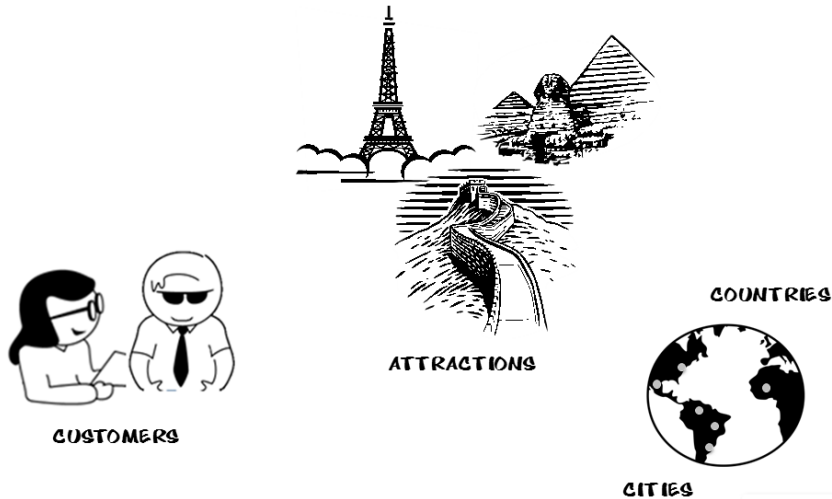


Relaciones entre entidades de la realidad

Relations between actors of reality



GeneXus[®]

En varios ejemplos de nuestra agencia de viajes, encontramos que los actores de la realidad se relacionan entre sí de distintas maneras, por ejemplo cuando vemos que una atracción pertenece a una categoría y que a su vez esa categoría, puede ser categoría de muchas atracciones.

The screenshot shows the 'Attraction Information' form for the Eiffel Tower. The form includes the following fields:

- Name**: Eiffel Tower
- General** section with a table:

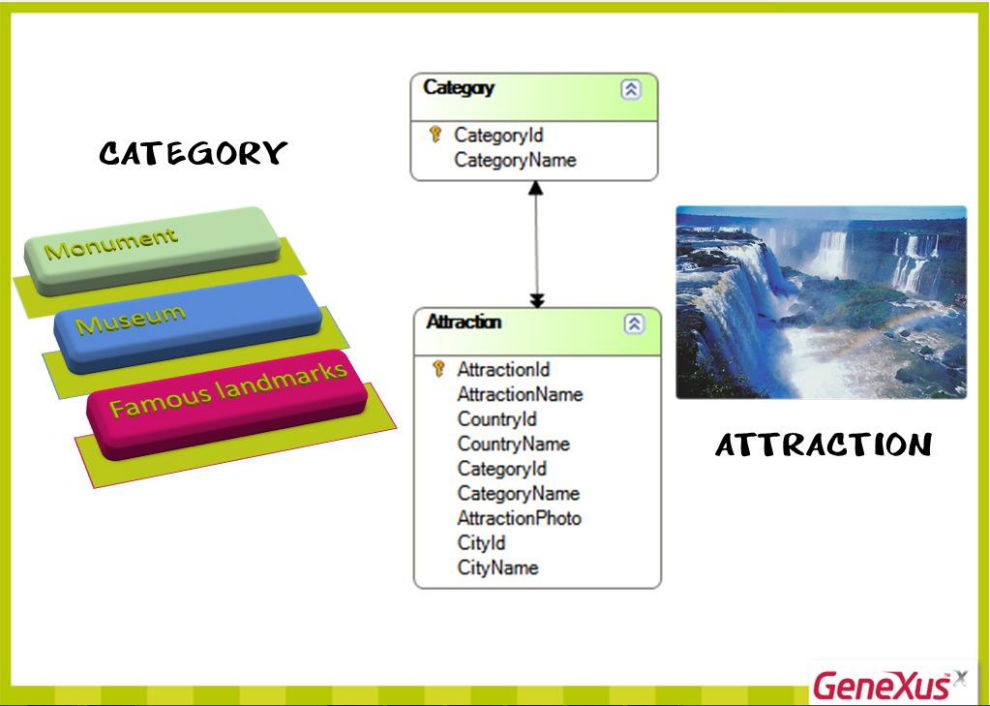
Id	Value
3	Eiffel Tower
Country Id	2
Country Name	France
Category Id	Monument
Category Name	Monument

Below the table is a 'Photo' field with a large image of the Eiffel Tower. At the bottom, there are fields for 'City Id' (1) and 'City Name' (Paris), along with 'Update' and 'Delete' buttons.

Red arrows point from the 'Monument' category in the table to three images: the Christ the Redeemer statue, the Eiffel Tower, and a large building with a garden.

GeneXus[®]

Vimos que la forma que tenemos para representar esas relaciones es **cuando diseñamos transacciones, incluir atributos de una transacción en otra.**

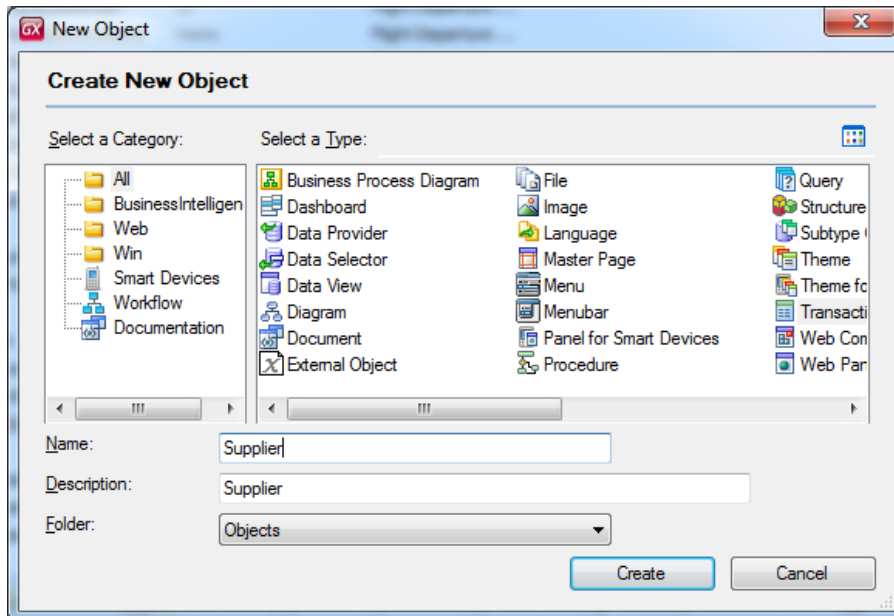


En nuestra agencia nos cuentan ahora que ellos trabajan con proveedores, quienes les ofrecen periódicamente visitas a atracciones turísticas en distintas partes del mundo.

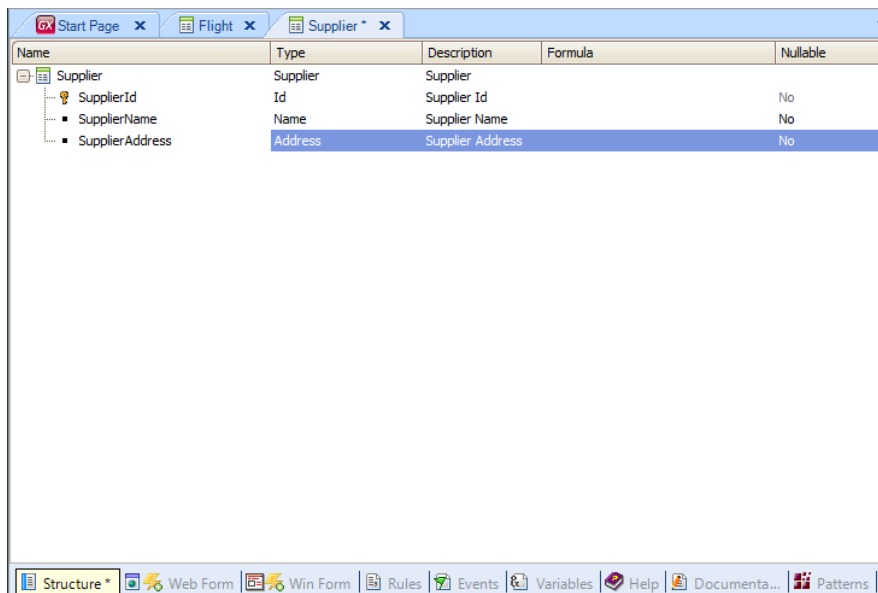


Cada proveedor ofrece muchas atracciones turísticas, pero cada atracción es manejada por un único proveedor. Para representar esta realidad, vamos a crear la transacción Supplier, en la cual registraremos a los proveedores...

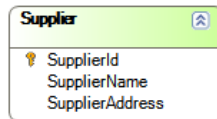
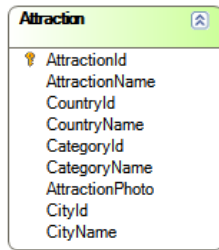
Hacemos File...New...Object...llamamos Supplier.... Y agregamos los atributos:



SupplierId como identificador, SupplierName para almacenar el nombre del proveedor y SupplierAddress para guardar su dirección.



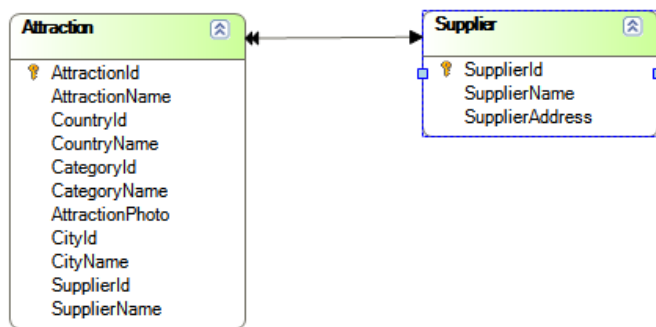
Observemos mediante el objeto diagrama de transacciones, la relación entre los proveedores y las atracciones. Hacemos ...New..Object del tipo Diagrama y arrastramos las transacciones Attraction y Supplier desde el Folder View. Vemos que no hemos establecido aún ninguna relación entre estos dos actores.



Como una atracción turística tiene un único proveedor que la ofrece, vamos a incluir al identificador de proveedor en la estructura de la transacción Attraction, así que abrimos dicha transacción y agregamos al atributo SupplierId. Agregamos también al atributo Supplier Name porque de esa forma podemos mostrar el nombre del proveedor en la pantalla de las atracciones.

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		

Ahora creamos un nuevo diagrama, arrastramos ambas transacciones nuevamente... y vemos que hay una flecha simple apuntando a Supplier y una flecha doble que apunta a Attraction, indicándonos que una atracción tiene un único proveedor y que un proveedor puede ofrecer muchas atracciones.



Recapitulando, si agregamos el atributo identificador de una transacción a otra transacción (el cual como ya vimos cumplirá aquí el rol de llave foránea, se establece una relación 1 a muchos (también llamada “1 a N”), donde el lado “muchos” de la relación, es donde está la clave foránea.

Si analizamos ahora cuáles serían las tablas que GeneXus genera a partir de este diseño de transacciones, vemos que a partir de la transacción Supplier, se creará una tabla SUPPLIER con igual estructura que la transacción:

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

Y a partir de la estructura de la transacción Attraction, GeneXus crea una tabla ATTRACTION con la siguiente estructura:

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	

Si comparamos la estructura de la tabla ATTRACTION con la de la transacción Attraction,

The image shows two screenshots from the GeneXus IDE. The top screenshot displays the 'Attraction Structure' tree on the left and a table definition with columns: Name, Type, Description, and Formula. The bottom screenshot shows the same table definition but with an additional 'Nullable' column.

Name	Type	Description	Formula
Attraction	Attraction	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	
SupplierId	Id	Supplier Id	

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		

vemos que los atributos CountryName, CategoryName, CityName y SupplierName no se incluyen en la tabla ya que son atributos inferidos y como vimos antes, como están en la tabla extendida de la tabla ATTRACTION, su valor puede ser recuperado de la tabla donde están físicamente almacenados.

Esta es la forma más común de representar la relación 1 a muchos entre 2 actores de la realidad, es decir, entre 2 entidades de nuestro sistema.

Sin embargo, hay otros casos de relaciones 1 a muchos, en las que usaremos otro tipo de representación.



FLIGHT



SEAT

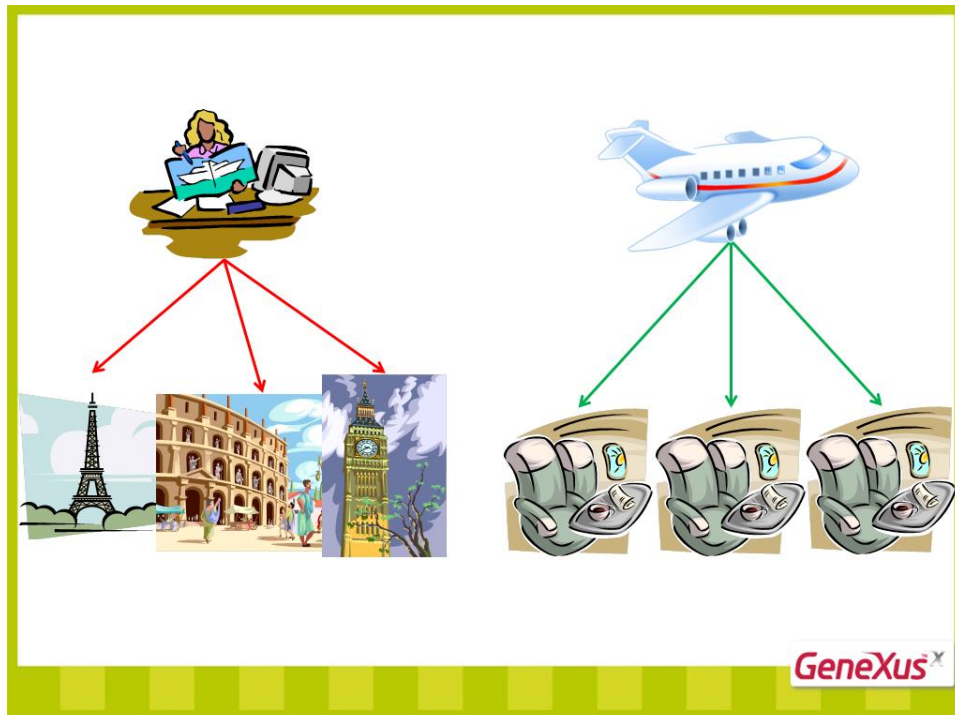
GeneXus[®]

Recordemos el caso de los vuelos, donde un vuelo tiene muchos asientos y cada asiento está asignado a un vuelo, es decir una relación 1 a muchos. Vamos a abrir la estructura de la transacción Flight para ver cómo representamos esta relación....

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	FlightPrice*(1-AirlineDiscountPercent...	
AirlineId	Id	Airline Id		Yes
AirlineName	Name	Airline Name		
AirlineDiscountPercentage	Percentage	Airline Discount P...		
FlightCapacity	Numeric(4.0)	Flight Capacity	count(FlightSeatLocation, FlightSeat...	
FlightSeatQty	Numeric(4.0)	Flight Seat Qty	count(FlightSeatChar)	
Seat	Seat	Seat		
FlightSeatId	Id	Flight Seat Id		No
FlightSeatLocation	Location	Flight Seat Location		No
FlightSeatChar	SeatChar	Flight Seat Char		No

Vemos que en este caso el asiento (Seat) está como un segundo nivel en la transacción Flight.

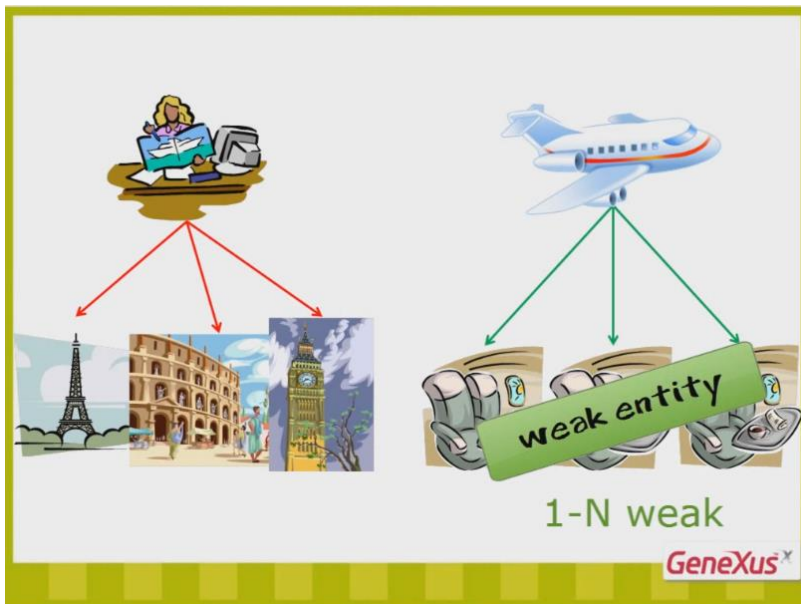
Entonces ¿en qué se diferencia esta relación de 1 a muchos con la relación de 1 a muchos que vimos entre las Atracciones y sus Proveedores?



¿Por qué no representamos de la misma manera (con el mismo diseño de transacciones) ambos casos?

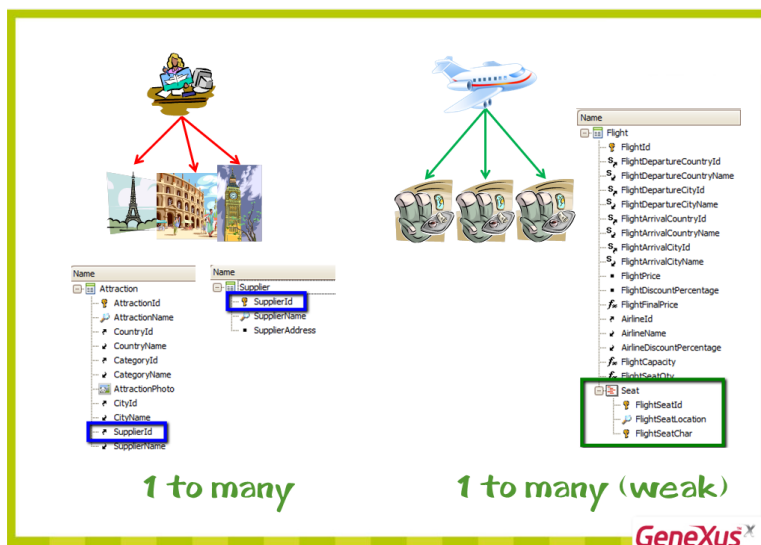
Notemos que los asientos no tiene sentido de que existan si no están en un vuelo, es decir, no tiene sentido considerar un asiento sin relacionarse **siempre** con el vuelo al que pertenece... Sin embargo, una atracción podría no tener un proveedor que la ofrezca y sin embargo puede existir por sí misma como tal...

La otra diferencia es que cuando ingresamos los datos de un vuelo ya ingresamos también los datos de sus asientos (igual que cuando ingresamos una factura con sus líneas, ingresamos toda la información junta, a la vez). En cambio, los datos de Proveedores y Atracciones no tienen por qué ingresarse todos en el mismo momento.



Una entidad como los asientos, que solamente tiene sentido que exista si se representa en función de otra entidad (en este caso los vuelos), decimos que es una **entidad débil**.

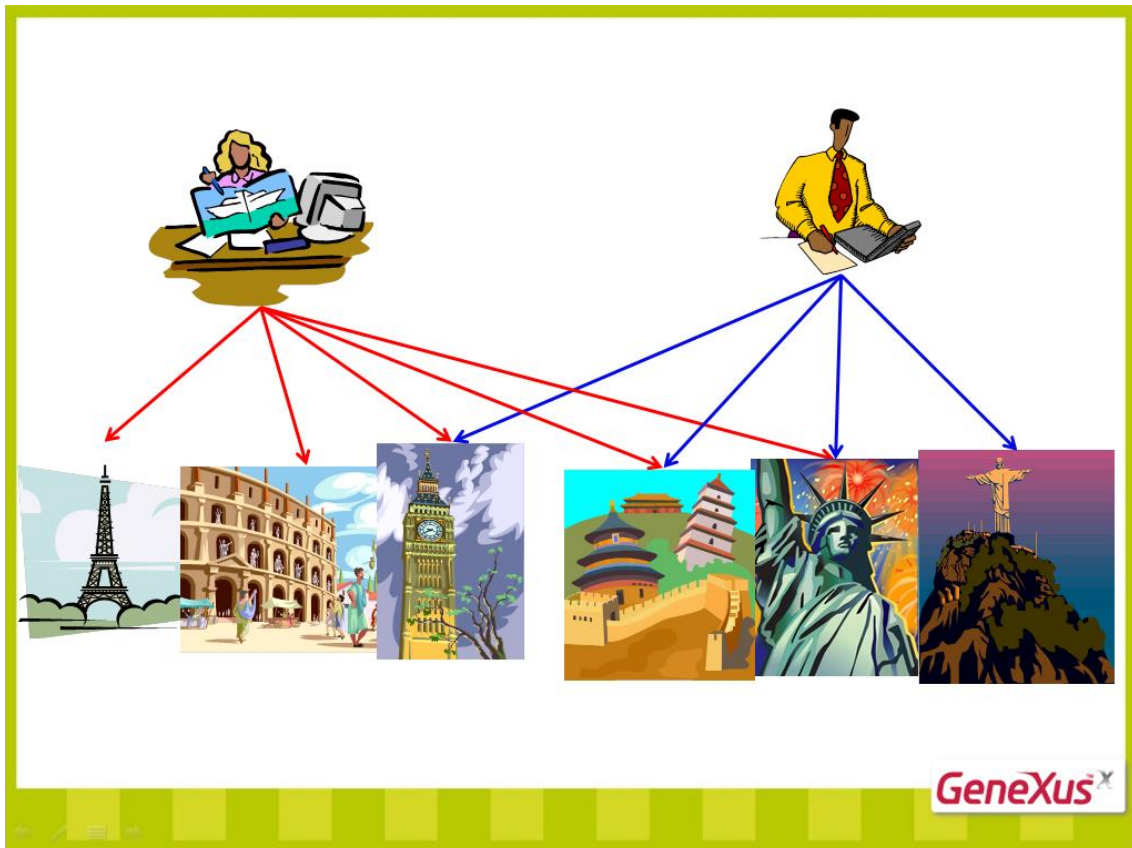
Este tipo de relación **1 a N débil** la representamos con una única transacción, de 2 niveles, donde la entidad débil está en el segundo nivel. A diferencia de la relación 1 a N de Proveedores y Atracciones, donde creamos 2 transacciones, y en una pusimos como clave foránea la clave primaria de la otra.



Muy bien, hasta ahora vimos relaciones 1 a muchos, pero este no es siempre el caso de la realidad que debemos representar.

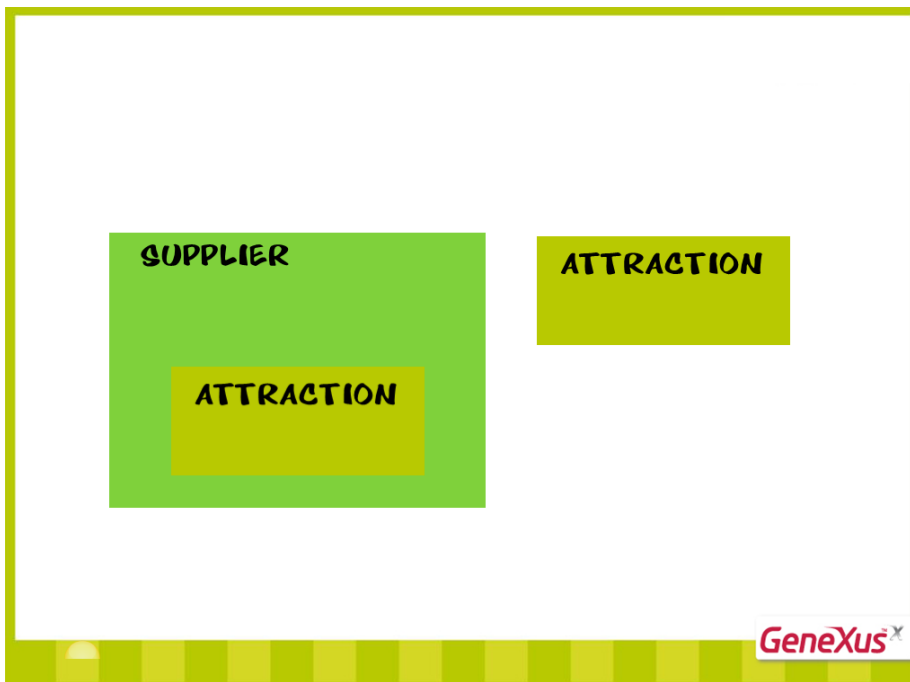
Supongamos por ejemplo que la agencia de viajes nos dice que **su realidad ha cambiado**:

Cada proveedor ofrece muchas atracciones turísticas (como hasta ahora), pero *cada atracción puede ser manejada por VARIOS proveedores (y no 1 como hasta ahora)*.



O sea que la relación entre Proveedores y Atracciones ya no es “1 a muchos” sino “**muchos a muchos**”.

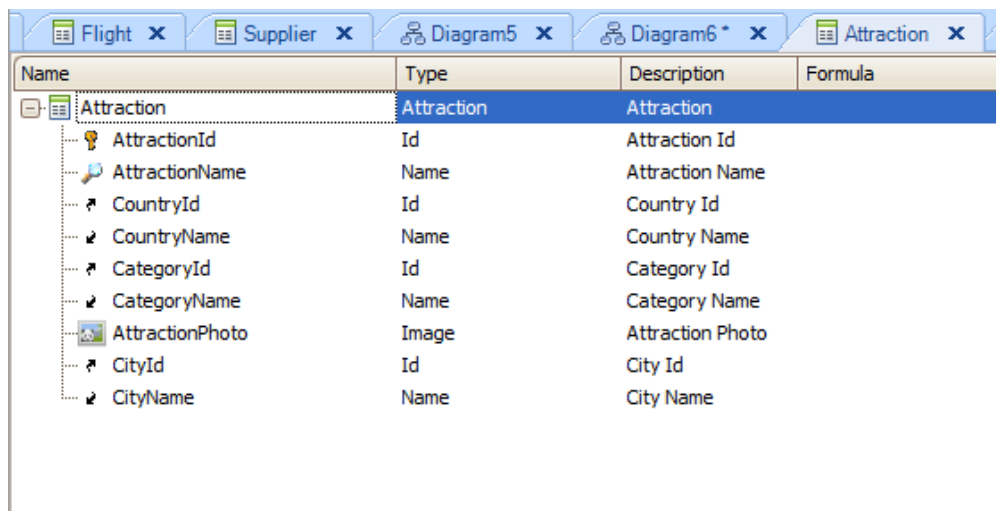
¿Cómo representamos esto en GeneXus?



La solución es utilizando 2 transacciones, una para cada entidad. Además, a una de ellas la agregamos como segundo nivel de la otra. Esto último lo hacemos teniendo en cuenta cómo es que se van a ingresar los datos, si para c/proveedor todas sus atracciones turísticas, o para c/atracción todos los proveedores que la proveen. En este caso lo más lógico, es que cuando vayamos a ingresar los datos de un proveedor, ya ingresemos todas las atracciones que éste provee.

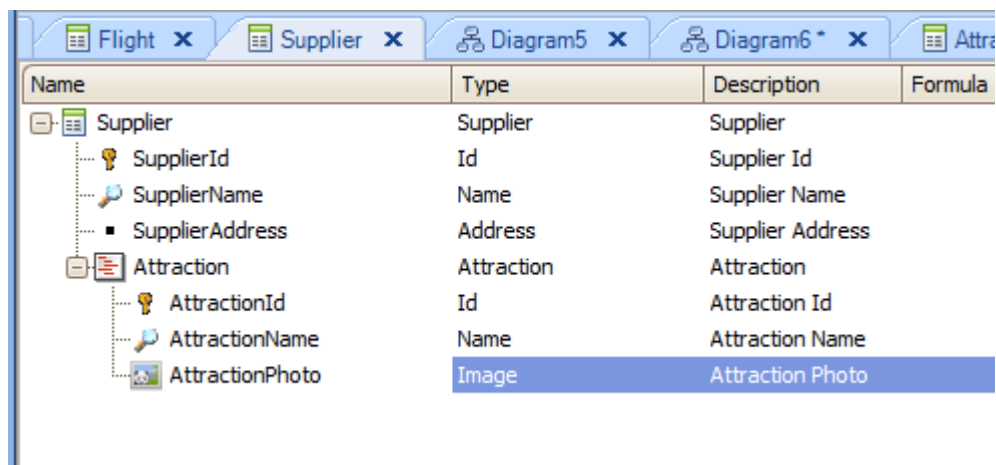
Vamos a hacer esto en GeneXus...

Abrimos la transacción Attraction y le quitamos los atributos SupplierId y SupplierName y salvamos.



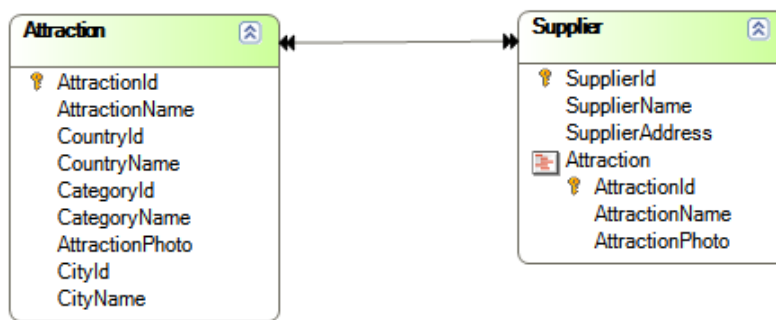
Name	Type	Description	Formula
Attraction	Attraction	Attraction	
AttractionId	Id	Attraction Id	
AttractionName	Name	Attraction Name	
CountryId	Id	Country Id	
CountryName	Name	Country Name	
CategoryId	Id	Category Id	
CategoryName	Name	Category Name	
AttractionPhoto	Image	Attraction Photo	
CityId	Id	City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

Ahora abrimos la transacción Supplier, donde agregamos un segundo nivel y agregamos los atributos: AttractionId, Attraction Name y AttractionPhoto. Vemos que el segundo nivel quedó con el nombre 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	

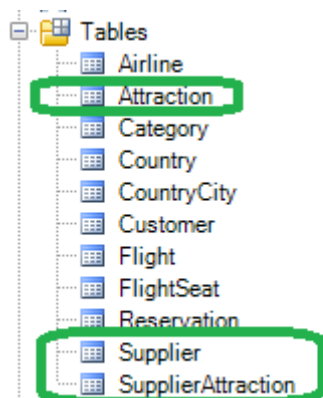
Vamos a ver cómo nos quedó esta relación creando un diagrama nuevo ...File ...New...Diagram... y arrastramos a las transacciones Attraction y Supplier al diagrama.



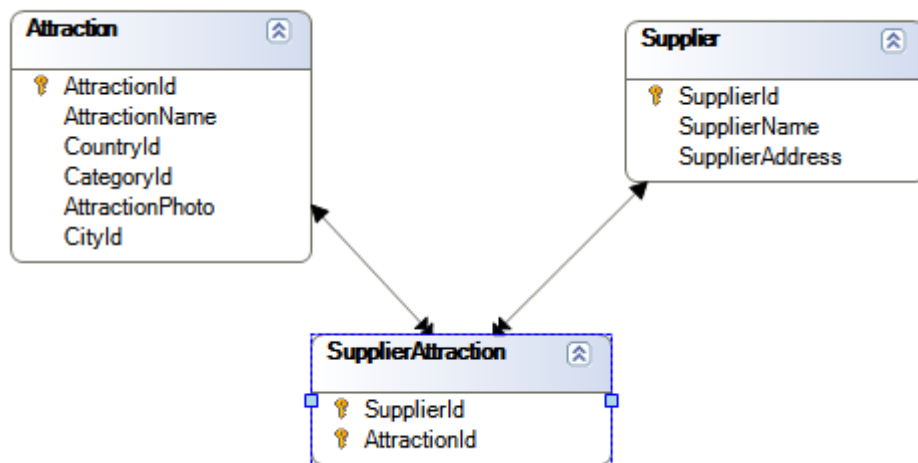
Ahora hay una doble flecha en cada extremo de la relación, lo que indica que la relación es de “muchos” a “muchos”, es decir que una atracción es provista por muchos proveedores y que un proveedor provee muchas atracciones.

Veamos ahora las tablas que GeneXus creará a partir del diseño anterior...

Vemos que encontramos una tabla ATTRACTION, una tabla SUPPLIER y una tabla llamada SUPPLIERATTRACTION.



Creemos un objeto diagrama nuevo y arrastremos las 3 tablas al diagrama....

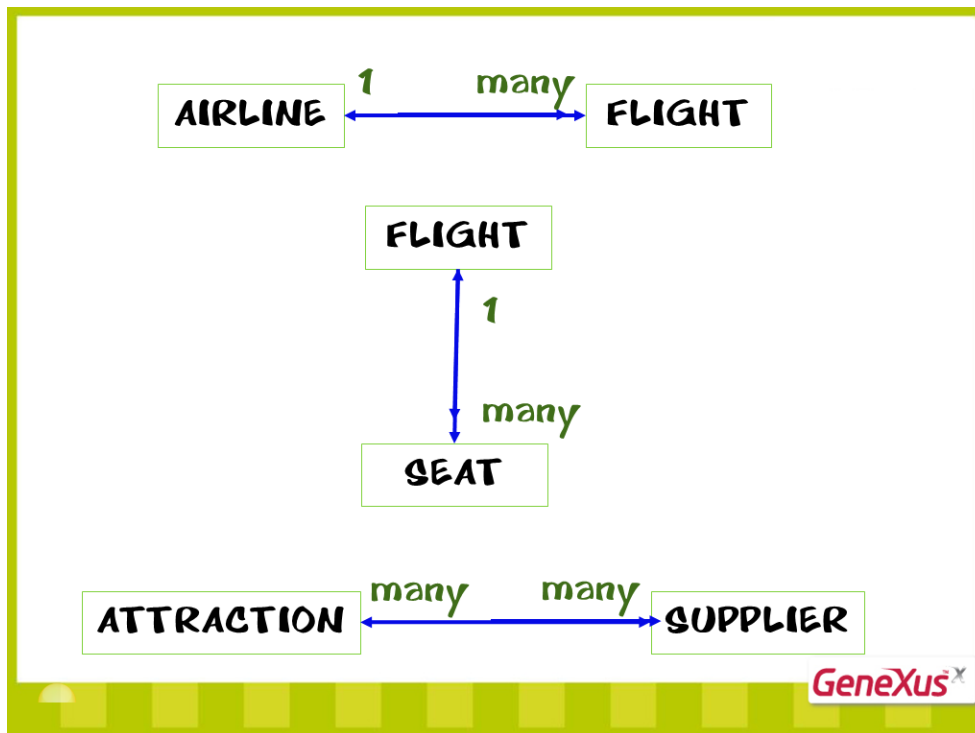


Notemos que en este caso, GeneXus crea una tabla por cada transacción que interviene en la relación muchos a muchos (ATTRACTION y SUPPLIER), pero además crea una tercera tabla llamada SUPPLIERATTRACTION, para establecer la relación.

Si observamos la estructura de esta tercera tabla, vemos que solamente se incluyen los atributos identificadores de las otras dos tablas.

Por lo tanto, cada vez que GeneXus establezca una relación de muchos a muchos, dicha relación será representada en la base de datos por 3 tablas, una por cada entidad interviniente y una tercera con los identificadores de ambas.

Vemos que la relación de muchos a muchos entre Attraction y Supplier, se descompuso en 2 relaciones uno a muchos, utilizando la tabla SUPPLIERATTRACTION para establecer la relación entre las anteriores.



Hemos visto así que mediante transacciones y sus atributos, podemos representar distintas relaciones entre los actores de nuestra realidad.