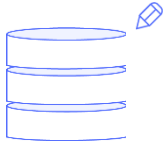


Database update

Business Components. Rules, events, and checks. Review

GeneXus™



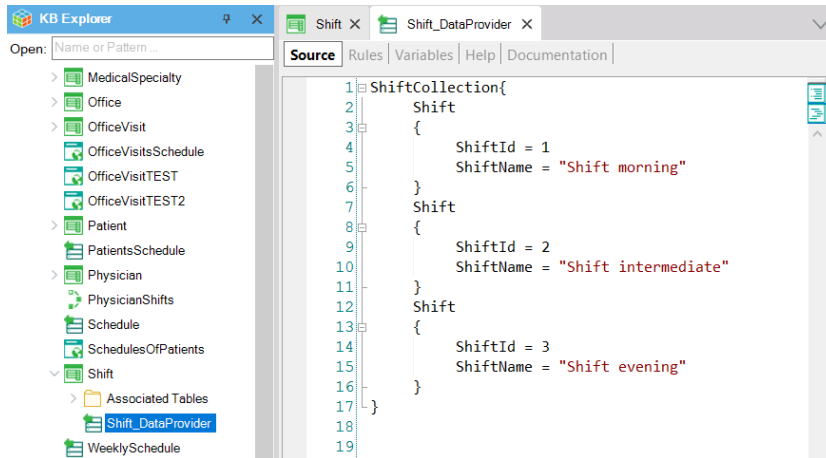
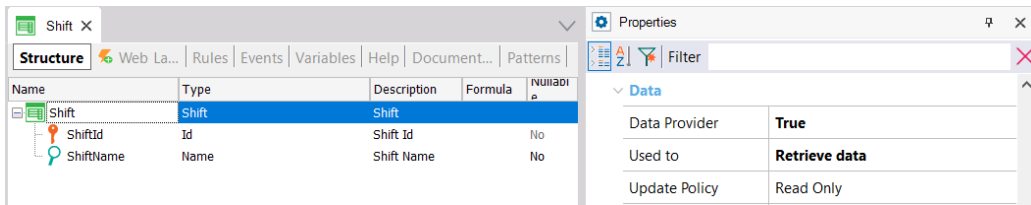
Insert, Update, Delete

Business Component

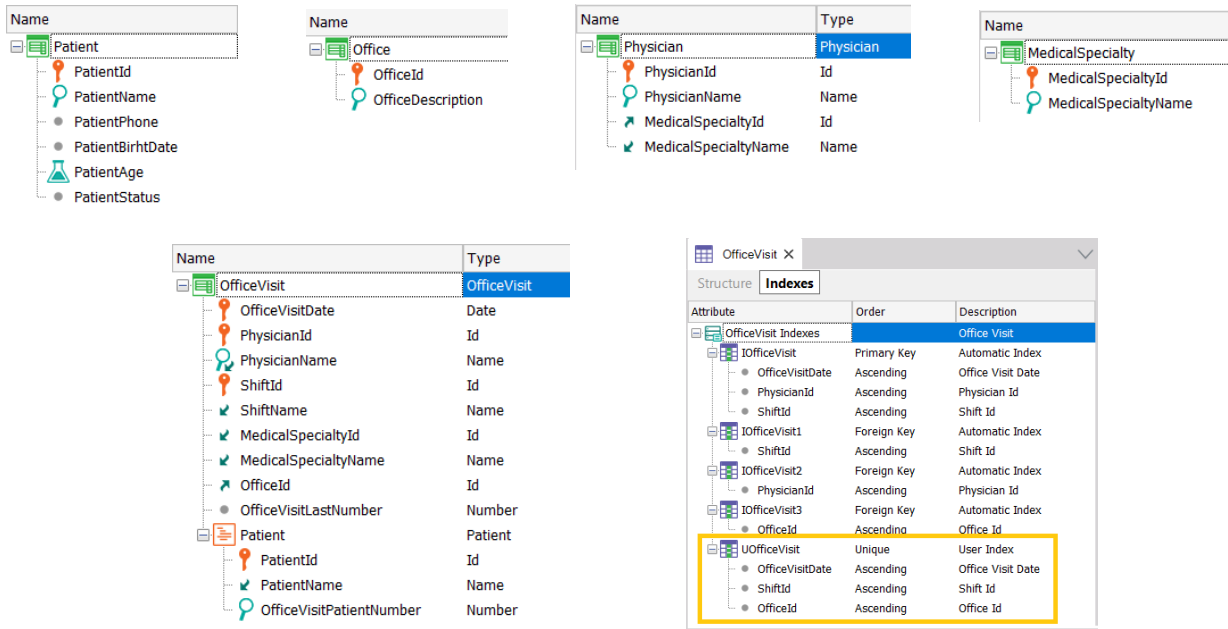
Hospital KB

We are going to integrate what we know about updating the database through Business Components and add some more advanced concepts.

For this we are going to focus on a KB that we are developing for a health care service...



... that offers its patients the possibility to see doctors in one of three shifts: morning, intermediate, or evening. To represent these shifts we have the Shift transaction that is dynamic (we turned on the Data Provider property and said that it was to retrieve data, and only to retrieve them, because we've set it as Read Only). Here we indicate the 3 shifts.



In addition, we have the Patient transaction to record the patients; Office to record the offices in which these doctor's visits will take place; and Physician to record the information of each doctor—doctors have one (and only one) medical specialty. Finally, and we are going to focus on this one, the OfficeVisit transaction that records each office visit in which a doctor sees a list of patients on a date and shift (morning, intermediate, or evening).

Since patients are to be assigned a number to receive care, an attribute is added on the first level to record the last number given, so that a serial rule can be used each time a new patient is entered through the transaction.

It is also necessary to indicate the office where the visit will take place. We can already note that although we have defined that the trio composed of “office visit date, physician, and shift” cannot be repeated (it is the primary key), neither should the office visit date, shift, and office be repeated (that is, there cannot be two office visits on the same date, shift, and office). Therefore, these three attributes form a candidate key, and we create a unique index to represent it (and secure it).

Name	Type
OfficeVisit	OfficeVisit
OfficeVisitDate	Date
PhysicianId	Id
PhysicianName	Name
ShiftId	Id
ShiftName	Name
MedicalSpecialtyId	Id
MedicalSpecialtyName	Name
OfficeId	Id
OfficeVisitLastNumber	Number
Patient	Patient
PatientId	Id
PatientName	Name
OfficeVisitPatientNumber	Number

```

1  /* Generated by Work With Pattern [Start] - Do not change */
2  [web]
3  {
4  param(in:&Mode, in:&OfficeVisitDate, in:&PhysicianId, in:&ShiftId);
5
6  OfficeVisitDate = &OfficeVisitDate if not &OfficeVisitDate.IsEmpty();
7  noaccept(OfficeVisitDate) if not &OfficeVisitDate.IsEmpty();
8  PhysicianId = &PhysicianId if not &PhysicianId.IsEmpty();
9  noaccept(PhysicianId) if not &PhysicianId.IsEmpty();
10 ShiftId = &ShiftId if not &ShiftId.IsEmpty();
11 noaccept(ShiftId) if not &ShiftId.IsEmpty();
12
13 OfficeId = &Insert_OfficeId if &Mode = TrnMode.Insert and not &Insert_OfficeId.IsEmpty();
14 noaccept(OfficeId) if &Mode = TrnMode.Insert and not &Insert_OfficeId.IsEmpty();
15 /* Generated by Work With Pattern [End] - Do not change */
16
17 Serial(OfficeVisitPatientNumber, OfficeVisitLastNumber, 1);
18
19 noaccept(OfficeVisitPatientNumber);
20
21 &shifts = PhysicianShifts(OfficeVisitDate, PhysicianId)
22   If Insert;
23 error("Physician " + PhysicianId.ToString() + " already has 2 shifts assigned for the date: "
24   + OfficeVisitDate.ToString(), PhysicianNotAvailable) if &shifts >= 2 and Insert;
25
26 &IsOk = isPatientOk(PatientId, MedicalSpecialtyId)
27   if Insert;
28 //checks there is no other office visit pending for the patient for the same specialty
29
30 error("There is a pending office visit scheduled for this patient for the same specialty",
31   PatientAlreadyScheduledForMedicalSpecialty)
32   if not &IsOk and Insert
33   Level PatientId;
34

```

In addition, we will want to add two business rules: the same physician can see patients in up to two shifts on the same day, but not three. On the other hand, it should not be possible to enter a patient who already has an office visit scheduled with a doctor of the same specialty.

We check them in this way:

Here we are calling this procedure only if in Insert mode; it counts the number of office visits of the physician on the date we want to assign a new appointment. And we will fire an error if it returns 2 or more.

Here we have the serial rule so that when a new patient is being entered for the office visit, the next number to see the doctor is assigned. And here we disable the field, because we don't want the user to modify it.

We are calling another procedure that checks if the patient has a pending office visit for the same specialty as the office visit for which we are trying to schedule them. In that case, an error is triggered.

We can see that the procedures and the error rules are only executed when we try to **insert**: the header for one case and a line for the other.

Also, note that since we have applied the **WorkWith** pattern to the transaction, it has automatically added these rules (among them, the **parm** rule to receive the key and mode).

Our objective will be to review and explore in depth what of all this is

executed when the operations are made through the Business Component.

	Monday	Tuesday	Wednesday	Thursday	Friday
Shift 1 morning	Physician: 1 Office: 2		Physician: 2 Office: 1	Physician: 1 Office: 2 Patient: 2	
Shift 2 intermediate	Physician: 1 Office: 2				
Shift 3 evening		Physician: 1 Office: 2			

Let's assume we have this data in the database for next week, with no patients scheduled yet except for Thursday, which has patient 2.

	Monday	Tuesday	Wednesday	Thursday	Friday
Shift 1 morning	Physician: Office:				
Shift 2 intermediate	Physician: Office:				
Shift 3 evening					

Office Visits

apps5.genexus.com/ld9df182c70350001219afeaceeb30e9e2/wwofficevisit.aspx

Hospital Backoffice by GeneXus

Recents Office Visit — Office Visits

SHOW FILTERS **Office Visits** + INSERT

Visit Date	Physician Id	Physician Name	Shift Name	Medical Speci...	Office Id	Last Number		
11/14/22	1	Doctor 1	Shift morning	Family Medicine	2	0	UPDATE	DELETE
11/14/22	1	Doctor 1	Shift intermediate	Family Medicine	2	0	UPDATE	DELETE
11/15/22	1	Doctor 1	Shift evening	Family Medicine	2	0	UPDATE	DELETE
11/16/22	2	Doctor 2	Shift morning	Family Medicine	1	0	UPDATE	DELETE
11/17/22	1	Doctor 1	Shift morning	Family Medicine	2	1	UPDATE	DELETE

Here we see them in the Work With.

We will want to insert a new office visit for Monday, and test that it works as expected in all cases.

Checks	Error Id	Error Description
FK: PhysicianId	ForeingKeyNotFound	No matching 'Physician'
<code>error("Physician " + PhysicianId.ToString() +" already has 2 shifts assigned for the date: " + OfficeVisitDate.ToString(), PhysicianNotAvailable) if &shifts >= 2 and Insert;</code>	PhysicianNotAvailable	Physician... already has 2 shifts...
FK: ShiftId	ForeingKeyNotFound	No matching 'Shift'
FK: OfficeId	ForeingKeyNotFound	No matching 'Office'
CK: {OfficeVisitDate, ShiftId, OfficeId}		Office Visit Date, Shift Id, Office Id already exists
PK: {OfficeVisitDate, PhysicianId, ShiftId}	DuplicatePrimaryKey	Record already exists
FK: PatientId	ForeingKeyNotFound	No matching 'Patient'
<code>error("There is a pending office visit scheduled for this patient for the same specialty", PatientAlreadyScheduledForMedicalSpecialty) if not &IsOk and Insert Level PatientId;</code>	PatientAlreadyScheduledForMedicalSpecialty	There is a pending office visit scheduled...
PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}	DuplicatePrimaryKey	Record already exists

Let's summarize everything we know that the running transaction will check when trying to insert data, and then we will see what the Business Component checks.

Checks

FK: PhysicianId

```
error("Physician " + PhysicianId.ToString() + " already has 2 shifts assigned for the date: "
+ OfficeVisitDate.ToString(), PhysicianNotAvailable) if &shifts >= 2 and Insert;
```

FK: ShiftId

FK: Officeld

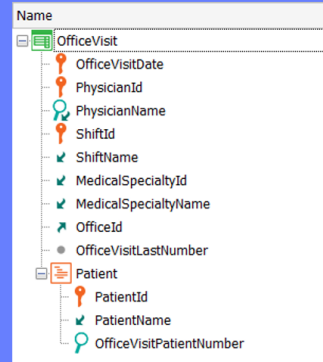
CK: {OfficeVisitDate, ShiftId, Officeld}

PK: {OfficeVisitDate, PhysicianId, ShiftId}

FK: PatientId

```
error("There is a pending office visit scheduled for this patient for the same specialty",
PatientAlreadyScheduledForMedicalSpecialty)
if not &isOk and Insert
Level PatientId;
```

PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}



Because of the order of the attributes in the transaction structure, the order of the checks will be as shown.

Checks

FK: PhysicianId

```
error("Physician " + PhysicianId.ToString() + " already has 2 shifts assigned for the date: "
+ OfficeVisitDate.ToString(), PhysicianNotAvailable) if &shifts >= 2 and Insert;
```

FK: ShiftId

FK: Officeld

CK: {OfficeVisitDate, ShiftId, Officeld}

PK: {OfficeVisitDate, PhysicianId, ShiftId}

FK: PatientId

```
error("There is a pending office visit scheduled for this patient for the same specialty",
PatientAlreadyScheduledForMedicalSpecialty)
if not &IsOk and Insert
Level PatientId;
```

PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}

Office Visit

apps5.genexus.com/id9df182c70350001219afeaceeb30e8e2/office...

GeneXus

Hospital Backoffice by GeneXus

Recents Office Visits — Office Visit

Office Visit

Visit Date 141122 Invalid date

Physician Id

Physician Name

Shift Id 0

Shift Name

Medical Specialty Id 0

Medical Specialty Name

Office Id 0

We have not added the data type checks. For example, for the first field it will clearly check that the date is valid.

Checks

FK: PhysicianId

```
error("Physician " + PhysicianId.ToString() + " already has 2 shifts assigned for the date: "
+ OfficeVisitDate.ToString(), PhysicianNotAvailable) if &shifts >= 2 and Insert;
```

FK: ShiftId

FK: Officeld

CK: {OfficeVisitDate, ShiftId, Officeld}

PK: {OfficeVisitDate, PhysicianId, ShiftId}

FK: PatientId

```
error("There is a pending office visit scheduled for this patient for the same specialty",
PatientAlreadyScheduledForMedicalSpecialty)
if not &IsOk and Insert
Level PatientId;
```

PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}

Office Visit

Recents Office Visits — Office Visit

Office Visit

Visit Date: 11/14/22

Physician Id: 25 (No matching 'Physician')

Physician Name

Shift Id

Shift Name

Medical Specialty Id: 0

Medical Specialty Name

Office Id: 0

The next check will ensure that there is a physician with that ID in the physician table. Otherwise, this referential integrity error will be displayed. Let's enter physician 2.

Checks	Error Id	Error Description
FK: PhysicianId	ForeingKeyNotFound	No matching 'Physician'
<code>error("Physician " + PhysicianId.ToString() +" already has 2 shifts assigned for the date: " + OfficeVisitDate.ToString(), PhysicianNotAvailable) if &shifts >= 2 and Insert;</code>	PhysicianNotAvailable	Physician... already has 2 shifts...
FK: ShiftId	ForeingKeyNotFound	No matching 'Shift'
FK: OfficeId	ForeingKeyNotFound	No matching 'Office'
CK: {OfficeVisitDate, ShiftId, OfficeId}		Office Visit Date, Shift Id, Office Id already exists
PK: {OfficeVisitDate, PhysicianId, ShiftId}	DuplicatePrimaryKey	Record already exists
FK: PatientId	ForeingKeyNotFound	No matching 'Patient'
<code>error("There is a pending office visit scheduled for this patient for the same specialty", PatientAlreadyScheduledForMedicalSpecialty) if not &isOk and Insert Level PatientId;</code>	PatientAlreadyScheduledForMedicalSpecialty	There is a pending office visit scheduled...
PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}	DuplicatePrimaryKey	Record already exists

The next thing to do is to invoke the procedure that counts the number of office visits that physician already has on that day and if it gives 2 or more, the error rule will be triggered, which shows this message on the screen.

Checks				
FK: PhysicianId				
Monday	Tuesday	Wednesday	Thursday	Friday
Physician: 1 Office: 2		Physician: 2 Office: 1	Physician: 1 Office: 2 Patient: 2	
Physician: 1 Office: 2				
	Physician: 1 Office: 2			
if not &IsOk and Insert Level PatientId;				
PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}				

Office Visit

apps5.genexus.com/ld9df182c70350001219afaceeb30e8e2/office...

GeneXus

Hospital Backoffice

by GeneXus

Recents Office Visits — Office Visit

Office Visit

Visit Date: 11/14/22

Physician Id: 1 Physician 1 already has 2 shifts assigned for the date: 11/14/22

Physician Name: Doctor 1

Shift Id:

Shift Name:

Medical Specialty Id: 1

Medical Specialty Name: Family Medicine

Office Id: 0

With physician 2 we did not get an error, but if we enter physician 1, who we know already has 2 shifts that day... we do see the error.

Checks	Error Id	Error Description
FK: PhysicianId	ForeingKeyNotFound	No matching 'Physician'
<code>error("Physician " + PhysicianId.ToString() + " already has 2 shifts assigned for the date: " + OfficeVisitDate.ToString(), PhysicianNotAvailable) if &shifts >= 2 and Insert;</code>	PhysicianNotAvailable	Physician... already has 2 shifts...
FK: ShiftId	ForeingKeyNotFound	No matching 'Shift'
FK: OfficeId	ForeingKeyNotFound	No matching 'Office'
CK: {OfficeVisitDate, ShiftId, OfficeId}		Office Visit Date, Shift Id, Office Id already exists
PK: {OfficeVisitDate, PhysicianId, ShiftId}	DuplicatePrimaryKey	Record already exists
FK: PatientId	ForeingKeyNotFound	No matching 'Patient'
<code>error("There is a pending office visit scheduled for this patient for the same specialty", PatientAlreadyScheduledForMedicalSpecialty) if not &isOk and Insert Level PatientId;</code>	PatientAlreadyScheduledForMedicalSpecialty	There is a pending office visit scheduled...
PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}	DuplicatePrimaryKey	Record already exists

This internal error code corresponds to the second parameter of the error rule. If we do not set it, the error ID will remain empty (which does not affect the operation at all).

Let's insert physician 2 again to continue.

Checks

FK: PhysicianId

```
error("Physician " + PhysicianId.ToString() + " already has 2 shifts assigned for the date: "
+ OfficeVisitDate.ToString(), PhysicianNotAvailable) if &shifts >= 2 and Insert;
```

FK: ShiftId

FK: Officeld

CK: {OfficeVisitDate, ShiftId, Officeld}

PK: {OfficeVisitDate, PhysicianId, ShiftId}

FK: PatientId

```
error("There is a pending office visit scheduled for this patient for the same specialty",
PatientAlreadyScheduledForMedicalSpecialty)
if not &IsOk and Insert
Level PatientId;
```

PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}

Office Visit

Recents Office Visits — Office Visit

Office Visit

Visit Date 11/14/22

Physician Id 2

Physician Name Doctor 2

Shift Id 25 No matching Shift

Shift Name

Medical Specialty Id 1

Medical Specialty Name Family Medicine

Office Id

The next step is to check referential integrity with ShiftId.

Checks

FK: PhysicianId

```
error("Physician " + PhysicianId.ToString() + " already has 2 shifts assigned for the date: "
+ OfficeVisitDate.ToString(), PhysicianNotAvailable) if &shifts >= 2 and Insert;
```

FK: ShiftId

FK: Officeld

CK: {OfficeVisitDate, ShiftId, Officeld}

PK: {OfficeVisitDate, PhysicianId, ShiftId}

FK: PatientId

```
error("There is a pending office visit scheduled for this patient for the same specialty",
PatientAlreadyScheduledForMedicalSpecialty)
if not &isOk and Insert
Level PatientId;
```

PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}

Office Visit

apps5.genexus.com/d9df182c70350001219afeaceeb30e8e2/office... GeneXus

Hospital Backoffice by GeneXus

Recents Office Visits — Office Visit

Office Visit

Visit Date	11/14/22
Physician Id	2
Physician Name	Doctor 2
Shift Id	1
Shift Name	Shift morning
Medical Specialty Id	1
Medical Specialty Name	Family Medicine
Office Id	

Now, let's assign Shift 1, Morning.

Checks

FK: PhysicianId

```
error("Physician " + PhysicianId.ToString() + " already has 2 shifts assigned for the date: "
+ OfficeVisitDate.ToString(), PhysicianNotAvailable) if &shifts >= 2 and Insert;
```

FK: ShiftId

FK: OfficeId

CK: {OfficeVisitDate, ShiftId, OfficeId}

PK: {OfficeVisitDate, PhysicianId, ShiftId}

FK: PatientId

```
error("There is a pending office visit scheduled for this patient for the same specialty",
PatientAlreadyScheduledForMedicalSpecialty)
if not &IsOk and Insert
Level PatientId;
```

PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}

Office Visit

apps5.genexus.com/id9df182c70350001219afeaceeb30e8e2/office...

GeneXus

Visit Date: 11/14/22

Physician Id: 2

Physician Name: Doctor 2

Shift Id: 1

Shift Name: Shift morning

Medical Specialty Id: 1

Medical Specialty Name: Family Medicine

Office Id: 25 No matching 'Office'.

Last Number: 0

Patient

Patient Id	Patient Name	Patient Number
		0

It will then check that the office exists. For example, we type 25.

Checks
FK: PhysicianId
<code>error("Physician " + PhysicianId.ToString() + " already has 2 shifts assigned for the date: " + OfficeVisitDate.ToString(), PhysicianNotAvailable) if &shifts >= 2 and Insert;</code>
FK: ShiftId
FK: OfficeId
CK: {OfficeVisitDate, ShiftId, OfficeId}
PK: {OfficeVisitDate, PhysicianId, ShiftId}
FK: PatientId
<code>error("There is a pending office visit scheduled for this patient for the same specialty", PatientAlreadyScheduledForMedicalSpecialty) if not &IsOk and Insert Level PatientId;</code>
PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}

Office Visit

apps5.genexus.com/id9df182c70350001219afaceeb30e8e2/office...

GeneXus

Visit Date: 11/14/22

Physician Id: 2

Physician Name: Doctor 2

Shift Id: 1

Shift Name: Shift morning

Medical Specialty Id: 1

Medical Specialty Name: Family Medicine

Office Id: 3

Last Number: 0

Patient

Patient Id	Patient Name	Patient Number
		0

And now this one, which does exist.

Checks
FK: PhysicianId
<pre>error("Physician " + PhysicianId.ToString() + " already has 2 shifts assigned for the date: " + OfficeVisitDate.ToString(), PhysicianNotAvailable) if &shifts >= 2 and Insert;</pre>
FK: ShiftId
FK: OfficeId
CK: {OfficeVisitDate, ShiftId, OfficeId}
PK: {OfficeVisitDate, PhysicianId, ShiftId}
FK: PatientId
<pre>error("There is a pending office visit scheduled for this patient for the same specialty", PatientAlreadyScheduledForMedicalSpecialty) if not &isOk and Insert Level PatientId;</pre>
PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}

Office Visit

Recents Office Visits — Office Visit

Office Visit

Visit Date: 11/14/22 Office Visit Date, Shift Id, Office Id already exists

Physician Id: 2

Physician Name: Doctor 2

Shift Id: 1

Shift Name: Shift morning

Medical Specialty Id: 1

Medical Specialty Name: Family Medicine

Office Id: 2

When leaving the OfficeId field, it will already be possible to check the candidate key. In this case, we had entered office 3, but let's see what would happen if we entered office 2, which we know is already in use (by physician 1). This error message is displayed. To check it, the unique index is used.

Again, let's enter office 3, which we know is not being used.

Checks

FK: PhysicianId

```
error("Physician " + PhysicianId.ToString() + " already has 2 shifts assigned for the date: "
+ OfficeVisitDate.ToString(), PhysicianNotAvailable) if &shifts >= 2 and Insert;
```

FK: ShiftId

FK: Officeld

CK: {OfficeVisitDate, ShiftId, Officeld}

PK: {OfficeVisitDate, PhysicianId, ShiftId}

FK: PatientId

```
error("There is a pending office visit scheduled for this patient for the same specialty",
PatientAlreadyScheduledForMedicalSpecialty)
if not &isOk and Insert
Level PatientId;
```

PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}

Office Visit

apps5.genexus.com/d9df182c70350001219afeaceeb30e8e2/office...

GeneXus

Visit Date: 11/14/22

Physician Id: 2

Physician Name: Doctor 2

Shift Id: 1

Shift Name: Shift morning

Medical Specialty Id: 1

Medical Specialty Name: Family Medicine

Office Id: 3

Last Number: 0

Patient

Patient Id	Patient Name	Patient Number
		0

At this point, the uniqueness of the record we are trying to insert is checked. In this case, it did not find that the record already existed, so we do not see anything.

Checks				
FK: PhysicianId				
Monday	Tuesday	Wednesday	Thursday	Friday
Physician: 1 Office: 2		Physician: 2 Office: 1	Physician: 1 Office: 2 Patient: 2	
Physician: 1 Office: 2				
	Physician: 1 Office: 2			
if not &IsOk and Insert Level PatientId;				
PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}				

Office Visit

apps5.genexus.com/d9df182c70350001219afeaceeb30e8e2/office...

GeneXus

Visit Date: 11/14/22

Physician Id: 2

Physician Name: Doctor 2

Shift Id: 1

Shift Name: Shift morning

Medical Specialty Id: 1

Medical Specialty Name: Family Medicine

Office Id: 3

Last Number: 0

Patient

Patient Id	Patient Name	Patient Number
		0

If we had tried to enter a record that already existed...for example, note that for this physician and this shift, we already have an office visit scheduled for Wednesday, so let's see what would happen if we change the date for the new office visit that we want to assign, and we place this one for Wednesday.

Checks
FK: PhysicianId
<pre>error("Physician " + PhysicianId.ToString() + " already has 2 shifts assigned for the date: " + OfficeVisitDate.ToString(), PhysicianNotAvailable) if &shifts >= 2 and Insert;</pre>
FK: ShiftId
FK: Officeld
CK: {OfficeVisitDate, ShiftId, Officeld}
PK: {OfficeVisitDate, PhysicianId, ShiftId}
FK: PatientId
<pre>error("There is a pending office visit scheduled for this patient for the same specialty", PatientAlreadyScheduledForMedicalSpecialty) if not &isOk and Insert Level PatientId;</pre>
PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}

Office Visit

Recents Office Visits — Office Visit

Office Visit

- Record already exists

Visit Date: 11/16/22

Physician Id: 2

Physician Name: Doctor 2

Shift Id: 1

Shift Name: Shift morning

Medical Specialty Id: 1

Medical Specialty Name: Family Medicine

Office Id: 3

...because it only checks the Primary Key on the server, when it tries the insert

It is not showing it interactively, but if we click on confirm the message "Record already exists" will be displayed.

Let's place Monday's date again.

Checks

FK: PhysicianId

```
error("Physician " + PhysicianId.ToString() + " already has 2 shifts assigned for the date: "
+ OfficeVisitDate.ToString(), PhysicianNotAvailable) if &shifts >= 2 and Insert;
```

FK: ShiftId

FK: Officeld

CK: {OfficeVisitDate, ShiftId, Officeld}

PK: {OfficeVisitDate, PhysicianId, ShiftId}

FK: PatientId

```
error("There is a pending office visit scheduled for this patient for the same specialty",
PatientAlreadyScheduledForMedicalSpecialty)
if not &isOk and Insert
Level PatientId;
```

PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}

Office Visit

Physician Name: Doctor 2

Shift Id: 1

Shift Name: Shift morning

Medical Specialty Id: 1

Medical Specialty Name: Family Medicine

Office Id: 3

Last Number: 0

Patient Id	Patient Name	Patient Number
25	No matching 'Patient'	1
		0
0		0

This is the end of the first level checks.

Now we check each line of the second level. If we enter a nonexistent patient ID, the referential integrity will fail. Let's enter an existing one, number 1.

Checks

FK: PhysicianId

```
error("Physician " + PhysicianId.ToString() + " already has 2 shifts assigned for the date: "
+ OfficeVisitDate.ToString(), PhysicianNotAvailable) if &shifts >= 2 and Insert;
```

FK: ShiftId

FK: Officeld

CK: {OfficeVisitDate, ShiftId, Officeld}

PK: {OfficeVisitDate, PhysicianId, ShiftId}

FK: PatientId

```
error("There is a pending office visit scheduled for this patient for the same specialty",
PatientAlreadyScheduledForMedicalSpecialty)
if not &IsOk and Insert
Level PatientId;
```

PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}

Office Visit

Physician Name: Doctor 2

Shift Id: 1

Shift Name: Shift morning

Medical Specialty Id: 1

Medical Specialty Name: Family Medicine

Office Id: 3

Last Number: 1

Patient

Patient Id	Patient Name	Patient Number
1	Patient1	1
2		0

There is a pending office visit scheduled for this patient for the same specialty

The next thing that will be checked is the error. Let's try with patient 2. The error rule we had declared is being triggered. Why?

Because patient 2 was already scheduled for Thursday's office visit, with physician 1, of the same specialty as physician 2 (Family Medicine). Therefore, we cannot schedule this patient.

Checks

FK: PhysicianId

```
error("Physician " + PhysicianId.ToString() + " already has 2 shifts assigned for the date: "
+ OfficeVisitDate.ToString(), PhysicianNotAvailable) if &shifts >= 2 and Insert;
```

FK: ShiftId

FK: Officeld

CK: {OfficeVisitDate, ShiftId, Officeld}

PK: {OfficeVisitDate, PhysicianId, ShiftId}

FK: PatientId

```
error("There is a pending office visit scheduled for this patient for the same specialty",
PatientAlreadyScheduledForMedicalSpecialty)
if not &isOk and Insert
Level PatientId;
```

PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}

Office Visit

Physician Name: Doctor 2

Shift Id: 1

Shift Name: Shift morning

Medical Specialty Id: 1

Medical Specialty Name: Family Medicine


Office Id: 3

Last Number: 1

Patient Id	Patient Name	Patient Number
1	Patient1	1
1	Patient already exists	0
		0

Then, the uniqueness of the primary key of this second level will be checked. Since no other patients are entered, it will not fail. But let's try another line. Let's enter number 1 again. Here we see that the uniqueness is indeed being checked.

If we now press Confirm, we will see that the office visit has been entered.

	Monday	Tuesday	Wednesday	Thursday	Friday
Shift 1 morning	<div style="border: 1px solid red; padding: 2px; margin-bottom: 5px;">Physician: 1 Office: 2</div> <div style="border: 1px dashed red; padding: 2px; margin-bottom: 5px;">Physician: 2 Office: 3</div> <div style="border: 1px dashed red; padding: 2px;">Patient: 1</div> 		<div style="border: 1px solid red; padding: 2px;">Physician: 2 Office: 1</div>	<div style="border: 1px solid red; padding: 2px; margin-bottom: 5px;">Physician: 1 Office: 2</div> <div style="border: 1px solid red; padding: 2px;">Patient: 2</div>	
Shift 2 intermediate	<div style="border: 1px solid red; padding: 2px;">Physician: 1 Office: 2</div>				
Shift 3 evening		<div style="border: 1px solid red; padding: 2px;">Physician: 1 Office: 2</div>			

Now let's try to repeat the same thing, but inserting through the Business Component. So let's delete the office visit we have just entered and try again.

Web Layout Rules Events Conditions Variables Help Docum

<No action group selected>

MainTable

Schedule Date	&ScheduleDate	Insert Office Visit
Physician Id	&PhysicianId	
Shift Id	&ShiftId	
Office Id	&OfficeId	
Patient Id	&patientId	

```

Event 'Insert Office Visit'
  &officeVisit = new()
  &officeVisit.OfficeVisitDate = &ScheduleDate
  &officeVisit.PhysicianId = &PhysicianId
  &officeVisit.ShiftId = &ShiftId
  &officeVisit.OfficeId = &OfficeId
  If not &patientId.IsEmpty()
    &officeVisitPatient = new()
    &officeVisitPatient.PatientId = &patientId
    &officeVisit.Patient.Add(&officeVisitPatient)
  endif
  if &officeVisit.Insert()
    Commit
  endif
  &messages = &officeVisit.GetMessages()
Endevent

```

Type Definition

Based on	(none)
Data Type	OfficeVisit.Patient
Collection	OfficeVisit OfficeVisit.Patient External Objects
Initial value	
Validation	
Value range	

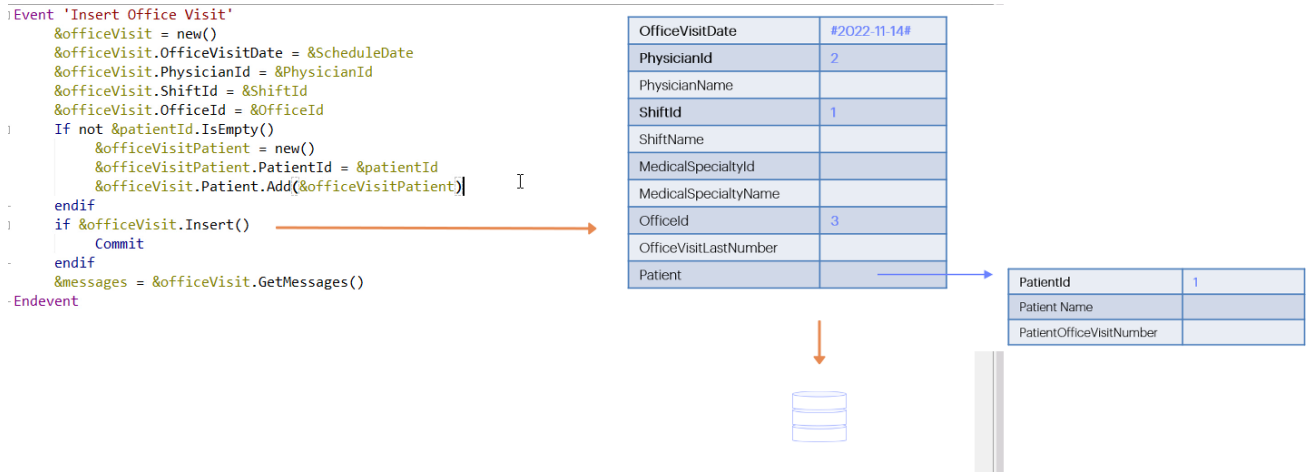
To make it really easy we have created a web panel with 5 input variables: for the 4 relevant attributes of the office visit header, and for entering a patient. First we will try to insert through the BC. For this, of course, we have enabled the Business Component property of the transaction.

From there we were able to create this variable of the Business Component data type, through which we will insert data.

As a good practice, we first ask for new memory space for the variable, to make sure it is clean.

We assign values to the 4 elements of the header, from the variables on the screen.

Next, if the field on the screen for Patient ID was not left empty, then we will try to add a patient. We have this variable of the data type of each line; we clean it with new, assign a value only to the PatientId element, and add it to the patient collection of the Business Component.



In this way, we have loaded the structure of the &officeVisit variable. Now we have to give the order to try the insertion...

```

)Event 'Insert Office Visit'
  &officeVisit = new()
  &officeVisit.OfficeVisitDate = &ScheduleDate
  &officeVisit.PhysicianId = &PhysicianId
  &officeVisit.ShiftId = &ShiftId
  &officeVisit.OfficeId = &OfficeId
  If not &patientId.IsEmpty()
    &officeVisitPatient = new()
    &officeVisitPatient.PatientId = &patientId
    &officeVisit.Patient.Add(&officeVisitPatient)
  endif
  if &officeVisit.Insert()
    Commit
  endif
  &messages = &officeVisit.GetMessages()
-Endevent

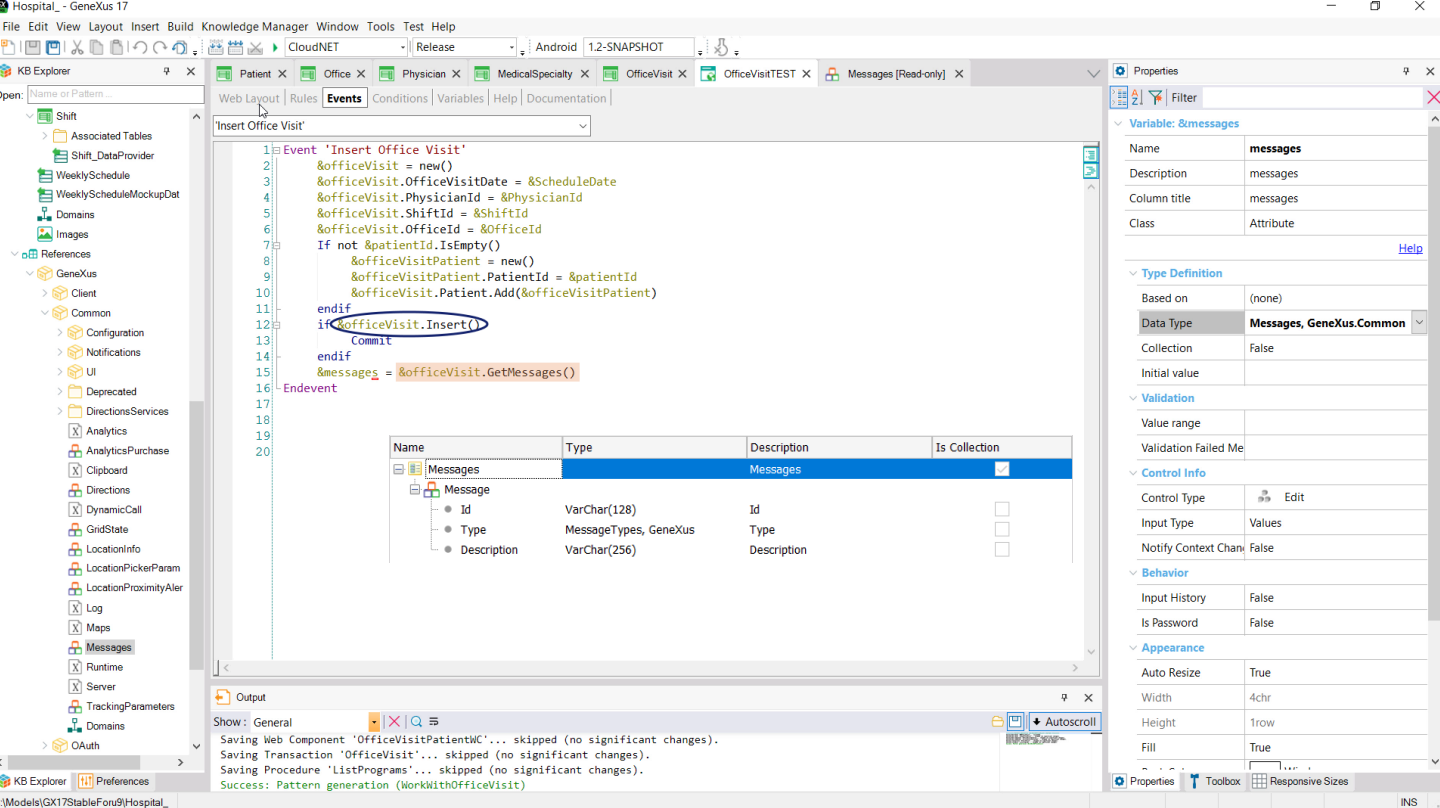
```

I

OfficeVisitDate	#2022-11-14#
PhysicianId	2
PhysicianName	Doctor 2
ShiftId	1
ShiftName	Shift morning
MedicalSpecialtyId	1
MedicalSpecialtyName	Family Medicine
OfficeId	3
OfficeVisitLastNumber	1
Patient	

PatientId	1
Patient Name	Patient 1
PatientOfficeVisitNumber	1

And if it is successful, commit.



In the &messages variable of the SDT data type that comes predefined in the GeneXus module, we get all the messages that have resulted from running the Insert method. And we insert it on the screen to view all those messages.

Then let's run to repeat everything we did through the transaction, but now in this way.

Hospital Backoffice



Recents Office Visit TEST

Schedule Date: 11/14/22 29 Insert Office Visit

Physician Id: 25

Shift Id: 25

Office Id: 25

Patient Id: 25

Id	Type	Description
ForeignKeyNotFound	Error	No matching 'Physician'.
ForeignKeyNotFound	Error	No matching 'Medical Specialty'.
ForeignKeyNotFound	Error	No matching 'Shift'.
ForeignKeyNotFound	Error	No matching 'Office'.

Let's choose Monday, the date we wanted. Also, let's place a nonexistent physician, a nonexistent shift, a nonexistent office, and even a nonexistent patient to see all the messages that will be displayed when we try to insert data.

We see all the referential integrity errors. Patient didn't even get there because that would only happen if the header could be inserted.

Office Visits Office Visit TEST

apps5.genexus.com/ld9df182c70350001219afeaceeb30e8e2/officevisittest.aspx

GeneXus

Hospital Backoffice

Recents Office Visit TEST

Schedule Date: 11/14/22 29

Physician Id: 1

Shift Id: 3

Office Id: 3

Patient Id:

	Monday	Tuesday	Wednesday	Thursday	Friday
Physician: 1 Office: 2			Physician: 2 Office: 1	Physician: 1 Office: 2 Patient: 2	
Physician: 1 Office: 2					
		Physician: 1 Office: 2			

Id	Type	Checks	Error Id	Error Description
ForeignKeyNotFound	Error	FK: PhysicianId	ForeignKeyNotFound	No matching 'Physician'
ForeignKeyNotFound	Error	error("Physician " + PhysicianId.ToString() + " already has 2 shifts assigned for the date: " + OfficeVisitDate.ToString(), PhysicianNotAvailable) if &shifts >= 2 and Insert;	PhysicianNotAvailable	Physician... already has 2 shifts...
ForeignKeyNotFound	Error	FK: ShiftId	ForeignKeyNotFound	No matching 'Shift'
ForeignKeyNotFound	Error	FK: OfficeId	ForeignKeyNotFound	No matching 'Office'
		CK: {OfficeVisitDate, ShiftId, OfficeId}		Office Visit Date, Shift Id, Office Id already exists
		PK: {OfficeVisitDate, PhysicianId, ShiftId}	DuplicatePrimaryKey	Record already exists
		FK: PatientId	ForeignKeyNotFound	No matching 'Patient'
		error("There is a pending office visit scheduled for this patient for the same specialty", PatientAlreadyScheduledForMedicalSpecialty) if not &isOk and Insert	PatientAlreadySchedule dForMedicalSpecialty	There is a pending office visit scheduled...

Now let's place physician 1, on shift 3, in office 3, with no patient. We know that it must fail because of the error rule.

Hospital Backoffice



Recents Office Visit TEST

Schedule Date: 11/14/22

Physician Id:

Shift Id:

Office Id:

Patient Id:

Id	Type	Description
PhysicianNotAvailable	Error	Physician 1 already has 2 shifts assigned for the date: 11/14/22

Checks	Error Id	Error Description
<pre>error("Physician " + PhysicianId.ToString() + " already has 2 shifts assigned for the date: " + OfficeVisitDate.ToString(), PhysicianNotAvailable) if &shifts >= 2 and Insert;</pre>	PhysicianNotAvailable	Physician... already has 2 shifts...

Note that the error message ID is the one we placed in the rule.

Hospital Backoffice



Recents Office Visit TEST

Schedule Date: 11/14/22 29 Insert Office Visit

Physician Id: 1

Shift Id: 25

Office Id: 3

Patient Id:

Id	Type	Description
PhysicianNotAvailable	Error	Physician 1 already has 2 shifts assigned for the date: 11/14/22
ForeignKeyNotFound	Error	No matching 'Shift'.

We can break more checks, and we will see it in the message collection. For example, let's enter the shift 25, which does not exist, and the two error messages are displayed.

Office Visits Office Visit TEST

apps5.genexus.com/ld9df182c70350001219afeaceeb30e8e2/officevisittest.aspx

GeneXus

Hospital Backoffice

Recents Office Visit TEST

Schedule Date: 11/16/22 29

Physician Id: 2

Shift Id: 1

Office Id: 3

Patient Id:

	Monday	Tuesday	Wednesday	Thursday	Friday
Physician: 1 Office: 2			Physician: 2 Office: 1	Physician: 1 Office: 2 Patient: 2	
Physician: 1 Office: 2					
		Physician: 1 Office: 2			

Id	Type	Description
PhysicianNotAvailable	Error	Physician 1 already has 2 shifts assigned for the date: 11/14/22
ForeignKeyNotFound	Error	No matching 'Shift'.

Now let's try the primary key uniqueness check. Let's try inserting a record for Wednesday, physician 2, shift 1, office 3.

Hospital Backoffice



Recents Office Visit TEST

Schedule Date: 11/16/22

Physician Id:

Shift Id:

Office Id:

Patient Id:

Id	Type	Description
DuplicatePrimaryKey	Error	Record already exists

Good, it was what we expected.

Office Visits Office Visit TEST

apps5.genexus.com/Id9df182c70350001219afeaceeb30e8e2/officevisittest.aspx

GeneXus

Hospital Backoffice

Recents Office Visit TEST

Schedule Date: 11/14/22 29

Physician Id: 2

Shift Id: 1

Office Id: 2

Patient Id:

Monday	Tuesday	Wednesday	Thursday	Friday
Physician: 1 Office: 2		Physician: 2 Office: 1	Physician: 1 Office: 2 Patient: 2	
Physician: 1 Office: 2				
	Physician: 1 Office: 2			

Id	Type	Description
DuplicatePrimaryKey	Error	Record already exists

Now let's try inserting the office visit for Monday on shift 1, but for an office that is already being used, that is, 2.

Checks	Error Id	Error Description
FK: PhysicianId	ForeingKeyNotFound	No matching 'Physician'
error("Physician " + PhysicianId.ToString() + " already has 2 shifts assigned for the date: " + OfficeVisitDate.ToString(), PhysicianNotAvailable) if &shifts >= 2 and Insert;	PhysicianNotAvailable	Physician... already has 2 shifts...
FK: ShiftId	ForeingKeyNotFound	No matching 'Shift'
FK: OfficeId	ForeingKeyNotFound	No matching 'Office'
CK: {OfficeVisitDate, ShiftId, OfficeId}		Office Visit Date, Shift Id, Office Id already exists
PK: {OfficeVisitDate, PhysicianId, ShiftId}	DuplicatePrimaryKey	Record already exists
FK: PatientId	ForeingKeyNotFound	No matching 'Patient'
error("There is a pending office visit scheduled for this patient for the same specialty", PatientAlreadyScheduledForMedicalSpecialty) if not &isOk and Insert Level PatientId;	PatientAlreadySchedule dForMedicalSpecialty	There is a pending office visit scheduled...
PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}	DuplicatePrimaryKey	Record already exists

Id	Type	Description
	Error	Office Visit Date,Shift Id,Office Id already exists

Note that the uniqueness of the candidate key has been checked.

Hospital Backoffice

Recents Office Visit TEST

Schedule Date: 11/14/22

Physician Id:

Shift Id:

Office Id:

Patient Id:

Id	Type	Description
SuccessfullyAdded	Warning	Data has been successfully added.

Now, let's insert the correct office. If we leave Patient ID empty, the header will be inserted successfully. The message is of Warning type and it is the one displayed in the transaction when the pattern is not applied to it; that is, when it does not return to the caller object.

Office Visits x Office Visit TEST x + Checks Error Id Error Description

apps5.genexus.com/ld9df182c70350001219afeaceeb30e88

GeneXus

Hospital Backoffice

Recents Office Visit TEST

Schedule Date 11/14/22 29

Physician Id 2

Shift Id 1

Office Id 3

Patient Id 25

FK: PhysicianId	ForeignKeyNotFound	No matching 'Physician'
error("Physician " + PhysicianId.ToString() + " already has 2 shifts assigned for the date: " + OfficeVisitDate.ToString(), PhysicianNotAvailable) if &shifts >= 2 and Insert;		
FK: ShiftId	ForeignKeyNotFound	No matching 'Shift'
FK: OfficeId	ForeignKeyNotFound	No matching 'Office'
CK: {OfficeVisitDate, ShiftId, OfficeId}		Office Visit Date, Shift Id, Office Id already exists
PK: {OfficeVisitDate, PhysicianId, ShiftId}	DuplicatePrimaryKey	Record already exists
FK: PatientId	ForeignKeyNotFound	No matching 'Patient'
error("There is a pending office visit scheduled for this patient for the same specialty", PatientAlreadyScheduledForMedicalSpecialty) if not &isok and Insert Level PatientId;		
PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}	DuplicatePrimaryKey	Record already exists

Id	Type	Description
ForeignKeyNotFound	Error	No matching 'Patient'.

Let's delete the record and try again, this time with a patient. We enter a nonexistent patient and the referential integrity error is displayed. So the header is not entered.

Office Visits x Office Visit TEST x +

apps5.genexus.com/ld9df182c70350001219afeaceeb30e8

GeneXus

Hospital Backoffice

Recents Office Visit TEST

Schedule Date 11/14/22 29

Physician Id 2

Shift Id 1

Office Id 3

Patient Id 2

Checks	Error Id	Error Description
FK: PhysicianId	ForeingKeyNotFound	No matching 'Physician'
error("Physician " + PhysicianId.ToString() + " already has 2 shifts assigned for the date: " + OfficeVisitDate.ToString(), PhysicianNotAvailable) if &shifts >= 2 and Insert;	PhysicianNotAvailable	Physician... already has 2 shifts...
FK: ShiftId	ForeingKeyNotFound	No matching 'Shift'
FK: OfficeId	ForeingKeyNotFound	No matching 'Office'
CK: {OfficeVisitDate, ShiftId, OfficeId}		Office Visit Date, Shift Id, Office Id already exists
PK: {OfficeVisitDate, PhysicianId, ShiftId}	DuplicatePrimaryKey	Record already exists
FK: PatientId	ForeingKeyNotFound	No matching 'Patient'
error("There is a pending office visit scheduled for this patient for the same specialty", PatientAlreadyScheduledForMedicalSpecialty) if not &isok and Insert Level PatientId;	PatientAlreadySchedule dForMedicalSpecialty	There is a pending office visit scheduled...
PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}	DuplicatePrimaryKey	Record already exists

Id	Type	Description
PatientAlreadyScheduledForMedicalSpecialty	Error	There is a pending office visit scheduled for this patient for the same special

If we now try to enter patient 2, who already has a pending office visit for the same specialty... the error is also thrown and it does not save either.

Hospital Backoffice



Recents Office Visit TEST

Schedule Date: 11/14/22 29 Insert Office Visit

Physician Id: 2

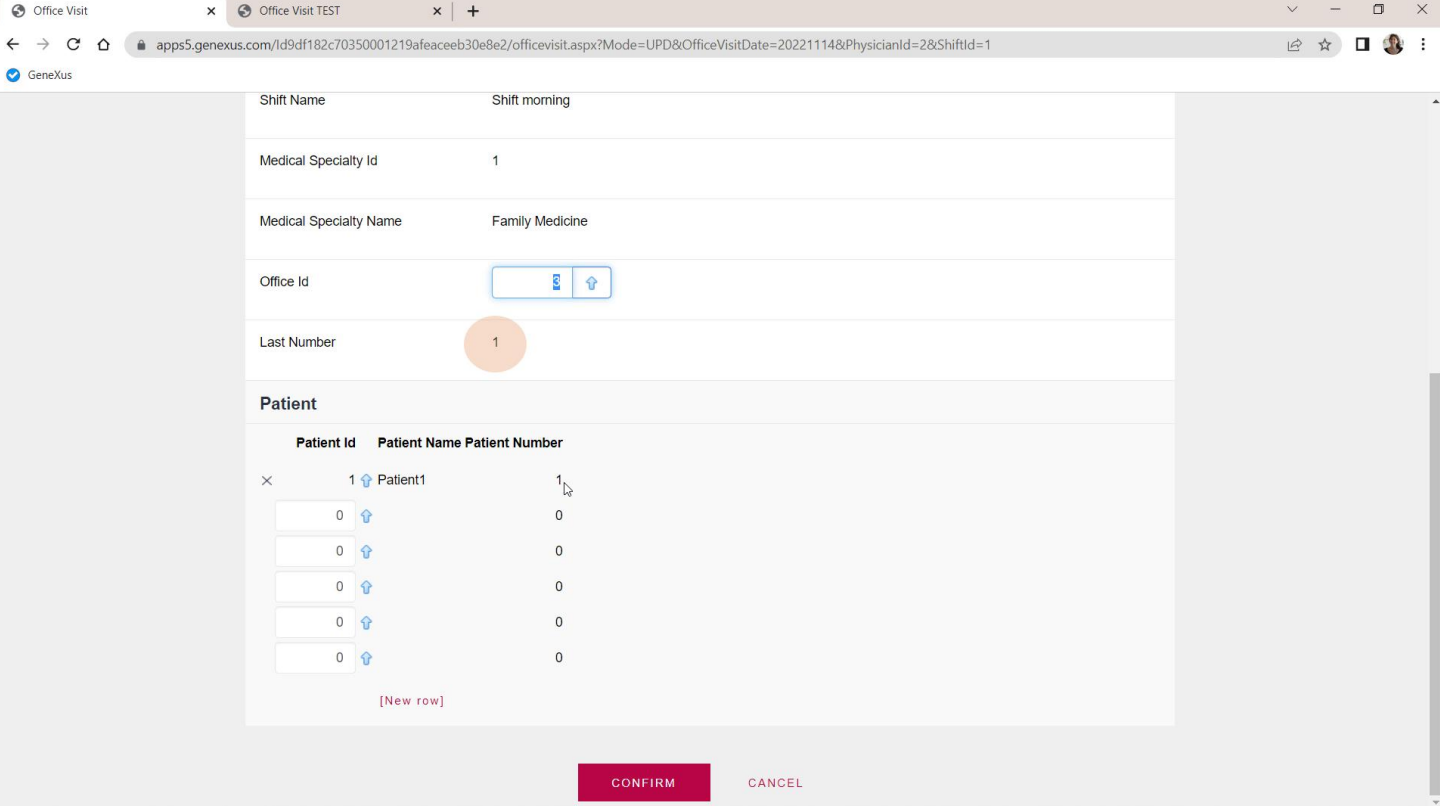
Shift Id: 1

Office Id: 3

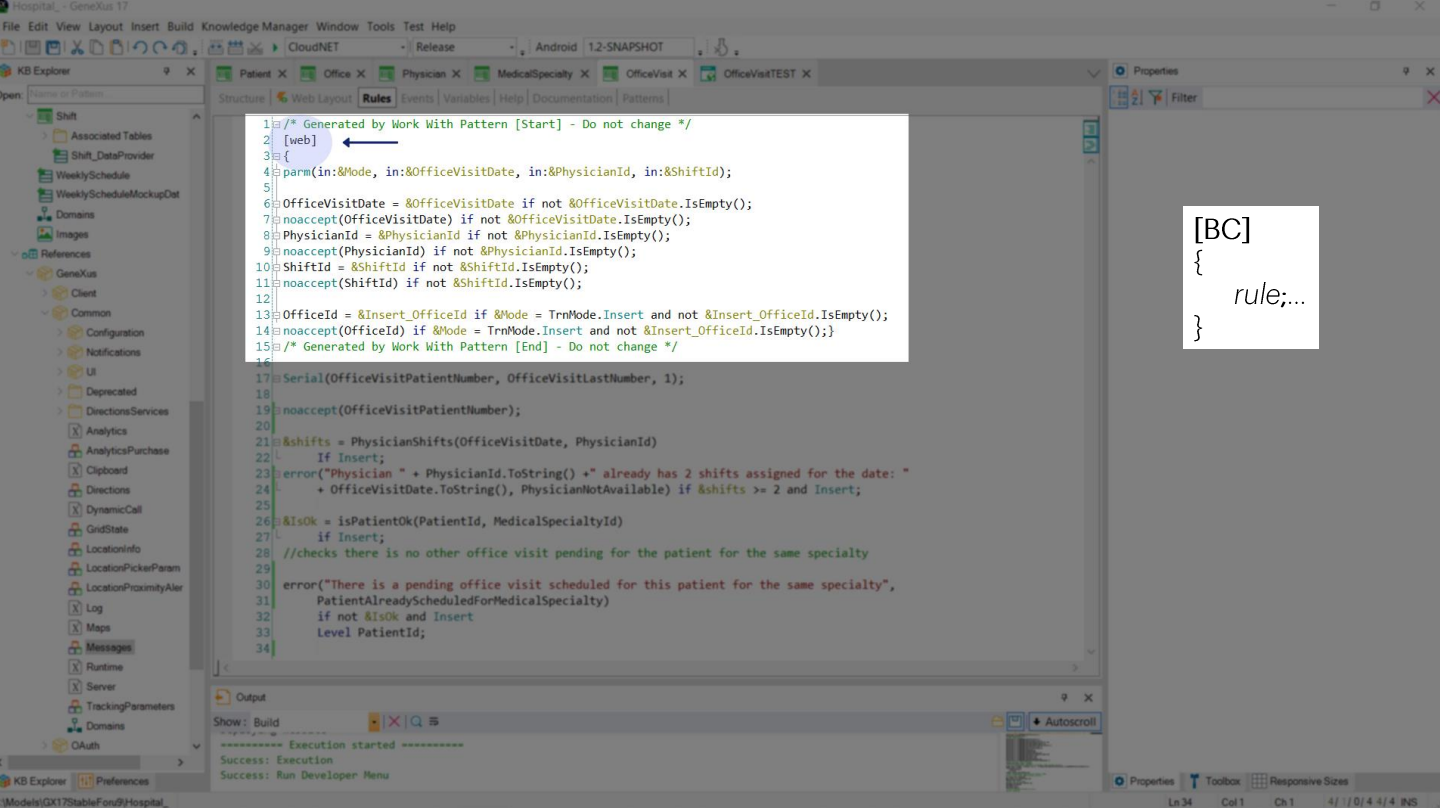
Patient Id: 1

Id	Type	Description
SuccessfullyAdded	Warning	Data has been successfully added.

Now, let's enter patient 1. Success!



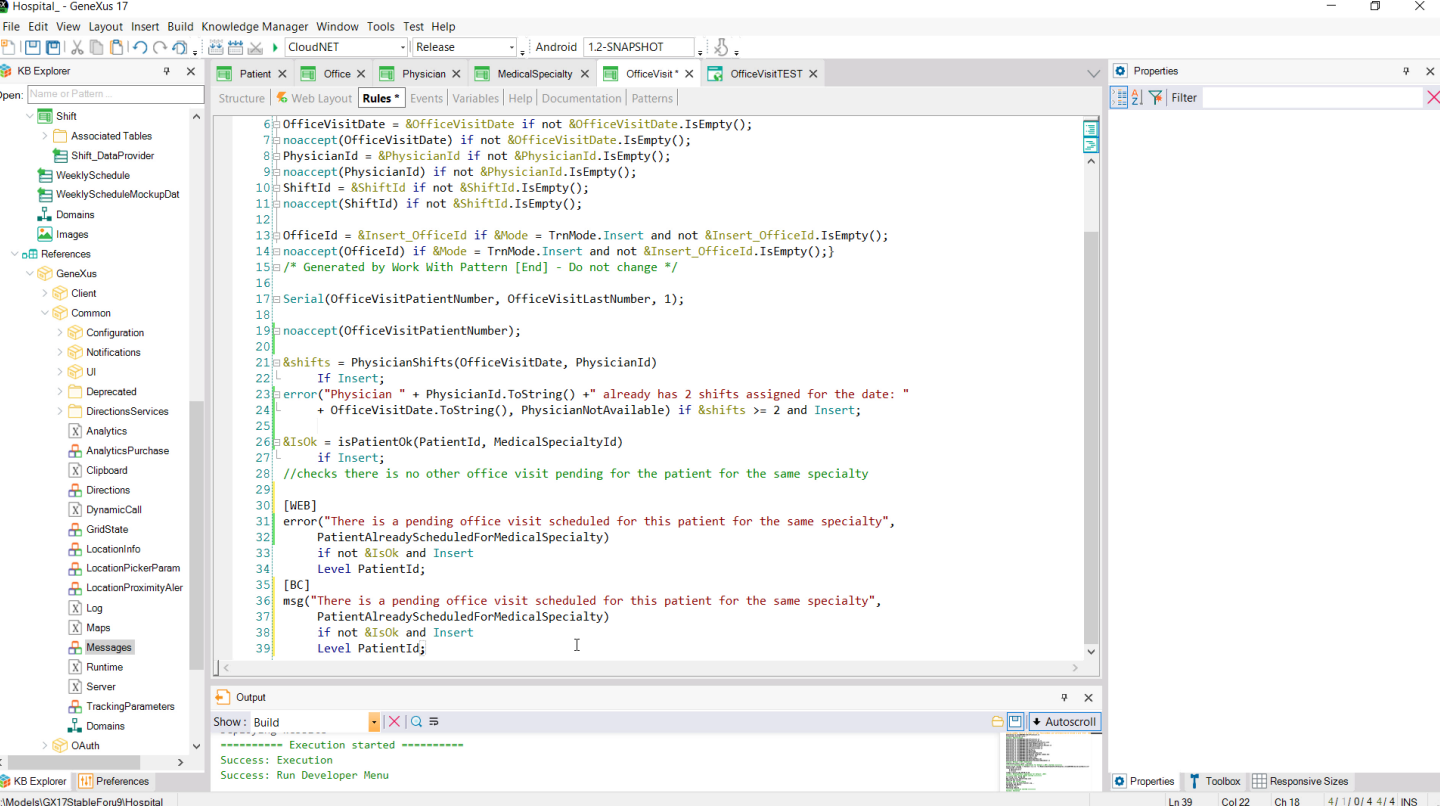
If we look at how this office visit was saved, it looks exactly the same as when we inserted it through the transaction. Note that the serial rule was also triggered, giving the number 1 to the patient and updating the attribute of the last number given.



Of all the transaction rules, which ones were triggered when inserting through the Business Component?

As we already know, all those that do not depend on the interface and do not make sense in this context are fired, such as the parm rule, logically, because the BC is not an object that can be invoked like the transaction, but a special data type, with special methods.

In this case, the parm rule was not only not triggered because it does not make sense, but also because the entire block of rules added by the pattern were qualified with this, which is known as environment attribute. It indicates that these rules are only included when the Web transaction is being executed, and not when it is executed as a Business component. If, on the other hand, we wanted some rule to be executed only in the context of the Business Component, then we can qualify it like this. The curly braces are needed if several rules are qualified and not just one.



For example, let's suppose that when inserting through the Business Component, we do not want to prevent a patient with a pending office visit from being entered. We only want to throw a message in that case. Then we qualify the error rule to be executed only in the Web environment, and add a message rule for the case of the Business Component.

Hospital Backoffice



Recents Office Visits — Office Visit TEST

Schedule Date: 11/21/22 29 Insert Office Visit

Physician Id: 1

Shift Id: 1

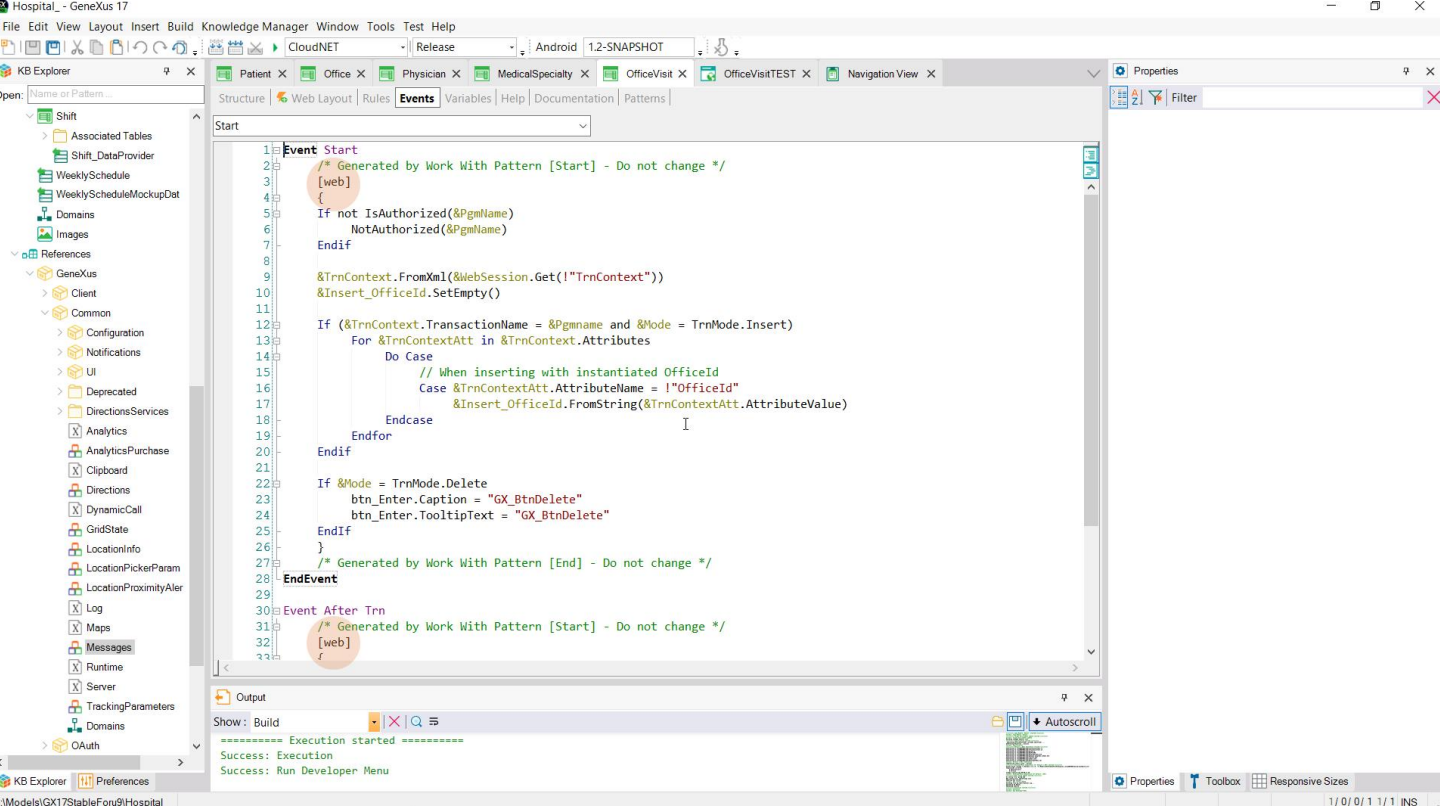
Office Id: 1

Patient Id: 1

Id	Type	Description
PatientAlreadyScheduledForMedicalSpecialty	Warning	There is a pending office visit scheduled for this patient for the same specialt
SuccessfullyAdded	Warning	Data has been successfully added.

Let's try it. If we try through the transaction, an error is displayed as we expected. If, on the other hand, we try through the BC, the message type is a Warning and it is inserted without any problem.

Let's leave it as it was.



Events can also be qualified.

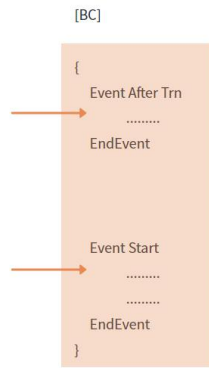
Of all the events defined in a transaction, which are executed when the business component is used? Only the Start and After Trn. If these events include references to objects with a user interface, these references are ignored.

Coincidentally, these are the two events that the pattern added here. But let's note that they are qualified as such; that is, they will only be executed in the context of the Web transaction and not when the Business Component is executed.

And you can also specify you want to execute both events only when the Transaction is executed as Business Component, like this example shows:

```
[BC]
{
  Event After Trn
  .....
  EndEvent

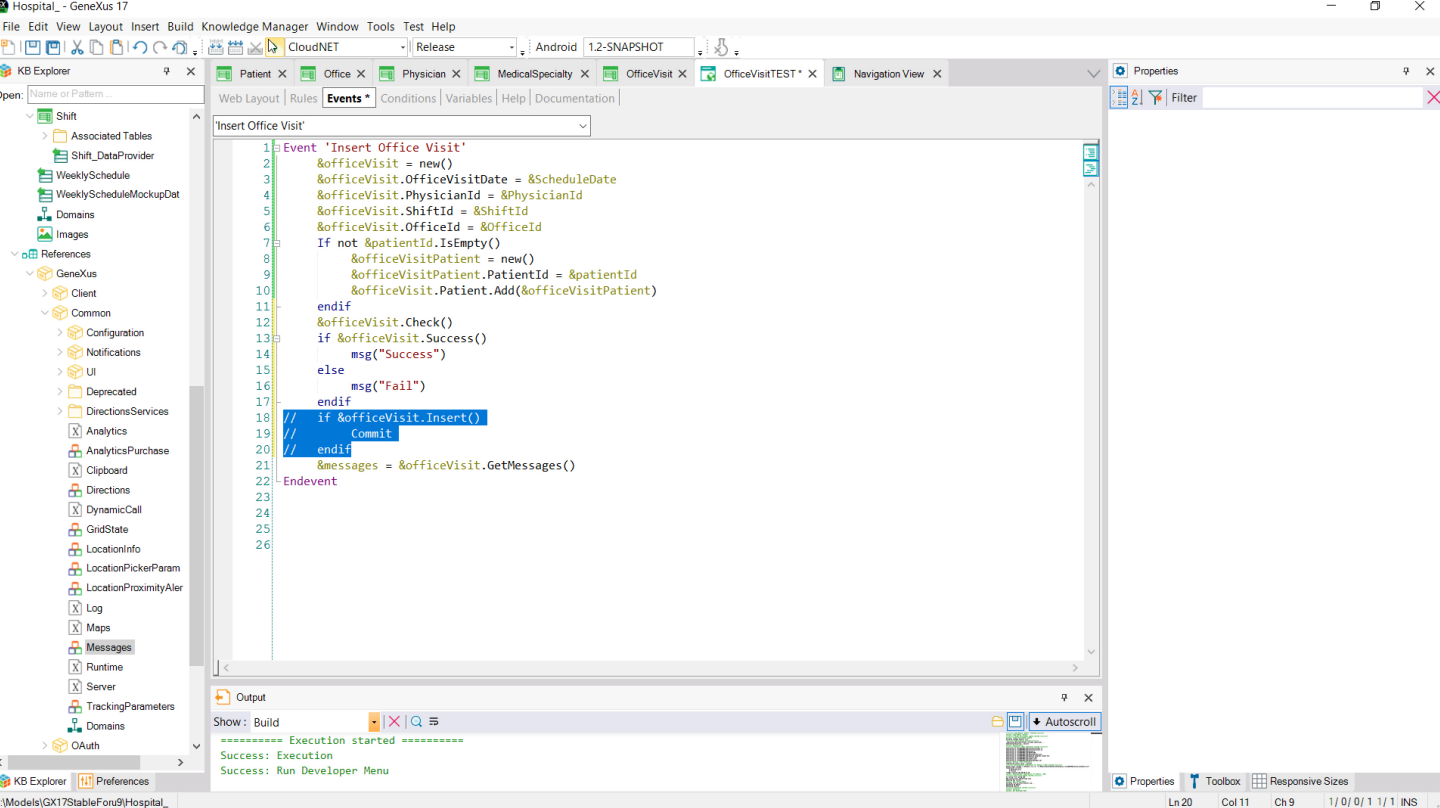
  Event Start
  .....
  .....
  EndEvent
}
```



Likewise, you have the qualifier [WEB] to indicate that one or both events must be executed only if the Transaction is running in web environment with its web form, and not when it is executed as Business Component.

If we wanted to define other options for those events that are valid only for the Business Component, then we would write them between BC square brackets.

Either by encompassing them, or within each one, distinguishing what is web from what is BC.



Let's go back to the insertion with BC. We may want to do the same validations that the Insert will do, but without updating the database. For this we have the Check method.

We invoke it and then check the result of the operation through the Success method, which will give true if it was successful and false otherwise. Let's stop here for a moment: the result of applying this method to a BC will refer to the result of the last operation performed on the BC. Here it is Check, but it could have been Load, or Delete, or Save, or even Insert or Update or InsertOrUpdate. In particular, these last methods already return their result, so we do not need the Success method to know it. Therefore, we perform the operation and ask for its result in the same statement.

Although it is not essential, we also have the opposite method, Fail.

A message will inform the user whether the check was successful or not. To test it, let's comment the database update. We run it.

Hospital Backoffice by GeneXus

Recents Office Visit TEST

Fail

Schedule Date: 11/22/22 29 Insert Office Visit

Physician Id: 25

Shift Id: 25

Office Id: 25

Patient Id: 0

Id	Type	Description
ForeignKeyNotFound	Error	No matching 'Physician'.
ForeignKeyNotFound	Error	No matching 'Medical Specialty'.
ForeignKeyNotFound	Error	No matching 'Shift'.
ForeignKeyNotFound	Error	No matching 'Office'.

The last office visit date scheduled is this one. Let's try to check an entry for the next day. We enter a nonexistent physician, nonexistent shift, nonexistent office. The Error and Fail messages are displayed.

Hospital Backoffice



Recents Office Visit TEST

Fail

Schedule Date: 11/22/22

Physician Id:

Shift Id:

Office Id:

Patient Id:

Insert Office Visit

Id	Type	Description
ForeignKeyNotFound	Error	No matching 'Patient'.

Now let's enter everything correctly for the first level and a nonexistent patient. The Error and Fail messages are displayed.

Hospital Backoffice



Recents Office Visit TEST

Fail

Schedule Date 11/22/22 29

Insert Office Visit

Physician Id 1

Shift Id 1

Office Id 1

Patient Id 1

Id	Type	Description
PatientAlreadyScheduledForMedicalSpecialty	Error	There is a pending office visit scheduled for this patient for the same specialt

With patient 1, we know that the error rule has to be triggered. OK.

Hospital Backoffice

Recents Office Visit TEST

Success

Schedule Date: 11/22/22

Physician Id:

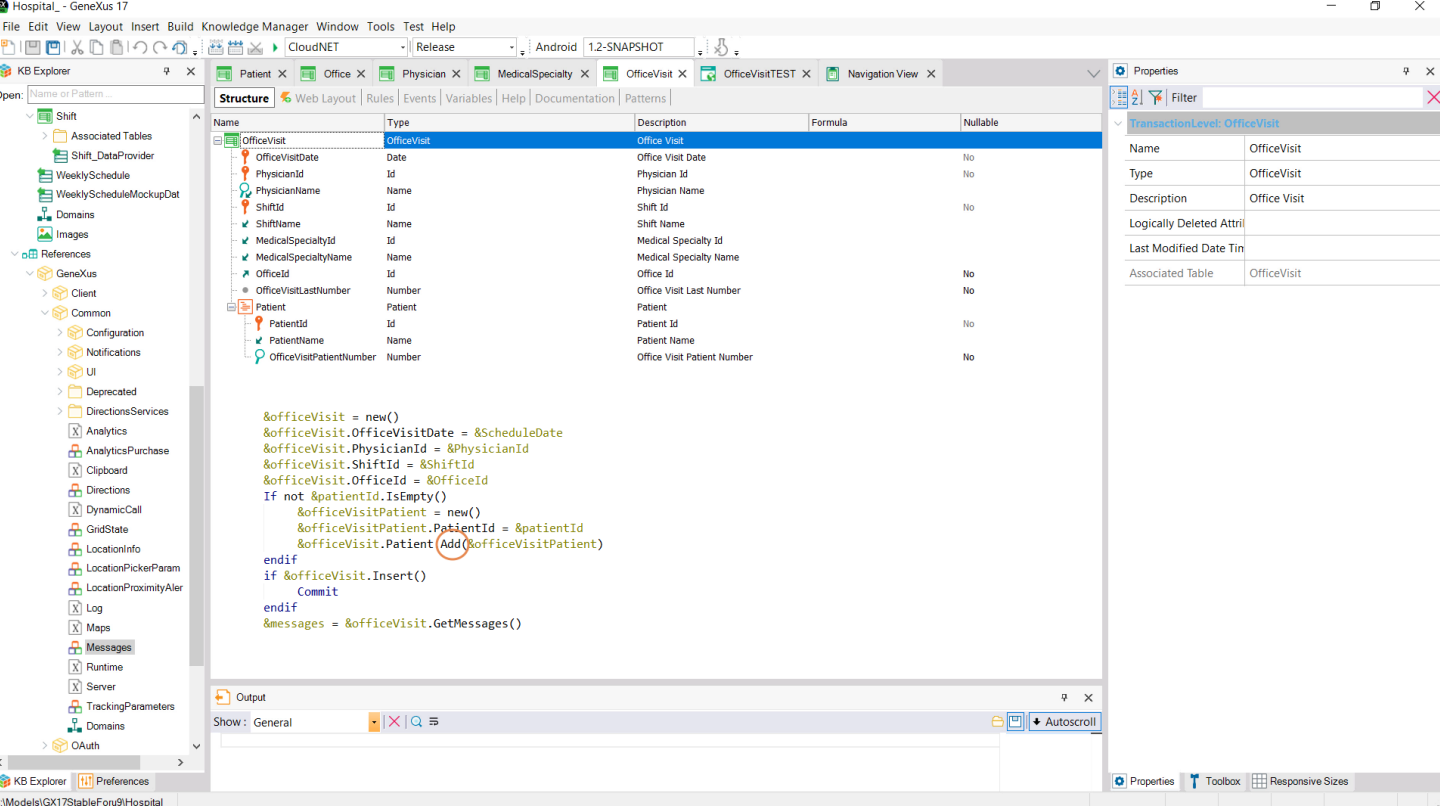
Shift Id:

Office Id:

Patient Id:

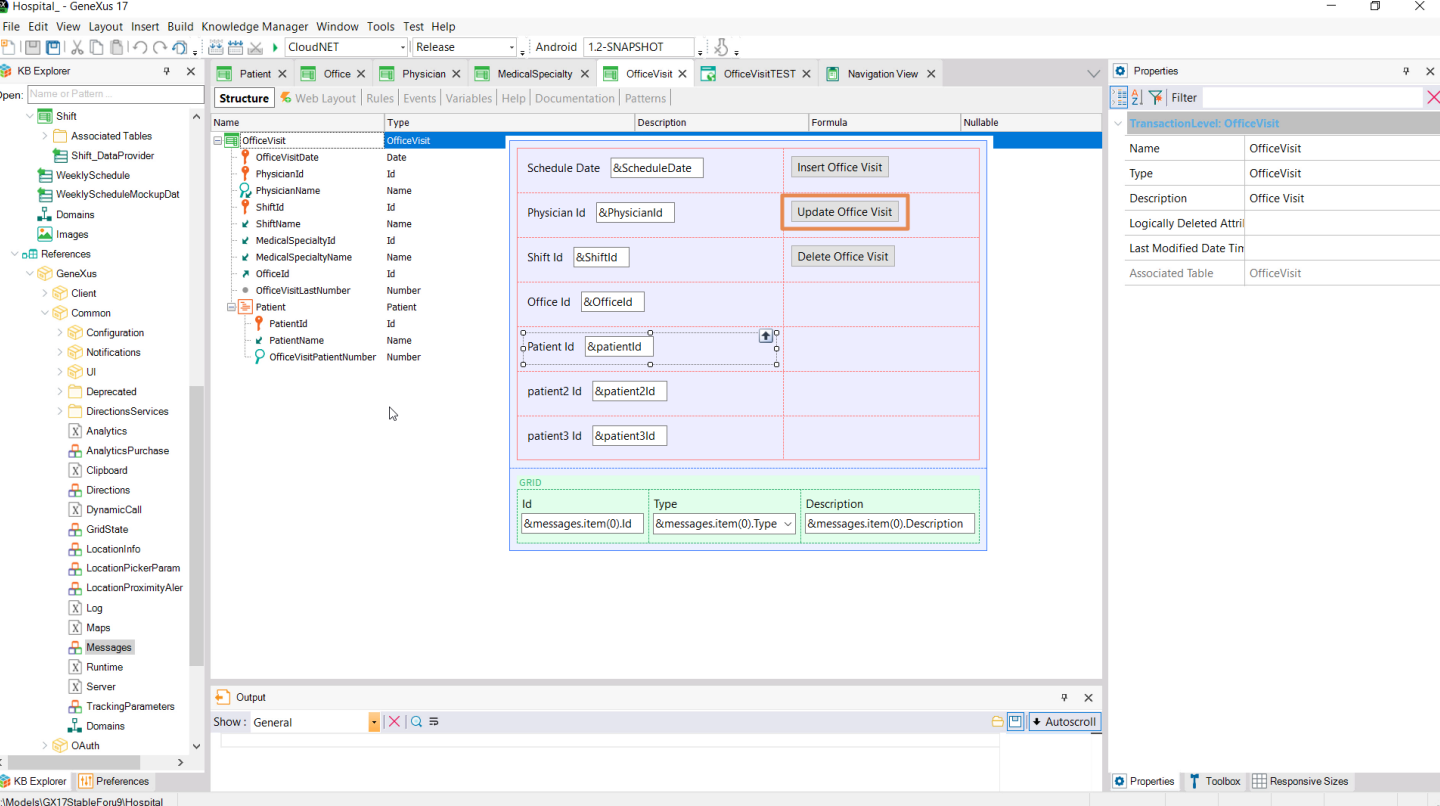
Id	Type	Description
----	------	-------------

What if now we choose a correct one? The Success message is displayed.
Logically, nothing was entered in the database.



Well, let's leave everything as it was.

So far, we have seen that regarding the checks and execution of business rules and events, there is almost no difference between entering a header and its lines through the transaction and doing it through the business component. We also reviewed how to work with the business component and, in particular, we added lines with the Add method of the collection. Then we will review another one, with a Data Provider.



Note: here we test the operation of the Business Component manually, through a web panel that we specially built and tested at runtime. But the way to test this so that it can always be tested again after any changes or even integration is with unit tests. It is not advisable to do manual tests like the ones we did here.

In the next video we will change an office visit, not insert it. Changing it involves changing data in the header and also in the lines: inserting new ones, modifying existing ones and deleting others. And we will see, as we did here, that the rules are triggered. We'll also look at deletion.

You could skip that video if you think you've mastered these cases and move on to the next one.

GeneXus™

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