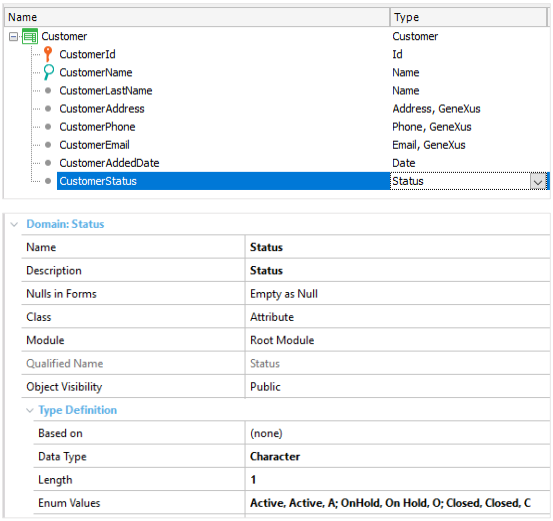
Data Selectors

GeneXus

GeneXus

Data Selectors



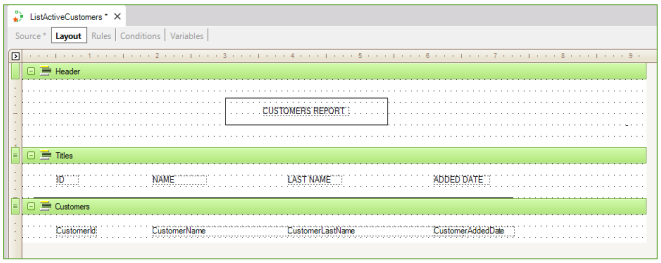
Suppose that the Customer transaction of our application has the CustomerStatus attribute to represent one of the three statuses (active, on hold, closed) that a customer can have in the travel agency system. A new enumerated data type has been defined for this purpose.

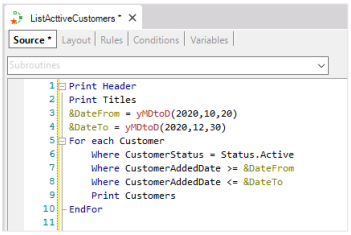
Suppose that in several places of the application we need to work with the active customers entered between two given dates.
For example:

To generate a PDF list that receives a date range and shows the active customers that were entered into the system between these two dates.

GeneXus

Data Selectors





≈

To this end, we create a procedure called ListActiveCustomers.

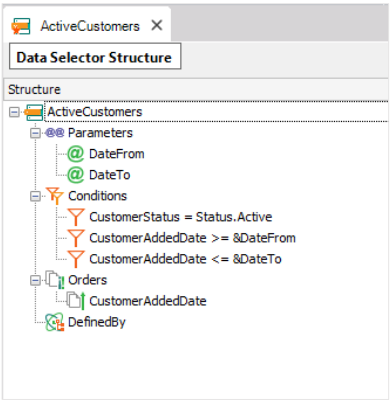
If we were to implement the queries with what we know so far, it would be through the following three conditions.

Since we are going to use these specifications in several places, to avoid repeating them in all the places where we need them, we can make these definitions in a single place, giving them a name, and from there on use that name as a reference. That place is the Data Selector object.

GeneXus

Data Selectors

Structure:



We create a Data Selector called "ActiveCustomers," and there we define the conditions that we have just seen.
Let's look at the structure of this object.

In parameters, we declare the parameters that the Data Selector will receive, which will then be used in Conditions. In this example, there will be two variables, &DateFrom and &DateTo.

In Conditions, we enter the conditions that we want to be fulfilled to filter the data to be retrieved. In this case, we indicate that the customer's status should be active. Also, the customer's entry date must be greater than or equal to the DateFrom variable, and lower than or equal to DateTo.

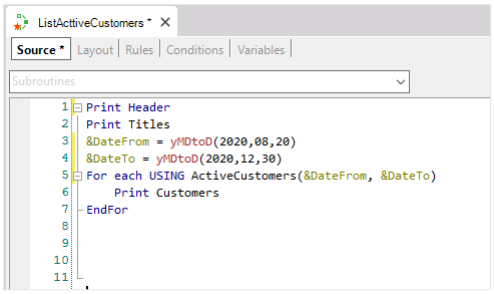
Then in Orders, we will be able to indicate the order in which we want to receive the retrieved data. In this case we enter CustomerAddedDate, to sort the data by date.

Lastly, we have Defined By, where we can enter an attribute or list of attributes that help to define the final base table. This may be useful to solve some ambiguity problem in determining the base table. For this example, we leave it empty.

GeneXus

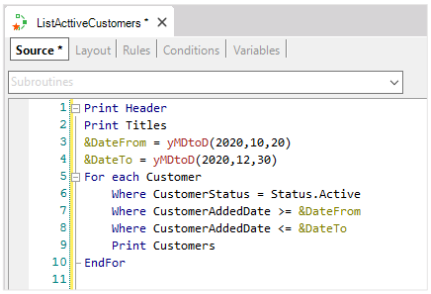
“Using” clause

With Data Selectors

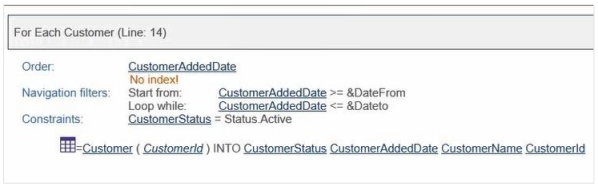


≈

Without Data Selector



Navigation View



As we’ve said, this centralized definition will allow us to reuse it everywhere this query is needed, making its maintenance easier (if we need to change something in the definition, it is made in one place and automatically applied everywhere the KB is used).

A Data Selector specifies, based on the parameters received, a set of conditions and orders for the data in a centralized manner, so as to avoid having to repeat the Order, Where and Defined by clauses everywhere they are needed.

Let's see how, once the data selector has been created, we would use it from our procedure that lists the customers.

We use it through the Using clause. Here we see how it should be used for the example given.
Within this For Each command we send a Using clause to the ActiveCustomers Data Selector, passing through parameters two variables that will correspond to the dates by which to filter the data.
Its behavior is similar to the previous specification.

Even when determining the base table, or resolving its navigation, it is the same as if instead of the DataSelector, its clauses had been written

directly.
Can order clauses and where clauses be added to the For each, in addition to what already comes from the Data Selector? The answer is Yes. The clauses add up.

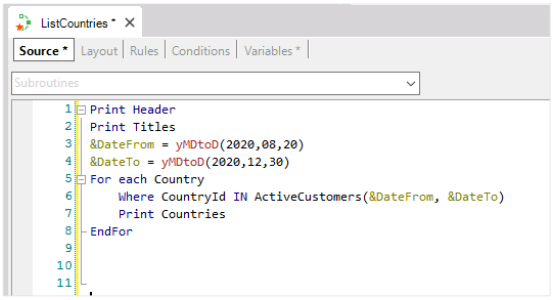
In the navigation list, we can see that the Customer table will be run through, it will be sorted by Customer AddedDate, the records will be run through according to the filters that we applied within Conditions, and as a restriction it will filter only the customers with active status.

GeneXus

"IN" operator

PDF Report:

ListActiveCustomers (Source)



Navigation View

For example, suppose that we added the customers' country, and we wanted to list the countries with active customers who have been entered into the system between two given dates. To solve this by taking advantage of our Data Selector we would do the following:

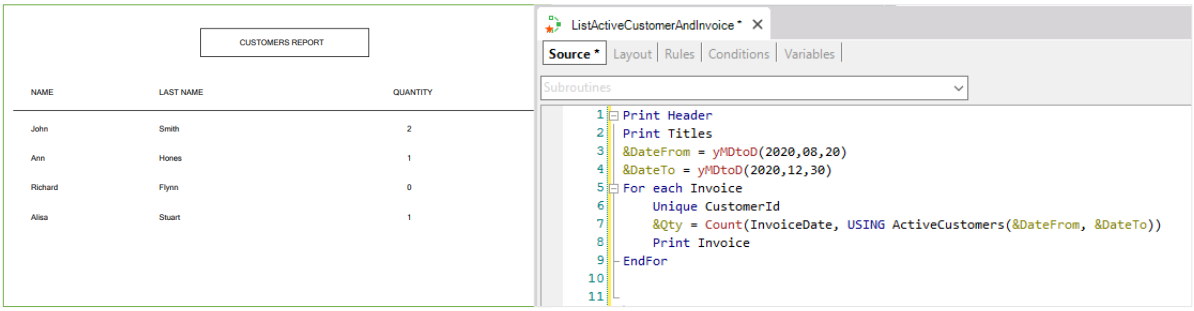
For each Country
Where CountryId in ActiveCustomers and we pass by parameter the variables DateFrom and DateTo.
In this case, instead of the Using clause, we are using the IN operator.
In this way, we use the Data Selector as if it were an independent database query.

There are two queries here: one of the Data Selector, which will return the set of active customers who have been entered between the two dates indicated and their corresponding countries. The other, corresponding to the For Each command, will filter the countries included in that set.

In the navigation list we see that the entire Country table is run through, sorting by CountryId, and as restriction (Constraint), is what we declared inside the where clause of the For each; that is, we are interested in filtering only the countries that are within the list of customers of the Data Selector.

GeneXus

Using Data Selectors in formula



Here is a third example where we can use our Data Selector.
For this case, suppose that we need to show in a PDF list all the customers with invoices and their amount, entering a range of dates to be able to count only the invoices of active customers entered into the system between those dates. If a customer is inactive or was entered outside those dates, it will be included in the list but the number of invoices will show zero.

Here we use the data selector within a formula, more specifically within the Count formula.
Remember that the second parameter of an Aggregate formula is for writing the conditions that must be met by the records in order to be "aggregated."

If we look closely at this case, we are running through the Invoice table in the For each. As each customer can have N invoices, we place the Unique clause to keep one of those invoices for the client, so with the Count formula on the same table, we count the invoices of that customer, which also comply with the clauses of the DataSelector: that is, if the customer is not active, the Count will give 0, because for all its invoices the customer will not be an active one. If instead the customer is active, only the invoices within the considered range will be counted.

GeneXus

Ways to use Data Selector in For each

- **USING** clause ≈ Add where clauses in the For each

The attributes intervene in the determination of the base table of the For Each

- **IN** operator

Different queries to the database.

The attributes do **NOT** intervene in the determination of the base table of the For Each

Summarizing what we have seen so far, the Data Selector can be used in two very different ways. If it is through the USING clause, there will be no difference with having the Order or Where clauses of the Data Selector directly inside the For each. Or as we've just seen, if we use it inside an aggregate formula, it is similar to entering there the conditions that we entered in "Conditions" in the Data Selector.

As mentioned before, by declaring the Using clause, we will be telling it to add the orders and filters of the Data Selector to those of the For each. For this reason, the attributes included in the Data Selector will have to belong to the extended table of the For each command's base table.

On the other hand, if we use the IN operator in a Where clause, for example, we are making an independent query, so the Data Selector, in order to be resolved, will have to navigate its base table as if it were an independent For each.
The IN operator is used when you want to perform a subquery, whose result will be used later as a filter in the For each, as we have just seen.

Depending on the way in which we invoke it, the attributes defined within the Data Selector will or will not take part in determining the table base of the For each:

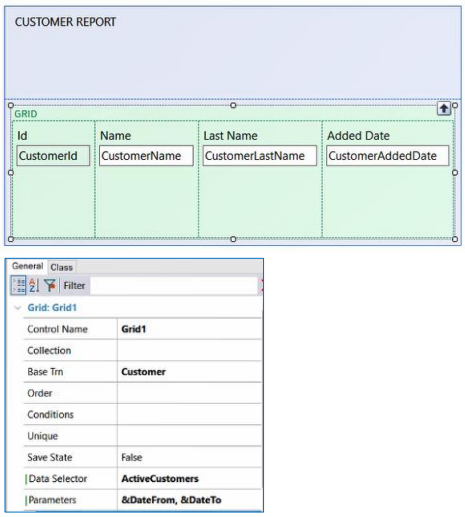
If we invoke the Data Selector through the USING clause, the attributes declared within this object are involved in determining the base table of the For each where we are calling it.

If we invoke it through the IN clause, the attributes of the Data Selector are not involved in determining the base table of the For each.

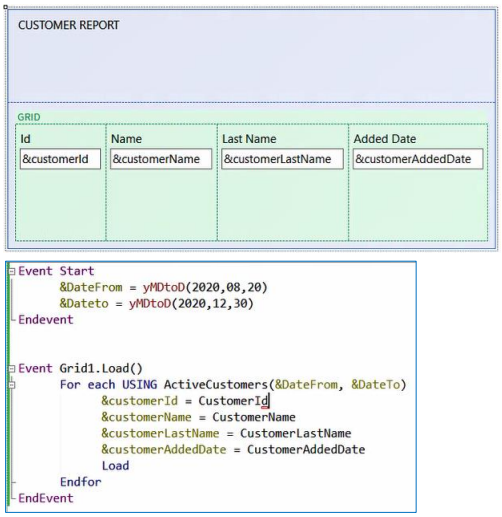
GeneXus

Data Selector In Web Panel

Web Panel with base table



Web Panel without base table



The three examples we have just discussed were made on procedural objects, since we wanted to list the information in PDF files. But if we simply wanted to display the information on screen through a Web Panel, how would we do it?

Let's go back for a moment to the first example, in which we were interested in listing the active customers in a given date range, using the Data Selector we created. But this time we will show the records on screen with a Web Panel.

In the Web Layout of our Web Panel, we declare a grid, and add the attributes we are interested in listing.
Within the Grid properties, we see one called Data Selector. There we must enter the name of our Data Selector that we want to use. This is similar to when a procedure object is invoked from the For each through the Using clause.

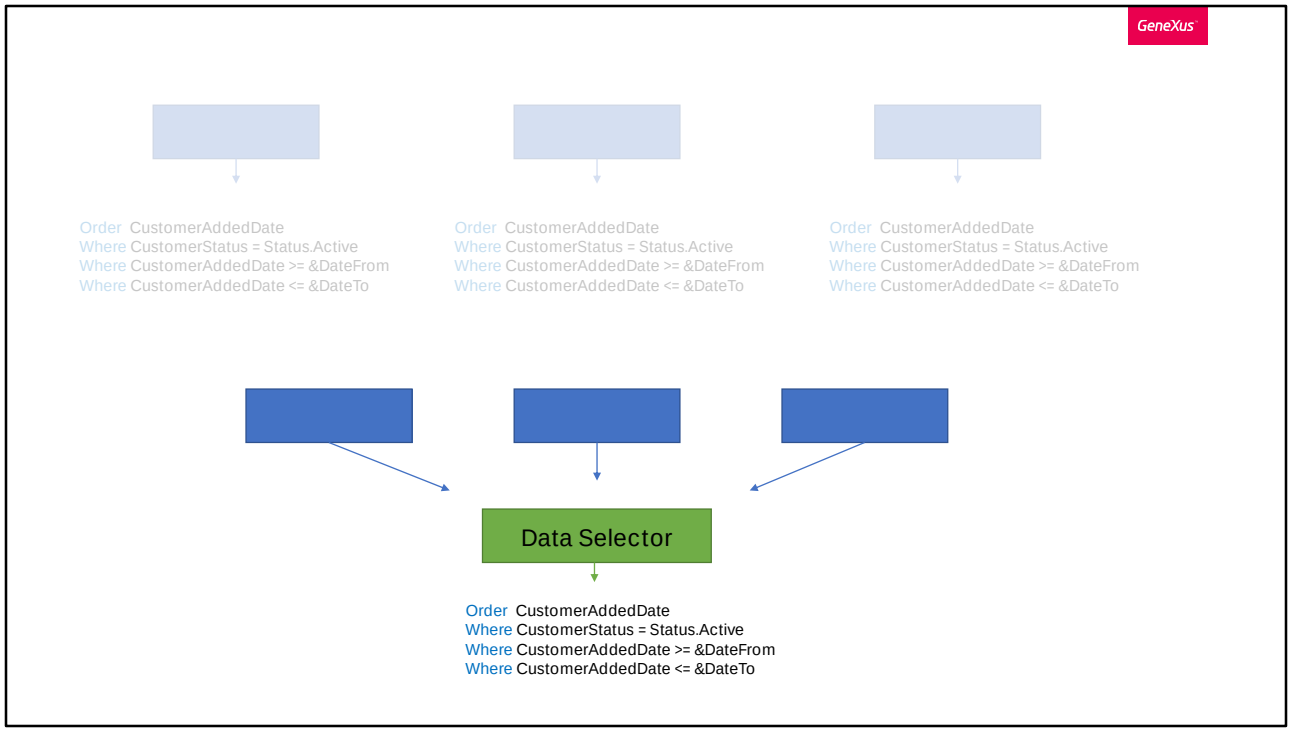
Let's try it. We see that it throws an error, since we remember that the ActiveCustomers Data Selector received by parameter two variables, the ones that will contain the dates. So we will have to pass it this information.
For this we create DateFrom and DateTo variables of type Date. In the Start event we declare them and for this test we initialize them with these dates.
To pass the parameters to the Data Selector we do it from the Parameters property of the grid.

Now we execute again.
We can see that the clients are shown on the screen, filtering by the active ones and between the indicated dates. Sorted by date entered, as defined in the Data Selector.

Let's consider the same example, assuming that the Web Panel, instead of having attributes, has variables. In this case it is a Web Panel without a base table, unlike the previous one. For this reason, here it will not be useful to use the Data Selector property of the grid, since to run through all the clients we will have to do it with a For each, and it is there where we must call our Data Selector. So in these cases, the statement will be similar to the one we saw with the procedure object.

In the events section, we must declare the Load event of the grid. Inside a For each with the USING clause followed by the name of our Data Selector, passing it by parameter the variables that it will receive to filter the dates.
And inside the For each, to the grid variables we assign the value of the attributes that we are interested in showing.
Lastly, we define the Load command, and close the For each and the Load event.

We execute, and see the desired results.



In short, the Data selector is an object that allows us to store a set of parameters, conditions, orders, to be used/invoked from different queries and calculations, and to reuse the same navigation several times.

So, where can we use a Data Selector? In all cases that make database queries. For example, in the grids of panels and web panels or in groups of Data Providers.

You can find more information about Data Selectors in our wiki.

GeneXus™

training.genexus.com
wiki.genexus.com