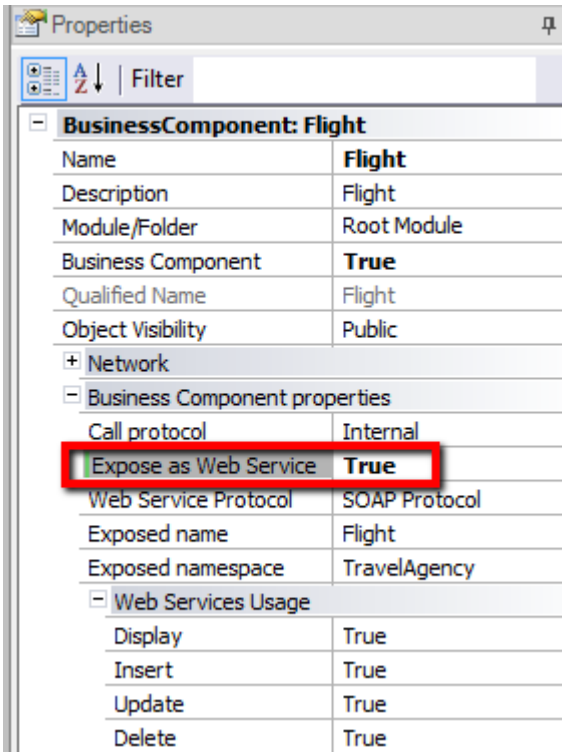
Así hemos visto otra forma de actualizar la base de datos, en particular usando business components.

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



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

También es posible exponer a los business components como servicio web, es decir que puedan utilizarse las facilidades para actualizar la base de datos, desde una aplicación web independiente que consuma este servicio.
Para lograr esto debemos configurar algunas de sus propiedades:



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.