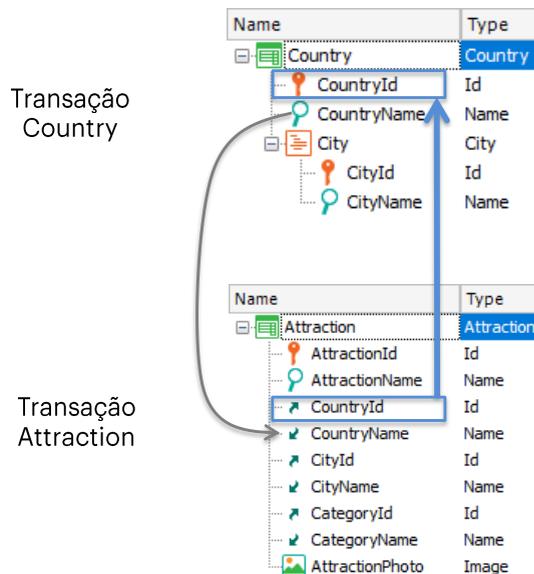


Nomes de atributos diferentes para o mesmo conceito

Grupo de subtipo

GeneXus 16

- GeneXus determina relações para atributos com o mesmo nome.



Até agora vimos que GeneXus estabelece relações entre transações -e entre tabelas- baseando-se nos nomes dos atributos iguais que encontra.

Por exemplo, na transação Attraction se encontra o atributo CountryId **com o papel chave estrangeira**, visto que com o mesmo nome está presente na transação Country, e ali não é chave primária.

Por sua parte, o atributo CountryName também é encontrado em ambas as transações com o mesmo nome, então GeneXus interpreta que se trata do mesmo atributo.

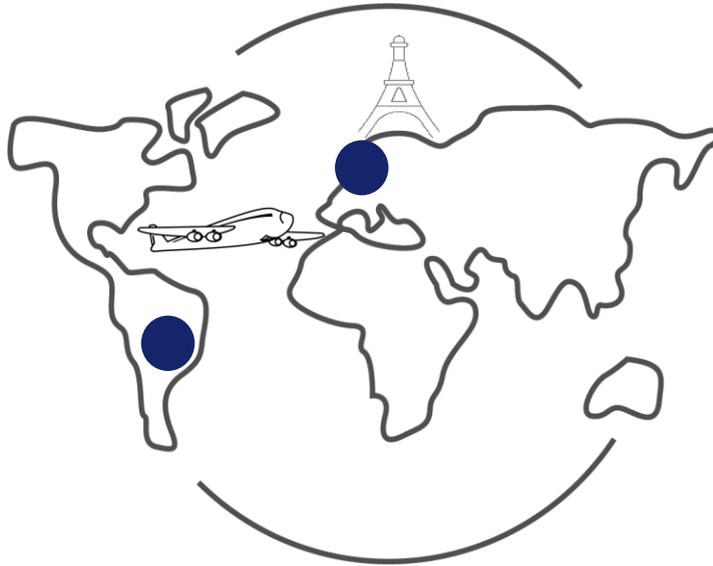
Neste caso não é um atributo primário, ou seja, chave primária ou parte de uma chave primária em uma tabela, portanto, GeneXus determinará que ele deva ser armazenado na tabela COUNTRY e não na tabela ATTRACTION.

Então GeneXus sempre pressupõe que se usarmos o mesmo nome de atributo, estamos representando o mesmo conceito.

No entanto, há casos em que talvez precisemos usar nomes diferentes para o mesmo conceito, e dizer a GeneXus que ambos os nomes **significam o mesmo**.

Requerimento

- Registrar os voos oferecidos aos clientes para chegar a uma atração turística particular

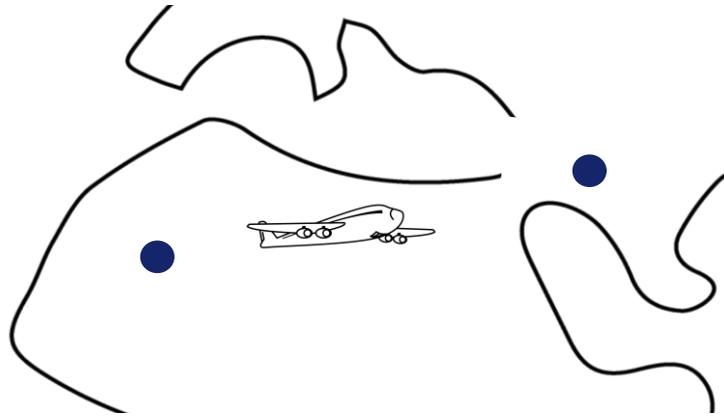


Vamos ver isto.

Suponhamos que na agência de viagens nos pedem para registrar os voos que oferecem aos clientes para chegar a uma atração turística.

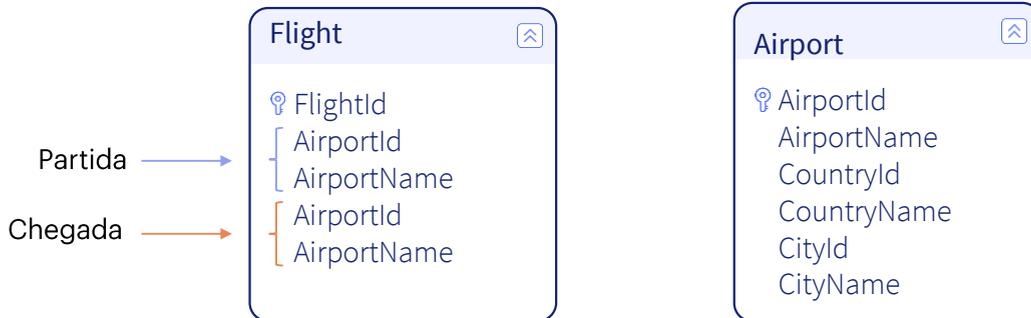
Requerimento

- Para cada voo, o aeroporto de partida, bem como o aeroporto de chegada



E devemos registrar para cada voo, o aeroporto de onde parte, bem como o aeroporto de chegada.

Necessidade de atributos com nomes diferentes para representar o mesmo conceito



Para representar isso, criaremos em primeiro lugar uma transação de nome: Flight

Definimos o atributo FlightId, que é automaticamente baseado no domínio: Id.

E agora vamos pensar sobre que **outras informações devemos registrar**.

Para representar isso, criaremos em primeiro lugar uma transação de nome Flight.

Definimos o atributo FlightId, que é automaticamente baseado no domínio: Id.

E agora vamos pensar sobre que **outras informações devemos registrar**.

Cada voo, como dissemos, terá um aeroporto de partida e um aeroporto de chegada...

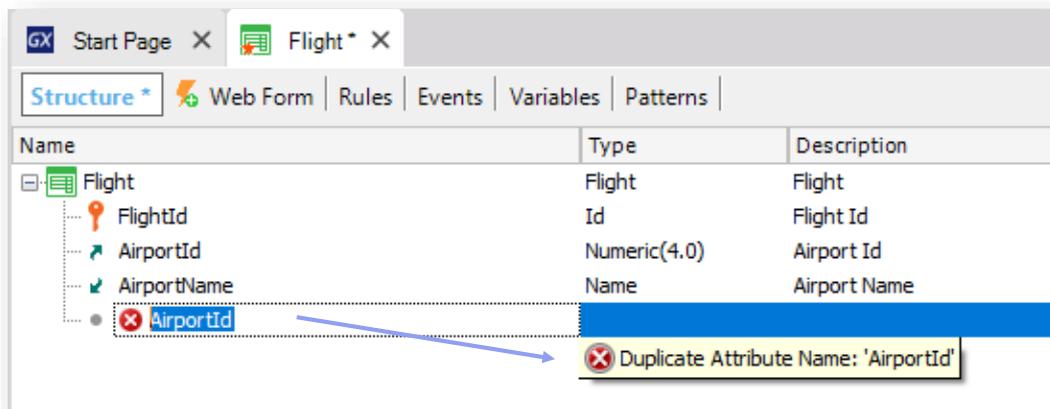
Mas para os aeroportos, teremos que ser capazes de registrá-los por si mesmos..., então podemos referenciá-los a partir dos voos.

Então vamos parar de trabalhar na transação Flight por um instante, e vamos criar outra transação de nome Airport. Definimos então que cada aeroporto tem um identificador AirportId, um nome AirportName e cada aeroporto se encontra em 1 país e em 1 cidade, assim adicionaremos os atributos: CountryId, CountryName, CityId e CityName. Salvamos...

E agora voltemos a ver qual era nossa necessidade na transação Flight.

Precisamos adicionar a cada voo, seu aeroporto de partida e seu aeroporto de chegada.

Nome duplicado



Então voltamos para a transação Flight, e vamos adicionar os atributos AirportId, e AirportName.... Mas quando tentamos adicionar novamente AirportId:

GeneXus nos diz que há um erro!, que estamos adicionando um atributo com nome duplicado.

E o mesmo vai acontecer com o atributo AirportName, que pensávamos adicionar para representar o nome do aeroporto de chegada.

Como podemos então incluir 2 aeroportos na mesma transação? Obviamente, teremos que usar **nomes de atributo diferentes** para armazenar as informações de origem e destino do voo que queremos registrar.

Definindo a solução ...



Nomes de atributos diferentes
¡Não há relação
entre transações!

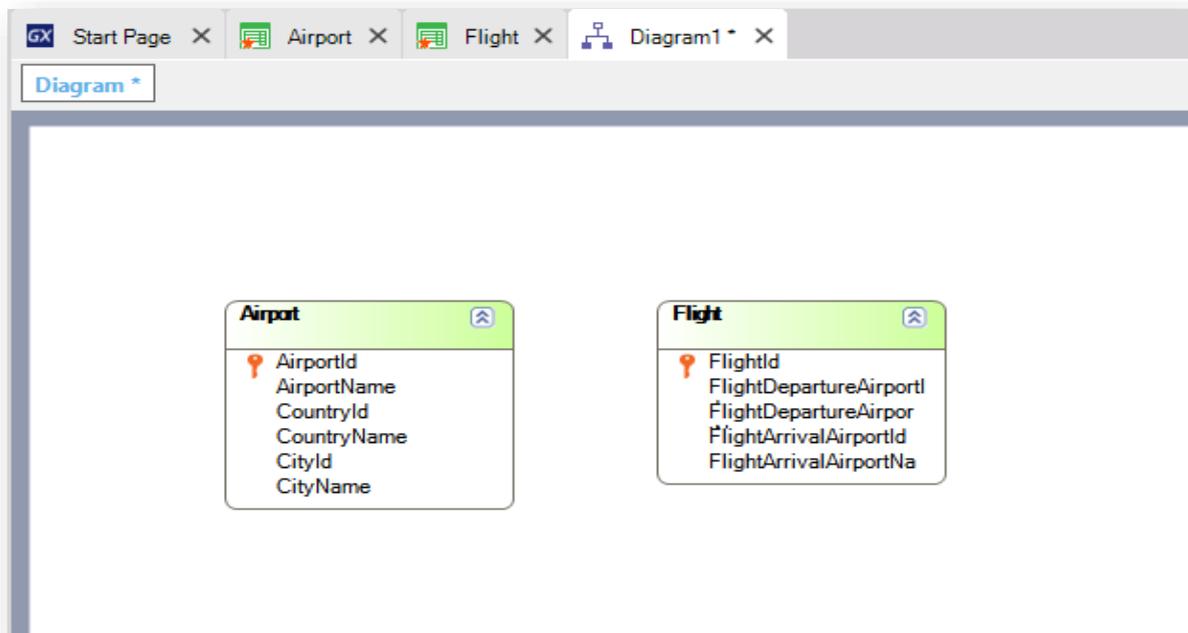
Então, eliminaremos os atributos que originalmente havíamos incluído e definiremos atributos com novos nomes.

Chamaremos de FlightDepartureAirportId ao identificador do aeroporto de origem do voo, e FlightDepartureAirportName ao nome do aeroporto de origem.

Bem, definimos novos nomes de atributo... mas para GeneXus esses nomes de atributo não estão relacionados a AirportId ou AirportName.

Como dissemos antes, se usamos nomes diferentes na transação Flight e na transação Airport para identificar o conceito de aeroporto, GeneXus não estabelecerá nenhuma relação entre as duas transações.

Diagrama

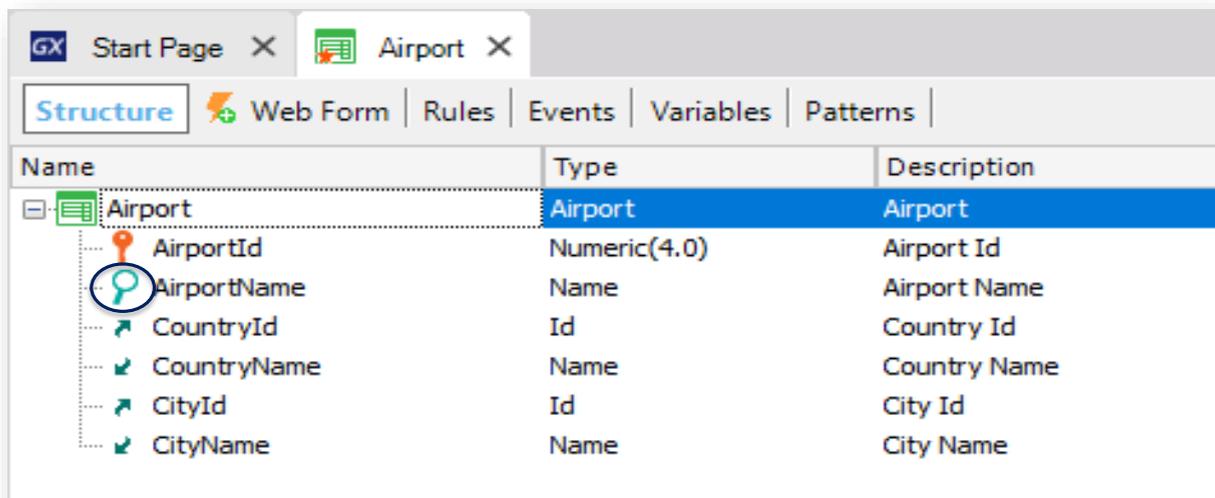


Para verificar isso acabamos de dizer, vamos criar um diagrama de transações.

E vamos arrastar as transações Airport e Flight e vemos que, de fato, GeneXus não encontra nenhuma relação entre elas, uma vez que nenhuma chave estrangeira foi identificada em Flight que permita a relação com o Airport.

Se tivesse encontrado relação apareceria uma seta entre as duas transações” .

Atributo descritor



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

Outra maneira de ver isto é prestar atenção à maneira como GeneXus nos mostra, na transação Flight, o atributo identificador do aeroporto.

Vemos que é sinalizado com este símbolo, que na transação Airport também está em AirportName.

Este símbolo está indicando que o atributo é o que melhor descreve o aeroporto em um caso, ou ao voo no outro. Quando criamos a estrutura de uma transação, GeneXus escolhe com base nos tipos de dados de seus atributos ao qual parece melhor descrevê-la, mas o usuário pode alterá-lo para outro, ou decidir que não há nenhum atributo desse tipo.

Atributo descritor

The screenshot displays the GeneXus interface. At the top, there is a search bar labeled 'Name' and a '+ INSERT' button. Below it is a table with the following data:

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

Overlaid on this is the 'Structure' window for the 'Flight' entity. It shows a tree view of attributes:

- Flight
 - FlightId (Id)
 - FlightDepartureAirportId (Id) - marked with a secondary attribute symbol (a circle with a dot)
 - FlightDepartureAirportName

A context menu is open over the 'FlightDepartureAirportId' attribute, showing options: 'Open Id', 'Delete' (Del), 'Toggle Key' (Ctrl+K), 'Toggle Description Attribute' (Ctrl+D), and 'Toggle Image Attribute'.

Este "atributo descritor" é usado por exemplo pelo pattern Work With para permitir a filtragem por ele, ordenar por ele, etc.

Recordemos do pattern aplicado a Attraction.

Para Flight não nos interessa que o aeroporto da partida seja o atributo que melhor descreva o voo, assim o removemos.

Vemos, então, que fica sinalizado com este símbolo, que indica que é um atributo secundário e não é considerado como chave estrangeira...

Chave estrangeira

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

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

Atributo secundário

Comparemos isso com a definição do identificador de país na transação Attraction

Em Attraction, o atributo CountryId tem este ícone que nos indica que ele é um atributo chave estrangeira... mas não é o caso do atributo FlightDepartureAirportId na transação Flight.

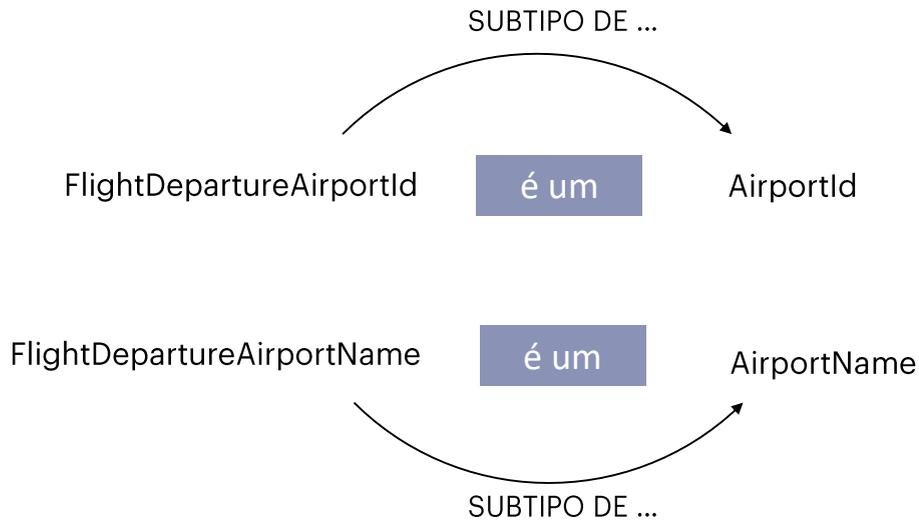
Então, como fazemos para que GeneXus possa associar nomes distintos para o mesmo conceito?

Precisamos que FlightDepartureAirportId mesmo se chamando diferente de AirportId, seja considerado como tal, ou seja, como um identificador de aeroporto.

E o mesmo vale para o nome do aeroporto.

Definindo a solução ...

- Por meio de subtipos, é possível criar dois atributos com nomes diferentes correspondentes ao mesmo conceito.



Como podemos conseguir isso?

A resposta é: **mediante definição de subtipos.**

Quando um atributo tem nome diferente de outro já definido, mas ambos representam o mesmo conceito, **podemos "dizer" para GeneXus** que o novo atributo é subtipo do outro

e a partir desse momento GeneXus vai considerá-los como se fossem exatamente a mesma coisa... portanto, GeneXus tratará o atributo FlightDepartureAirportId **exatamente como se fosse um AirportId, ou seja, irá identificá-lo como chave estrangeira na transação Flight.**

E o mesmo teremos com FlightDepartureAirportName: indicaremos que é subtipo de AirportName.

The screenshot shows three panels in the GeneXus IDE:

- FlightDepartureAirport**: A subtype group with attributes FlightDepartureAirportId (supertype: AirportId) and FlightDepartureAirportName (supertype: AirportName).
- FlightArrivalAirport**: A subtype group with attributes FlightArrivalAirportId (supertype: AirportId) and FlightArrivalAirportName (supertype: AirportName).
- Flight**: A main object with attributes FlightId (type: Id), FlightDepartureAirportId (type: Id), FlightDepartureAirportName (type: Name), FlightArrivalAirportId (type: Id), and FlightArrivalAirportName (type: Name).

Arrows indicate that FlightDepartureAirportId and FlightArrivalAirportId inherit from AirportId, and FlightDepartureAirportName and FlightArrivalAirportName inherit from AirportName.

S
O
L
U
Ç
Ã
O

Vamos agora para a transação Flight e observamos que o atributo FlightDepartureAirportId tem o símbolo indicando que será tratado como uma chave estrangeira... e também o símbolo da letra s, que indica que ele é um atributo definido como subtipo.

Vamos agora proceder da mesma forma para definir os atributos que permitam registrar o aeroporto onde o voo chega.

Vamos definir então os atributos FlightArrivalAirportId e FlightArrivalAirportName.

E gravamos.

Criamos agora um novo objeto, de tipo: Subtype group e colocamos como nome FlightArrivalAirport.

Digitamos o ponto ('.')... GeneXus nos sugere os atributos que começam com "FlightArrivalAirport" e escolhemos FlightArrivalAirportId.

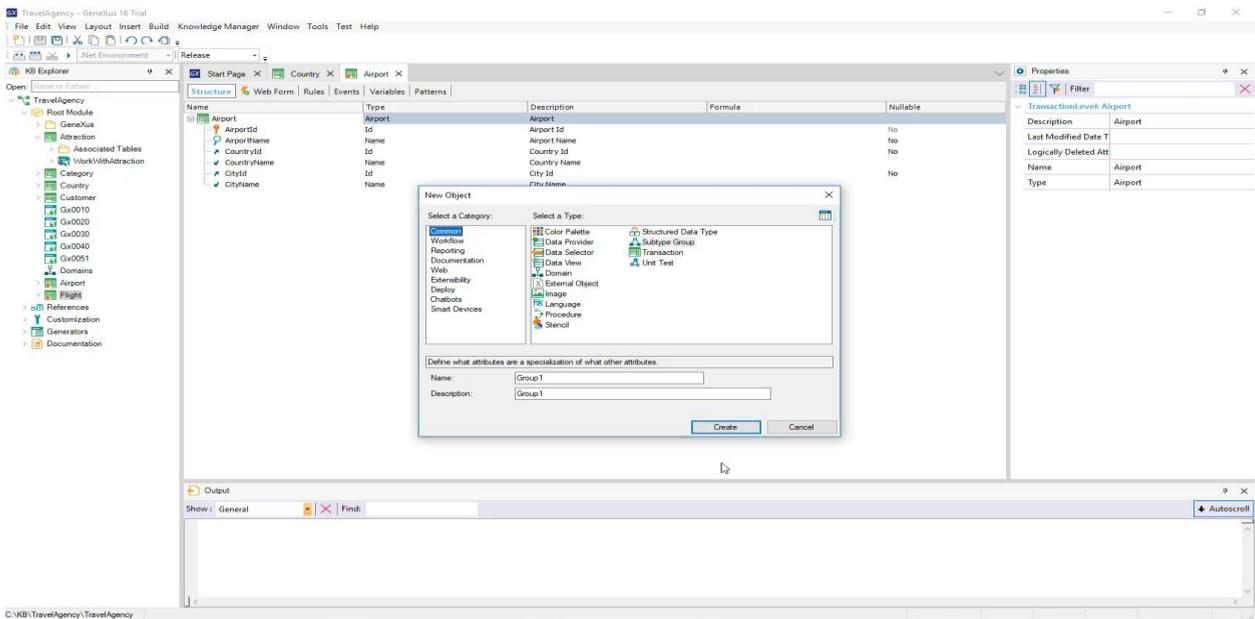
Pressionamos Tab e declaramos que seja subtipo de AirportId.

Agora adicionamos FlightArrivalAirportName... e definimos que seu supertipo é: AirportName.

Gravamos.

Vejam novamente a estrutura da transação Flight...

DEMO



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

Vamos ver isso na prática.

Para definir subtipos, a primeira coisa que precisamos fazer é criar um grupo de subtipos.

Assim, criamos um novo objeto do tipo Subtype group, e colocamos como nome FlightDepartureAirport:

Agora, na primeira linha digitamos a tecla com o ponto ('.') e GeneXus nos sugere os atributos que começam com "FlightDepartureAirport", que já havíamos definido na transação Flight.

Escolhemos então FlightDepartureAirportId.... pressionamos Tab e como queremos que FlightDepartureAirportId seja subtipo de AirportId, escolhemos como supertipo o atributo AirportId.

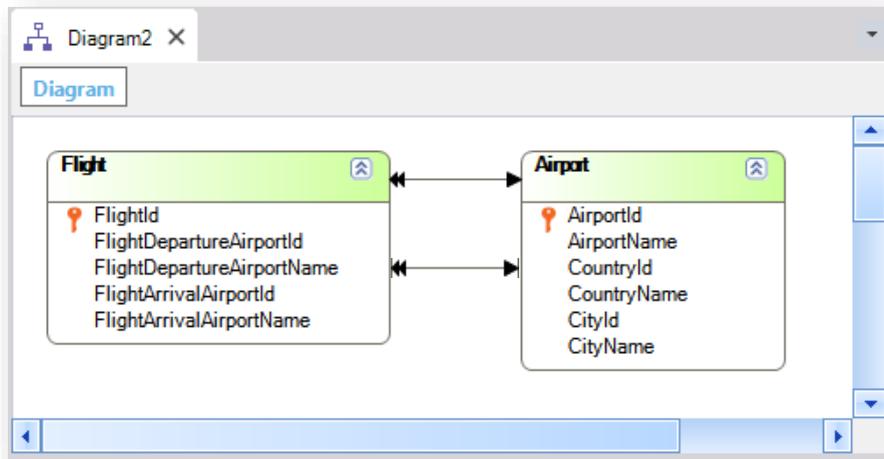
Podemos dizer que o supertipo é o atributo original, e subtipo é o atributo que conceitualmente corresponde a esse atributo original, mas que tem outro nome.

Agora adicionamos FlightDepartureAirportName, e definimos que seu supertipo é: AirportName

Gravamos.

Esse atributo se torna o identificador deste grupo de subtipos, portanto, o chamamos de "primário", e todos os atributos que adicionamos neste grupo, como FlightDepartureAirportName, dependerão dele, como na transação.

Diagrama: GeneXus encontrou uma relação



¡Os dados serão verificados quanto à integridade referencial em tempo de execução!

E voltamos agora a analisar o diagrama de transações que havíamos criado antes.

Vemos que agora GeneXus sim coloca a seta: GeneXus considera os atributos subtipos de identificadores de aeroporto em Flight, exatamente como se tivéssemos referenciado AirportId.

Vemos então que GeneXus encontrou a relação entre o Flight e o Airport.

Observemos que, embora em GeneXus esteja aparecendo uma única seta no diagrama, sendo rigorosos deveriam aparecer duas.

DEMO

The screenshot shows the GeneXus IDE interface. A dialog box titled "Database needs to be reorganized" is open, displaying details for a table named "Flight". The dialog includes sections for "Table Structure", "Indexes", and "Foreign key constraints". The "Table Structure" section lists attributes: "FlightId" (Numeric (4), Not null, Autonumber), "FlightDepartureAirportId" (Numeric (4), Not null), and "FlightArrivalAirportId" (Numeric (4), Not null). The "Indexes" section lists "IFLIGHT" (primary key, Clustered), "IFLIGHT1" (duplicate), and "IFLIGHT2" (duplicate). The "Foreign key constraints" section shows a constraint from "Flight" to "Airport" on the "FlightDepartureAirportId" attribute. The "Reorganize" button is highlighted in the dialog. The background shows the GeneXus project explorer and a status bar indicating "0 Errors, 0 Warnings, 2 Success".

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

Vejamos em funcionamento tudo isto. Pressionemos F5...

O banco de dados deverá ser reorganizado pois deverão criar-se tabelas Flight e Airport. Concordamos, então, pressionamos Reorganize.

Em primeiro lugar definiremos aeroportos, então por isso executamos a transação Airport.

Incluímos o aeroporto "Guarulhos", indicamos o país: Brasil e a cidade: São Paulo.

Agora vamos registrar o aeroporto "Charles de Gaulle"... selecionamos: França e a cidade: Paris.

Confirmamos.

Agora vamos registrar um voo.

Executamos a transação Flight... Como Aeroporto de partida vamos escolher "Guarulhos" e como aeroporto de chegada: "Charles de Gaulle".

Vemos que as etiquetas dos atributos não nos indicam que este é o aeroporto de partida e este de chegada. Se formos ao form da transação em GeneXus e nos posicionamos no campo do primeiro aeroporto, vemos que o que diz o rótulo, ou seja, seu Caption, está sendo extraído da propriedade ContextualTitle do atributo. Se vamos pesquisar é aqui onde ele aparece:

Lhe adicionamos "Departure".

E fazemos o mesmo com o nome e adicionamos "Arrival" ao aeroporto de chegada.

Vemos o Form.

Pressionamos F5 novamente.

Vamos agora incluir outro voo.

Tentamos digitar no Aeroporto, 15... aparece o aviso de que este aeroporto não existe.

A consistência dos dados está sendo controlada. E a lista de seleção está sendo oferecida.....
Vemos, assim, que temos os mesmos controles e ajudas que apareceram quando os atributos eram chaves estrangeiras com seus nomes originais... como vimos neste vídeo... **mas agora se trata de atributos subtipos deles.**

E esta é justamente a ideia: **que definindo subtipos que conseguimos definir que nomes de atributos diferentes correspondem ao mesmo conceito.**

É por isso que este atributo e este outro serão interpretados como chaves estrangeiras e que estes serão inferidos a partir dos primeiros.

Mas como é que GeneXus sabe que o nome deste aeroporto deve ser inferido neste e não a partir deste?

The screenshot displays the 'Structure' view of a 'Flight' entity in GeneXus. It shows the main entity and two subtypes: 'FlightDepartureAirport' and 'FlightArrivalAirport'. Each entity is represented by a table with columns for Name, Type, Description, Formula, and Nullable. The 'Flight' entity has five attributes: FlightId (Id), FlightDepartureAirportId (Id), FlightDepartureAirportName (Name), FlightArrivalAirportId (Id), and FlightArrivalAirportName (Name). The 'FlightDepartureAirport' subtype has two attributes: FlightDepartureAirportId (AirportId) and FlightDepartureAirportName (AirportName). The 'FlightArrivalAirport' subtype has two attributes: FlightArrivalAirportId (AirportId) and FlightArrivalAirportName (AirportName). A blue arrow points from the 'FlightArrivalAirportId' attribute in the 'FlightArrivalAirport' group to the 'FlightArrivalAirportId' attribute in the 'Flight' entity.

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

Não é pela ordem em que os atributos aparecem nem pelo nome que lhes demos. Sabe disso porque esse atributo foi definido em um grupo com este. E este, em **outro** grupo, com este outro.

Portanto, é essencial agrupar no mesmo grupo de subtipos os atributos que se correspondem.

O uso de subtipos permitiu-nos representar uma situação que ocorre na realidade: neste caso, um voo tem dois aeroportos que cumprem um papel diferente: um é o aeroporto de partida e outro é o aeroporto de chegada. E os grupos nos permitem diferenciar ambos os papéis e conectar os atributos de cada papel.

Vamos destacar que não incluímos todos os subtipos no mesmo grupo, nem os dois atributos de subtipos primários de um lado em um grupo e os dois atributos secundários em outro. Não. Agrupamos os atributos que definem o aeroporto de partida juntos em um grupo e os atributos que definem o aeroporto de chegada juntos em outro grupo.

Grupos de subtipo

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

Repito: isto é assim porque GeneXus entende com este grupo que quando se insere valor para este identificador de aeroporto `FlightDepartureAirportId` o nome do aeroporto correspondente a **este identificador**, deve ser carregado neste atributo `FlightDepartureAirportName` e não no outro nome de aeroporto que está na transação.

Da mesma forma, GeneXus entende que ao digitar valor para o identificador do aeroporto: `FlightArrivalAirportId` o nome do aeroporto correspondente deve ser carregado no atributo `FlightArrivalAirportName`.

Grupos de subtipo

Flight

« < > » SELECT

Id

Departure Airport Id No matching Flight Departure Airport.

Departure Airport Name

Arrival Airport Id

Arrival Airport Name

CONFIRM CANCEL

Flight

« < > » SELECT

Id

Departure Airport Id **FK**  

Departure Airport Name Guarulhos

Arrival Airport Id **FK**  

Arrival Airport Name Chales de Gaulle

CONFIRM CANCEL DELETE

- Controle IR
- Lista de seleção
- Inferências agrupadas

E se, para cada aeroporto, quisermos ver o país e a cidade?

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	...

Subtype	Description	Supertype	Descr...
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...

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

Bem. Agora vamos supor que na transação Flight, queremos para cada aeroporto, ver, além de seu nome, seu país e sua cidade.

Isso simplesmente se resolve **definindo mais atributos de subtipos em cada grupo, ajudando o desenvolvedor, dando-lhes nomes significativos e não se esquecendo de indicar quem são seus supertipos...** e desta forma GeneXus vai entender que para o subtipo primário do grupo, deverá inferir todo o resto da informação associada.

Vamos fazê-lo.

Neste grupo de subtipos, iremos definir os atributos.

FlightDepartureCountryId como subtipo de CountryId,
 FlightDepartureCountryName como subtipo de CountryName,
 FlightDepartureCityId como subtipo de CityId
 e FlightDepartureCityName como subtipo de CityName.

Gravamos.

E agora vamos adicionar estes novos atributos à estrutura da transação Flight.

Gravamos.

E fazemos o mesmo para o outro grupo de subtipos....

Definimos

FlightArrivalCountryId como subtipo de CountryId,

FlightArrivalCountryName subtipo de CountryName,

FlightArrivalCityId subtipo de CityId

e FlightArrivalCityName subtipo de CityName.

Salvamos e também os adicionamos à estrutura da transação de Flight.

Gravamos.

Como fizemos antes, a cada atributo adicionado alteramos o título contextual para que o rótulo na tela indique a função:

Novamente pressionamos F5 para executar a aplicação.

Transação 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 a transação Flight, consultamos nosso primeiro voo e podemos ver de cada aeroporto seu país e sua cidade.

Desta forma, vimos como resolver uma dupla referência ao mesmo conceito, mas com diferentes papéis, uma vez que os dois aeroportos deviam ser obtidos a partir da mesma tabela mas cada um tinha um papel diferente.

Outras possíveis soluções

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

Definindo apenas um grupo de subtipos:
"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

Definindo apenas um grupo de subtipos:
"FlightArrivalAirport"

Por último, é importante saber que **estas** poderiam também ter sido soluções válidas.

Nesta primeira proposta foi definido um único grupo de subtipos: o grupo correspondente ao aeroporto de partida... e foi deixado aos atributos com o nome original (airportid, airportname, countryId, etc.) para a entrada do destino.

E isso é completamente válido.

Nesta segunda proposta... um único grupo de subtipos foi definido também, mas neste caso para o grupo que corresponde ao **aeroporto da chegada**.

Em resumo, subtipos nos permitem dizer a GeneXus como associar diferentes nomes de atributo a um mesmo conceito. E como vimos, as validações e todo o comportamento dos subtipos serão idênticos como se tivéssemos usado os atributos originais.

Há muitos outros casos em que é necessário modificar o nome de um atributo para evitar conflito ou ambiguidade. Neste vídeo vimos o caso de referências múltiplas de uma tabela a outra, mas estas referências não precisam ser diretas.

Resumo e observações

Os subtipos nos permitiram representar a realidade em que um voo tem dois aeroportos com diferentes funções.

Os grupos de subtipos nos permitem definir os dois papéis separados (não definimos um único grupo com todos os subtipos).

A forma correta de fazer isso é:

- Definir um grupo de subtipos para cada conjunto de atributos correspondentes um ao outro.
- Cada grupo de subtipos deve necessariamente incluir um subtipo de atributo principal (que, por sua vez, é a chave primária de uma tabela) ou um conjunto de atributos que formam uma chave primária.
- Em cada grupo de subtipos, defina todos os atributos de subtipo que precisam ser conhecidos e que pertencem à tabela base e/ou a estendida da chave primária do grupo.

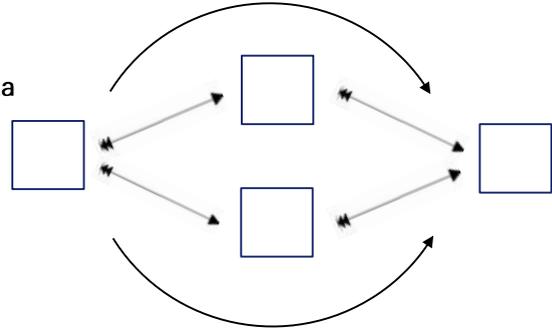
Mais casos de uso

- Múltiplas referências

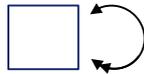
Direta



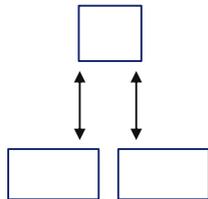
Indireta



- Subtipos Recursivos



- Especialização



A partir desta tabela temos duas maneiras de chegar a esta outra tabela. Então foram necessários subtipos para diferenciá-los.

Temos também o caso de uma entidade que deve referenciar a si mesma, por exemplo, uma transação de funcionários, na qual entre as informações do funcionário deve ser registrado o chefe, que também é um funcionário.

Ou o caso em que temos uma entidade que registra informações gerais, por exemplo, de pessoas (como o nome, número de telefone, endereço, etc.) e então temos entidades que são uma especialização do outro, por exemplo, clientes e passageiros, que em particular são pessoas.

Para finalizar, atualizamos as alterações no GeneXus Server. Adicionamos os comentários e pressionamos Commit.

GeneXus™

The power of doing.

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