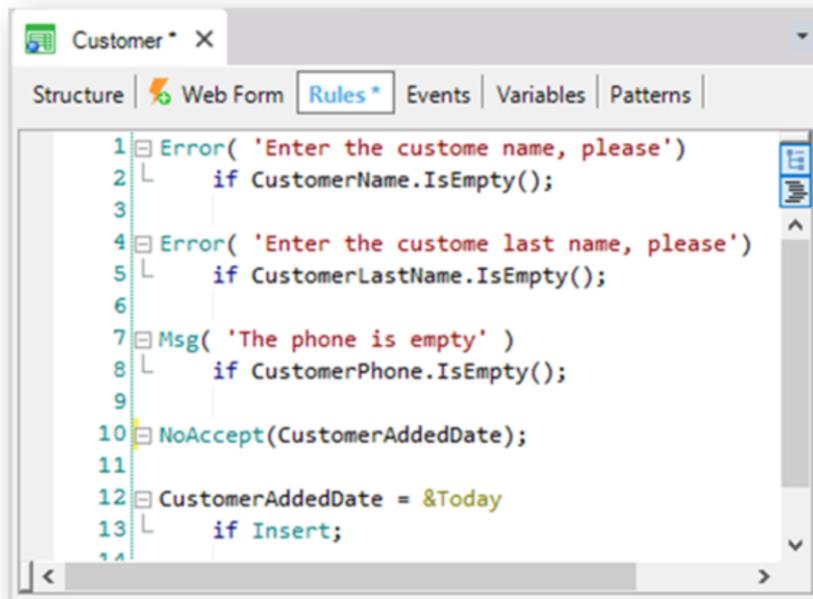


# Momentos de disparo de reglas

*GeneXus 16*

## Reglas



```
1 Error( 'Enter the custome name, please')
2   L   if CustomerName.IsEmpty();
3
4 Error( 'Enter the custome last name, please')
5   L   if CustomerLastName.IsEmpty();
6
7 Msg( 'The phone is empty' )
8   L   if CustomerPhone.IsEmpty();
9
10 NoAccept(CustomerAddedDate);
11
12 CustomerAddedDate = &Today
13   L   if Insert;
14
```

Cuando vimos las reglas que podemos escribir en las transacciones, dijimos que no era necesario especificar cuándo debe ejecutarse cada una de ellas, ya que GeneXus determina dichos momentos de disparo.

La mayoría de las veces, las reglas que definimos se ejecutan en el momento que pretendemos, sin embargo en algunos casos puede ser necesario modificar dicho momento.

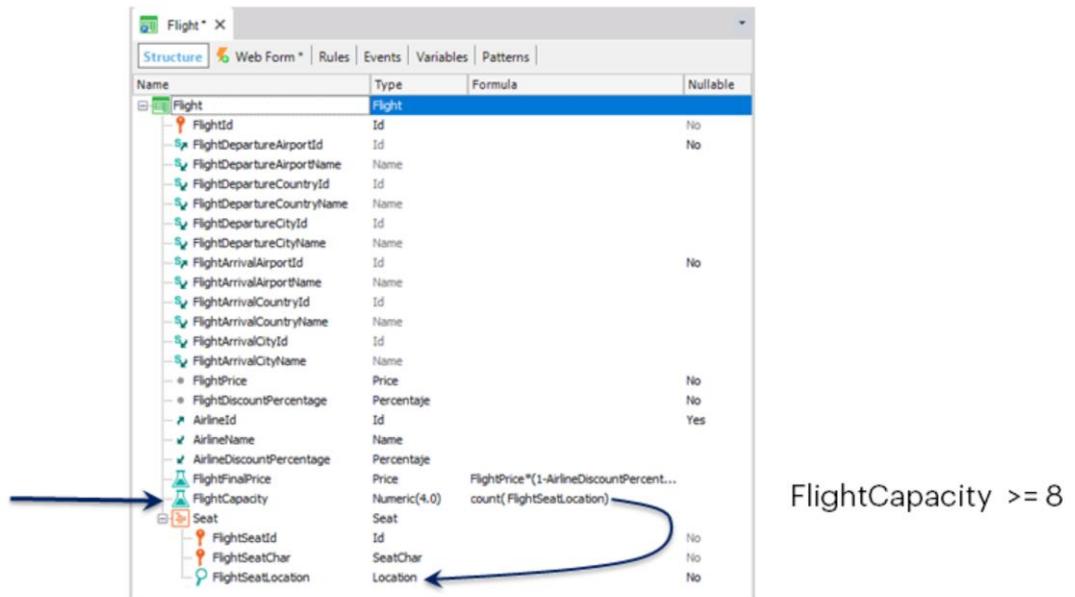
Capacity: 7

Seat

Seat Id	Seat Char	Seat Location
x	1 A	Window
x	1 B	Middle
x	1 C	Aisle
x	1 D	Window
x	1 E	Middle
x	1 F	Aisle
x	2 A	Window
	0 A	Window

Veamos un ejemplo. Supongamos que por cada vuelo, la aerolínea desea controlar que no pueda ingresarse una cantidad de asientos incorrecta, por ejemplo, que cada vuelo no pueda tener menos de ocho asientos. Recordemos que teníamos el atributo FlightCapacity, fórmula, que contaba la cantidad de asientos.

## Ejemplo: Cantidad de asientos de un vuelo



Name	Type	Formula	Nullable
Flight	Flight		
FlightId	Id		No
FlightDepartureAirportId	Id		No
FlightDepartureAirportName	Name		
FlightDepartureCountryId	Id		
FlightDepartureCountryName	Name		
FlightDepartureCityId	Id		
FlightDepartureCityName	Name		
FlightArrivalAirportId	Id		No
FlightArrivalAirportName	Name		
FlightArrivalCountryId	Id		
FlightArrivalCountryName	Name		
FlightArrivalCityId	Id		
FlightArrivalCityName	Name		
FlightPrice	Price		No
FlightDiscountPercentage	Porcentaje		No
AirlineId	Id		Yes
AirlineName	Name		
AirlineDiscountPercentage	Porcentaje		
FlightFinalPrice	Price	FlightPrice*(1-AirlineDiscountPercent...	
FlightCapacity	Numeric(4,0)	count(FlightSeatLocation)	
Seat	Seat		
FlightSeatId	Id		No
FlightSeatChar	SeatChar		No
FlightSeatLocation	Location		No

FlightCapacity >= 8

Al ingresar un vuelo nuevo, queremos que se realice el control correspondiente y que no sea posible salvar el vuelo si no cumple la condición deseada.

Para lograr esto, en la transacción que registra los vuelos, vamos a declarar una regla a partir del atributo FlightCapacity que cuenta la totalidad de asientos del avión.

## Ejemplo: Cantidad de asientos de un vuelo

The image shows a screenshot of the GeneXus IDE interface. The 'Rules' tab is active, displaying a rule definition: `1 Error( "The seat quantity mustn't be less than eight") if FlightCapacity < 8;`. Below the IDE, a web browser window shows the 'Flight' application. The 'Id' field contains the value '0', and a yellow error message is displayed: 'The seat quantity mustn't be less than eight'. A large blue 'X' is overlaid on the error message. A callout box with the text '¡El error se dispara antes de ingresar los asientos!' (The error is triggered before entering the seats!) points to the error message. An arrow labeled 'F5' points from the IDE to the browser window.

Así que vamos a la sección Rules y declaramos una regla Error que no nos permita almacenar un vuelo si tiene menos de 8 asientos. Escribimos...  
Error... la cantidad de asientos no puede ser menor a ocho if FlightCapacity es menor que ocho...  
Cerramos con punto y coma...

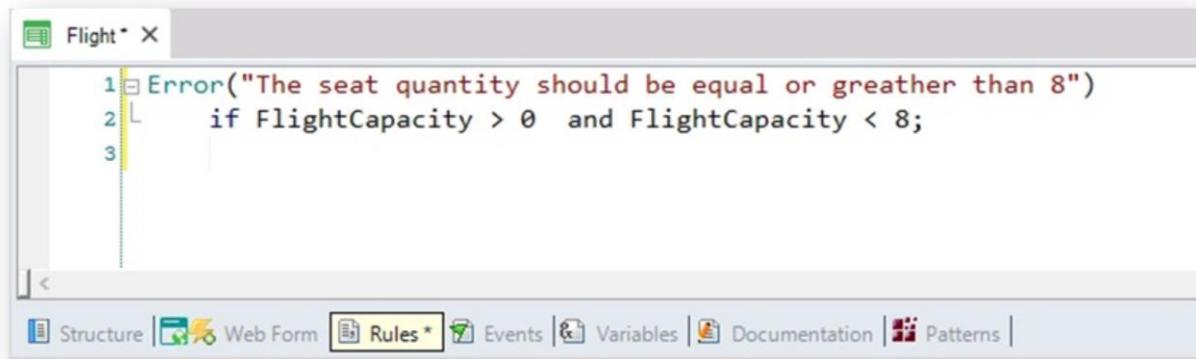
Presionamos F5...

Vamos a abrir la transacción Flight para crear un vuelo nuevo.

Vemos que ya se está disparando el error.

¿Por qué? Porque la fórmula se dispara apenas puede, y va cambiando conforme se van agregando líneas. El problema es que al principio no hemos tenido tiempo de ingresar ninguna línea, por lo que la fórmula FlightCapacity se disparará dando por resultado cero, que es menor que ocho.

## Ejemplo: Cantidad de asientos de un vuelo



```
1 Error("The seat quantity should be equal or greather than 8")
2   if FlightCapacity > 0 and FlightCapacity < 8;
3
```

The screenshot shows a code editor window titled "Flight \* X". The code contains a rule definition with three lines: a call to the Error function, an if statement, and a blank line. The if statement condition is "FlightCapacity > 0 and FlightCapacity < 8;". Below the code editor is a toolbar with icons for Structure, Web Form, Rules (highlighted), Events, Variables, Documentation, and Patterns.

Se nos podría ocurrir, entonces, condicionar el error:

## Ejemplo: Cantidad de asientos de un vuelo

Airline Name	TAM
Airline Discount Percentage	30
Final Price	2100.00
Capacity	0

### Seat

Seat Id	Seat Char	Seat Location
1	A	Window
	A	Window
0	A	Window
0	A	Window
0	A	Window
[New row]		

The seat quantity mustn't be less than eight

CONFIRM CANCEL

¡El error se dispara antes de lo esperado porque la cantidad de asientos es menor que 8!

Para dar tiempo a ingresar alguna línea.

Presionamos F5.

Dejamos el valor del identificador vacío ya que es autonumerado...

Ingresamos un vuelo desde el aeropuerto de Guarulhos, en Sao Paulo, Brasil..., hasta el aeropuerto Charles de Gaulle en París, Francia. El precio del vuelo es de 3000, el descuento de un 10% y la aerolínea es TAM.

Ahora ingresamos un asiento 1, letra A, ventana...30 y al salir del renglón, vemos que se despliega el mensaje de error:

Obviamente no queremos que el mensaje de error se dispare aquí, porque todavía no pudimos ingresar todos los asientos. Es claro que si ingresamos un único asiento, tenemos menos de 8 asientos ingresados y que corresponde que la regla Error se dispare, pero en realidad necesitamos que el control de la cantidad de asientos se realice **después** de que el usuario termine de ingresar todos los asientos.

## Ejemplo: Cantidad de asientos de un vuelo

The screenshot shows the GeneXus IDE interface. The top window displays the 'Structure' tab for the 'Flight' entity, listing attributes: FlightId (Id, Nullable), FlightDepartureAirportId (Id, Nullable), and FlightCapacity (Numeric(4,0), Nullable). The bottom window shows the 'Rules' tab for the same entity. A rule is defined with the following code:

```

1 Error( "The seat quantity mustn't be less than eight")
2   if FlightCapacity < 0 and FlightCapacity < 8
3     on AfterLevel
4       Level FlightSeatChar;
5

```

A blue box highlights the 'on AfterLevel' and 'Level FlightSeatChar;' lines. A blue arrow points from this box to the 'FlightSeatChar' attribute in the 'Seat' entity structure shown in the bottom window. The 'Seat' entity structure lists attributes: FlightSeatId (Id, Nullable), FlightSeatChar (SeatChar, Nullable), and FlightSeatLocation (Location, Nullable).

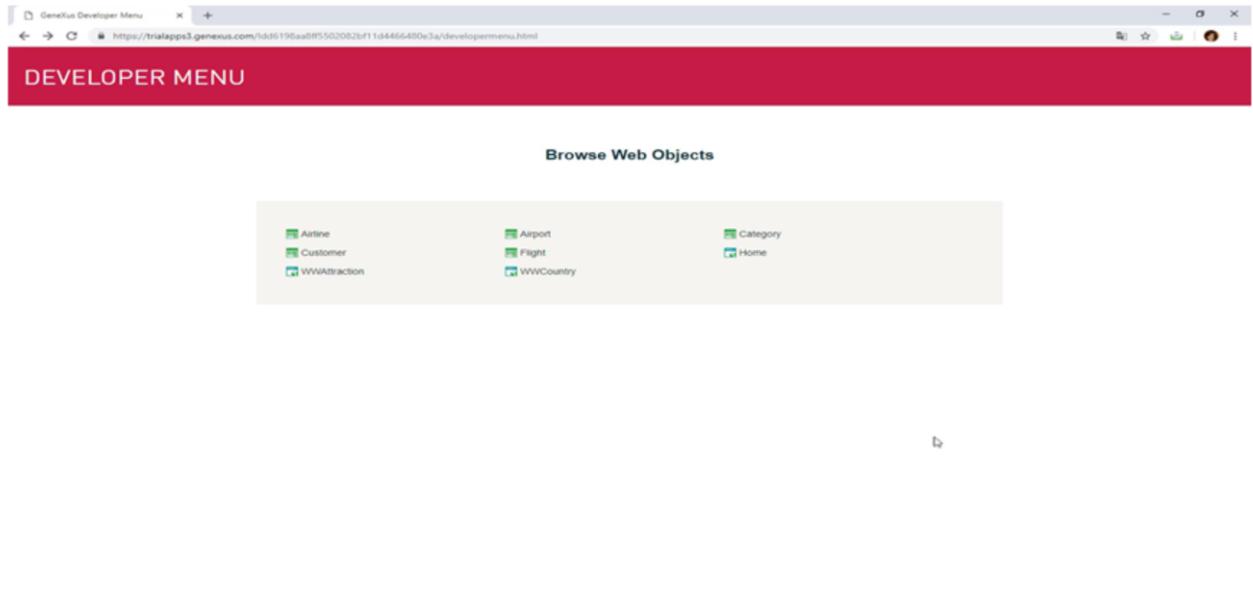
Para lograr esto, debo condicionar la regla a dispararse después de que termine de trabajar con las líneas del grid, para eso escribimos... On afterlevel level FlightSeatChar.

El momento "on after level" hace que la regla se dispare después de terminar un nivel. Como en nuestro caso sería después de terminar el nivel de las líneas del grid de asientos, entonces agregamos "level FlightSeatChar" ya que este atributo está en el nivel de las líneas de asientos. Podríamos haber utilizado cualquiera de los otros atributos del nivel, por ejemplo FlightSeatLocation.

De esta forma indicamos a GeneXus que esa regla debe dispararse **después** de que se terminen de ingresar los datos donde está el atributo FlightSeatChar, es decir, después de ingresar los datos del cabezal de todos los asientos del vuelo.

La evaluación que hace la regla Error tiene sentido, ya que al momento de dispararse ya estarán ingresados todos los asientos que el usuario deseaba ingresar y se podrá verificar que al menos se hayan ingresado 8 asientos.

## DEMO



[ DEMO: <https://youtu.be/PblNZf47qXM> ]

Presionamos F5...

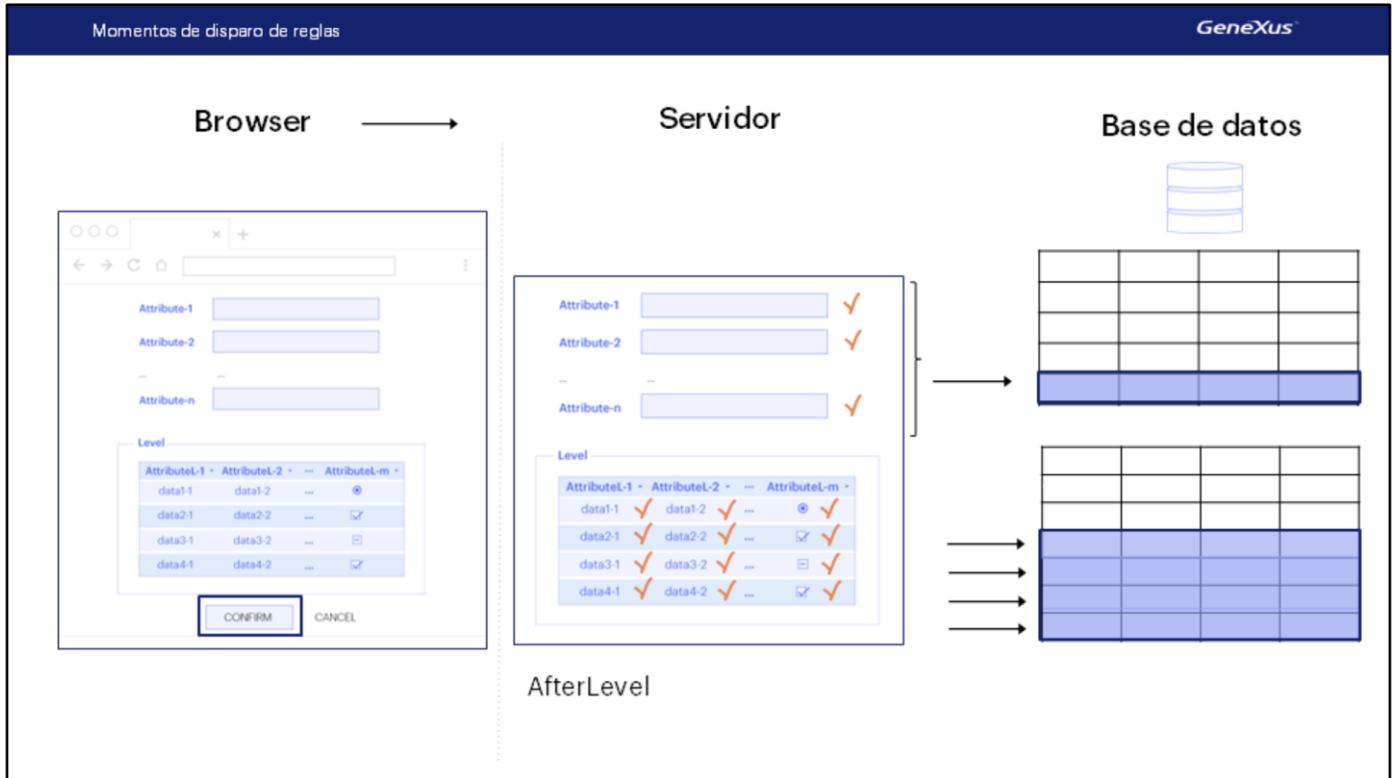
Abrimos la transacción Flight y vamos a ingresar un nuevo vuelo otra vez.

Repetimos los datos que usamos antes... El vuelo desde el aeropuerto de Guarulhos, hasta el aeropuerto Charles de Gaulle. El precio de 3000, con un 10% y la aerolínea TAM.

Ahora ingresamos los asientos...

1, A, ventana  
1, B, pasillo  
2, A, ventana  
2, B, pasillo

Y presionamos Confirmar, para indicar que terminamos de ingresar los datos del vuelo (incluyendo los asientos) y que el vuelo puede ser grabado en la base de datos.



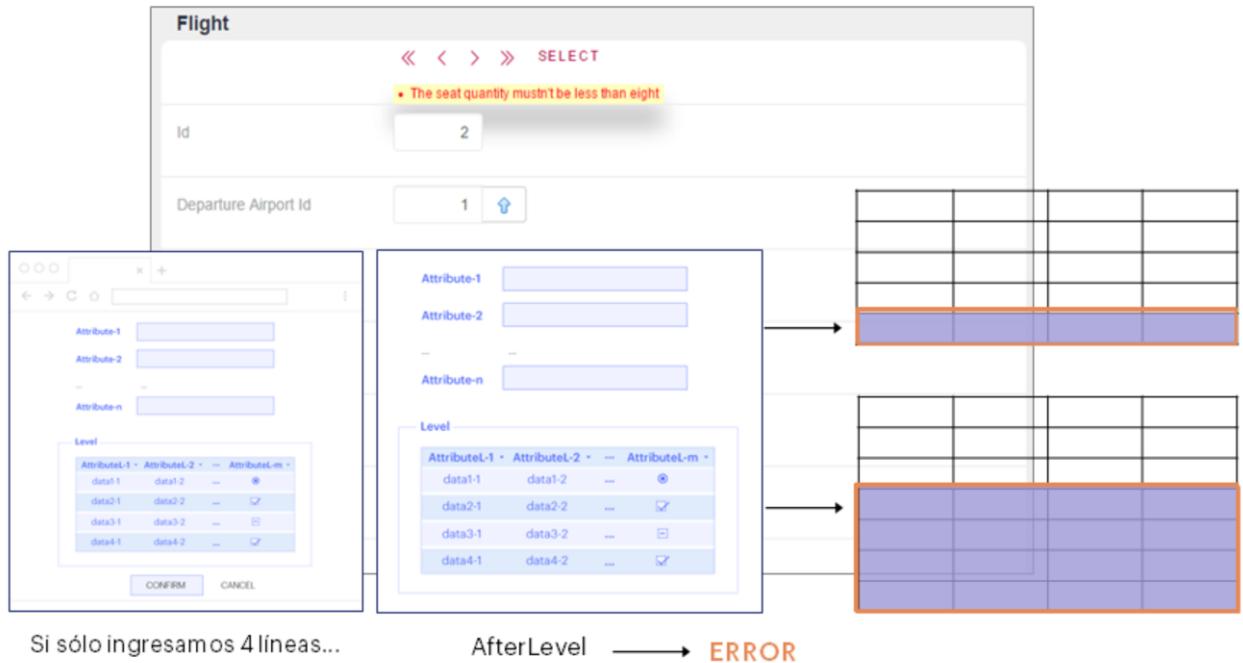
¿Qué es lo que ocurre a partir de allí? Se envían los datos de cabecal y líneas al servidor, y se van procesando uno por uno, disparando las reglas correspondientes.

Cuando termina la validación de los datos del cabezal, se inserta el registro en la tabla.

Y luego se va haciendo lo mismo para cada línea.

Cuando terminan de procesarse todas las líneas, ese es el momento **AfterLevel**. Ahí se dispararán todas las reglas que se hayan condicionado a ese momento.

Observemos que ya habrán quedado grabados los datos del cabezal y líneas en la base de datos.



Volviendo a nuestra transacción en ejecución, vemos que al Confirmar, GeneXus nos indica el error, tal como esperábamos pues ingresamos sólo cuatro asientos y **no** va a dejar grabado este vuelo en la base de datos. Es que una regla Error deshace toda grabación que se hubiera efectuado.

## Ejemplo: Cantidad de asientos de un vuelo

Capacity **8**

Seat

Seat Id	Seat Char	Seat Location
1	A	Window
1	B	Aisle
2	A	Window
2	B	Aisle
3	A	Window
3	B	Aisle
4	A	Window
4	B	Aisle

[New row]

**CONFIRM** CANCEL

**8 asientos  
grabados**

Completemos los 8 asientos requeridos. Digitamos...

3, A, ventana

3, B, pasillo

4, A, ventana, A...y por último...

4, B, pasillo

Ahora presionamos Confirmar y vemos que GeneXus nos dejó salvar el vuelo sin problemas.

## Ejemplo: Cantidad de asientos de un vuelo

¡Vuelo sin asientos!

```

1 Error("The seat quantity mustn't be less than eight")
2 if FlightCapacity > 0 and FlightCapacity < 8
3 on AfterLevel
4 Level FlightSeatChar;
5

```

```

1 Error("The seat quantity mustn't be less than eight")
2 if FlightCapacity < 8
3 on AfterLevel
4 Level FlightSeatChar;
5

```

The seat quantity mustn't be less than eight

Resumiendo: conseguimos nuestro propósito retrasando el momento que GeneXus había elegido inicialmente para disparar la regla Error.

El vuelo 1 nos había quedado con 7 asientos, porque la regla de error la agregamos después. Mientras no intentemos grabar este vuelo, la regla de error no se controlará, porque, como vimos, se ejecutará después de CONFIRMAR, cuando todos los datos viajen al servidor. Presionemos Confirm, y aquí vemos el mensaje:

Entonces, agreguémosle un asiento a este vuelo:  
2-B-Pasillo

y salvamos. Ahora sí.

Como último paso, ingresemos un nuevo vuelo sin asientos. ¡Me permitió grabar!

¿Por qué? Es que tenemos esta condición:

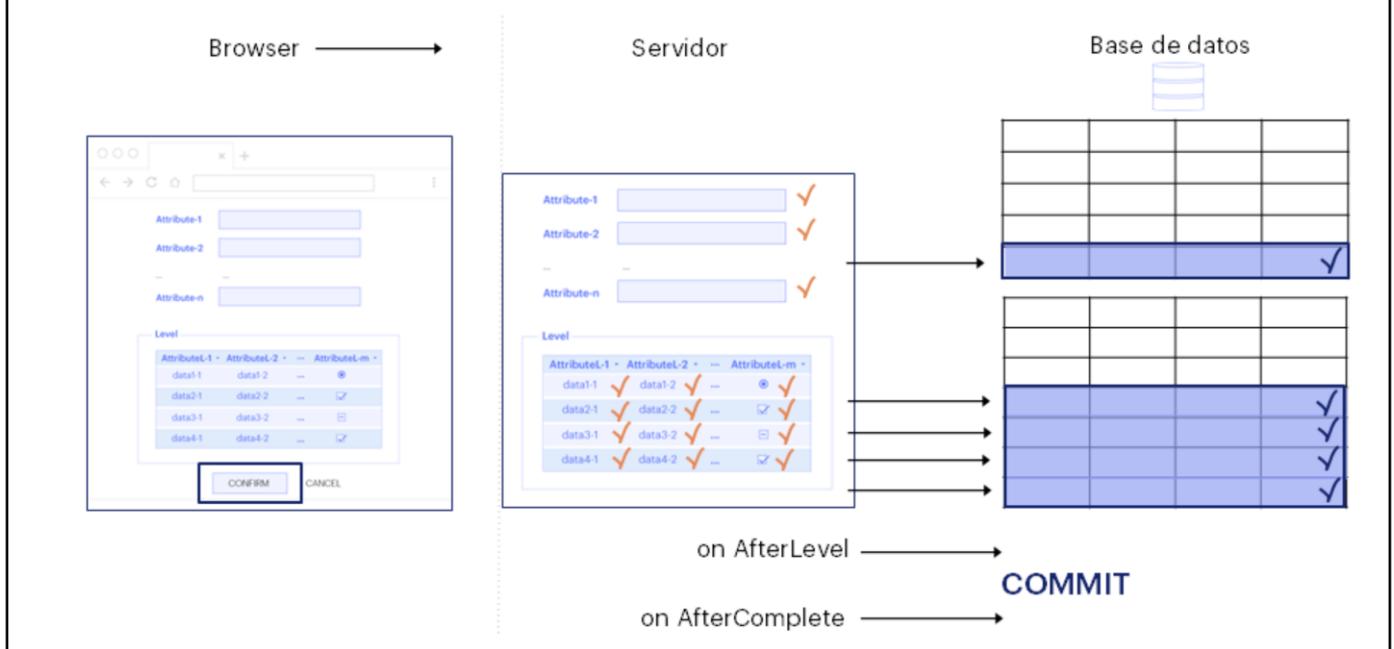
que ingresamos equivocadamente, antes de conocer que existía el AfterLevel. Por lo tanto la eliminamos, y la regla quedará escrita de la siguiente manera:

Damos F5, editemos el vuelo sin asientos y veamos que si ahora Confirmamos nuevamente, sí controla que no podamos ingresarlo.

Eliminémoslo, presionando Delete.

Volvamos a GeneXus para grabar los cambios que hicimos a nuestra KB, en GeneXus Server.

## AfterLevely AfterComplete



Con este ejemplo vimos que hay casos en los que el momento elegido por GeneXus para disparar una regla no se adecua a nuestros intereses, de modo que debemos indicarle a GeneXus en qué momento queremos que dicha regla se ejecute.

En este caso estudiamos el momento “**on Afterlevel**”, para indicar que queremos que la regla se dispare después de recorrer un nivel. Nos puede servir, por ejemplo, para llamar a un listado que imprima datos del vuelo, ya que vimos que en el AfterLevel los datos ya estarán grabados en la base de datos, aunque si se dispara una regla de error, se borrarán.

En el caso del listado, sería mejor invocarlo luego de eso, cuando estemos seguros de que los datos no se borrarán. Eso será luego de que se realice un Commit, comando que luego mencionaremos pero cuyo efecto es dar por buenos los datos insertados.

El momento que sigue al commit es **AfterComplete** y allí invocaríamos al listado.

### AfterInsert

Servidor

Base de datos

Attribute-1	<input type="text"/>	✓
Attribute-2	<input type="text"/>	✓
...	...	
Attribute-n	<input type="text"/>	✓

Level			
AttributeL-1	AttributeL-2	...	AttributeL-m
data1-1	data1-2	...	⊕ ✓
data2-1	data2-2	...	☑ ✓
data3-1	data3-2	...	☐ ✓
data4-1	data4-2	...	☑ ✓


→ on AfterInsert


→ on AfterInsert

→ on AfterInsert

→ on AfterInsert

→ on AfterInsert

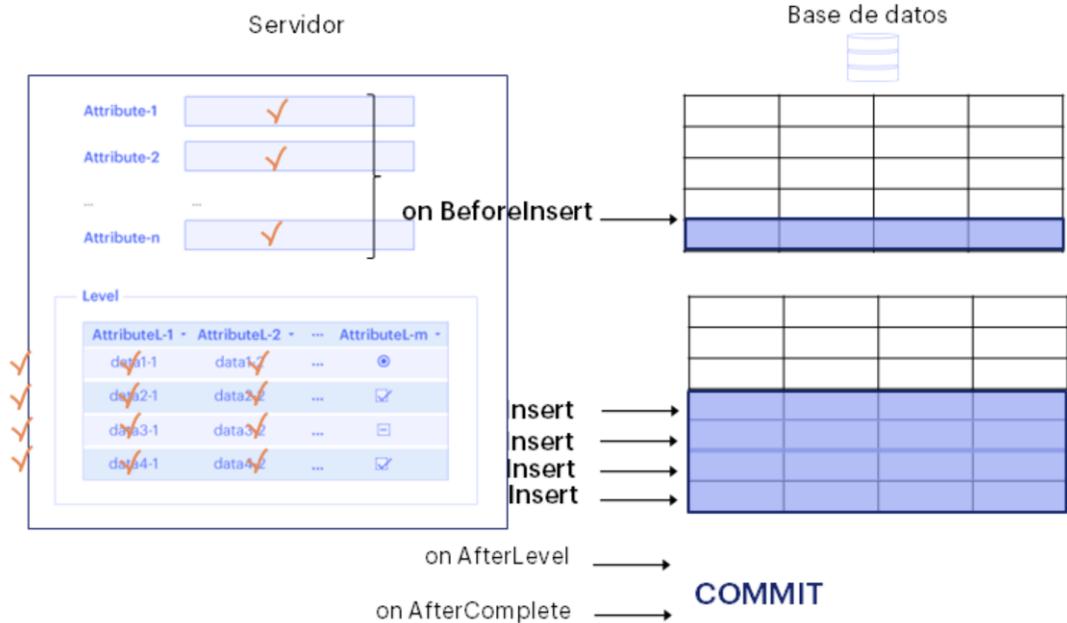
on AfterLevel →

on AfterComplete →

**COMMIT**

Tenemos otros momentos como "on AfterInsert" para indicar que la regla se dispare inmediatamente después de la grabación de cada cabezal o línea.

## BeforeInsert



u “on BeforeInsert” si quisiéramos hacer o evaluar algo, **inmediatamente antes** de que los datos del cabezal o de cada línea sean guardados en la base de datos.

Notemos que todos estos momentos de disparo empiezan con el prefijo **on** y siempre se escriben al finalizar la declaración de la regla.

Aquí sólo presentamos los más importantes, pero existen más momentos de disparo disponibles, que lo invitamos a descubrir. No profundizaremos en ellos en este nivel.

## Resumiendo

- A veces, el momento seleccionado por GeneXus para activar una regla no es el correcto. En estos casos, debemos indicar específicamente cuándo queremos que se active la regla.
  - On BeforeInsert
  - On AfterInsert
  - On AfterLevel
  - On AfterComplete
- Las reglas están condicionando estos eventos usando el prefijo "on" escrito al final de la regla.

# *GeneXus*<sup>™</sup>

**The power of doing.**

Videos	<a href="http://training.genexus.com">training.genexus.com</a>
Documentation	<a href="http://wiki.genexus.com">wiki.genexus.com</a>
Certifications	<a href="http://training.genexus.com/certifications">training.genexus.com/certifications</a>