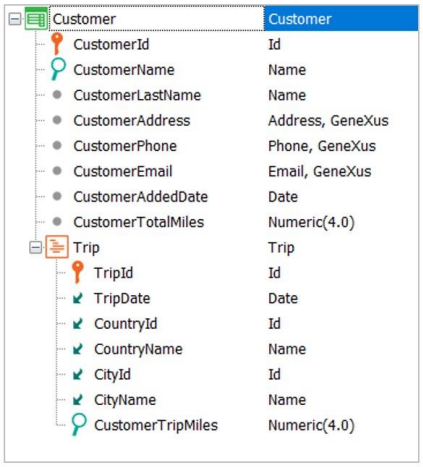
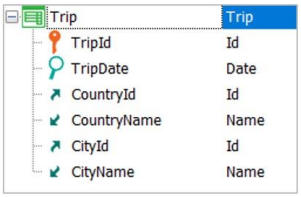
More rules to define behavior

GeneXus™

Add rule

Scenario: Customers accrue miles with every trip they make

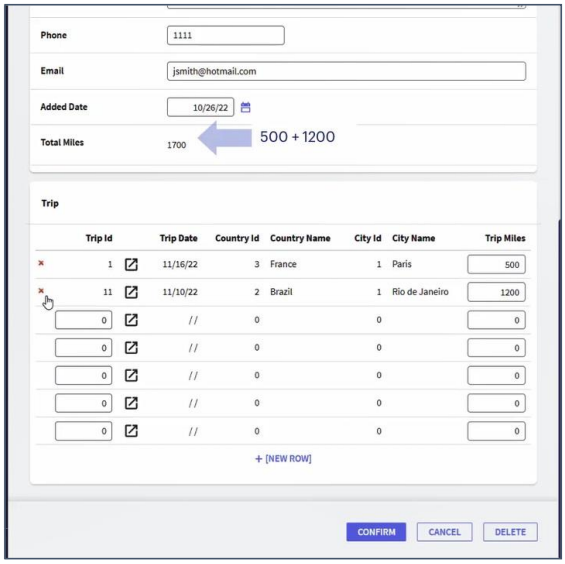


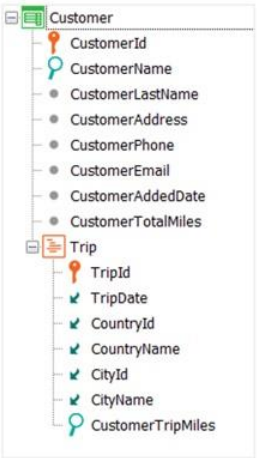


In this video, we will analyze the behavior of some rules that will help you simplify the development of your application.

Let's start with the Add rule: to understand how this rule works, suppose that a customer earns miles for each trip purchased.

Add rule



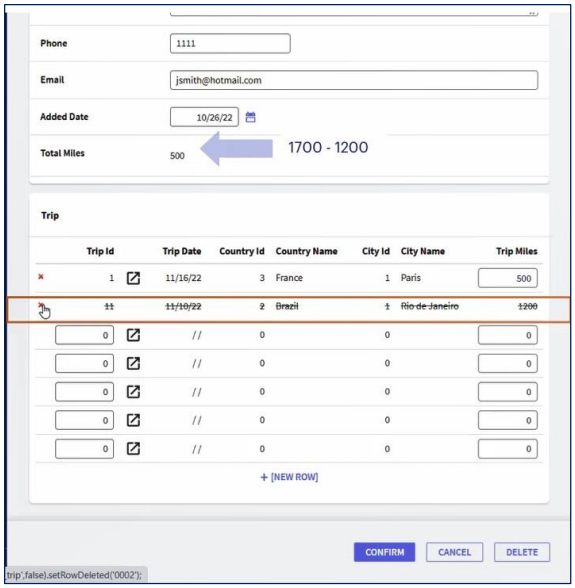


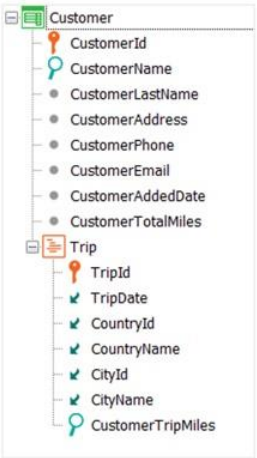

```
Add(CustomerTripMiles, CustomerTotalMiles)
```

Every time a trip is added for a customer, the miles corresponding to that trip must be added to the CustomerTotalMiles attribute, which contains the total miles earned by that customer.

To do so, we will use the Add rule.

Add rule



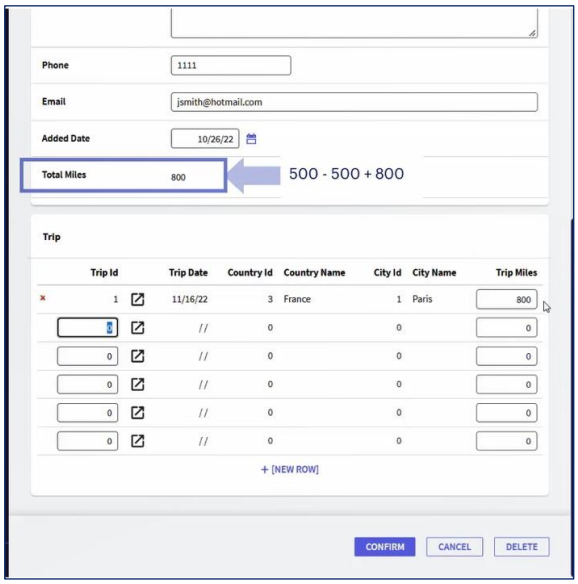


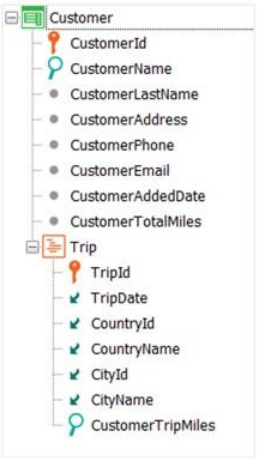
Add(CustomerTripMiles, CustomerTotalMiles)

But... what happens if, after doing this, the client cancels a trip?

In that case, the Add rule automatically subtracts the number of miles of the trip that is being deleted from the customer's total miles.

Add rule





Add(CustomerTripMiles, CustomerTotalMiles)

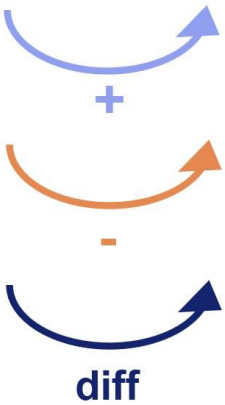
What if the number of miles awarded when taking a trip is modified; that is, the CustomerTripMiles attribute?

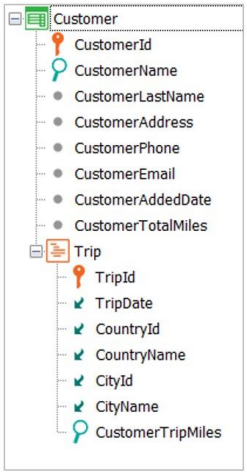
The rule subtracts the value of the miles associated with the trip from the customer's total number of miles, and then adds the new value, so that the information is up to date.

Add rule behavior

Add(CustomerTripMiles, CustomerTotalMiles);

- If a new trip is entered for the Customer
- If a trip is deleted for the Customer
- If a trip is changed for a Customer



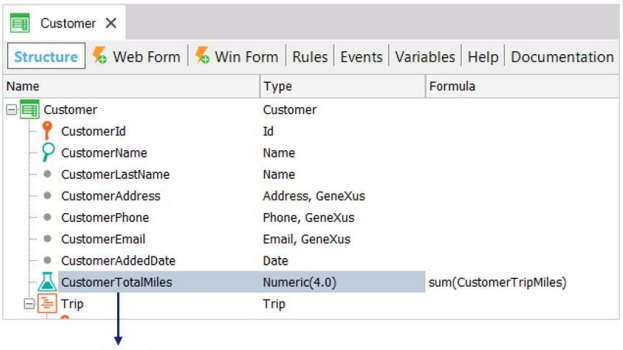


In short, this rule adjusts its behavior depending on how the transaction is being used: when inserting data, the value of the first attribute is added to the second one.

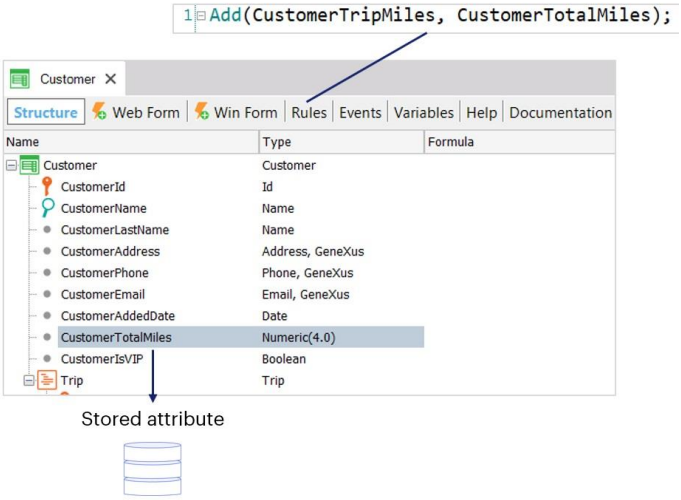
When deleting data, the value of the first attribute is subtracted from the second one.

And when changing data, the difference between the new value and the old value of the first attribute is added to the value of the second one.

Sum Formula / Add Rule



Virtual attribute



Stored attribute

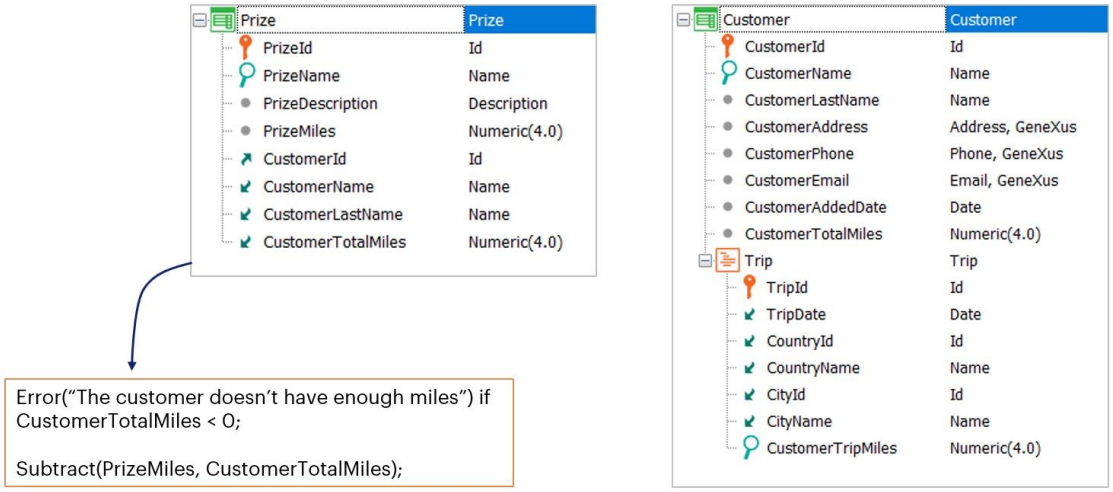
Perhaps the most natural thing in this case would have been to use a Sum formula, which would avoid all this.

But what if we don't always want the miles to be calculated according to this sum? What if we wanted to be able to increase the customer's miles according to some other criteria?
For example, because we want to give VIP customers miles as a gift from time to time. In this case, we need the customer's miles to be a stored attribute, which, although it is calculated by adding up the miles of each trip, it can also be modified by other means.

In the chapter on formulas we will study in more detail the difference between the Sum formula and the Add rule.

Subtract Rule

Scenario: Customers can trade their accrued miles for a reward if they have enough miles.



Let's now move on to the Subtract rule, which has a similar behavior to the Add rule.

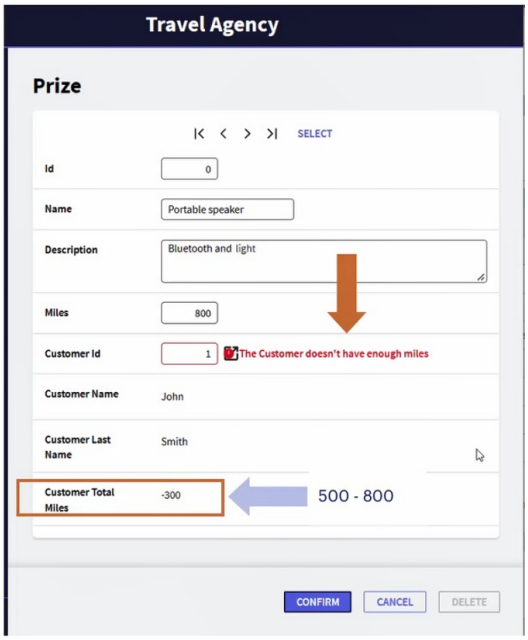
We have a Prize transaction that allows us to define rewards to be redeemed for miles. Each reward has a number of miles required to make the redemption, so when trying to assign a reward to a customer, you must confirm that the customer's miles are enough for the exchange. If they are enough and the reward is taken, the miles redeemed must be subtracted; otherwise, an error message must be displayed.

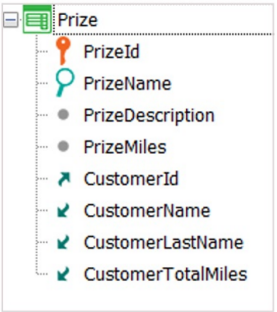
To this end, we will define these rules (show them) in the Prize transaction:

Error("The customer doesn't have enough miles") if CustomerTotalMiles < 0;
Subtract(PrizeMiles, CustomerTotalMiles);

Since both involve the CustomerTotalMiles attribute, with one rule updating the attribute and the other rule evaluating its value, GeneXus determines that it must first execute the subtraction that updates the CustomerTotalMiles attribute, and then evaluate what happened to its value.

Subtract Rule



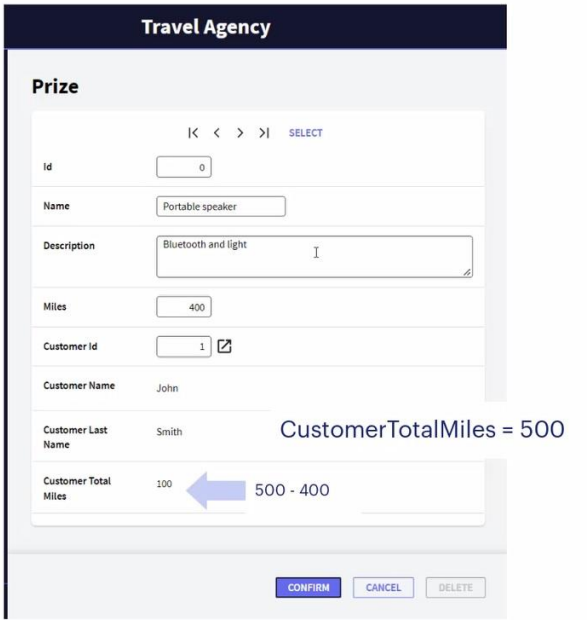


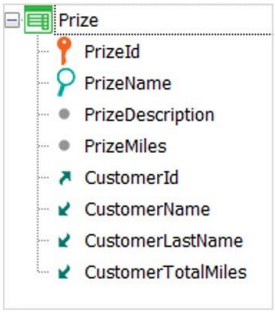

```
Error("The customer doesn't have enough miles")
if  CustomerTotalMiles < 0;

Subtract(PrizeMiles, CustomerTotalMiles);
```

Since the subtraction is made first, if the customer had fewer miles than those required by the reward, the CustomerTotalMiles attribute will end up with a negative value. This is why the error rule evaluates whether CustomerTotalMiles < 0.

If this happens, the error rule is triggered with the message that indicates it and the Subtract rule operation is undone; that is, its execution is reversed as if it had not been done and the client's total miles remain unchanged.

Subtract Rule

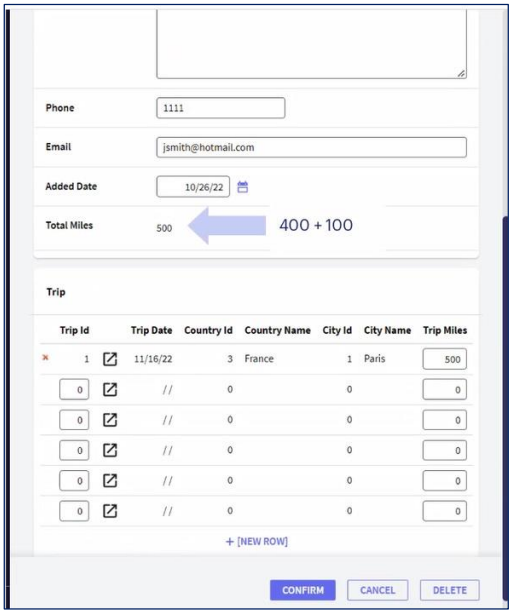


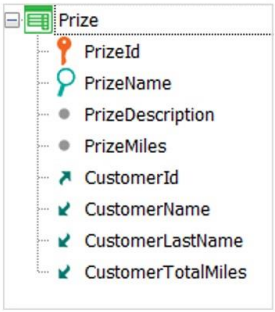



```
Error("The customer doesn't have enough miles")
if  CustomerTotalMiles < 0;

Subtract(PrizeMiles, CustomerTotalMiles);
```

If, on the other hand, CustomerTotalMiles did not end up with a negative value, the Subtract operation was carried out and the reward was associated with the customer, whose total number of miles decreased. All this provided, of course, that the user confirms it on the screen. Otherwise, this will only have been done in memory and nothing will be saved in the database.

Subtract Rule



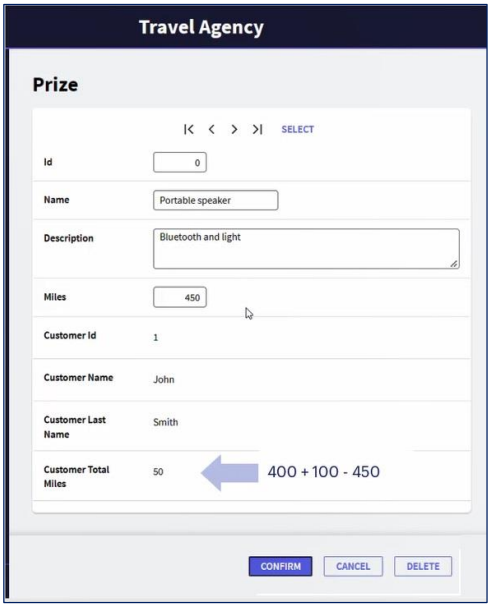


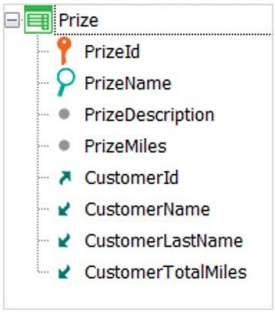

```
Error("The customer doesn't have enough miles")
if  CustomerTotalMiles < 0;

Subtract(PrizeMiles, CustomerTotalMiles);
```

What happens if after redeeming a reward, the customer changes his mind and wants to return it?

In this case, the Subtract rule adds the number of miles redeemed for the reward to the customer's total number of miles.

Subtract Rule






```
Error("The customer doesn't have enough miles")
if  CustomerTotalMiles < 0;

Subtract(PrizeMiles, CustomerTotalMiles);
```

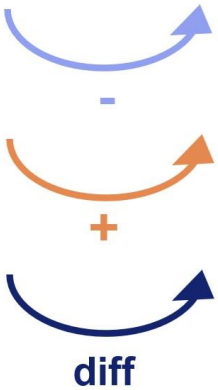
What if the value associated with the number of miles corresponding to a reward is modified, that is, the PrizeMiles attribute?

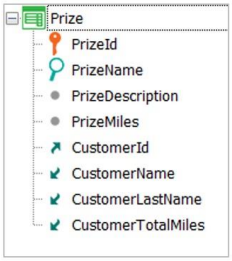
Its previous value is automatically added to the customer's total miles, and then the new value is subtracted.

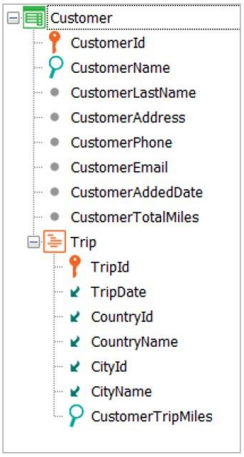
Subtract Rule behavior

Subtract(PrizeMiles, CustomerTotalMiles);

- If a new prize is entered for the Customer
- If a prize is deleted for the Customer
- If a prize is changed for a Customer

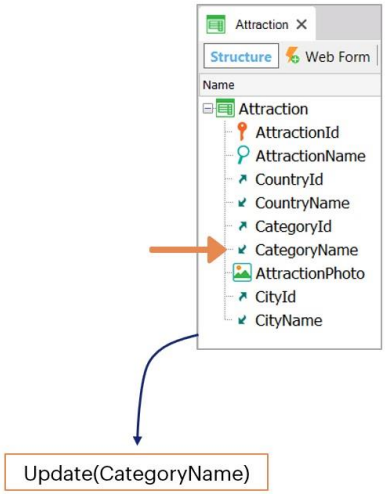


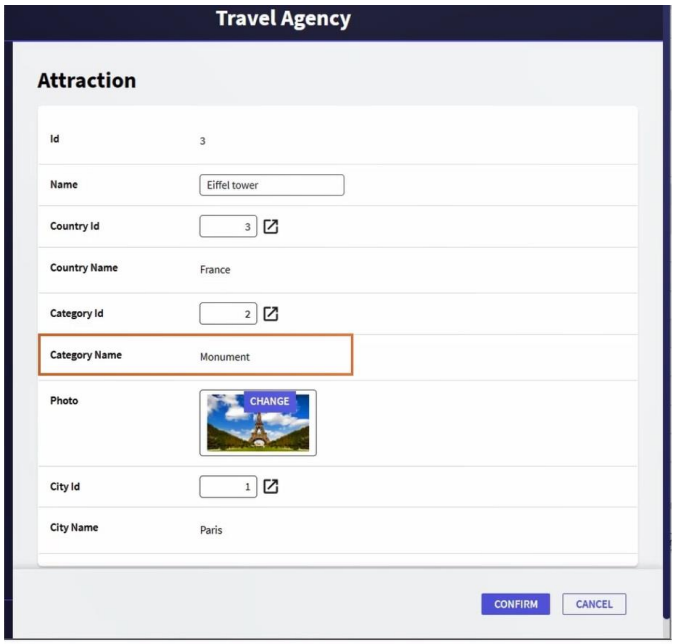




In short, the Subtract rule works in the opposite way to the Add rule: when inserting data, the value of the first attribute is subtracted from the second one.
When deleting data, the value of the first attribute is added to the second one.
And when making changes, the difference between the new value and the old value of the first attribute is subtracted from the value of the second one.

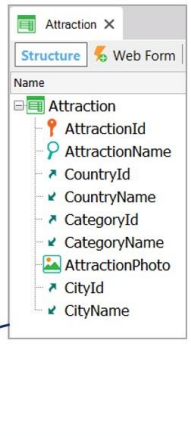
Update Rule



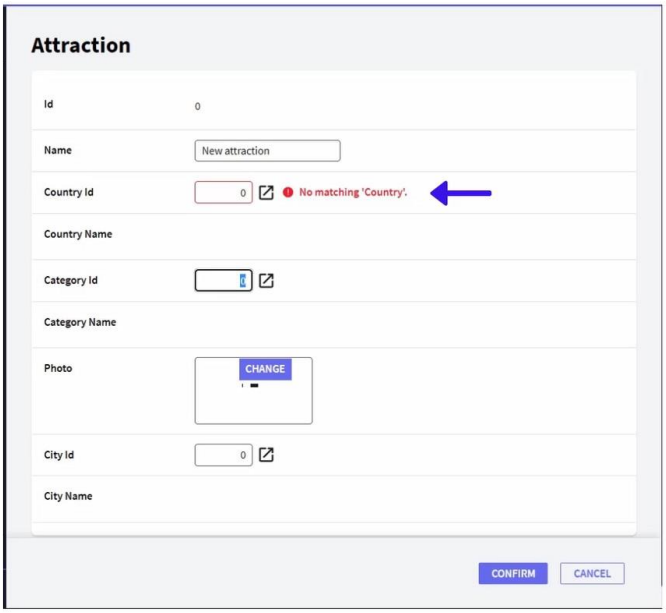


There are other very interesting rules that you can explore; for example, the **Update** rule, which allows you to modify the values of the attributes inferred from a transaction, updating them in their corresponding tables;

RefMsg Rule

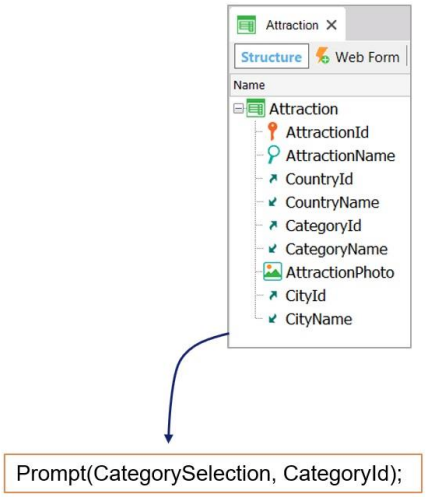


RefMsg("Enter a valid Country, please", CountryId);

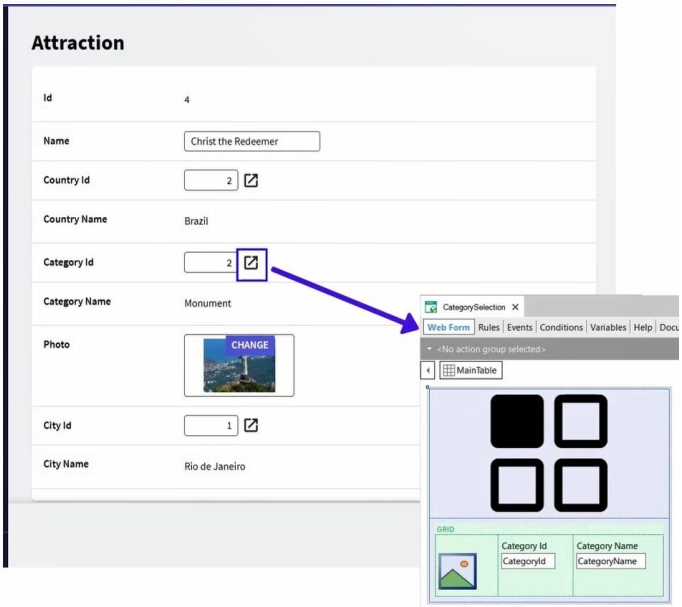


the **RefMsg** rule, which allows changing the default messages displayed by GeneXus when a certain referential integrity check fails;
;

Prompt Rule

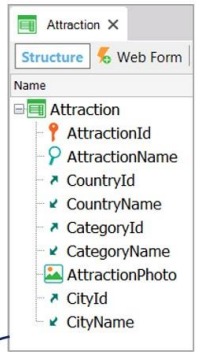


Prompt(CategorySelection, CategoryId);

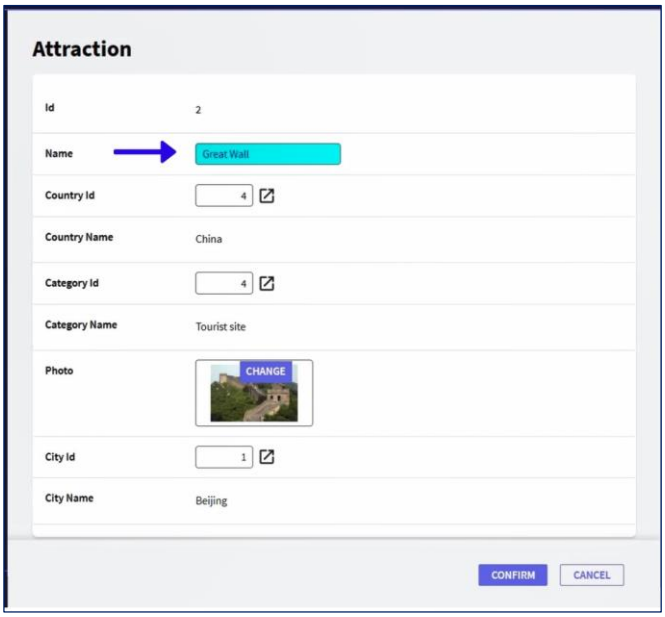


the **Prompt** rule, which allows changing the prompt or default selection list for each foreign key;

Color Rule

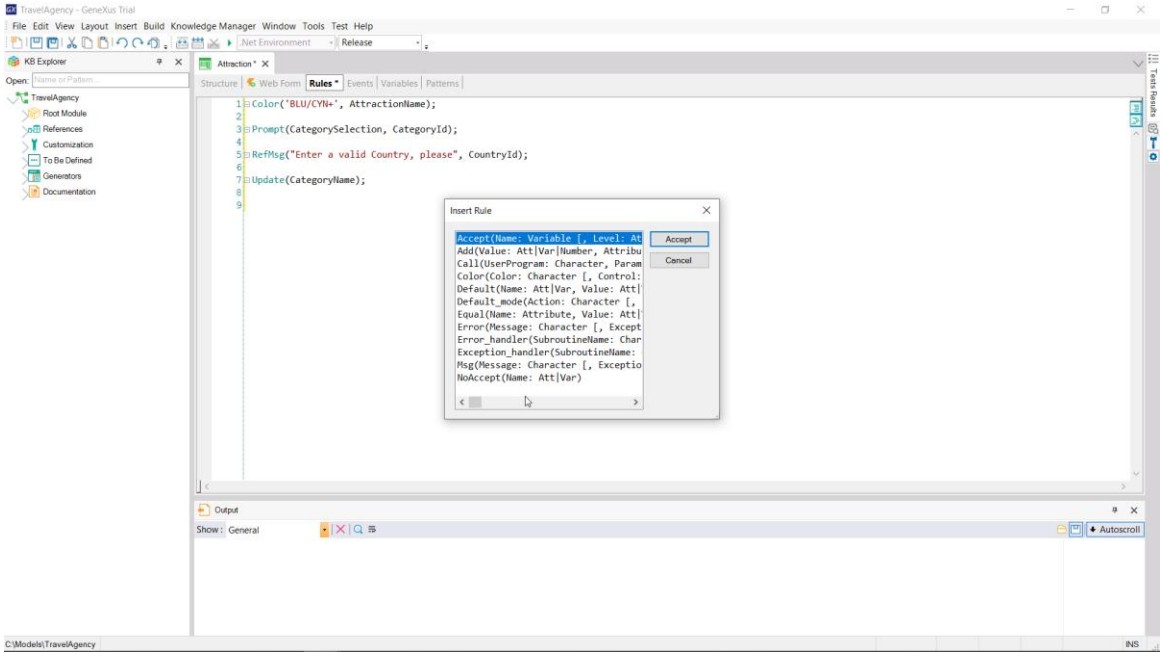



```
Color('BLU/CYN+', AttractionName);
```



the **Color** rule, which allows using colors to quickly improve the application's appearance;

More rules…



and many others that you can get to know through the Insert → Rule dialog…

We encourage you to discover its uses in the GeneXus Wiki.

GeneXus™

training.genexus.com
wiki.genexus.com