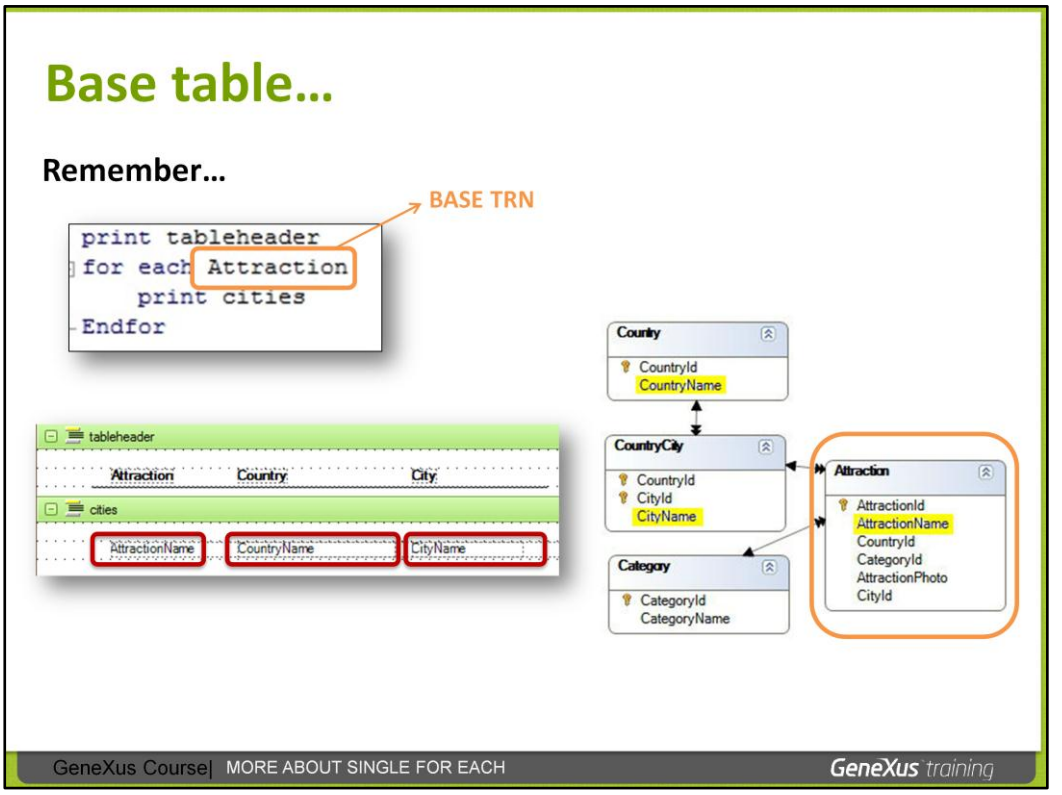
FOR EACH COMMAND

More about single For each, indexes, clauses, etc.

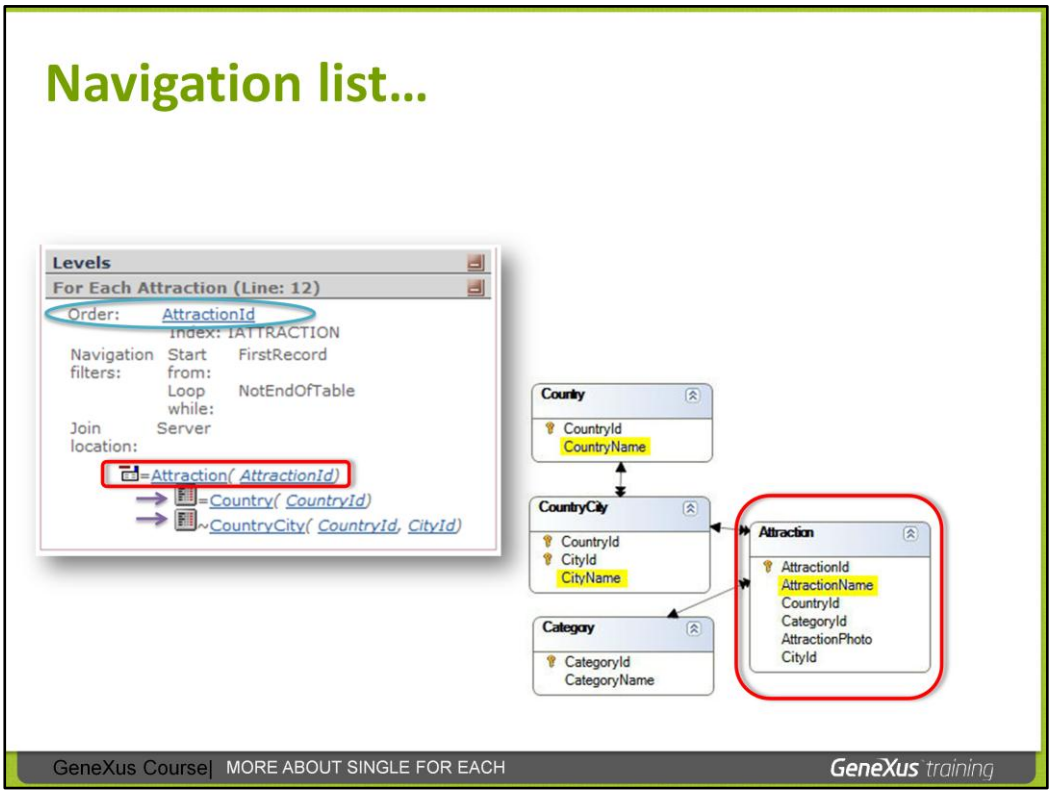

BASE TABLE



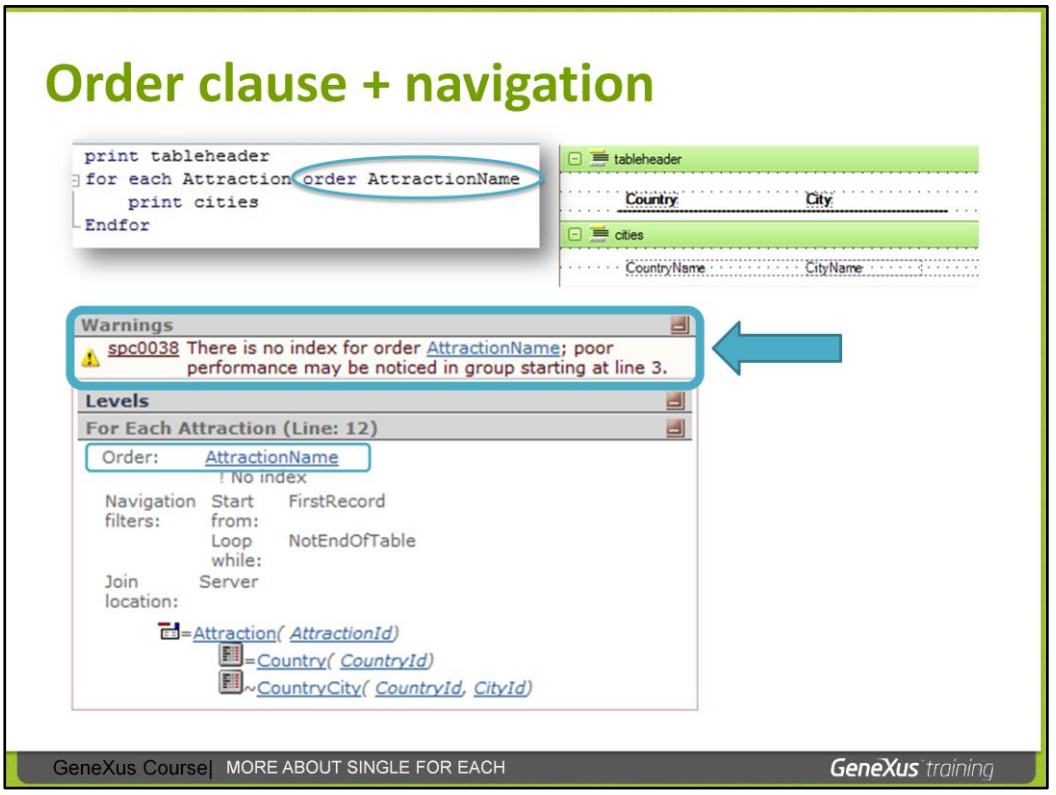
Remember that GeneXus determines the base table of the For each by the transaction name that is mentioned nexto to it. In the example: Attraction
Why? Because it is the name of the transaction **whose associated physical table** we want to navigate.

In addition, all the other attributes (in the for each body, printblocks, etc) must belong to the extended table of the base table of the For each.

In our example, we called Attraction as Base transaction and, as we have said, it corresponds to the name of the transaction whose associated physical table we want to navigate.



The navigation list informs us that the base table is ATTRACTION, that the navigation order will be determined by the primary key, AttractionId, and that the entire table will be run through. COUNTRY will be accessed –to retrieve CountryName, the attraction country-, and COUNTRYCITY to retrieve CityName.



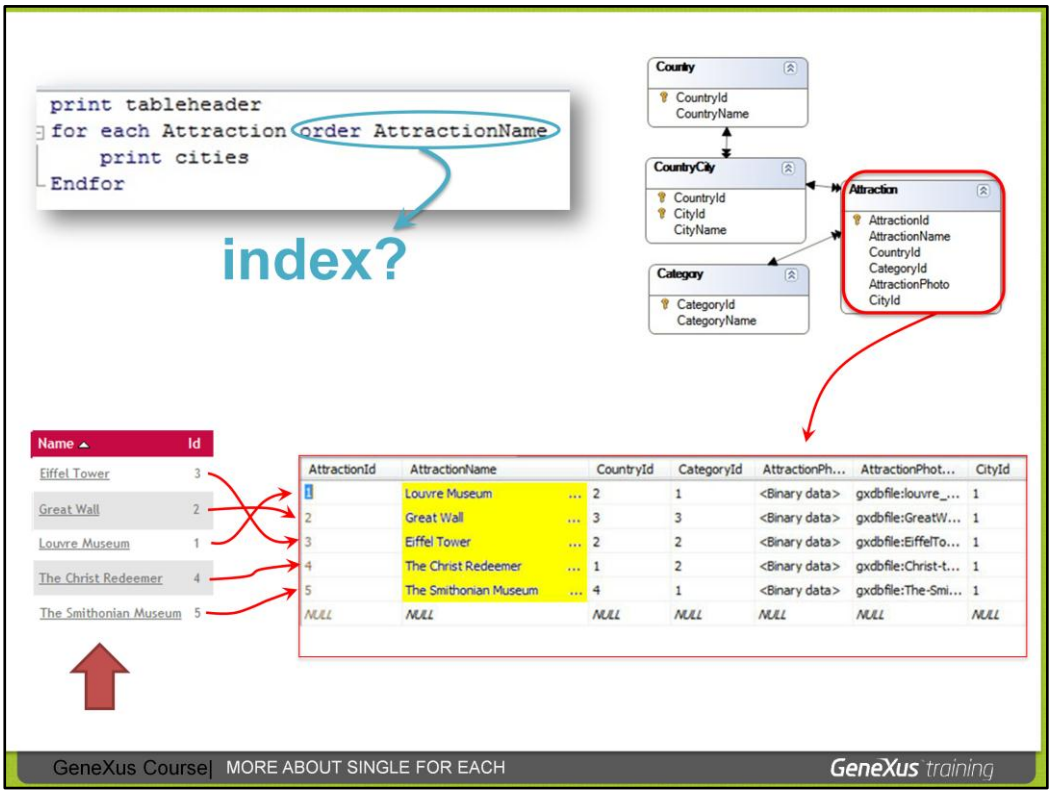
We would obtain the same base table (Attraction) if instead of adding the AttractionName attribute to the printblock we added it, for instance, in the order clause.

In this case the navigation will be ordered by **AttractionName**, and for each record the Country table will be accessed through the foreign key value, CountryId, to retrieve the CountryName value. The same will happen with CountryCity to retrieve the CityName value.

In the pdf file, only the country and city of each attraction will be printed (because we didn't include AttractionName to the printblock).

In addition, the navigation list informs us that the database doesn't have an **index** for the attribute used to order data, so we could experience low performance for this query. Why?

ORDERS AND INDEXES



Suppose that the Attraction table contains the data displayed on the slide. If we need to obtain its records ordered by the AttractionName attribute, it will have to be reordered somehow by this attribute.

The existence of an index at the physical level table would optimize this query.

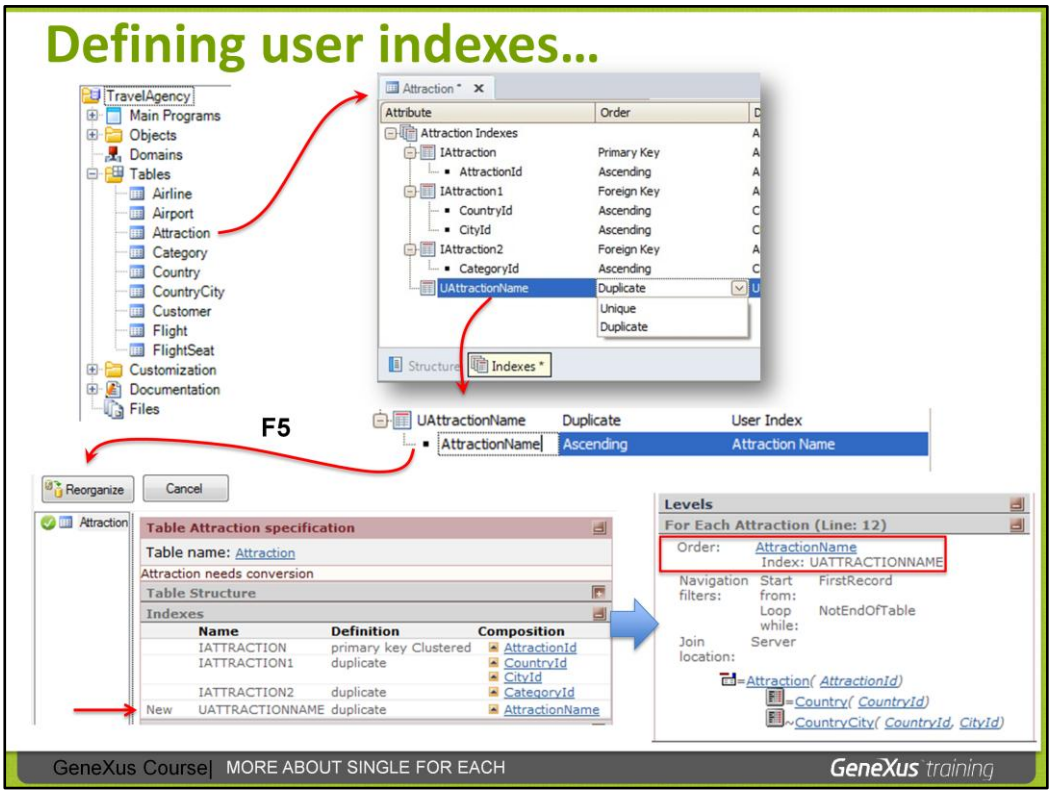
Remember that **indexes** are efficient access paths to data. They act as dictionaries that index by a certain attribute or set of attributes. In this case it is done one by one: AttractionName.

The drawback of creating an index is that once it is created it must be maintained. That is to say, if an attraction is added, such as the obelisk of Sao Paulo, the index must be rearranged (as we can see above, by comparing the index to the left with that of the previous page).

Creating an index from GeneXus for a database table is easy and it can be done at any time.

Database management systems usually save data and access statistics, which give them intelligence to choose the best access strategy based on the attributes included in the query. For a better understanding of this topic, we recommend reading our documentation according to the generator that you're using.

Here we only need to know that if we indicate an order and no index has been created, either a temporary one will be created, used for the query and then discarded, or the DBMS will solve it with another strategy.



How? We look for the table, open it and go to the section that informs us about the indexes defined.

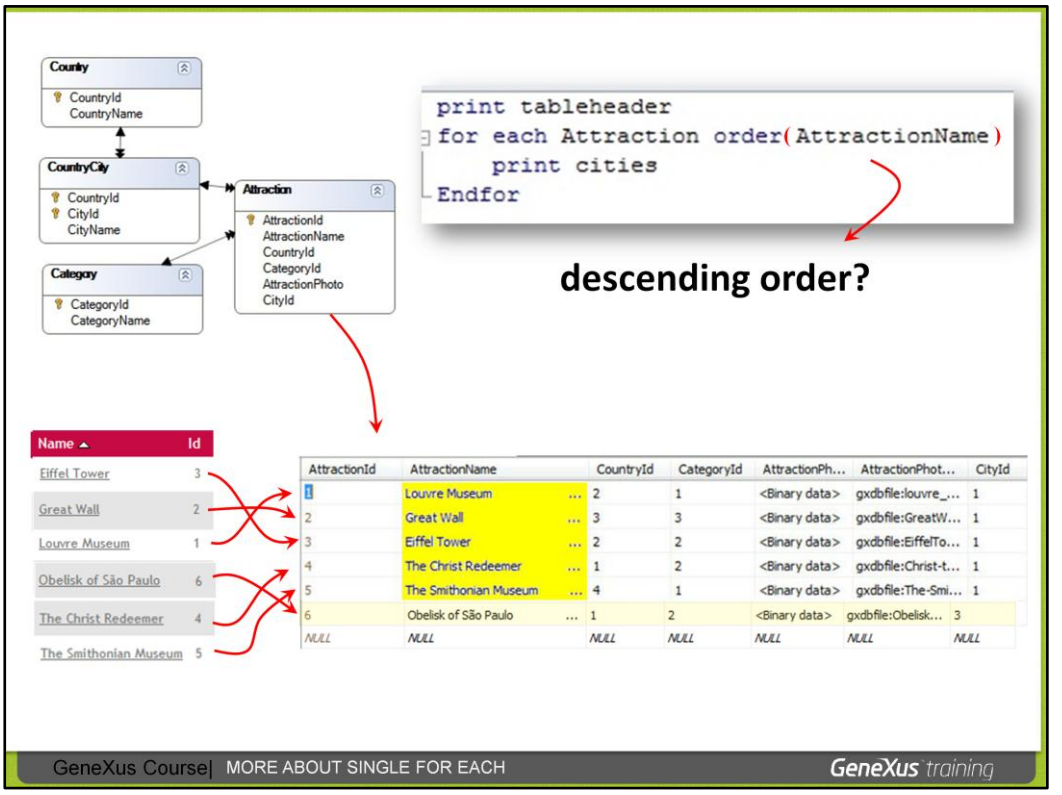
The first three ones, which are preceded by the "I" prefix, are those automatically created by GeneXus based on the primary and foreign keys in order to make referential integrity controls more efficient.

We need to create a user index. To do so, we press Enter, which will display the UAttraction default name. We change it as we want –by adding Name at the end, for instance. Note the "U" prefix which makes reference to **U**ser.

Our index will be made up by the AttractionName attribute, ordered in ascending order. If attraction names can't be repeated, we can control this by setting the index as **Unique** instead of **Duplicate.** In this case, a control will be automatically performed when entering an attraction to confirm that there isn't another attraction with the same name –using this index. In our case, they can be repeated (for instance, consider that countries usually have Obelisks), so we select the Duplicate value.

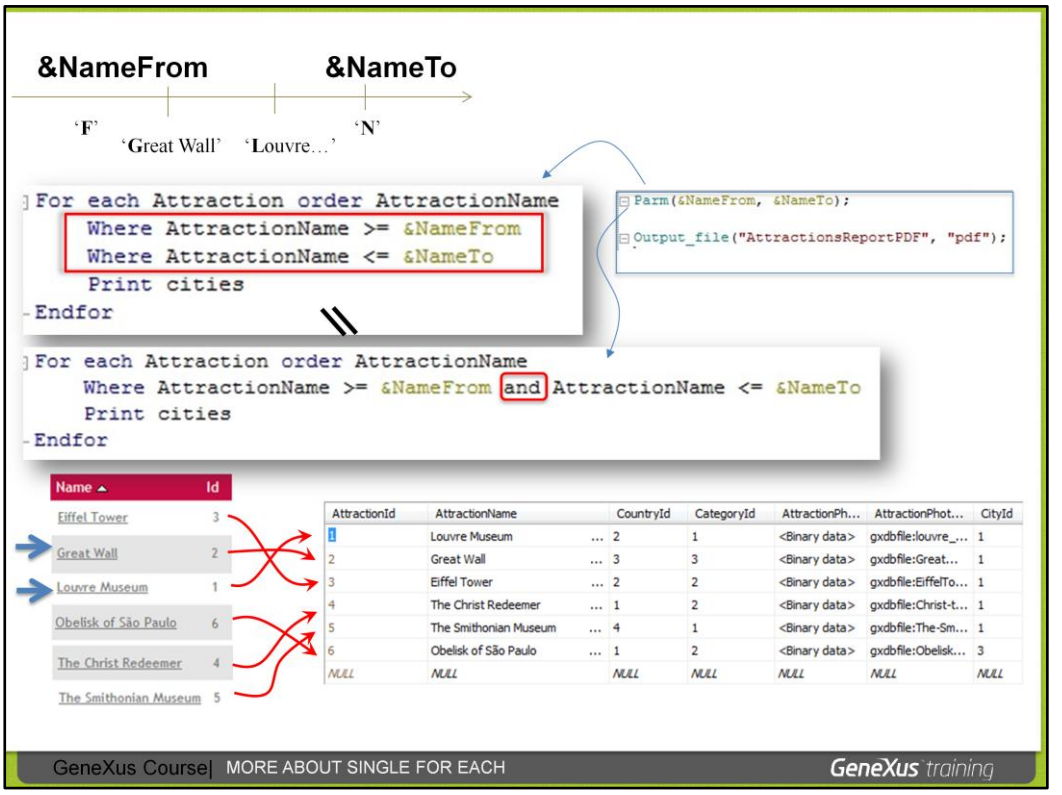
Once we do this, pressing F5 will cause the database to be reorganized to create this new index. Remember that the navigation report informed us that we didn't have an index to make this query. Note what it reads after the reorganization has been completed: it tells us that the index we've just created will be used.

We can create and delete this index at any time, and upon pressing F5 and reorganizing, we go back to the same situation we had before creating it.



How do we configure it to use descending order? Simply by placing the attribute between brackets.

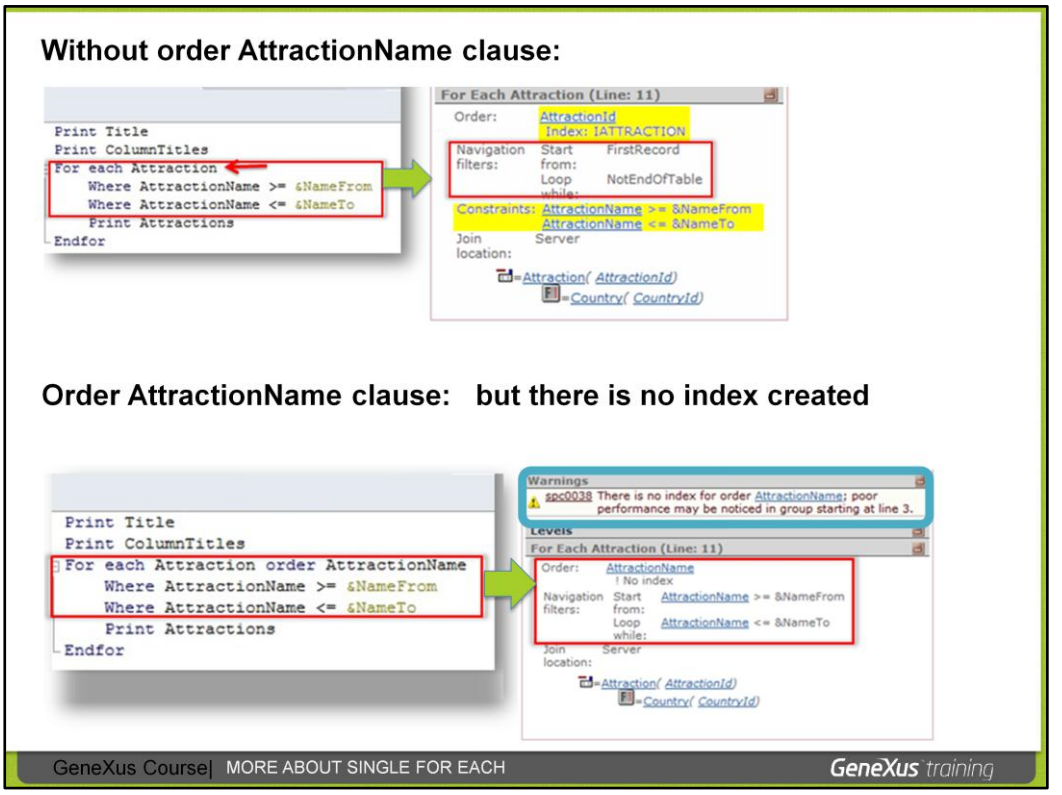
ORDERS
COMPATIBLES WITH
FILTERS



Now let's suppose that we're interested in obtaining a list of attractions whose names are shown in alphabetical order between two values received in a parameter. For instance, between "F" and "N".

To do so, we enter the Where clauses indicated above.

Several Where clauses are equivalent to only one, where conditions are combined with the "and" logical operator. That is to say, to be considered, records must meet all conditions **at once**.

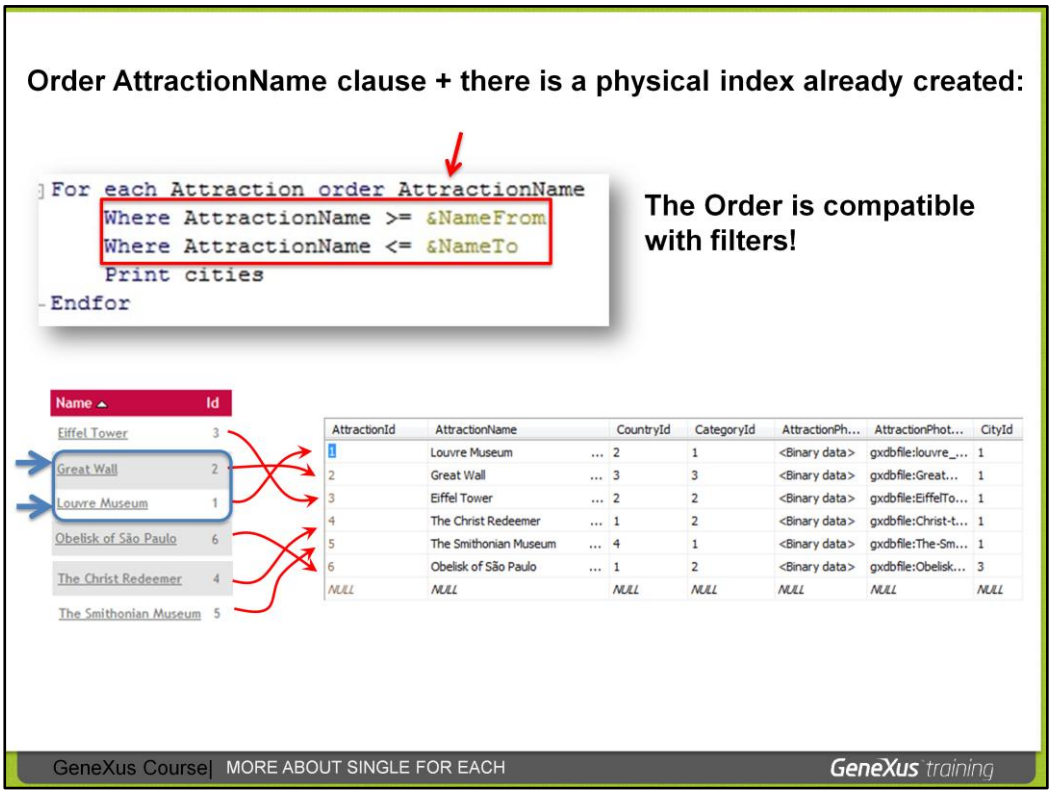


Note what happens when we remove the **order** clause. The data displayed will be the same, but since no ordering criteria has been indicated, it has selected the primary key order. This means that it will have to run through the entire table to keep the records that meet all restrictions. The first record might fall within the range and the last one too. This listing is not optimized!

Instead, if we specify the **order AttractionName** clause, there are 2 possibilities:

1) The physical index does not exist.
2) The physical index is defined as a user index.

We have already explained the first case. Let's see the second case.



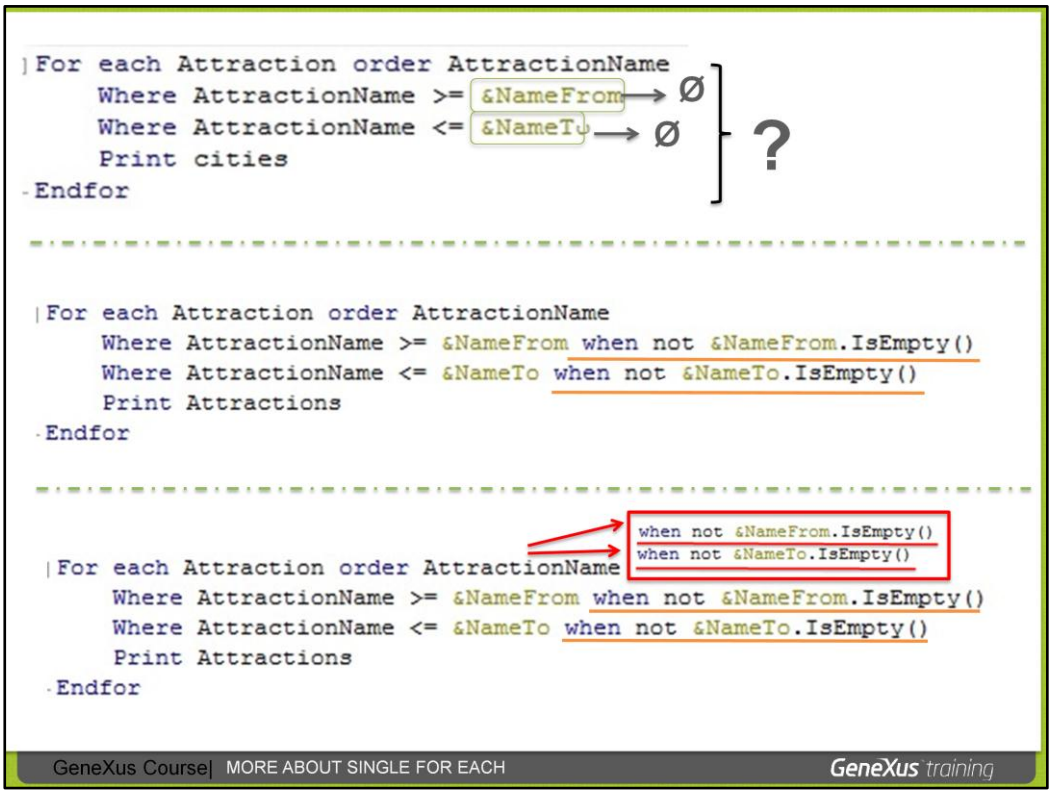
If the query is sorted by AttractionName and there is a physical index created for that attribute, GeneXus will use it. In this case the navigation list will inform us that it will not be necessary to navigate all the base table.

Imagine that we are looking for a word that starts with "G" on a paper dictionary. We will not search in the dictionary from the beginning to the end because it is sorted!! Here is the same.

In order to determine the base table, it is the same if the **order AttractionName** clause is present or not, because the AttractionName attribute is already present in the **Where clause**, and the base transaction is specified next to the For each.

If possible, it is suggested to sort by compatible criteria with filters, so the query will be optimized.
If this is not possible, anyway the DBMS will efficiently solve the query by creating temporal indexes.

WHEN CLAUSE FOR
ORDERS AND FILTERS



What result will be obtained for the above For Each command if the &start and &end variables are empty? If there is an attraction with an empty name, it will be the only one returned because it will be the only one that meets both conditions. Otherwise, there won't be any attractions listed.

Is it possible to add conditions to orders and filters so that they are applied only under certain circumstances? For example, to only apply the first Where **when** the &start variable is not empty. And to apply second Where **when** the &end variable is not empty. The answer is yes. We can do it by adding conditions to Where clauses with **when**, as we can see in the second For Each command. Where clauses will only be applied when the condition is met. Thus, at runtime, when both variables are left empty, none of the Where clauses will be applied and all the attractions in the table will be listed.

In the same way, we can add a condition to an order to have it applied or not, as we showed in the third For Each command. In fact, a series of conditioned orders can be indicated, in order to choose the first one whose condition is met.

Read more about orders and filters in the GeneXus wiki (http://wiki.gxtechnical.com/commwiki/servlet/hwikibypageid?6075).

WHEN NONE

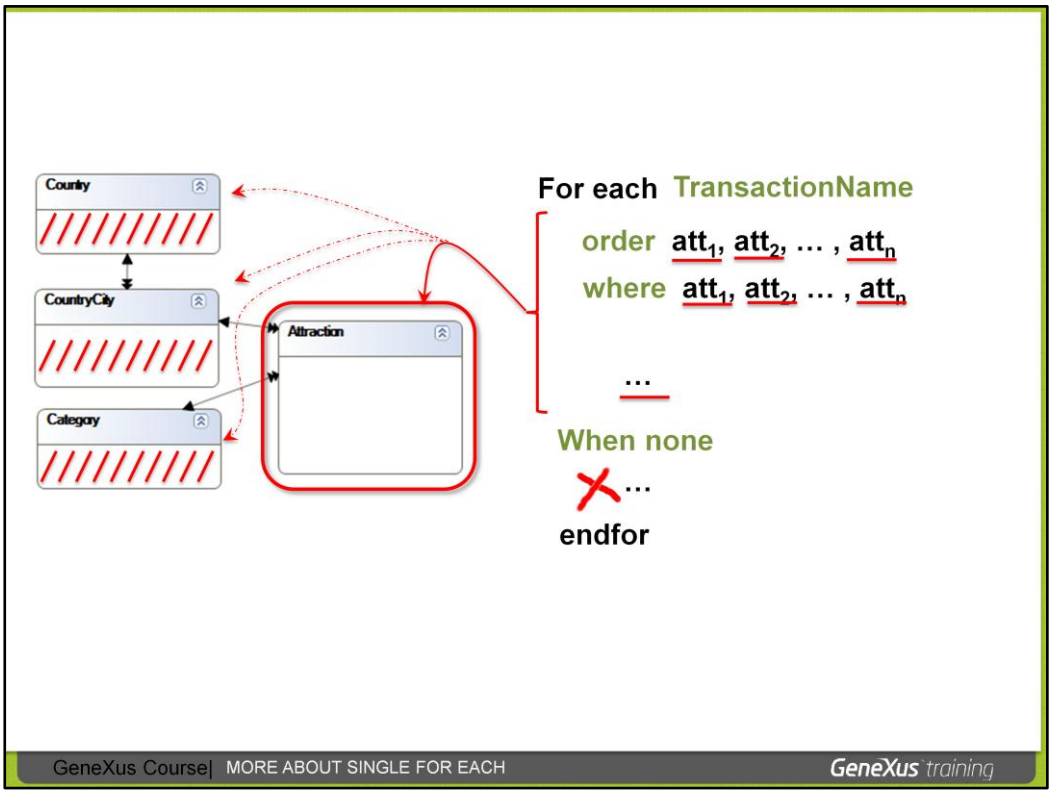
What happens when none of the records in the base table meet the conditions?

Suppose that in this case we want to print a message in the output to warn about this… to do so we program the **when none** clause that closes the For Each command.

All the commands written between when none and endfor will be executed in sequence, only in cases when no records of the For each **base table** that meet the conditions can be found.

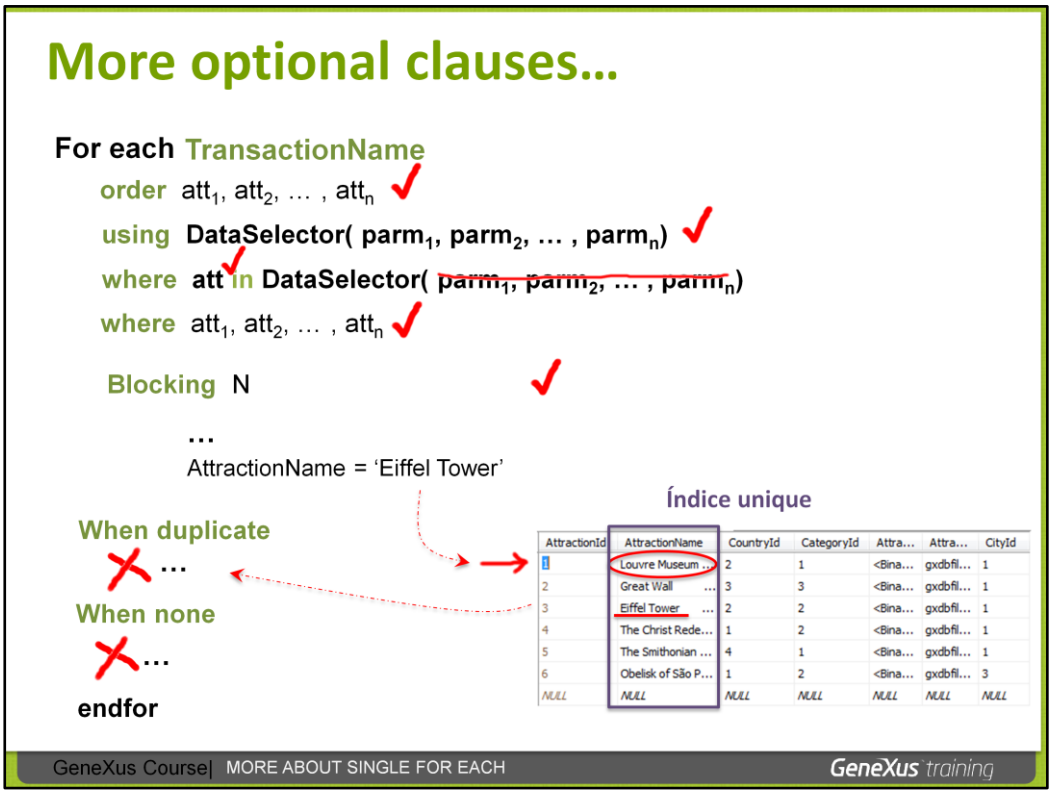
The execution of what comes after **when none** will mean that the search hasn't found any results; therefore, any attributes included **will not be used** to determine that table of the For Each command –unlike the others.

SUMMARY



As we already know, GeneXus determines the base table of the For each by the transaction name that is mentioned nexto to it. In addition, all the other attributes (present for example in Order and Where clauses) must belong to the extended table.

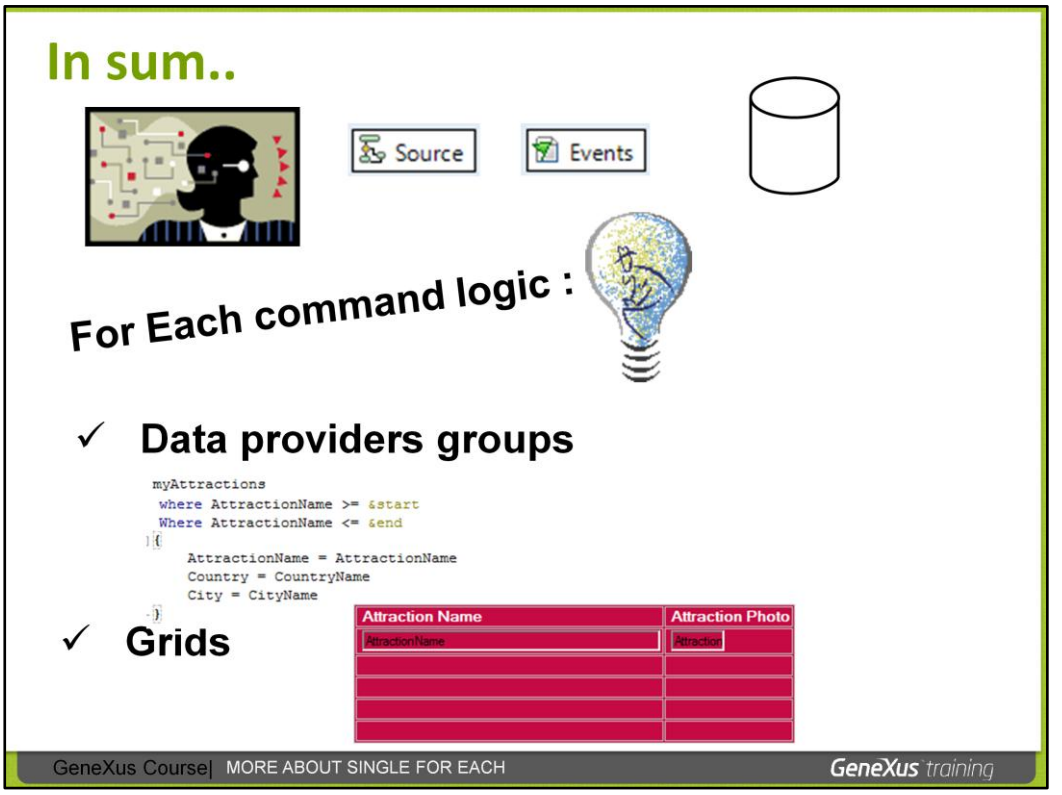
The attributes that are not considered are those mentioned within the When none clause.



The For Each command accepts more optional clauses. For example, there is another way to filter the data we want to work with, such as **Data Selectors**. This topic will be examined later. Read more about this object in our wiki: http://wiki.gxtechnical.com/commwiki/servlet/hwikibypageid?5271.

Another optional clause that is added when the For Each command is being run within a procedure, to update or delete database records, is the **Blocking** clause which allows indicating that the update or deletion should be made in blocks of N records thus reducing the number of accesses to the database.

The For Each command is included in a procedure, and we want to update the value of an attribute that can't have repeated values. For example, suppose that AttractionName has a Unique index and the For Each command is run on record 1… When trying to change the value of AttractionName of that record, so that it takes the "Eiffel Tower" value, the Unique index will indicate that there is a record with that value and the update will not take place. But if we want to perform an action in this case, we program the **when duplicate** clause.



For Each commands can be used in procedures and events of other objects in which database queries are allowed.

The same For Each logic is present in other ways to query the database, such as:

•Data Provider groups, and
•Grids with base table

Therefore, mastering this logic means that we have learned the fundamentals behind GeneXus.