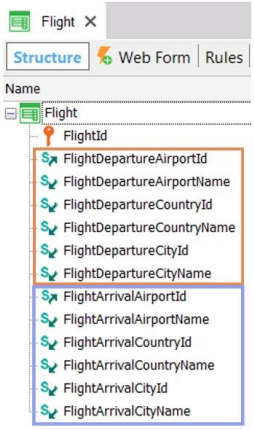
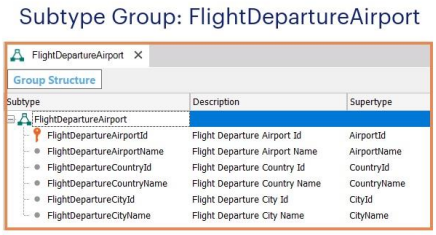
Subtipos

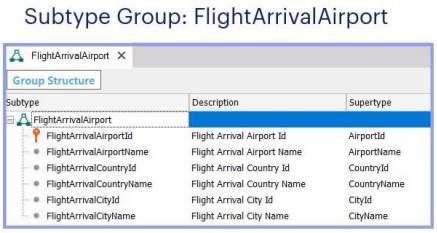
Referência Múltipla e Especialização

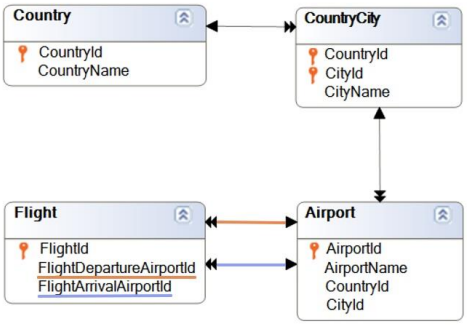
GeneXus™

Revisão









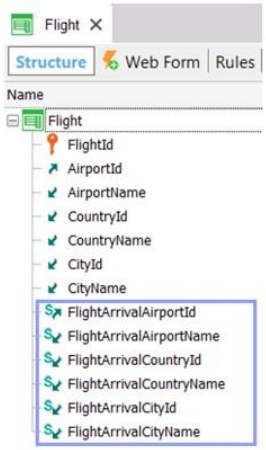
No vídeo anterior vimos a necessidade de definir grupos de subtipos, já que tínhamos em uma transação uma referência dupla a um mesmo ator da realidade.

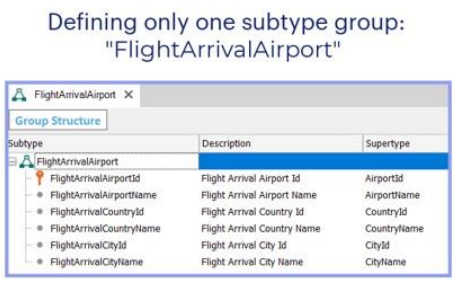
Era o caso da transação Flight, na qual tínhamos para cada voo, um aeroporto de partida e um aeroporto de chegada. Não podíamos incluir na estrutura da transação duas vezes o mesmo atributo, AirportId, já que GeneXus nos dava um erro por adicionar um nome de atributo duplicado. Por essa razão, decidimos definir dois grupos de subtipos: “FlightDepartureAirport”, para identificar o aeroporto de partida, e “FlightArrivalAirport”, para identificar o aeroporto de chegada.

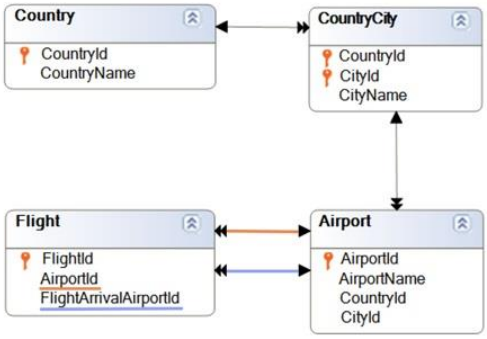
Como queríamos inferir o país e cidade de cada aeroporto, definimos em cada grupo, subtipos também dos atributos correspondentes a país e cidade, assim como também ao nome do aeroporto.

Portanto, quando na transação Flight, nomeamos FlightDepartureCountryName, sabemos que será um CountryName inferido através do aeroporto de partida: FlightDepartureAirportId. Estes subtipos foram definidos dentro do mesmo grupo e portanto foi estabelecida a associação e relação entre eles.
E quando na transação Flight, nomeamos FlightArrivalCountryName, sabemos que será inferido através do atributo FlightArrivalAirporId.
Ou seja, que não há nenhuma ambiguidade. Temos dois caminhos perfeitamente diferenciados para chegar a Country a partir de Flight.

Revisão

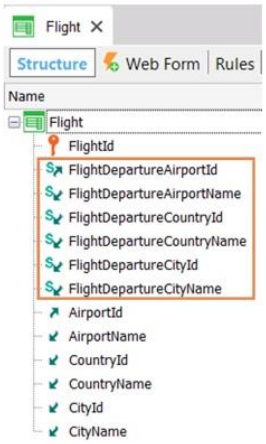


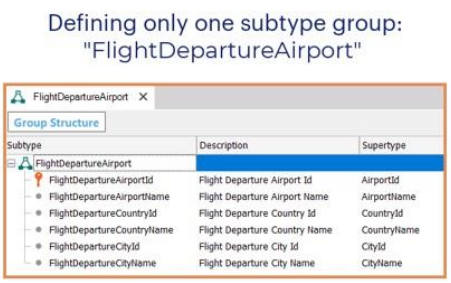


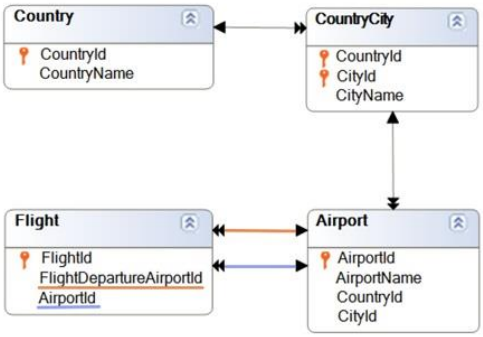


Outra opção era deixar o atributo de nome AirportId para o papel do aeroporto de partida e definir um grupo de subtipos para identificar o aeroporto de chegada.

Revisão



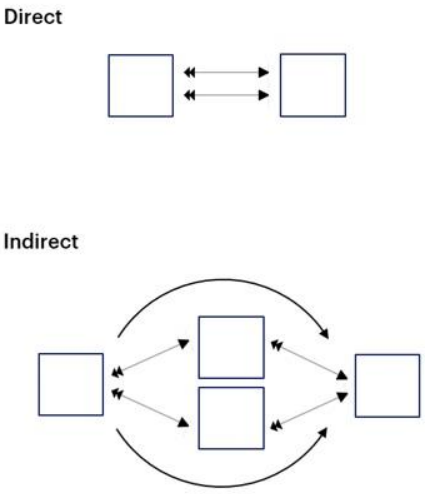




ou do contrário, deixar o atributo de nome AirportId para o papel do aeroporto de chegada e definir um grupo de subtipos para identificar o aeroporto de partida.

Em ambas as opções, o modelo de dados refletirá as mesmas relações que na solução anterior.

Referência Múltipla



No vídeo anterior, vimos então, o caso de referências múltiplas de uma tabela a outra relacionada diretamente com ela.

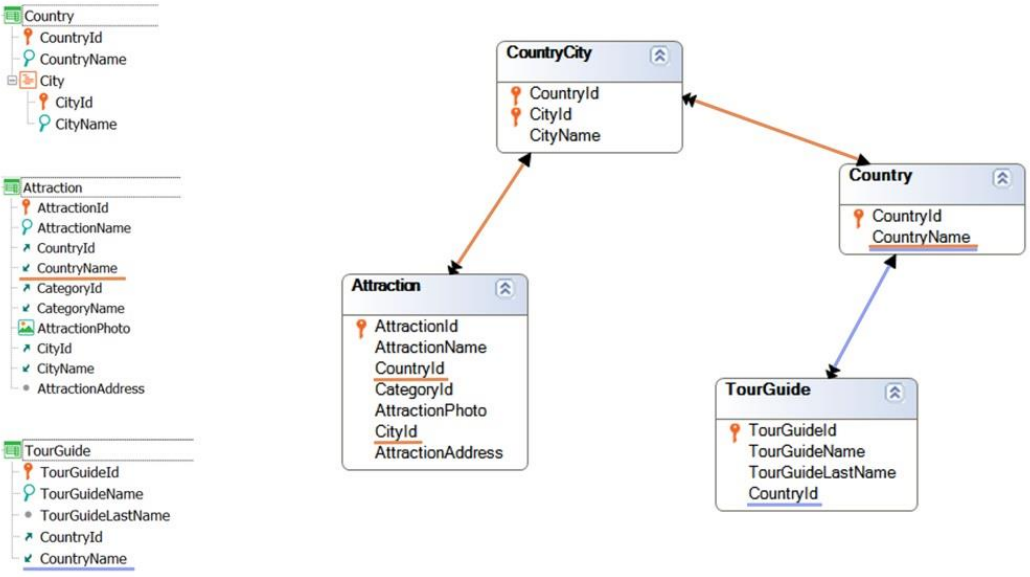
Mas estas referências não precisam ser diretas.

A partir de uma tabela podemos ter dois caminhos para chegar a outra, tratando-se de uma relação indireta, para o que serão necessários subtipos para diferenciá-los.

Nova transação para cadastrar guias turísticos

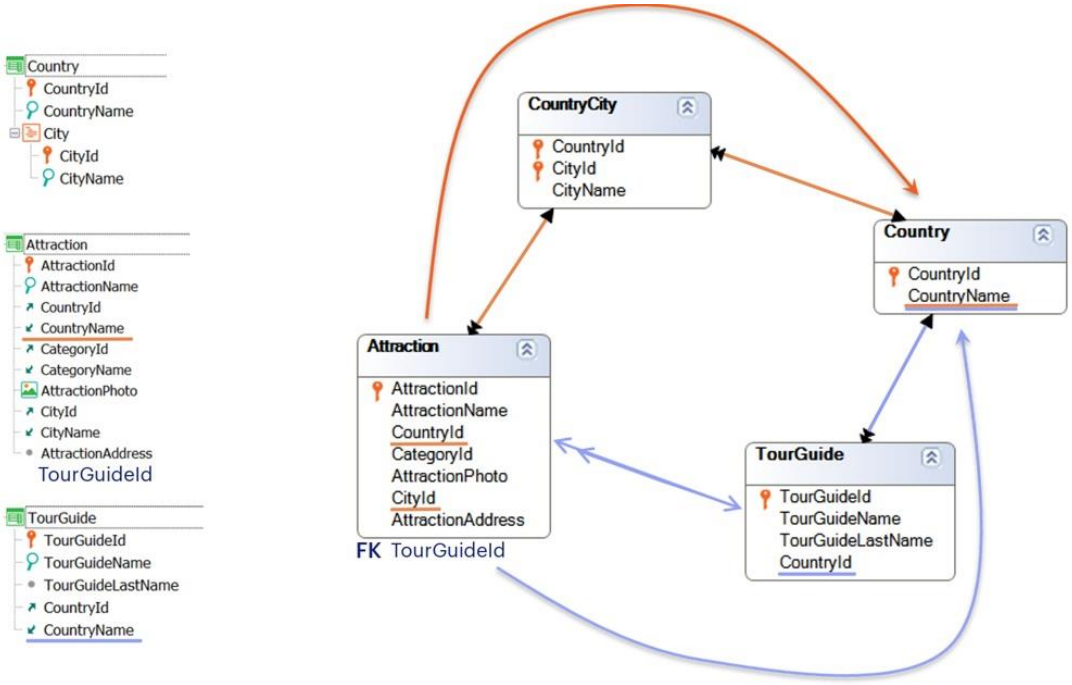
Suponhamos que devemos adicionar uma transação para registrar a informação dos guias turísticos. Cada guia tem uma nacionalidade determinada, e por esse motivo adicionamos à estrutura de sua transação, o atributo CountryId.

Nova transação para cadastrar guias turísticos



Se observarmos o diagrama de tabelas, da transação Attraction podemos inferir o CountryName correspondente a essa atração, já que se encontra em sua tabela estendida. Analogamente, da TourGuide também podemos inferir seu próprio CountryName, ou seja, o país do guia.

Guia de Turismo: referência múltipla



Se agora vinculamos as entidades Attraction e TourGuide adicionando à primeira o atributo identificador do guia, TourGuideId para representar que uma atração turística tem um guia atribuído e só um, como se relacionam agora as tabelas?

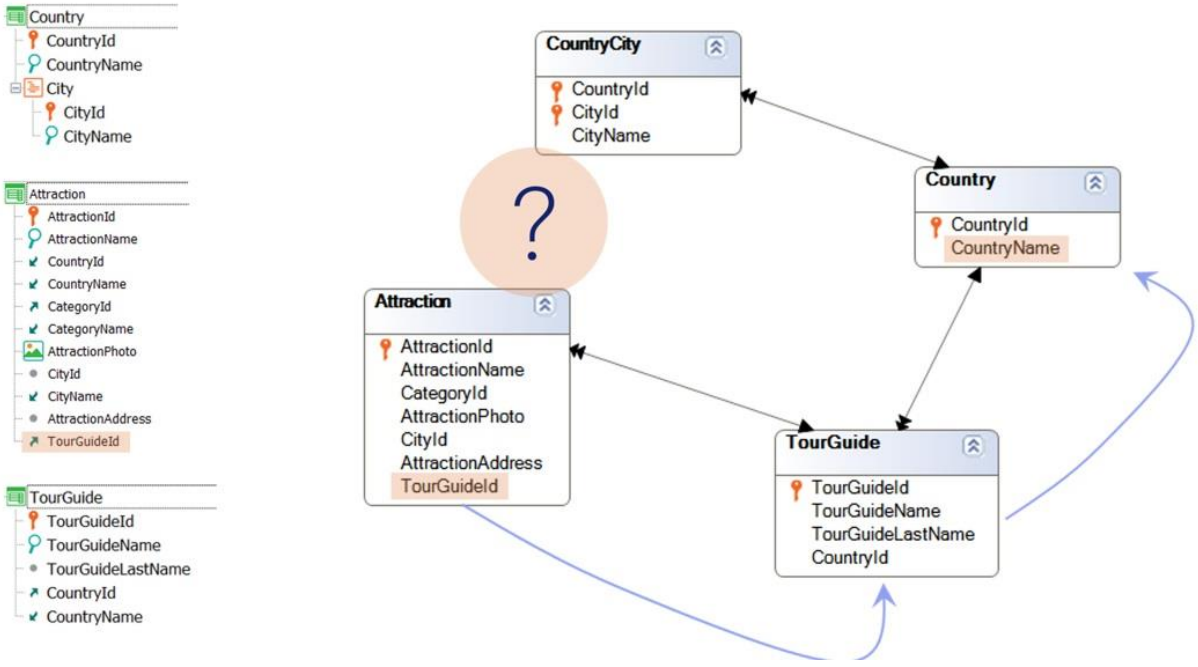
TourGuideId será uma chave estrangeira em Attraction para a tabela TourGuide, de modo que aparecerá a relação marcada com a seta azul. Isso pode nos levar a pensar que da Attraction, agora existem dois caminhos para inferir CountryName, o que significa que há uma ambiguidade.

Em outras palavras, uma vez que GeneXus tem o atributo CountryName em Attraction, de onde o infere? da cidade da atração ou do guia turístico da atração? Se ambos os valores coincidissem, não teria importância, mas neste caso não têm que coincidir. O país onde se encontra a atração não precisa coincidir com o país natal do guia turístico.

Para poder diferenciar os dois papéis de CountryName, precisaremos utilizar subtipos.

Mas neste caso, não precisamos deles apenas para diferenciar os papéis.

Guia de Turismo: referência múltipla



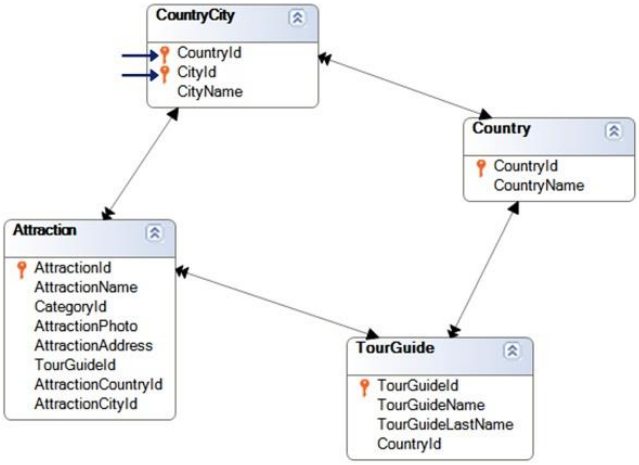
Observemos o diagrama de tabelas depois de adicionar o atributo TourGuideId na transação Attraction: vemos que desaparece a relação entre Attraction e CountryCity, pois CountryId já não é uma chave estrangeira em Attraction. Observemos que não está mais na tabela, ou seja, será um atributo inferido. Mas inferido a partir de quem? De TourGuideId!

Não esqueçamos que antes de mais nada GeneXus normaliza as tabelas. Ou seja, com base nos nomes dos atributos, em conjunto com os identificadores, determina qual atributo é colocado em cada tabela e as relações entre elas. Como em Attraction aparece TourGuideId, que é identificador de TourGuide, e ao mesmo tempo em TourGuide aparece CountryId, que é identificador de Country, está entendendo que dado um TourGuideId em Attraction, o CountryName o infere a partir dele, passando pela tabela intermediária TourGuide.

Portanto, com este desenho de transações perdemos a possibilidade de indicar qual é o país da atração, pois o CountryName será o do guia turístico.

Não temos outra alternativa senão utilizar subtipos para conseguir que Attraction tenha um país próprio, independente do país do guia turístico.

Guia de Turismo: referência múltipla



Criaremos um grupo de subtipos para representar o país e a cidade da atração, já que é em Attraction onde o problema ocorre.

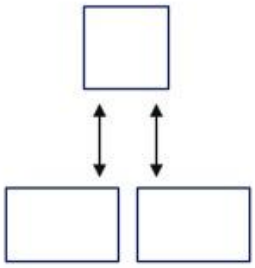
Observemos que, agora sim, GeneXus representa corretamente as relações no diagrama de tabelas e, além disso, o atributo CountryName agora se infere a partir de TourGuideId, sem ambiguidade. O atributo que representa e no qual se infere o país da atração será o de nome AttractionCountryName, subtipo de CountryName, pertencente ao grupo AttractionCountryCity.

Também observemos que este grupo tem dois atributos primários: AttractionCountryId e AttractionCityId, que correspondem à chave primária da tabela CountryCity, pelos supertipos que indicamos: {CountryId, CityId}.

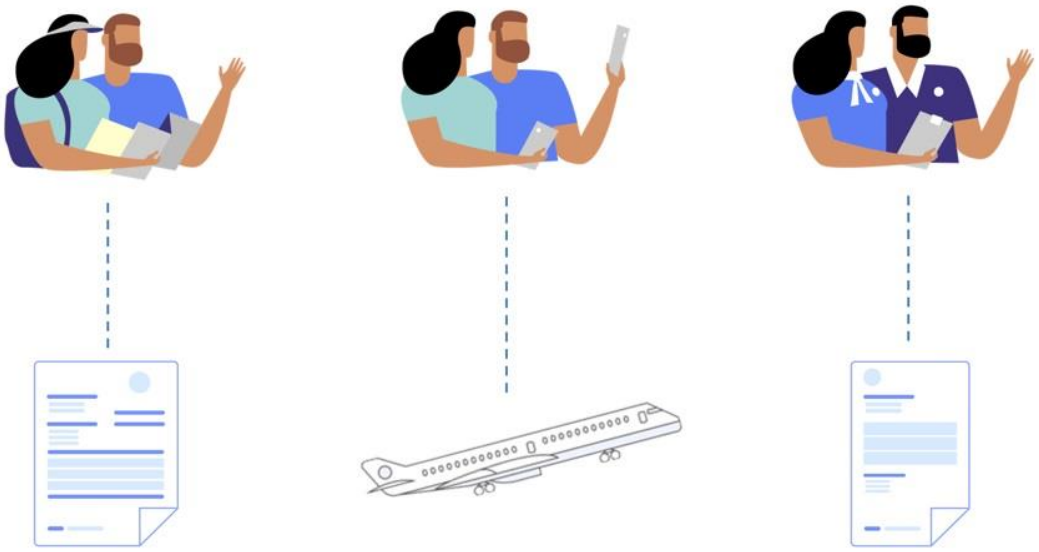
Como vimos, esta solução resolve o problema, mas devemos levar em conta que se já existem outros objetos que para consultar o país da atração, já utilizaram os atributos originais CountryId e CountryName, é possível que tenhamos que modificá-los nesses objetos e utilizar agora os atributos subtipos AttractionCountryId e AttractionCountryName respectivamente.

Existem outras soluções possíveis que não vamos estudar neste curso.

Especialización



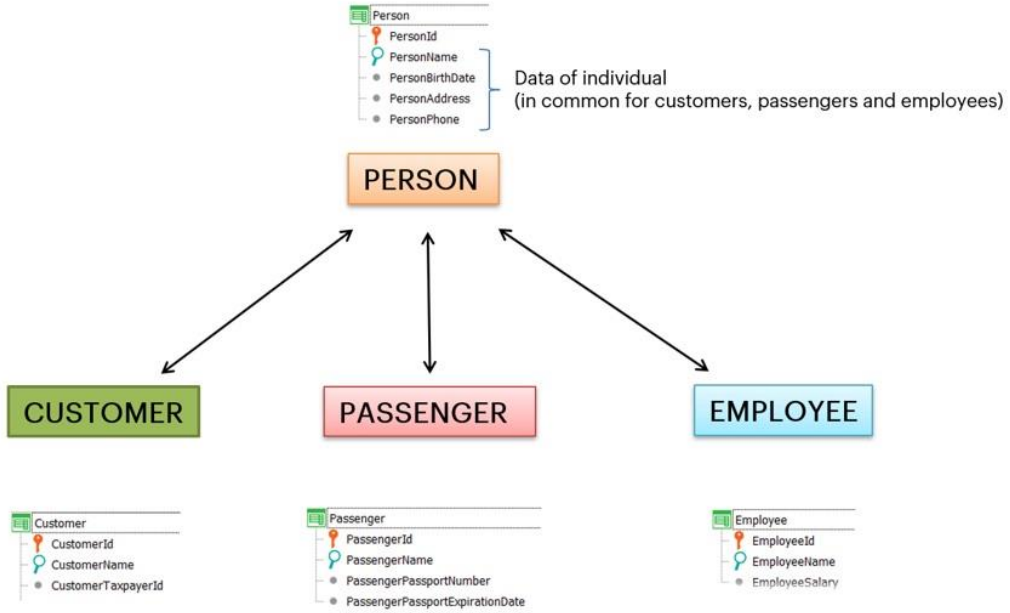
Vejamos agora outro caso de uso de subtipos, ao qual chamamos especialização, em que temos uma transação que registra informação geral e, depois, temos transações com informação particular, que são uma especialização dessa outra.



Suponhamos que a agência de viagens necessita trabalhar com informações específicas dos clientes a quem vende passagens e pacotes turísticos (por exemplo, seu número de contribuintes no escritório de impostos do Estado, se tiver), informações específicas dos passageiros (por exemplo, seu número de passaporte e validade do mesmo) e também informações específicas dos funcionários da agência, para os quais deverá registar, por exemplo, o seu salário.

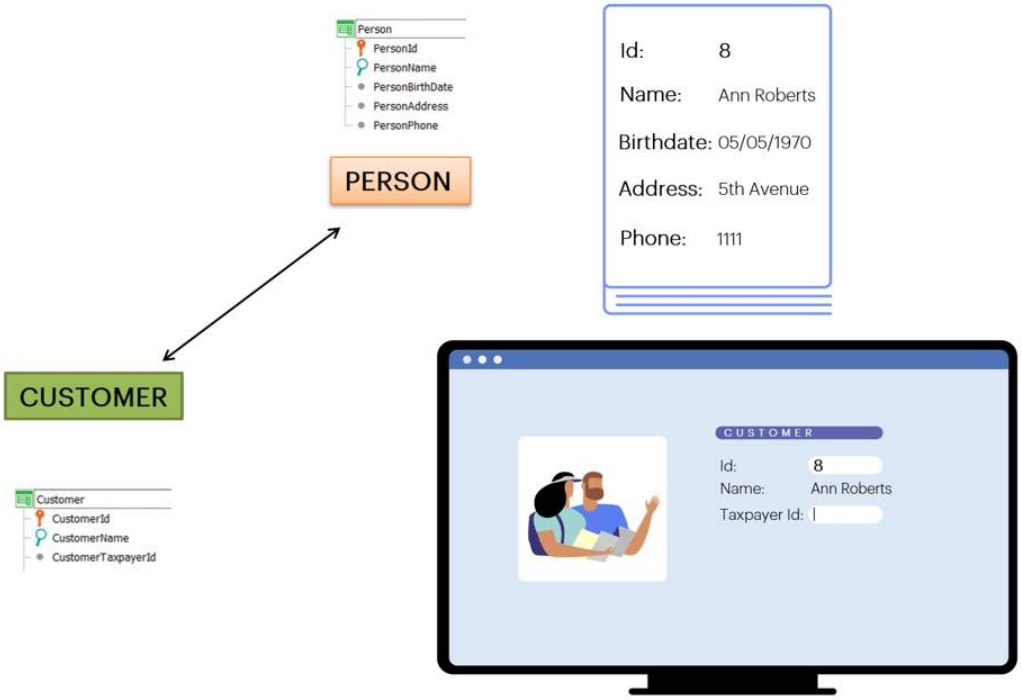
Ou seja, a agência de viagens emitirá faturas aos clientes, registrará nos assentos dos voos a passageiros e emitirá recibos de salário aos funcionários.

Especialização

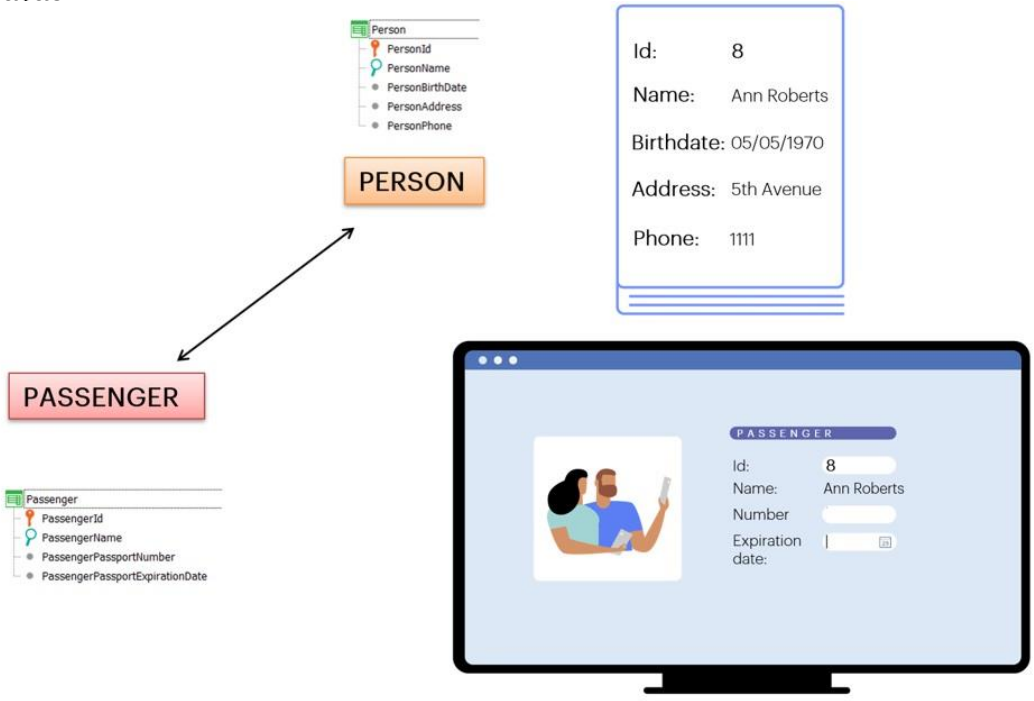


Poderíamos definir então, em vez do que antes era somente uma transação, Customer, uma transação chamada Person que manipule a informação que é comum para todas as pessoas (por exemplo, um nome, uma data de nascimento, um endereço, um telefone, etc) e transações que sejam especializações de Person, já que tanto os clientes, como os passageiros, como os empregados, SÃO pessoas.

Cada especialização terá seus dados específicos (Customer terá o número de contribuinte; Passenger, o número do passaporte e data de validade do mesmo e Employee, o salário).

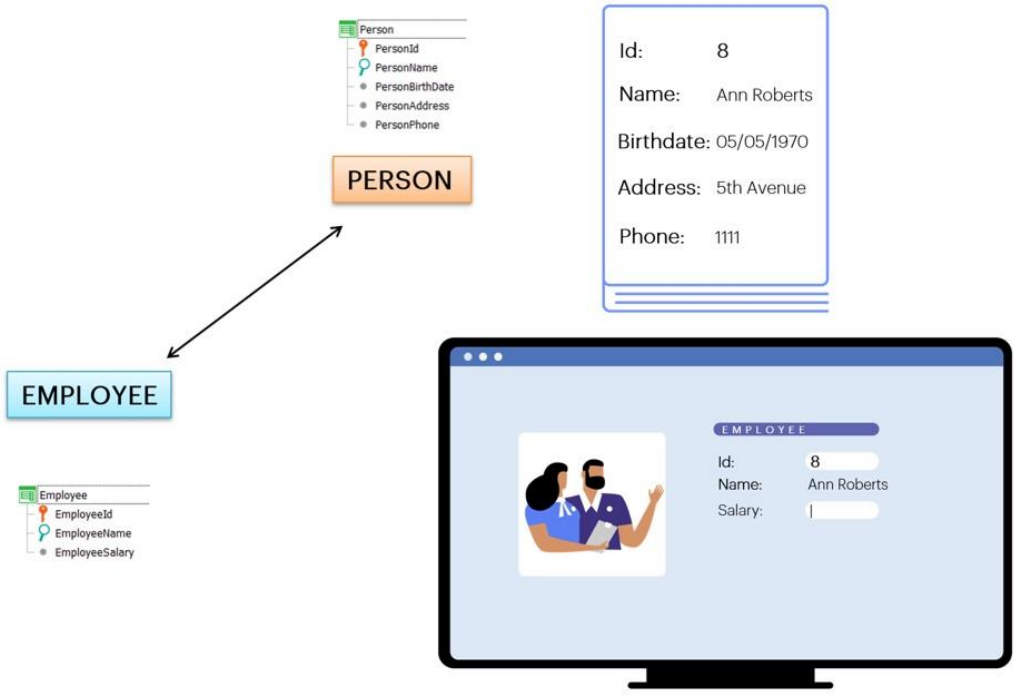


Queremos que o identificador de cliente coincida exatamente com o de uma pessoa, para refletir que o cliente é uma pessoa. Ou seja, se a pessoa de id 8 se chama Ann Roberts, e nasceu em 05/05/1970, quando vamos adicionar suas informações como cliente, precisamos que o usuário possa digitar na transação Customer o id 8, e que ao sair do campo lhe mostre o nome Ann Roberts e possa adicionar o número de contribuinte no atributo CustomerTaxpayerId.

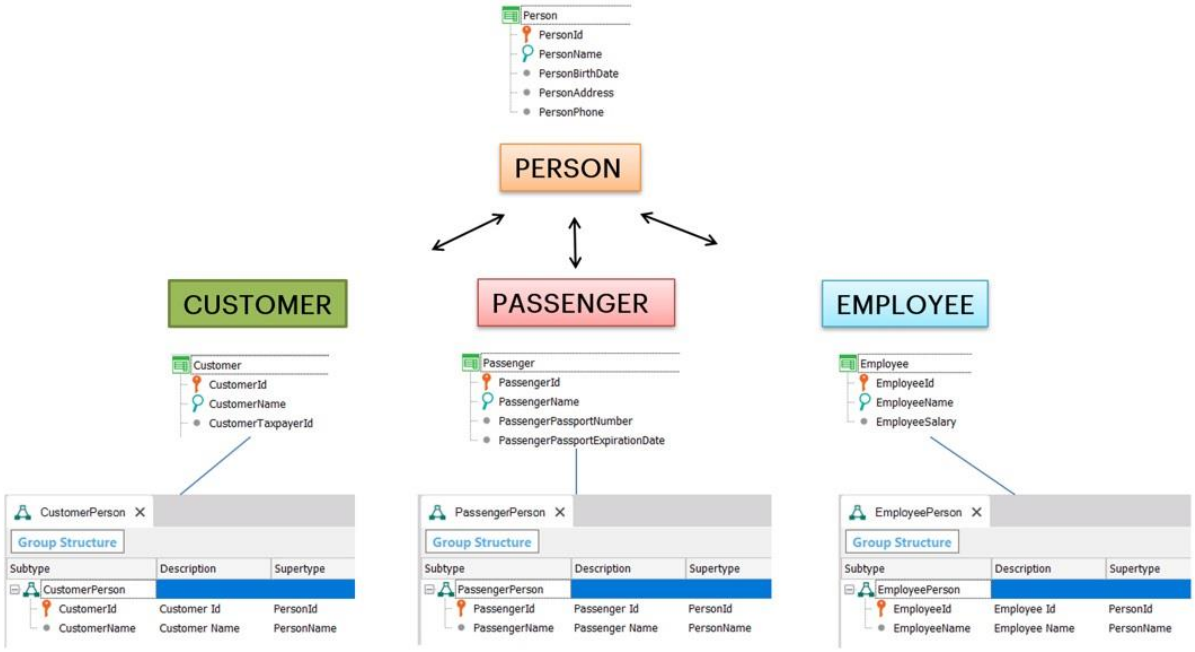


Da mesma maneira, se se executa a transação Passenger, desejamos que quando o usuário digite em PassengerId o valor 8, em PassengerName, se infira Ann Roberts, e o usuário possa atribuir o número de passaporte e a data de validade do mesmo nos atributos próprios (PassengerPassportNumber e PassengerPassportExpirationDate).

Especialização



Analogamente com os funcionários.



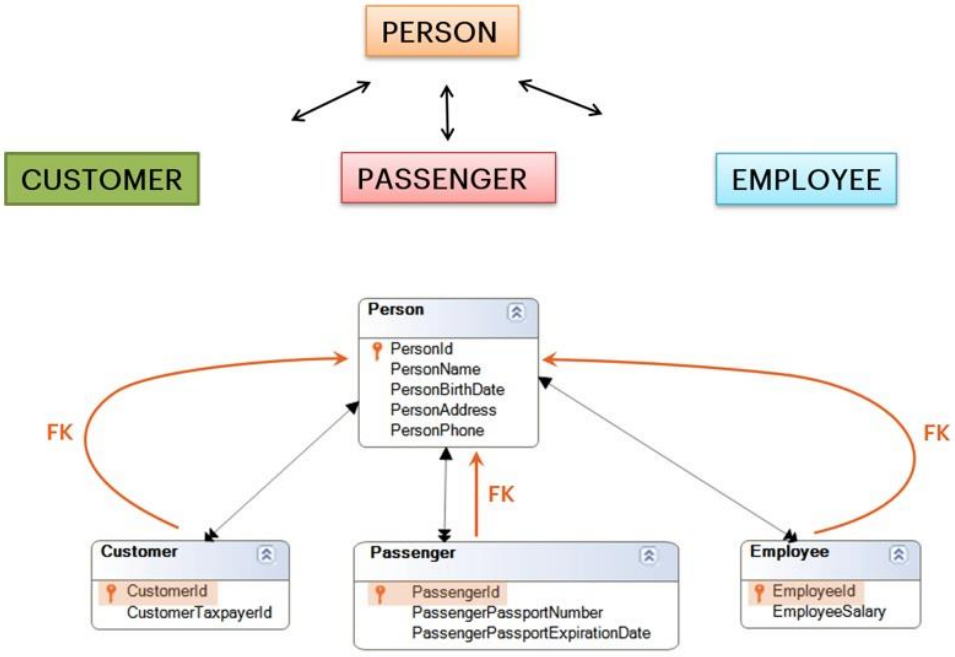
Se simplesmente definimos como chaves primárias de Customer, Passenger e Employee, os atributos CustomerId, PassengerId e EmployeeId, respectivamente, sem relacioná-los de nenhum modo com PersonId (o mesmo que CustomerName, PassengerName e EmployeeName, sem relacioná-los com PersonName), não conseguiremos o que buscamos. Para GeneXus, serão transações completamente independentes.

Para relacioná-las vamos definir grupos de subtipos, para representar que tanto os clientes, como os passageiros e os funcionários, devem ser pessoas válidas (ou seja, previamente registradas).

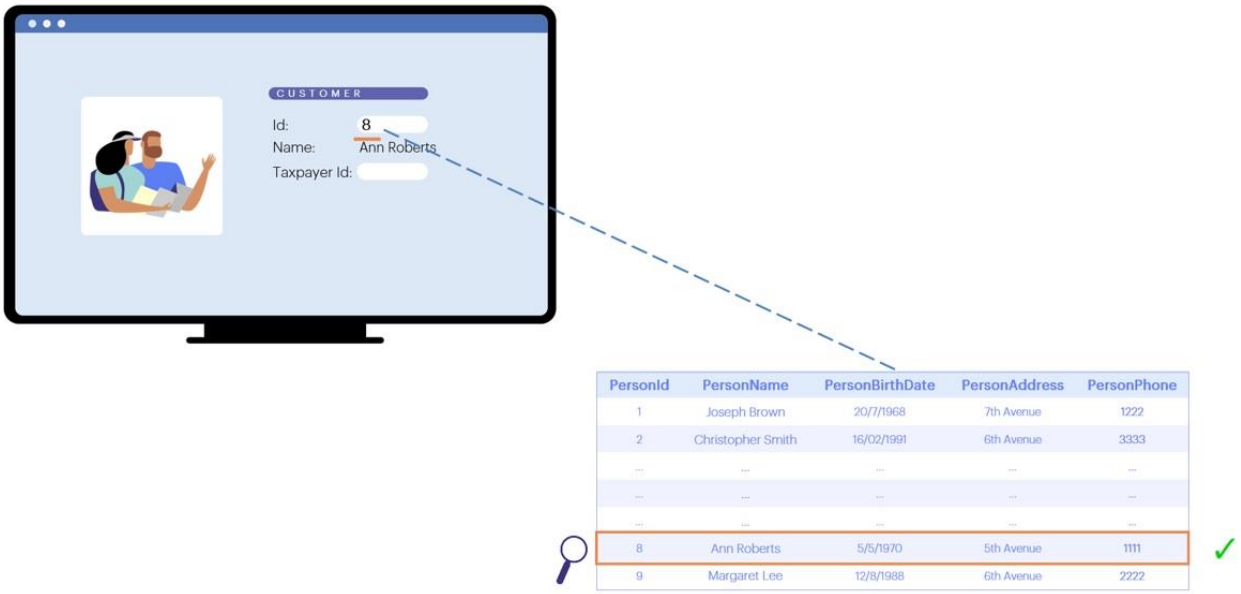
Criamos um grupo CustomerPerson, onde definimos CustomerId e CustomerName como subtipos de PersonId e PersonName, respectivamente.

Outro de nome PassengerPerson, onde definimos PassengerId e PassengerName como subtipos de PersonId e PersonName, respectivamente.

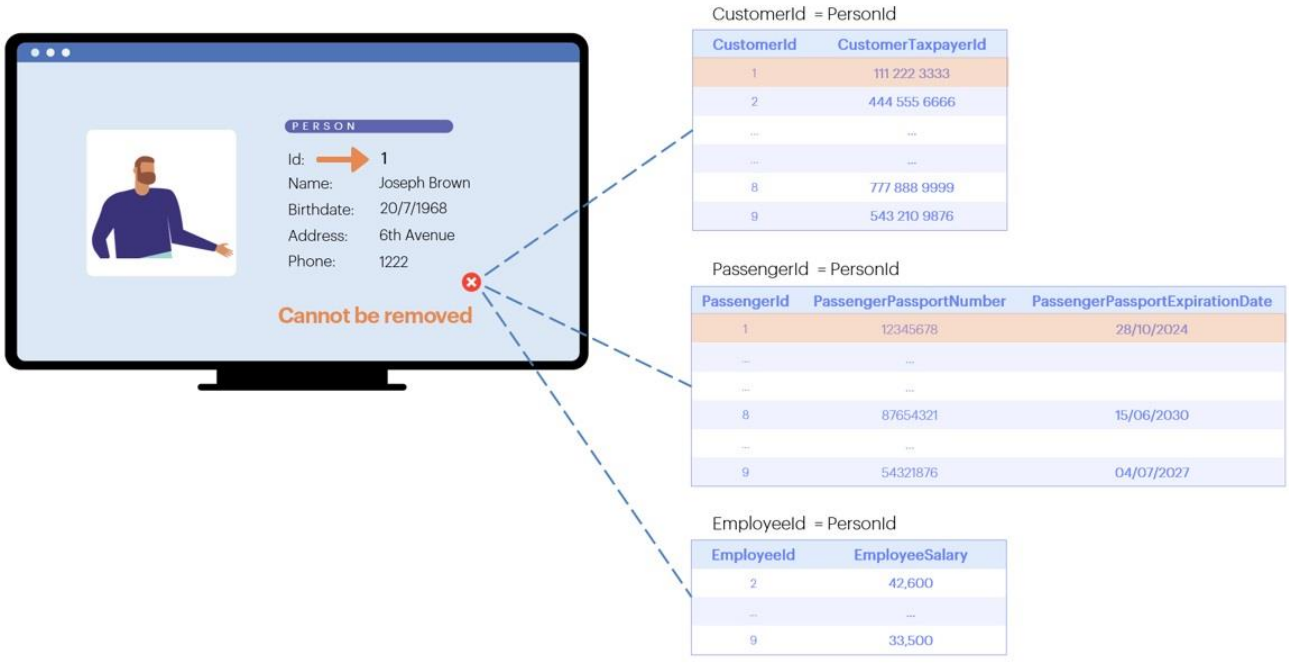
E, por último, um grupo EmployeePerson, onde definimos EmployeeId e EmployeeName como subtipos de PersonId e PersonName, respectivamente.



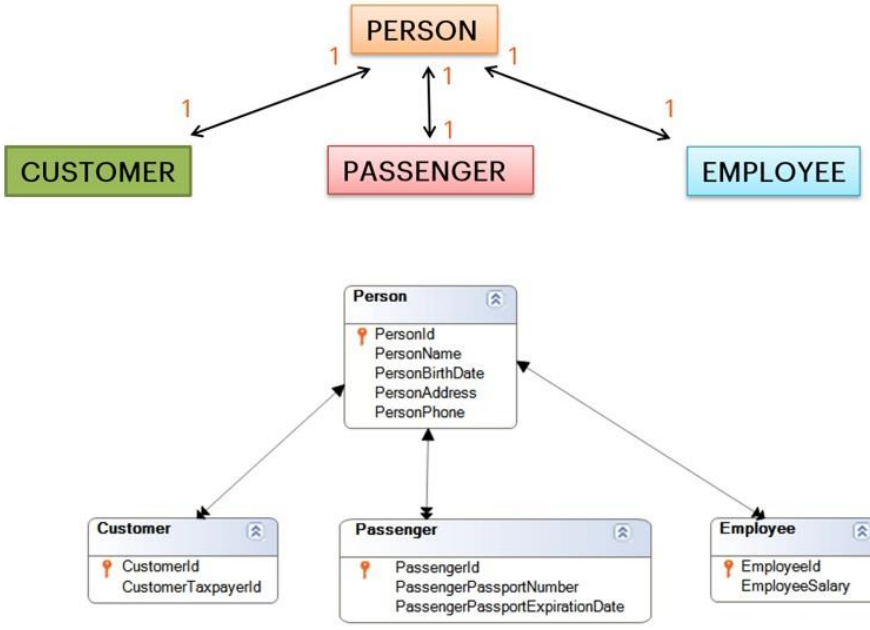
Ao fazer isto, os atributos CustomerId, PassengerId e EmployeeId, além de serem os identificadores das tabelas Customer, Passenger e Employee, respectivamente, e portanto, suas chaves primárias, serão, ao mesmo tempo, chaves estrangeiras para a tabela Person, então, GeneXus controlará a consistência dos dados.



Isto significa que quando o usuário digitar um valor no Id de qualquer uma das três transações (Customer, Passenger ou Employee), irá buscar na tabela Person um registro que tenha como id esse mesmo valor.



Da mesma forma, se quiser excluir uma pessoa através da transação Person, será controlado que não exista um registro em Customer onde CustomerId seja igual ao PersonId que se está querendo excluir, nem um registro em Passenger onde PassengerId seja igual ao PersonId a excluir, nem em Employee onde EmployeeId seja igual ao PersonId a excluir. Se existir algum desses três registos, não será permitida a exclusão da pessoa.



Este desenho representa relações 1 para 1 entre a tabela geral e a correspondente a cada especialização, isto é:

uma pessoa pode estar registrada uma única vez como cliente, porque CustomerId é um PersonId válido e, além disso é chave primária. Da mesma forma, uma pessoa pode estar registada uma única vez como passageiro e uma única vez como funcionário.

Uma pessoa pode ter os 3 papéis, ou estar registrado somente como pessoa e não ter dados extra como cliente da agência, nem como funcionário nem como passageiro.

Nas transações CUSTOMER, PASSENGER e EMPLOYEE, CustomerName, PassengerName e EmployeeName serão atributos inferidos, por isso não estarão fisicamente nas tabelas CUSTOMER, PASSENGER e EMPLOYEE, respectivamente.

Lembre-se que os diagramas que GeneXus faz, não mostram as relações 1 para 1, com as setas apenas indica as relações de chaves estrangeiras. É por isso que vemos a seta dupla do lado das tabelas especializadas.

Existem outros casos de uso de subtipos que não estudaremos neste curso. Se está interessado, pode consultar as informações relacionadas a este tema no curso do nível seguinte.

GeneXus™

training.genexus.com
wiki.genexus.com