Rule-Triggering Events in Transactions

Ending

GeneXus™

Rule Triggering Events in Transactions

Attribute

Rule

Formula

Attribute

Formula

Attribute

Formula

Attribute

Variable

on BeforeUpdate

on BeforeInsert

on BeforeDelete

on AfterUpdate

on AfterInsert

on AfterDelete

on AfterLevel Level att

on BeforeComplete
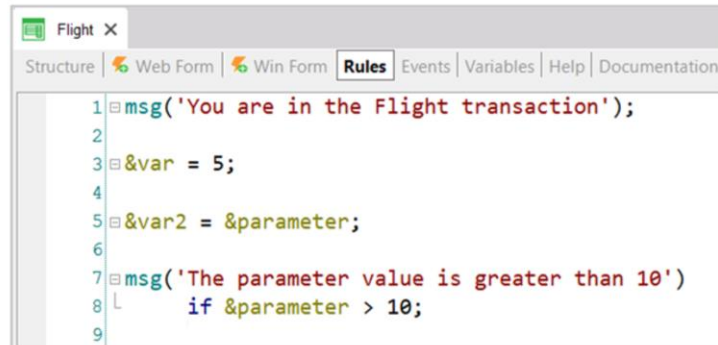
on AfterComplete
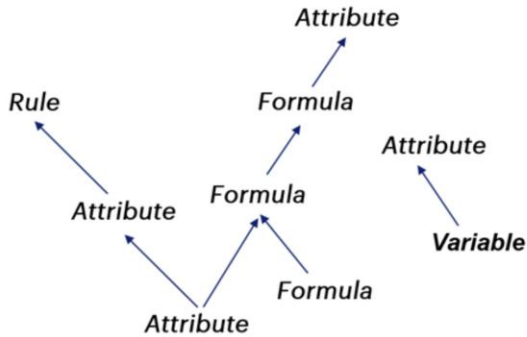
EVALUATION TREE

TRIGGERING EVENTS

Throughout the previous videos we have studied the evaluation tree that GeneXus creates to establish the order of execution of rules and formulas, as well as the triggering events available to condition the execution of the rules when the order established by the evaluation tree is not the one we need.

In this video, we will review and consolidate the knowledge acquired, and we will see a new triggering event that we haven't discussed yet.

Stand-alone Rules

- THEY ARE TRIGGERED AS SOON AS THE TRN IS EXECUTED

- THERE ARE NO CONDITIONS SET FOR THEIR EXECUTION, AND THEY DON'T HAVE TO WAIT FOR DATA IN ORDER TO BE EXECUTED

Flight X

Structure | Web Form | Win Form | **Rules** | Events | Variables | Help | Documentation

```
1  msg('You are in the Flight transaction');
2
3  &var = 5;
4
5  &var2 = &parameter;
6
7  msg('The parameter value is greater than 10')
8          if &parameter > 10;
9
```

The first rules to be triggered are the so-called stand-alone rules, which are those that don't depend on anything to be executed, or that already have the necessary information provided by the parameters received.

Rule Triggering Events in Transactions

Airline Id	1
Airline Name	TAM
Airline Discount Percentage	10
Final Price	900.00
Capacity	6

Seat

|	Seat Id	Seat Char	Seat Location
×	1	A ⌄	Window ⌄
×	1	B ⌄	Middle ⌄
×	1	C ⌄	Aisle ⌄
×	1	D ⌄	Window ⌄
×	1	E ⌄	Middle ⌄
×	2	A ⌄	Window ⌄

[New row]

CONFIRM   CANCEL

- THEY ARE TRIGGERED AS SOON AS THE TRN IS EXECUTED
- THERE ARE NO CONDITIONS SET FOR THEIR EXECUTION, AND THEY DON'T HAVE TO WAIT FOR DATA IN ORDER TO BE EXECUTED

RULES AND FORMULAS TO THE EXTENT THAT ASSOCIATED DATA FROM THE 1ST LEVEL IS AVAILABLE

After the execution of the stand-alone rules, the rules and formulas associated with the first level of the transaction that are not conditioned to events are executed, according to the evaluation tree, as the values involved to execute them become available;

Rule Triggering Events in Transactions

Airline Id	1
Airline Name	TAM
Airline Discount Percentage	10
Final Price	900.00
Capacity	6

Seat

	Seat Id	Seat Char	Seat Location	
×	1	A ∨	Window ∨	←
×	1	B ∨	Middle ∨	←
×	1	C ∨	Aisle ∨	←
×	1	D ∨	Window ∨	←
×	1	E ∨	Middle ∨	←
×	2	A ∨	Window ∨	←

[New row]

CONFIRM   CANCEL

- THEY ARE TRIGGERED AS SOON AS THE TRN IS EXECUTED
- THERE ARE NO CONDITIONS SET FOR THEIR EXECUTION, AND THEY DON'T HAVE TO WAIT FOR DATA IN ORDER TO BE EXECUTED

RULES AND FORMULAS TO THE EXTENT THAT ASSOCIATED DATA FROM THE 1ST LEVEL IS AVAILABLE

RULES AND FORMULAS TO THE EXTENT THAT ASSOCIATED DATA FROM THE 2ND LEVEL IS AVAILABLE.

and when working at the second level, for each line, the rules and formulas associated with that level of the transaction are executed, with the same criteria.

Rule Triggering Events in Transactions
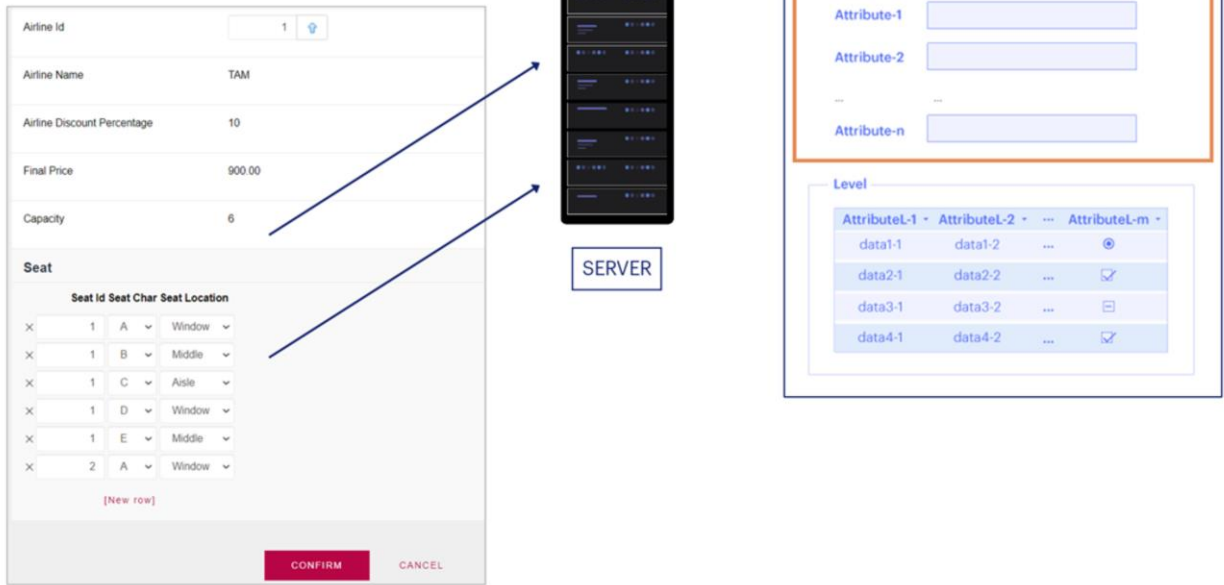
After the user presses Confirm, the data travels from the browser to the web server, which runs through the form again as if it were the user moving through each field one by one. It starts with the header, triggering those rules that don't depend on anything else, or those that depend on the mode (e.g. the Default rule).

Rule Triggering Events in Transactions
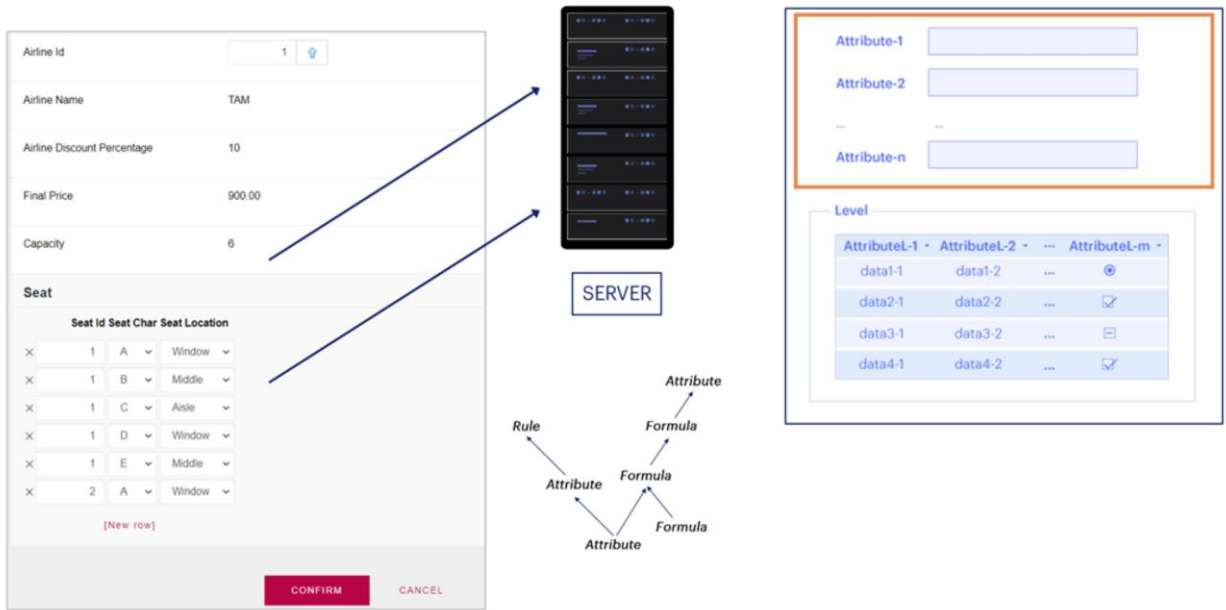
Then each field is run through and the corresponding rules are triggered according to the dependencies found.

Rule Triggering Events in Transactions

Airline Id	1
Airline Name	TAM
Airline Discount Percentage	10
Final Price	900.00
Capacity	6

Seat

	Seat Id	Seat Char	Seat Location
×	1	A	Window
×	1	B	Middle
×	1	C	Aisle
×	1	D	Window
×	1	E	Middle
×	2	A	Window

[New row]

CONFIRM   CANCEL

SERVER

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	...	☑

VALIDATION STAGE:

- Referential Integrity checks
- Concurrency control mechanism
- Values within the allowed range

Once GeneXus has finished executing these rules and formulas, a **validation** stage begins in which all the referential integrity checks are verified, the concurrency control mechanism checks that the relevant data has not been modified, and it is checked whether the values that were entered for the attributes are within their allowed range.

Rule Triggering Events in Transactions

	Seat Id	Seat Char	Seat Location
×	1	A	Window
×	1	B	Middle
×	1	C	Aisle
×	1	D	Window
×	1	E	Middle
×	2	A	Window

on BeforeValidate Event

VALIDATION STAGE:

- Referential Integrity checks

- Concurrency control mechanism

- Values within the allowed range

on AfterValidate Event

GeneXus allows us to condition the rules to be executed immediately before performing this validation, using the triggering event *on BeforeValidate*, and immediately afterwards, using the triggering event *on AfterValidate*.

On BeforeValidate Event
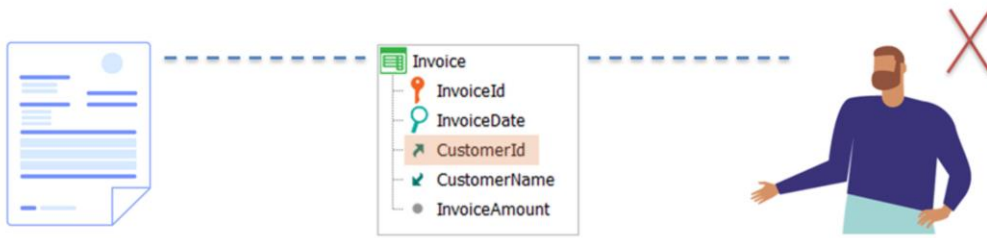
Selection List Invoice

	Date	Customer Id	Amount
✓	03/28/21	3	3422
✓	01/05/21	1	5200
✓	02/24/21	3	900

Selection List Customer

	Id	Name	Last Name
✓	1	Joseph	Smith
✓	2	Robert	Jones
✓	3	Anne	Brown

Let's see an example of use of BeforeValidate:

Let's suppose that we want to delete a customer from the system, who has an associated invoice:

On BeforeValidate Event

Customer

- Invalid delete, related information in Invoice ⬅

« ‹ › » SELECT

Id	1
Name	Joseph
Last Name	Smith
Address	5th Avenue
Phone	1111
Email	jsmith@gmail.com
Added Date	03/22/21 📅
Total Purchases	5

CONFIRM CANCEL DELETE

When trying to do so, we will see this error message, which indicates that it is not possible to perform the deletion because there is a related record in the Invoice table.

On BeforeValidate Event

DeleteRelatedInformation ×
Source | Layout | Rules | Conditions | Variables | Help | Docum

Subroutines

```
1  For each Invoice
2       where CustomerId = &CustomerId
3       delete
4  Endfor
```

DeleteRelatedInformation ×   Customer ×
Structure | Web Layout | Win Form | **Rules** | Events | Vari

```
1  DeleteRelatedInformation(CustomerId)
2       if delete
3       on BeforeValidate;
```

Customer

• Invalid delete, related information in Invoice

« ‹ › » SELECT

Id	1
Name	Joseph
Last Name	Smith
Address	5th Avenue
Phone	1111
Email	jsmith@gmail.com
Added Date	03/22/21
Total Purchases	5

CONFIRM   CANCEL   DELETE

To solve this, we can call a procedure that deletes the related information from the corresponding table, so that there are no data integrity violations when trying to perform the deletion.

This procedure must be executed <u>before</u> GeneXus starts making the referential integrity checks, which, as we have just mentioned, occurs during the validation stage, so we will condition the rule to be executed in the event on BeforeValidate:

On BeforeValidate Event

DeleteRelatedInformation

Invoices

CUSTOMER

Id: 1
Name: Joseph
Last Name: Smith
Address: 5th Avenue
Phone: 1111

CustomerId = 1   **CustomerId = 1**
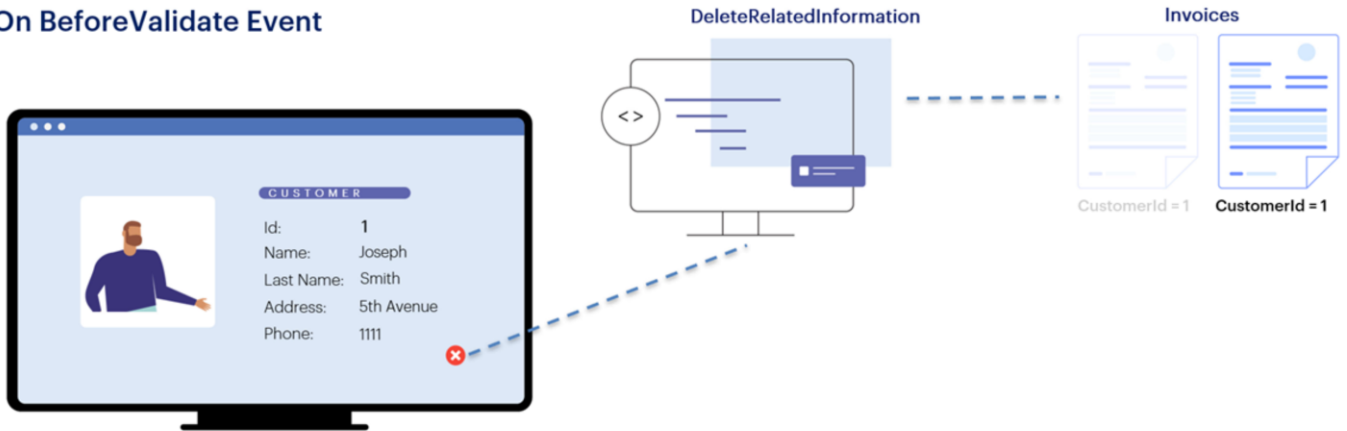
In this way, when we are about to delete the customer with ID 1, before starting the validation checks, the procedure will be invoked to eliminate the possible invoices associated with that customer.

On BeforeValidate Event

DeleteRelatedInformation

Invoices

No invoices for Customer 1

Customer 1 was successfully removed.

VALIDATION STAGE

Attribute-1	
Attribute-2	
...	...
Attribute-n	

No related information in Invoice
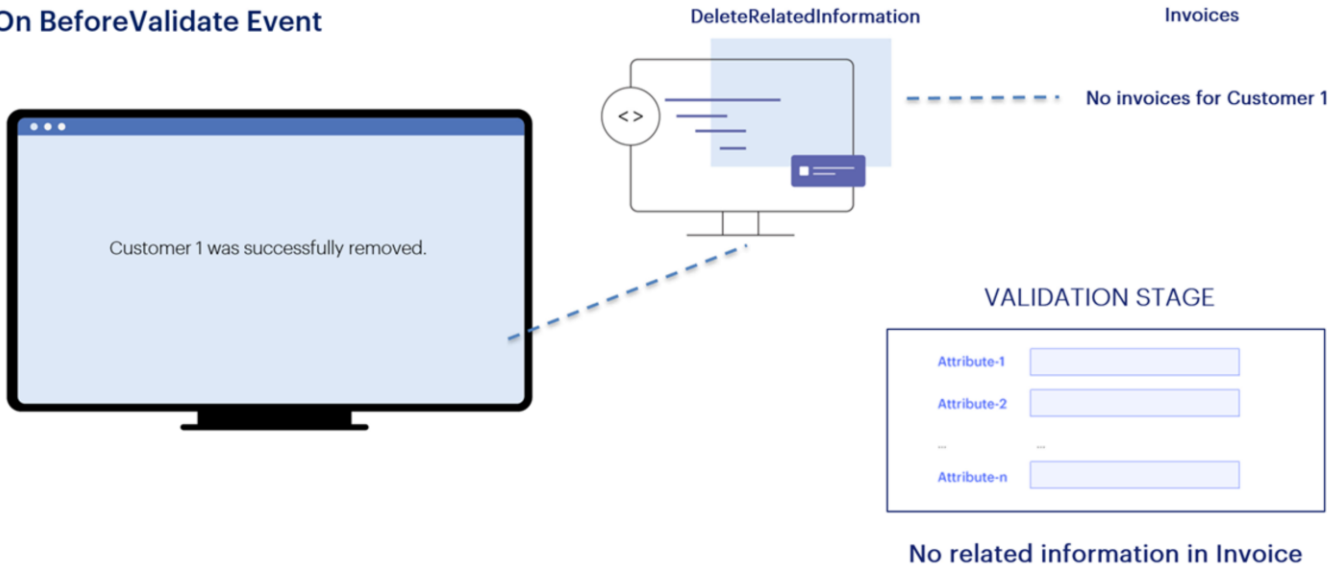
Therefore, when the validation is performed, no more related records are found, and the customer is deleted.

On BeforeValidate / On AfterValidate

In short, the **BeforeValidate** triggering event occurs right before the information of the instance we're working with (header or line) is validated. That is, it will take place right before the "header validation" or "line validation" action, as appropriate. Remember that here all the rules that are not conditioned to any triggering events and that are related with the level will have already been triggered; also, the triggering event **on AfterValidate** allows determining that a rule will be executed immediately after the data of that instance has been validated, before it is physically saved in the corresponding table.

All the rules that are conditioned to the event on BeforeValidate or on AfterValidate will be triggered at these precise moments. In what order? In the order in which they are written, unless there are dependencies between them, as we have already seen.

Rule Triggering Events in Transactions

Next, if the validation did not fail and:

- the recording corresponded to an insertion: the rules associated with the first level of the transaction that have *BeforeInsert* triggering event will be executed.
- if the recording corresponded to an update: the rules associated with the first level of the transaction that have *BeforeUpdate* triggering event will be executed.
- and if the recording corresponded to a deletion: the rules associated with the first level of the transaction that have *BeforeDelete* triggering event will be executed.

Rule Triggering Events in Transactions

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	...	☑

on *BeforeValidate*
VALIDATE THE HEADER
on *AfterValidate*

on *BeforeInsert / on BeforeUpdate / on BeforeDelete*
SAVE THE HEADER

⟶

The header data will be recorded in the corresponding table...

Rule Triggering Events in Transactions

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	...	☑

on BeforeValidate
VALIDATE THE HEADER
on AfterValidate

on BeforeInsert / on BeforeUpdate / on BeforeDelete
SAVE THE HEADER
on AfterInsert / on AfterUpdate / on AfterDelete

...and then the rules associated with this first level that have "on AfterInsert" event (if the recording corresponded to an insertion), "on AfterUpdate" (if the recording corresponded to an update), and "on AfterDelete" (if the recording corresponded to a deletion) will be executed.

Rule Triggering Events in Transactions

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	...	☑

on BeforeValidate
VALIDATE THE HEADER
on AfterValidate

on BeforeInsert / on BeforeUpdate / on BeforeDelete
SAVE THE HEADER
on AfterInsert / on AfterUpdate / on AfterDelete

(for each line)
on BeforeValidate
VALIDATE THE LINE
on AfterValidate

If the transaction is a two-level transaction, after recording the header, the rules and formulas of the second level attributes that don't have an associated triggering event will be executed, and afterwards this will happen for each line:

- Validation, so all the rules that are conditioned to **BeforeValidate** and **AfterValidate** events involving attributes of the second level will be triggered.

Rule Triggering Events in Transactions

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	...	☑

on *BeforeValidate*
VALIDATE THE HEADER
on *AfterValidate*

on *BeforeInsert / on BeforeUpdate / on BeforeDelete*
SAVE THE HEADER
on *AfterInsert / on AfterUpdate / on AfterDelete*

(for each line)
on *BeforeValidate*
VALIDATE THE LINE
on *AfterValidate*

on *BeforeInsert / on BeforeUpdate / on BeforeDelete*
SAVE THE LINE
on *AfterInsert / on AfterUpdate / on AfterDelete*

- and Recording of the line, so all the rules that are conditioned to the events **BeforeInsert** (or **BeforeUpdate**, or **BeforeDelete**) and **AfterInsert** (or **AfterUpdate**, or **AfterDelete**) involving attributes of the second level will be triggered.

Rule Triggering Events in Transactions

Database

Attribute-1		✓
Attribute-2		✓
...	...	
Attribute-n		✓

on *BeforeValidate*
VALIDATE THE HEADER
on *AfterValidate*

on *BeforeInsert* / on *BeforeUpdate* / on *BeforeDelete*
SAVE THE HEADER
on *AfterInsert* / on *AfterUpdate* / on *AfterDelete*

(for each line)
on *BeforeValidate*
VALIDATE THE LINE
on *AfterValidate*

on *BeforeInsert* / on *BeforeUpdate* / on *BeforeDelete*
SAVE THE LINE
on *AfterInsert* / on *AfterUpdate* / on *AfterDelete*

on *AfterLevel Level* Line attribute
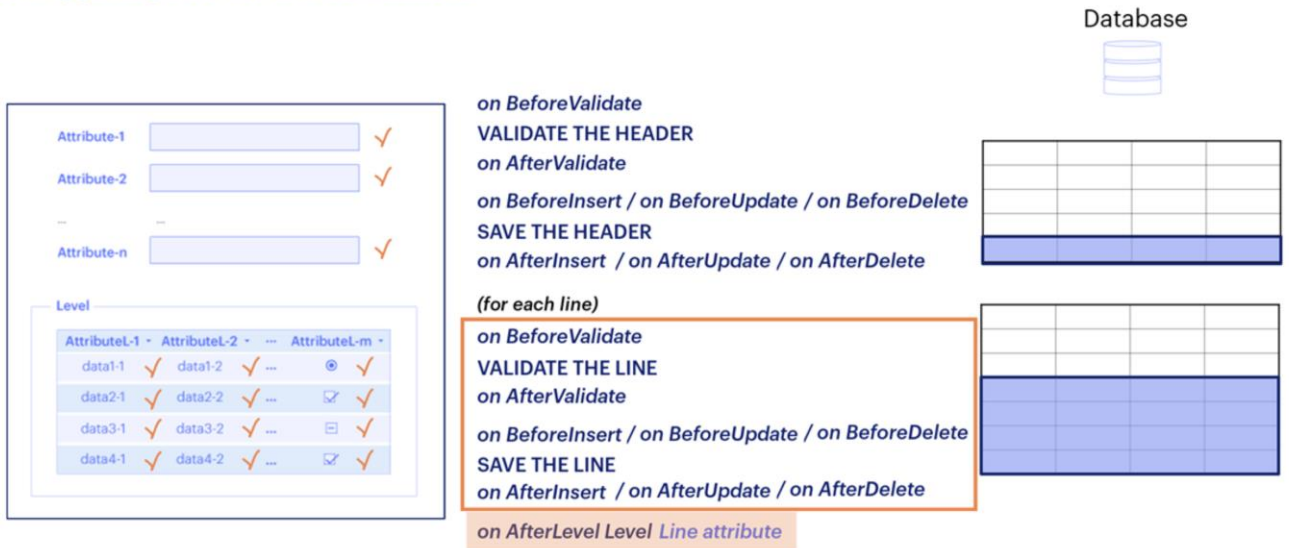
Next, the rules with triggering event **"on AfterLevel"** to do or evaluate something right after running through a certain level will be triggered;

If there is another grid, that is, another level, the same thing that was done for the first grid will be repeated and the on AfterLevel will also take place but of this other level. And so on until the last grid is completed.

Rule Triggering Events in Transactions

Database

Attribute-1

Attribute-2

...   ...

Attribute-n

on BeforeValidate
VALIDATE THE HEADER
on AfterValidate

on BeforeInsert / on BeforeUpdate / on BeforeDelete
SAVE THE HEADER
on AfterInsert / on AfterUpdate / on AfterDelete

Level

AttributeL-1 AttributeL-2 ... AttributeL-m

data1-1 data1-2 ...

data2-1 data2-2 ...

data3-1 data3-2 ...

data4-1 data4-2 ...

(for each line)
on BeforeValidate
VALIDATE THE LINE
on AfterValidate

on BeforeInsert / on BeforeUpdate / on BeforeDelete
SAVE THE LINE
on AfterInsert / on AfterUpdate / on AfterDelete

on AfterLevel Level Line attribute

on BeforeComplete
COMMIT
on AfterComplete

After having executed all the operations explained so far, a **COMMIT** will be performed, which will consolidate in the database the header data and all the lines of the transaction. We have the *"on BeforeComplete"* event to execute rules immediately before the Commit, and *"on AfterComplete"* which corresponds to the moment immediately afterwards.

```
STAND-ALONE RULES

RULES AND FORMULAS OF 1ST LEVEL ATTRIBUTES WITHOUT TRIGGERING EVENTS

RULES ASSOCIATED WITH THE 1ST LEVEL WITH TRIGGERING EVENTS
On BeforeValidate
VALIDATE the header
On AfterValidate
On BeforeInsert / BeforeUpdate / BeforeDelete
SAVE the header
On AfterInsert / AfterUpdate / AfterDelete

RULES AND FORMULAS OF 2ND LEVEL ATTRIBUTES WITHOUT TRIGGERING EVENTS

RULES ASSOCIATED WITH THE 2ND LEVEL WITH TRIGGERING EVENTS
        On BeforeValidate
        VALIDATE the line
         On AfterValidate                        For each line
        On BeforeInsert / BeforeUpdate / BeforeDelete
        SAVE the line
        On AfterInsert / AfterUpdate / AfterDelete
On AfterLevel Level Line attribute

On BeforeComplete
COMMIT
On AfterComplete
```

Learning the order in which the rules in a transaction are executed, the triggering events which are available to assign to them, the exact moment when they are triggered, and what actions take place before and after each triggering event is very important, because this knowledge is essential to be able to program the transactions' behavior correctly.

GeneXus™

training.genexus.com
wiki.genexus.com
training.genexus.com/certifications