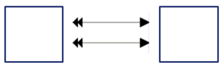
Subtypes

An encompassing look

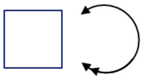
Subtypes allow us to indicate GeneXus how to associate various attribute names to a single concept.

Cases of Subtypes

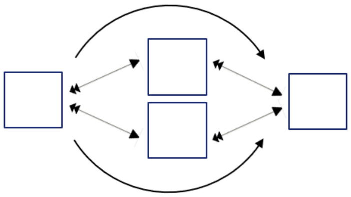
Direct references



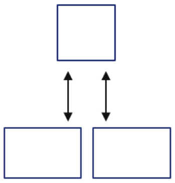
Recursive Subtypes



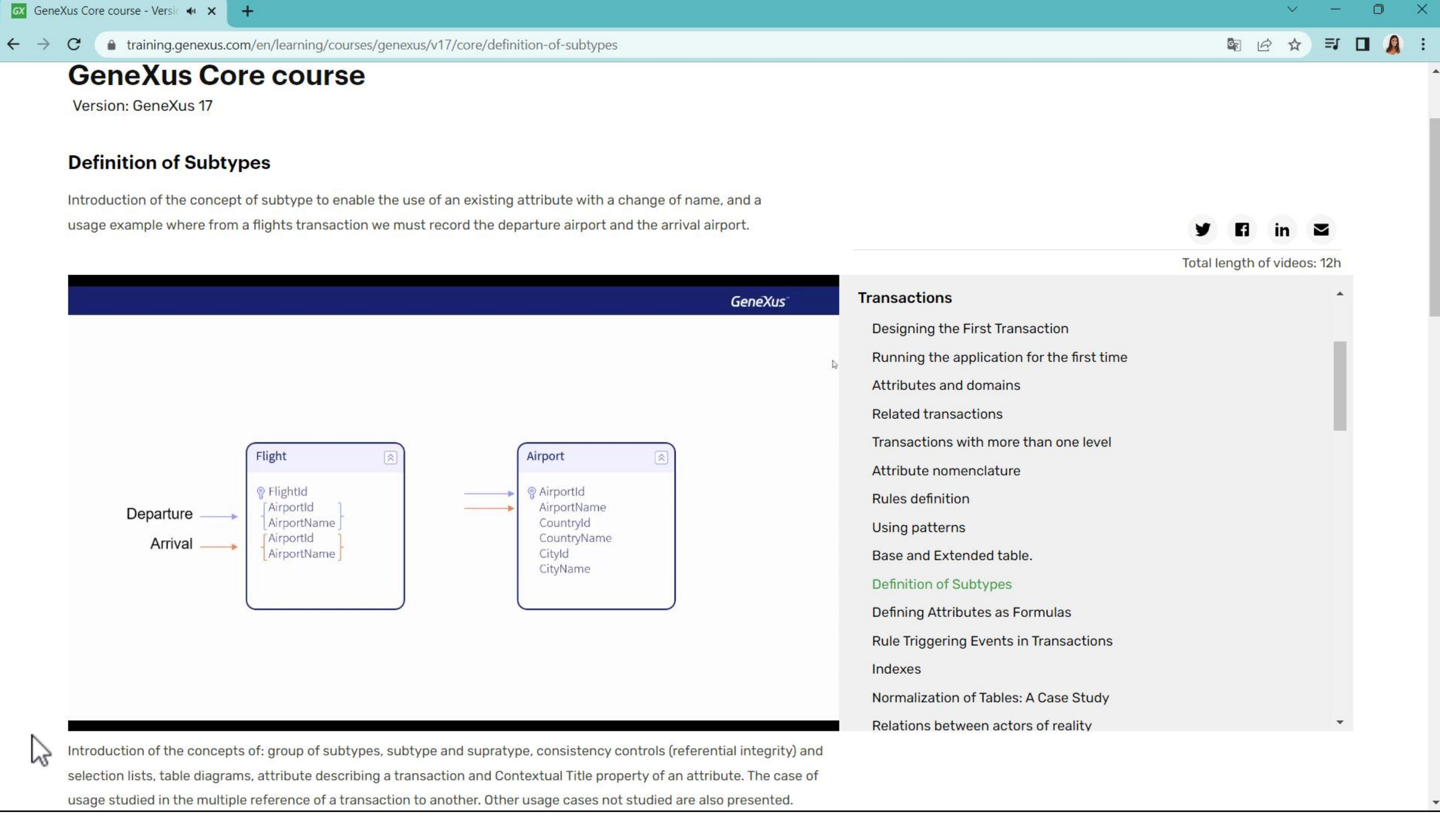
Indirect references



Specialization



In previous courses, we have analyzed and studied different cases where we defined groups of subtypes to resolve conflicts or ambiguities that arise in our applications.



GeneXus Core course

Version: GeneXus 17

Definition of Subtypes

Introduction of the concept of subtype to enable the use of an existing attribute with a change of name, and a usage example where from a flights transaction we must record the departure airport and the arrival airport.

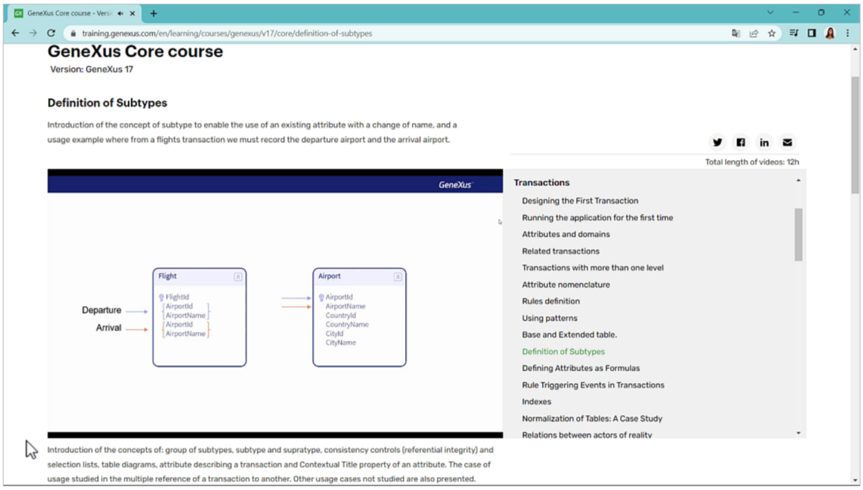
Total length of videos: 12h

Transactions

- Designing the First Transaction
- Running the application for the first time
- Attributes and domains
- Related transactions
- Transactions with more than one level
- Attribute nomenclature
- Rules definition
- Using patterns
- Base and Extended table.
- Definition of Subtypes
- Defining Attributes as Formulas
- Rule Triggering Events in Transactions
- Indexes
- Normalization of Tables: A Case Study
- Relations between actors of reality

Introduction of the concepts of: group of subtypes, subtype and supratype, consistency controls (referential integrity) and selection lists, table diagrams, attribute describing a transaction and Contextual Title property of an attribute. The case of usage studied in the multiple reference of a transaction to another. Other usage cases not studied are also presented.

We started with the simplest example, where we had a double reference to the same concept, but with different roles.


GeneXus Core course - Versi
training.genexus.com/en/learning/courses/genexus/v17/core/definition-of-subtypes

GeneXus Core course

Version: GeneXus 17

Definition of Subtypes

Introduction of the concept of subtype to enable the use of an existing attribute with a change of name, and a usage example where from a flights transaction we must record the departure airport and the arrival airport.

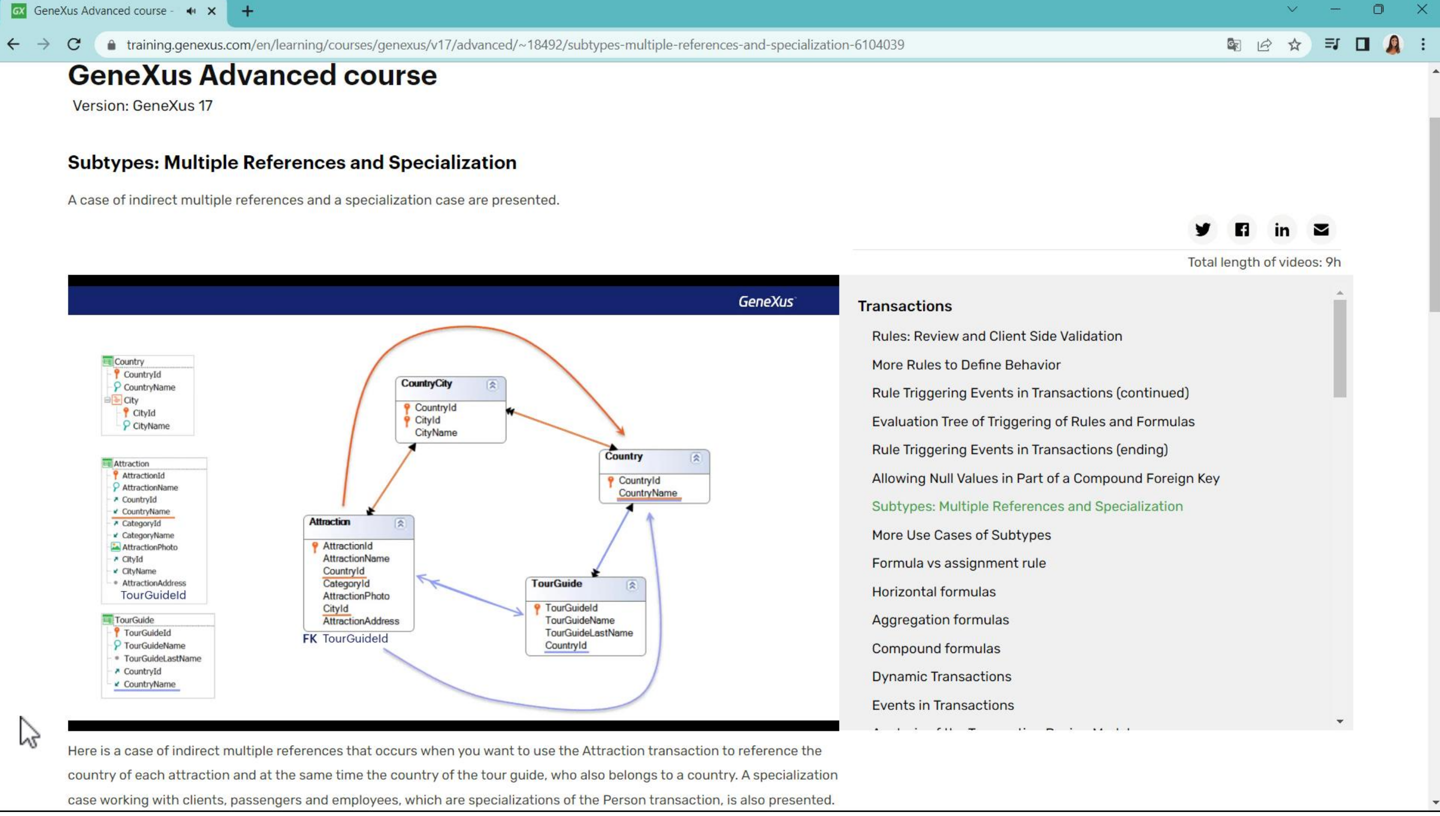
Total length of videos: 12h

Transactions

- Designing the First Transaction
- Running the application for the first time
- Attributes and domains
- Related transactions
- Transactions with more than one level
- Attribute nomenclature
- Rules definition
- Using patterns
- Base and Extended table.
- Definition of Subtypes
- Defining Attributes as Formulas
- Rule Triggering Events in Transactions
- Indexes
- Normalization of Tables: A Case Study
- Relations between actors of reality

Introduction of the concepts of: group of subtypes, subtype and supratype, consistency controls (referential integrity) and selection lists, table diagrams, attribute describing a transaction and Contextual Title property of an attribute. The case of usage studied in the multiple reference of a transaction to another. Other usage cases not studied are also presented.

In this case, we were asked to be able to register both the departure airport and the arrival airport of a certain flight:



GeneXus Advanced course

Version: GeneXus 17

Subtypes: Multiple References and Specialization

A case of indirect multiple references and a specialization case are presented.

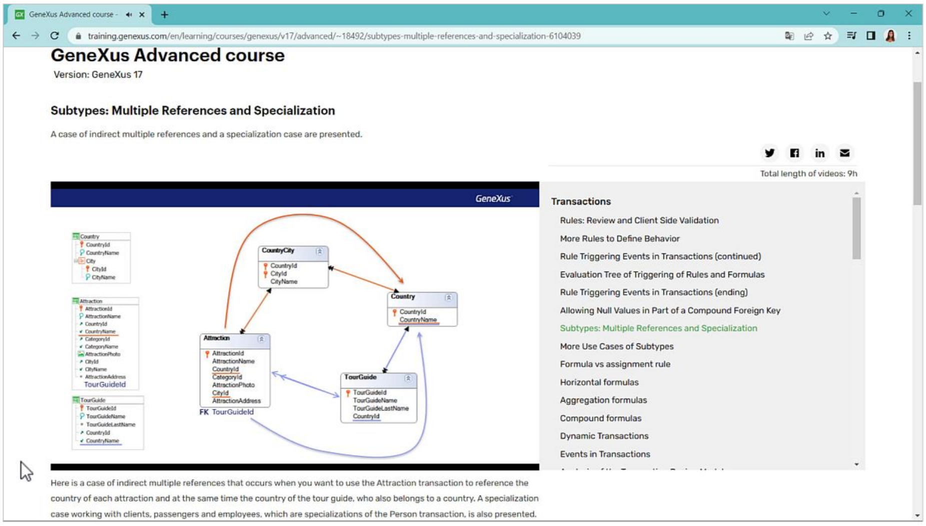
Total length of videos: 9h

Transactions

Rules: Review and Client Side Validation
More Rules to Define Behavior
Rule Triggering Events in Transactions (continued)
Evaluation Tree of Triggering of Rules and Formulas
Rule Triggering Events in Transactions (ending)
Allowing Null Values in Part of a Compound Foreign Key
Subtypes: Multiple References and Specialization
More Use Cases of Subtypes
Formula vs assignment rule
Horizontal formulas
Aggregation formulas
Compound formulas
Dynamic Transactions
Events in Transactions

Here is a case of indirect multiple references that occurs when you want to use the Attraction transaction to reference the country of each attraction and at the same time the country of the tour guide, who also belongs to a country. A specialization case working with clients, passengers and employees, which are specializations of the Person transaction, is also presented.

Later on, we studied a case of multiple indirect references, since, given a table, we had two paths to get to another one:



GeneXus Advanced course

Version: GeneXus 17

Subtypes: Multiple References and Specialization

A case of indirect multiple references and a specialization case are presented.

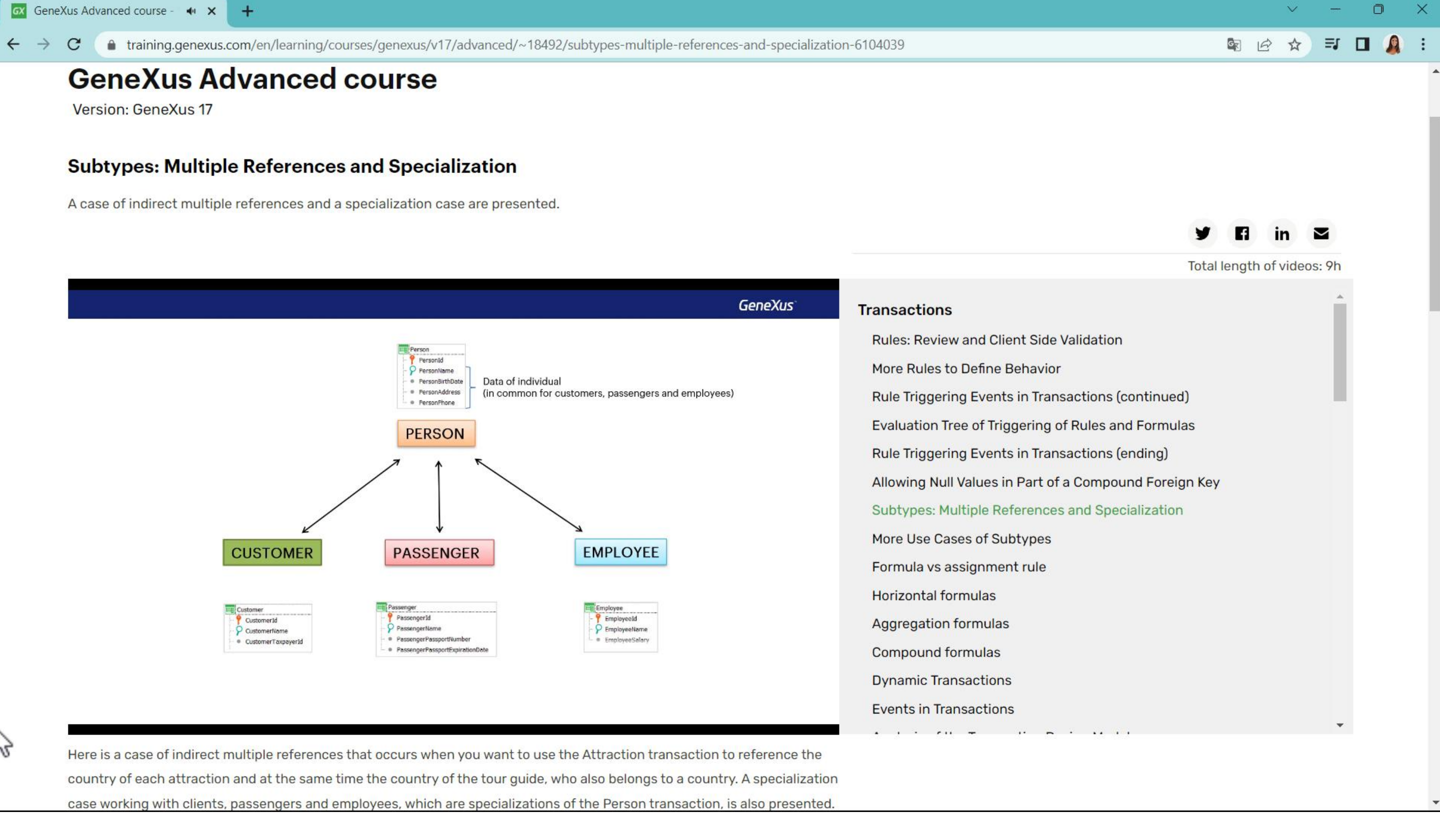
Total length of videos: 9h

Transactions

- Rules: Review and Client Side Validation
- More Rules to Define Behavior
- Rule Triggering Events in Transactions (continued)
- Evaluation Tree of Triggering of Rules and Formulas
- Rule Triggering Events in Transactions (ending)
- Allowing Null Values in Part of a Compound Foreign Key
- Subtypes: Multiple References and Specialization
- More Use Cases of Subtypes
- Formula vs assignment rule
- Horizontal formulas
- Aggregation formulas
- Compound formulas
- Dynamic Transactions
- Events in Transactions

Here is a case of indirect multiple references that occurs when you want to use the Attraction transaction to reference the country of each attraction and at the same time the country of the tour guide, who also belongs to a country. A specialization case working with clients, passengers and employees, which are specializations of the Person transaction, is also presented.

It was the case where we had to record information about tour guides, and there were two paths to get to the CountryId attribute to identify their country.



GeneXus Advanced course

Version: GeneXus 17

Subtypes: Multiple References and Specialization

A case of indirect multiple references and a specialization case are presented.

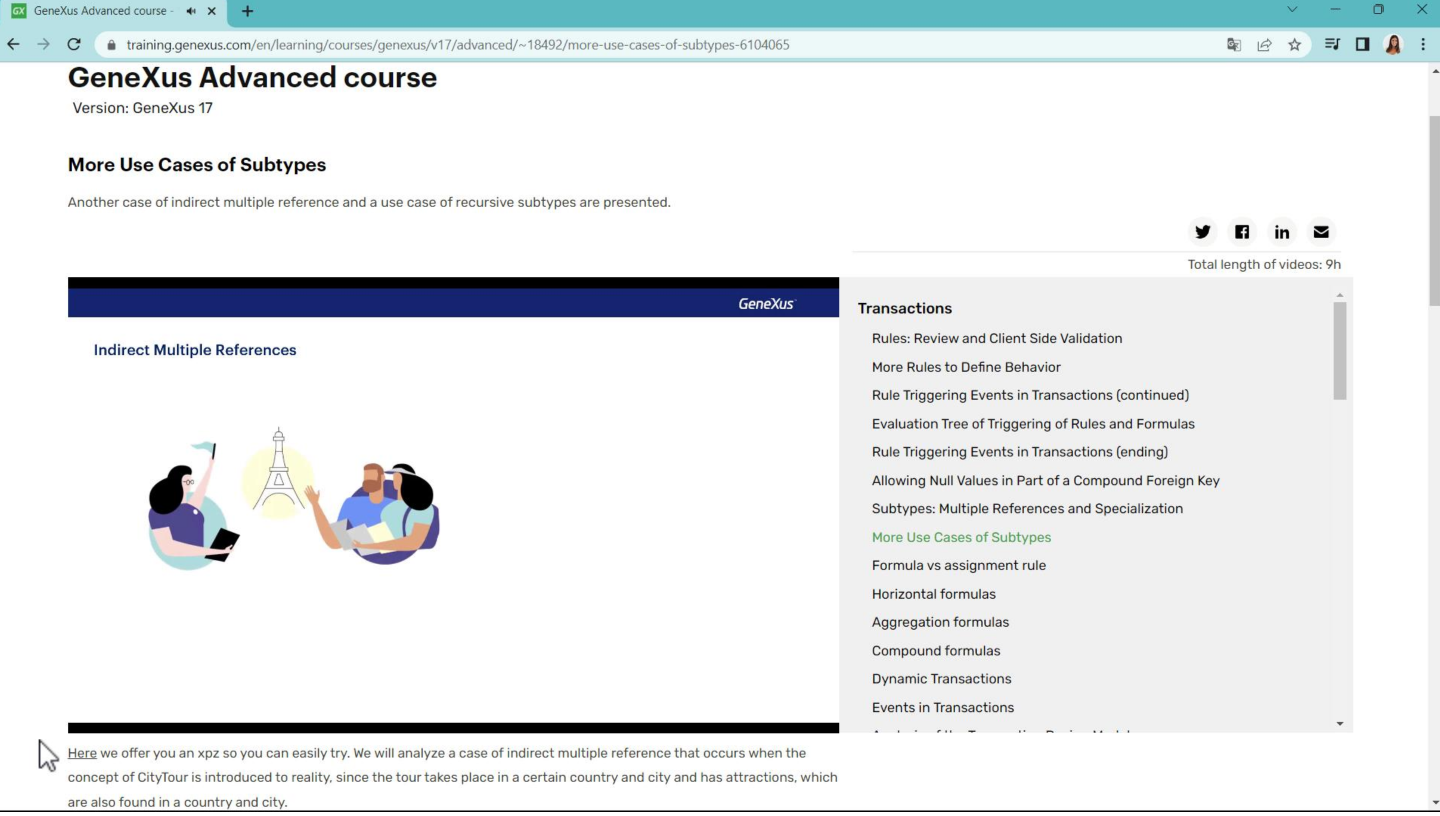
Total length of videos: 9h

Transactions

- Rules: Review and Client Side Validation
- More Rules to Define Behavior
- Rule Triggering Events in Transactions (continued)
- Evaluation Tree of Triggering of Rules and Formulas
- Rule Triggering Events in Transactions (ending)
- Allowing Null Values in Part of a Compound Foreign Key
- Subtypes: Multiple References and Specialization
- More Use Cases of Subtypes
- Formula vs assignment rule
- Horizontal formulas
- Aggregation formulas
- Compound formulas
- Dynamic Transactions
- Events in Transactions

Here is a case of indirect multiple references that occurs when you want to use the Attraction transaction to reference the country of each attraction and at the same time the country of the tour guide, who also belongs to a country. A specialization case working with clients, passengers and employees, which are specializations of the Person transaction, is also presented.

In the same video, we saw a case of using subtypes to represent a specialization, where one transaction recorded the data common to people, and other transactions (specializations of the first one) recorded their particular information.



GeneXus Advanced course

training.genexus.com/en/learning/courses/genexus/v17/advanced/~18492/more-use-cases-of-subtypes-6104065

GeneXus Advanced course

Version: GeneXus 17

More Use Cases of Subtypes

Another case of indirect multiple reference and a use case of recursive subtypes are presented.

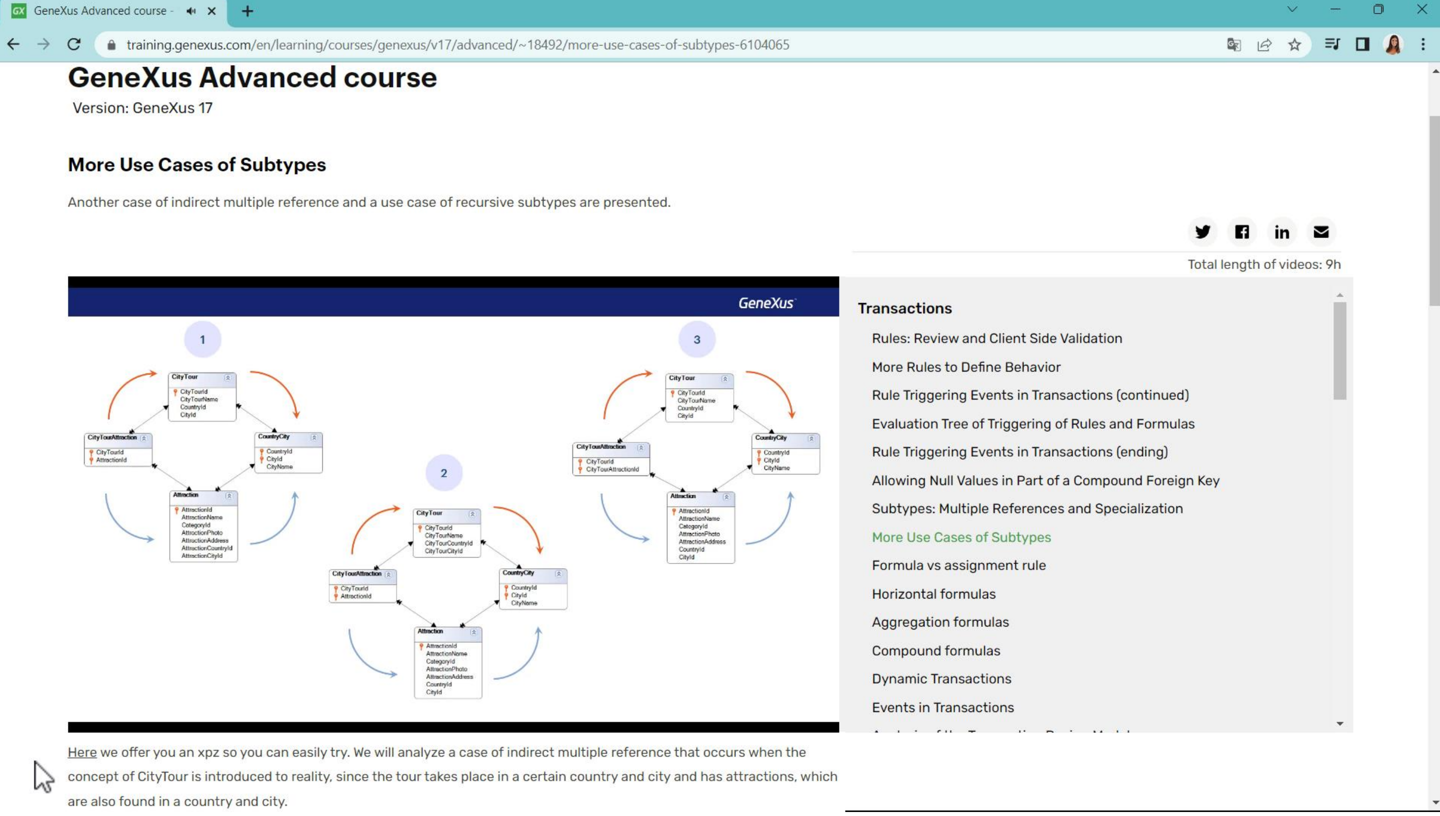
Total length of videos: 9h

Transactions

- Rules: Review and Client Side Validation
- More Rules to Define Behavior
- Rule Triggering Events in Transactions (continued)
- Evaluation Tree of Triggering of Rules and Formulas
- Rule Triggering Events in Transactions (ending)
- Allowing Null Values in Part of a Compound Foreign Key
- Subtypes: Multiple References and Specialization
- More Use Cases of Subtypes
- Formula vs assignment rule
- Horizontal formulas
- Aggregation formulas
- Compound formulas
- Dynamic Transactions
- Events in Transactions

Here we offer you an xpz so you can easily try. We will analyze a case of indirect multiple reference that occurs when the concept of CityTour is introduced to reality, since the tour takes place in a certain country and city and has attractions, which are also found in a country and city.

In this other video, we look even more closely at a case of indirect multiple references, when we had to record the tours offered to the travel agency's clients to visit the tourist attractions of a given city.



GeneXus Advanced course

Version: GeneXus 17

More Use Cases of Subtypes

Another case of indirect multiple reference and a use case of recursive subtypes are presented.

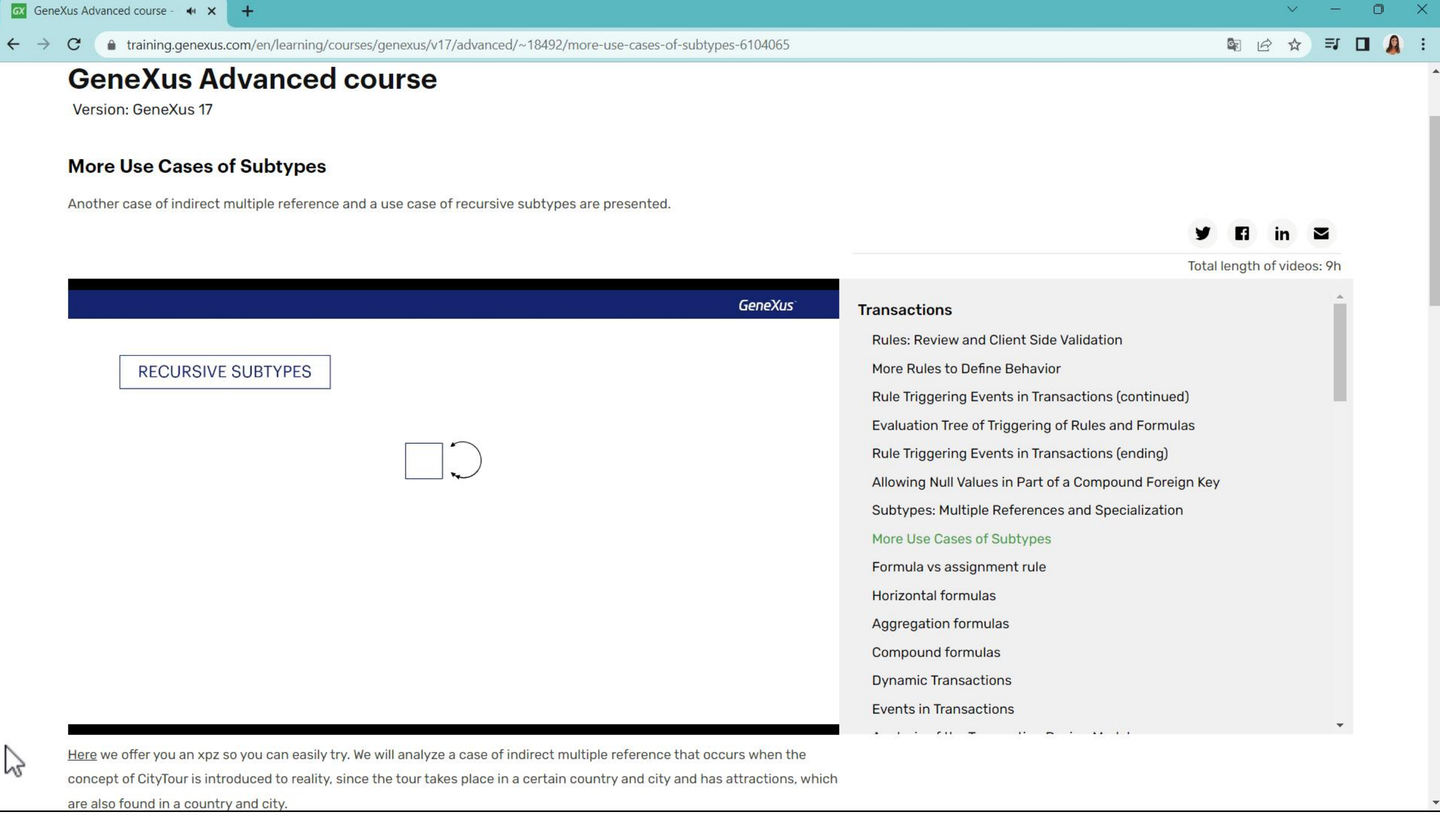
Total length of videos: 9h

Transactions

- Rules: Review and Client Side Validation
- More Rules to Define Behavior
- Rule Triggering Events in Transactions (continued)
- Evaluation Tree of Triggering of Rules and Formulas
- Rule Triggering Events in Transactions (ending)
- Allowing Null Values in Part of a Compound Foreign Key
- Subtypes: Multiple References and Specialization
- More Use Cases of Subtypes
- Formula vs assignment rule
- Horizontal formulas
- Aggregation formulas
- Compound formulas
- Dynamic Transactions
- Events in Transactions

Here we offer you an xpz so you can easily try. We will analyze a case of indirect multiple reference that occurs when the concept of CityTour is introduced to reality, since the tour takes place in a certain country and city and has attractions, which are also found in a country and city.

In this example, we thoroughly analyze the problem with its different solutions, each with its advantages and disadvantages. Also, we highlight the importance of studying each particular case and determining when and where to use subtypes as necessary, and not arbitrarily.


GeneXus Advanced course -
training.genexus.com/en/learning/courses/genexus/v17/advanced/~18492/more-use-cases-of-subtypes-6104065
GeneXus

GeneXus Advanced course

Version: GeneXus 17

More Use Cases of Subtypes

Another case of indirect multiple reference and a use case of recursive subtypes are presented.

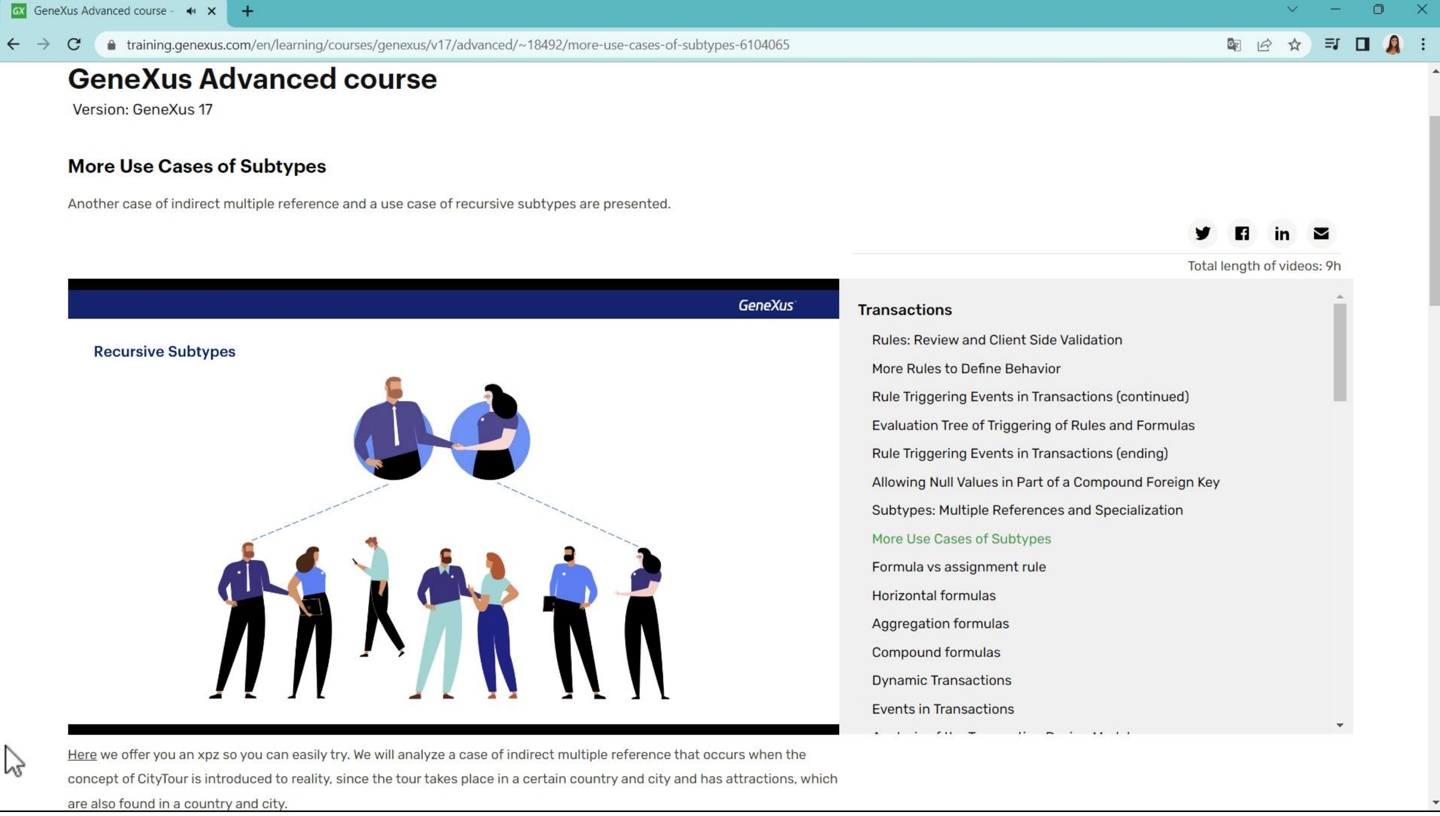
Total length of videos: 9h

RECURSIVE SUBTYPES

Transactions
- Rules: Review and Client Side Validation
- More Rules to Define Behavior
- Rule Triggering Events in Transactions (continued)
- Evaluation Tree of Triggering of Rules and Formulas
- Rule Triggering Events in Transactions (ending)
- Allowing Null Values in Part of a Compound Foreign Key
- Subtypes: Multiple References and Specialization
- More Use Cases of Subtypes
- Formula vs assignment rule
- Horizontal formulas
- Aggregation formulas
- Compound formulas
- Dynamic Transactions
- Events in Transactions

Here we offer you an xpz so you can easily try. We will analyze a case of indirect multiple reference that occurs when the concept of CityTour is introduced to reality, since the tour takes place in a certain country and city and has attractions, which are also found in a country and city.

We also presented a use case of recursive subtypes, where an entity had to be self-referenced.


GeneXus Advanced course -
training.genexus.com/en/learning/courses/genexus/v17/advanced/~18492/more-use-cases-of-subtypes-6104065
GeneXus
Recursive Subtypes

GeneXus Advanced course

Version: GeneXus 17

More Use Cases of Subtypes

Another case of indirect multiple reference and a use case of recursive subtypes are presented.

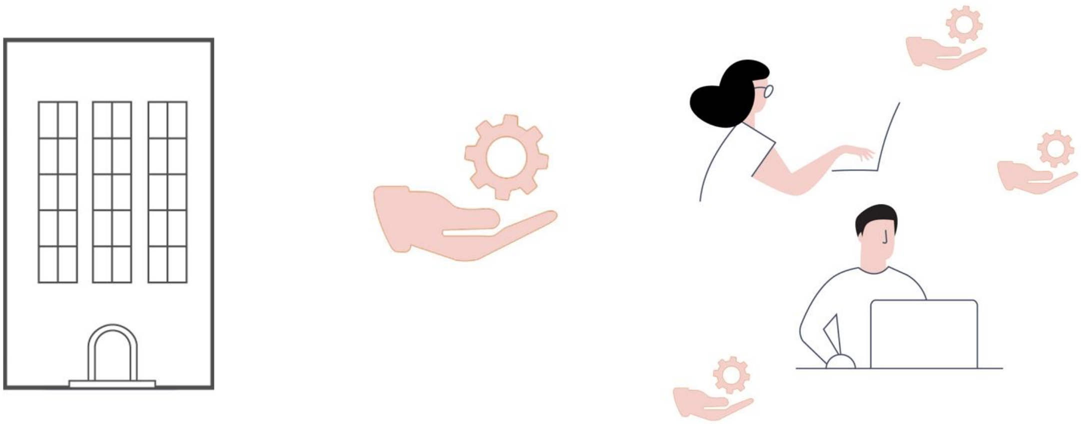
Total length of videos: 9h

Transactions
- Rules: Review and Client Side Validation
- More Rules to Define Behavior
- Rule Triggering Events in Transactions (continued)
- Evaluation Tree of Triggering of Rules and Formulas
- Rule Triggering Events in Transactions (ending)
- Allowing Null Values in Part of a Compound Foreign Key
- Subtypes: Multiple References and Specialization
- More Use Cases of Subtypes
- Formula vs assignment rule
- Horizontal formulas
- Aggregation formulas
- Compound formulas
- Dynamic Transactions
- Events in Transactions

Here we offer you an xpz so you can easily try. We will analyze a case of indirect multiple reference that occurs when the concept of CityTour is introduced to reality, since the tour takes place in a certain country and city and has attractions, which are also found in a country and city.

This was the reality where we represented the information of the employees of the travel agency, where each employee could be, in turn, the manager of one or more other employees.

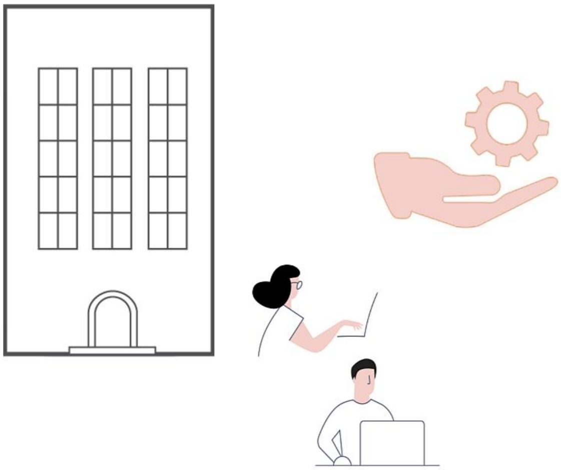
New case: Avoid referential relationship



Now let's study one last example where we must avoid the referential relationship.

Suppose we must model transactions for a reality in which we have companies and services that they can purchase (such as, for example, an emergency healthcare service).
In turn, companies have employees who may also have purchased services that don't necessarily match those of the company they work for. We are interested in recording these employee services because, for example, if many employees have purchased a certain emergency healthcare service, an agreement can be sought with that service to obtain a benefit.

New case: Avoid referential relationship

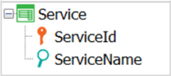


Employees can only work in one company, but they are not to be represented as a strong entity, but as dependent on the company.

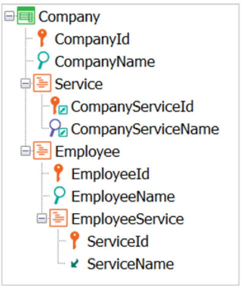
In our reality, employees can only work in a company, but we don't want to represent them as a strong entity, but as dependent on the company. Let's look at these two proposed solutions, where one is correct and the other is not.

Two possible solutions

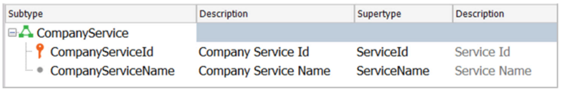
Solution A)



```
Service
  ServiceId
  ServiceName
```



```
Company
  CompanyId
  CompanyName
  Service
    CompanyServiceId
    CompanyServiceName
  Employee
    EmployeeId
    EmployeeName
    EmployeeService
      ServiceId
      ServiceName
```

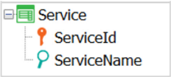


Subtype	Description	Supertype	Description
CompanyService			
CompanyServiceId	Company Service Id	ServiceId	Service Id
CompanyServiceName	Company Service Name	ServiceName	Service Name

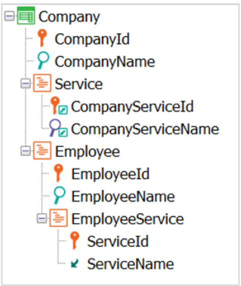
A) The first solution would be to create these two transactions and the following group of subtypes:

Two possible solutions

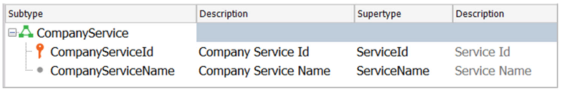
Solution A)



- Service
  - ServiceId
  - ServiceName

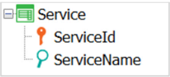


- Company
  - CompanyId
  - CompanyName
  - Service
    - CompanyServiceId
    - CompanyServiceName
  - Employee
    - EmployeeId
    - EmployeeName
    - EmployeeService
      - ServiceId
      - ServiceName

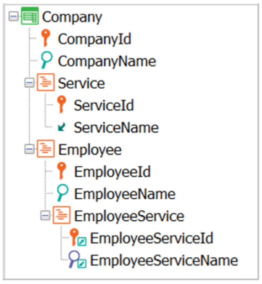


Subtype	Description	Supertype	Description
CompanyService			
CompanyServiceId	Company Service Id	ServiceId	Service Id
CompanyServiceName	Company Service Name	ServiceName	Service Name

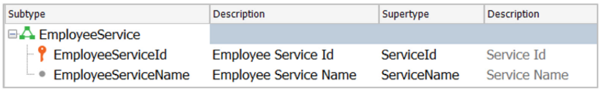
Solution B)



- Service
  - ServiceId
  - ServiceName



- Company
  - CompanyId
  - CompanyName
  - Service
    - ServiceId
    - ServiceName
  - Employee
    - EmployeeId
    - EmployeeName
    - EmployeeService
      - EmployeeServiceId
      - EmployeeServiceName

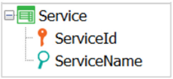


Subtype	Description	Supertype	Description
EmployeeService			
EmployeeServiceId	Employee Service Id	ServiceId	Service Id
EmployeeServiceName	Employee Service Name	ServiceName	Service Name

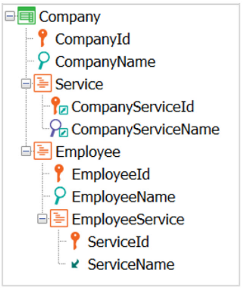
B) The second solution would be to create these two transactions and the following group of subtypes:

Two possible solutions

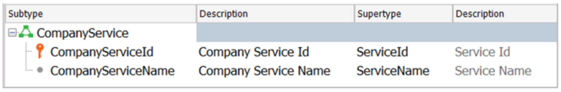
Solution A) ✓



- Service
  - ServiceId
  - ServiceName

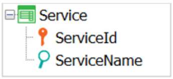


- Company
  - CompanyId
  - CompanyName
  - Service
    - CompanyServiceId
    - CompanyServiceName
  - Employee
    - EmployeeId
    - EmployeeName
    - EmployeeService
      - ServiceId
      - ServiceName

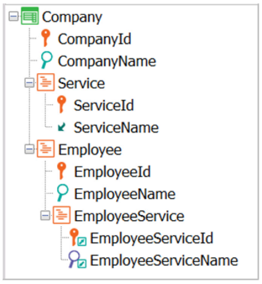


Subtype	Description	Supertype	Description
CompanyService			
CompanyServiceId	Company Service Id	ServiceId	Service Id
CompanyServiceName	Company Service Name	ServiceName	Service Name

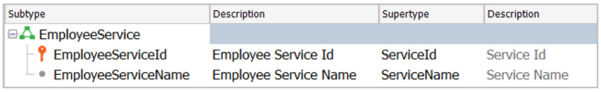
Solution B) ✗



- Service
  - ServiceId
  - ServiceName



- Company
  - CompanyId
  - CompanyName
  - Service
    - ServiceId
    - ServiceName
  - Employee
    - EmployeeId
    - EmployeeName
    - EmployeeService
      - EmployeeServiceId
      - EmployeeServiceName

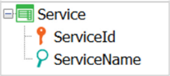


Subtype	Description	Supertype	Description
EmployeeService			
EmployeeServiceId	Employee Service Id	ServiceId	Service Id
EmployeeServiceName	Employee Service Name	ServiceName	Service Name

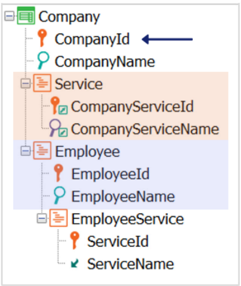
A) is the correct solution and not B).

Two possible solutions

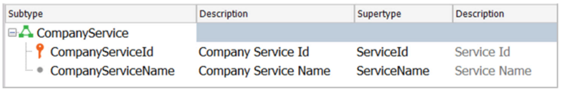
Solution A)



Service
- ServiceId
- ServiceName

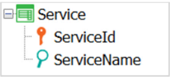


Company
- CompanyId ←
- CompanyName
- Service
  - CompanyServiceId
  - CompanyServiceName
- Employee
  - EmployeeId
  - EmployeeName
  - EmployeeService
    - ServiceId
    - ServiceName

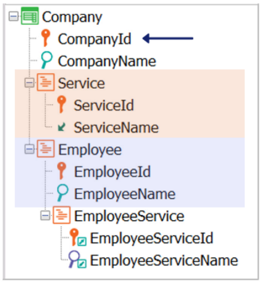


Subtype	Description	Supertype	Description
CompanyService			
CompanyServiceId	Company Service Id	ServiceId	Service Id
CompanyServiceName	Company Service Name	ServiceName	Service Name

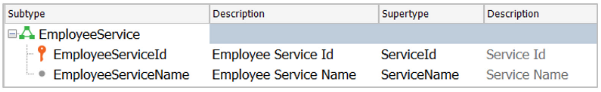
Solution B)



Service
- ServiceId
- ServiceName

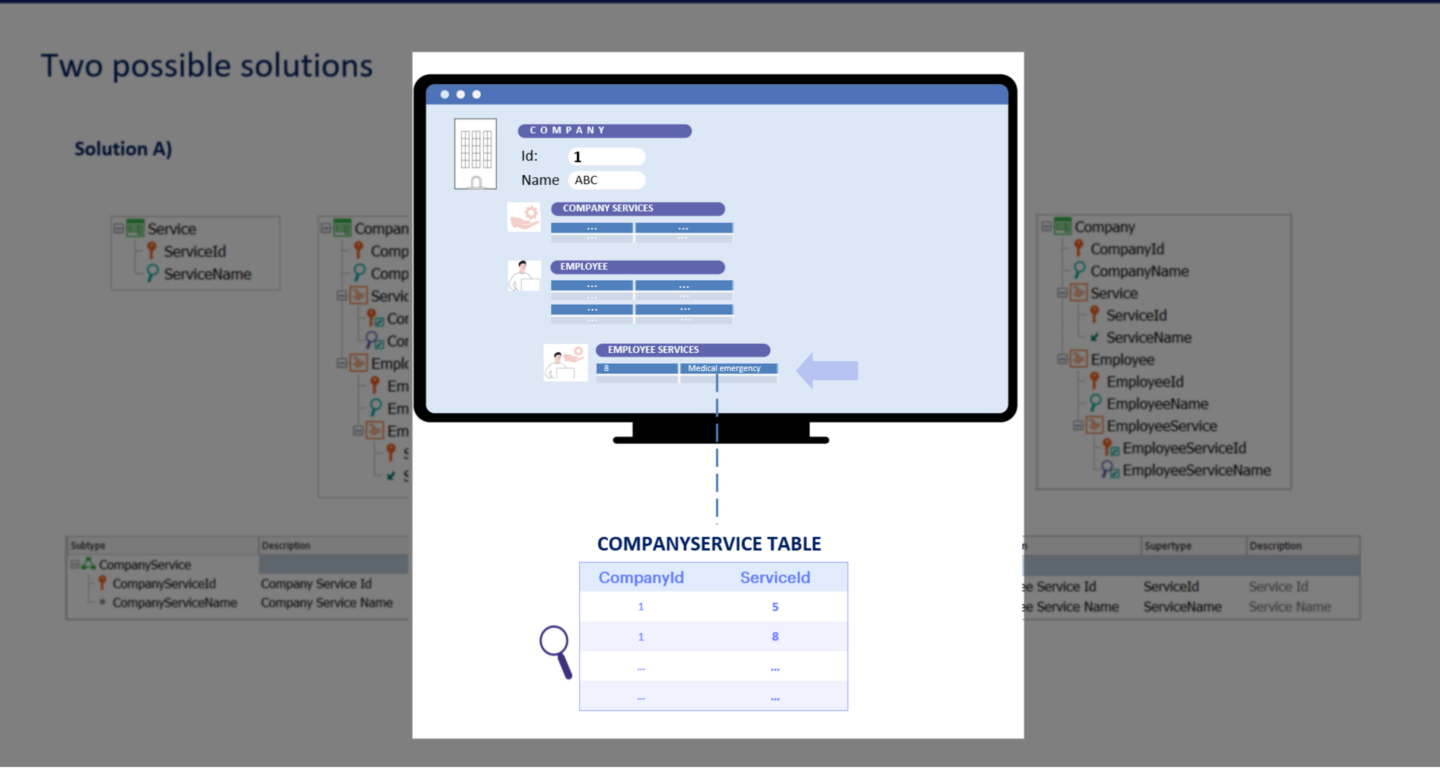


Company
- CompanyId ←
- CompanyName
- Service
  - ServiceId
  - ServiceName
- Employee
  - EmployeeId
  - EmployeeName
  - EmployeeService
    - EmployeeServiceId
    - EmployeeServiceName



Subtype	Description	Supertype	Description
EmployeeService			
EmployeeServiceId	Employee Service Id	ServiceId	Service Id
EmployeeServiceName	Employee Service Name	ServiceName	Service Name

If we look closely, in the Company transaction there are two parallel levels: Service and Employee. This means that everything that is inferred from any of these levels will correspond to the same company. However, we do not want the employee's service to exist as a company service, since in our reality the employee may have purchased services different from those of the company he or she works for.



Two possible solutions

Solution A)

Subtype	Description
CompanyService	
CompanyServiceId	Company Service Id
CompanyServiceName	Company Service Name

n	Supertype	Description
e Service Id	ServiceId	Service Id
e Service Name	ServiceName	Service Name

COMPANY

Id: 1

Name ABC

COMPANY SERVICES

EMPLOYEE

EMPLOYEE SERVICES

8	Medical emergency

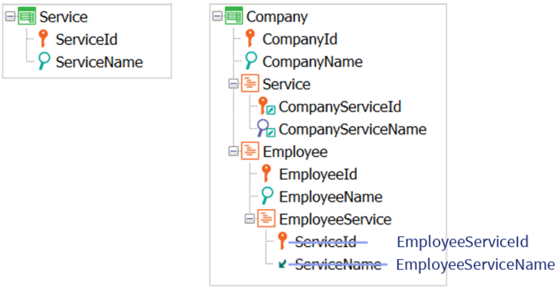
COMPANYSERVICE TABLE

CompanyId	ServiceId
1	5
1	8
...	...
...	...

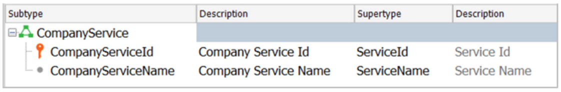
In other words: when the user enters the employee services grid, we don't want to check whether the entered service exists as a record in the table corresponding to Company.Service.

Two possible solutions

Solution A)



- Service
  - ServiceId
  - ServiceName
- Company
  - CompanyId
  - CompanyName
  - Service
    - CompanyServiceId
    - CompanyServiceName
  - Employee
    - EmployeeId
    - EmployeeName
    - EmployeeService
      - ~~ServiceId~~ EmployeeServiceId
      - ~~ServiceName~~ EmployeeServiceName

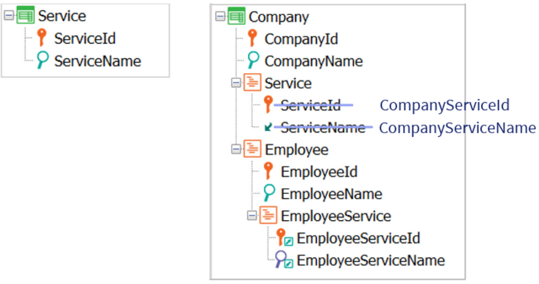


Subtype	Description	Supertype	Description
CompanyService			
CompanyServiceId	Company Service Id	ServiceId	Service Id
CompanyServiceName	Company Service Name	ServiceName	Service Name

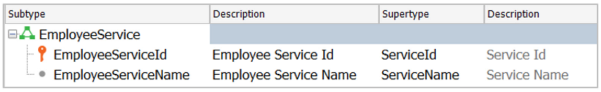
EmployeeService

EmployeeServiceId	...	ServiceId	...
EmployeeServiceName	...	ServiceName	...

Solution B)



- Service
  - ServiceId
  - ServiceName
- Company
  - CompanyId
  - CompanyName
  - Service
    - ~~ServiceId~~ CompanyServiceId
    - ~~ServiceName~~ CompanyServiceName
  - Employee
    - EmployeeId
    - EmployeeName
    - EmployeeService
      - EmployeeServiceId
      - EmployeeServiceName

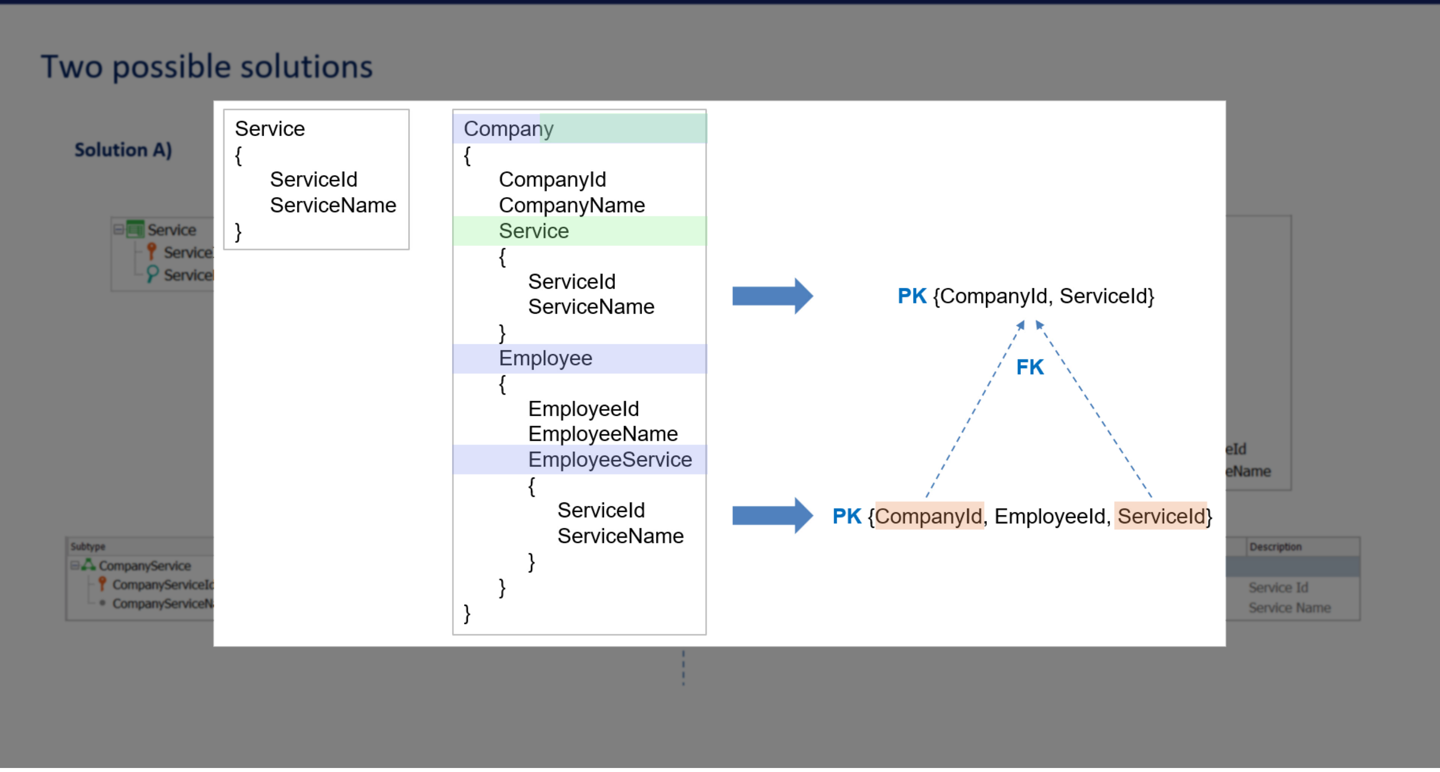


Subtype	Description	Supertype	Description
EmployeeService			
EmployeeServiceId	Employee Service Id	ServiceId	Service Id
EmployeeServiceName	Employee Service Name	ServiceName	Service Name

CompanyService

CompanyServiceId	...	ServiceId	...
CompanyServiceName	...	ServiceName	...

Clearly, we need to define a group of subtypes because in the same transaction GeneXus will not allow us to repeat the same attribute name. So, the question that arises is: does it make any difference whether you define it on one level or the other? The answer is no. We could define two groups of subtypes and solve the problem, but, as we have already seen in previous videos, it is not good practice to define more subtypes than strictly necessary. That’s because it is never exactly the same to have the subtype than having the supertype, as will be made clear with this example.



Two possible solutions

Solution A)

```
Service
{
    ServiceId
    ServiceName
}
```

```
Company
{
    CompanyId
    CompanyName
    Service
    {
        ServiceId
        ServiceName
    }
    Employee
    {
        EmployeeId
        EmployeeName
        EmployeeService
        {
            ServiceId
            ServiceName
        }
    }
}
```

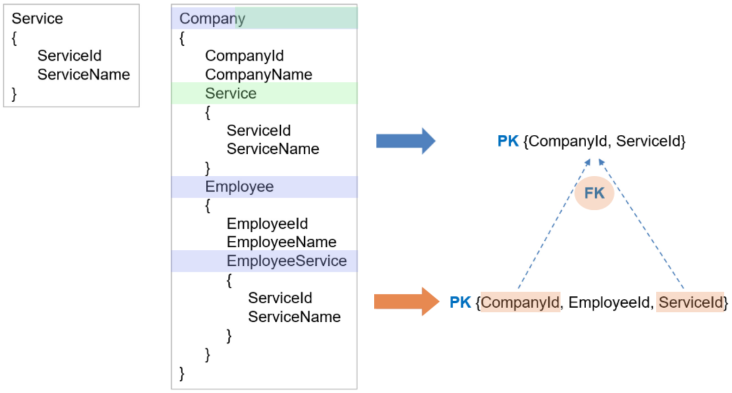
PK {CompanyId, ServiceId}

FK

PK {CompanyId, EmployeeId, ServiceId}

Therefore, to solve the problem it is enough to have only one group. Why, then, is the correct solution A) and not B)? Because if GeneXus allowed us to repeat the same attribute name, it would clearly find that in the table associated with the level Company.Employee.EmployeeService, with primary key {CompanyId, EmployeeId, ServiceId} the attributes {CompanyId, ServiceId} would form a foreign key to the table corresponding to the level Company.Service (because its primary key would be {CompanyId, ServiceId}).

Two possible solutions

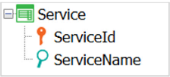


```
Service
{
    ServiceId
    ServiceName
}
```

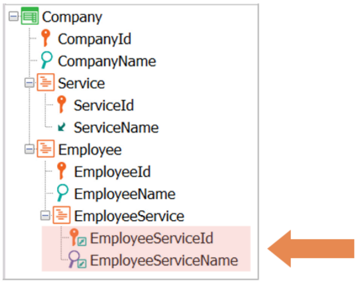
```
Company
{
    CompanyId
    CompanyName
    Service
    {
        ServiceId
        ServiceName
    }
    Employee
    {
        EmployeeId
        EmployeeName
        EmployeeService
        {
            ServiceId
            ServiceName
        }
    }
}
```

PK {CompanyId, ServiceId}
FK
PK {CompanyId, EmployeeId, ServiceId}

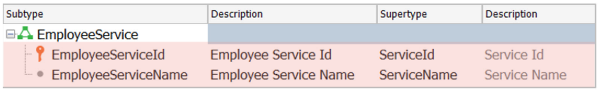
Solution B)



```
Service
    ServiceId
    ServiceName
```



```
Company
    CompanyId
    CompanyName
    Service
        ServiceId
        ServiceName
    Employee
        EmployeeId
        EmployeeName
        EmployeeService
            EmployeeServiceId
            EmployeeServiceName
```



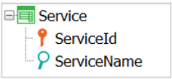
Subtype	Description	Supertype	Description
EmployeeService			
EmployeeServiceId	Employee Service Id	ServiceId	Service Id
EmployeeServiceName	Employee Service Name	ServiceName	Service Name

This does not delete for GeneXus its referential function

But if we change the name (with a subtype) of ServiceId in the table in which this attribute is part of a foreign key, for GeneXus this doesn't delete its referential role.

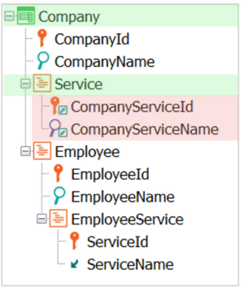
Two possible solutions

Solution A)



```
Service
    ServiceId
    ServiceName
```

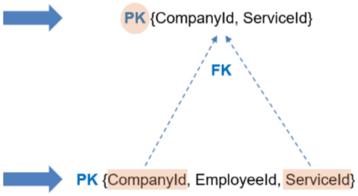
Company.Service

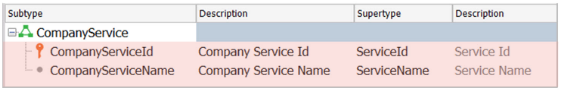


```
Company
    CompanyId
    CompanyName
    Service
        CompanyServiceId
        CompanyServiceName
    Employee
        EmployeeId
        EmployeeName
        EmployeeService
            ServiceId
            ServiceName
```

```
Service
{
    ServiceId
    ServiceName
}
```

```
Company
{
    CompanyId
    CompanyName
    Service
    {
        ServiceId
        ServiceName
    }
    Employee
    {
        EmployeeId
        EmployeeName
        EmployeeService
        {
            ServiceId
            ServiceName
        }
    }
}
```


PK {CompanyId, ServiceId}
FK
PK {CompanyId, EmployeeId, ServiceId}

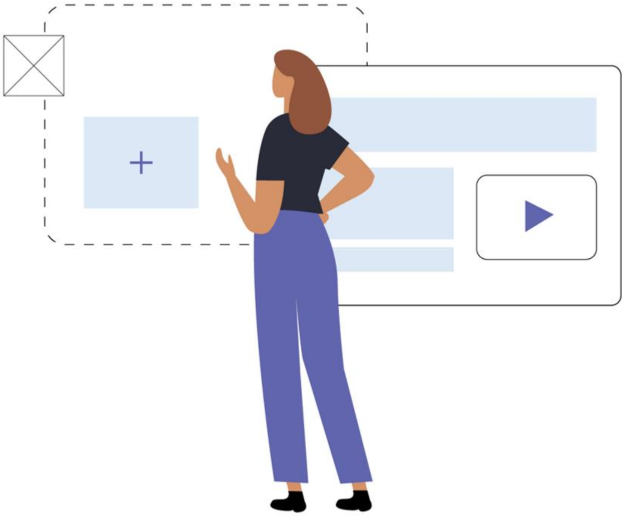


Subtype	Description	Supertype	Description
CompanyService			
CompanyServiceId	Company Service Id	ServiceId	Service Id
CompanyServiceName	Company Service Name	ServiceName	Service Name

GeneXus does not establish the referential relationship

On the other hand, if the attribute to which we change the name (using a subtype) is the one that plays the primary key role, then in the table in which the supertype attribute appears, it does not establish the referential relation.

Subtypes are a compromise solution to solve problems and should be used cautiously.



Both these more complex cases and the simpler ones are common in real-life applications. It is up to the developer to analyze the pros and cons of the different solutions to find the one that best suits each particular case, while keeping in mind that subtypes are a compromise solution to solve problems and should be used cautiously.

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