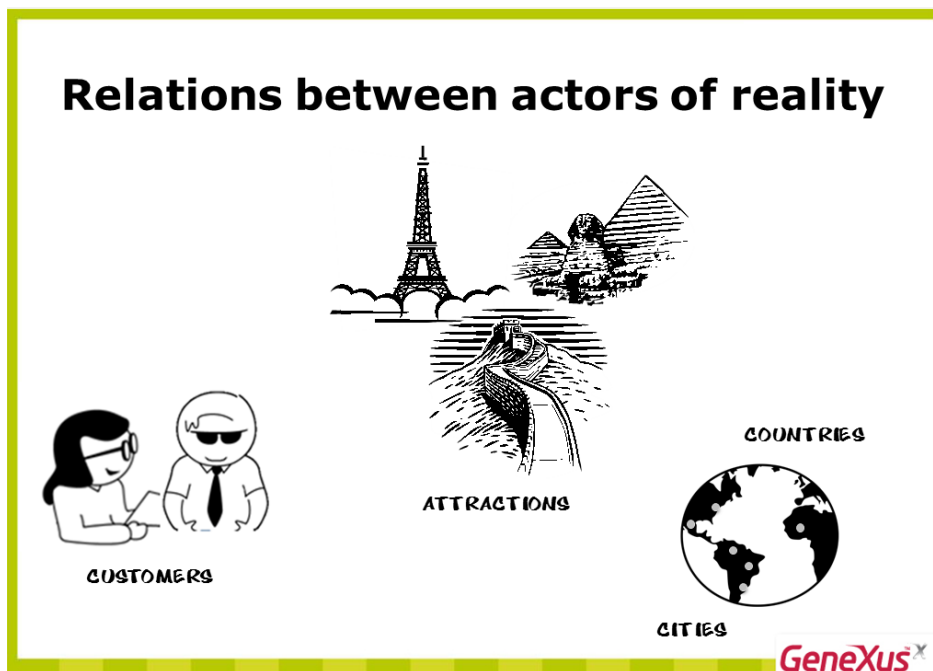
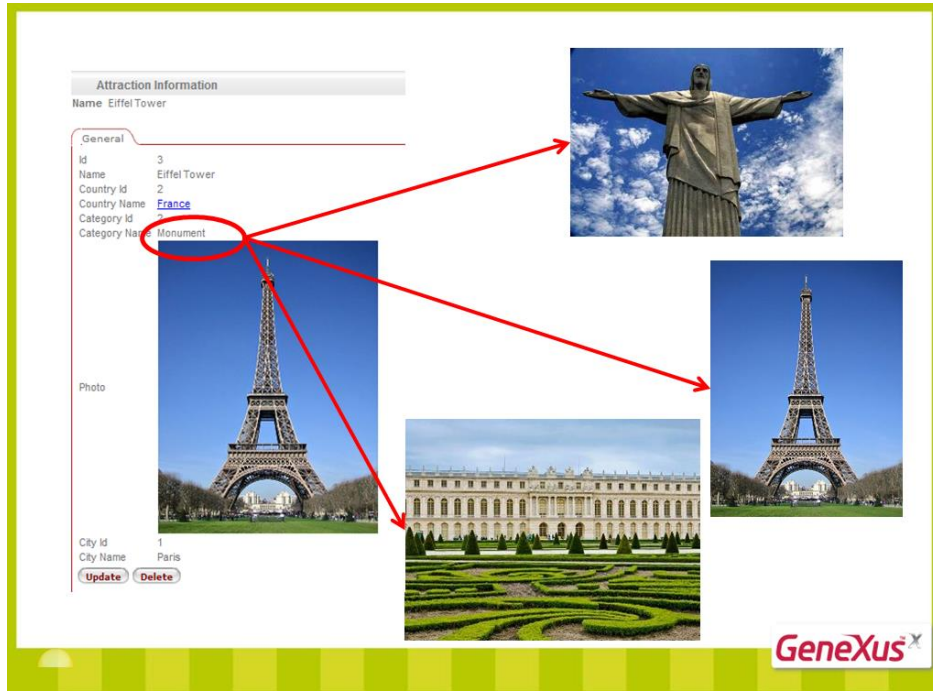


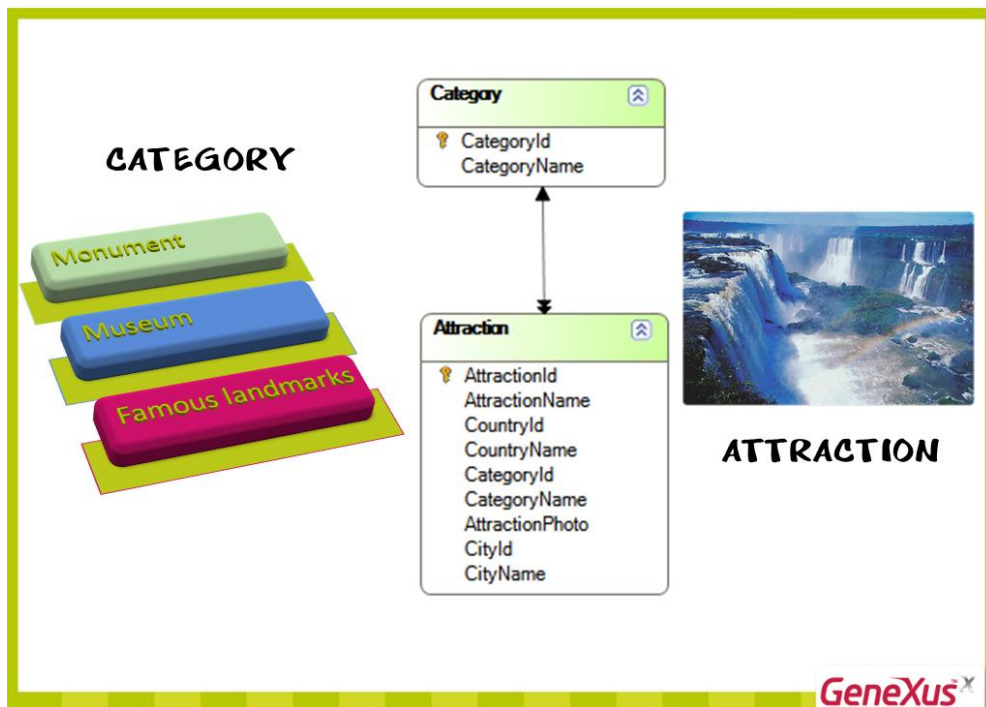
GeneXus Course– Relations between actors from reality



In several examples at our travel agency we found that actors from reality relate to one another in different ways. An example is an attraction that belongs to a category which in turn may be the category of many attractions.



We saw that we can represent these relations, as we design transactions, by including a transaction's attributes in another transaction.

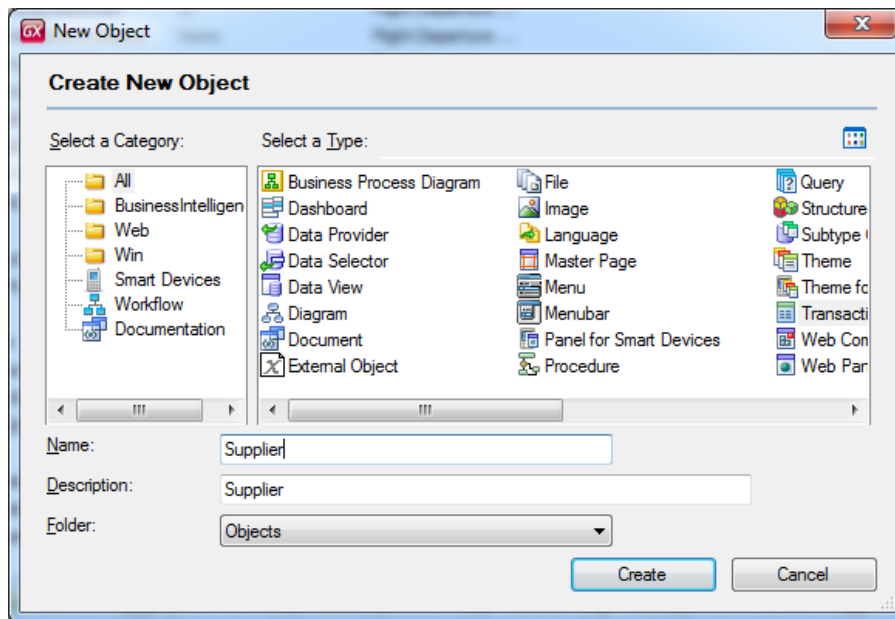


We have been told that the agency now works with providers who periodically offer the agency visits to tourist attractions in different parts of the world.

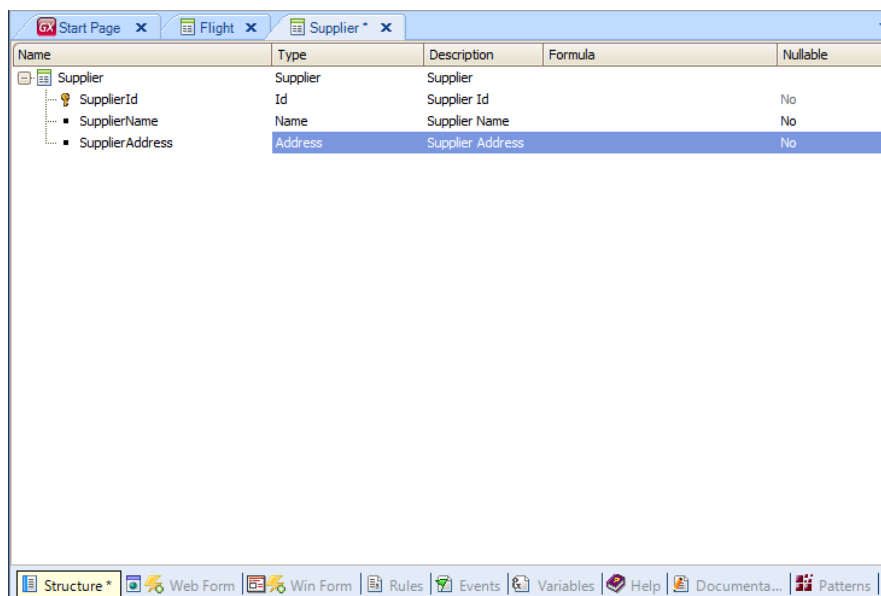


Each provider offers numerous tourist attractions, but each attraction is managed by a single provider. To represent this reality we will create the Supplier transaction, where we will register providers...

We do File...New...Object...name it Supplier.... And then add the attributes:



SupplierId as identifier, SupplierName to save the name of the provider, and SupplierAddress to save its address.



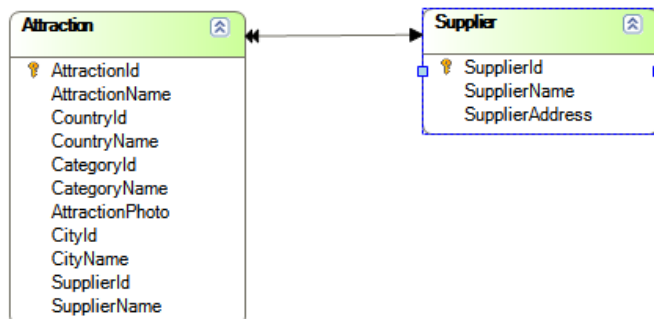
Through the transactions diagram object we can analyze the relation between suppliers and attractions. We do ...New..Object of the Diagram type, and drag the Attraction and Supplier transactions from the View Folder. We will see that we have not established any relation between those two actors yet.



Because a tourist attraction has an only supplier offering it, we will include the supplier identifier in the structure of the Attraction transaction, so we open that transaction and add the SupplierId attribute. We also add the Supplier Name attribute because we will then be able to show the name of the supplier in the screen corresponding to the attractions.

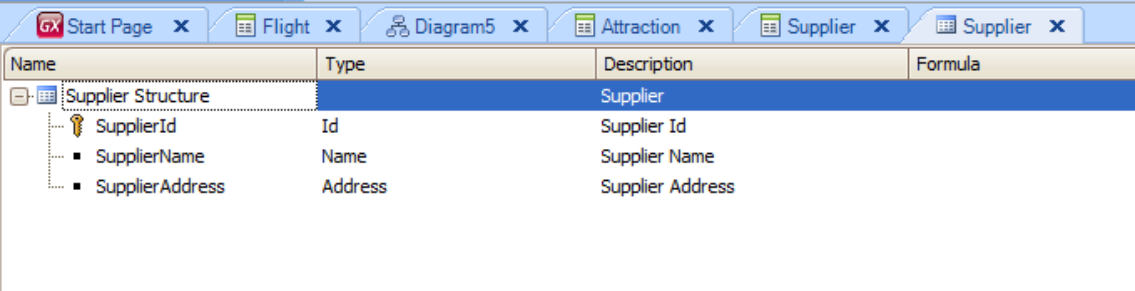
Name	Type	Description	Formula	Nullable
Attraction	Attraction	Attraction		
AttractionId	Id	Attraction Id		No
AttractionName	Name	Attraction Name		No
CountryId	Id	Country Id		No
CountryName	Name	Country Name		
CategoryId	Id	Category Id		Yes
CategoryName	Name	Category Name		
AttractionPhoto	Image	Attraction Photo		No
CityId	Id	City Id		Yes
CityName	Name	City Name		
SupplierId	Id	Supplier Id		No
SupplierName	Name	Supplier Name		

We now create a new diagram, drag both transactions again... and we will see that there is plain arrow pointing at Supplier, and a double arrow pointing at Attraction. And that tells us that an attraction will have only one supplier and a supplier may offer numerous attractions.



In sum, if we add a transaction’s identifier attribute to another transaction (which, as we saw, will play a role as foreign key here) a relation of 1 to many (also known as “1 to N”) will be established, where the “many” side of the relation is where the foreign key is located.

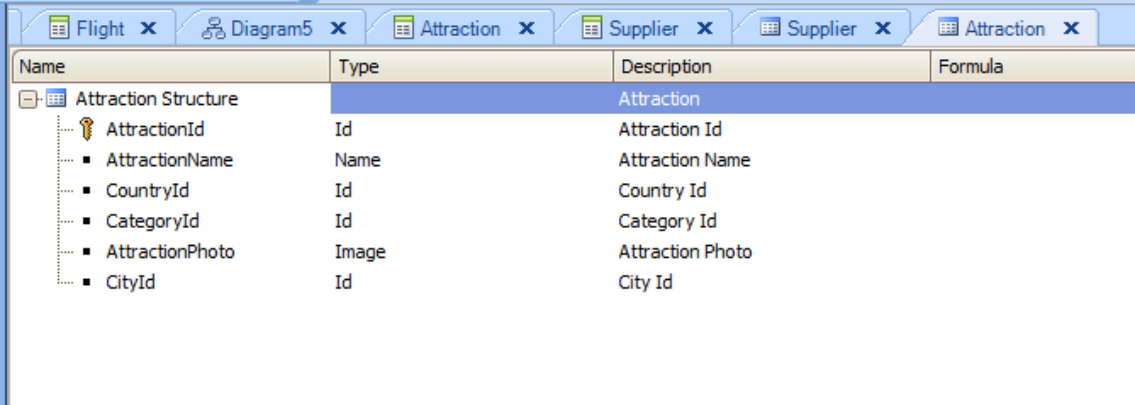
If we now analyze which tables GeneXus will generate on the basis of that transaction design, we will see that, from the Supplier transaction, it will create a SUPPLIER table with the same structure as the transaction:



The screenshot shows the GeneXus interface with the 'Supplier' transaction selected. The 'Supplier Structure' is displayed in a tree view on the left, and a table view on the right shows the attributes and their types.

Name	Type	Description	Formula
Supplier Structure		Supplier	
SupplierId	Id	Supplier Id	
SupplierName	Name	Supplier Name	
SupplierAddress	Address	Supplier Address	

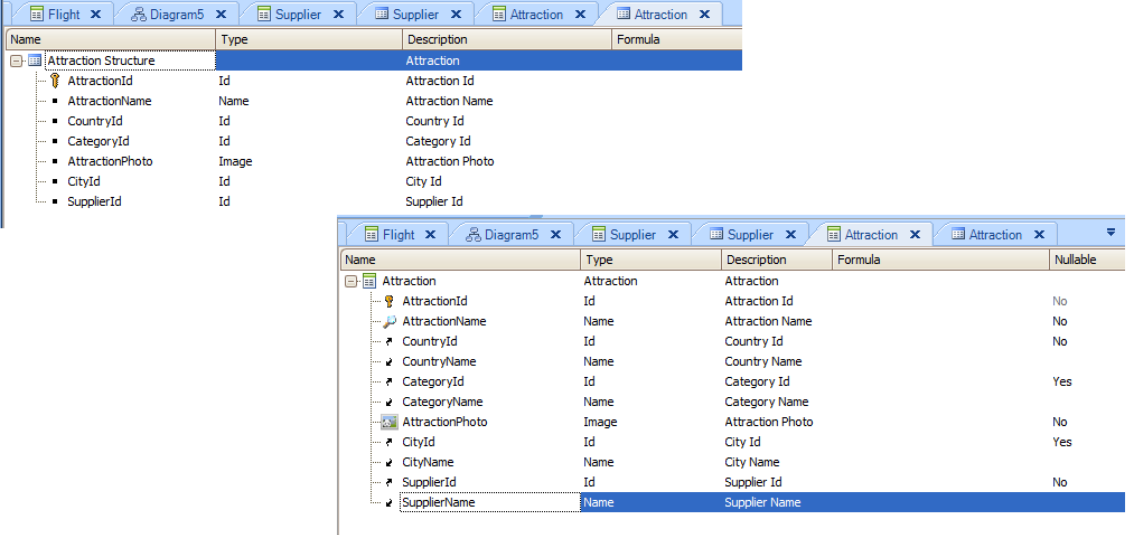
And from the structure of the Attraction transaction GeneXus will create an ATTRACTION table with the following structure:



The screenshot shows the GeneXus interface with the 'Attraction' transaction selected. The 'Attraction Structure' is displayed in a tree view on the left, and a table view on the right shows the attributes and their types.

Name	Type	Description	Formula
Attraction Structure		Attraction	
AttractionId	Id	Attraction Id	
AttractionName	Name	Attraction Name	
CountryId	Id	Country Id	
CategoryId	Id	Category Id	
AttractionPhoto	Image	Attraction Photo	
CityId	Id	City Id	

If we compare the structure of the ATTRACTION table against that of the Attraction transaction,



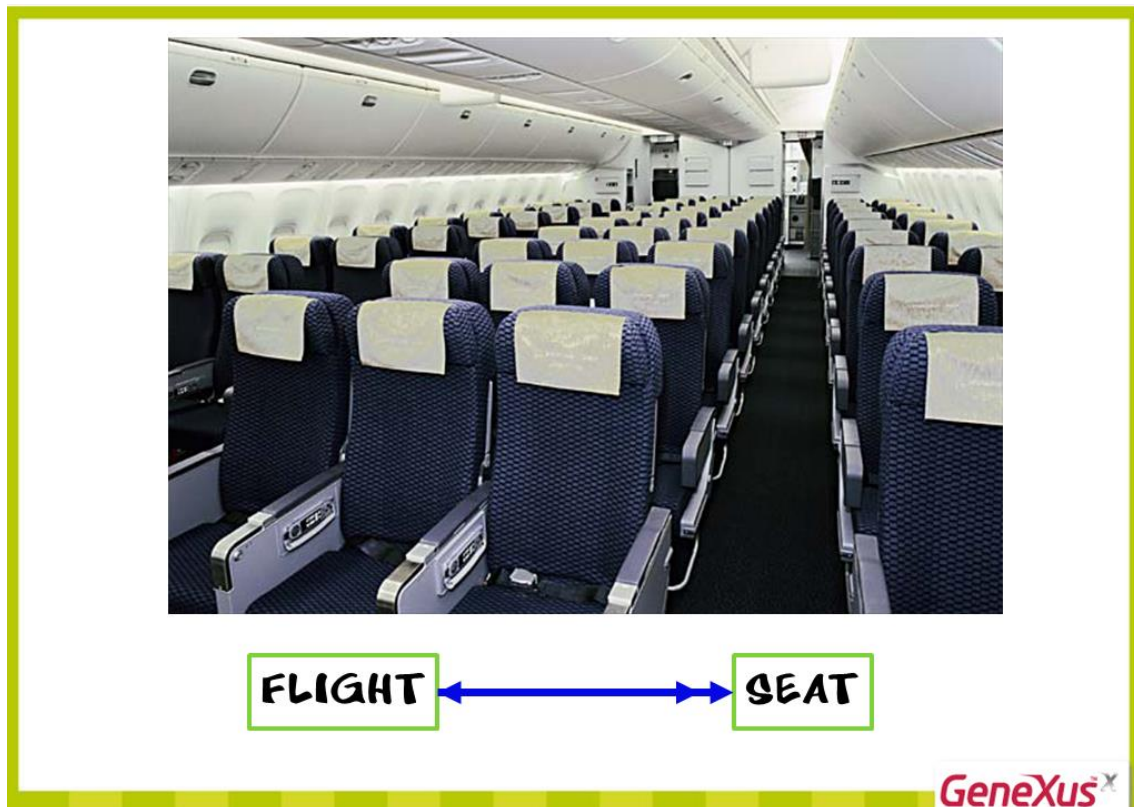
The screenshot shows the GeneXus interface with the 'Attraction' transaction selected. The 'Attraction Structure' is displayed in a tree view on the left, and a table view on the right shows the attributes and their types. The table view includes a 'Nullable' column.

Name	Type	Description	Formula	Nullable
Attraction Structure		Attraction		
AttractionId	Id	Attraction Id		No
AttractionName	Name	Attraction Name		No
CountryId	Id	Country Id		No
CountryName	Name	Country Name		
CategoryId	Id	Category Id		Yes
CategoryName	Name	Category Name		
AttractionPhoto	Image	Attraction Photo		No
CityId	Id	City Id		Yes
CityName	Name	City Name		
SupplierId	Id	Supplier Id		No
SupplierName	Name	Supplier Name		

we will see that the CountryName, CategoryName, CityName and SupplierName attributes are not included in the table because they are attributes inferred, which, as we saw, because they are in the extended table of the ATTRACTION table, their value may be recovered from the table where they are stored physically.

This is the most usual way of representing the relation 1 to many between two actors from reality, that is: between two entities in our system.

However, there are other cases of 1 to many relations where we will use another type of representation.

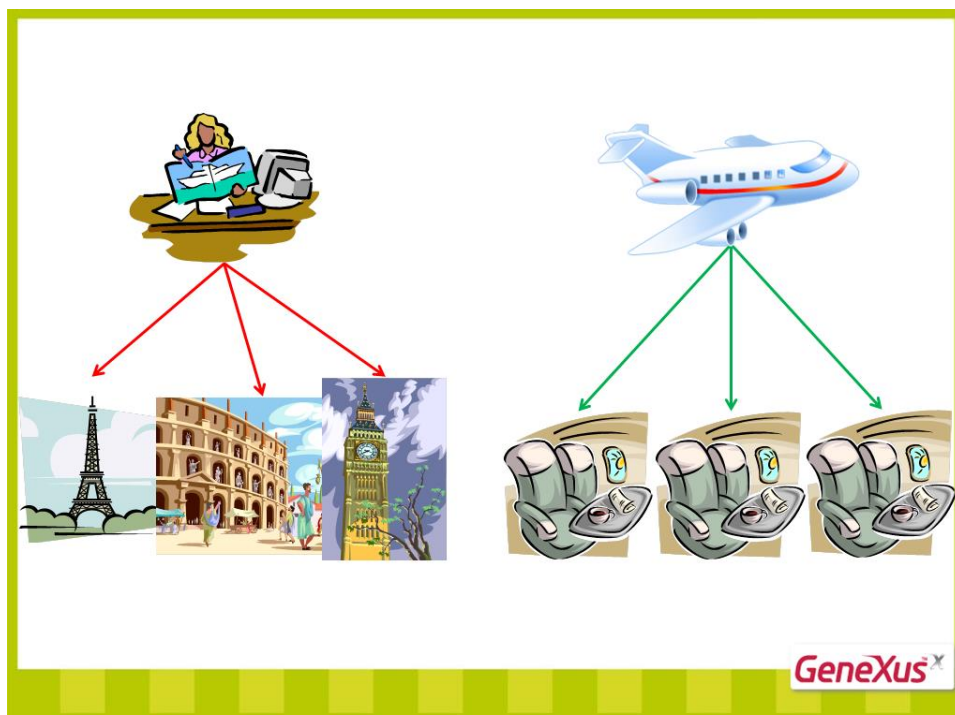


Let's recall the case of flights, where a flight has numerous seats and each seat is assigned to a single flight, that is: in a relation 1 to many. Let's open the structure of the Flight transaction to see how we represent that relation....

Name	Type	Description	Formula	Nullable
Flight	Flight	Flight		
FlightId	Id	Flight Id		No
FlightDepartureCountryId	Id	Flight Departure ...		No
FlightDepartureCountryName	Name	Flight Departure ...		
FlightDepartureCityId	Id	Flight Departure ...		No
FlightDepartureCityName	Name	Flight Departure ...		
FlightArrivalCountryId	Id	Flight Arrival Cou...		No
FlightArrivalCountryName	Name	Flight Arrival Cou...		
FlightArrivalCityId	Id	Flight Arrival City...		No
FlightArrivalCityName	Name	Flight Arrival City...		
FlightPrice	Price	Flight Price		No
FlightDiscountPercentage	Percentage	Flight Discount P...		No
FlightFinalPrice	Price	Flight Final Price	$\text{FlightPrice} * (1 - \text{AirlineDiscountPercent} \dots)$	
AirlineId	Id	Airline Id		Yes
AirlineName	Name	Airline Name		
AirlineDiscountPercentage	Percentage	Airline Discount P...		
FlightCapacity	Numeric(4.0)	Flight Capacity	$\text{count}(\text{FlightSeatLocation}, \text{FlightSeat} \dots)$	
FlightSeatQty	Numeric(4.0)	Flight Seat Qty	$\text{count}(\text{FlightSeatChar})$	
Seat	Seat	Seat		
FlightSeatId	Id	Flight Seat Id		No
FlightSeatLocation	Location	Flight Seat Location		No
FlightSeatChar	SeatChar	Flight Seat Char		No

We see in this case that Seat is like a second level in the Flight transaction.

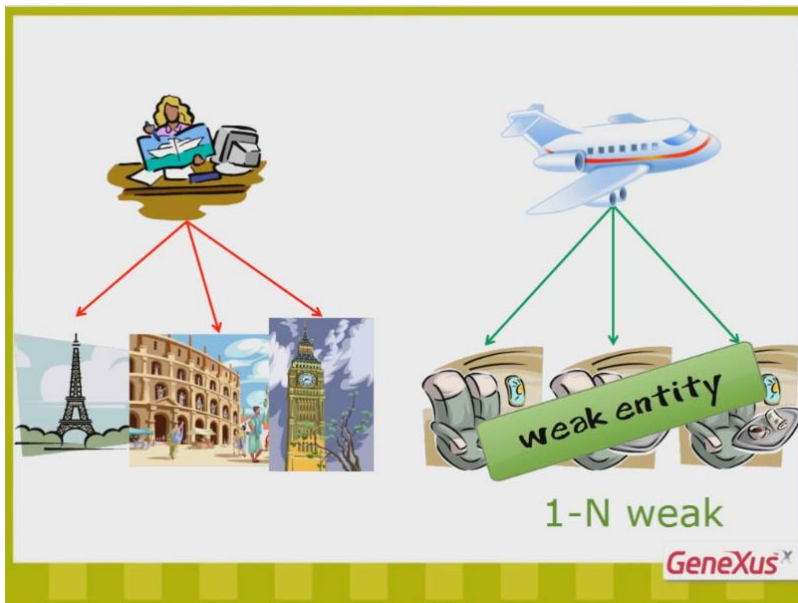
So, what is the difference between this 1 to many relation and the relation of 1 to many we saw between Attractions and Providers?



Why don't we represent both cases the same way (with the same transaction design)?

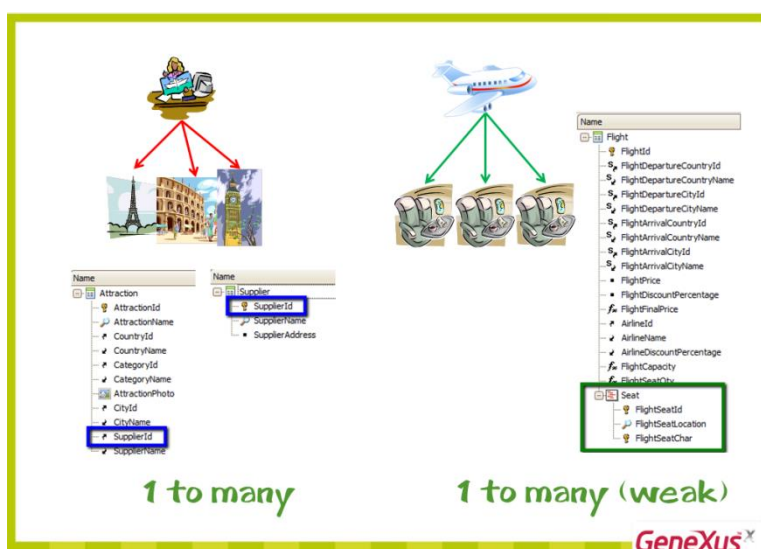
Note that the existence of seats would have no sense if they were not in a flight. It is senseless to consider a seat without it **always** relating to the flight it belongs ... However, an attraction could not have a provider offering it and it could still exist on its own as such...

The other difference is that when we enter the data of a flight, we are also entering the data relative to the flight's seats (just as when we enter an invoice and the lines in it, we will be entering all the information at once). However, the data relative to Suppliers and Attractions do not necessarily have to be entered at the same time.



An entity like seats, whose existence only makes sense when represented in relation to another entity (in this case flights) is called a **weak entity**.

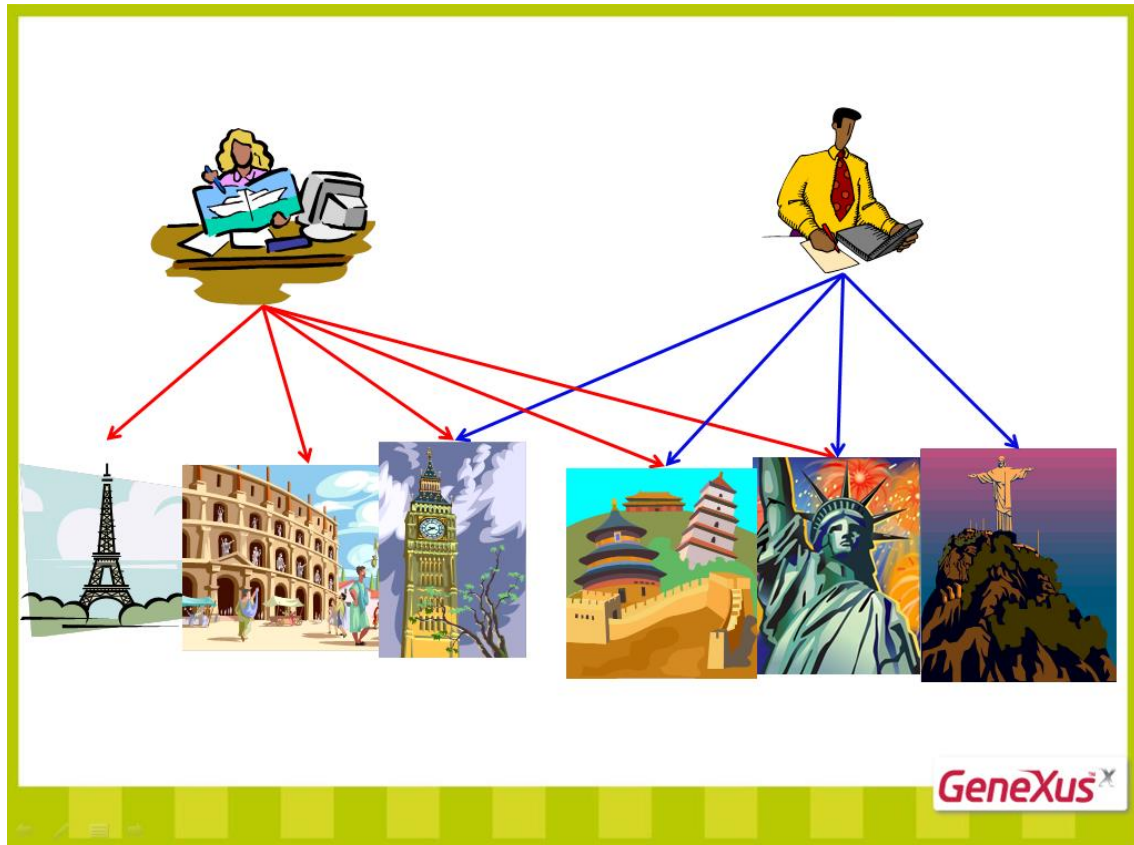
We represent this type of **weak 1 to N relation** with a single two-level transaction, where the weak entity is in the second level. As opposed to the 1 to N relation between Suppliers and Attractions, where we created **2** transactions and in one we set the other's primary key as the foreign key.



So far we saw relations of 1 to many, but this is not always the case we must represent from reality.

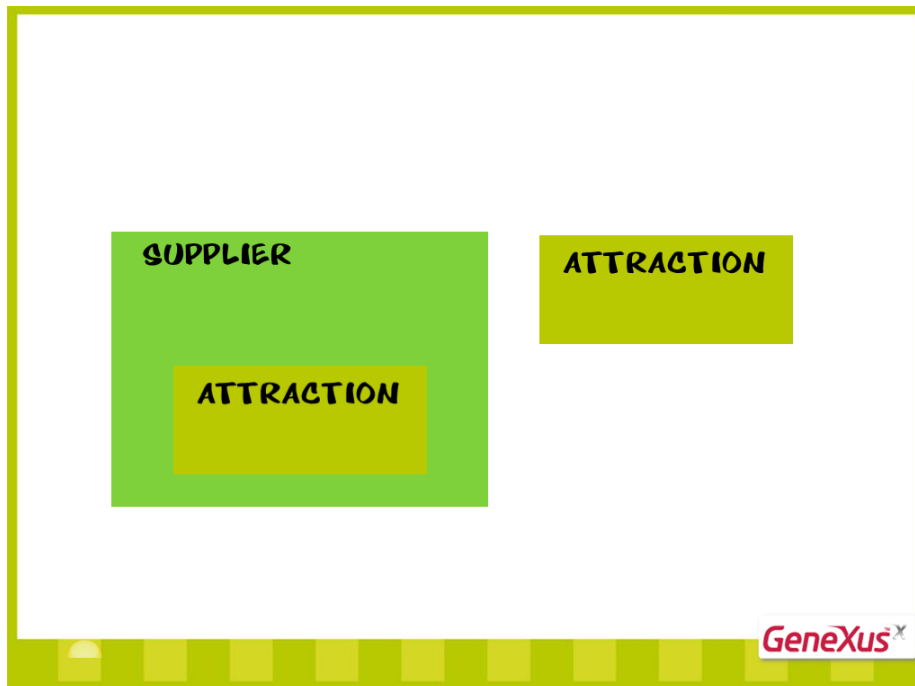
Let's suppose that the **reality of the travel agency has changed:**

Each supplier offers several tourist attractions (as we have seen so far), but *each attraction may be administered by DIFFERENT suppliers (instead of just 1 as we have had).*



So, the relation between Suppliers and Attractions is no longer a “1 to many” relation but rather a “**many to many**” relation.

And how should we represent this in GeneXus?



The solution is to use two transactions, one for each entity. Also, to one of them we add the other as a second level. This is done upon considering how the data will be entered, either all the tourist attractions for each supplier, or for each attraction all the suppliers that provide it. In this case, the most logical thing to do is to enter the attractions provided by each supplier as we enter the supplier.

We will do this in GeneXus...

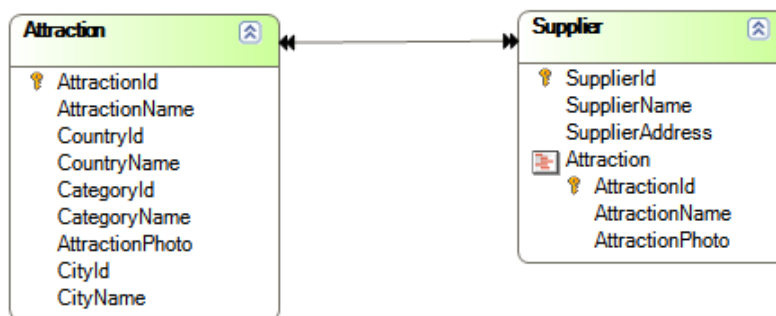
We open the Attraction transaction and remove the SupplierId and SupplierName attributes and save.

Name	Type	Description	Formula
Attraction	Attraction	Attraction	
AttractionId	Id	Attraction Id	
AttractionName	Name	Attraction Name	
CountryId	Id	Country Id	
CountryName	Name	Country Name	
CategoryId	Id	Category Id	
CategoryName	Name	Category Name	
AttractionPhoto	Image	Attraction Photo	
CityId	Id	City Id	
CityName	Name	City Name	

We now open the Supplier transaction, where we add a second level and add the attributes: AttractionId, Attraction Name and AttractionPhoto. We will see that the second level remained with the name Attraction.

Name	Type	Description	Formula
Supplier	Supplier	Supplier	
SupplierId	Id	Supplier Id	
SupplierName	Name	Supplier Name	
SupplierAddress	Address	Supplier Address	
Attraction	Attraction	Attraction	
AttractionId	Id	Attraction Id	
AttractionName	Name	Attraction Name	
AttractionPhoto	Image	Attraction Photo	

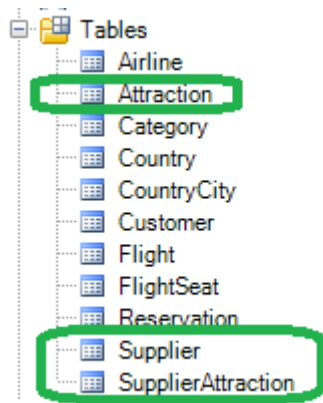
We will now see how this relation turned out by creating a new diagram...File ...New...Diagram... and drag the Attraction and Supplier transactions to the diagram.



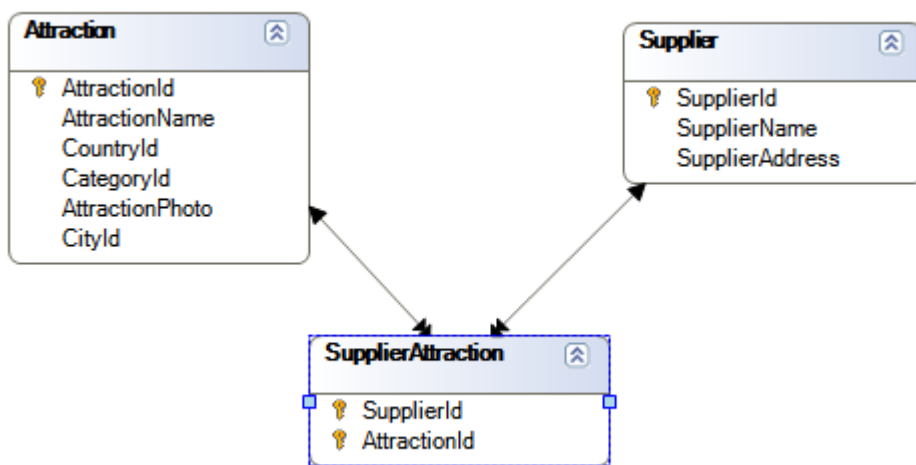
Now there is a double arrow at each end of the relation, indicating that the relation is a “many” to “many” relation, meaning that one attraction is provided by several suppliers and one supplier provides many attractions.

Let’s now see the tables that GeneXus will create base don the above design...

We see that there is an ATTRACTION table, a SUPPLIER table and a table called SUPPLIERATTRACTION.



We now create a new diagram object and drag the three tables to the diagram....

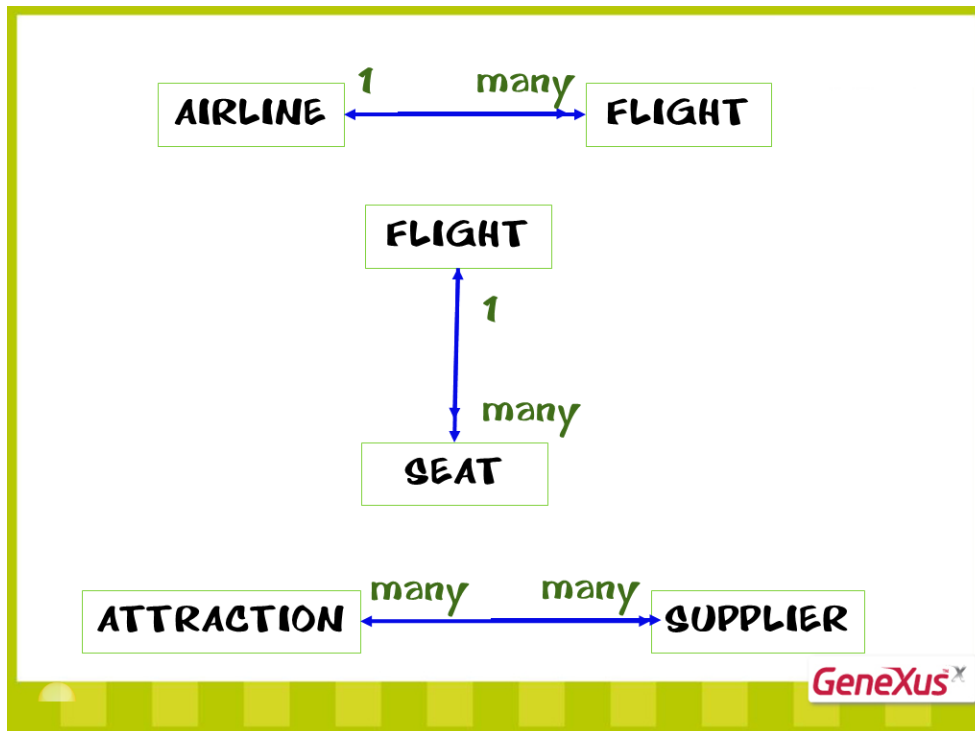


We should note that in this case, GeneXus creates a table for each transaction that is part of the relation of many to many (ATTRACTION and SUPPLIER), and it also creates a third table called SUPPLIERATTRACTION, to establish the relation.

If we analyze the structure of this third table, we will see that only the attributes identifying the other two tables are included.

So, every time that GeneXus establishes a relation of many to many, that relation will be represented in the database by three tables, one for each entity that is included, and a third one con the identifiers of the other two.

We can see that the many to many relation between Attraction and Supplier broke down into two relations of one to many, using the SUPPLIERATTRACTION table to establish the relation between the previous ones.



So, we have seen that we can represent different relations between actors from our reality with transactions and their attributes.