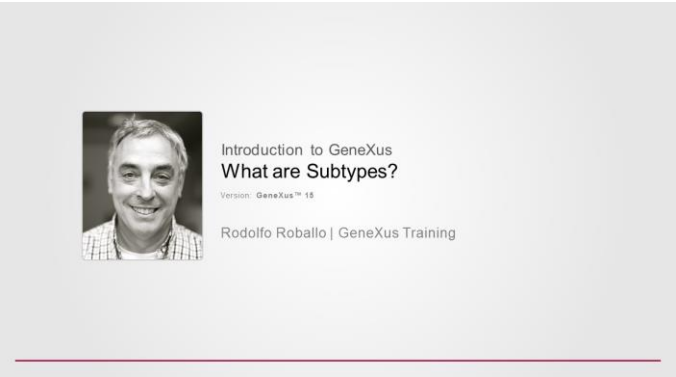
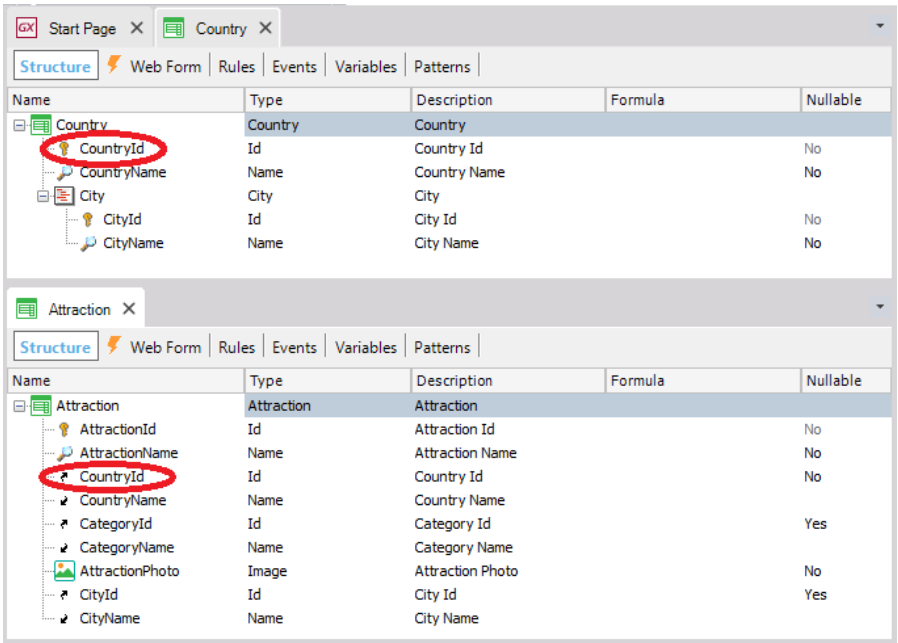
¿Qué son los subtipos?



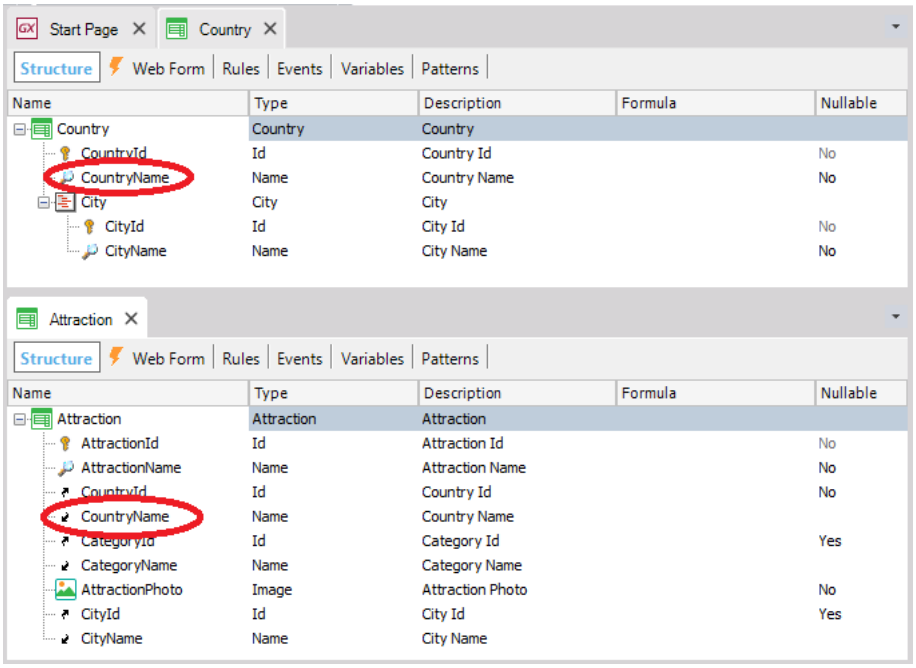
Hasta ahora hemos visto que GeneXus establece relaciones entre transacciones -y entre tablas- basándose en los nombres de atributos que encuentra iguales.

Por ejemplo, en la transacción Attraction se encuentra el atributo CountryId



con rol de llave foránea, dado que con igual nombre está presente en la transacción Country, y allí es llave primaria

Por su parte, el atributo CountryName también se encuentra en ambas transacciones con el mismo nombre:



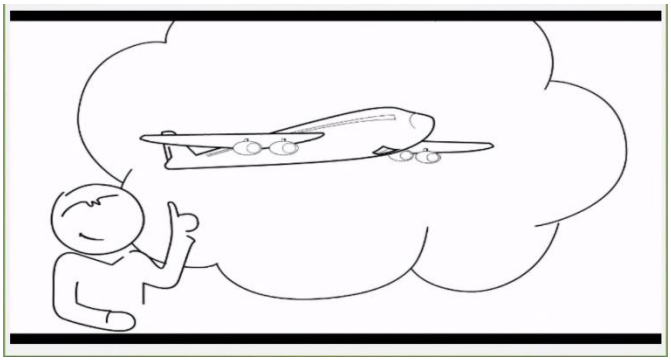
por lo que GeneXus interpreta que se trata del mismo atributo. En este caso no es un atributo primario, es decir, llave primaria o parte de una llave primaria en alguna tabla, así que GeneXus determinará que debe almacenarlo en la tabla COUNTRY y no en la tabla ATRACTION.

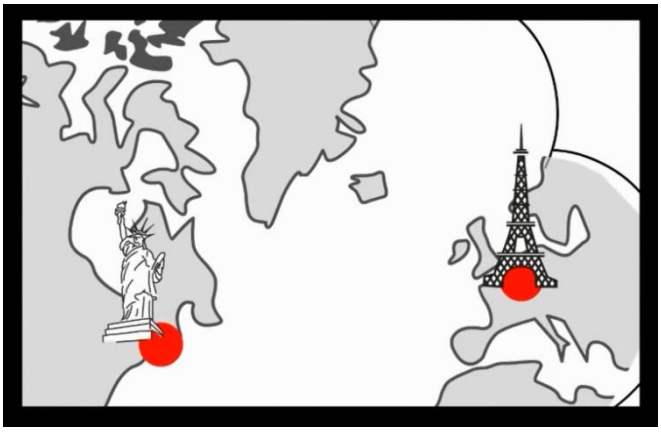
De modo que **GeneXus siempre asume que si usamos el mismo nombre de atributo, estamos representando al mismo concepto**.

Sin embargo, hay casos en los que podríamos necesitar usar nombres distintos para el mismo concepto, e indicarle a GeneXus que ambos nombres **significan lo mismo**.

Veamos esto.

Supongamos que en la agencia de viajes nos piden registrar los vuelos que ofrecen a los clientes para arribar a una atracción turística.



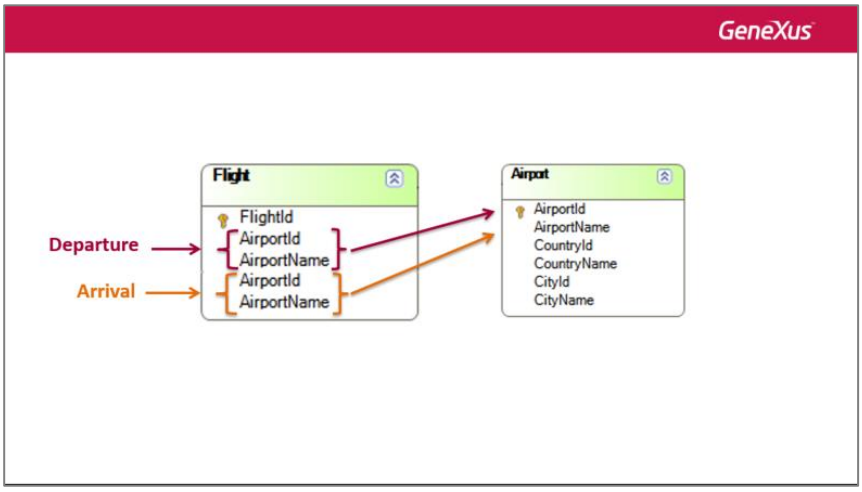


Y debemos registrar para cada vuelo, el aeropuerto desde donde parte, así como también el aeropuerto de llegada.

Para representar esto, vamos a crear en primer lugar una transacción de nombre: Flight

Definimos el atributo FlightId, que automáticamente queda basado en el dominio: Id.

Y ahora detengámonos a pensar **qué otra información debemos registrar**.

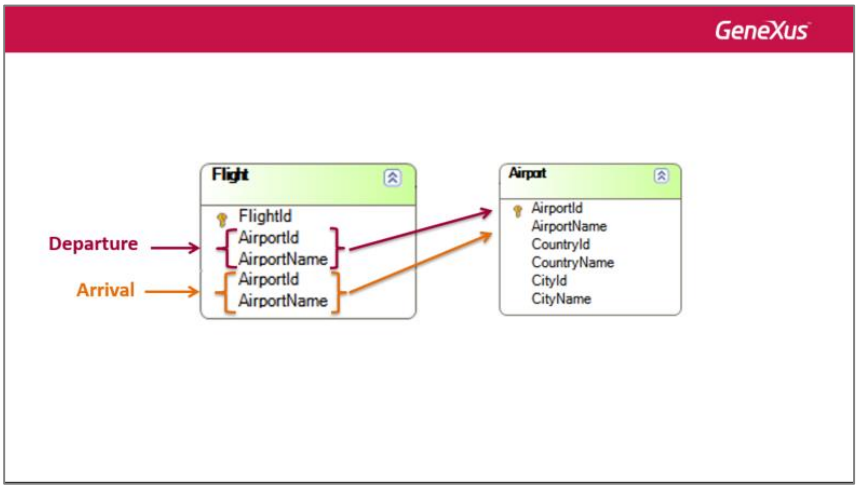


Cada vuelo como decíamos, tendrá un aeropuerto de partida y un aeropuerto de llegada…

Pero a los aeropuertos tendremos que poder registrarlos por sí mismos…., así luego podremos referenciarlos desde los vuelos.

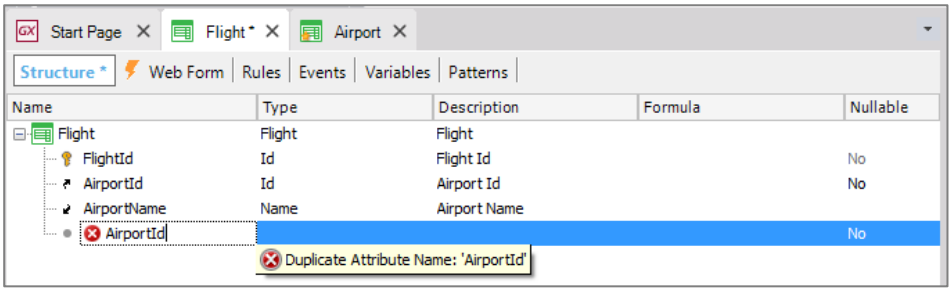
Así que dejemos de trabajar en la transacción Flight por un instante, y vamos a crear otra transacción de nombre Airport. Definimos entonces que cada aeropuerto tiene un identificador AirportId, un nombre AirportName y cada aeropuerto se encuentra en 1 país y en 1 ciudad, así que vamos a agregar los atributos: CountryId, CountryName, CityId y CityName. Salvamos...

Y ahora volvamos a ver cuál era nuestra necesidad en la transacción Flight.



Necesitamos agregar a cada vuelo, su aeropuerto de partida y su aeropuerto de llegada.

Así que volvemos a la transacción Flight, y vamos a agregar los atributos AirportId, y AirportName…. Pero cuando intentamos agregar nuevamente AirportId:



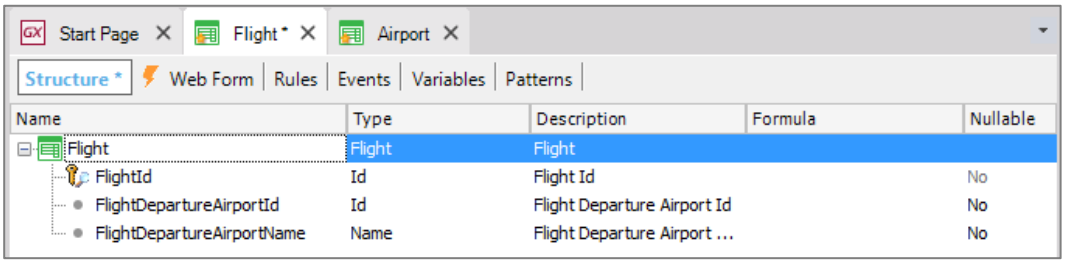
¡GeneXus nos dice que hay un error!, que estamos agregando un atributo con nombre duplicado.

Y lo mismo nos va a pasar con el atributo AirportName, que pensábamos agregar para representar el nombre del aeropuerto de llegada.

¿Cómo podemos hacer entonces para ingresar 2 aeropuertos en una misma transacción? Evidentemente vamos a tener que usar **nombres de atributos diferentes** para almacenar la información de origen y de destino del vuelo que queremos registrar.

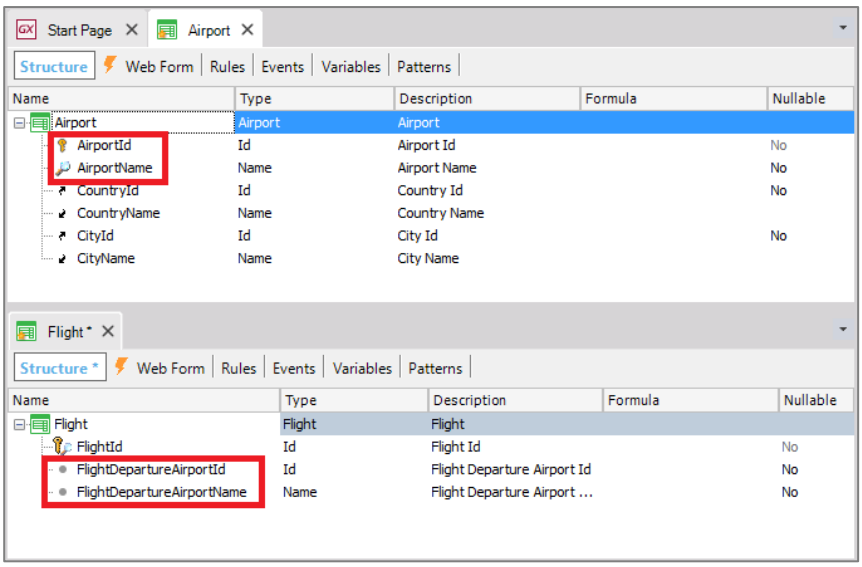
Vamos a borrar entonces los atributos que originalmente habíamos ingresado y vamos a definir atributos con nombres nuevos.

Vamos a llamar FlightDepartureAirportId al identificador del aeropuerto de origen del vuelo,
Y FlightDepartureAirportName al nombre del aeropuerto de origen.



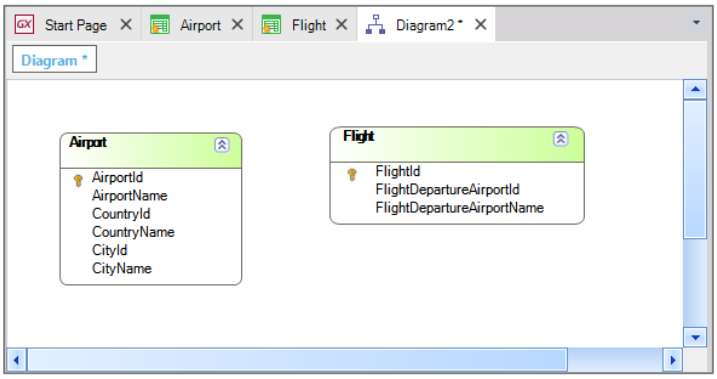
Bien, hemos definido nombres de atributos nuevos… pero para GeneXus estos nombres de atributos no tienen relación con AirportId ni con AirporName.

Tal como dijimos antes, si usamos nombres distintos en la transacción Flight y en la transacción Airport para identificar al concepto de aeropuerto, GeneXus no establecerá ninguna relación entre ambas transacciones.

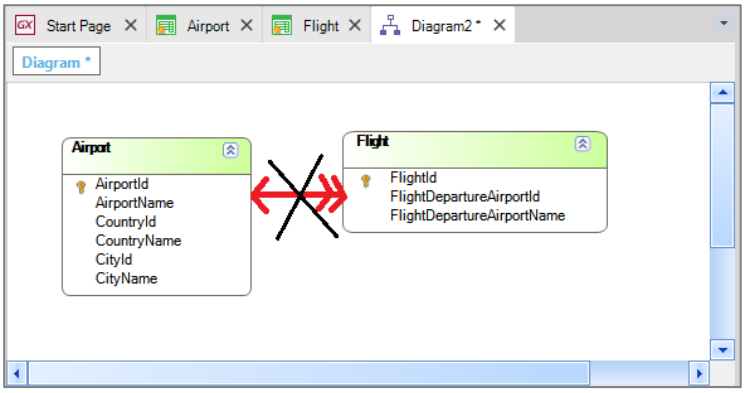


Para verificar esto que acabamos de decir, vamos a crear un diagrama de transacciones.

Y vamos a arrastrar las transacciones Airport y Flight y vemos que efectivamente, GeneXus no encuentra relación entre ellas, ya que no se identificó ninguna clave foránea en Flight que permita la relación con Airport.

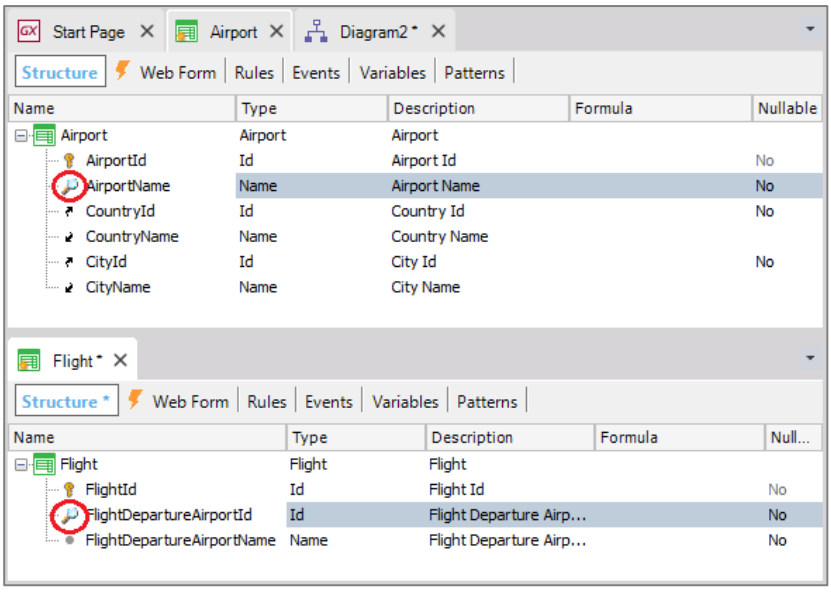


Si hubiera encontrado relación aparecería una flecha entre ambas transacciones”



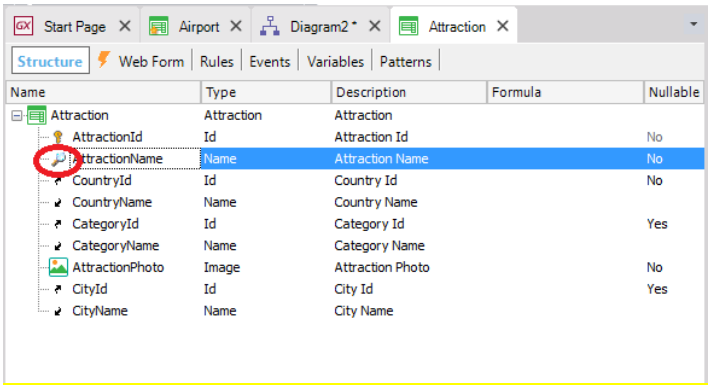
Otra forma de ver esto es prestar atención a la forma en que GeneXus nos muestra, en la transacción Flight, al atributo identificador del aeropuerto.

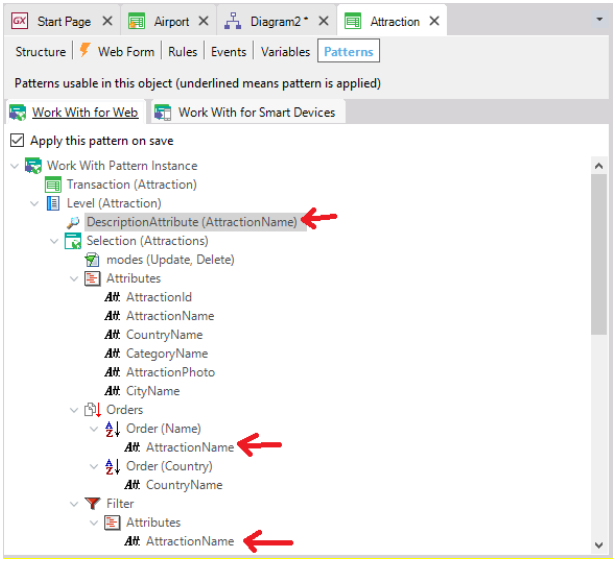
Vemos que está señalizado con este símbolo, que en la transacción Airport está también en AirportName.

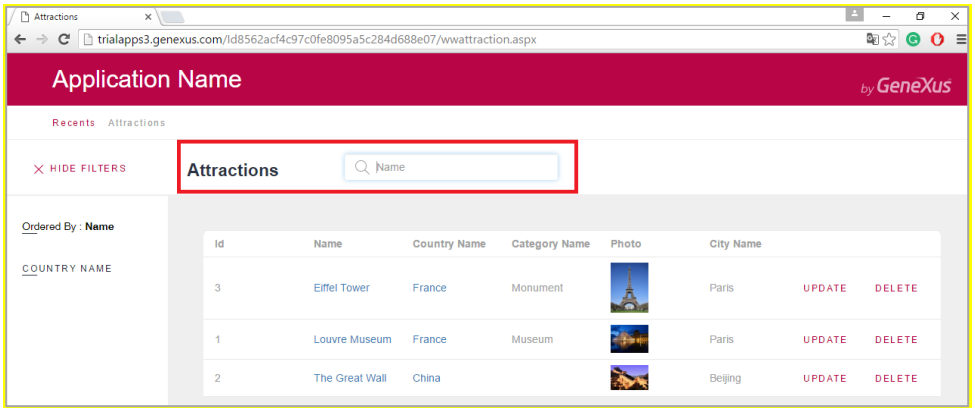


Este símbolo está indicando que el atributo es el que mejor describe al aeropuerto en un caso, o al vuelo en el otro. Cuando creamos la estructura de una transacción, GeneXus elige en base a los tipos de datos de sus atributos al que parece mejor describirla, pero el usuario puede cambiarlo por otro, o decidir que no haya ningún atributo de esa clase. Este "atributo descriptor" se utiliza por ejemplo por el pattern Work With para permitir filtrar por él, ordenar por él, etcétera.

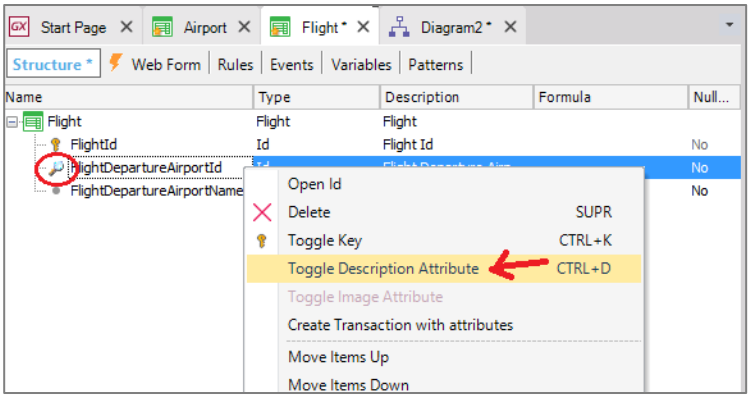
Recordemos el pattern aplicado a Attraction:





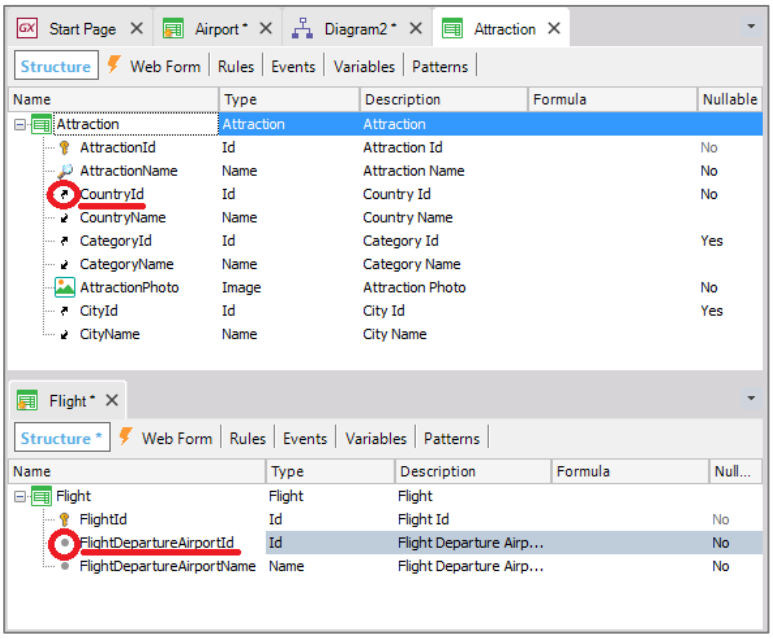


Para Flight no nos interesa que el aeropuerto de partida sea el atributo que mejor describa al vuelo, así que lo quitamos.



Vemos, así, que queda señalizado con este símbolo, que indica que es un atributo secundario y no es considerado como clave foránea…

Comparemos esto con la definición del identificador de país en la transacción Attraction



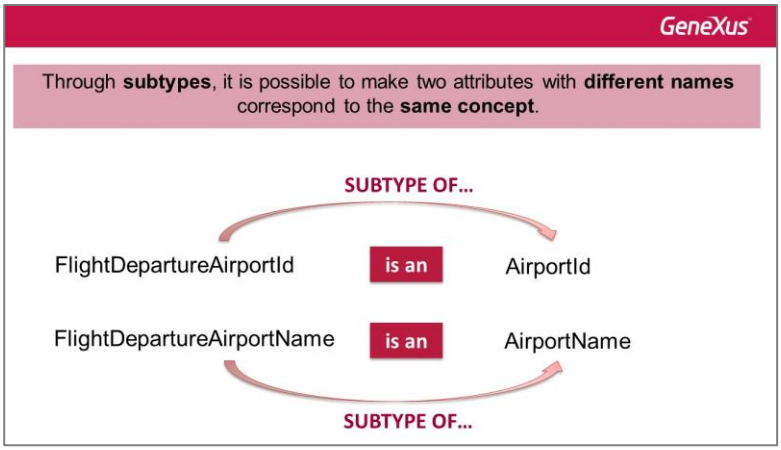
En Attraction, el atributo CountryId tiene este ícono que nos indica que es un atributo clave foránea… pero no es el caso del atributo FlightDepartureAirportId en la transacción Flight.

Entonces, **¿cómo hacemos para que GeneXus pueda asociar distintos nombres a un mismo concepto?**

Necesitamos que FlightDepartureAirportId aunque se llame distinto que AirportId, sea considerado como tal, o sea, **como un identificador de aeropuerto**.

Y lo mismo ocurre con el nombre del aeropuerto.

¿Cómo lo podremos lograr?



La respuesta es: **mediante la definición de subtipos**.

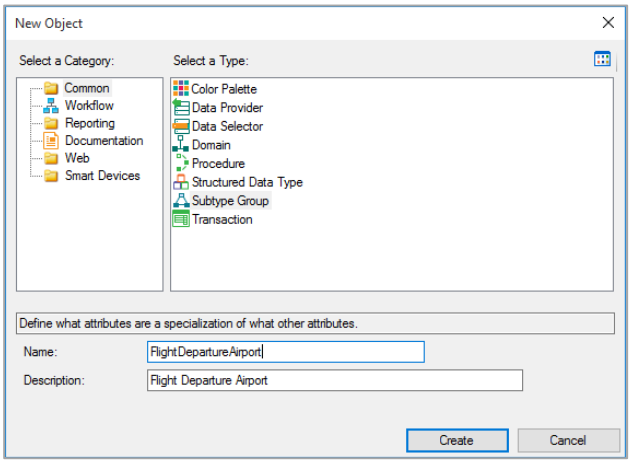
Cuando un atributo se llama distinto a otro ya definido, pero ambos representan el mismo concepto, **podemos "decirle" a GeneXus** que el nuevo atributo es subtipo del otro

y a partir de ese momento GeneXus los considerará exactamente como si fueran la misma cosa… por lo tanto, GeneXus tratará al atributo FlightDepartureAirportId **exactamente como si fuera un AirportId, es decir lo identificará como clave foránea en la transacción Flight**.

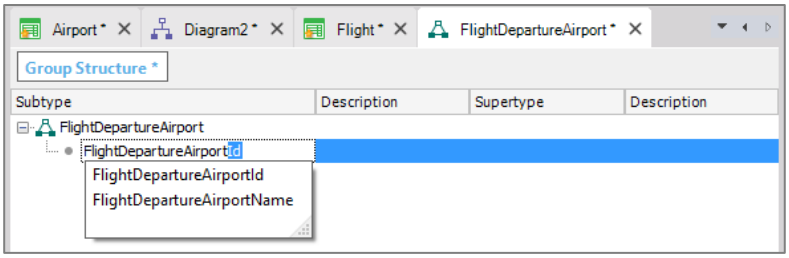
Y lo mismo haremos con FlightDepartureAirportName: indicaremos que es subtipo de AirportName.

Veamos esto en la práctica.
Para definir subtipos, lo primero que debemos hacer es crear un grupo de subtipos.

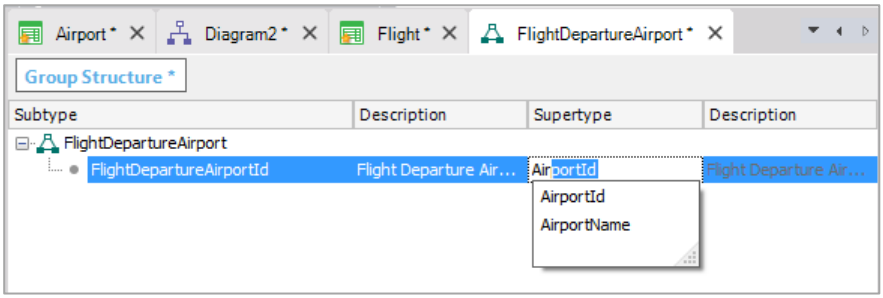
Así que creamos un nuevo objeto de tipo Subtype group, y ponemos como nombre FlightDepartureAirport:



Ahora en la esta primera línea digitamos la tecla con el punto (‘.’) y GeneXus nos sugiere los atributos que comienzan con “FlightDepartureAirport”, que ya habíamos definido en la transacción Flight:

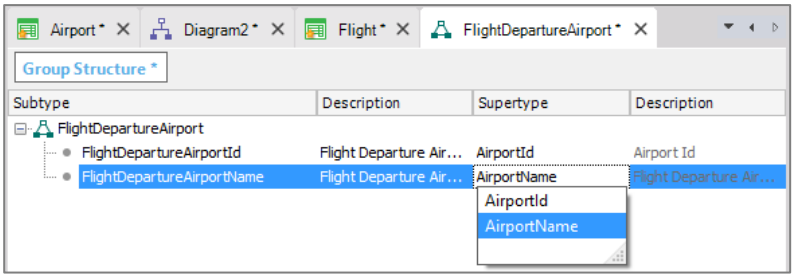


Elegimos entonces a FlightDepartureAirportId….presionamos tabulador y como queremos que FlightDepartureAirportId sea subtipo de AirportId, elegimos como supertipo al atributo AirportId:



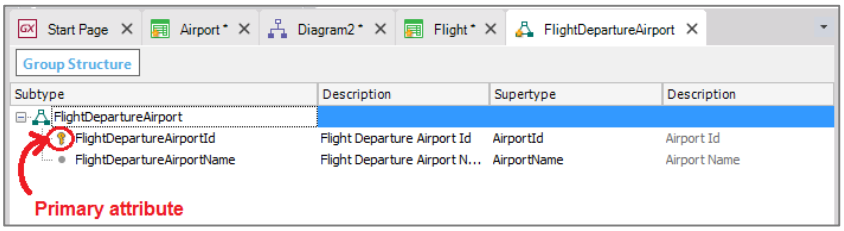
Podemos decir que el supertipo es el atributo original, y subtipo es el atributo que conceptualmente coincide con ese atributo original, pero que tiene otro nombre.

Ahora agregamos a FlightDepartureAirportName, y definimos que su supertipo es: AirportName

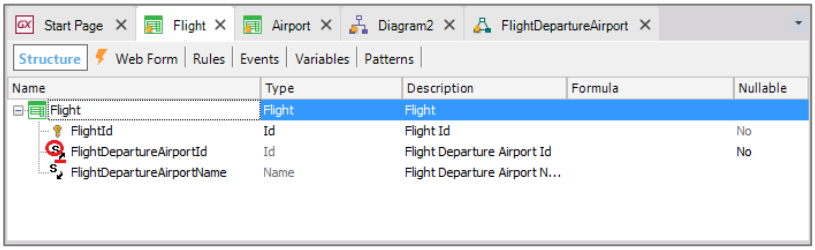


Grabamos.

Este atributo pasa a ser el identificador de este grupo de subtipos, por eso lo llamamos **“primario”**, y todos los atributos que agreguemos en este grupo, como FlightDepartureAirportName, dependerán de él, al igual que en la transacción.

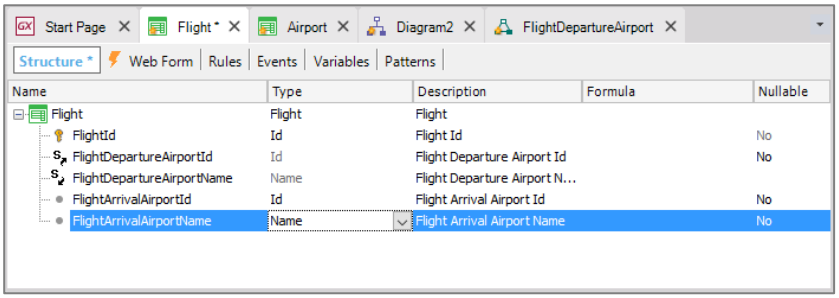


Vayamos ahora a la transacción Flight y observemos que el atributo FlightDepartureAirportId, tiene el símbolo que indica que será tratado como clave foránea… y además el símbolo de la letra S, que indica que es un atributo definido como subtipo:



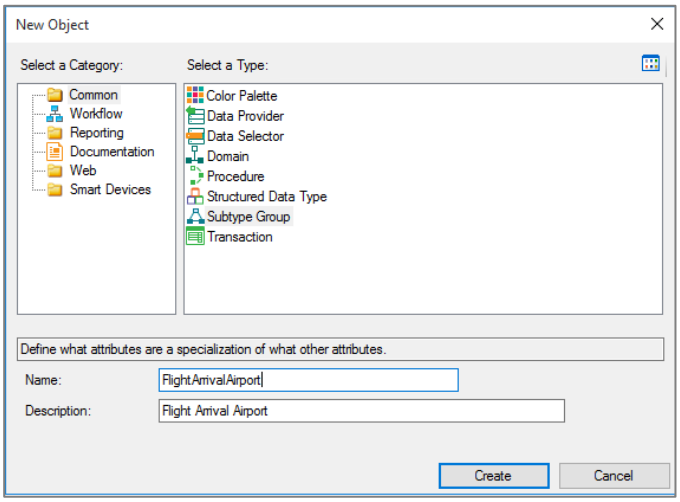
Vamos ahora a proceder de igual manera para definir los atributos que permitan registrar el aeropuerto hacia donde llega el vuelo.

Vamos a definir entonces los atributos FlightArrivalAirportId y FlightArrivalAirportName.



Y grabamos.

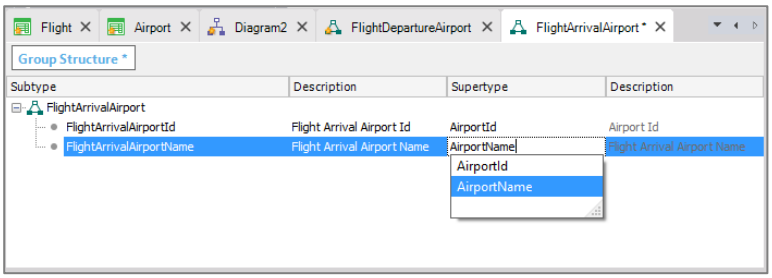
Creamos ahora un nuevo objeto, de tipo: Subtype group y ponemos como nombre FlightArrivalAirport:



Digitamos el punto (‘.’)… GeneXus nos sugiere los atributos que comienzan con “FlightArrivalAirport” y elegimos a FlightArrivalAirportId.

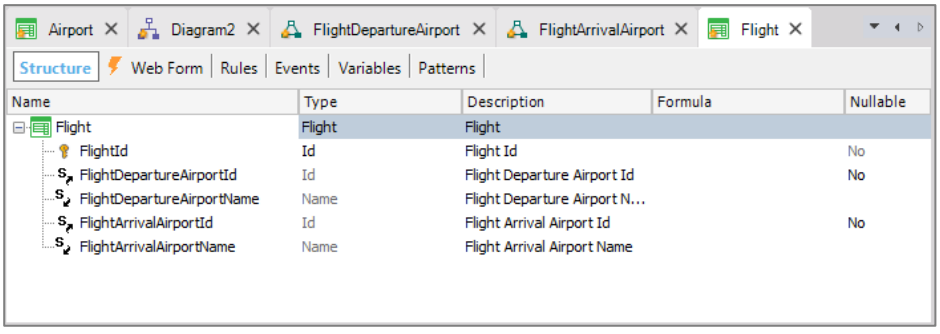
Damos tabulador y declaramos que sea subtipo de AirportId

Ahora agregamos a FlightArrivalAirportName… y definimos que su supertipo es: AirportName.



Grabamos.

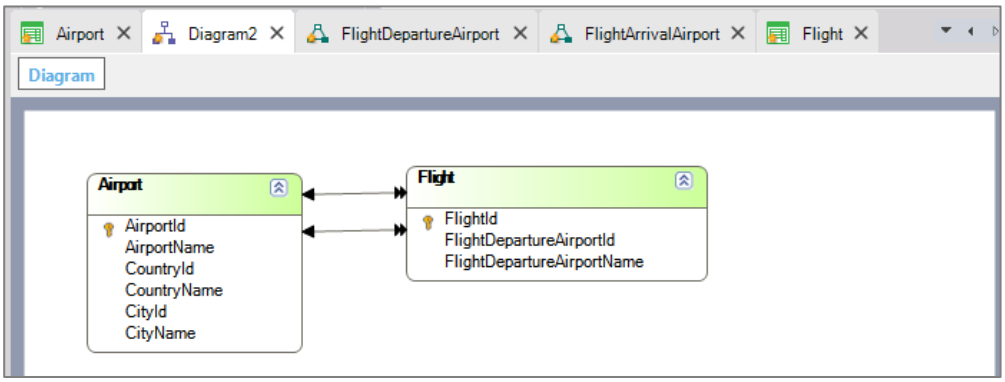
Veamos nuevamente la estructura de la transacción Flight...



Y volvamos a analizar ahora el diagrama de transacciones que habíamos creado antes.

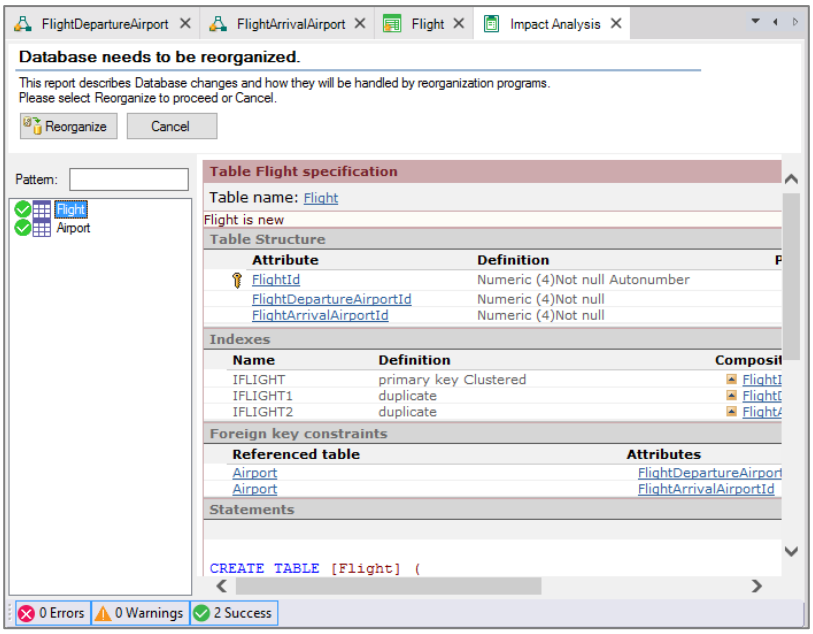
Vemos que ahora GeneXus sí coloca la flecha: GeneXus considera a los atributos subtipos identificadores de aeropuerto en Flight, exactamente igual que si hubiéramos referenciado a AirportId.

Vemos entonces que GeneXus ha encontrado la relación entre Flight y Airport.
Observemos que aunque en GeneXus está apareciendo una única flecha en el diagrama, siendo estrictos deberían aparecer dos.



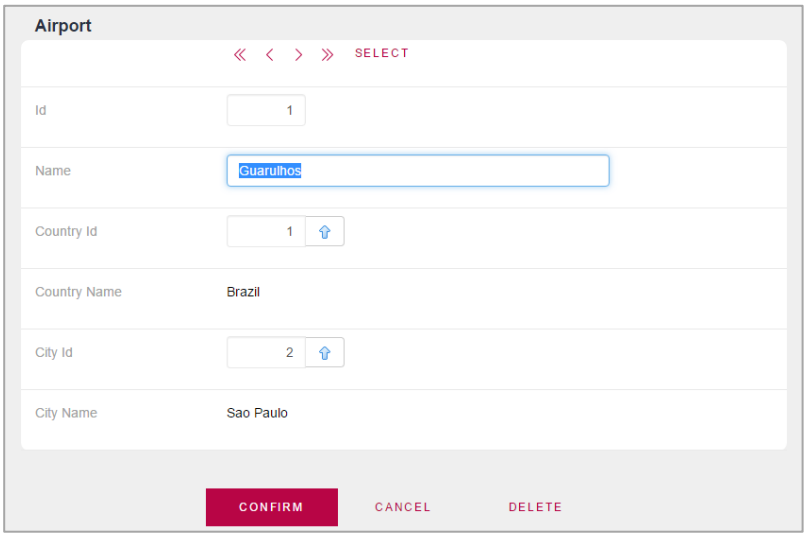
Veamos en funcionamiento todo esto. Presionemos F5...

Deberá reorganizarse la base de datos pues deberán crearse las tablas Flight y Airport. Estamos de acuerdo, así que presionamos Reorganize.

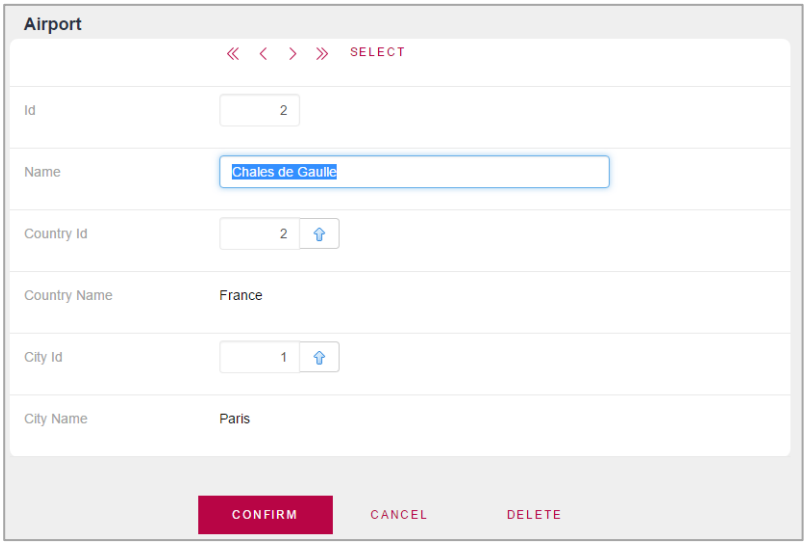


En primer lugar vamos a definir aeropuertos, así que ejecutamos la transacción Airport.

Vamos a ingresar al aeropuerto “Guarulhos”, indicamos el país: Brasil y la ciudad: Sao Paulo:

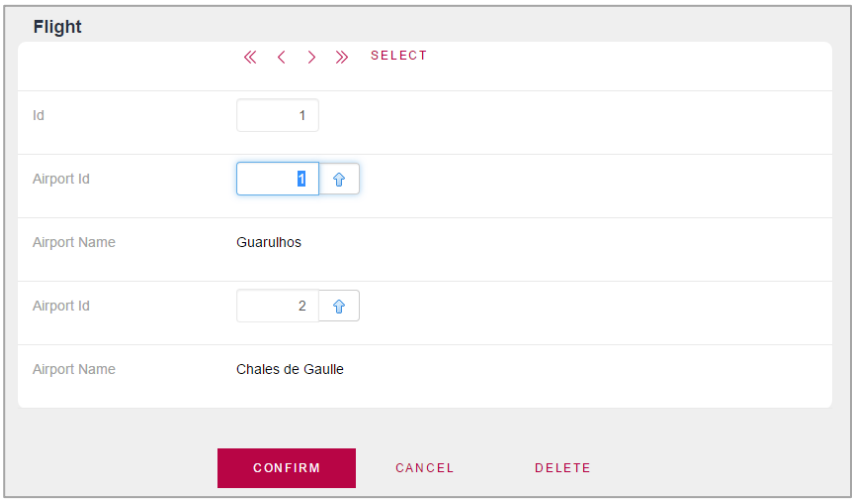


Ahora vamos a registrar al aeropuerto “Charles de Gaulle”… seleccionamos: Francia y la ciudad: París. Confirmamos.

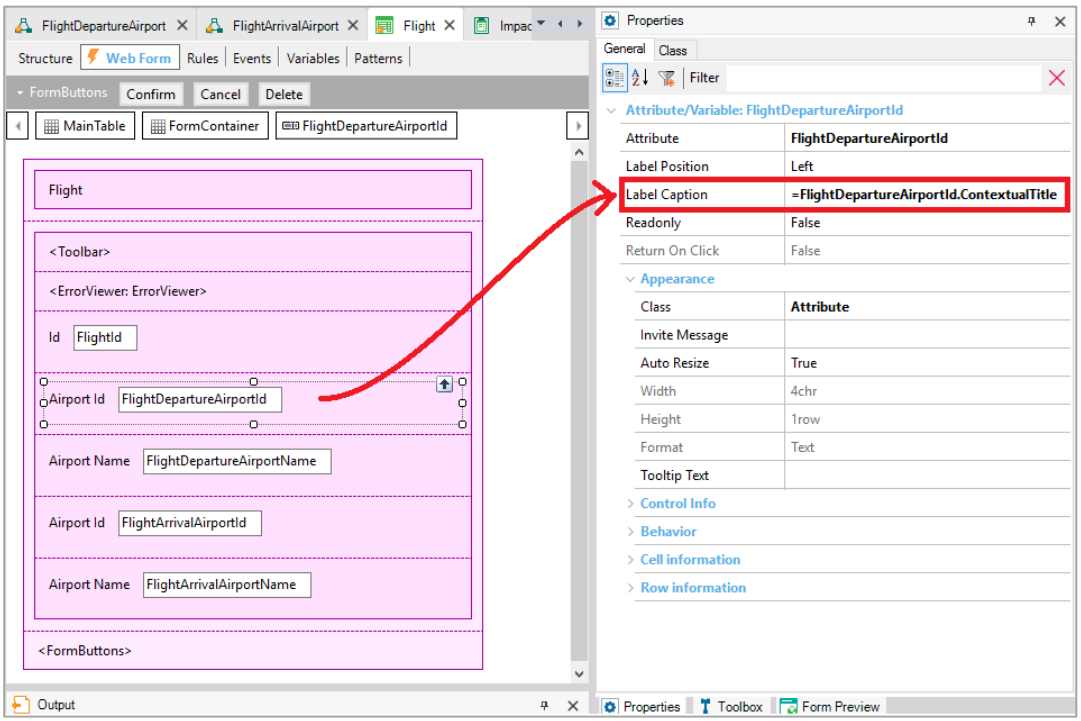


Pasemos ahora a registrar un vuelo.

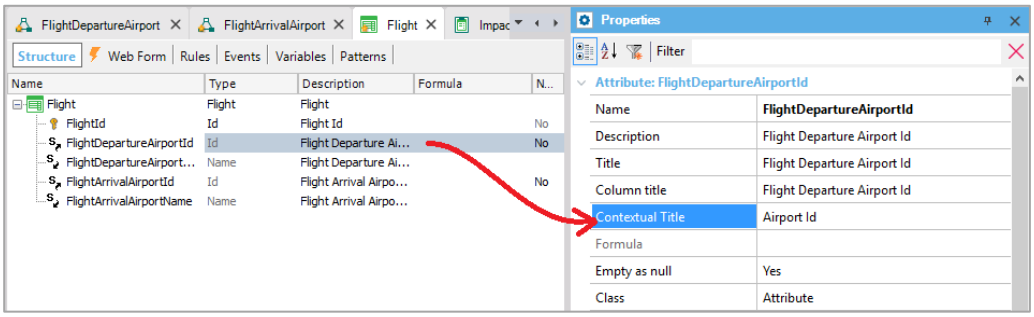
Ejecutamos la transacción Flight…Como aeropuerto de partida vamos a elegir “Guarulhos” y como aeropuerto de llegada: “Charles de Gaulle”:



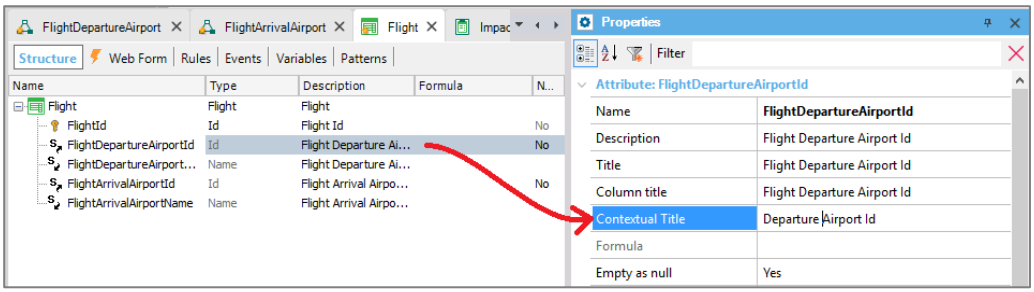
Vemos que las etiquetas de los atributos no nos están indicando que este es el aeropuerto de partida y este el de llegada. Si vamos al form de la transacción en GeneXus y nos posicionamos sobre el campo del primer aeropuerto:



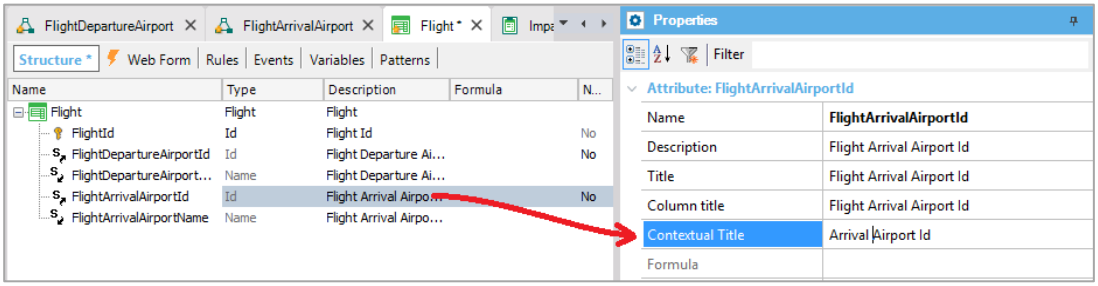
vemos que lo que dice la etiqueta, es decir, su Caption, se está tomando de la propiedad ContextualTitle del atributo . Si la vamos a buscar aquí es donde aparece:



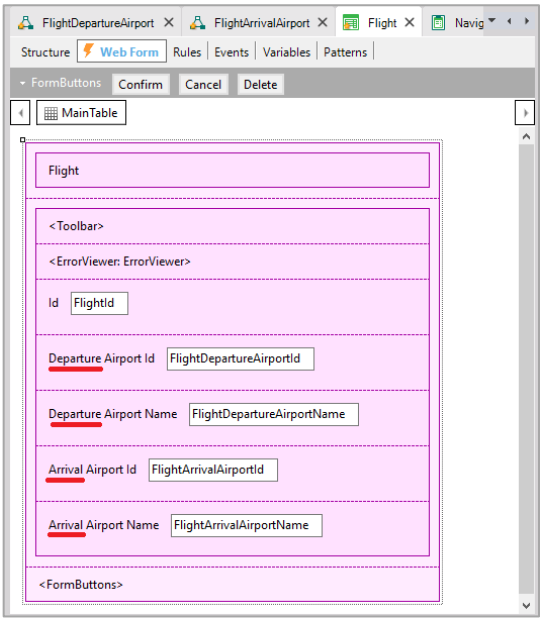
Le agregamos “Departure”:



Y hacemos lo mismo con el nombre y le agregamos “Arrival” al aeropuerto de llegada:



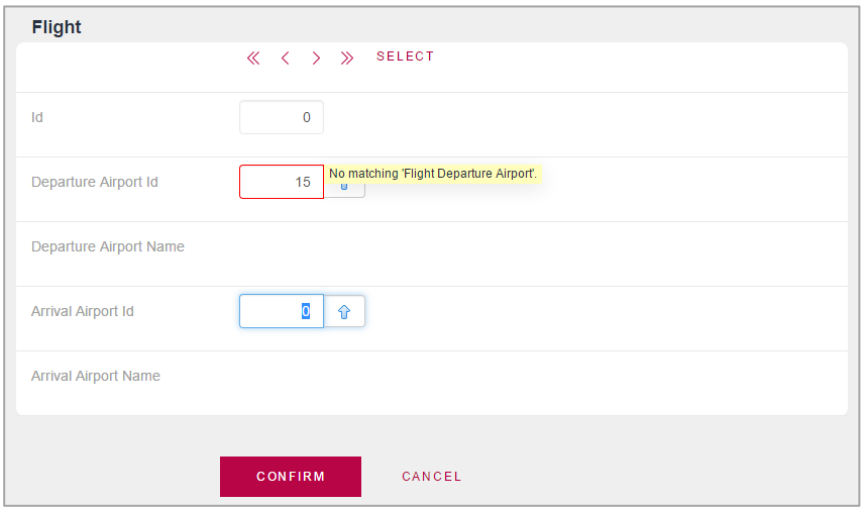
Vemos el Form:



Hagamos nuevamente F5.

Vamos ahora a ingresar otro vuelo.

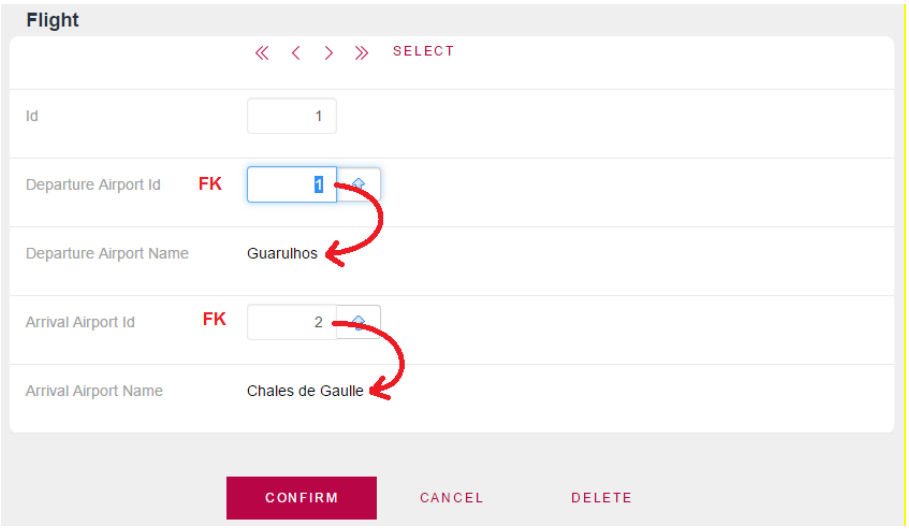
Probemos de digitar en el aeropuerto 15… sale el aviso de que este aeropuerto no existe.



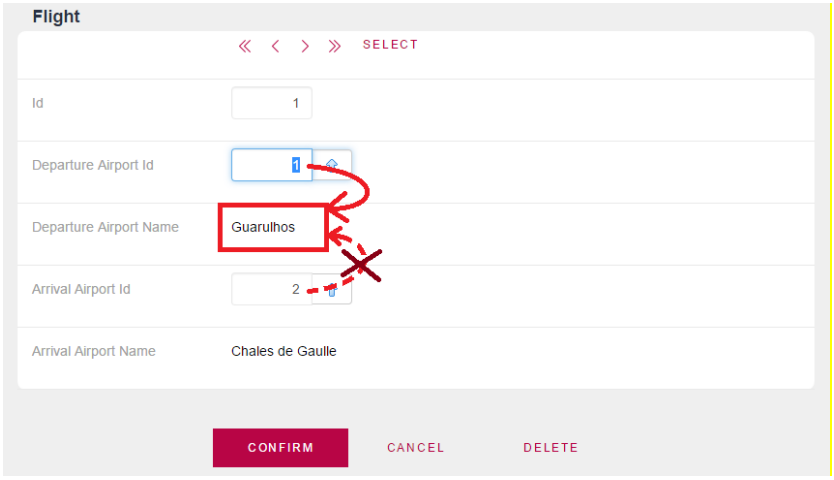
Se está controlando la consistencia de los datos. Y se está ofreciendo la lista de selección…… Vemos, así, que contamos con los mismos controles y ayudas que aparecían cuando los atributos eran llaves foráneas con sus nombres originales… como vimos en este video… **pero ahora se trata de atributos subtipos de ellos**.

Y esta es justamente la idea: **que definiendo subtipos logramos definir que nombres de atributos distintos correspondan al mismo concepto**.

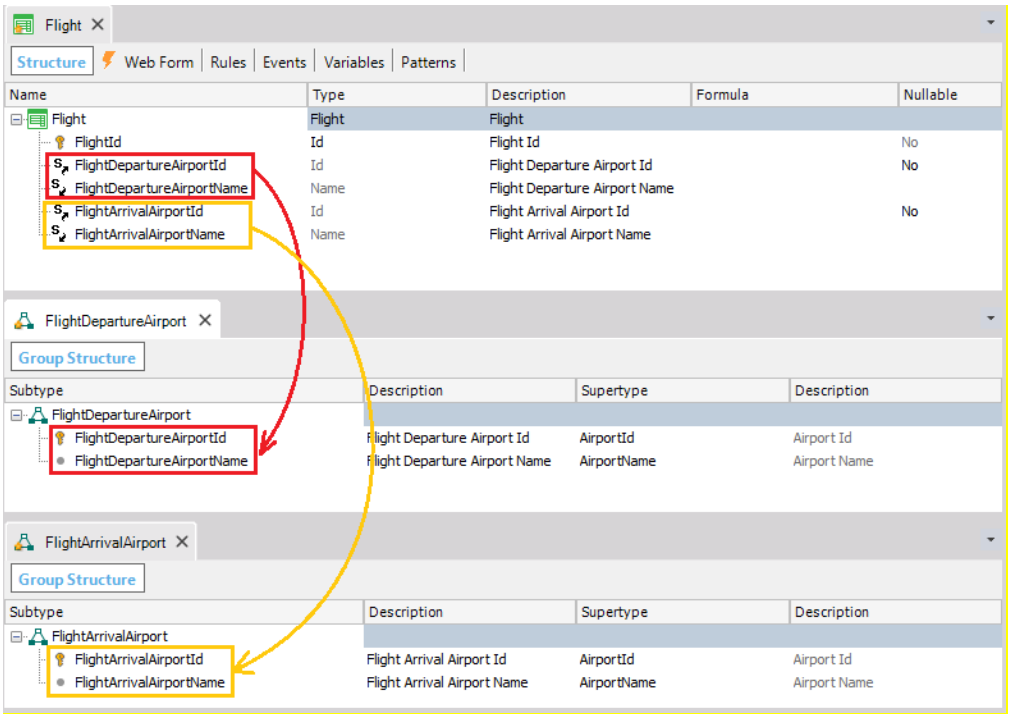
Es por eso que este atributo y este otro serán interpretados como llaves foráneas y que estos serán inferidos a partir de los primeros.



Pero ¿cómo sabe GeneXus que en este deberá inferirse el nombre de este aeropuerto, y no de este otro?



No es por el orden en el que aparecen los atributos ni por el nombre que les dimos. Lo sabe porque este atributo se ha definido en un grupo con este. Y este otro, en **otro** grupo, con este otro:



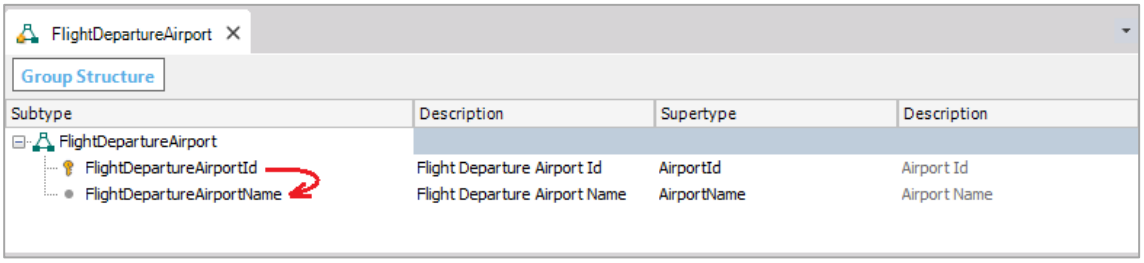
Por lo que es fundamental **agrupar en el mismo grupo de subtipos a los atributos que se corresponden**.

El uso de subtipos nos permitió representar una situación que se da en la realidad: en este caso que un vuelo tiene dos aeropuertos que cumplen distinto rol: uno es el aeropuerto de partida y otro es el aeropuerto de llegada. Y los grupos nos permiten diferenciar ambos roles y conectar los atributos de cada rol.

Destaquemos que no hemos incluido todos los subtipos en un mismo grupo, ni a los dos atributos subtipos primarios por un lado en un grupo y a los dos atributos secundarios por otro. No.
Hemos agrupado los atributos que definen al aeropuerto de partida juntos en un grupo y **a los atributos que definen al aeropuerto de llegada juntos en otro grupo**.

Repitámoslo: esto es así porque GeneXus entiende con este grupo:

que cuando se ingresa valor para este identificador de aeropuertoFlightDepartureAirportId

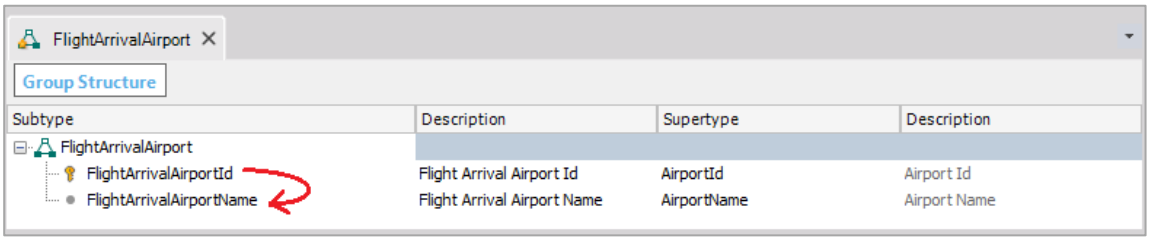


el nombre de aeropuerto correspondiente a **este identificador**, tiene que cargarse **en este atributo FlightDepartureAirportName**

y no en el otro nombre de aeropuerto que hay en la transacción.

De la misma manera GeneXus entiende que cuando se digita valor para el identificador de aeropuerto: FlightArrivalAirportId

el nombre del aeropuerto correspondiente debe cargarse en el atributo FlightArrivalAirportName.

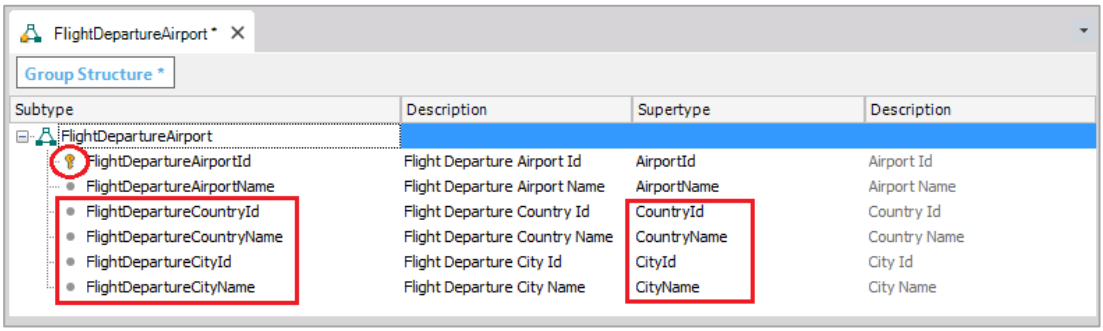


Bien. Ahora supongamos que en la transacción Flight, queremos para cada aeropuerto, ver además de su nombre, su país y su ciudad.

Esto simplemente se resuelve **definiendo más atributos subtipos en cada grupo, ayudando al desarrollador dándoles nombres significativos y no olvidando indicar quiénes son sus supertipos… y de esta manera GeneXus entenderá que para el subtipo primario del grupo, deberá inferir todo el resto de la información asociada.**

Vamos a hacerlo.

En este grupo de subtipos vamos a definir los atributos



FlightDepartureCountryId como subtipo de CountryId,

FlightDepartureCountryName como subtipo de CountryName,

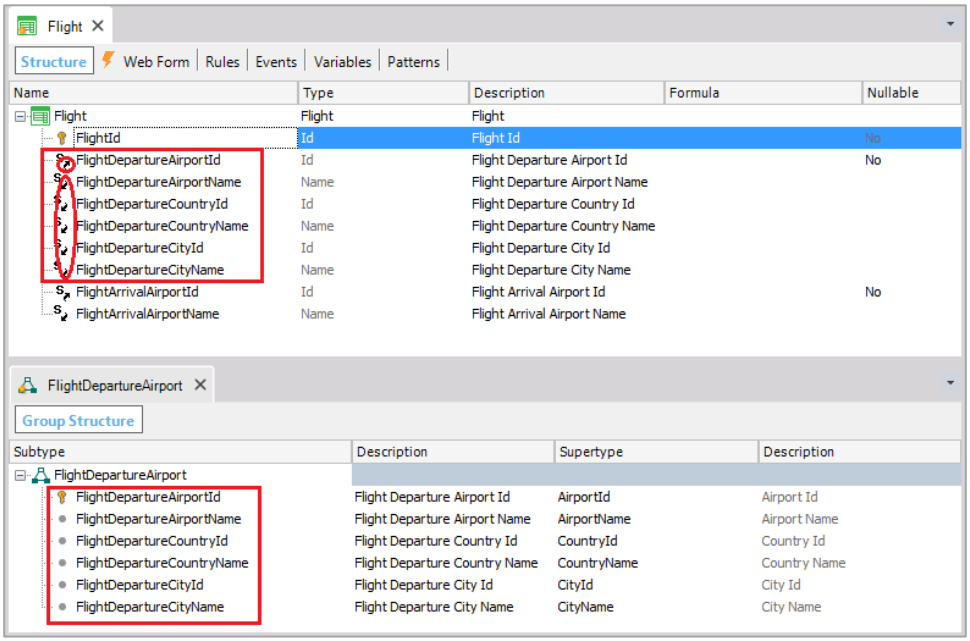
FlightDepartureCityId como subtipo de CityId

y FlightDepartureCityName como subtipo de CityName.

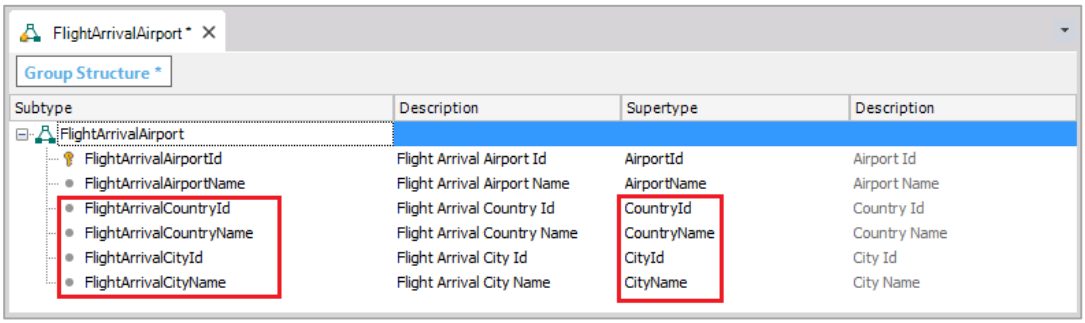
Grabamos.

Y ahora vamos a agregar estos nuevos atributos a la estructura de la transacción Flight

Grabamos.



Y lo mismo hacemos para el otro grupo de subtipos….



Definimos

FlightArrivalCountryId como subtipo de CountryId,

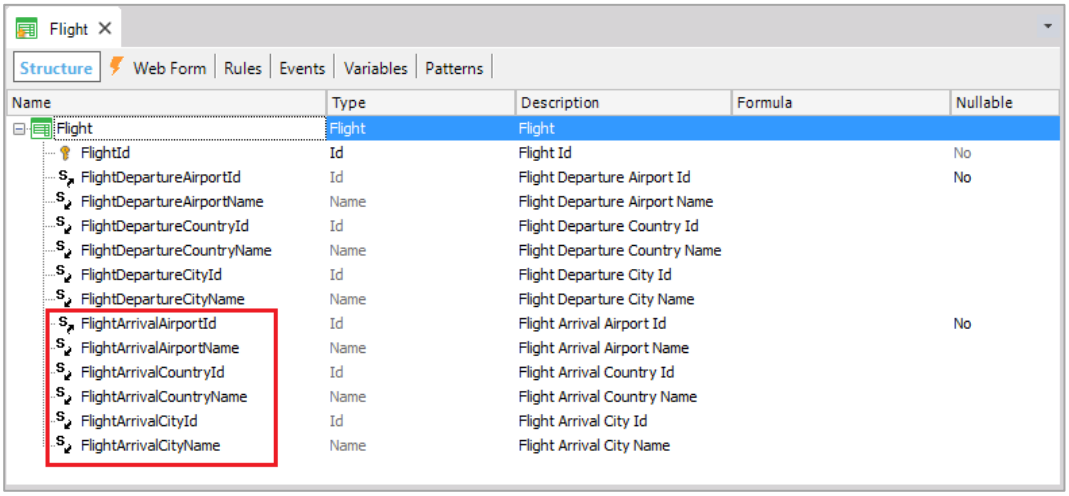
FlightArrivalCountryName subtipo de CountryName,

FlightArrivalCityId subtipo de CityId

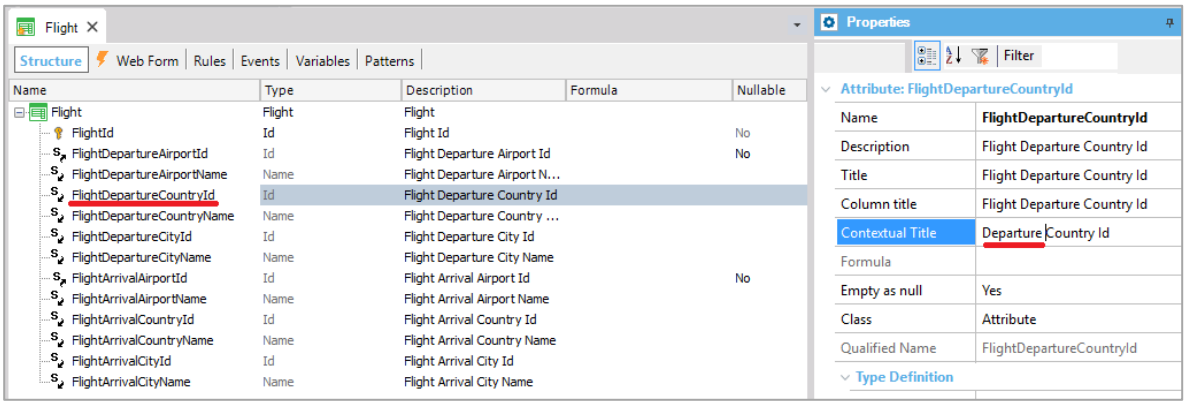
y FlightArrivalCityName subtipo de CityName.

Salvamos y también los agregamos en la estructura de la transacción Flight.

Grabamos.

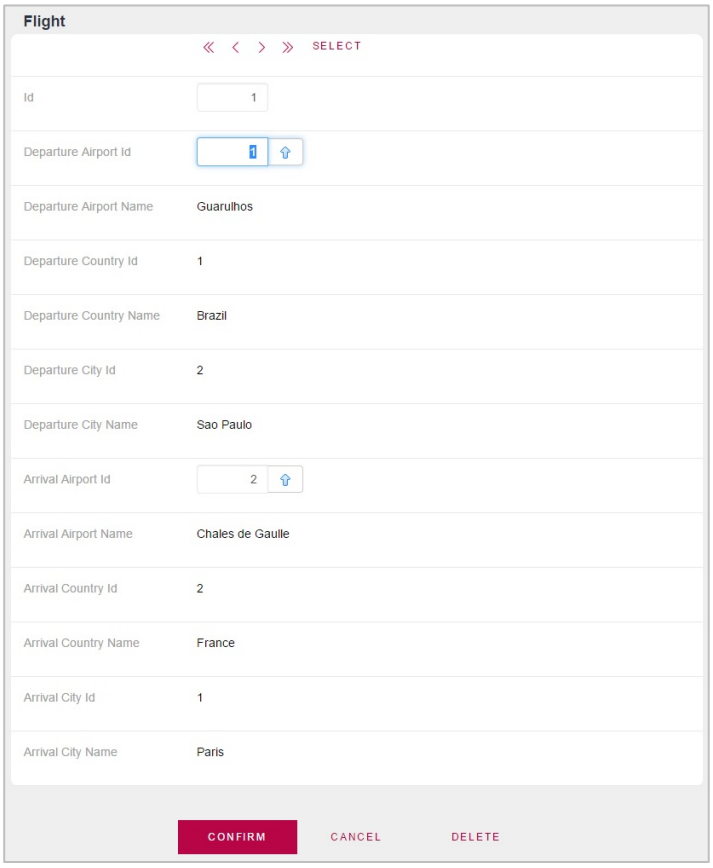


Como hicimos antes, a cada atributo agregado le cambiamos el título contextual para que la etiqueta en la pantalla indique el rol:

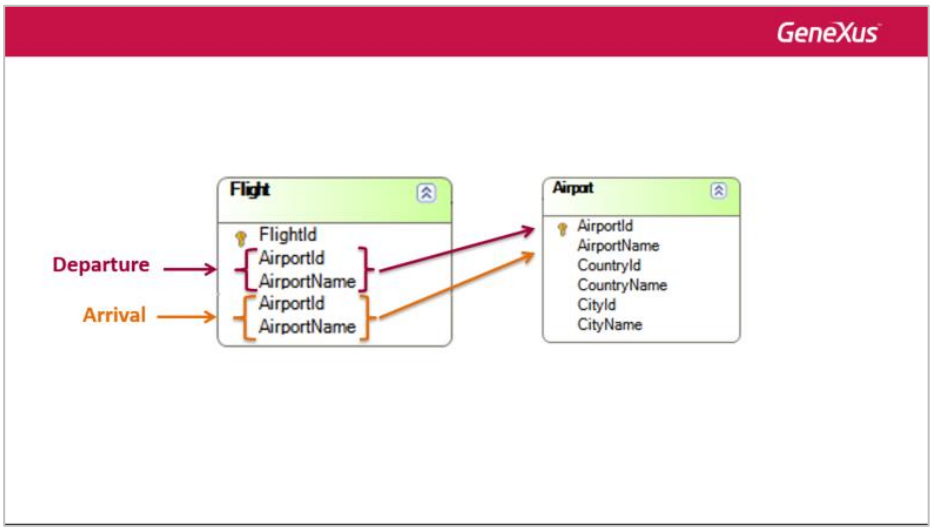


Nuevamente presionemos F5 para ejecutar la aplicación.

Abrimos la transacción Flight, consultamos nuestro primer vuelo y podemos ver de cada aeropuerto su país y su ciudad.

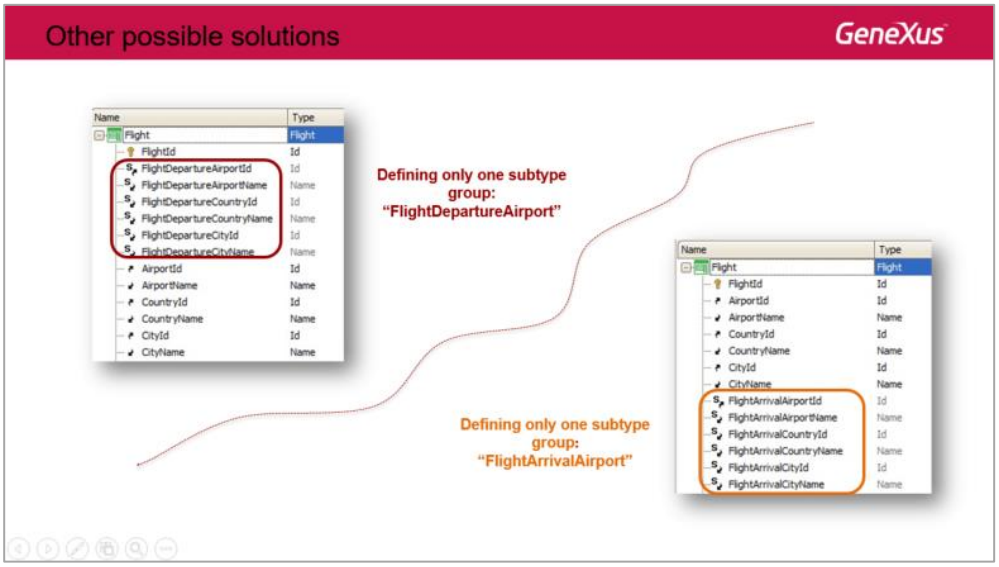


De esta forma hemos visto cómo resolver una doble referencia a un mismo concepto pero con distintos roles, ya que los dos aeropuertos debían obtenerse de la misma tabla



pero cada uno tenía diferente rol.

Para finalizar, es importante saber que **éstas** podrían haber sido también soluciones válidas:



En esta primera propuesta

se ha definido un único grupo de subtipos: el grupo correspondiente al aeropuerto de partida… y se ha dejado a los atributos con el nombre original (AirportId, AirportName, CountryId, etc.) para el ingreso del destino.

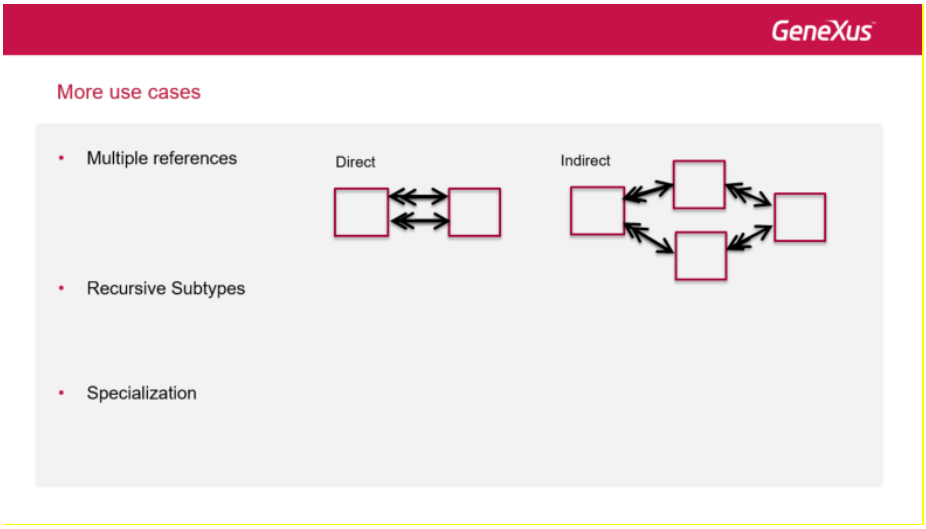
Y esto es completamente válido.

En esta segunda propuesta… se ha definido un único grupo de subtipos también, pero en este caso para el grupo correspondiente al **aeropuerto de llegada**.

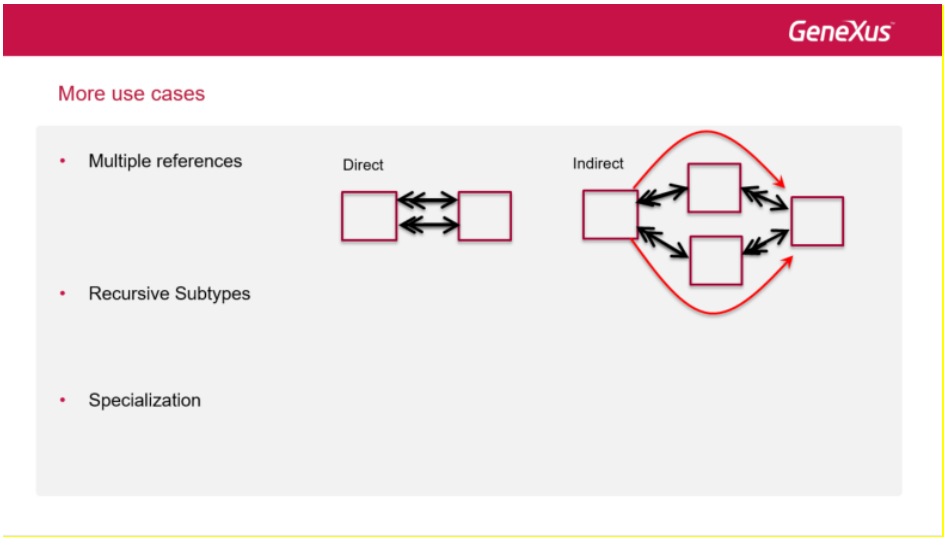
En resumen, los subtipos nos permiten indicarle a GeneXus cómo asociar distintos nombres de atributo a un

mismo concepto. Y como vimos, las validaciones y todo el comportamiento de los subtipos será idéntico a si hubiéramos usado los atributos originales.

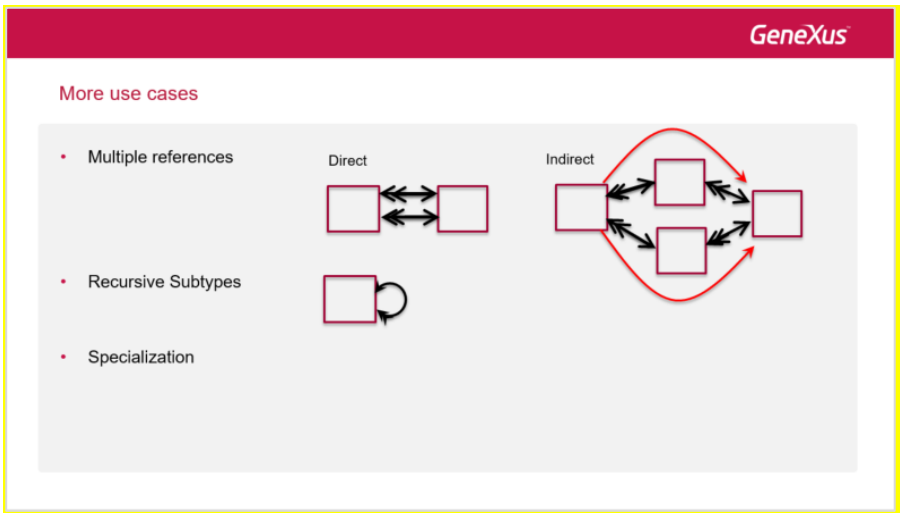
Hay muchos otros casos en los que es necesario modificarle el nombre a un atributo para evitar un conflicto o ambigüedad. En este video vimos el caso de referencias múltiples de una tabla a otra pero estas referencias no tienen por qué ser directas.



Desde esta tabla tenemos dos caminos para llegar a esta otra tabla. Por lo que se necesitarán subtipos para diferenciarlos.

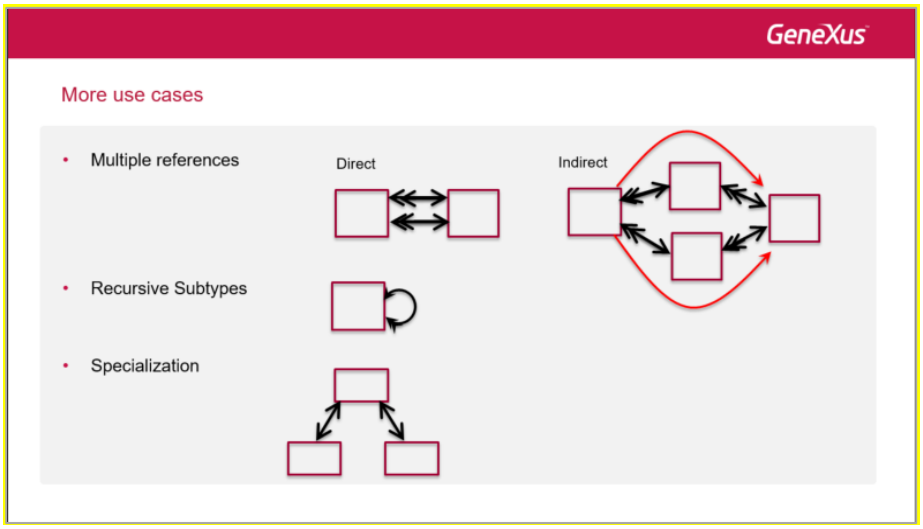


También tenemos el caso de una entidad que debe referenciarse a ella misma:



por ejemplo una transacción de empleados, en la que entre la información del empleado hay que registrar al jefe, que también es un empleado.

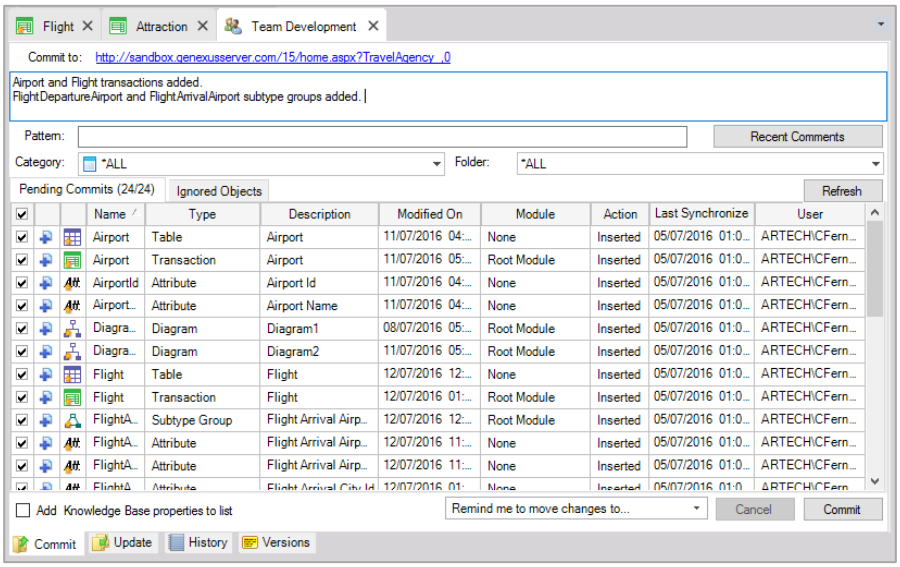
O el caso en el que tenemos una entidad que registra información general, por ejemplo, de personas (como el nombre, número de teléfono, dirección, etcétera) y luego tenemos entidades que son una especialización de esa otra, por ejemplo, clientes y pasajeros, que en particular son personas.



Estos son sólo algunos ejemplos de los múltiples que hay. Sólo los mencionamos. No los estudiaremos en este curso.

Para finalizar, actualizamos los cambios en GeneXus Server. Agregamos los comentarios:

Y presionamos Commit:


Flight
Attraction
Team Development
Commit to: http://sandbox.genexusserver.com/15/home.aspx?TravelAgency_0
Airport and Flight transactions added.
FlightDepartureAirport and FlightArrivalAirport subtype groups added.
Pattern:
Recent Comments
Category: *ALL
Folder: *ALL
Pending Commits (24/24)
Ignored Objects
Refresh
Name
Type
Description
Modified On
Module
Action
Last Synchronize
User
Airport Table Airport 11/07/2016 04:... None Inserted 05/07/2016 01:0... ARTECH\CFern...
Airport Transaction Airport 11/07/2016 05:... Root Module Inserted 05/07/2016 01:0... ARTECH\CFern...
AirportId Attribute Airport Id 11/07/2016 04:... None Inserted 05/07/2016 01:0... ARTECH\CFern...
Airport... Attribute Airport Name 11/07/2016 04:... None Inserted 05/07/2016 01:0... ARTECH\CFern...
Diagra... Diagram Diagram1 08/07/2016 05:... Root Module Inserted 05/07/2016 01:0... ARTECH\CFern...
Diagra... Diagram Diagram2 11/07/2016 05:... Root Module Inserted 05/07/2016 01:0... ARTECH\CFern...
Flight Table Flight 12/07/2016 12:... None Inserted 05/07/2016 01:0... ARTECH\CFern...
Flight Transaction Flight 12/07/2016 01:... Root Module Inserted 05/07/2016 01:0... ARTECH\CFern...
FlightA... Subtype Group Flight Arrival Airp... 12/07/2016 12:... Root Module Inserted 05/07/2016 01:0... ARTECH\CFern...
FlightA... Attribute Flight Arrival Airp... 12/07/2016 11:... None Inserted 05/07/2016 01:0... ARTECH\CFern...
FlightA... Attribute Flight Arrival Airp... 12/07/2016 11:... None Inserted 05/07/2016 01:0... ARTECH\CFern...
Add Knowledge Base properties to list
Remind me to move changes to...
Cancel
Commit
Commit
Update
History
Versions


GeneXus
Videos training.genexus.com
Documentation wiki.genexus.com
Certifications training.genexus.com/certifications