

Relaciones entre entidades de la realidad

GeneXus™

| Attraction Information | |
|---|--------------|
| Name | Eiffel Tower |
| General | |
| Id | 3 |
| Name | Eiffel Tower |
| Country Id | 2 |
| Country Name | France |
| Category Id | 1 |
| Category Name | Museum |
|  | |
| <input type="button" value="UPDATE"/> <input type="button" value="DELETE"/> | |



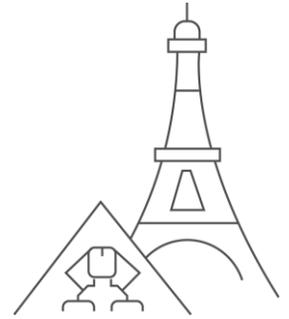
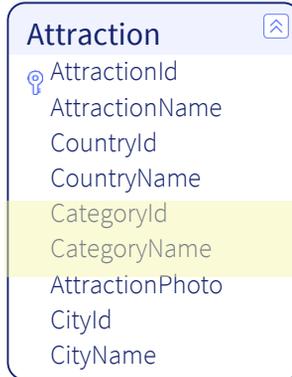
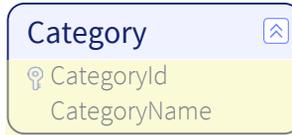
En varios ejemplos de nuestra agencia de viajes, encontramos que los actores de la realidad se relacionan entre sí de distintas maneras, por ejemplo cuando vemos que una atracción pertenece a una categoría y que a su vez esa categoría, puede ser categoría de muchas atracciones.

CATEGORY

Monumento

Museo

Punto de interés



Attractions

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



SUPPLIER

1

muchos

ATTRACTION

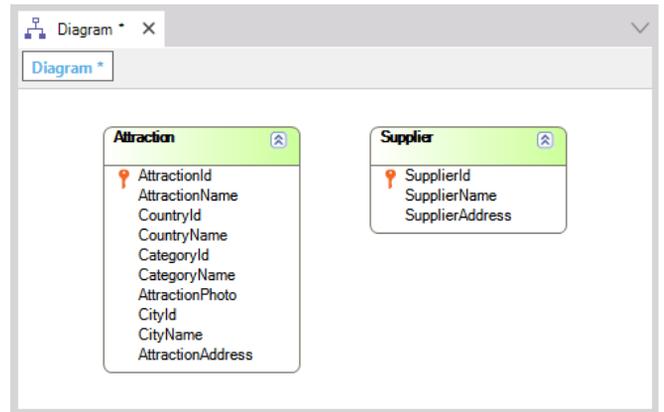
En nuestra agencia nos cuentan ahora que ellos trabajan con proveedores, quienes les ofrecen periódicamente visitas a atracciones turísticas en distintas partes del mundo.

Cada proveedor ofrece muchas atracciones turísticas, pero cada atracción es manejada por un único proveedor.

Nueva transacción Supplier

| Name | Type | Description | Formula | Nullable |
|-----------------|------------------|------------------|---------|----------|
| Supplier | Supplier | Supplier | | |
| SupplierId | Id | Supplier Id | | No |
| SupplierName | Name | Supplier Name | | No |
| SupplierAddress | Address, GeneXus | Supplier Address | | No |

¿Relación entre Attraction y Supplier?



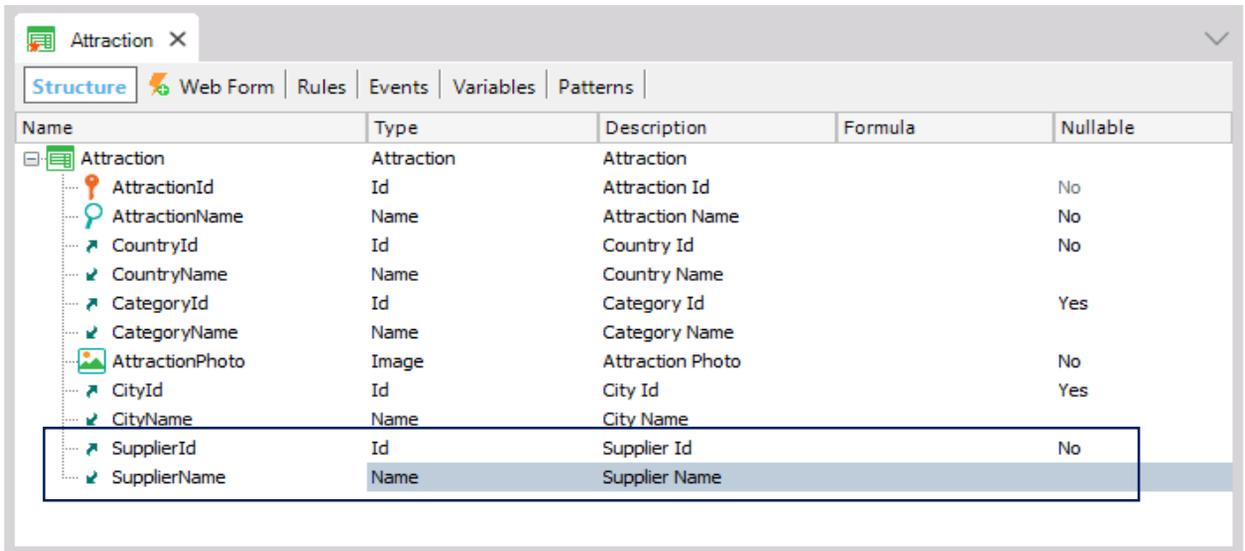
Para representar esta realidad, vamos a crear la transacción Supplier, en la cual registraremos a los proveedores...

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 aquí hasta el diagrama. Vemos que no hemos establecido aún ninguna relación entre estos dos actores. Lo grabamos

Requerimiento: cada atracción turística es ofrecida po un único proveedor.



| 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 | | |

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

Otro vistazo a la relación entre proveedores y atracciones

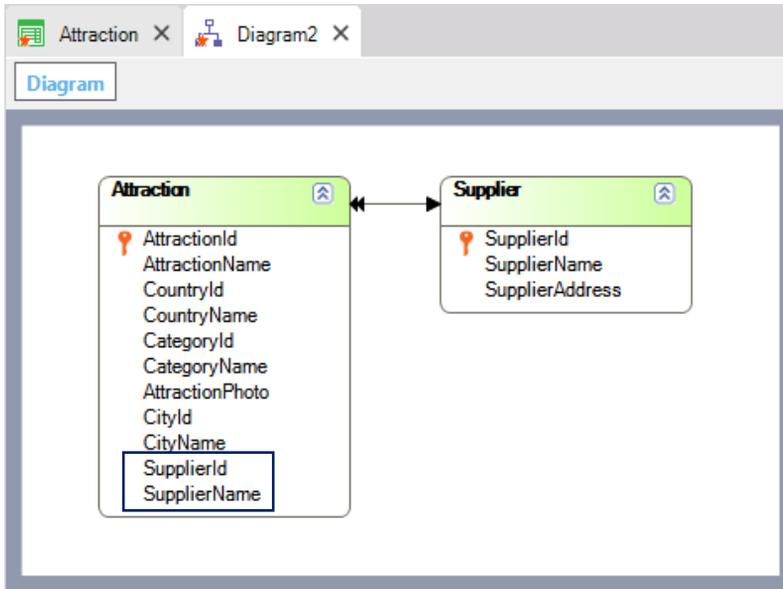


Diagrama de Transacciones
(no diagrama de tablas)

Volvamos a abrir el diagrama.

Vemos que ahora hay una flecha cuya punta simple apunta a Supplier y cuya punta doble 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").

Tablas creadas por Genexus en base al diseño implementado

The image displays two screenshots from the GeneXus IDE, showing the structure of a 'Supplier' entity and its corresponding transaction.

Top Screenshot: Supplier Structure (Table)

| Name | Type | Description | Formula |
|--------------------|------------------|------------------|---------|
| Supplier Structure | | Supplier | |
| SupplierId | Id | Supplier Id | |
| SupplierName | Name | Supplier Name | |
| SupplierAddress | Address, GeneXus | Supplier Address | |

Bottom Screenshot: Supplier (Transaction)

| Name | Type | Description | Formula | Nullable |
|-----------------|------------------|------------------|---------|----------|
| Supplier | Supplier | Supplier | | |
| SupplierId | Id | Supplier Id | | |
| SupplierName | Name | Supplier Name | | No |
| SupplierAddress | Address, GeneXus | Supplier Address | | No |

En 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:

The image shows two screenshots from the GeneXus IDE. The top screenshot displays the 'Attraction Structure' window, which lists the attributes of the transaction. The bottom screenshot displays the 'Attraction' table structure window, which lists the attributes of the table generated from the transaction.

Attraction Structure (Transaction):

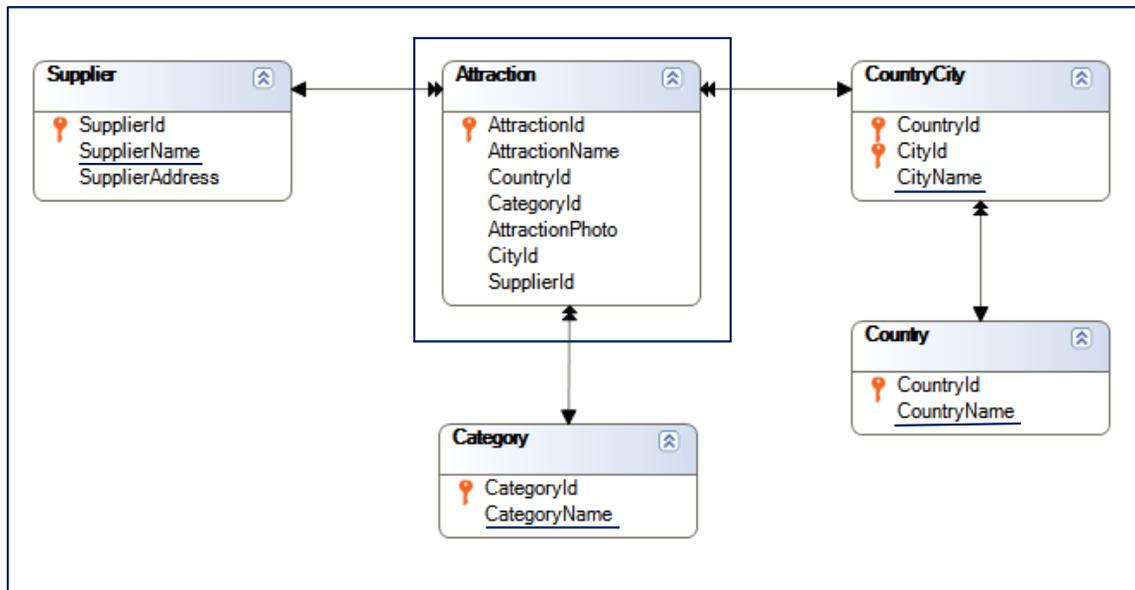
| Name | Type | Description | Formula |
|-----------------|-------|------------------|---------|
| AttractionId | Id | Attraction Id | |
| AttractionName | Name | Attraction Name | |
| CountryId | Id | Country Id | |
| CategoryId | Id | Category Id | |
| AttractionPhoto | Image | Attraction Photo | |
| CityId | Id | City Id | |
| SupplierId | Id | Supplier Id | |

Attraction (Table):

| Name | Type | Description | Formula | Nullable |
|-----------------|-------|------------------|---------|----------|
| AttractionId | Id | Attraction Id | | No |
| AttractionName | Name | Attraction Name | | No |
| CountryId | Id | Country Id | | No |
| CountryName | Name | Country Name | | |
| CategoryId | Id | Category Id | | |
| 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 | | |

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

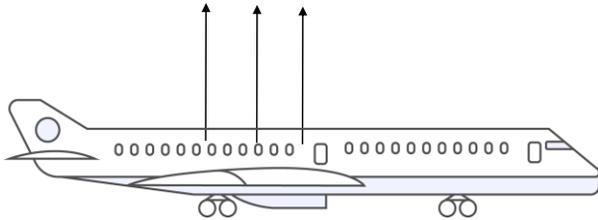
Si comparamos la estructura de la tabla ATTRACTION con la de la transacción Attraction, vemos que los atributos CountryName, CategoryName, CityName y SupplierName no se incluyen en la tabla ya que son atributos inferidos.



Como vimos antes, como están en la tabla extendida de la tabla ATTRACTION, su valor puede ser recuperado de las tablas donde están físicamente almacenados.

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

Seat



Flight

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

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.

Otra forma de modelar una relación 1 a N

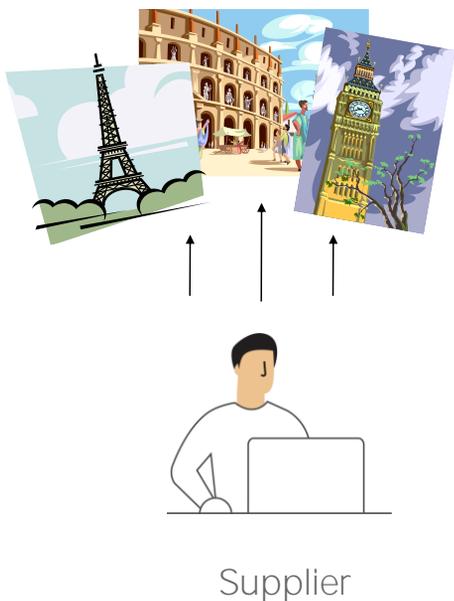
| Name | Type | Description | Formula | Nullable |
|----------------------------|--------------|-------------------------------|--------------------------------------|----------|
| Flight | Flight | Flight | | |
| FlightId | Id | Flight Id | | No |
| FlightDepartureAirportId | Id | Flight Departure Airport 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(FlightSeat.Location) | |
| Seat | Seat | Seat | | |
| FlightSeatId | Id | Flight Seat Id | | No |
| FlightSeatChar | SeatChar | Flight Seat Char | | No |
| FlightSeatLocation | Location | Flight Seat Location | | No |

Vamos a abrir la estructura de la transacción Flight para ver cómo representamos esta relación....

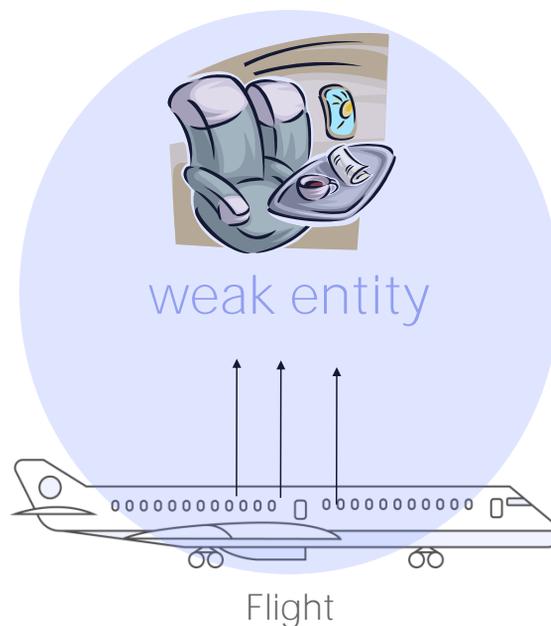
Vemos que en este caso el asiento (Seat) está como un segundo nivel en la transacción Flight.

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?

Attraction



Seat



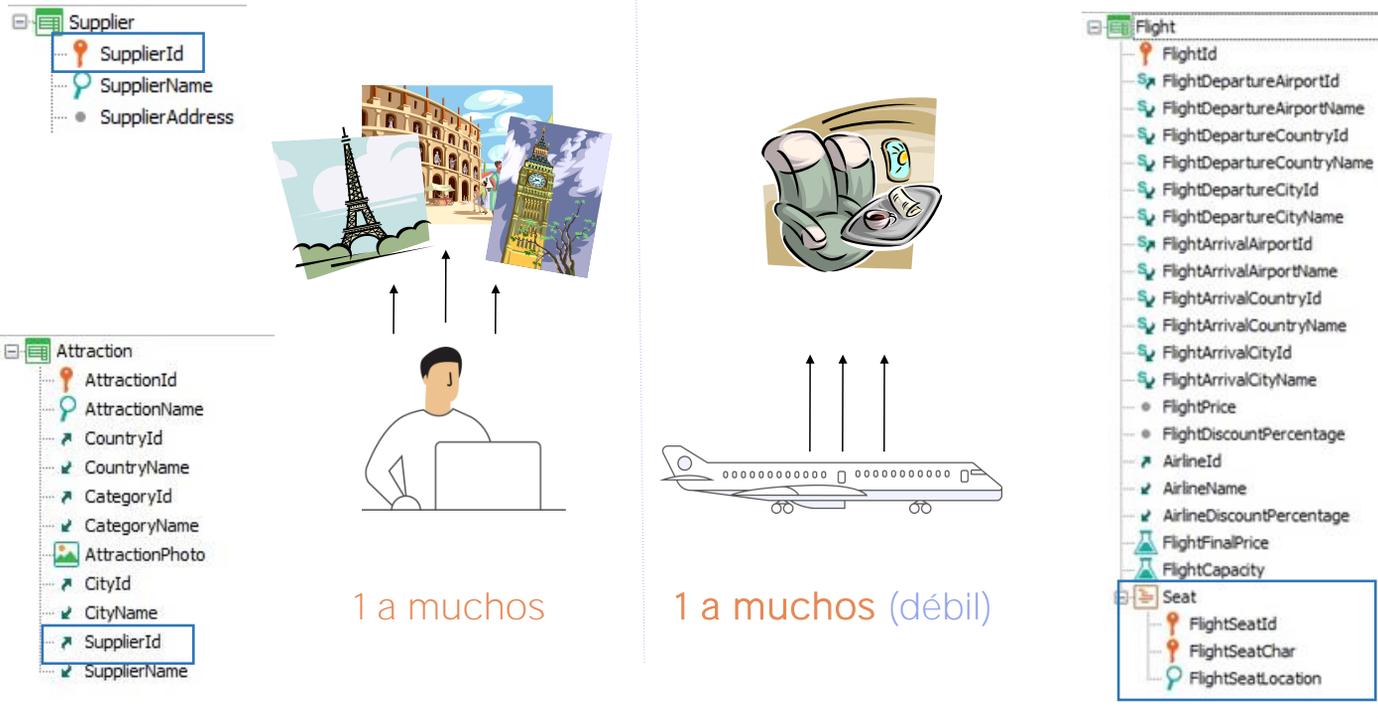
¿Por qué no representamos de la misma manera (con el mismo diseño de transacciones) ambos casos?

Notemos que los asientos no tiene sentido 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...

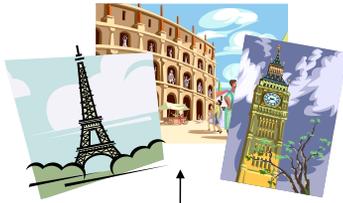
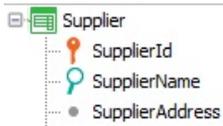
Por otro lado, una atracción podría no tener un proveedor que la ofrezca y sin embargo puede existir por sí misma como tal...

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.

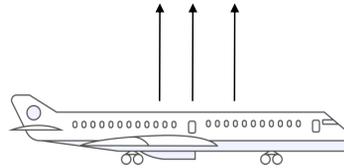
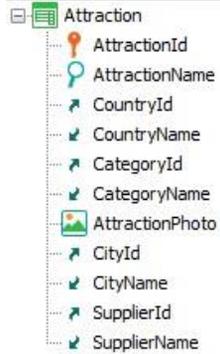
Una entidad como los asientos, que solamente tiene sentido que exista si se representa en función de otra entidad (en este caso los vuelos), decimos que es una **entidad débil**.



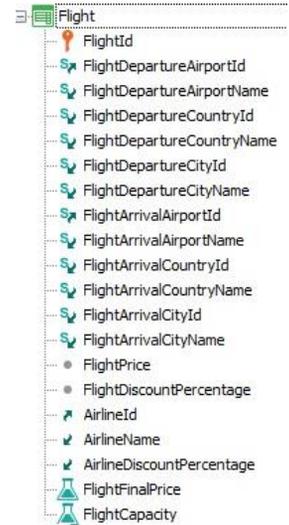
Este tipo de relación **1 a N débil** la solemos representar con una única transacción, de dos niveles, donde la entidad débil está en el segundo nivel. A diferencia de la relación 1 a N de Proveedores y Atracciones, donde creamos **dos** transacciones, y en una pusimos como clave foránea la clave primaria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



1 a muchos



1 a muchos (débil)



La relación 1 a N débil también puede representarse con dos transacciones (a los efectos del modelo de datos es exactamente lo mismo), donde la de los asientos tiene como parte de su clave primaria al atributo FlightId, que en particular será la llave foránea a la tabla Flight. Allí radica la diferencia entre entidad fuerte y débil. Observemos que por ser FlightId parte de la clave primaria, es imposible definir un asiento de vuelo, por ejemplo el 2 A Ventana, sin dar valor al vuelo, FlightId. En cambio es perfectamente posible ingresar aquí una atracción y no especificar su proveedor, si ese atributo tiene el valor Yes en su propiedad Nullable.

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

“Muchos ” a “muchos” (relación N a N)



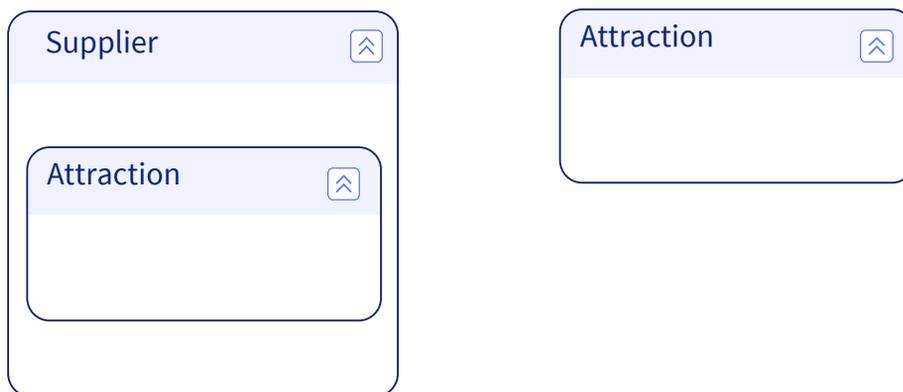
Supongamos, por ejemplo, que la agencia de viajes nos dice que **su realidad ha cambiado**.

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

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

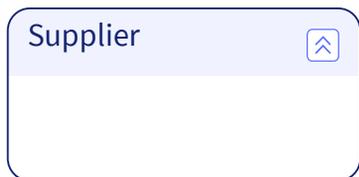
¿Cómo representamos esto en GeneXus?

Cómo representar una relación N a N



La solución es utilizando dos 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...

Cómo representar una relación N a N



o para c/atracción todos los proveedores que la proveen.

Modelando la relación N a N en GeneXus

| 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 | | |

| Name | Type | Description | Formula | Nullable |
|-----------------|------------------|------------------|---------|----------|
| Supplier | Supplier | Supplier | | |
| SupplierId | Id | Supplier Id | | No |
| SupplierName | Name | Supplier Name | | No |
| SupplierAddress | Address, GeneXus | Supplier Address | | No |
| Attraction | Attraction | Attraction | | |
| AttractionId | Id | Attraction Id | | No |
| AttractionName | Name | Attraction Name | | |
| AttractionPhoto | Image | Attraction Photo | | |

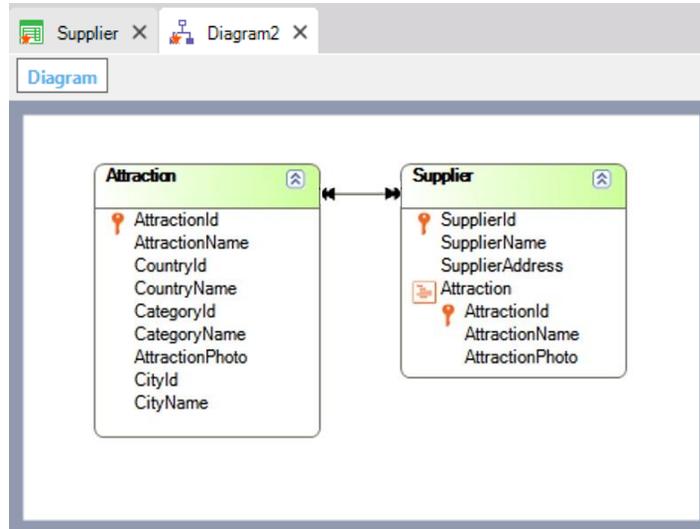
En nuestro caso la agencia nos pide sea cuando el usuario vaya a ingresar los datos de un proveedor, que ingrese todas las atracciones que éste provee.

Vamos a hacer esto en GeneXus...

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

Ahora abrimos la transacción Supplier, donde agregamos un segundo nivel y agregamos los atributos: AttractionId, (observemos que al escribir como atributo clave uno que empieza con "Attraction", automáticamente le cambió el nombre al nivel por Attraction), agregamos también Attraction Name y AttractionPhoto.

Un vistazo a la relación establecida entre proveedores y atracciones



Vamos a ver cómo nos quedó esta relación abriendo el diagrama que teníamos entre la transacción Attraction y Supplier.

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.

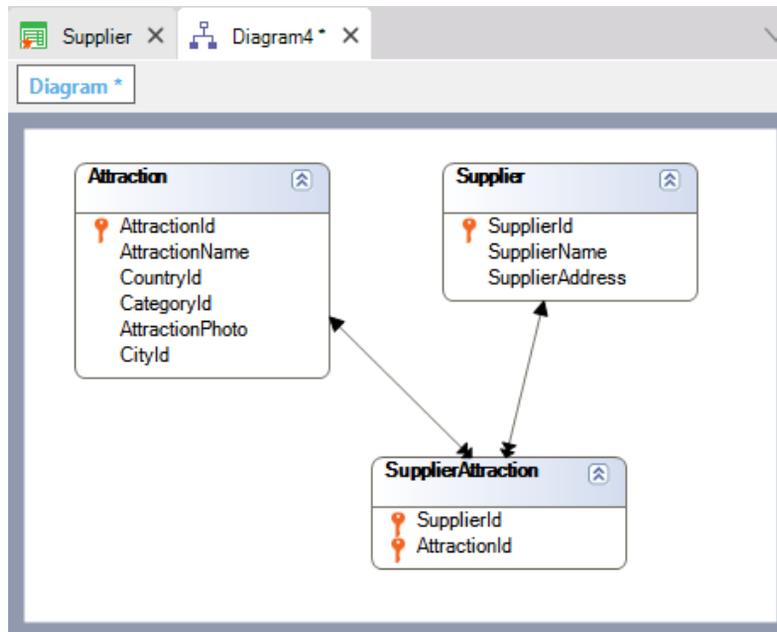
Tablas creadas por Genexus en base al diseño implementado



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

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

Relaciones entre tablas: N a N



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

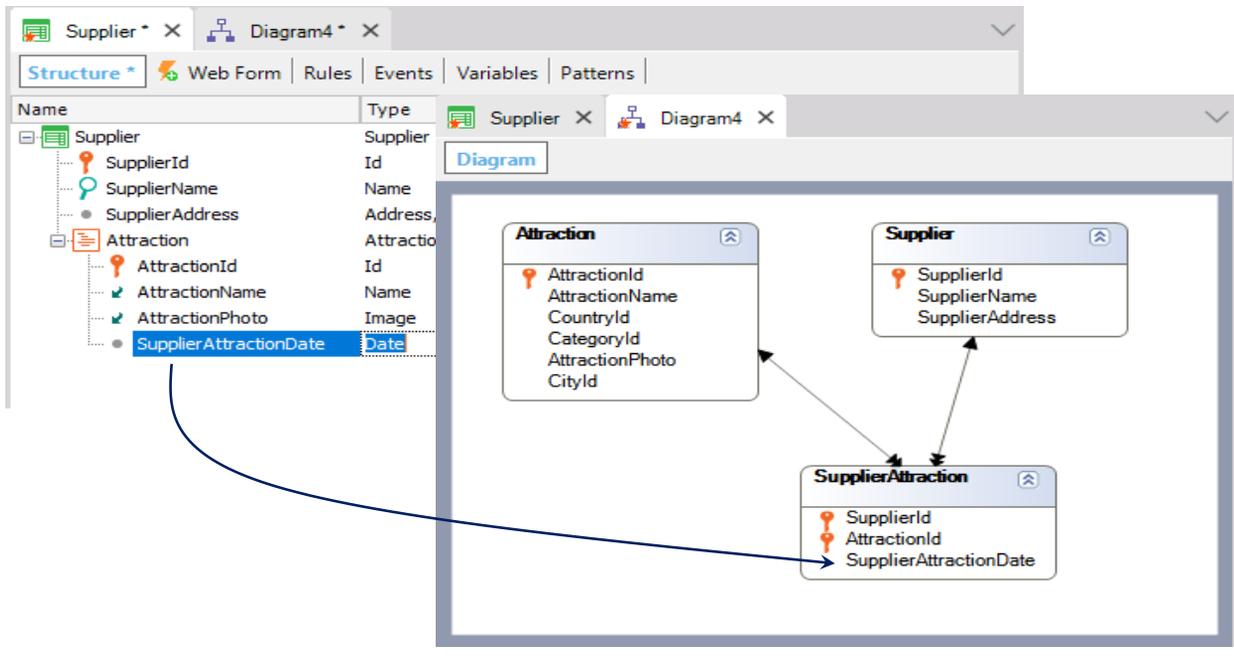
Notemos que en este caso, GeneXus crea una tabla por cada transacción que interviene en la relación muchos a muchos (ATTRACTION y SUPPLIER), pero además crea una tercera tabla llamada SUPPLIERATTRACTION, para establecer la relación.

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 tres tablas, una por cada entidad interviniente y una tercera con los identificadores de ambas. Esta tercera tabla podrá tener, eventualmente, atributos propios, como por ejemplo, la fecha en la que ese proveedor empezó a proveer esa atracción.

Ir a la estructura de Supplier y agregar en el 2do nivel, usando el punto para que aparezca automáticamente el prefijo, el att SupplierAttractionDate]

Relaciones entre tablas: N a N

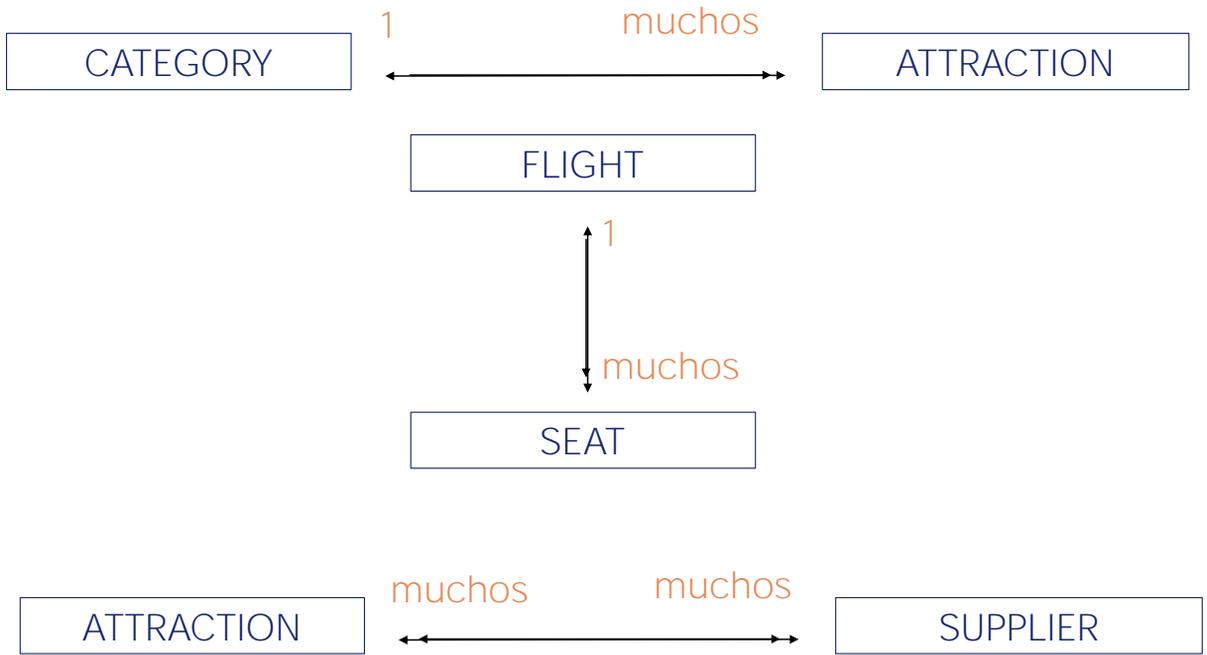


Si ahora volvemos a abrir el diagrama, vemos el atributo en la tabla relación.

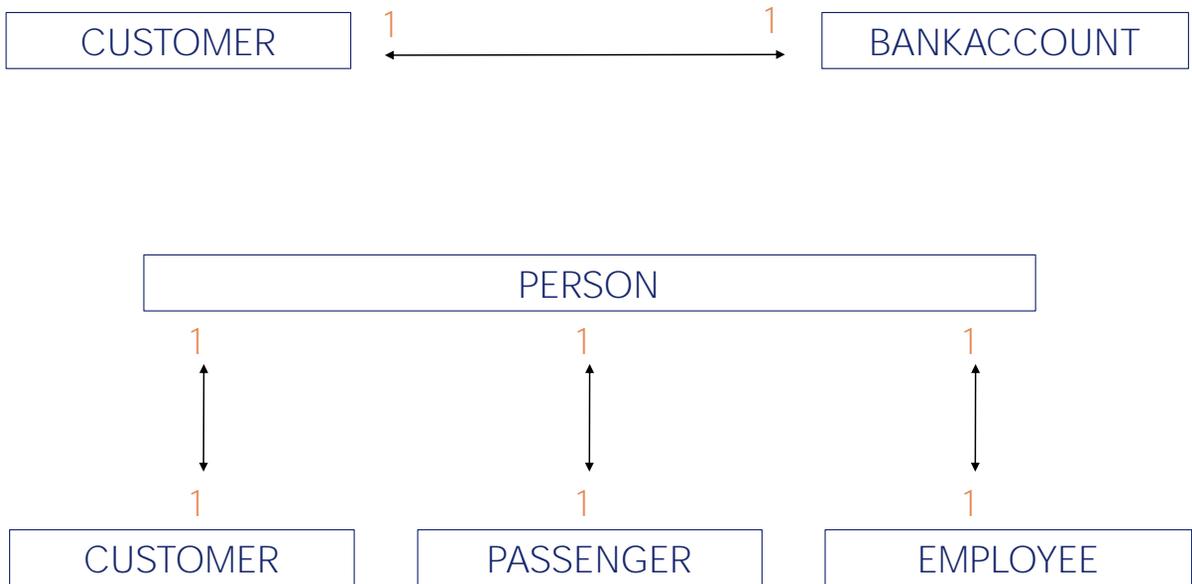
La relación de muchos a muchos entre *Attraction* y *Supplier*, se descompuso en 2 relaciones uno a muchos, utilizando la tabla *SUPPLIERATTRACTION* para establecer la relación entre las anteriores.

Para finalizar, actualicemos nuestra KB en GeneXus Server...

Y reorganicemos para que queden las tablas creadas...



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



Por ejemplo, cuando la agencia de viajes necesita asociar a cada cliente la cuenta bancaria que se le abre a los efectos de realizar los pagos de los servicios contratados.

Otro de los escenarios de relaciones 1 a 1 lo mencionamos cuando estudiamos los subtipos. Era el caso de la especialización: cuando una entidad **es** un caso particular de otra.

Pasemos, ahora, a nuestro siguiente tema.

*GeneXus*TM

training.genexus.com
wiki.genexus.com