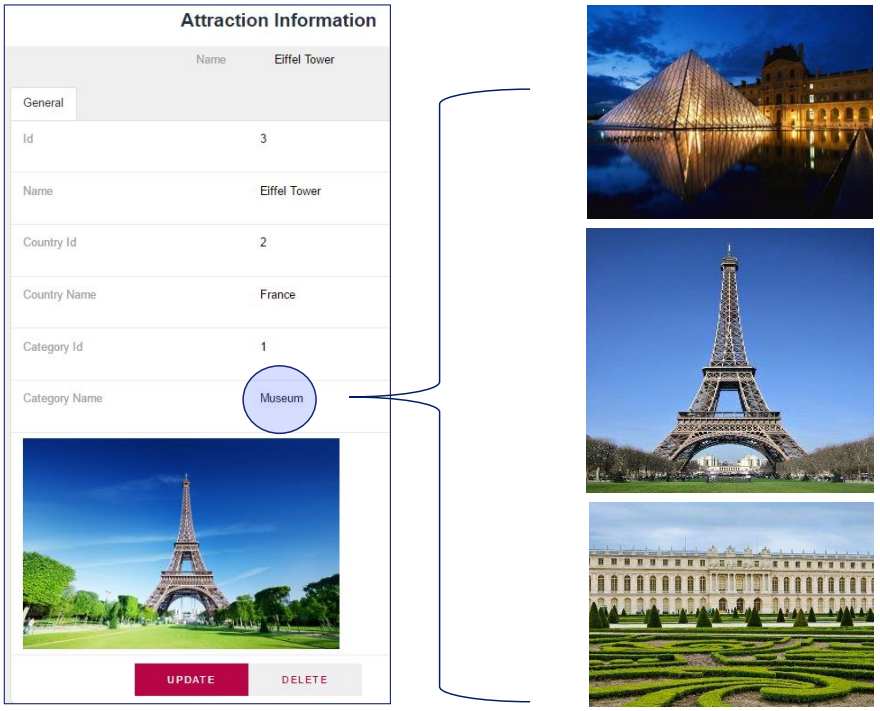
Relations between entities of reality

GeneXus™



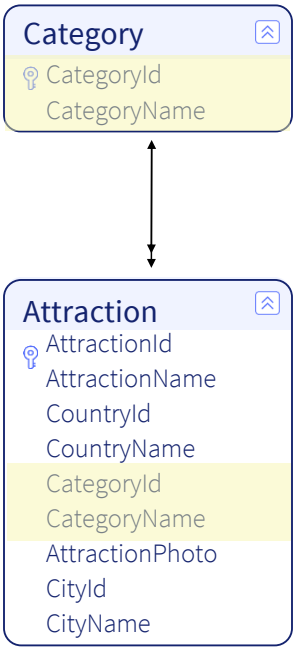
In several examples of our travel agency, we see that the actors of reality are related in different ways; for example, when an attraction belongs to a category and, in turn, this category can be the category of many attractions.

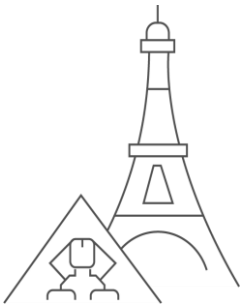
CATEGORY

Monument

Museum

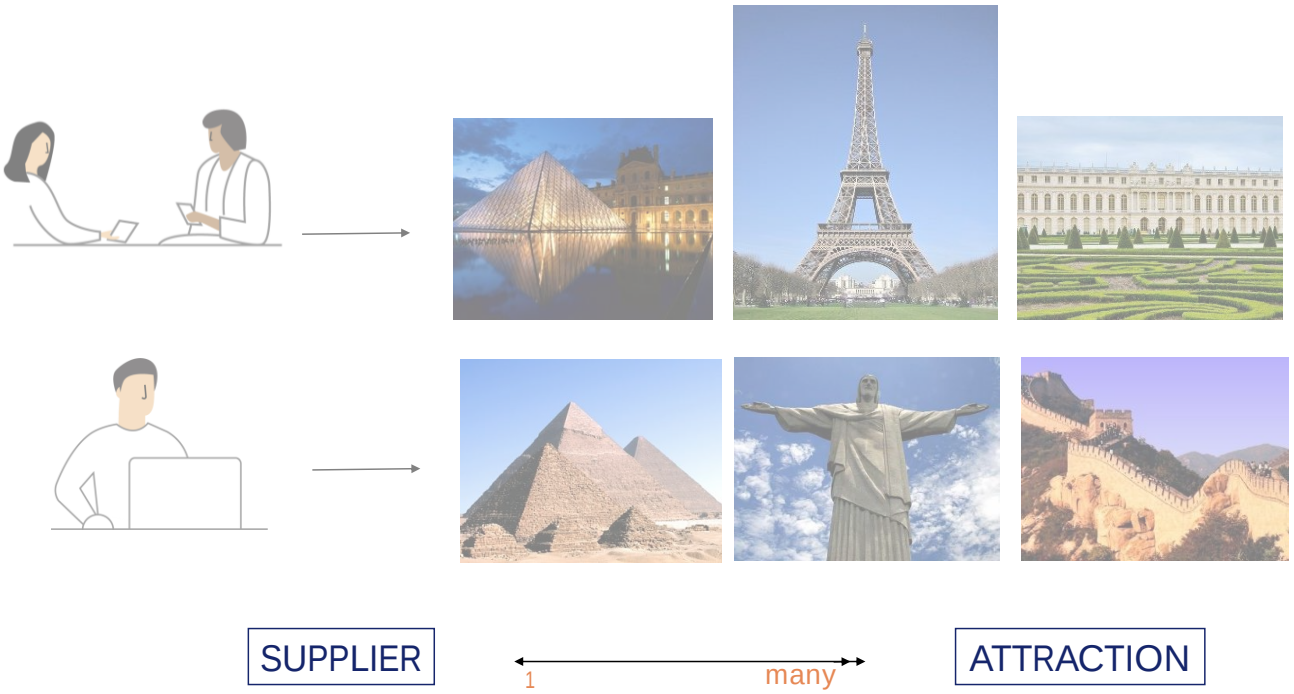
Famous Landmarks





Attractions

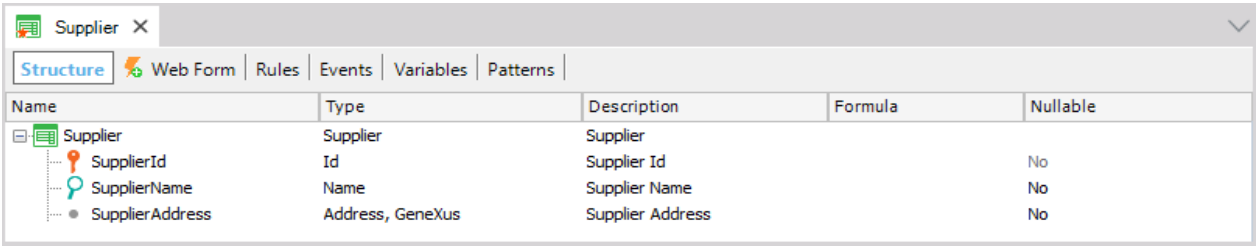
We've seen that **when we design transactions**, we can represent these relationships **by including the attributes of one transaction in another**.



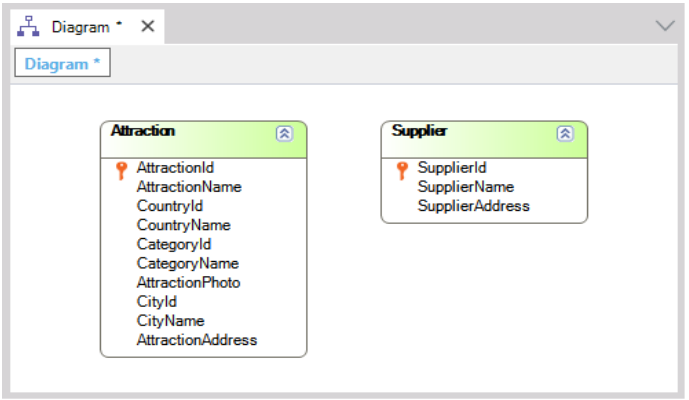
The agency's staff tells us that they work with suppliers, who from time to time offer them visits to tourist attractions in different parts of the world.

Each supplier offers many tourist attractions, but each attraction is managed by a single supplier.

New Supplier transaction



Relationship between Attraction and Supplier?



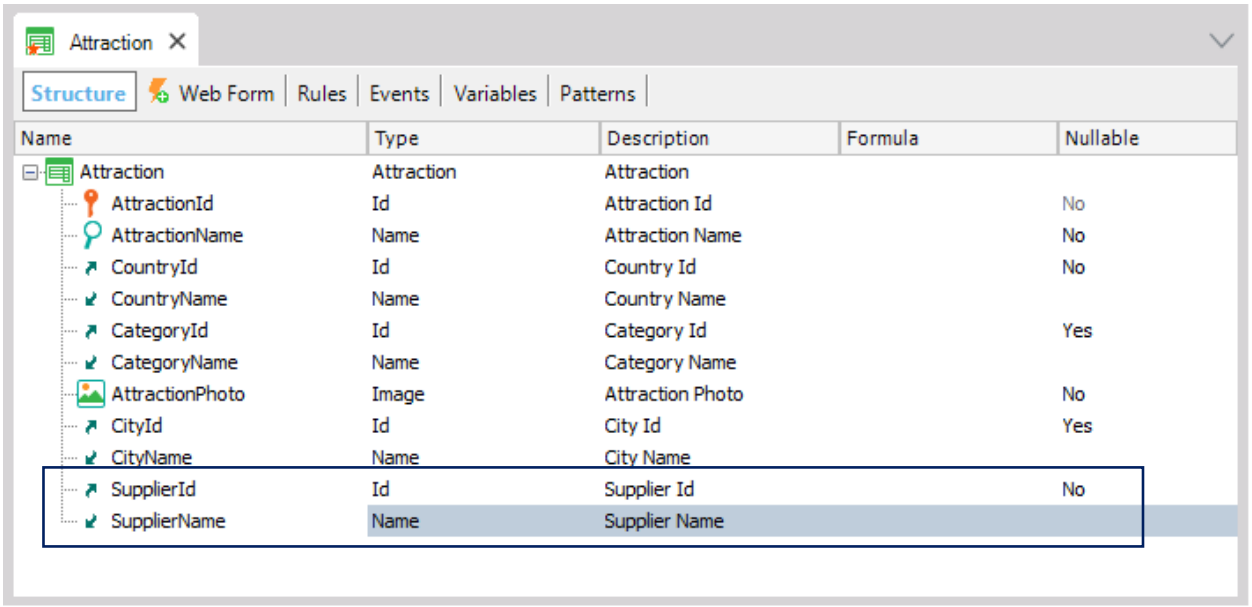
To represent this reality, we will create the Supplier transaction, where we will record suppliers...

We select File... New... Object... we call it Supplier.... and add these attributes:

SupplierId as identifier, SupplierName to store the supplier's name and SupplierAddress to save its address.

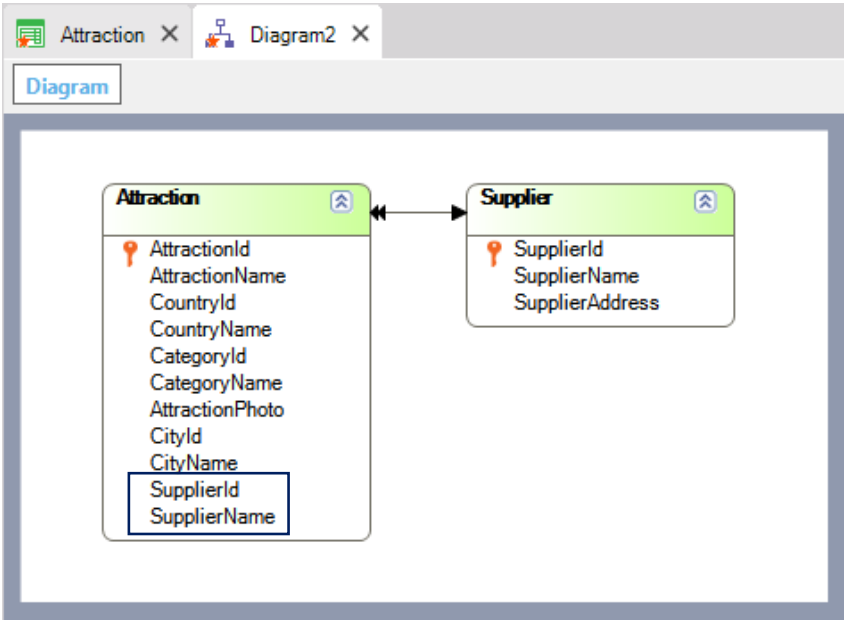
Using the transaction diagram object, let's look at the relationship between suppliers and attractions. We select New Object, of Diagram type, and drag the Attraction and Supplier transactions from here to the diagram. Note that we haven't established any relationship between these two actors yet. We save.

Requirement: each tourist attraction is offered by a unique supplier.



Since a tourist attraction has a single supplier that offers it, we will include the supplier identifier in the Attraction transaction structure. To do this, we open this transaction and add the SupplierId attribute. We also add the SupplierName attribute to be able to show the supplier's name in the attractions screen.

Another Look at the Relationship between Suppliers and Attractions



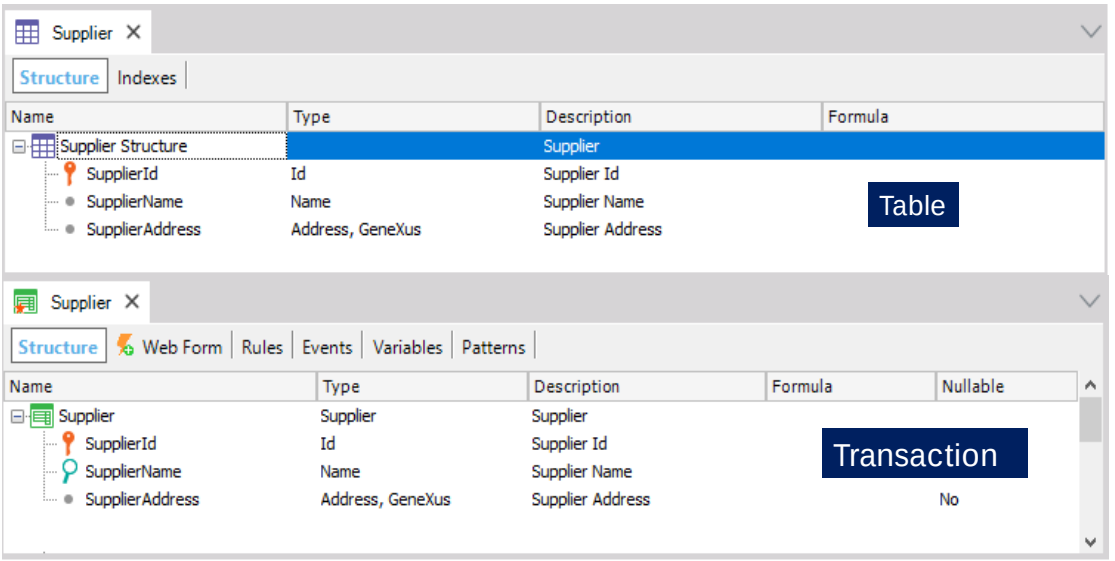
Transaction diagram (not table diagram)

We open the diagram again.

Now there's an arrow whose simple head is pointing to Supplier, and whose double head is pointing to Attraction. This indicates that an attraction has a single supplier and that a supplier can offer many attractions.

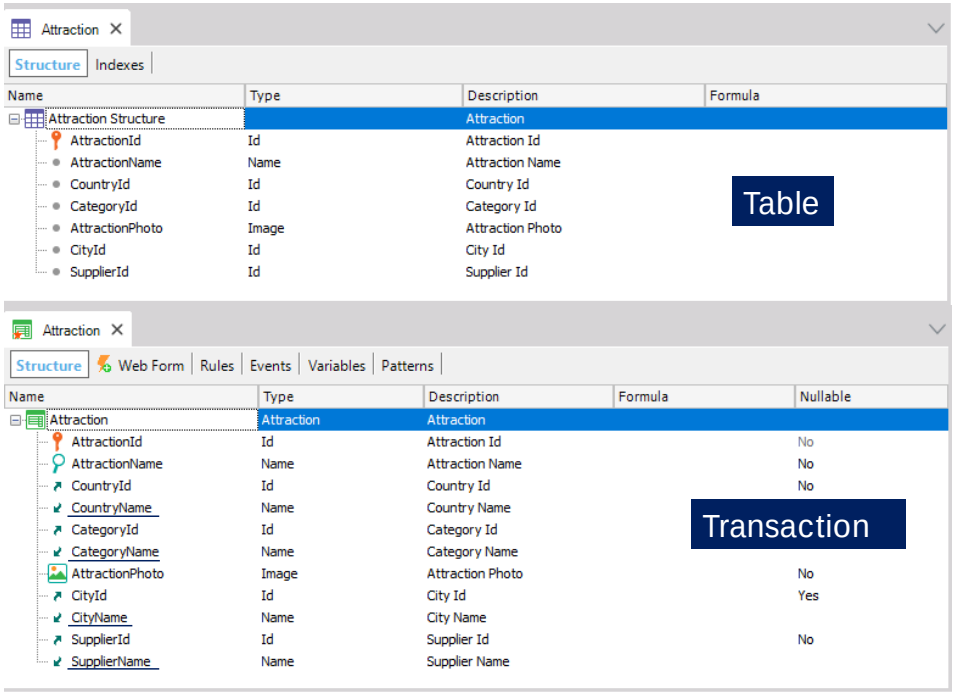
In sum, if we add the identifier attribute of a transaction to another transaction (which, as we have seen, will be a foreign key), a 1 to many relationship (also called "1 to N") will be established.

Tables Created by GeneXus Based on the Implemented Design



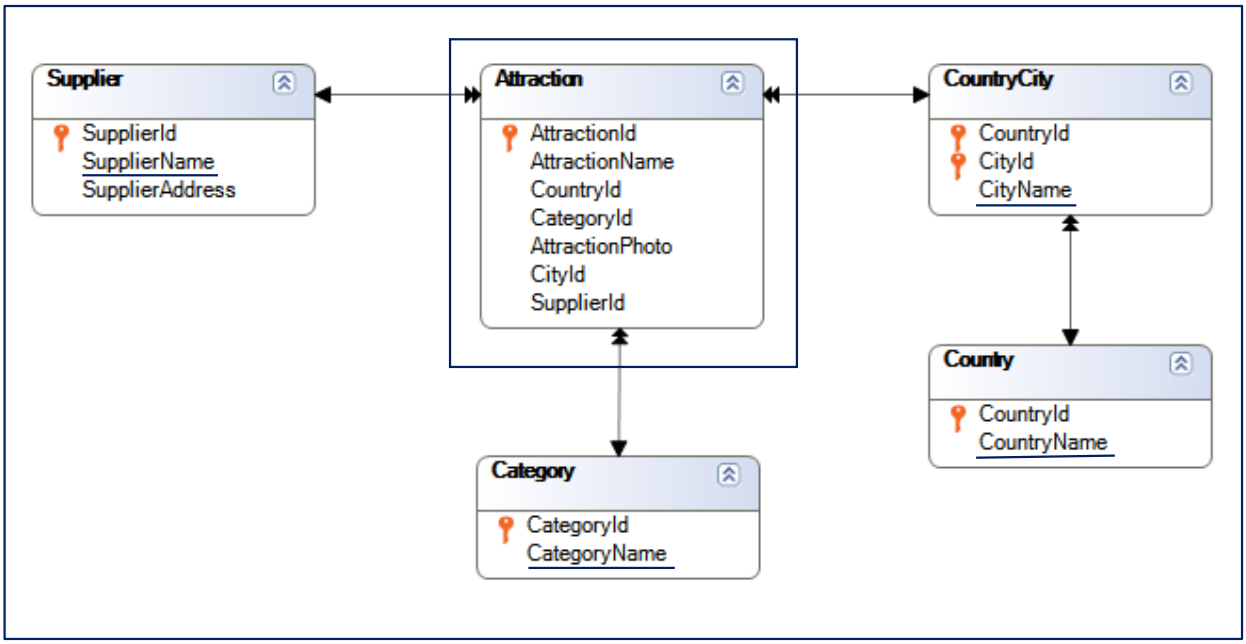
In it, the "many" side of the relationship is where the foreign key is located.

Now, if we examine the tables generated by GeneXus starting from this transaction design, we can see that based on the Supplier transaction, a SUPPLIER table will be created with the same structure as the transaction.



Based on the structure of the Attraction transaction, GeneXus creates an ATTRACTION table with the following structure.

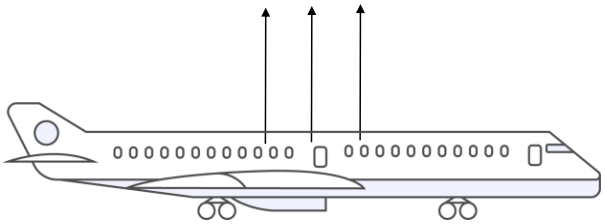
If we compare the structure of the ATTRACTION table to that of the Attraction transaction, we see that the CountryName, CategoryName, CityName and SupplierName attributes are not included in the table because they are inferred attributes.



As we've seen before, since they are in the extended table of the ATTRACTION table, their value can be retrieved from the tables where they are physically stored.

This is the most common way to represent a 1 to many relationship between two actors of reality; that is to say, between two entities in our system.

Seat

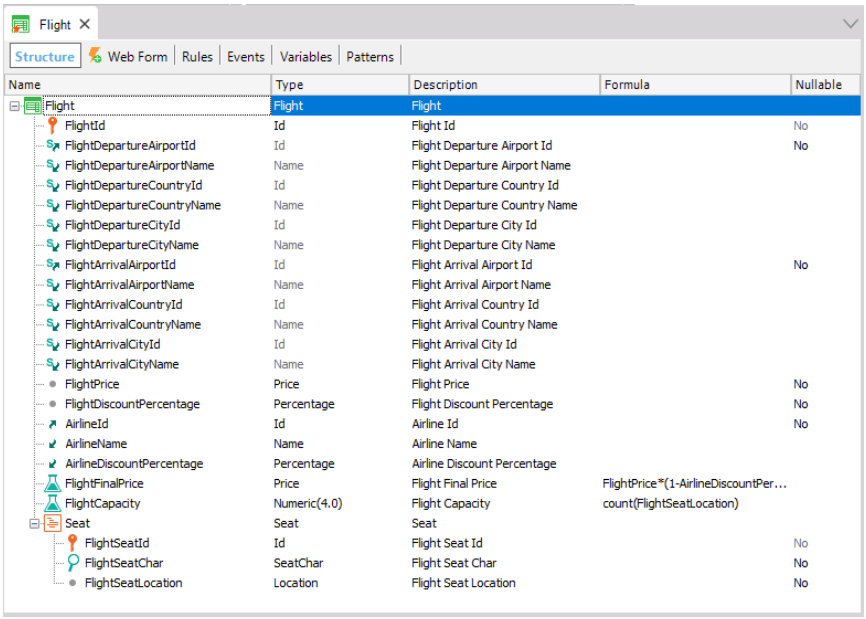


Flight

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

Remember flights, where one flight has many seats and each seat is assigned to a flight; that is to say, a 1 to many relationship.

Another Way to Model the 1 to N Relationship

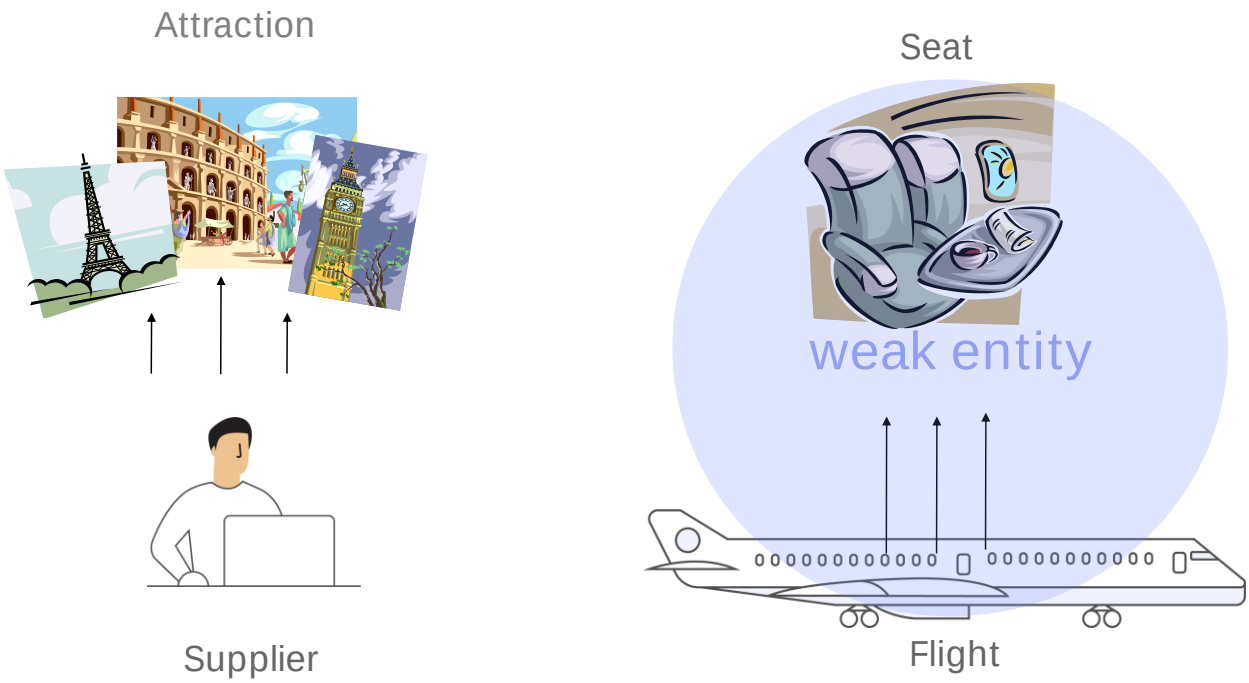


Name	Type	Description	Formula	Nullable
Flight	Flight	Flight		
FlightId	Id	Flight Id		No
FlightDepartureAirportId	Id	Flight Departure Airport Id		No
FlightDepartureAirportName	Name	Flight Departure Airport Name		
FlightDepartureCountryId	Id	Flight Departure Country Id		
FlightDepartureCountryName	Name	Flight Departure Country Name		
FlightDepartureCityId	Id	Flight Departure City Id		
FlightDepartureCityName	Name	Flight Departure City Name		
FlightArrivalAirportId	Id	Flight Arrival Airport Id		No
FlightArrivalAirportName	Name	Flight Arrival Airport Name		
FlightArrivalCountryId	Id	Flight Arrival Country Id		
FlightArrivalCountryName	Name	Flight Arrival Country Name		
FlightArrivalCityId	Id	Flight Arrival City Id		
FlightArrivalCityName	Name	Flight Arrival City Name		
FlightPrice	Price	Flight Price		No
FlightDiscountPercentage	Percentage	Flight Discount Percentage		No
AirlineId	Id	Airline Id		No
AirlineName	Name	Airline Name		
AirlineDiscountPercentage	Percentage	Airline Discount Percentage		
FlightFinalPrice	Price	Flight Final Price	FlightPrice*(1-AirlineDiscountPer…	
FlightCapacity	Numeric(4.0)	Flight Capacity	count(FlightSeatLocation)	
Seat	Seat	Seat		
FlightSeatId	Id	Flight Seat Id		No
FlightSeatChar	SeatChar	Flight Seat Char		No
FlightSeatLocation	Location	Flight Seat Location		No

We will open the Flight transaction structure to see how to represent this relationship...

In this case, Seat is included as a second level of the Flight transaction.

So, how is this 1 to many relationship different from the 1 to many relationship that we saw between Attractions and Suppliers?



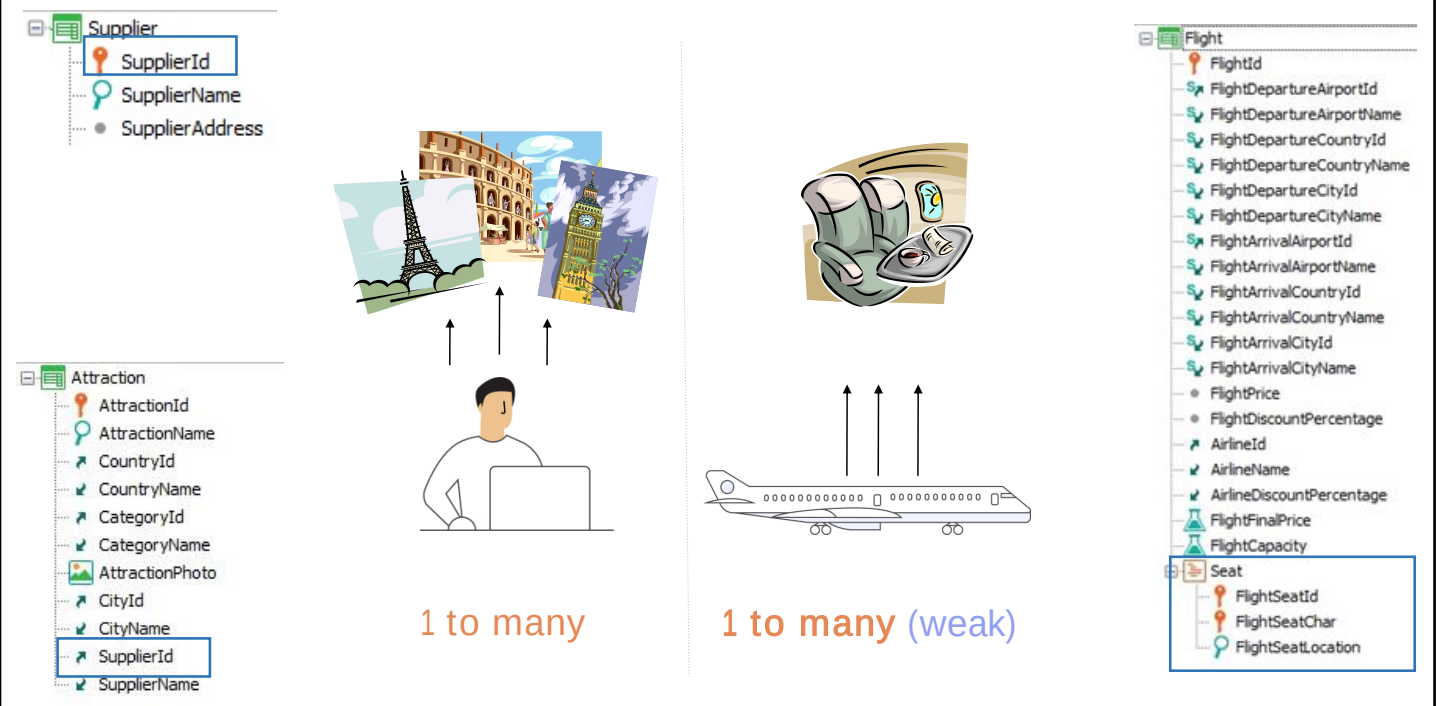
Why don't we represent both cases in the same way (with the same transaction design)?

Note that the existence of seats doesn't make sense unless they are in a flight; that is to say, it doesn't make sense to consider a seat without **always** relating it to the flight it belongs to...

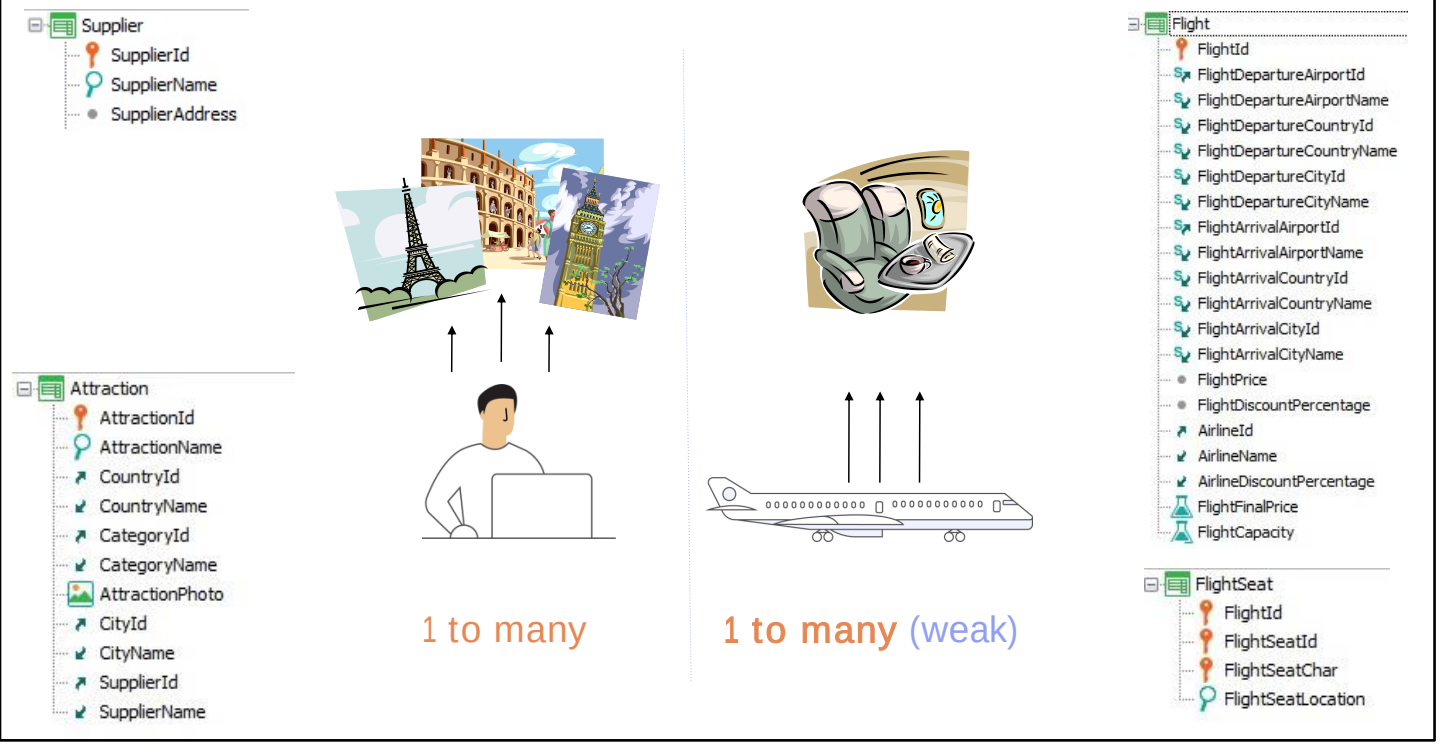
On the other hand, an attraction may not have a supplier that offers it, and it would nonetheless exist on its own...

The other difference is that when we're entering the details of a flight, we're also entering the details of its seats (just like when we enter an invoice with lines, all the information is entered at once). On the other hand, the Suppliers and Attractions details don't have to be entered all at the same time.

An entity such as seats, which only makes sense if it's represented in relation to another entity (in this case, flights), is called a **weak entity**.



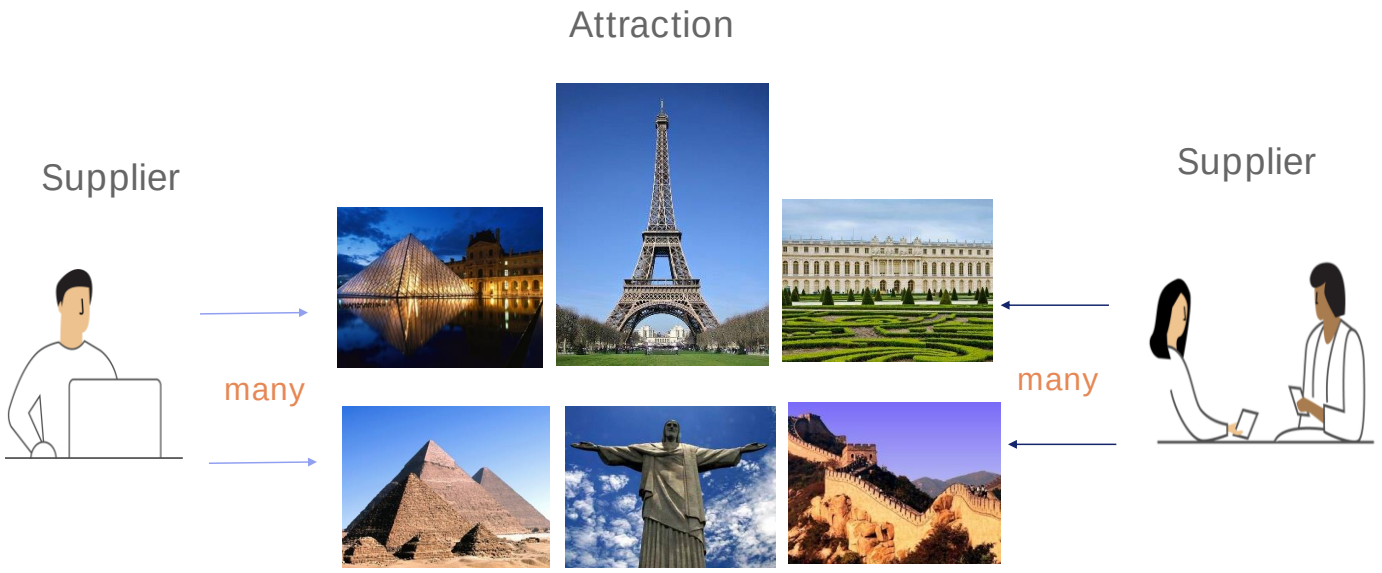
This type of **weak 1 to N** relationship is usually represented with a single two-level transaction, where the weak entity is in the second level. It is different from the 1 to N relationship of Suppliers and Attractions, where we created **two** transactions and set as foreign key the primary key of the other.



The weak 1 to N relationship can also be represented with two transactions (it is exactly the same for data modeling purposes), where part of the primary key of the seats transaction is the FlightId attribute. More specifically, this attribute will be the foreign key of the Flight table. There lies the difference between a strong and a weak entity. Note that since FlightId is part of the primary key, it is not possible to set a flight seat, such as 2 A Window, without giving a value to the flight, FlightId. On the other hand, it is possible to enter an attraction here without indicating its supplier, if that attribute has the Nullable property set to Yes.

So far, we've seen 1 to many relationships, but they don't always fit the reality that we want to represent.

"Many" to "many" (M to N) relationship



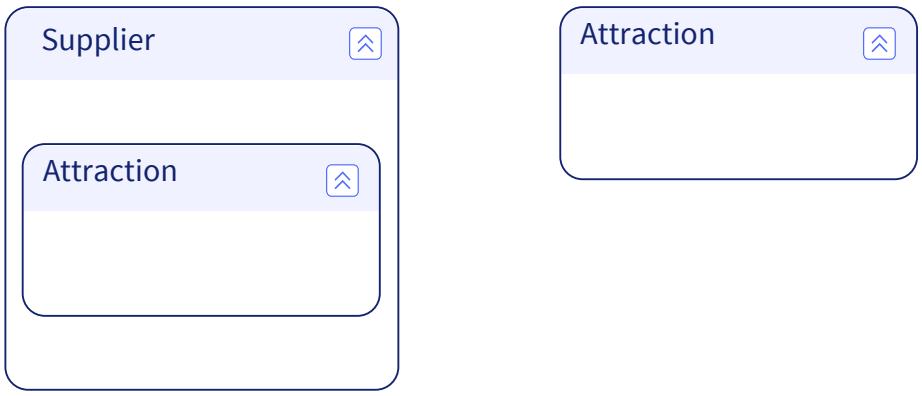
For example, suppose that the travel agency tells us that **their reality has changed.**

Each supplier offers many tourist attractions (as before), but *each attraction can be managed by SEVERAL suppliers (and not only one, as it has been the case until now)*.

That is to say, the relationship between Suppliers and Attraction is no longer "1 to many" but "**many to many**".

How do we represent this in GeneXus?

How to Represent an M to N Relationship

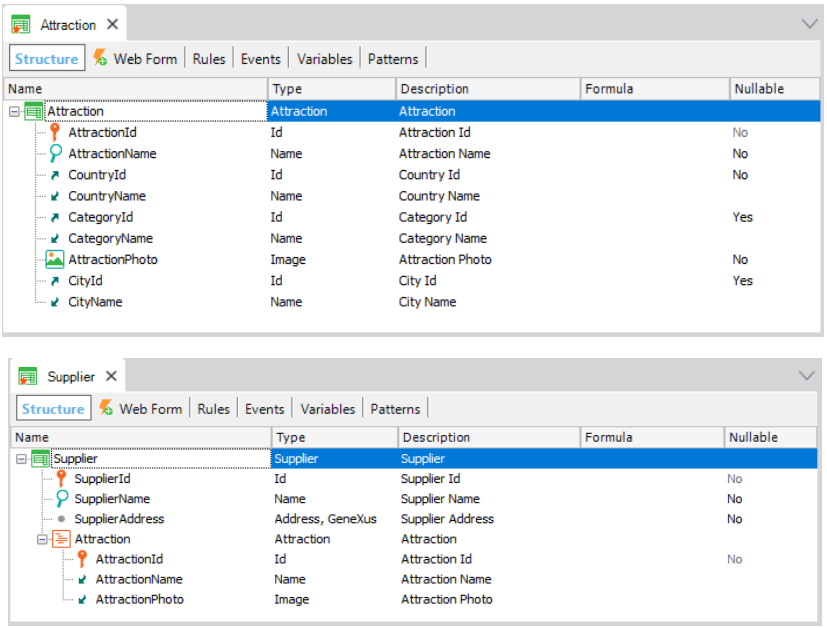


By using two transactions, one for each entity. In addition, one of them is added as the second level of the other. This is done by taking into account the way in which data will be entered: for each supplier, all its tourist attractions will be entered...

How to Represent an M to N Relationship

or... for each attraction, all its suppliers will be entered.

Modelling the M to N relationship in GeneXus



In this case, the agency has asked that users enter all the attractions of each supplier.

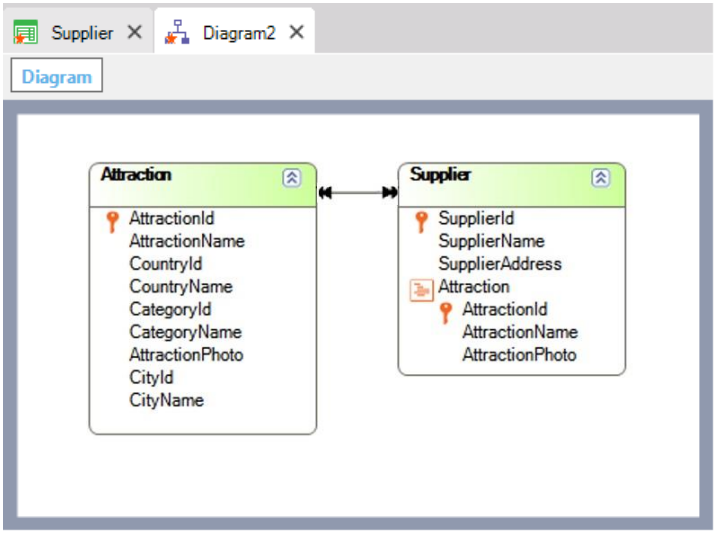
We will implement this in GeneXus.

To do so, we open the Attraction transaction and remove the SupplierId and SupplierName attributes, and save.

Now we open the Supplier transaction, where we add a second level and add these attributes: AttractionId (note that when we type a primary key attribute that begins with “Attraction”, the level name is automatically changed to Attraction).

Also, we add Attraction Name and AttractionPhoto.

A look at the relationship established between suppliers and attractions



Let's see how this relationship looks by opening the diagram of the Attraction and Supplier transactions.

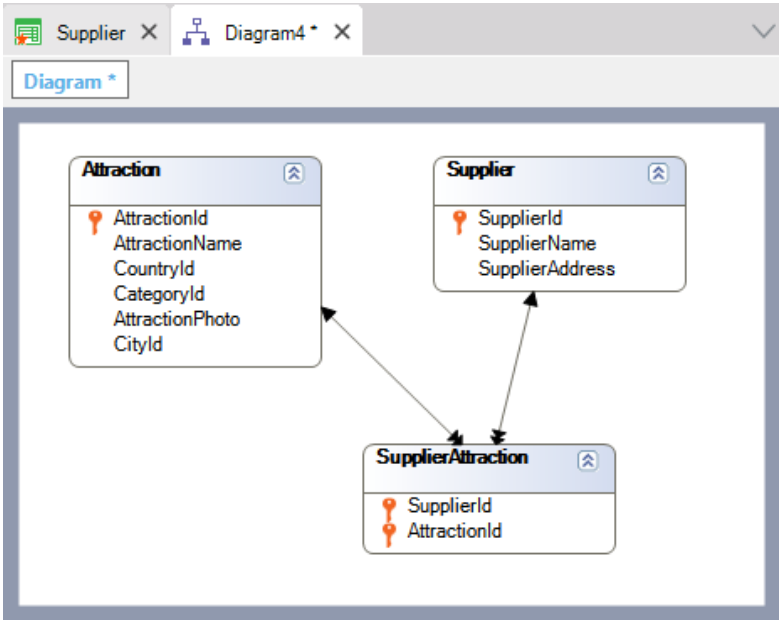
Now there's a double-headed arrow in each end of the relationship, which indicates that the relationship is “many” to “many”; that is to say, one attraction is offered by many suppliers, and one supplier offers many attractions.

Tables Created by GeneXus Based on the Implemented Design

Let's take a look at the tables created by GeneXus based on the previous design...

There's an ATTRACTION table, a SUPPLIER table and a SUPPLIERATTRACTION table.

Relationship between tables: M to N



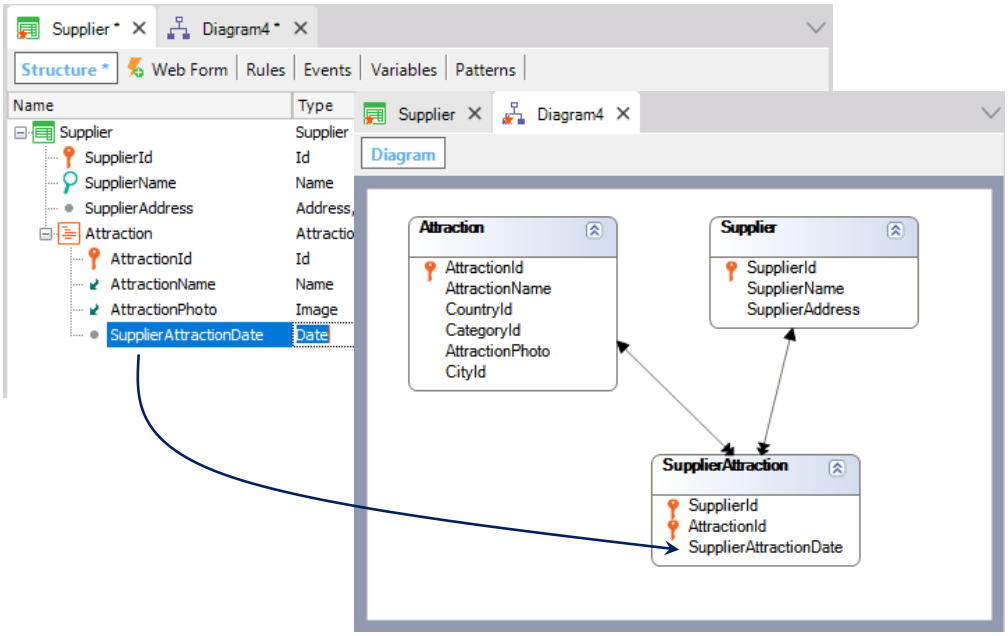
We create a new diagram object and drag the three tables to the diagram...

Note that, in this case, GeneXus creates a table for each transaction included in the many to many relationship (ATTRACTION and SUPPLIER), but it also creates a third table called SUPPLIERATTRACTION to establish the relationship.

Looking at the structure of this third table, we notice that only the identifier attributes of the other two tables are included.

Therefore, every time that GeneXus establishes a many to many relationship, it will be represented in the database with three tables; one for each entity involved and a third one with the identifiers of both tables. This third table may have its own attributes, such as, for example, the date in which the supplier started to offer that attraction.

Relationship between tables: M to N

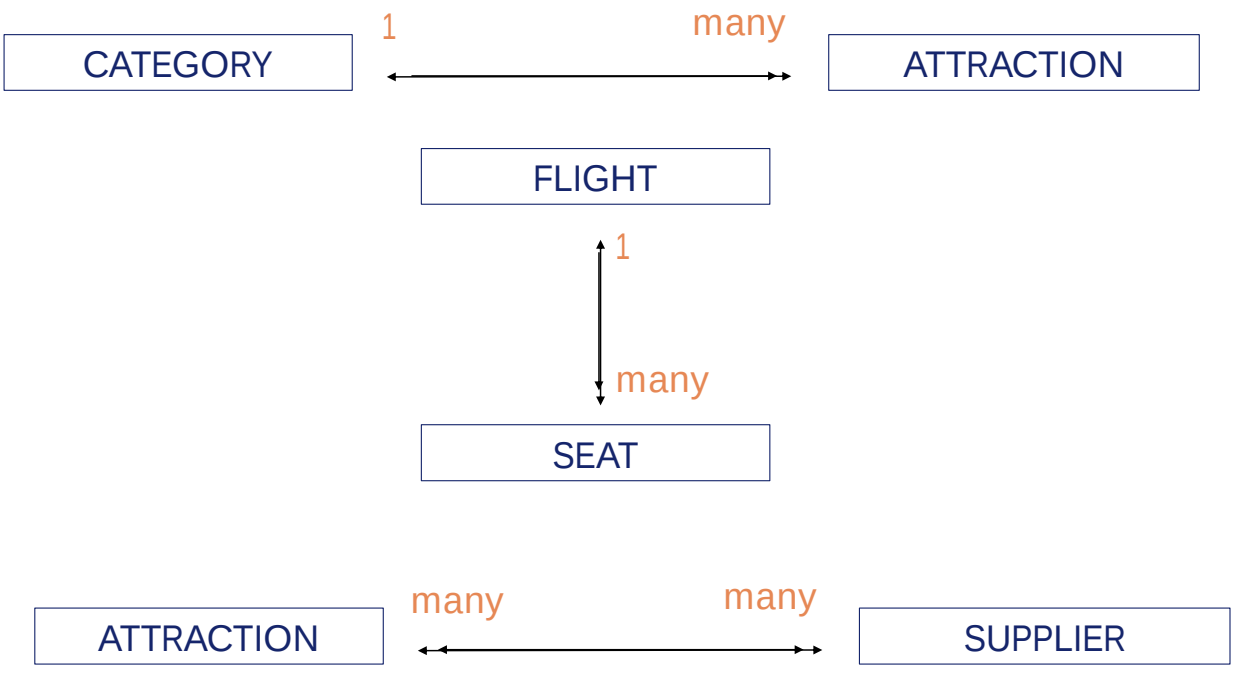


If we open the diagram again, we see the attribute in the relationship table.

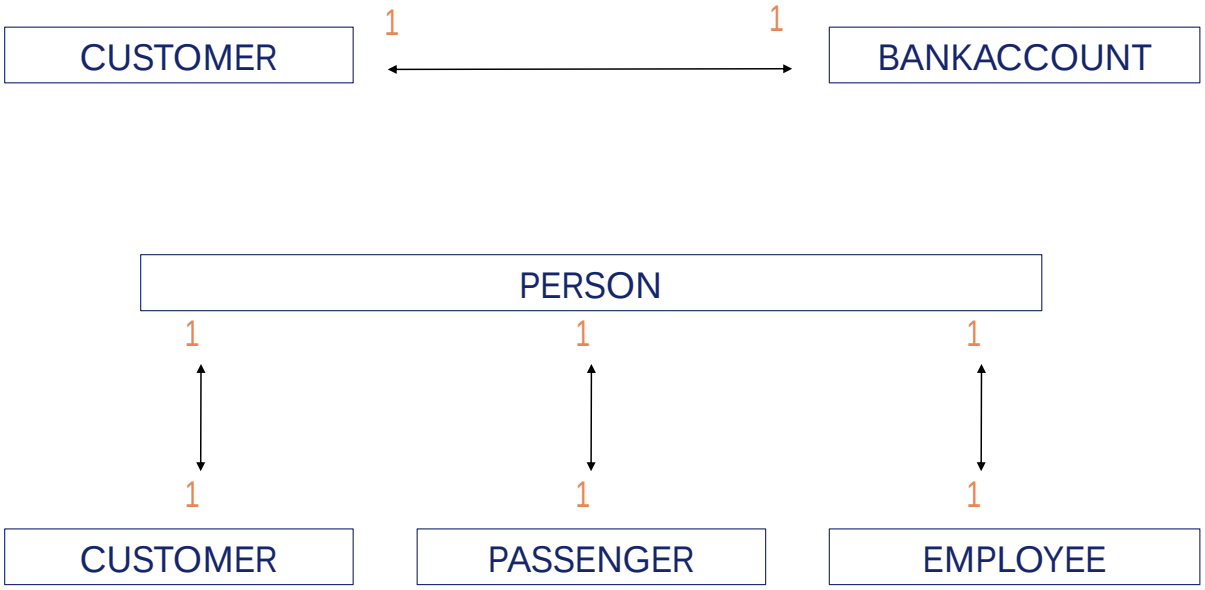
The many to many relationship between Attraction and Supplier has been divided into 2 one to many relationships, using the SUPPLIERATTRACTION table to establish the relationship between the previous ones.

Finally, let's update our KB in GeneXus Server...

And reorganize it to have the tables created...



So far, we've seen that using transactions and their attributes we can represent different relationships between the actors of our reality.



For example, when the travel agency needs to associate with each customer the bank account opened to pay for the services hired.

Another scenario of 1 to 1 relationships was mentioned when we talked about subtypes. It was an example of specialization: when an entity **is** a particular case of another.

Now, let's move on to the following topic.

GeneXus™

training.genexus.com
wiki.genexus.com