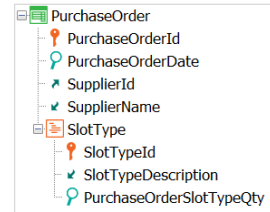
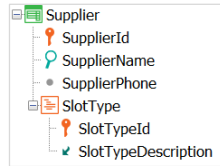
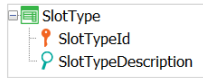


Logic subordination

GeneXus[™]

In this video, we will address the topic of logical subordination in GeneXus.

Reality: Casino



The type of slot that is being entered in the lines must be provided by the supplier of the order.

To do so, we will focus on a KB where we are developing an application for a casino, which, among other things, allows registering the purchase orders of game slots from its suppliers.

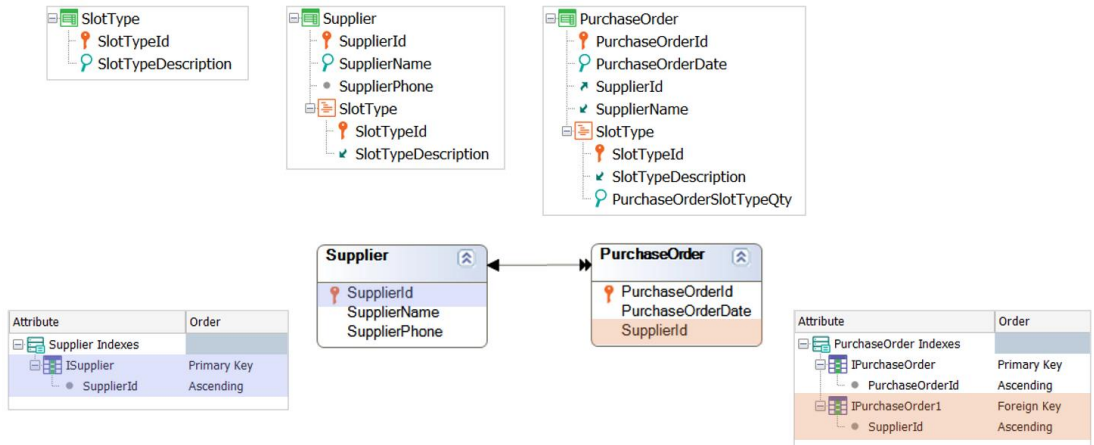
We will use the SlotType transaction to register the types of game slots, the Supplier transaction to register the suppliers and the slots they offer, and finally—the transaction we are going to focus on for this case study—PurchaseOrder, to register the purchase orders for certain types of slots that the casino places with one of its suppliers.

When entering a purchase order, it must be checked in the PurchaseOrder transaction that the type of slot being entered in the lines is provided by the supplier of the order (first-level attribute).

Is this ensured by the design?

The answer to the question is no. With this design, this check will not be made. Let's see why.

Subordination relationship



We'll start with a simpler case to see the difference. Between the first level of PurchaseOrder and the first level of Supplier there is a subordination relationship. In other words: SupplierId will be a foreign key in the PurchaseOrder table, pointing to the Supplier table, which has SupplierId as its primary key. For this reason, an index will be automatically created in Supplier by primary key and an index in PurchaseOrder by foreign key.

Uniqueness and referential integrity controls

- The index by primary key in Supplier will be used to control that:

Attribute	Order
Supplier Indexes	
ISupplier	Primary Key
SupplierId	Ascending

- When inserting a supplier, there isn't another one with the same SupplierId value
- When an order for a supplier is inserted, it exists.

Order Id	<input type="text" value="1"/>
Order Date	<input type="text" value="09/15/22"/>
Supplier Id	<input type="text" value="5"/> <input checked="" type="checkbox"/> No matching 'Supplier'.
Supplier Name	<input type="text"/>

These indexes will be used to make uniqueness and referential integrity controls efficiently. That is to say:

- The index by primary key in Supplier will be used to control that:
 - When inserting a supplier there is no other with the same value of SupplierId, and that
 - When an order for a supplier is inserted, it exists.

Uniqueness and referential integrity controls

- The index by primary key in Supplier will be used to control that:

Attribute	Order
Supplier Indexes	
ISupplier	Primary Key
SupplierId	Ascending

- When inserting a supplier, there isn't another one with the same SupplierId value
- When an order for a supplier is inserted, it exists.

Order Id	<input type="text" value="1"/>
Order Date	<input type="text" value="09/15/22"/>
Supplier id	<input type="text" value="5"/> No matching 'Supplier'
Supplier Name	<input type="text"/>

- The index by foreign key in PurchaseOrder will be used to control that:

Attribute	Order
PurchaseOrder Indexes	
IPurchaseOrder	Primary Key
PurchaseOrderId	Ascending
IPurchaseOrder1	Foreign Key
SupplierId	Ascending

- When deleting a supplier, there is no order referencing it.

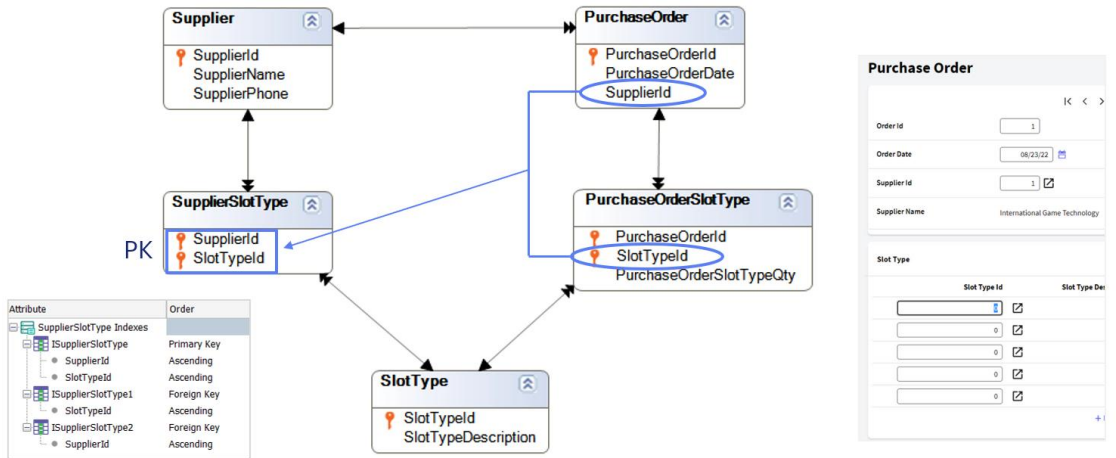
❗ Invalid delete, related information in Purchase Order

- The index by foreign key in PurchaseOrder will be used to control that:
 - When deleting a supplier there is no order referencing it.

Since there is a (physical) subordination between Supplier and PurchaseOrder, these checks are performed.

Now, what happens when we enter a slot type in the PurchaseOrder transaction?

Table relationships



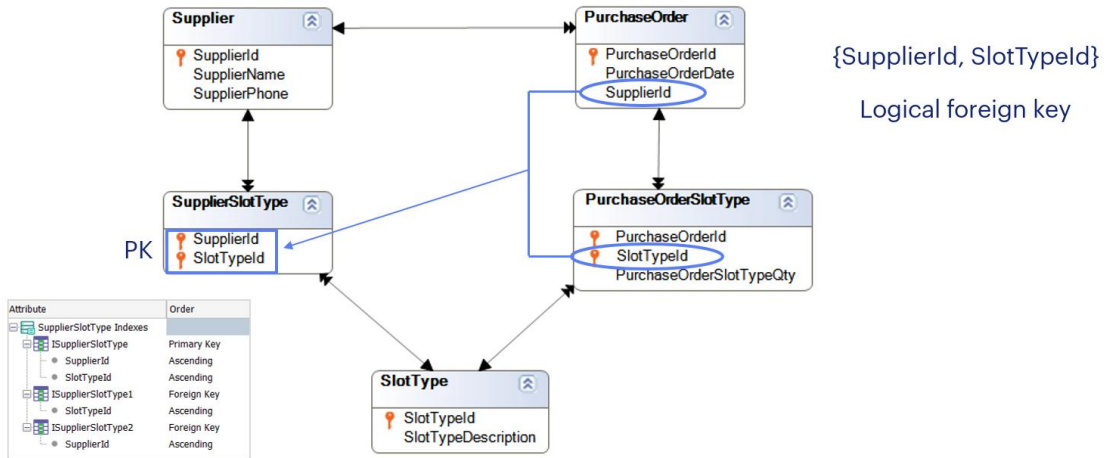
Let's look at the relationship between the tables:

When a record is entered in the PurchaseOrderSlotType table, we want to check in the SupplierSlotType table whether there is a record with the SupplierId value corresponding to PurchaseOrderId, and with the value of SlotTypeId corresponding to the SlotTypeId of the record being inserted, since that information is stored in memory.

That is to say, when trying to insert a new SlotTypeId in the grid of the PurchaseOrder transaction, you have the SlotTypeId value that the user has just entered or selected (using the selection list), and also the SupplierId value (specified by the user on the first level of the transaction). Since these two attributes—together—constitute the primary key of the SupplierSlotType table, then there is an index in that table by primary key that would allow the integrity check to be made efficiently. But, will it be done?

The answer is no.

Table relationships



Note that the difference with the simple case we saw at the beginning is that actually {SupplierId, SlotTypeId} is not a foreign key in PurchaseOrder, because the two attributes are not in the same table. However, we could say that they form a logical foreign key, even though they don't exist for relational databases. The problem is that GeneXus performs referential integrity checks if it understands that there is a physical subordination relationship between the tables. Therefore, it will not make them in this case, allowing you to enter a type of slot that is not provided by the order supplier.

Running example

SUPPLIER TABLE

SupplierId	SupplierName	SupplierPhone
1	International Game Technology	11222233
2	Pragmatic Play	55537777

SLOTTYPE TABLE

SlotTypeId	SlotTypeDescription
1	Classic slot
2	Five-reel slot
3	Virtual reality slot
4	3D slot

SUPPLIERSLOTTYPE TABLE

SupplierId	SlotTypeId
1	3
1	4
2	1

Purchase Order

|< < > >| SELECT

Order Id	<input type="text" value="1"/>
Order Date	<input type="text" value="08/23/22"/> 📅
Supplier Id	<input type="text" value="1"/> ↩️
Supplier Name	International Game Technology ↩️

Slot Type			
	Slot Type Id	Slot Type Description	Type Qty
-	<input type="text" value="1"/> ↩️	Classic slot	<input type="text" value="5"/>
-	<input type="text" value="2"/> ↩️	Five-reel slot	<input type="text" value="3"/>
	<input type="text" value="0"/> ↩️		<input type="text" value="0"/>
	<input type="text" value="0"/> ↩️		<input type="text" value="0"/>
	<input type="text" value="0"/> ↩️		<input type="text" value="0"/>

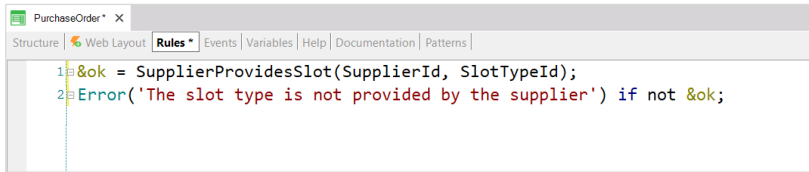
+ NEW ROW

Let's see this in the following example: the supplier with identifier 1, named *International Game Technology*, provides only the slots with identifiers 3 and 4 (*Virtual reality slot* and *3D slot* respectively). However, in the purchase order we indicate that we will buy from that supplier the slots of type 1 and 2 (precisely the ones they do not provide).

How can we solve this problem?

Possible solution

In the rules, invoke a procedure that performs the control:



```
PurchaseOrder* X
Structure | Web Layout | Rules | Events | Variables | Help | Documentation | Patterns
1: &ok = SupplierProvidesSlot(SupplierId, SlotTypeId);
2: Error('The slot type is not provided by the supplier') if not &ok;
```

The first thing we can think of is to invoke a procedure in the PurchaseOrder transaction rules that performs the check:

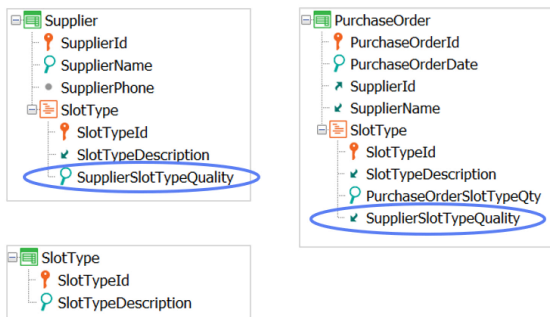
```
&ok = SupplierProvidesSlot(SupplierId, SlotTypeId);
```

And in case of returning False, show the following error:

```
Error('The slot type is not provided by the supplier') if not &ok;
```

Another possible solution

Infer in PurchaseOrder a secondary attribute of Supplier's second level:



Purchase Order

|< < > >| SELECT

Order Id

Order Date

Supplier Id

Supplier Name International Game Technology

Slot Type

Slot Type Id	Slot Type Description	Type Qty
<input type="text" value="1"/>	● No matching 'Slot Type'.	<input type="text" value="0"/>
<input type="text" value="0"/>	<input type="checkbox"/>	<input type="text" value="0"/>
<input type="text" value="0"/>	<input type="checkbox"/>	<input type="text" value="0"/>
<input type="text" value="0"/>	<input type="checkbox"/>	<input type="text" value="0"/>
<input type="text" value="0"/>	<input type="checkbox"/>	<input type="text" value="0"/>

We would have achieved the same effect in another way, without writing anything, if in the Supplier transaction there had been a secondary attribute on the second level, so as to infer it in PurchaseOrder, and force GeneXus to notice the relationship.

A “No matching” error will be displayed if the type of slot that is being entered in the lines is not provided by the supplier in the order.

Another possible solution

Infer in PurchaseOrder a secondary attribute of Supplier's second level:

The diagram illustrates the relationship between the **Supplier** and **PurchaseOrder** entities. The **Supplier** entity has attributes: **SupplierId** (primary key), **SupplierName**, **SupplierPhone**, **SlotType** (collection), **SlotTypeId**, **SlotTypeDescription**, and **SupplierSlotTypeQuality** (circled in blue). The **PurchaseOrder** entity has attributes: **PurchaseOrderId** (primary key), **PurchaseOrderDate**, **SupplierId**, **SupplierName**, **SlotType** (collection), **SlotTypeId**, **SlotTypeDescription**, **PurchaseOrderSlotTypeQty**, and **SupplierSlotTypeQuality** (circled in blue). A dashed blue line indicates a relationship between the **SupplierSlotTypeQuality** attribute in the **Supplier** entity and the **SupplierSlotTypeQuality** attribute in the **PurchaseOrder** entity.

The screenshot shows the **Purchase Order** form. The **Order Id** is 2, **Order Date** is 08/30/22, **Supplier Id** is 1, and **Supplier Name** is International Game Technology. The **Slot Type** section shows a table with columns **Slot Type Id**, **Slot Type Description**, and **Type Qty**. The first row has **Slot Type Id** 1, **Slot Type Description** "The slot type is not provided by the supplier", and **Type Qty** 0. A red error message "The slot type is not provided by the supplier" is displayed next to the **Slot Type Id** field.

```

1= RefMsg('The slot type is not provided by the supplier',
2= SupplierId, SlotTypeId);

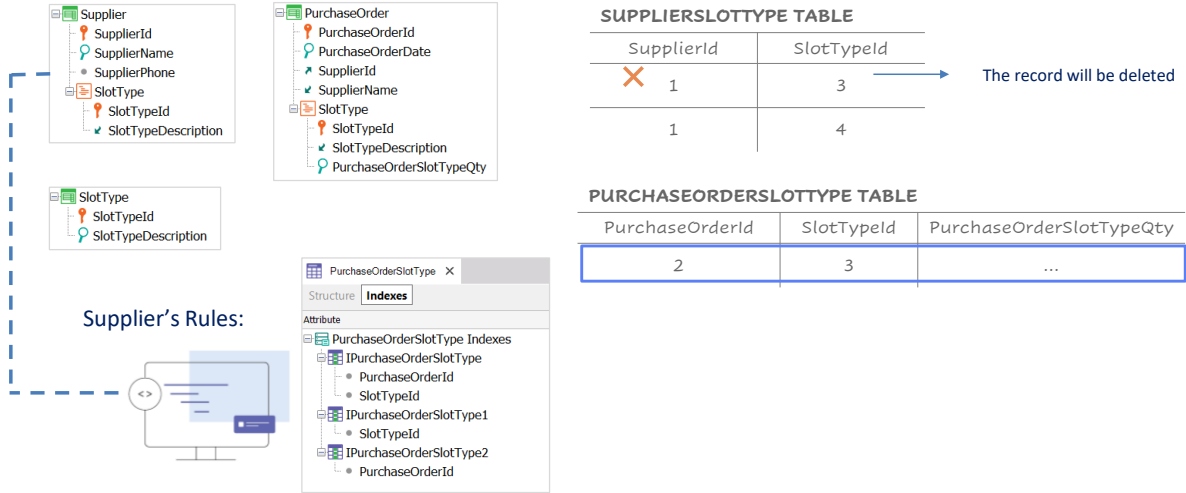
```

We could use the RefMsg rule, which is used to customize the error message when a referential integrity check is made on insert or update from a foreign key, and it fails.

This rule has as parameters—in addition to the message that we want to be displayed in case the RI check fails—the attribute(s) that make up the foreign key.

Something to think about:

What will happen in this solution if you delete a slot type from a supplier for which orders already exist?



An interesting question is what will happen with this solution if you want to delete a slot type from a supplier for which orders already exist.

That is to say, if the user, through the “Supplier” transaction tries to delete a line for which there is a “PurchaseOrder” with that supplier and slot type.

In this case, GeneXus will not be able to make a referential integrity check because there is no index by {SupplierId, SlotTypeId} in the PurchaseOrderSlotType table. Since that foreign key only exists at a logical level (in the extended table), there will be no index at all—given that an index corresponds to a physical concept. Therefore, this check will have to be programmed in the rules of the “Supplier” transaction, invoking a procedure to do it and then triggering an error according to the result returned.

GeneXus™

training.genexus.com

wiki.genexus.com

training.genexus.com/certifications