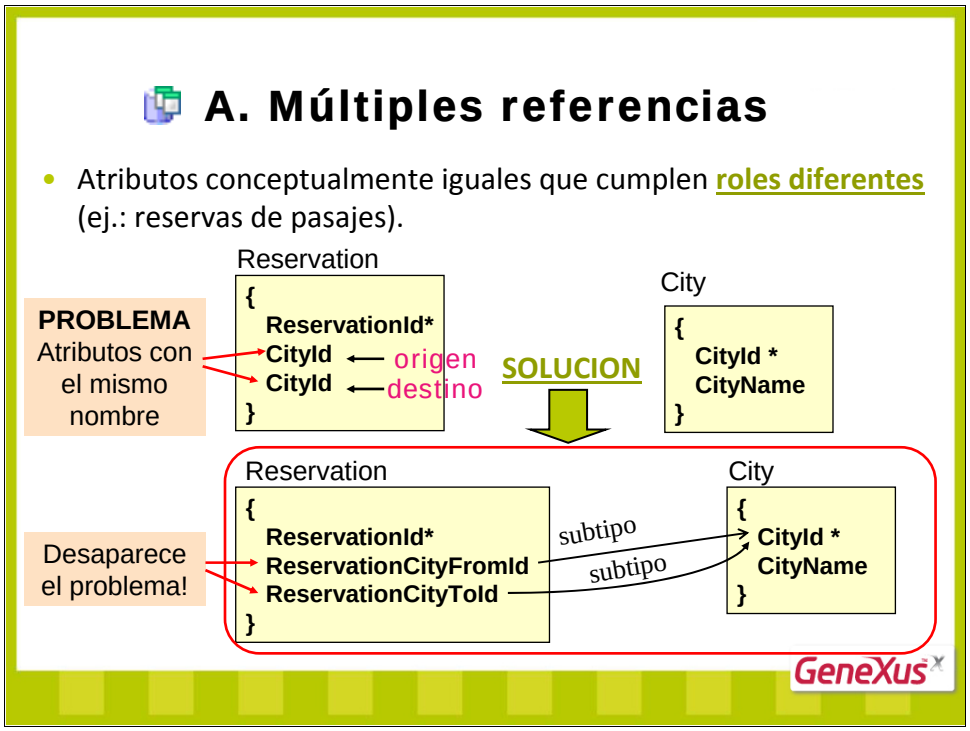
Uso de Subtipos

GeneXus

Definición de subtipos

- Las relaciones entre atributos GeneXus se establecen a través de sus nombres.
- Mediante subtipos se puede establecer que dos atributos que **se llaman diferente** corresponden al **mismo concepto**.
- Casos de subtipos:
  - A. Múltiples referencias
  - B. Especialización de un nivel (relación 1-1)
  - C. Subtipos recursivos



Realidad a representar/diseñar:

En cada reserva hay dos ciudades involucradas, las cuales cumplen roles diferentes.

El rol de una de las ciudades es el de ser la “ciudad de partida” (ciudad origen) y el rol de la otra es el de “ciudad de arribo” (ciudad destino).

El dominio de ambas ciudades es el mismo, el de la tabla CITY.

La forma de representar que tanto el “origen” como el “destino” son ciudades de la tabla CITY, es diseñando la transacción “Reservation” en la forma mencionada inicialmente en la transparencia. Sin embargo, no es posible que en la estructura de una transacción figure el mismo atributo más de una vez, pues no habría manera de identificarlos.

SOLUCIÓN: llamar a las dos ciudades de la reserva con diferentes nombres de atributos. Cualquiera de las siguientes opciones es válida. Elegimos la 3era por mayor claridad.

Opción 1) ReservationCityFromId ← ciudad origen
CityId ← ciudad destino (mismo nombre que la PK de CITY)

Opción 2) CityId ← ciudad origen (mismo nombre que la PK de CITY)
ReservationCityToId ← ciudad destino

Opción 3) ReservationCityFromId ← ciudad origen
ReservationCityToId ← ciudad destino

El problema es que al poner por ejemplo ReservationCityFromId en lugar de CityId, GeneXus deja de inferir que ReservationCityFromId corresponde al código de una ciudad de la tabla de CITY. ¿Cómo hacemos para relacionarlos, siendo que tienen diferente nombre de atributo? ver respuesta en próxima hoja …

Para estos casos GeneXus provee los SUBTIPOS, que permiten definir que **dos atributos que se llaman diferente corresponden al mismo concepto**.

En nuestro ejemplo, si definimos al atributo *ReservationCityFromId* como subtipo de *CityId*, estamos especificando que si bien *ReservationCityFromId* y *CityId* son diferentes atributos (de nombres diferentes), corresponden, no obstante, al mismo concepto (una ciudad de la tabla CITY).

Al establecer que un atributo es subtipo de otro, estamos estableciendo una dependencia funcional entre ellos.

Si *ReservationCityFromId* es **subtipo** de *CityId*, entonces decimos que *CityId* es el **supertipo** de *ReservationCityFromId*.

- Los atributos que se encuentran en una relación subtipo-supertipo comparten la misma definición (tipo de datos).

- Se realizan los controles de integridad referencial automáticamente.

- La tabla extendida que se obtiene con la definición del subtipo, es la misma que se obtendría si se utilizara directamente el supertipo.

IMPORTANTE:

Notar que este caso de múltiples referencias puede darse tanto:

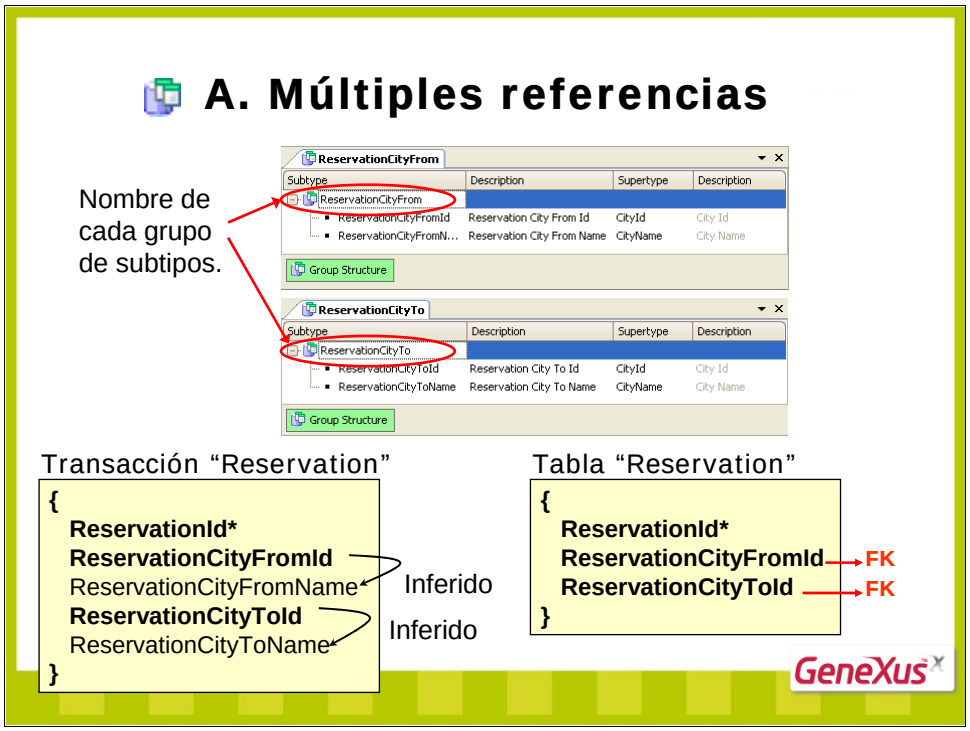
- **en la tabla base** (*)
- como **en la tabla extendida**

(*) es el caso del ejemplo, en el que **en la propia tabla** (RESERVATION) hay más de una referencia a otra tabla (CITY) y con valores diferentes.

RESUMIENDO:

siempre que desde una tabla se accede a otra que está en su tabla extendida **por "más de un camino" y con "valores diferentes"**, es necesario definir SUBTIPOS, para poder llamarle diferente a los atributos y haciéndose automáticamente todos los controles de integridad referencial.

Una vez definidos los grupos de subtipos que sean necesarios, la forma de indicarle a GeneXus cuál de los caminos debe tomar para acceder a la tabla destino, es mencionando los nombres de atributos que correspondan. Ej.: mencionar *ReservationCityFromName* si lo que se necesita en ese momento es el nombre de la ciudad origen, o mencionar *ReservationCityToName* si lo que se necesita es el nombre de la ciudad destino.



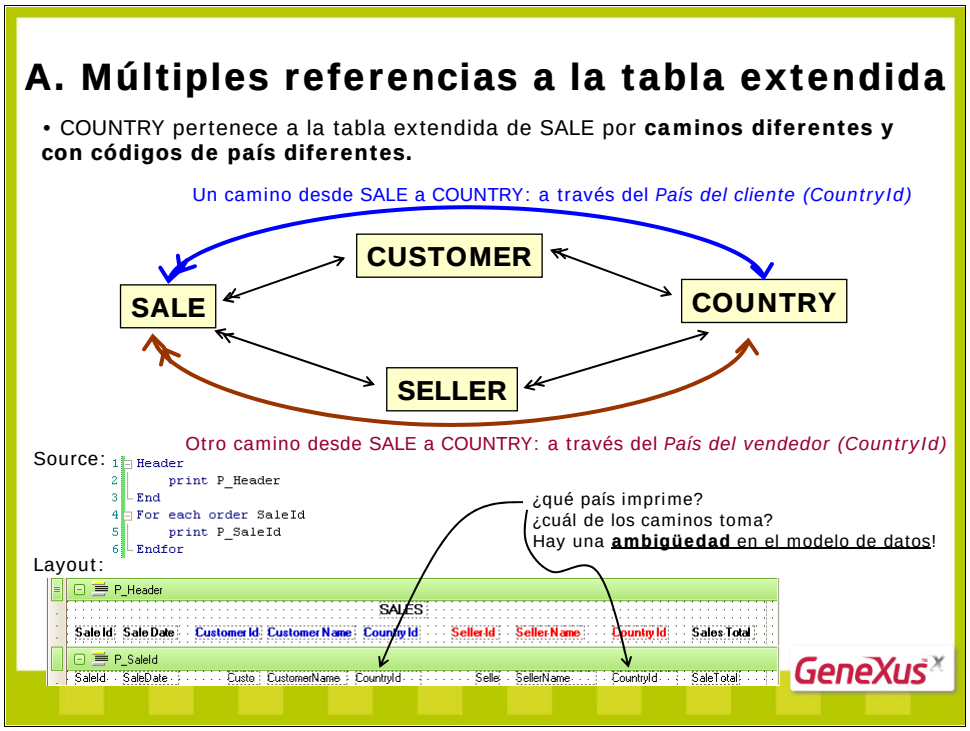
Con el grupo estamos indicando que los atributos pertenecientes al mismo grupo de subtipos, están relacionados. Por ej., en nuestro ejemplo, GeneXus sabrá que el atributo *ReservationCityToName* será inferido a través del atributo *ReservationCityToId (y no a través del ReservationCityFromId).* Esto es por pertenecer ambos al mismo grupo (al de nombre ReservationCityTo).

Cuando el usuario digite un valor sobre *ReservationCityToId*, no solo se va a hacer automáticamente el control de integridad referencial (que exista un ciudad con ese código en la tabla CITY), sino que se va a inferir en *ReservationCityToName* el nombre correspondiente a ese código de ciudad.

IMPORTANTE: Todo grupo de subtipos, debe contener un atributo o conjunto de atributos, cuyos supertipos, juntos, correspondan a la clave primaria de una tabla del modelo.

Los demás atributos del grupo deberán ser de tipo “Inferred”, es decir, deberán poder inferirse a través de esa clave.

En caso contrario estará mal definido el grupo.



Si quisiéramos por ejemplo listar las ventas (SALE), y de c/u de ellas mostrar los datos del cliente (nombre, país, etc.) y del vendedor (nombre, país, etc.):

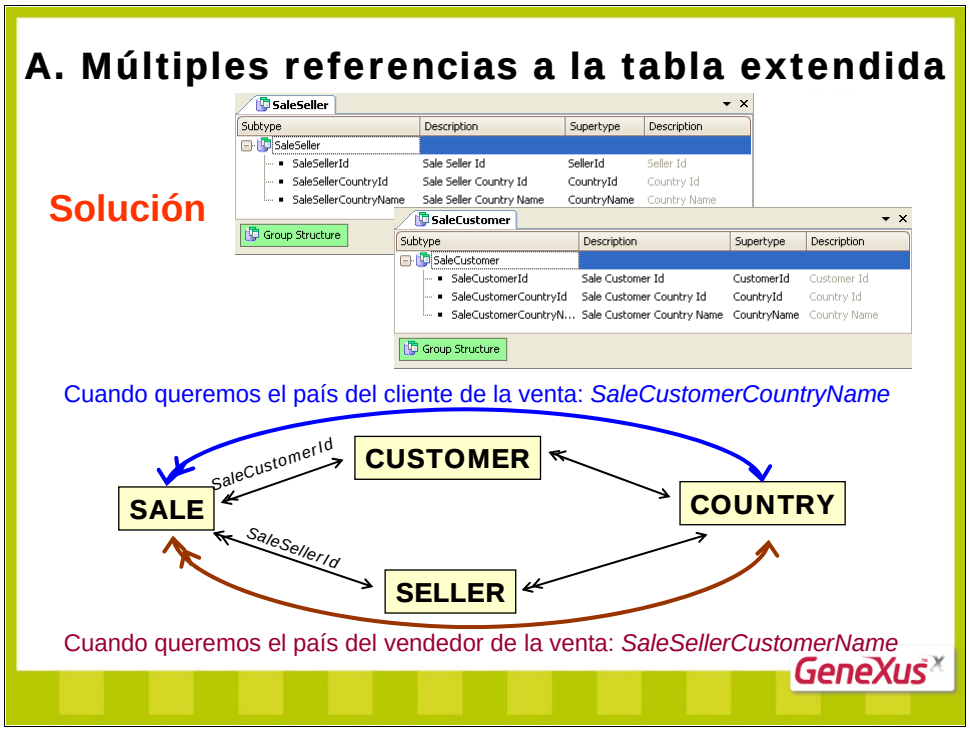
• necesitamos un for each con tabla base SALE y acceder a través de su extendida a las tablas CUSTOMER, SELLER y COUNTRY para listar los atributos secundarios del cliente, vendedor y país respectivamente.

Problema:

Los atributos de nombre *CountryId*, *CountryName* y *todos los de la tabla extendida de COUNTRY* pertenecen a la tabla extendida de SALE por dos caminos diferentes: 1) a través del país del cliente y 2) a través del país del vendedor.

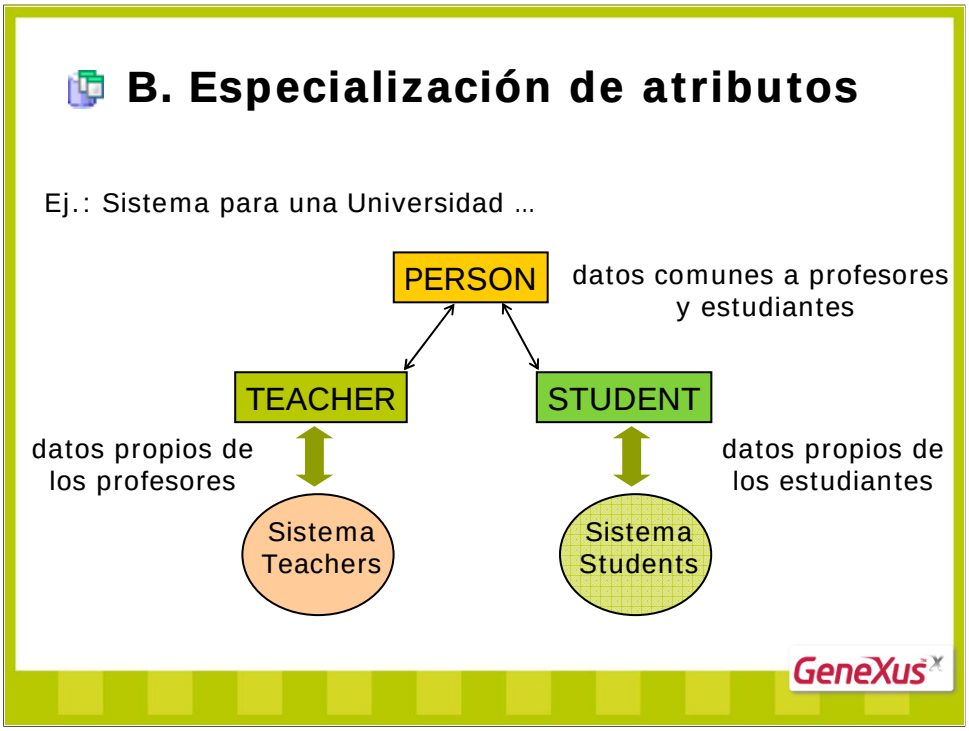
Solución:

Debemos diferenciarlos, llamarlos con **diferente nombre de atributo** pero queriendo que **se sigan representando todas las relaciones y haciéndose automáticamente todos los controles de integridad referencial.**



Una vez definidos los dos grupos de subtipos que se muestran en la figura, y haciendo el cambio correspondiente en la estructura de la transacción Sale, ¡queda resuelta la ambigüedad en el modelo de datos!

Una vez definidos los subtipos, tenemos que recordar usar el nombre de atributo que corresponda a lo que queremos acceder. Por ejemplo, en todos aquellos objetos GeneXus en los cuales queramos acceder al *código o al nombre del país del cliente de la venta* debemos usar los atributos *SaleCustomerCountryId y SaleCustomerCountryName* respectivamente.



Caso de subtipos "Especialización de atributos":
Cuando se está modelando una categorización. Generalmente es utilizada cuando un objeto del negocio comparte todas las características de otro objeto, pero agrega algunas más. La diferencia puede estar tanto en las propiedades, como en el comportamiento que tendrá.

Ejemplo "Sistema para una Universidad":

En este ejemplo, el profesor y el alumno tienen **roles y comportamientos claramente diferenciados**. Por ejemplo, el profesor tendrá cursos asignados, sueldo, etc. El alumno estará inscripto a un curso, tendrá asignados pagos, asistencia, escolaridad, etc.

Estamos frente a un caso en el que los roles y el tratamiento de las entidades de la categorización están claramente diferenciados.

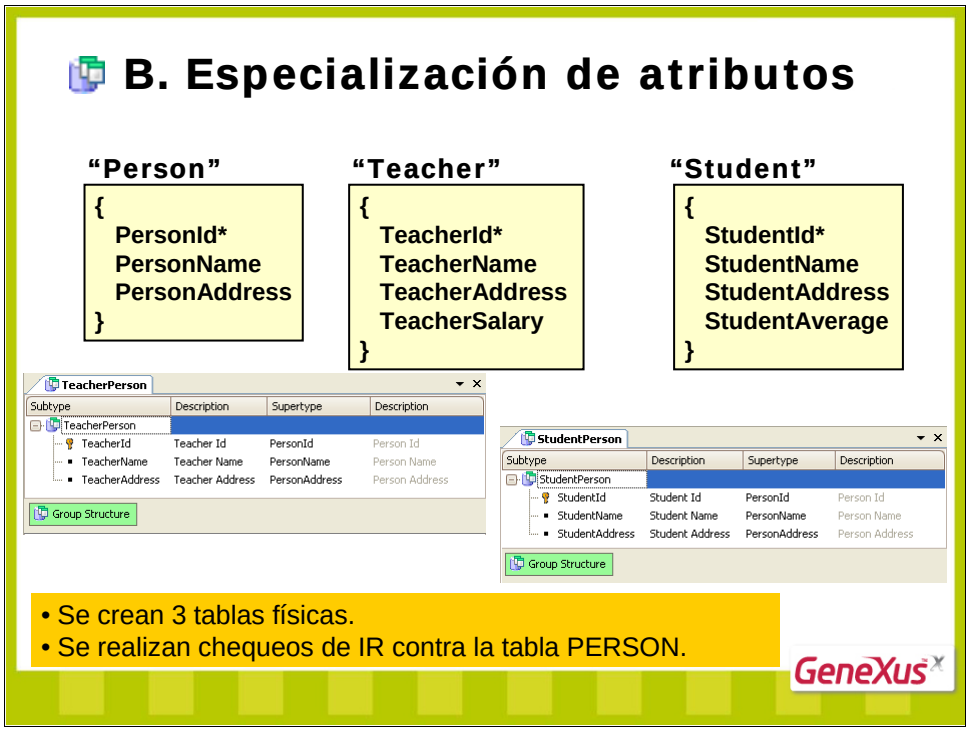
Tanto los estudiantes como los docentes comparten información común (ambos tienen un nombre, una dirección, etc) pero también tienen información que difiere, que es propia de c/u de ellos.

Para representar esta realidad, se crean las tres transacciones: "Person", "Teacher" y "Student".

En la transacción "Person" figura la información común. Para representar que tanto los estudiantes como los docentes son personas, se utilizan los subtipos.

Al definir que el identificador de "Teacher" es subtipo del identificador de "Person" estamos estableciendo esta relación.

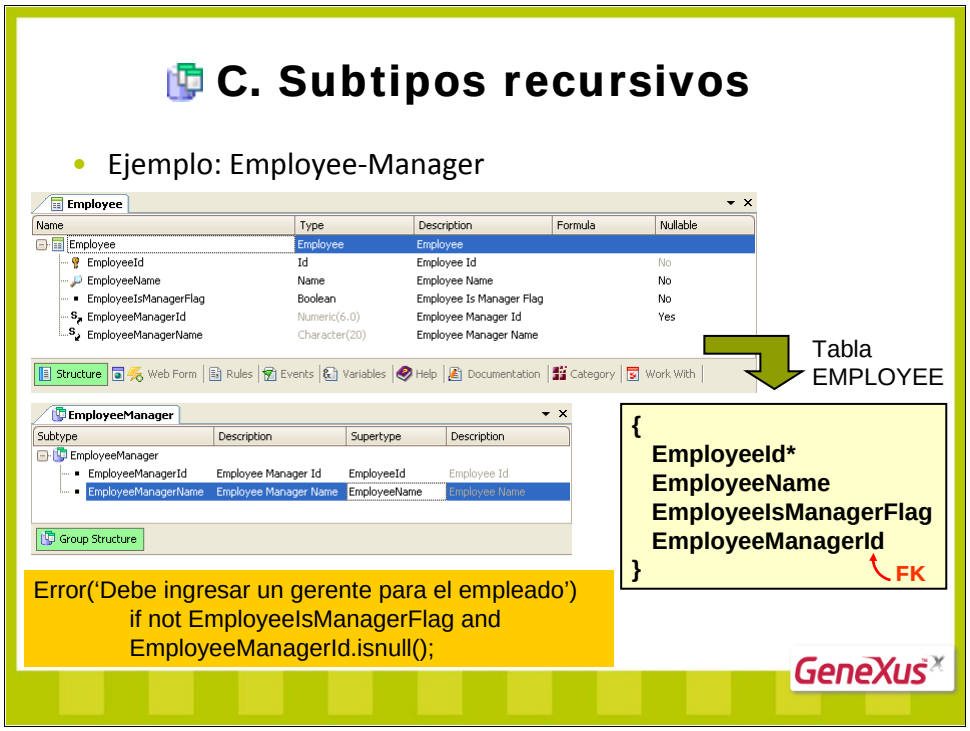
Cada vez que se inserte un registro en la tabla TEACHER a través de su transacción, se realizará el chequeo de integridad referencial contra "Person". Asimismo, cada vez que se intente eliminar un registro de "Person", se verificará primeramente que no exista ningún registro en la tabla TEACHER (ni en STUDENT) con el mismo valor en la clave primaria.



La transacción “Teacher” tiene asociada una tabla que contendrá físicamente sólo dos atributos: *TeacherId* y *TeacherSalary*.

Al ser *TeacherId* identificador de la transacción, será la clave primaria de la tabla asociada. Además, al ser un subtipo de *PersonId*, será una clave foránea a la tabla PERSON. Por lo tanto, se harán los chequeos de integridad referencial correspondientes.

Los atributos *TeacherName* y *TeacherAddress* son subtipos de *PersonName* y de *PersonAddress* respectivamente y están agrupados con *TeacherId*, por lo que serán inferidos de la tabla PERSON, a través de la clave foránea *TeacherId* (no están almacenados en la tabla TEACHER).



Es posible tener una tabla subordinada a sí misma definiendo subtipos.

Este tipo de subtipos se utiliza para modelar las relaciones recursivas. Por ejemplo, la relación entre Empleado y Gerente:

- cada empleado tiene un gerente. Un gerente, a su vez, es un empleado (aquí está la recursión).

- un gerente puede tener varios empleados a su cargo.

Si además la realidad a representar es que "sólo los empleados que no son gerentes tienen un gerente", entonces, cuando se ingresan los datos hay que realizar los siguientes controles:

- cuando se ingresan los gerentes, hay que permitir dejar en nulo el atributo *EmployeeManagerId.* Para esto, cambiamos a 'Yes' la columna Nulls del atributo *EmployeeManagerId,* el cual es FK en la tabla EMPLOYEE.

- que todo empleado que no es gerente, tenga un gerente. Este control lo hacemos con la regla error que se muestra en la figura.

El atributo *EmployeeManagerName* no queda almacenado en la tabla EMPLOYEE, se infiere luego de ingresar un valor en *EmployeeManagerId.*

Por ser *EmployeeManagerId* subtipo de *EmployeeId,* se realizan automáticamente los controles de integridad referencial de la tabla consigo misma. Esto se puede ver en la navegación de la transacción, como se muestra en la siguiente página.

C. Subtipos recursivos

- Listado de navegación detallado:

Transaction Employee Navigation Report

Name	Employee	Environment	C#
Description	Employee	Spec. Version	10_0_1-13868
		Form Class	HTML
		Program Name	Employee
		Parameters	

Levels

Level Employee

=Employee(*EmployeeId*)
~Employee(*EmployeeManagerId*)

Insert into Employee
(EmployeeId, EmployeeName, EmployeeIsManagerFlag, EmployeeManagerId)

Update Employee (EmployeeName, EmployeeIsManagerFlag, EmployeeManagerId)

Delete from Employee

Referential integrity controls on delete:
- Employee(*EmployeeManagerId*)

Prompts

Table	Program	In Parameters	Out Parameters
Employee			EmployeeManagerId
Employee			EmployeeId

Consideraciones

- Cuando se define un subtipo éste "hereda" la definición del supertipo.
- Al menos uno de los supertipos del grupo (o conjunto de supertipos del grupo) debe(n) corresponder a la PK de una tabla del modelo.