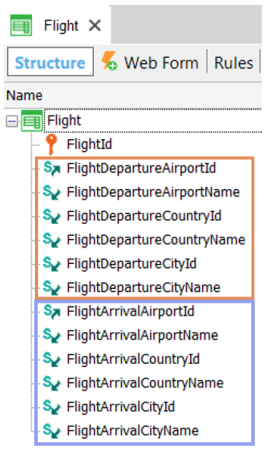
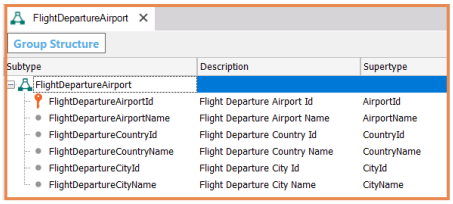
Subtipos

Referencia Múltiple y Especialización

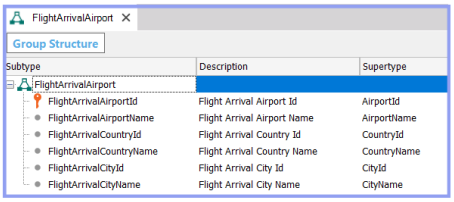
GeneXus™

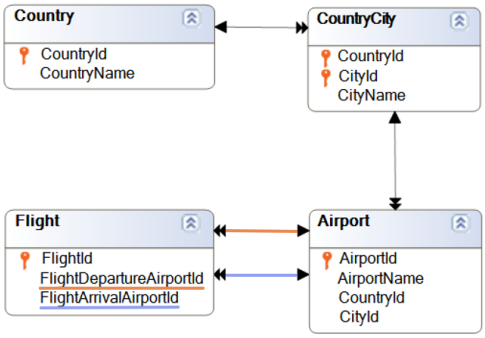


Subtype Group: FlightDepartureAirport



Subtype Group: FlightArrivalAirport



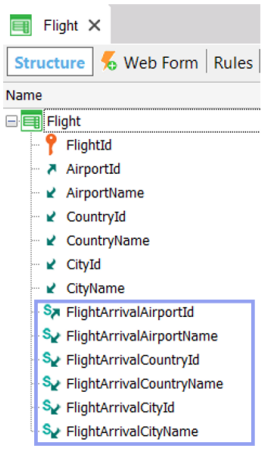


En el video anterior vimos la necesidad de definir grupos de subtipos, ya que teníamos en una transacción una doble referencia a un mismo actor de la realidad.

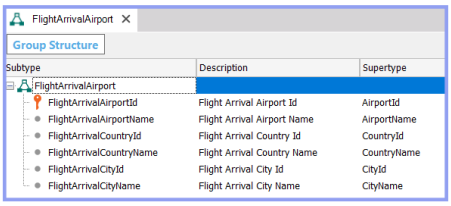
Era el caso de la transacción Flight, en la cual teníamos para cada vuelo un aeropuerto de partida y un aeropuerto de llegada. No podíamos incluir en la estructura de la transacción dos veces el mismo atributo, AirportId, ya que GeneXus nos daba un error por agregar un nombre de atributo duplicado. Por esa razón, decidimos definir dos grupos de subtipos: "FlightDepartureAirport", para identificar al aeropuerto de partida, y "FlightArrivalAirport", para identificar al aeropuerto de llegada.

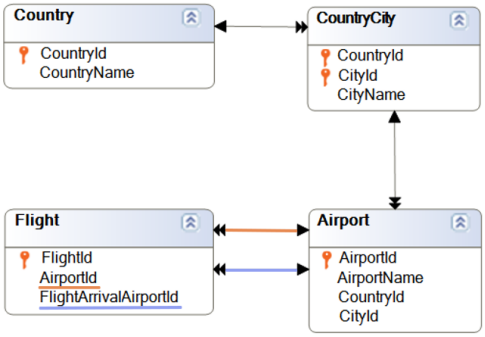
Como queríamos inferir el país y ciudad de cada aeropuerto, definimos en cada grupo subtipos también de los atributos correspondientes a país y ciudad, así como también al nombre del aeropuerto.

Por lo tanto, cuando en la transacción Flight nombramos a FlightDepartureCountryName, sabemos que será un CountryName inferido a través del aeropuerto de partida: FlightDepartureAirportId. Estos subtipos se han definido dentro del mismo grupo y por lo tanto se ha establecido la asociación y relación entre ellos.
Y cuando en la transacción Flight nombramos a FlightArrivalCountryName, sabemos que será inferido a través del atributo FlightArrivalAirporId.
O sea que no hay ninguna ambigüedad. Tenemos dos caminos perfectamente diferenciados para llegar a Country desde Flight.

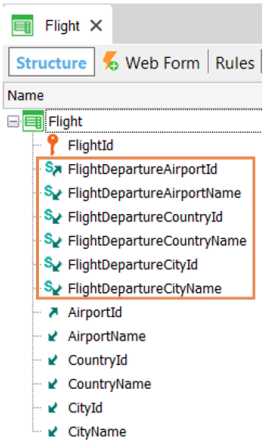


Defining only one subtype group: "FlightArrivalAirport"

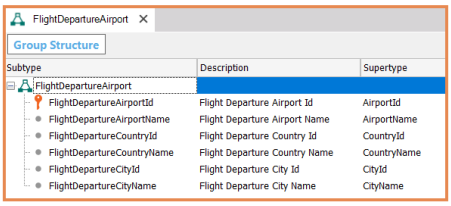


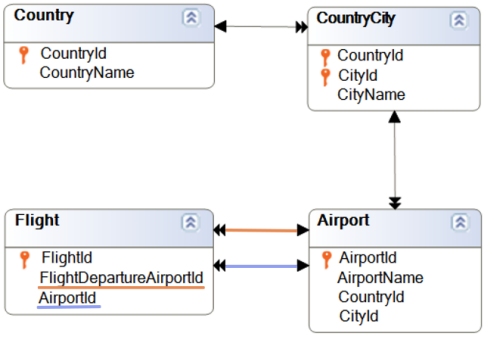


Otra opción era dejar el atributo de nombre AirportId para el rol del aeropuerto de partida y definir un grupo de subtipos para identificar al aeropuerto de llegada



Defining only one subtype group: "FlightDepartureAirport"

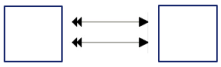




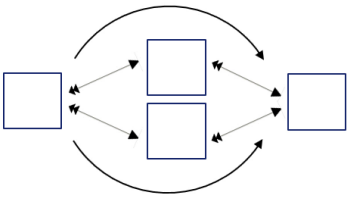
o de lo contrario, dejar el atributo de nombre AirportId para el rol del aeropuerto de llegada y definir un grupo de subtipos para identificar al aeropuerto de partida.
En ambas opciones, el modelo de datos reflejará las mismas relaciones que en la solución anterior.

MULTIPLE REFERENCES

Direct

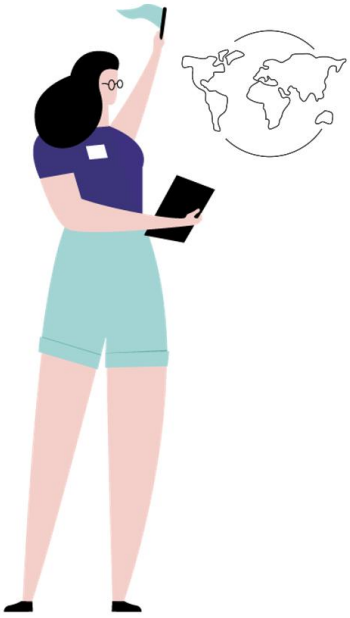


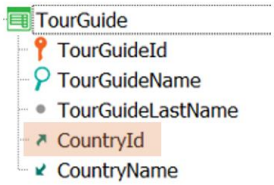
Indirect



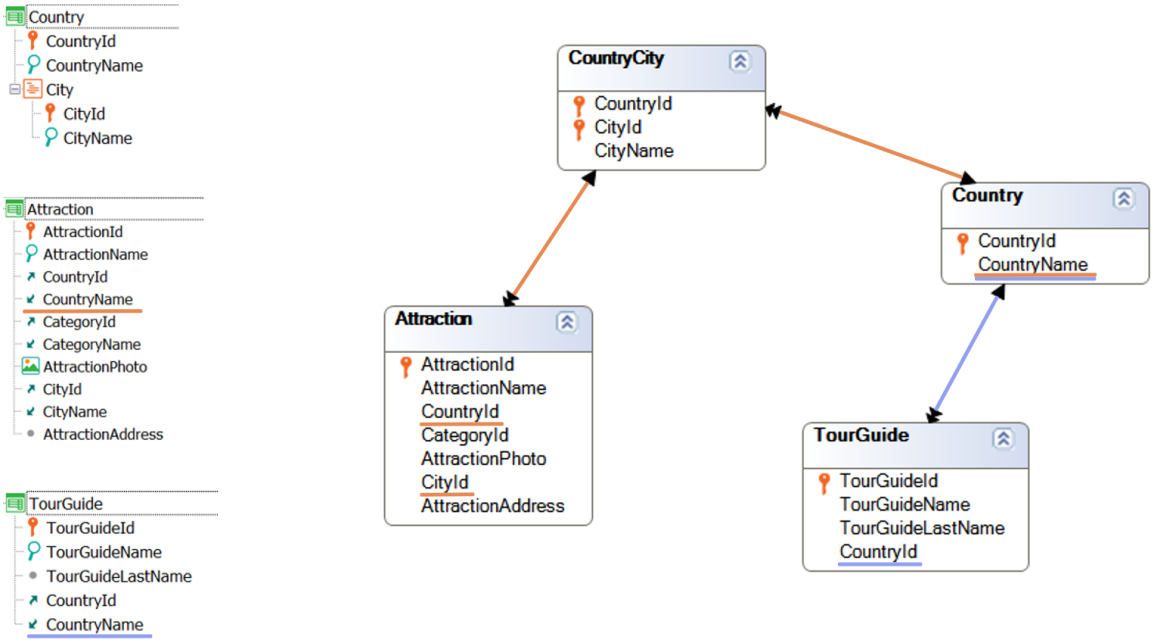
En el video anterior vimos entonces el caso de referencias múltiples de una tabla a otra relacionada directamente con ella.
Pero estas referencias no tienen por qué ser directas.

Desde una tabla podemos tener dos caminos para llegar a otra, tratándose de una relación indirecta, por lo que se necesitarán subtipos para diferenciarlos.

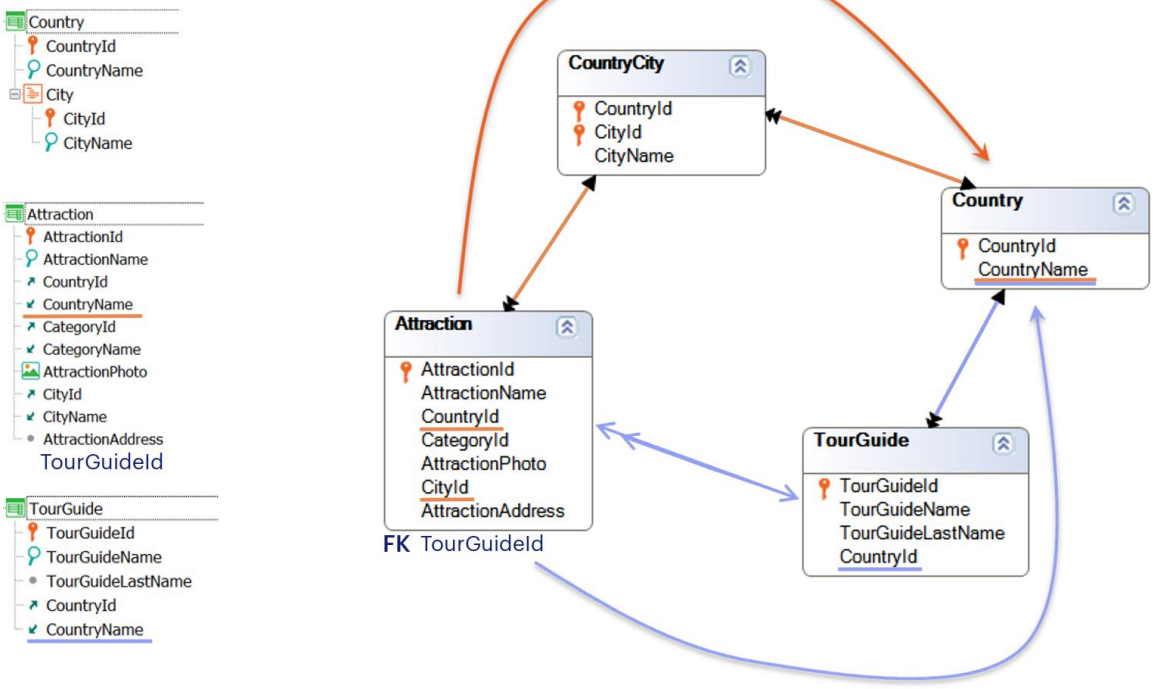




Supongamos que debemos agregar una transacción para registrar la información de los guías turísticos. Cada guía tiene una nacionalidad determinada, y por ese motivo hemos agregado a la estructura de su transacción el atributo CountryId.



Si observamos el diagrama de tablas, desde la transacción Attraction podemos inferir el CountryName correspondiente a esa atracción, ya que se encuentra en su tabla extendida. Análogamente, desde TourGuide también podemos inferir su propio CountryName, es decir, el país del guía.



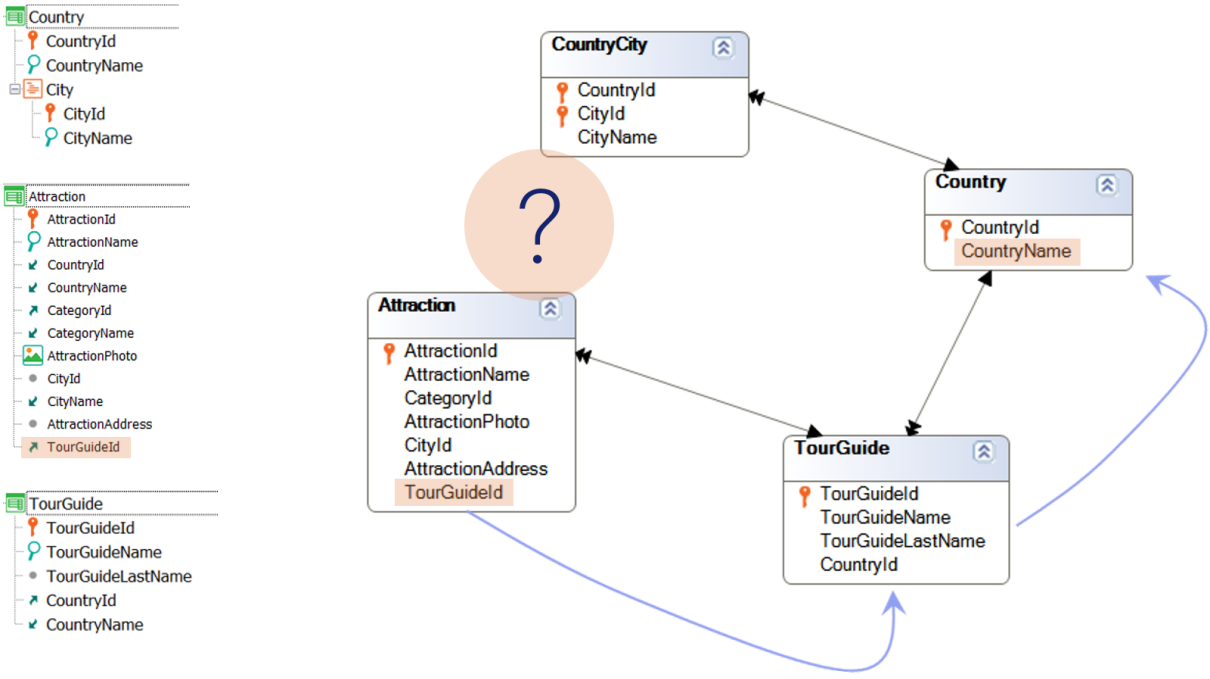
Si ahora vinculamos las entidades Attraction y TourGuide agregando a la primera el atributo identificador del guía, TourGuideId para representar que una atracción turística tiene un guía asignado y sólo uno, ¿cómo se relacionan ahora las tablas?

TourGuideId será una llave foránea en Attraction a la tabla TourGuide, por lo que aparecerá la relación marcada con la flecha celeste. Eso nos puede conducir a pensar que desde Attraction ahora existen dos caminos para inferir CountryName, lo que significa que hay una ambigüedad.

Dicho de otro modo, dado que GeneXus tiene el atributo CountryName en Attraction, ¿de dónde lo infiere?: ¿de la ciudad de la atracción o del guía turístico de la atracción? Si ambos valores coincidieran, no importaría, pero en este caso no tienen por qué coincidir. El país donde se encuentra la atracción no tiene por qué coincidir con el país natal del guía turístico.

Para poder diferenciar ambos roles de CountryName, necesitaremos utilizar subtipos.

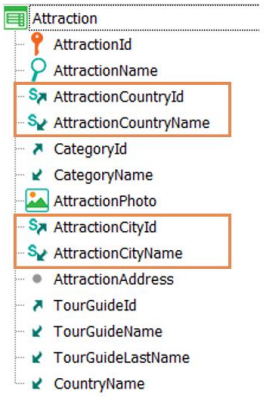
Pero en este caso no sólo los necesitaremos para diferenciar los roles.

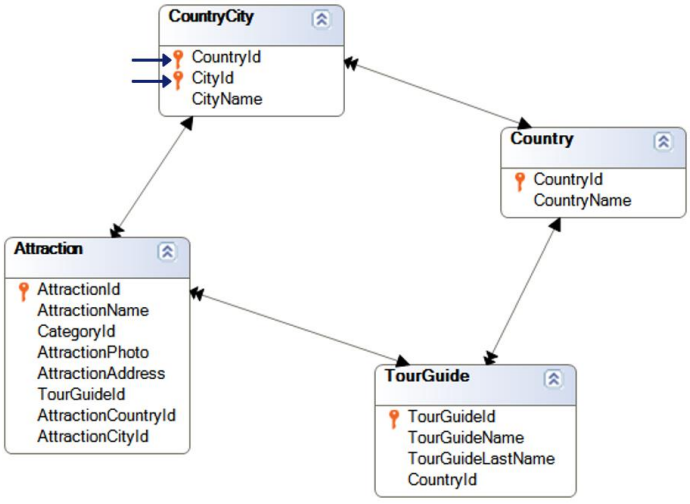


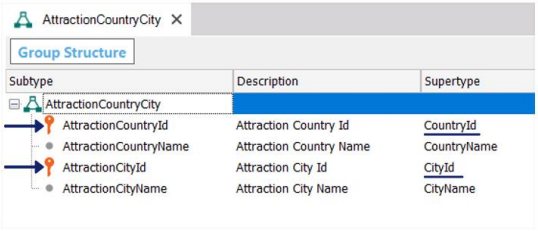
Observemos el diagrama de tablas después de agregar el atributo TourGuideId en la transacción Attraction: vemos que desaparece la relación entre Attraction y CountryCity, pues CountryId ya no es una llave foránea en Attraction. Observemos que ya no está en la tabla, es decir, será un atributo inferido. Pero, ¿inferido a partir de quién? ¡De TourGuideId!

No olvidemos que antes que nada GeneXus normaliza las tablas. Es decir, en base a los nombres de los atributos, en conjunto con los identificadores, determina qué atributo se coloca en cada tabla y las relaciones entre ellas. Como en Attraction aparece TourGuideId, que es identificador de TourGuide, y a la vez en TourGuide aparece CountryId, que es identificador de Country, está entendiendo que dado un TourGuideId en Attraction el CountryName lo infiere a partir de él, pasando por la tabla intermedia TourGuide.

Por lo tanto con este diseño de transacciones **hemos perdido** la posibilidad de indicar cuál es **el país de la atracción**, pues el CountryName será el del guía turístico.
No tenemos otra alternativa que utilizar subtipos para poder lograr que Attraction tenga un país propio, independiente del país del guía turístico.







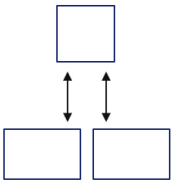
Crearemos un grupo de subtipos para representar el país y ciudad de la atracción, ya que es en Attraction donde se produce el problema.
Observemos que ahora sí GeneXus representa correctamente las relaciones en el diagrama de tablas, y además el atributo CountryName ahora se infiere a partir de TourGuideId sin ambigüedad. El atributo que representa y en el que se infiere el país de la atracción será el de nombre AttractionCountryName, subtipo de CountryName, perteneciente al grupo AttractionCountryCity.

También observemos que este grupo tiene dos atributos primarios: AttractionCountryId y AttractionCityId, que corresponden a la llave primaria de la tabla CountryCity, por los supertipos que indicamos: {CountryId, CityId}.

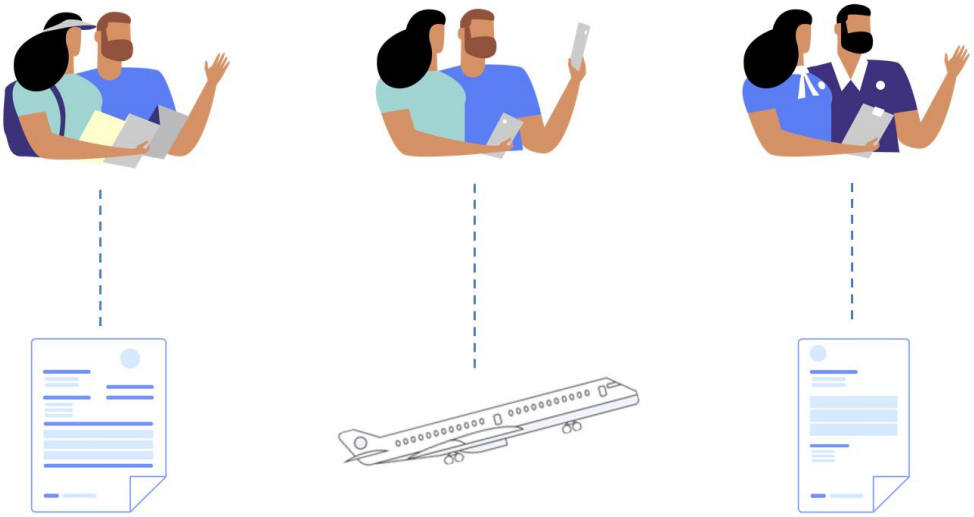
Como vimos esta solución resuelve el problema, pero debemos tener en cuenta que si ya existen otros objetos que para consultar el país de la atracción ya utilizaran los atributos originales CountryId y CountryName, es posible que tengamos que modificarlos en esos objetos y utilizar ahora los atributos subtipos AttractionCountrId y AttractionCountryName respectivamente.

Existen otras posibles soluciones, que no estudiaremos en este curso.

SPECIALIZATION

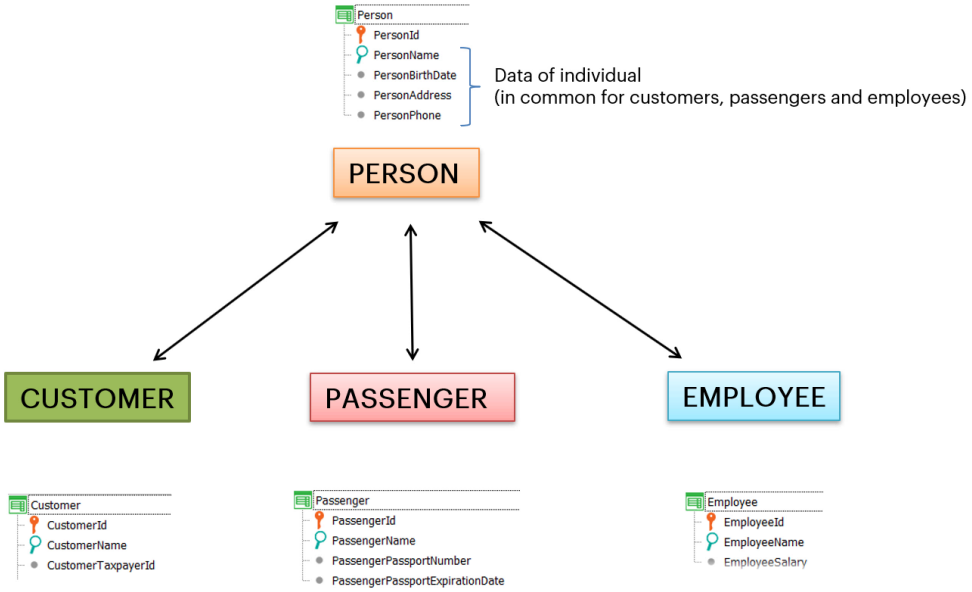


Veamos ahora otro caso de uso de subtipos, al que llamamos especialización, en el que tenemos una transacción que registra información general, y luego tenemos transacciones con información particular, que son una especialización de esa otra.



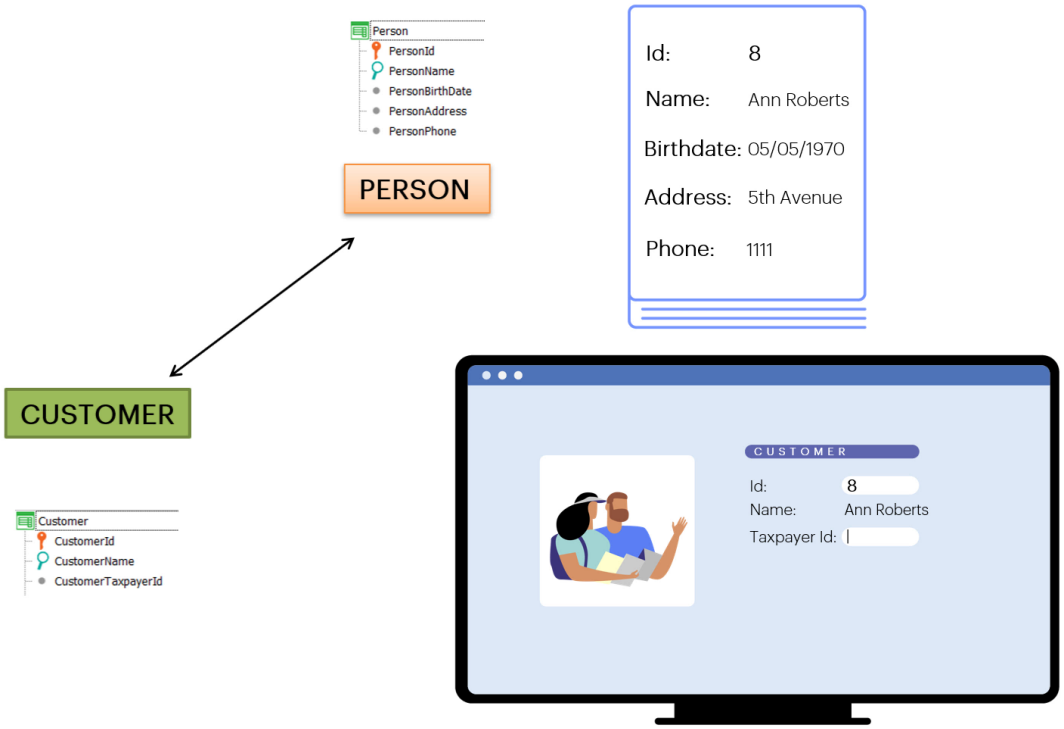
Supongamos que la agencia de viajes necesita manejar información específica de los clientes a los que les vende pasajes y paquetes turísticos (por ejemplo su número de contribuyente en la oficina estatal de impuestos, si lo tiene), información específica de los pasajeros (por ejemplo, su número de pasaporte y vigencia del mismo) y también información específica de los empleados de la agencia, para los que deberá registrar, por ejemplo, su salario.

Es decir, la agencia de viajes hará facturas a los clientes, registrará en los asientos de los vuelos a pasajeros y realizará recibos de sueldo a empleados.

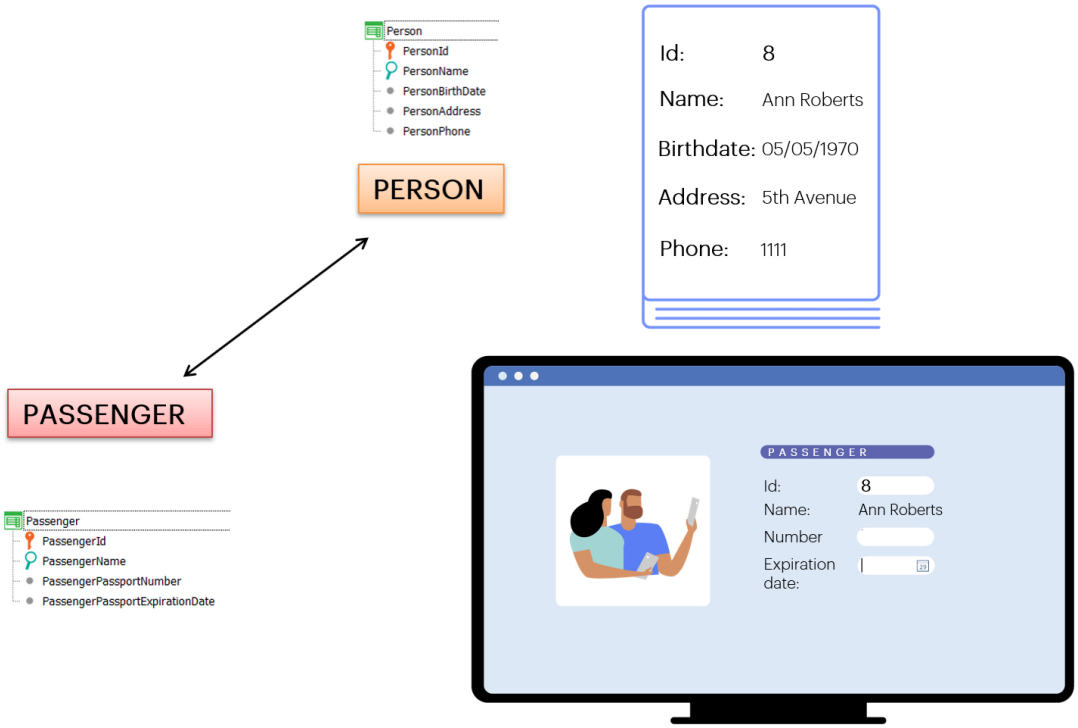


Podríamos definir entonces, en vez de lo que antes era solamente una transacción, Customer, una transacción llamada Person que maneje la información que es común para todas las personas (por ejemplo, un nombre, una fecha de nacimiento, una dirección, un teléfono, etc), y transacciones que sean especializaciones de Person, ya que tanto los clientes, como los pasajeros, como los empleados, SON personas.

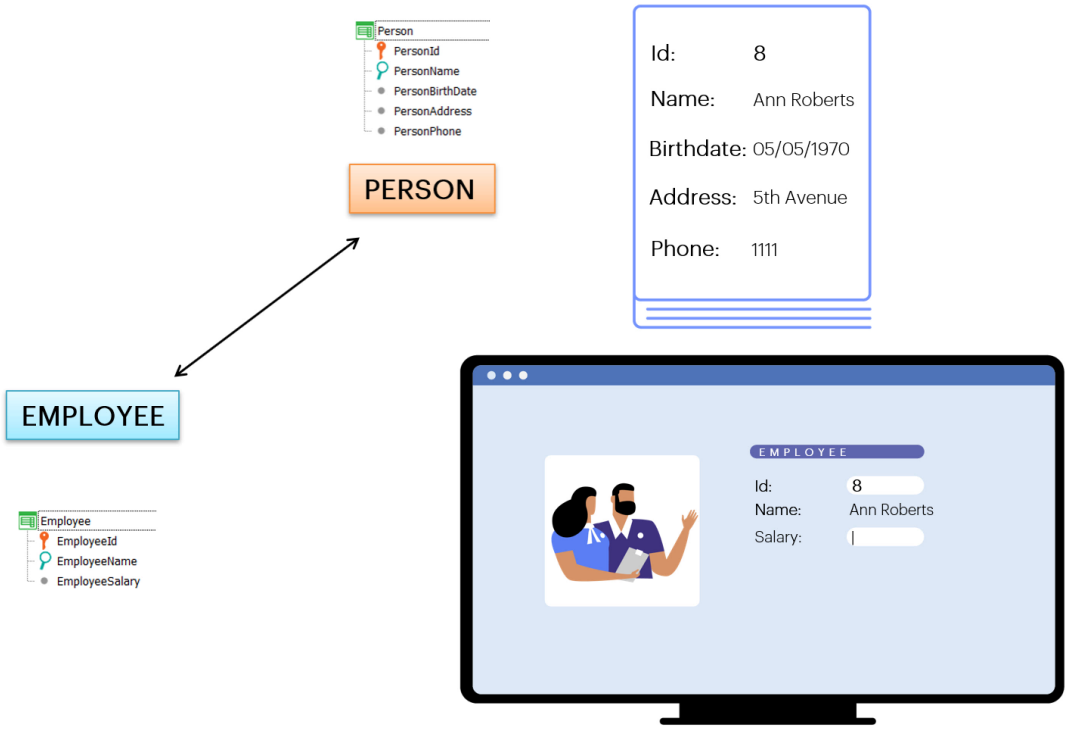
Cada especialización tendrá sus datos específicos (Customer tendrá el número de contribuyente, Passenger el número de pasaporte y fecha de expiración del mismo y Employee el salario).



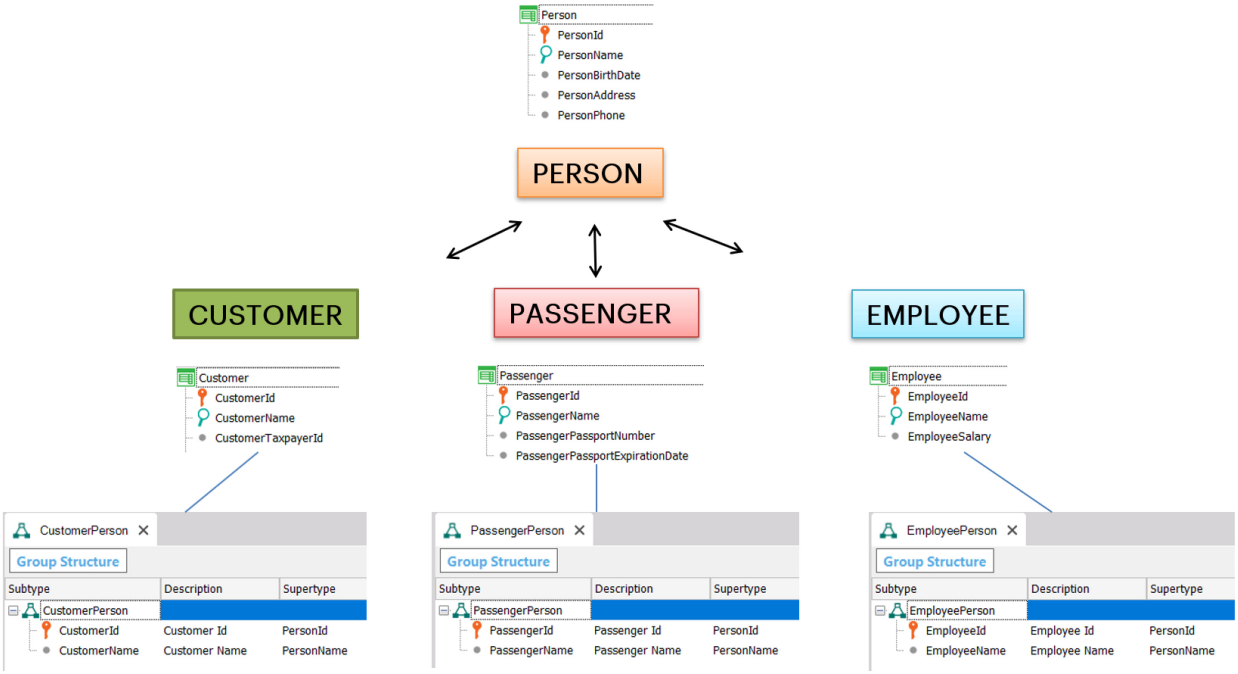
Queremos que el identificador de cliente **coincida exactamente** con el de una persona, para reflejar que el cliente **es** una persona. Es decir, si la persona de id 8 se llama Ann Roberts, y nació el 05/05/1970, cuando vamos a ingresar su información como cliente, necesitamos que el usuario pueda digitar en la transacción Customer el id 8, y que al salir del campo se le muestre el nombre Ann Roberts y pueda ingresar el número de contribuyente en el atributo CustomerTaxpayerId.



De la misma manera, si se ejecuta la transacción Passenger, deseamos que cuando el usuario digite en PassengerId el valor 8, en PassengerName se infiera Ann Roberts, y el usuario pueda asignar el número de pasaporte y la fecha de expiración del mismo en los atributos propios (PassengerPassportNumber y PassengerPassportExpirationDate).



Análogamente con los empleados.



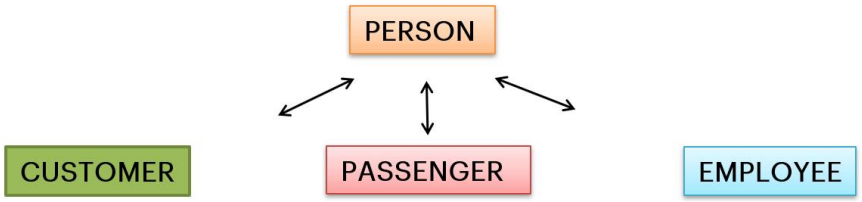
Si simplemente definimos como claves primarias de Customer, Passenger y Employee, los atributos CustomerId, PassengerId y EmployeeId respectivamente, sin relacionarlos de ningún modo con PersonId (lo mismo que CustomerName, PassengerName y EmployeeName sin relacionarlos con PersonName), no conseguiremos lo que buscamos. Para GeneXus serán transacciones completamente independientes.

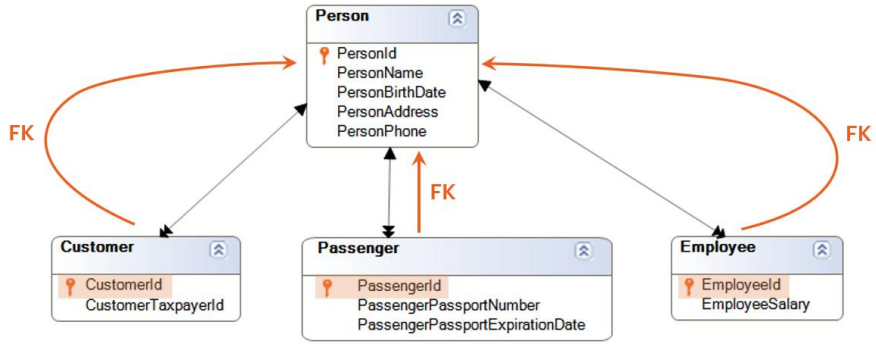
Para relacionarlas vamos a definir grupos de subtipos, para representar que tanto los clientes, como los pasajeros y los empleados, deben ser personas válidas (o sea, previamente registradas).

Creamos un grupo CustomerPerson, donde definimos a CustomerId y a CustomerName como subtipos de PersonId y PersonName, respectivamente.

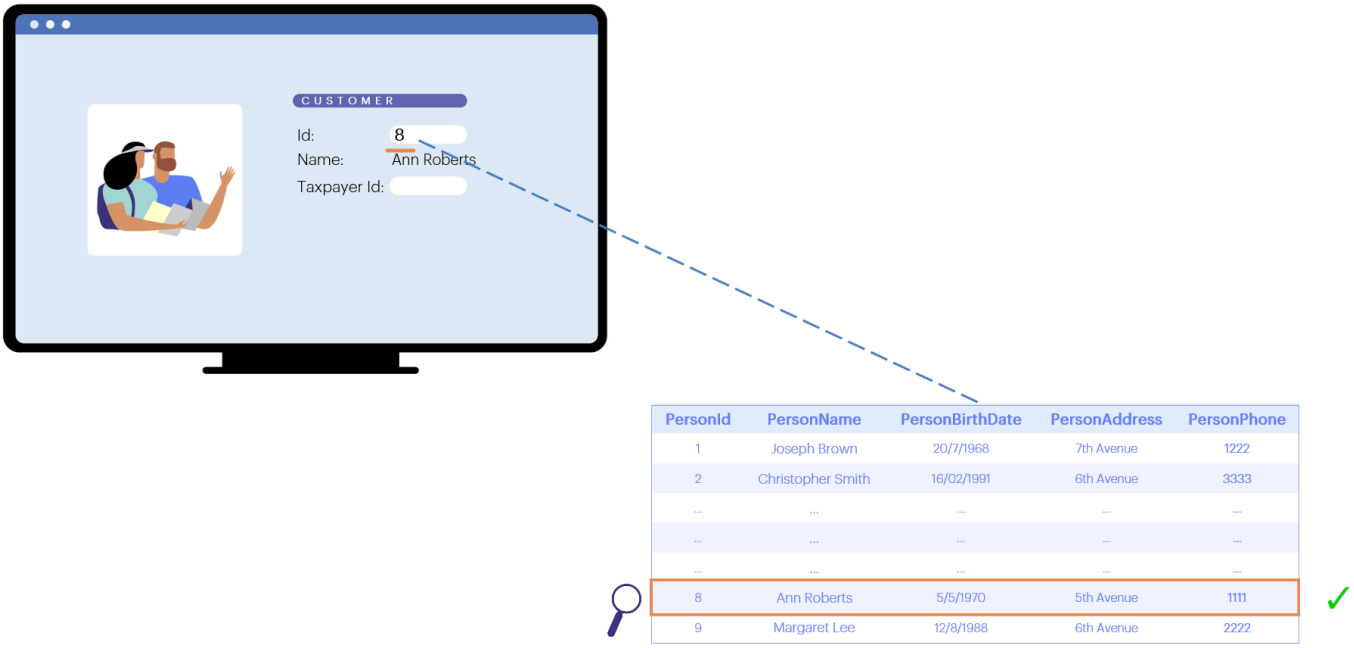
Otro de nombre PassengerPerson, donde definimos a PassengerId y a PassengerName como subtipos de PersonId y PersonName, respectivamente.

Y por último un grupo EmployeePerson, donde definimos a EmployeeId y EmployeeName como subtipos de PersonId y PersonName, respectivamente.

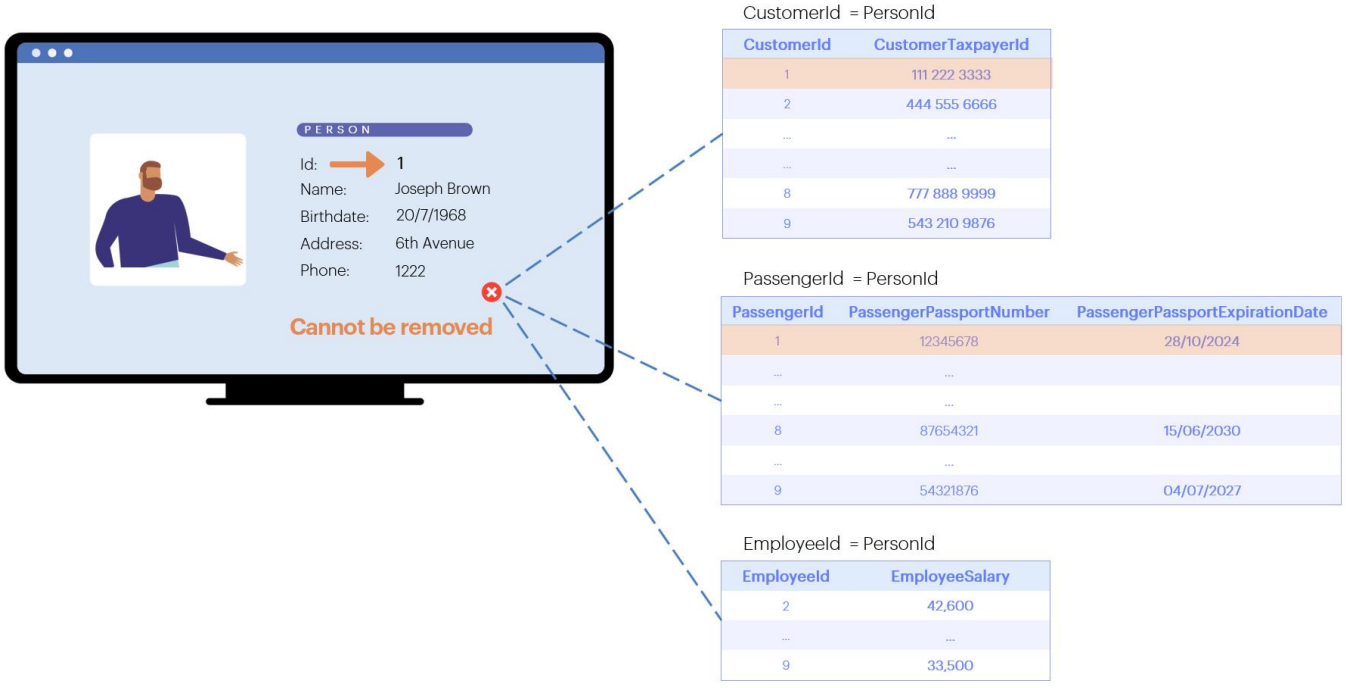




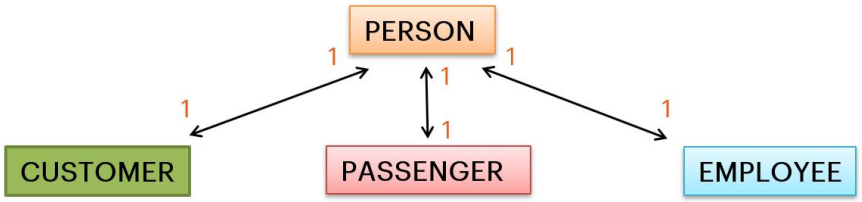
Al hacer esto, los atributos CustomerId, PassengerId y EmployeeId, además de ser los identificadores de las tablas Customer, Passenger y Employee respectivamente, y por tanto, sus claves primarias, serán, a la vez, claves foráneas a la tabla Person, por lo que GeneXus controlará la consistencia de los datos.

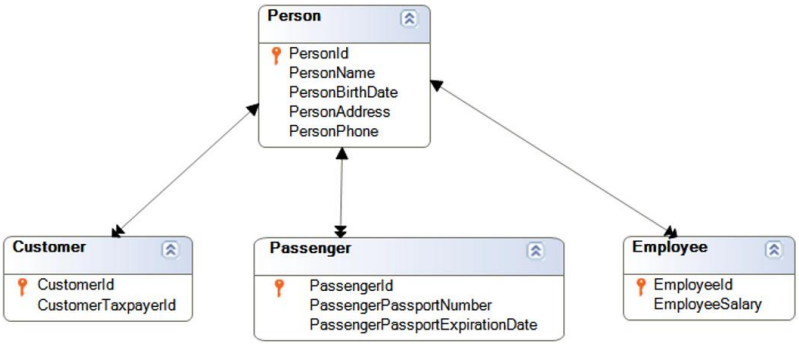


Esto quiere decir que cuando el usuario ingrese un valor en el id de cualquiera de las tres transacciones (Customer, Passenger o Employee), se irá a buscar a la tabla Person un registro que tenga como id ese mismo valor.



Del mismo modo, si se quiere eliminar una persona a través de la transacción Person, se controlará que no exista un registro en Customer donde CustomerId sea igual al PersonId que se está queriendo eliminar, ni un registro en Passenger donde PassengerId sea igual al PersonId a eliminar, ni en Employee donde EmployeeId sea igual al PersonId a eliminar. Si existiera alguno de esos tres registros, no se permitirá la eliminación de la persona.





Este diseño representa relaciones 1 a 1 entre la tabla general y la correspondiente a cada especialización, es decir:
una persona puede estar registrada una sola vez como cliente, porque CustomerId es un PersonId válido, y además es llave primaria. De igual manera una persona puede estar registrada una sola vez como pasajero y una sola vez como empleado.
Una persona puede tener los 3 roles, o estar registrado solamente como persona y no tener datos extra como cliente de la agencia, ni como empleado ni como pasajero.

En las transacciones CUSTOMER, PASSENGER y EMPLOYEE, CustomerName, PassengerName y EmployeeName serán atributos inferidos, por lo que no estarán físicamente en las tablas CUSTOMER, PASSENGER y EMPLOYEE respectivamente.

Recuerde que los diagramas que realiza GeneXus no muestra las relaciones 1 a 1, con las flechas solo indica las relaciones de claves foráneas. Es por ese motivo que vemos la flecha doble del lado de las tablas especializadas.

Existen otros casos de uso de subtipos que no estudiaremos en este curso. Si está interesado, puede buscar la información relacionada a este tema en el curso del siguiente nivel.

training.genexus.com
wiki.genexus.com
training.genexus.com/certifications