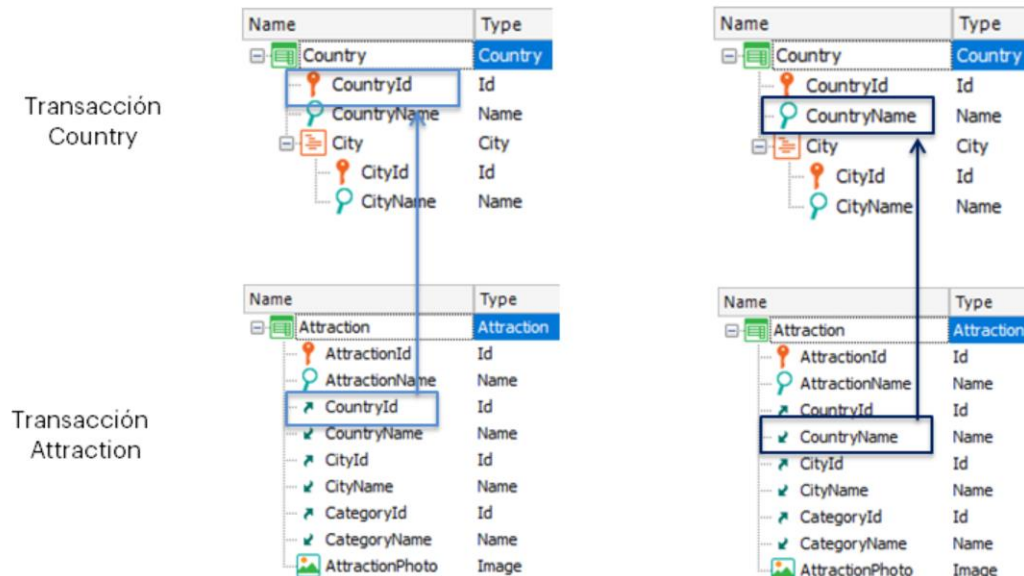


Diferentes nombres de atributo para un mismo concepto

Grupo de subtipo

GeneXus® 16

- GeneXus determina relaciones para atributos con el mismo nombre.



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**.

Requerimiento

- Registrar los vuelos que se ofrecen a los clientes para arribar a una atracción turística.

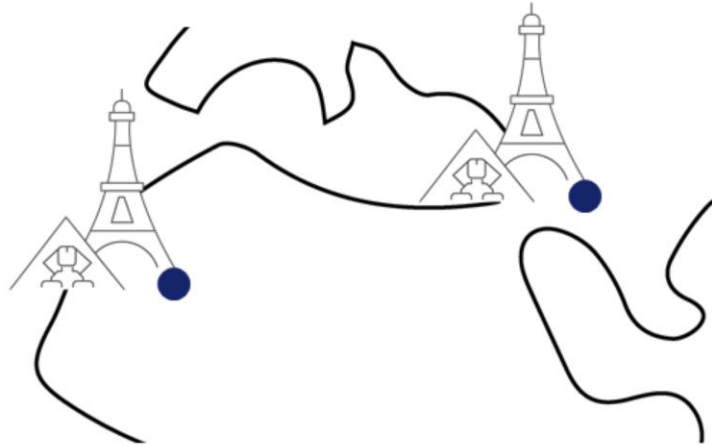


Veamos esto.

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

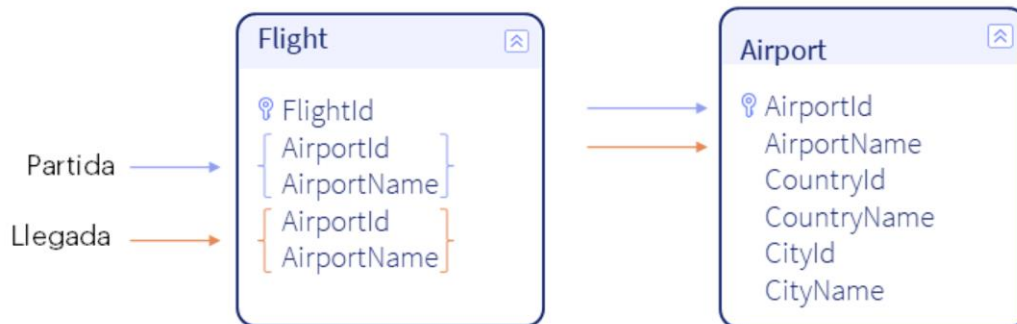
Requerimiento

- Para cada vuelo se debe registrar el aeropuerto de partida así como también el aeropuerto de llegada.



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

Se necesitan atributos con diferente nombre para representar un mismo concepto



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...

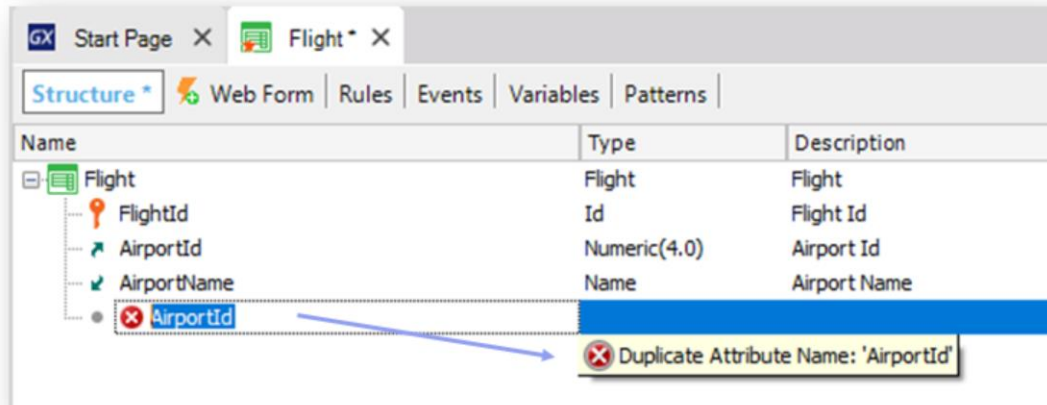
Pero a los aeropuertos tendremos que poder registrarlos por sí mismos..., así luego podremos referenciarlos desde los vuelos.

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.

Nombres duplicados

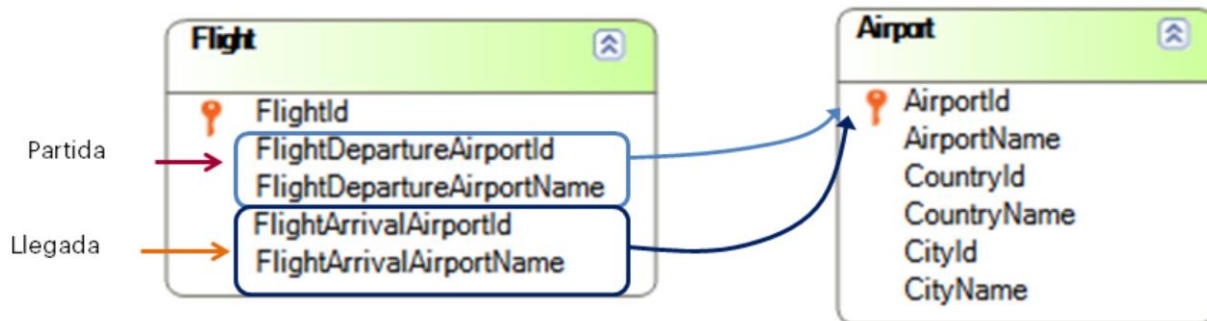


Así que volvemos a la transacción Flight, y vamos a agregar los atributos AirportId, y AirportName.... Pero cuando intentamos agregar nuevamente AirportId, ¡GeneXus nos dice que hay un error!, que estamos agregando un atributo con nombre duplicado.

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.

Definiendo la solución...



Diferentes nombres de atributo
¡No hay relación
entre las transacciones!

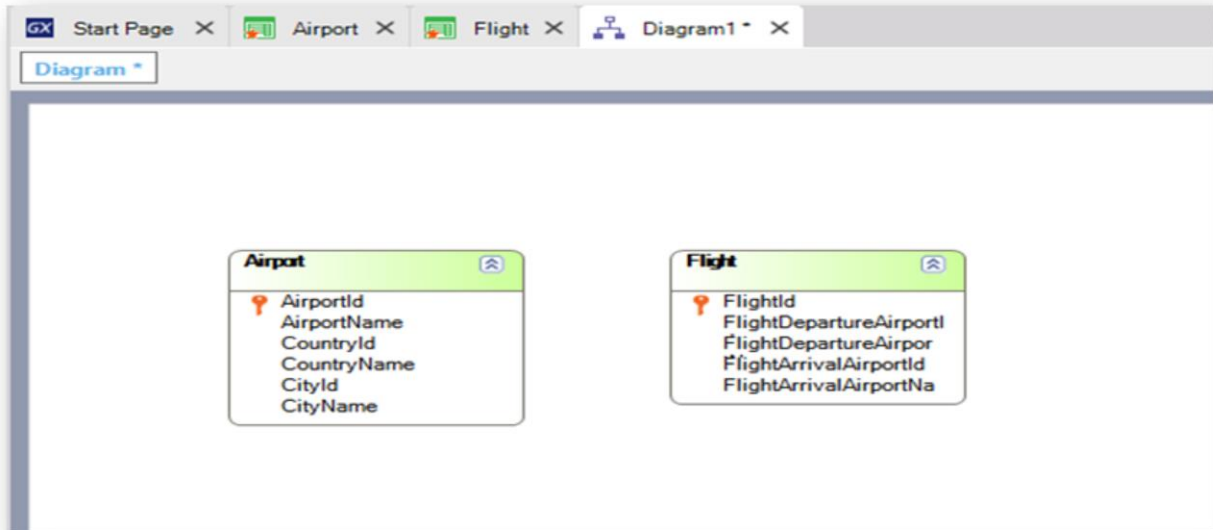
Vamos a borrar entonces los atributos que originalmente habíamos ingresado y vamos a definir atributos con nombres nuevos.

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.

Diagrama

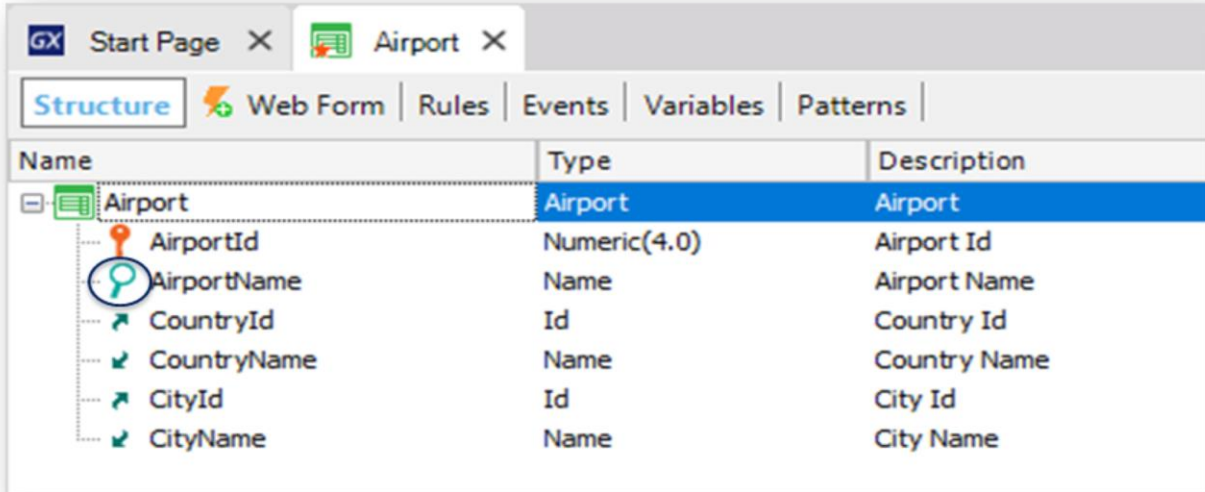


Para verificar esto que acabamos de decir, vamos a crear un diagrama de transacciones.

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”

Atributo descriptor



The screenshot shows the GeneXus IDE interface with the 'Airport' transaction selected. The 'Structure' tab is active, displaying a table of attributes. The 'AirportName' attribute is highlighted with a magnifying glass icon, indicating it is the descriptor attribute.

Name	Type	Description
Airport	Airport	Airport
AirportId	Numeric(4.0)	Airport Id
AirportName	Name	Airport Name
CountryId	Id	Country Id
CountryName	Name	Country Name
CityId	Id	City Id
CityName	Name	City Name

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.

Atributo descriptor

The screenshot displays the GeneXus IDE interface. In the background, a table named 'Attractions' is visible with columns: Name, Category Name, Country Name, Country City Name, Photo, and two action buttons (UPDATE, DELETE). In the foreground, the 'Structure' pane shows the 'Flight' entity. A context menu is open over the 'FlightDepartureAirportId' attribute, which is marked with a key icon. The menu options include 'Open Id', 'Delete', 'Toggle Key', 'Toggle Description Attribute' (highlighted in yellow), and 'Toggle Image Attribute'.

Name	Category Name	Country Name	Country City Name	Photo	UPDATE	DELETE
Great Wall	Monument	China	Beijing			
Louvre Museum	Museum	Francia	Paris			

Name	Type	Description
Flight	Flight	Flight
FlightId	Id	Flight Id
FlightDepartureAirportId	Id	Flight Departure Airport Id
FlightDepartureAirportName		

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 señalado con este símbolo, que indica que es un atributo secundario y no es considerado como clave foránea...

Clave foránea

Name	Type	Description
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
AttractionAddress	Address, GeneXus	Attraction Address

Atributo secundario

Name	Type	Description
Flight	Flight	Flight
FlightId	Id	Flight Id
FlightDepartureAirportId	Id	Flight Departure Airport Id
FlightDepartureAirportName	Name	Flight Departure Airport Name

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.

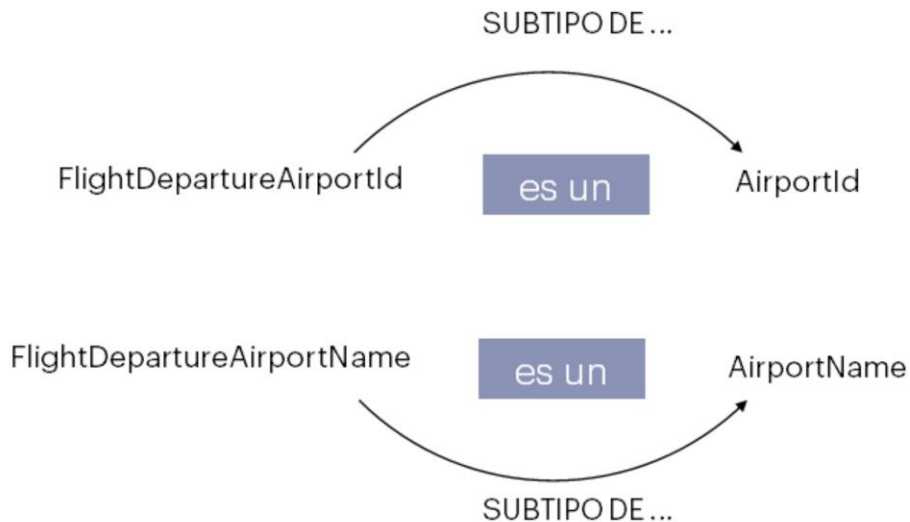
Entonces, ¿cómo hacemos para que GeneXus pueda asociar distintos nombres a un mismo concepto?

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.

Definiendo la solución...

- A través de los subtipos es posible hacer que dos atributos con diferente nombre correspondan al mismo concepto



¿Cómo lo podremos lograr?

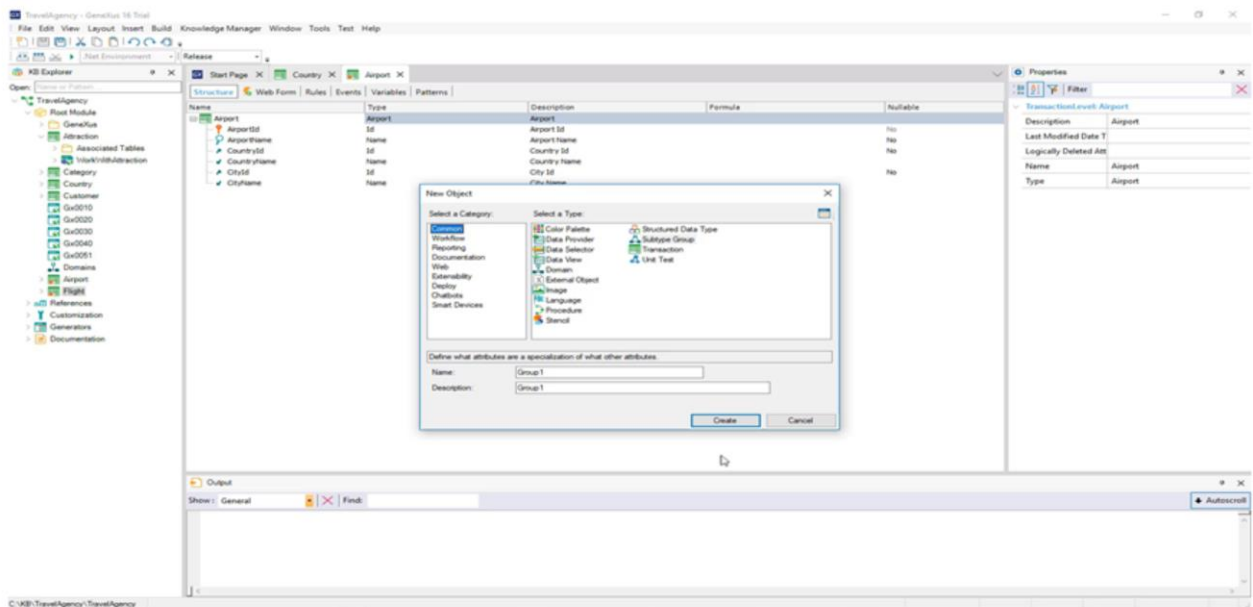
La respuesta es: **mediante la definición de subtipos.**

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.

DEMO



[DEMO: <https://youtu.be/swgogPuGnOM>]

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.

The screenshot displays three GeneXus object structure windows:

- FlightDepartureAirport**: Shows a subtype hierarchy where `FlightDepartureAirportId` and `FlightDepartureAirportName` are subtypes of `FlightDepartureAirport`. Their supertypes are `AirportId` and `AirportName` respectively.
- FlightArrivalAirport**: Shows a subtype hierarchy where `FlightArrivalAirportId` and `FlightArrivalAirportName` are subtypes of `FlightArrivalAirport`. Their supertypes are `AirportId` and `AirportName` respectively.
- Flight**: Shows a list of attributes including `FlightId` (Type: Flight), `FlightDepartureAirportId` (Type: Id), `FlightDepartureAirportName` (Type: Name), `FlightArrivalAirportId` (Type: Id), and `FlightArrivalAirportName` (Type: Name). Arrows from the first two windows point to the corresponding attributes in the `Flight` object.

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`.

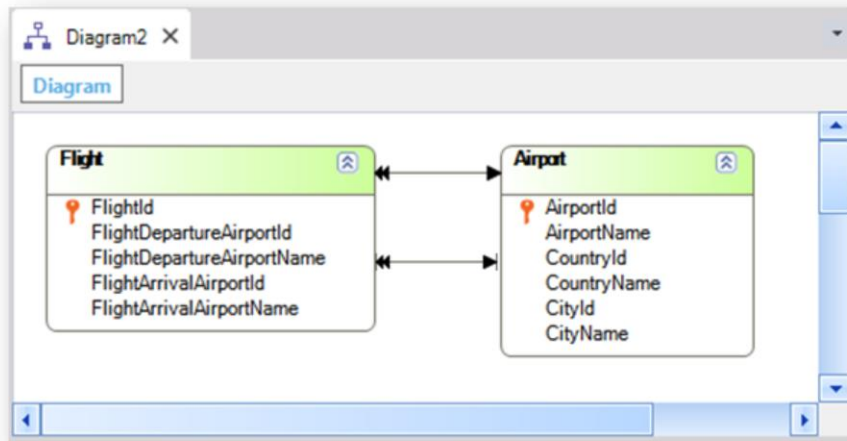
Damos tabulador y declaramos que sea subtipo de `AirportId`.

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

Grabamos.

Veamos nuevamente la estructura de la transacción Flight...

Diagrama: GeneXus encontró una relación



¡Se chequeará la integridad referencial de los datos en tiempo de ejecución!

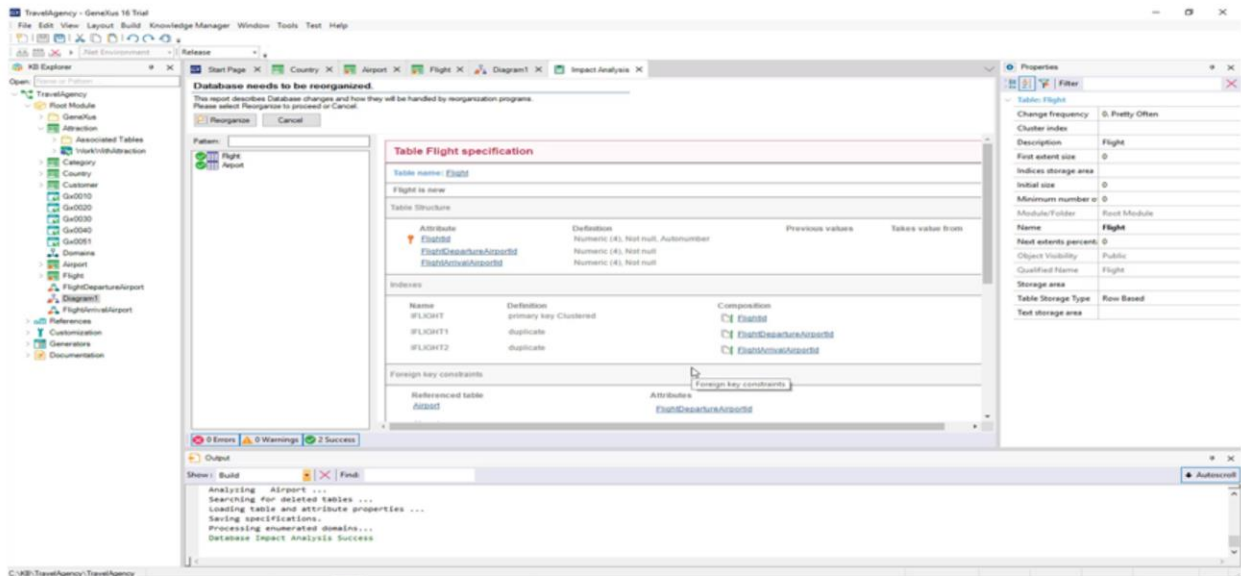
Y volvamos a analizar ahora el diagrama de transacciones que habíamos creado antes.

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.

DEMO



[DEMO: <https://youtu.be/3qKc9qMpDoo>]

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?

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		
FlightArrivalAirportId	Id	Flight Arrival Airport Id		No
FlightArrivalAirportName	Name	Flight Arrival Airport Name		

Subtype	Description	Supertype	Description
FlightDepartureAirport			
FlightDepartureAirportId	Flight Departure Airport Id	AirportId	Airport Id
FlightDepartureAirportName	Flight Departure Airport Name	AirportName	Airport Name

Subtype	Description	Supertype	Description
FlightArrivalAirport			
FlightArrivalAirportId	Flight Arrival Airport Id	AirportId	Airport Id
FlightArrivalAirportName	Flight Arrival Airport Name	AirportName	Airport Name

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.

Grupos de subtipos

Subtype	Description	Supertype	Description
FlightDepartureAirport			
FlightDepartureAirportId	Flight Departure Airport Id	AirportId	Airport Id
FlightDepartureAirportName	Flight Departure Airport Name	AirportName	Airport Name

Subtype	Description	Supertype	Description
FlightArrivalAirport			
FlightArrivalAirportId	Flight Arrival Airport Id	AirportId	Airport Id
FlightArrivalAirportName	Flight Arrival Airport Name	AirportName	Airport Name

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

que cuando se ingresa valor para este identificador de aeropuerto `FlightDepartureAirportId` 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`.

Grupos de subtipos

Flight

« < > » SELECT

Id

0

Departure Airport Id

15

No matching Flight Departure Airport.

Departure Airport Name

Arrival Airport Id

Arrival Airport Name

CONFIRM

CANCEL

Flight

« < > » SELECT

Id

1

Departure Airport Id

FK

Departure Airport Name

Guarulhos

Arrival Airport Id

FK

2

Arrival Airport Name

Chales de Gaulle

CONFIRM

CANCEL

DELETE

- Control de IR
- Lista de selección
- Inferencias agrupadas

¿Y si por cada aeropuerto queremos ver su país y ciudad?

FlightDepartureAirport X			
Group Structure			
Subtype	Description	Supertype	D
FlightDepartureAirport			
FlightDepartureAirportId	Flight Departure Airport Id	AirportId	...
FlightDepartureAirportName	Flight Departure Airport Name	AirportName	...
FlightDepartureCountryId	Flight Departure Country Id	CountryId	...
FlightDepartureCountryName	Flight Departure Country Name	CountryName	...
FlightDepartureCityId	Flight Departure City Id	CityId	...
FlightDepartureCityName	Flight Departure City Name	CityName	...

FlightArrivalAirport X			
Group Structure			
Subtype	Description	Supertype	Descri...
FlightArrivalAirport			
FlightArrivalAirportId	Flight Arrival Airport Id	AirportId	Airpor...
FlightArrivalAirportName	Flight Arrival Airport Name	AirportName	Airpor...
FlightArrivalCountryId	Flight Arrival Country Id	CountryId	Count...
FlightArrivalCountryName	Flight Arrival Country Name	CountryName	Count...
FlightArrivalCityId	Flight Arrival City Id	CityId	City Id
FlightArrivalCityName	Flight Arrival City Name	CityName	City N...

Flight X	
Structure	
Name	Type
Flight	Flight
FlightId	Id
FlightDepartureAirportId	Id
FlightDepartureAirportName	Name
FlightDepartureCountryId	Id
FlightDepartureCountryName	Name
FlightDepartureCityId	Id
FlightDepartureCityName	Name
FlightArrivalAirportId	Id
FlightArrivalAirportName	Name
FlightArrivalCountryId	Id
FlightArrivalCountryName	Name
FlightArrivalCityId	Id
FlightArrivalCityName	Name

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.

Transacción Flight

Application Name by GeneXus

Recents **Flight**

Flight

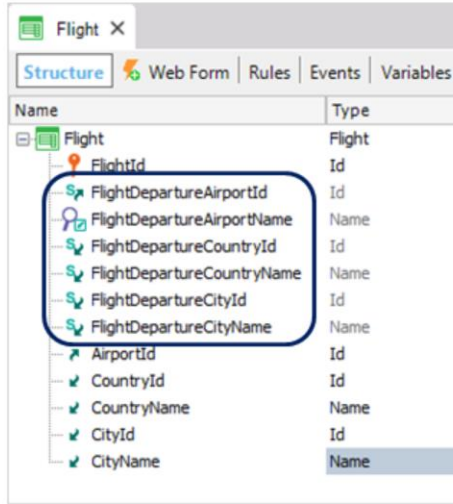
<< < > >> SELECT

Id	1
DepartureAirport Id	1
DepartureAirport Name	Guarulhos
DepartureCountry Id	1
DepartureCountry Name	Brazil
DepartureCity Id	2
DepartureCity Name	Sao Paulo
ArrivalAirport Id	2
ArrivalAirport Name	Charles de Gaulle
ArrivalCountry Id	2
ArrivalCountry Name	France

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

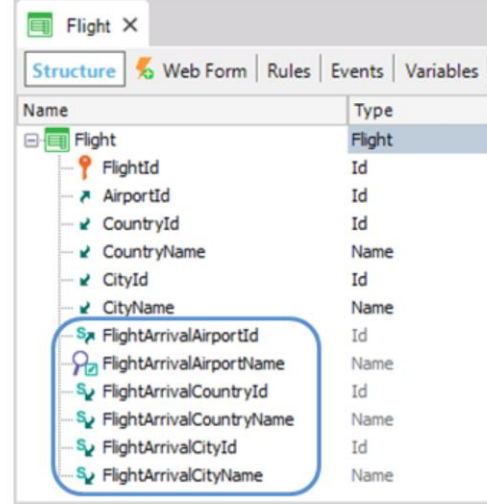
De esta forma hemos visto cómo resolver una doble referencia a un mismo concepto pero con distintos roles, ya que los dos aeropuertos debían obtenerse de la misma tabla pero cada uno tenía diferente rol.

Otras posibles soluciones



Name	Type
Flight	Flight
FlightId	Id
FlightDepartureAirportId	Id
FlightDepartureAirportName	Name
FlightDepartureCountryId	Id
FlightDepartureCountryName	Name
FlightDepartureCityId	Id
FlightDepartureCityName	Name
AirportId	Id
CountryId	Id
CountryName	Name
CityId	Id
CityName	Name

Definiendo solo un grupo de subtipo:
"FlightDepartureAirport"



Name	Type
Flight	Flight
FlightId	Id
AirportId	Id
CountryId	Id
CountryName	Name
CityId	Id
CityName	Name
FlightArrivalAirportId	Id
FlightArrivalAirportName	Name
FlightArrivalCountryId	Id
FlightArrivalCountryName	Name
FlightArrivalCityId	Id
FlightArrivalCityName	Name

Definiendo solo un grupo de subtipo:
"FlightArrivalAirport"

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.

Resumen y observaciones

Los subtipos nos han permitido representar una realidad en la que un vuelo tiene dos aeropuertos con roles diferentes.

Los grupos de subtipos nos permiten diferenciar ambos roles (no definimos un único grupo con todos los subtipos).

La forma correcta de hacer esto es la siguiente:

- Definir un grupo de subtipos para cada conjunto de atributos que coincidan.
- Cada grupo de subtipos debe incluir necesariamente un subtipo de atributo primario (que a su vez es la clave principal de una tabla), o un conjunto de atributos que forman una clave principal.
- En cada grupo de subtipos, definir todos los atributos de subtipo que se necesita conocer y que pertenecen a la table base y/o extendida de la clave primaria del grupo.

Más casos de uso

- Múltiples referencias

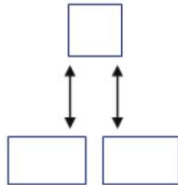
Directa



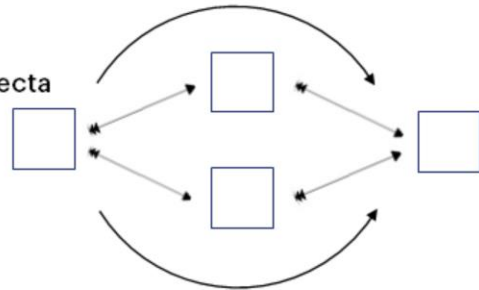
- Subtipos recursivos



- Especialización



Indirecta



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:



Videos

training.genexus.com

Documentation

wiki.genexus.com

Certifications

training.genexus.com/certifications