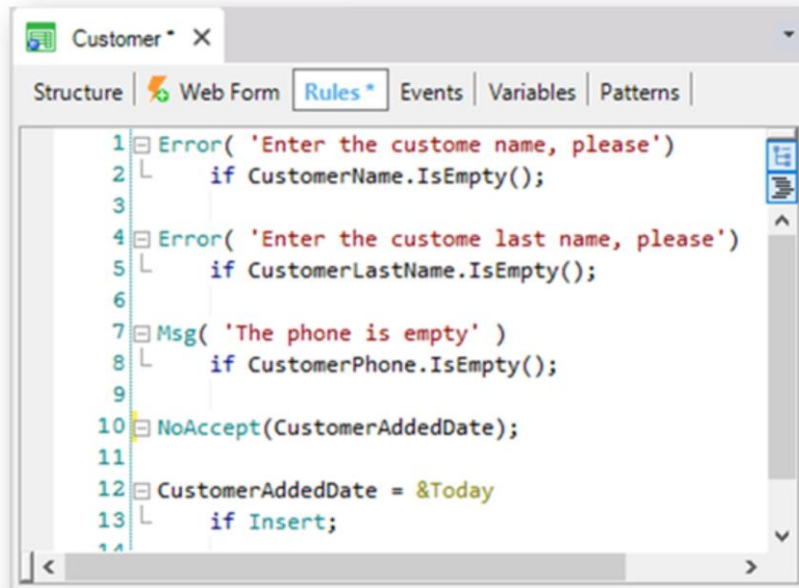


# Rule Triggering Events in Transactions

*GeneXus™ 16*

## Rules



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

When we saw the rules we may write in transactions, we said that we don't need to specify when each of them is to be executed, because GeneXus determines the triggering moments for that.

Most often, the rules we define are executed at the moment intended, but in some cases, we might need to modify those moments.

Capacity	7	
<b>Seat</b>		
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
	0 A	Window
	0 A	Window
	0 A	Window
	0 A	Window

Let's consider an example. Suppose that for each flight the airline will control that an incorrect number of seats is not entered, such as having each flight with no less than eight seats. Remember that we had the FlightCapacity formula attribute to count the number of seats.

## Example: Number of Seats in a Flight

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	Percentage		No
AirlineId	Id		Yes
AirlineName	Name		
AirlineDiscountPercentage	Percentage		
FlightFinalPrice	Price	FlightPrice*(1-AirlineDiscountPercent...	
FlightCapacity	Numeric(4,0)	count(FlightSeatLocation)	
Seat	Seat		
FlightSeatId	Id		No
FlightSeatChar	SeatChar		No
FlightSeatLocation	Location		No

FlightCapacity &gt;= 8

When we enter a new flight we want the corresponding control to take place with no possibility for saving that flight if the condition established is not fulfilled.

To this purpose, we will declare a rule in the transaction that records the flights, based on the FlightCapacity attribute that counts all seats on the flight.

## Example: Number of Seats in a Flight

The image shows a screenshot of the GeneXus IDE and a web browser. In the IDE, the 'Rules' tab is active, showing a rule definition: `1 Error( "The seat quantity mustn't be less than eight") if FlightCapacity < 8;`. A blue arrow labeled 'F5' points from the IDE to a web browser window. The browser window displays a web form titled 'Application Name by GeneXus' with a 'Flight' section. The 'id' field contains the value '0' and is highlighted with a yellow error message: 'The seat quantity mustn't be less than eight'. A large blue 'X' is overlaid on the error message. A yellow callout box with the text 'We enter the seats! The error is triggered before' points to the error message.

Therefore, we go to the Rules section and declare an Error rule that will prevent us from saving a flight with less than eight seats. We will write...

Error... the number of seats cannot be less than eight if FlightCapacity is less than eight...

We then close with semi-colon...

Then we press F5...

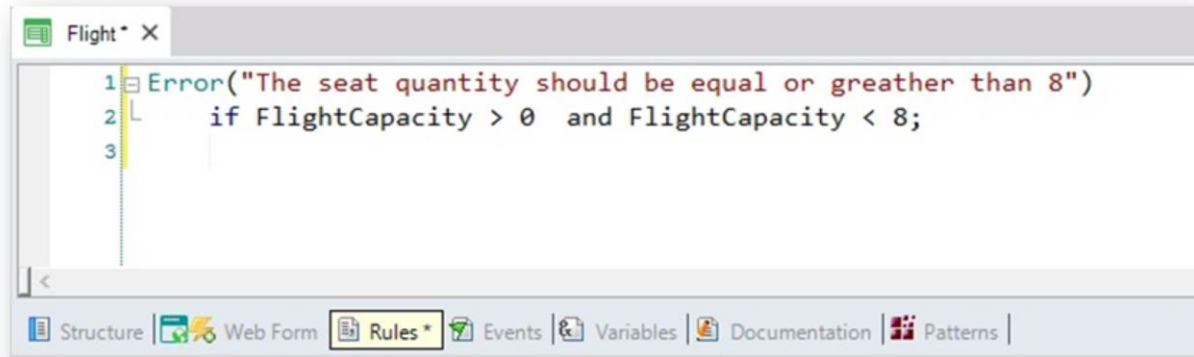
Now we will open the Flight transaction to create a new flight.

We can see that the error is now being triggered.

Why? Because the formula is triggered as soon as possible and it changes as new lines are added. The problem is that, at the beginning, we did not have time to enter any lines, so the FlightCapacity formula will be triggered with a result of zero, which is less than eight.

Example: Number of Seats in a Flight

We could modify the conditioning of the error:



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

We could then think of conditioning the error.

## Example: Number of Seats in a Flight

Metric Name	Value
Airline Discount Percentage	30
Final Price	2100.00
Capacity	0

Seat				
Seat Id	Seat Char	Seat	Location	
x 1	A	Window		The seat quantity mustn't be less than eight
	A	Window		
	A	Window		
	A	Window		
	A	Window		
	A	Window		
[New row]				

CONFIRM CANCEL

The error is triggered before we expect it because the number of seats is less than 8!

So we can allow some time to enter lines.

We press F5.

We open the Flight transaction, and we leave the identifier value empty because it is autonumbered...

We enter a flight from the Guarulhos airport in Sao Paulo, Brazil, to the Charles de Gaulle airport in Paris, France. The flight price is 3000, with a 10% discount, and the airline is TAM.

We now enter seat 1, letter A, window...30 and upon exiting the line we will see that the error message is displayed.

We obviously do not want the error message to be triggered here because we have not entered all seats yet. It is clear that, if we enter a single seat, we will have less than eight seats entered and the Error rule should be triggered. But what we need is for the seat number control to take place **after** the user has finished entering all seats.

## Example: Number of Seats in a Flight

The screenshot shows the GeneXus IDE interface. The top window displays the 'Structure' view for the 'Flight' entity, listing attributes: FlightId (Id, No), and FlightDepartureAirportId (Id, No). The bottom window shows the 'Rules' view for the 'Flight' 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 'Seat' entity structure in the bottom window, which lists attributes: FlightSeatId (Id, No), FlightSeatChar (SeatChar, No), and FlightSeatLocation (Location, No).

To make this possible we must condition the rule so it will be triggered after we have finished working with the grid lines. To do that we write the following: On afterlevel level FlightSeatChar.

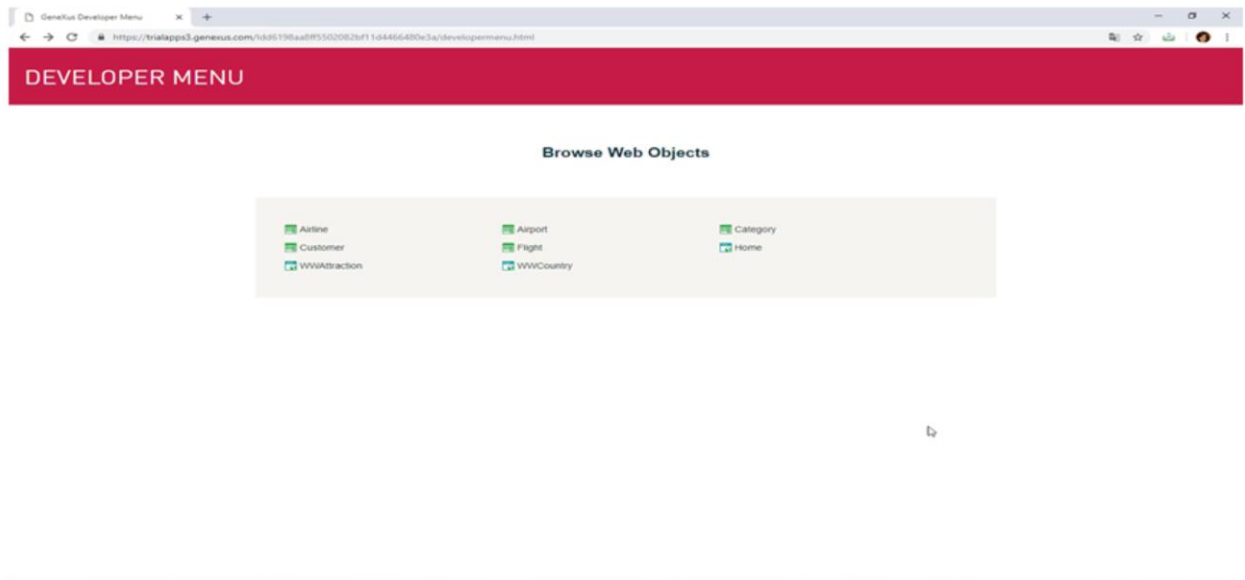
The "on after level" moment will cause the rule to be triggered after a level is finished. Since in our case this would have to be after the grid seat lines level is completed, we then add "level FlightSeatChar" because this attribute is at the seats level. We could have used any of the other attributes in the level, such as FlightSeatLocation.

This is how we instruct GeneXus that the rule must be triggered **after** entering the data in the location of the FlightSeatChar attribute, that is, after entering the header data for all seats on the flight.

The assessment carried out by the Error rule makes sense because, at the triggering moment, all seats that the user wished to enter will have been entered, and it will be possible to verify that at least eight seats have been entered.



## DEMO



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

We then press F5...

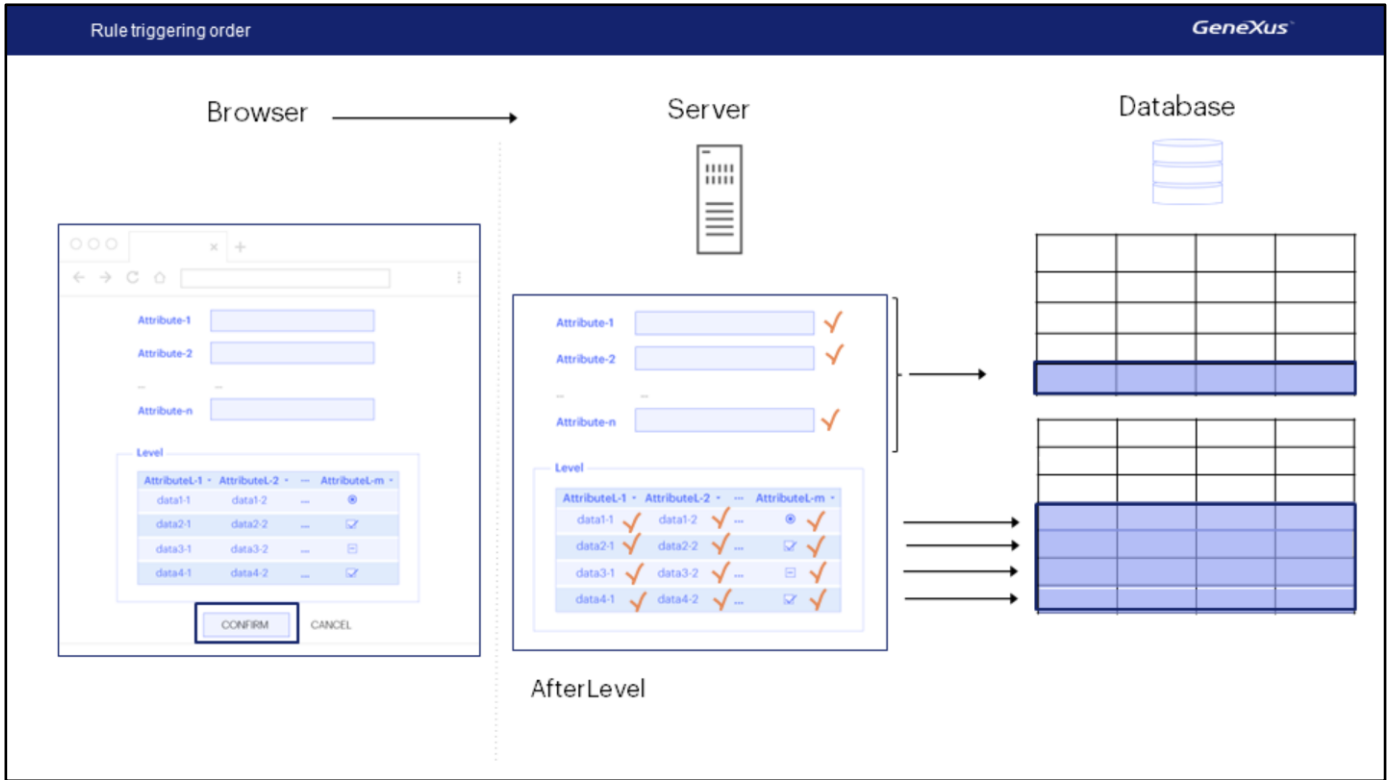
And now we open the Flight transaction, to enter a new flight.

We repeat the data that we used before... a flight from the Guarulhos airport to the Charles de Gaulle airport, with price of 3000, 10% discount and the TAM airline.

And then we enter the seats...

1, A, window  
1, B, aisle  
2, A, window  
2, B, aisle

And now we press Confirm, to indicate that we are finished entering the flight data (including the seats) and that the flight may be saved to the database.



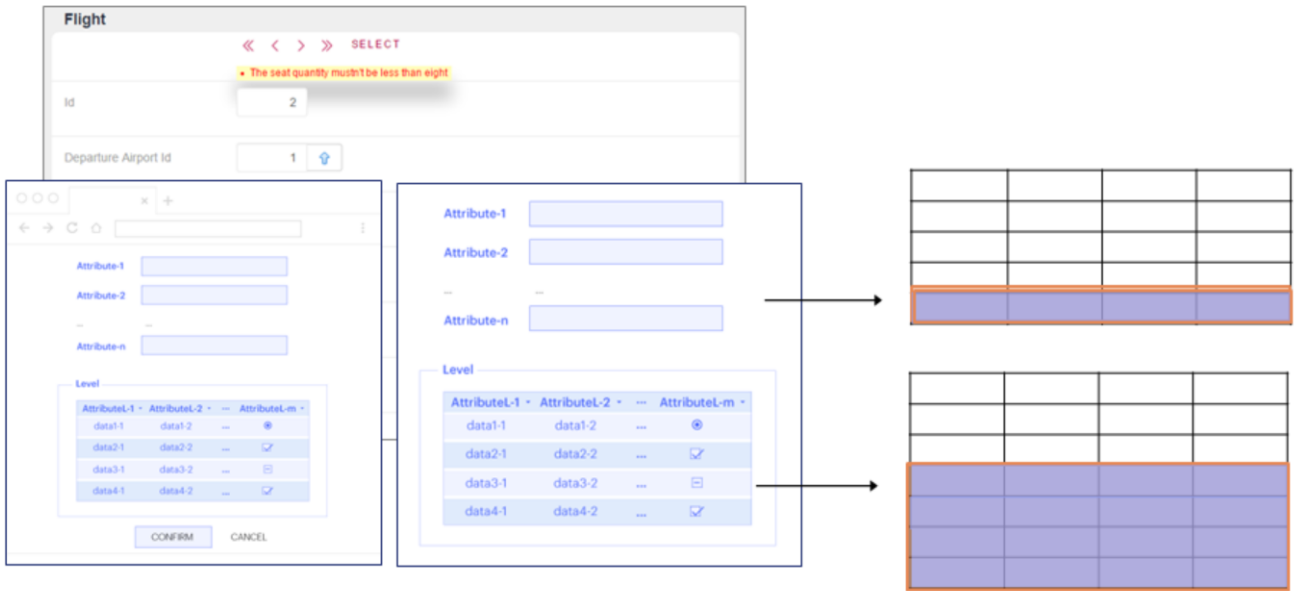
What happens from this point on? The header and lines data is sent to the server, and they are processed one by one, triggering the corresponding rules.

When all header data has been validated, the record is inserted in the table.

And the same is repeated for each line.

When the processing of all lines is completed, then comes the **AfterLevel** moment, when all the rules conditioned to that moment are triggered.

Note that the header and lines data is already saved in the database.



If we only enter 4 lines...

AfterLevel → ERROR

If we now go back to our transaction at runtime, we will see that upon Confirming, GeneXus will indicate the error, as expected, since we only entered four seats, and **it will not** save this flight to the database, because an Error rule will undo any saving done.

## Example: Number of Seats in a Flight

The screenshot shows a flight booking interface. At the top, the 'Capacity' is set to 8, which is circled in red. Below this is a 'Seat' section with a table of 8 seats. Each seat row has a red 'x' icon on the left, a 'Seat Id' column, a 'Seat Char' column, and a 'Seat Location' column. The seats are: 1 A Window, 1 B Aisle, 2 A Window, 2 B Aisle, 3 A Window, 3 B Aisle, 4 A Window, and 4 B Aisle. A red checkmark is placed to the right of the table. Below the table is a '[New row]' link and two buttons: 'CONFIRM' (highlighted in red) and 'CANCEL'. A red arrow points to the 'CONFIRM' button. To the right of the interface, a box contains the text '8 seats recorded'.

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

We complete the eight seats required by typing:

3, A, window

3, B, aisle

4, A, window... and to finish...

4, B, aisle

We now press Confirm, and we will see that GeneXus allows us to save the flight this time.

## Example: Number of Seats in a Flight

The screenshot illustrates a workflow in GeneXus. At the top, a flight form titled "Flight" shows a message "Data has been successfully updated." and a "SELECT" button. A callout box points to the form with the text "Flight with no seats!". Below this, a code editor window shows the rule logic for the "SELECT" event. The initial rule is:

```

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 arrow points to a second code editor window showing the updated rule logic:

```

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

```

At the bottom, the flight form is shown again with the message "The seat quantity mustn't be less than eight" and the "id" field set to 5. The "Departure Airport Id" field is set to 2.

In sum: we will achieve our goal by delaying the moment initially selected by GeneXus to trigger the Error rule.

We had flight 1 with 7 seats, because we added the error rule afterwards. As long as we don't try to save this flight, the error rule will not be controlled because, as we saw, it will be executed after CONFIRMING, when all the data is sent to the server.

If we press Confirm, we will see the message:

So, we add a seat to this flight:  
2-B-aisle

and we save, so now it will work.

Our last step will be to enter a new flight with no seats.

It allowed me to save it!

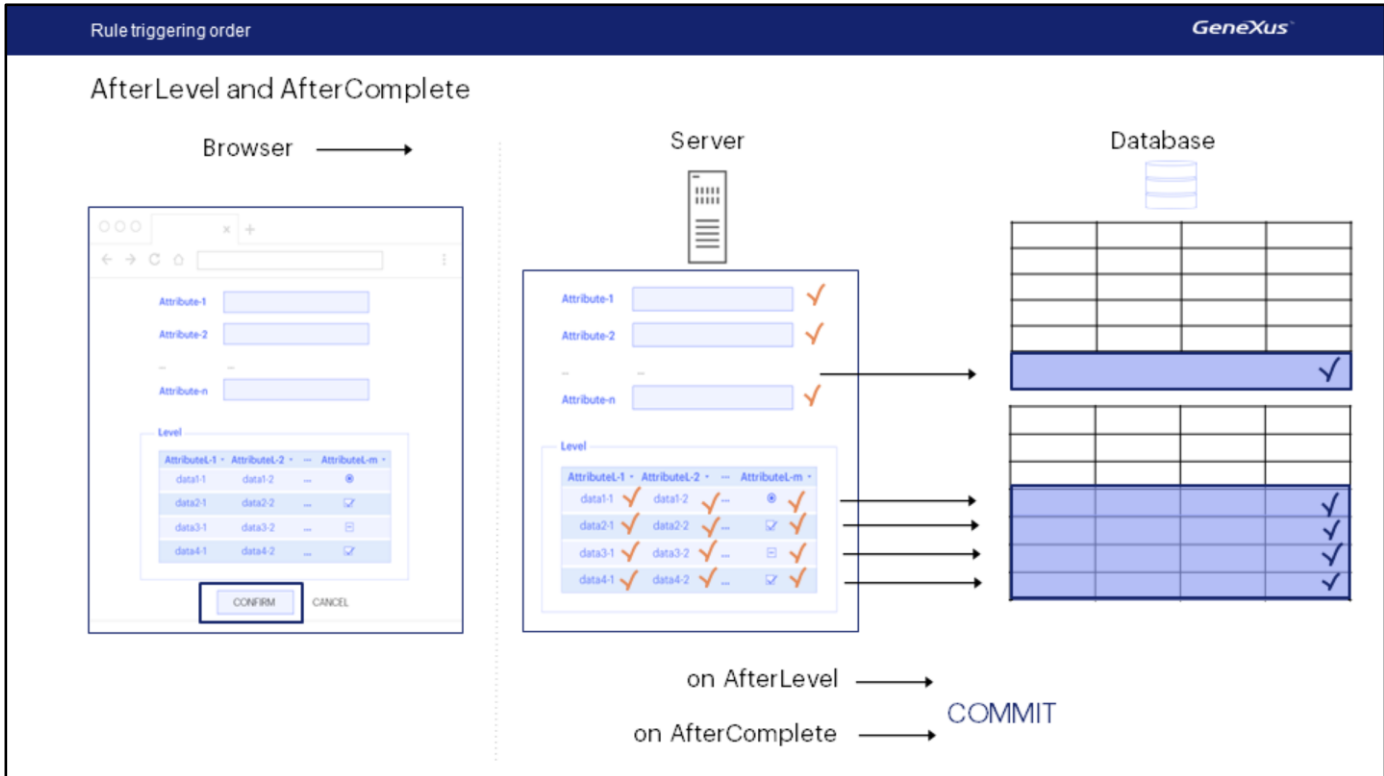
Why? Because we have the following condition:

which we entered wrongly, unaware of the existence of AfterLevel. So, we delete it and the rule will be written as follows.

We press F5, edit the flight with no seats, and we will now see that, if we Confirm again, the control will apply, not allowing us to enter it.

We eliminate it by pressing Delete.

Let's now get back to GeneXus to save the changes that we made to our Knowledge Base and to GeneXus Server.



With this example, we showed that there are cases where the moment selected by GeneXus to trigger a rule does not suit our needs, so we must tell GeneXus the moment in which we want the rule to be executed.

In this case we saw the “on Afterlevel” moment, to indicate that we want the rule to be triggered after we have run through one level.

This could be useful, for example, to call a listing that will print flight data, because, as we saw, in the AfterLevel, the data will already be saved to the database, though it will be deleted if an error rule is triggered. Everything that has been done will be undone, so if we are inserting data, it will be deleted. If we’re making changes, the changes made will be undone and the records will be restored to their previous values

For the case of the listing, it would be best to invoke it later on, when we are sure that the data (either new or modified) will not be deleted. This is possible after doing a Commit, a command that we will comment on further ahead, whose effect is to verify the data inserted as correct.

The moment following the Commit is **AfterComplete**, from where we could invoke the listing.

AfterInsert

Server



Database



Attribute-1  ✓

Attribute-2  ✓

...

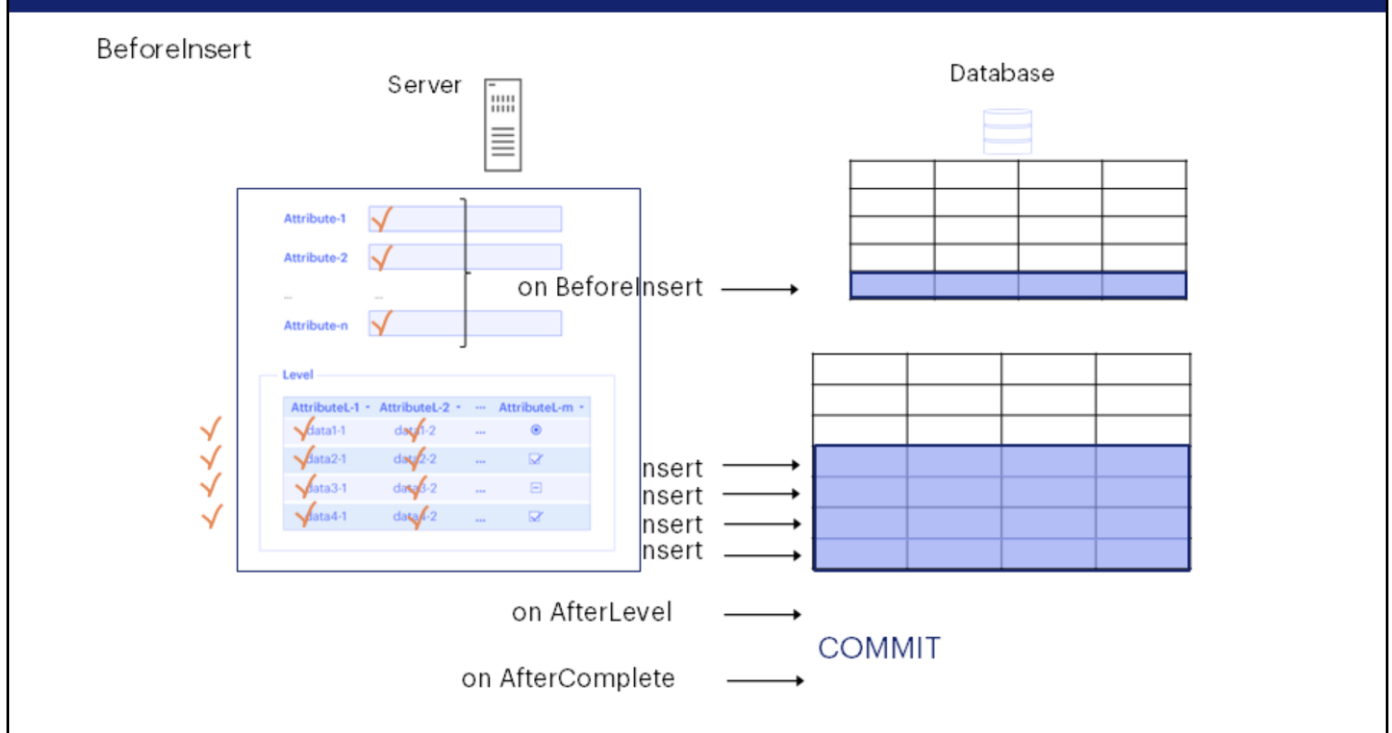
Attribute-n  ✓

Level

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



There are other moments such as “on AfterInsert” to indicate that the rule be triggered immediately following the saving of each header or line.



And “**on BeforeInsert**” for when we want to do or assess something **immediately before** the saving of the header or line data in the database.

Note that all the triggering moments start with the **on** prefix, and they are always written after the rule declaration.

Just as we have BeforeInsert and AfterInsert, which will be triggered only if we are trying to insert records, we have BeforeUpdate and AfterUpdate if we are trying to modify them, and BeforeDelete and AfterDelete if we are trying to delete them.

Here we have only presented the most important ones, but there are other triggering moments that we will not consider here, but which are available for you to find out about.



## Summing up

- Sometimes, the moment selected by GeneXus to trigger a rule is not the desired one. In these cases, we must specifically indicate when we want the rule to be triggered.
  - On BeforeInsert
  - On AfterInsert
  - On AfterLevel
  - On AfterComplete
- Rules are conditioning these events using the prefix "on" written at the end of the rule statement.

# GeneXus™

**The power of doing.**

Videos

[training.genexus.com](http://training.genexus.com)

Documentation

[wiki.genexus.com](http://wiki.genexus.com)

Certifications

[training.genexus.com/certifications](http://training.genexus.com/certifications)