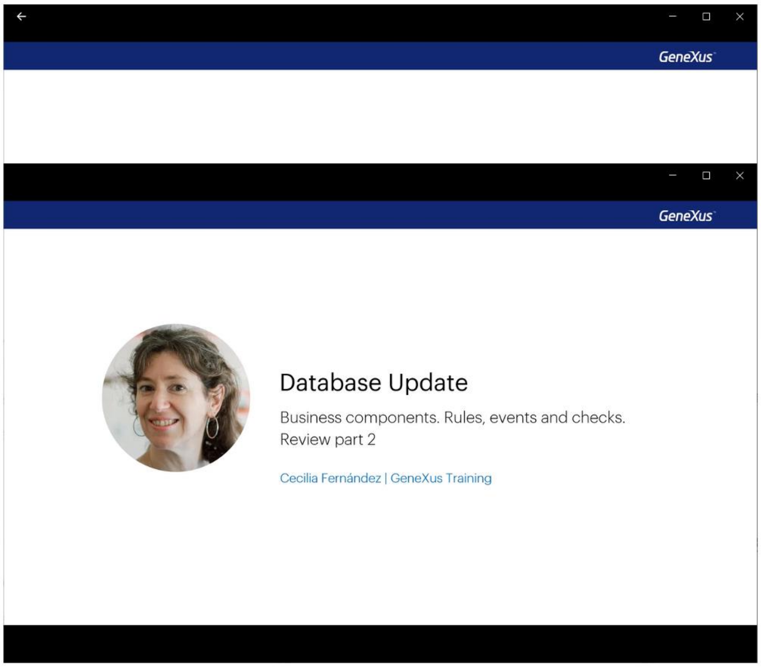
Database update

Business Components. Update or Insert (batch) and Data Provider

GeneXus™

Insert, Update, Delete

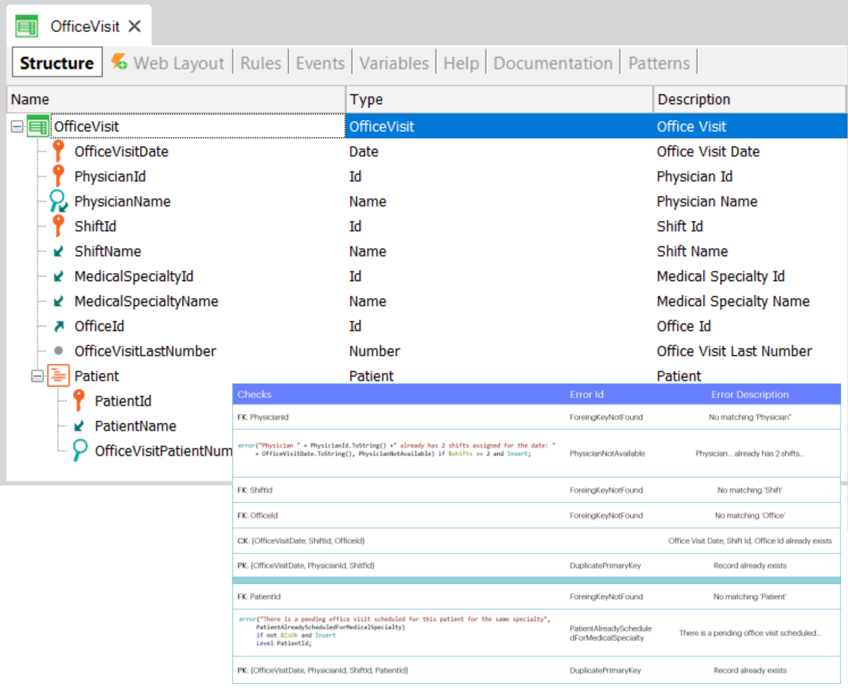


Database Update

Business components. Rules, events and checks. Review part 2

Cecilia Fernández | GeneXus Training

Focusing on a KB for a health care service, in previous videos we analyzed how to insert, update, and delete through the Business Component of a two-level transaction.

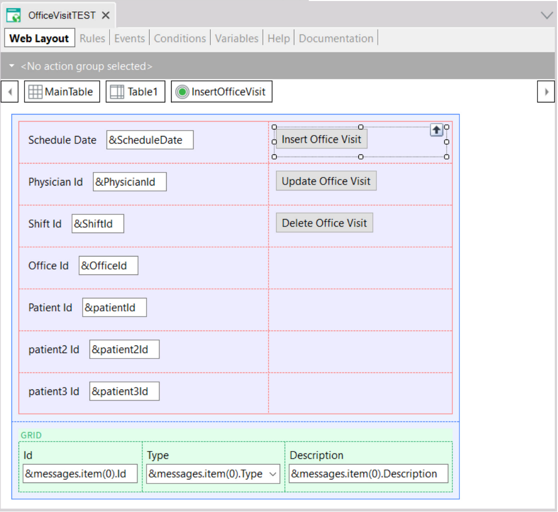


Name	Type	Description
OfficeVisit	OfficeVisit	Office Visit
OfficeVisitDate	Date	Office Visit Date
PhysicianId	Id	Physician Id
PhysicianName	Name	Physician Name
ShiftId	Id	Shift Id
ShiftName	Name	Shift Name
MedicalSpecialtyId	Id	Medical Specialty Id
MedicalSpecialtyName	Name	Medical Specialty Name
OfficeId	Id	Office Id
OfficeVisitLastNumber	Number	Office Visit Last Number
Patient	Patient	Patient
PatientId		
PatientName		
OfficeVisitPatientNum		

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, ShitfId}	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;
	PatientAlreadyScheduledForMedicalSpecialty	There is a pending office visit scheduled...

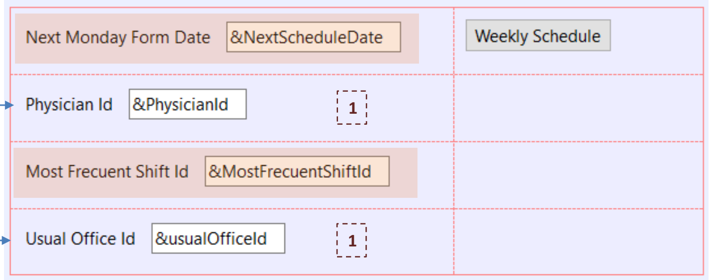
PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}	DuplicatePrimaryKey	Record already exists



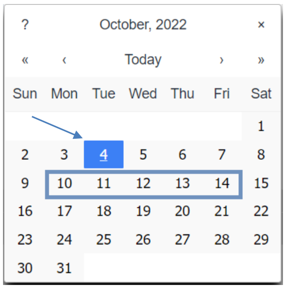
Schedule Date	&ScheduleDate
Physician Id	&PhysicianId
Shift Id	&ShiftId
Office Id	&OfficeId
Patient Id	&patientId
patient2 Id	&patient2Id
patient3 Id	&patient3Id

Id	Type	Description
&messages.item(0).Id	&messages.item(0).Type	&messages.item(0).Description

In those videos, starting from the two-level OfficeVisit transaction, which had a series of declared rules, in particular some error rules, what we tested was to insert an office visit with a patient, update an office visit (changing the office and modifying a line, deleting one and adding another) and deleting an office visit through a Business Component variable.



Next Monday Form Date	&NextScheduleDate		Weekly Schedule
Physician Id	&PhysicianId	1	
Most Frecuent Shift Id	&MostFrecuentShiftId		
Usual Office Id	&usualOfficeId	1	



? October, 2022 ×

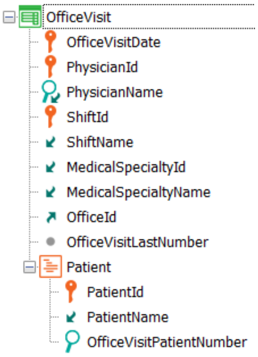
« ‹ Today › »

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

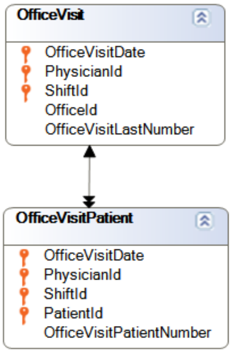
	Monday	Tuesday	Wednesday	Thursday	Friday
Shift 1 morning	Physician: 1 Office: 1		Physician: 1 Office: 1		Physician: 1 Office: 1
Shift 2 intermediate		Physician: 1 Office: 1		Physician: 1 Office: 1	
Shift 3 evening		Physician: 1 Office: 1		Physician: 1 Office: 1	
	10-10-2022	10-11-2022	10-12-2022	10-13-2022	10-14-2022

Now we want to assign in batch mode the office visits for a given physician in the week following a given date, in his or her preferred office.

For example, for doctor 1, office 1, we will want this office visit schedule: Monday, Wednesday and Friday in the shift chosen for those days, for example, 1....
Tuesdays and Thursdays on the next shift, 2....
And Tuesdays and Thursdays, in the remaining shift, shift 3...

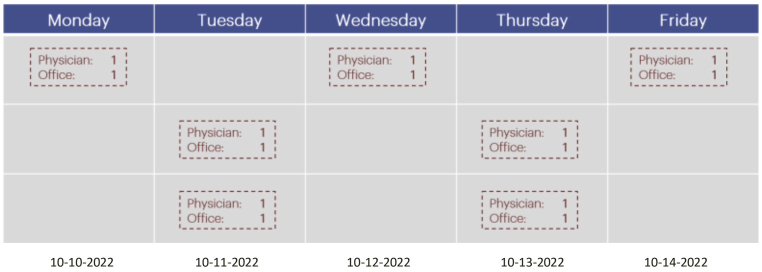


- OfficeVisit
  - OfficeVisitDate
  - PhysicianId
  - PhysicianName
  - ShiftId
  - ShiftName
  - MedicalSpecialtyId
  - MedicalSpecialtyName
  - OfficeId
  - OfficeVisitLastNumber
  - Patient
    - PatientId
    - PatientName
    - OfficeVisitPatientNumber



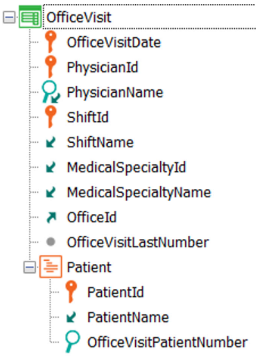
OfficeVisit
OfficeVisitDate
PhysicianId
ShiftId
OfficeId
OfficeVisitLastNumber

OfficeVisitPatient
OfficeVisitDate
PhysicianId
ShiftId
PatientId
OfficeVisitPatientNumber

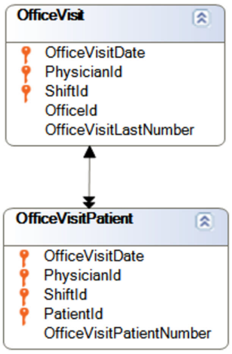


Monday	Tuesday	Wednesday	Thursday	Friday
Physician: 1 Office: 1		Physician: 1 Office: 1		Physician: 1 Office: 1
	Physician: 1 Office: 1		Physician: 1 Office: 1	
	Physician: 1 Office: 1		Physician: 1 Office: 1	
10-10-2022	10-11-2022	10-12-2022	10-13-2022	10-14-2022

Basically, we will only want to assign the office visits without patients; that is to say, only the records corresponding to the 7 headers of the transaction, meaning that we will work on the OfficeVisit table, complying with all the checks and rules of the transaction, so we will do it with a business component.



- OfficeVisit
  - OfficeVisitDate
  - PhysicianId
  - PhysicianName
  - ShiftId
  - ShiftName
  - MedicalSpecialtyId
  - MedicalSpecialtyName
  - OfficeId
  - OfficeVisitLastNumber
  - Patient
    - PatientId
    - PatientName
    - OfficeVisitPatientNumber

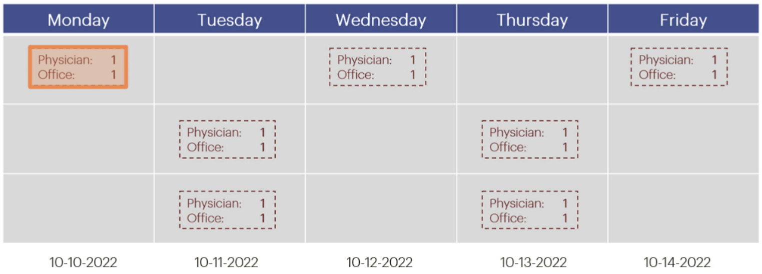


OfficeVisit
- OfficeVisitDate
- PhysicianId
- ShiftId
- OfficeId
- OfficeVisitLastNumber

OfficeVisitPatient
- OfficeVisitDate
- PhysicianId
- ShiftId
- PatientId
- OfficeVisitPatientNumber

&officeVisit

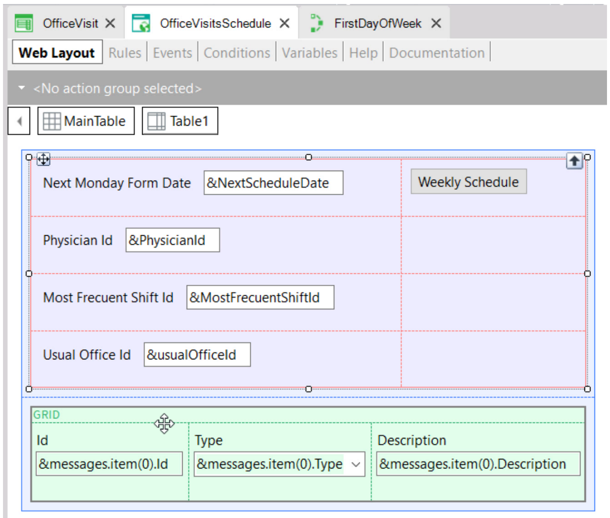
OfficeVisitDate	10-10-2022
PhysicianId	1
PhysicianName	Doctor 1
ShiftId	1
ShiftName	Shift morning
MedicalSpecialtyId	1
MedicalSpecialtyName	Family Medicine
OfficeId	1
OfficeVisitLastNumber	0
Patient	



Monday	Tuesday	Wednesday	Thursday	Friday
Physician: 1 Office: 1		Physician: 1 Office: 1		Physician: 1 Office: 1
	Physician: 1 Office: 1		Physician: 1 Office: 1	
	Physician: 1 Office: 1		Physician: 1 Office: 1	
10-10-2022	10-11-2022	10-12-2022	10-13-2022	10-14-2022

For example, using a single variable, filling in the required data for the first record and applying the Insert method to it; then clearing it and doing the same for the following office visit, on Wednesday: that is, loading the data and applying Insert...And so on...

We clear, load the data for Friday, shift 1... insert.
We clear, load data for Tuesday, shift 2... insert.
We clear, load data for Thursday, shift 2... insert.
We clear, load data for Tuesday, shift 3... insert.
We clear, load data for Thursday, shift 3... insert.



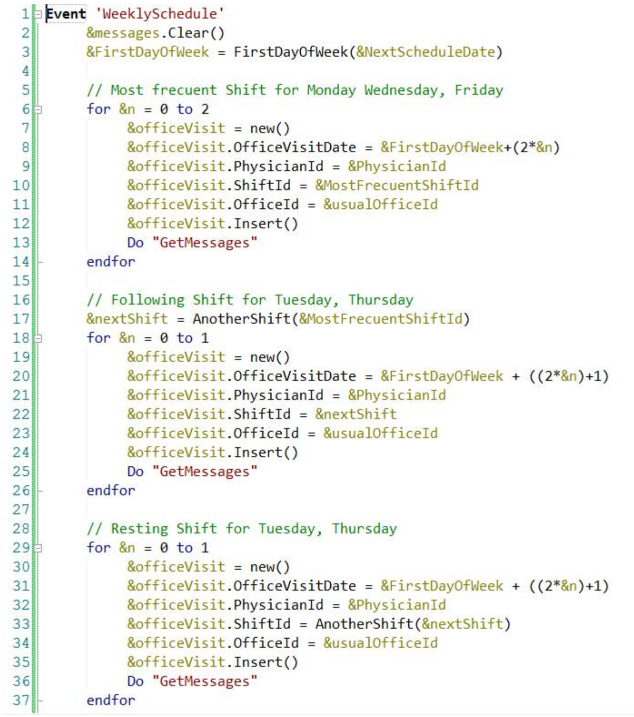
Next Monday Form Date &NextScheduleDate

Physician Id &PhysicianId

Most Frecuent Shift Id &MostFrecuentShiftId

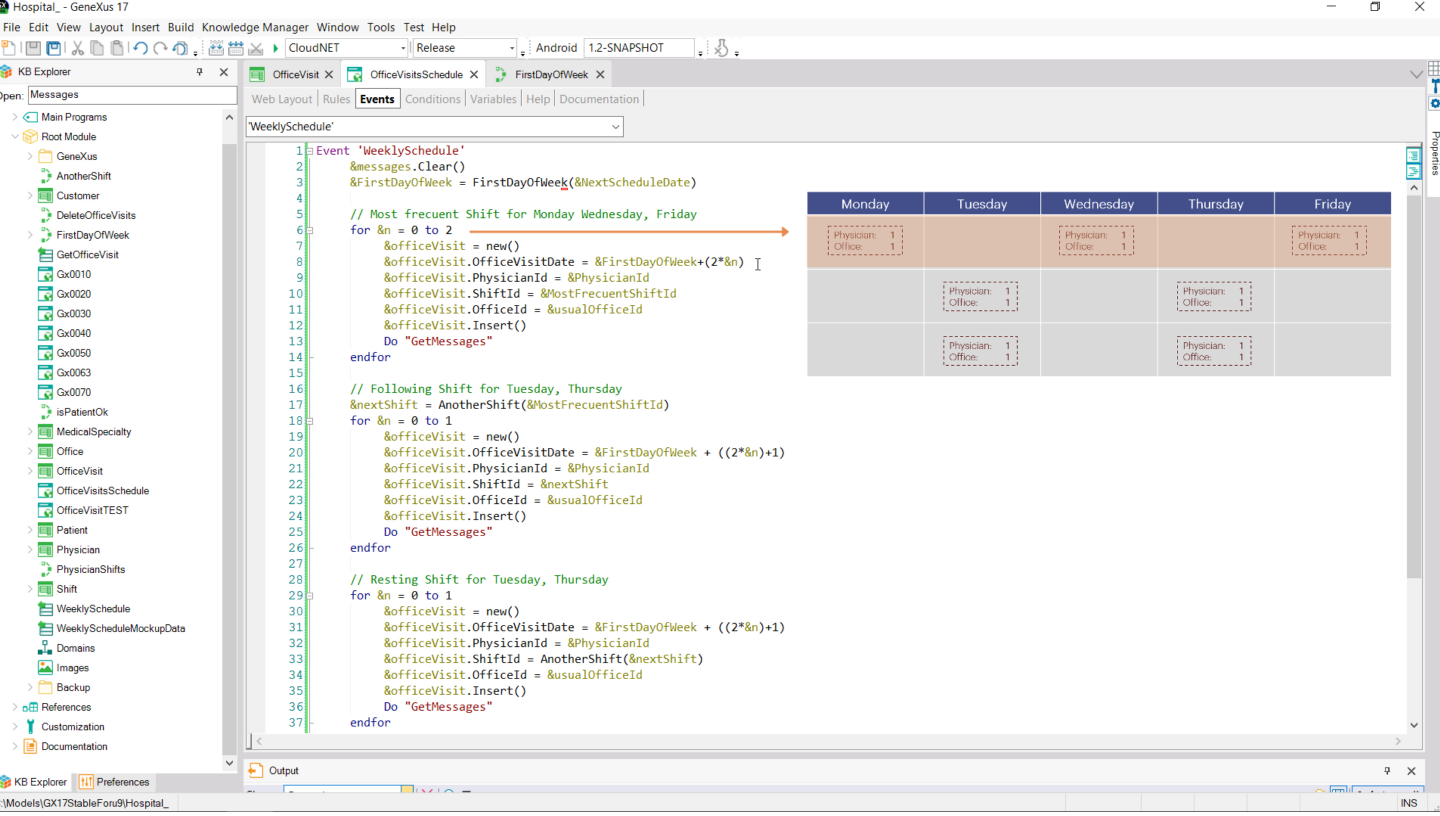
Usual Office Id &usualOfficeId

Id	Type	Description
&messages.item(0).Id	&messages.item(0).Type	&messages.item(0).Description



```
Event 'WeeklySchedule'
    &messages.Clear()
    &FirstDayOfWeek = FirstDayOfWeek(&NextScheduleDate)

    // Most frecuent Shift for Monday Wednesday, Friday
    for &n = 0 to 2
        &officeVisit = new()
        &officeVisit.OfficeVisitDate = &FirstDayOfWeek+(2*&n)
        &officeVisit.PhysicianId = &PhysicianId
        &officeVisit.ShiftId = &MostFrecuentShiftId
        &officeVisit.OfficeId = &usualOfficeId
        &officeVisit.Insert()
        Do "GetMessages"
    endfor

    // Following Shift for Tuesday, Thursday
    &nextShift = AnotherShift(&MostFrecuentShiftId)
    for &n = 0 to 1
        &officeVisit = new()
        &officeVisit.OfficeVisitDate = &FirstDayOfWeek + ((2*&n)+1)
        &officeVisit.PhysicianId = &PhysicianId
        &officeVisit.ShiftId = &nextShift
        &officeVisit.OfficeId = &usualOfficeId
        &officeVisit.Insert()
        Do "GetMessages"
    endfor

    // Resting Shift for Tuesday, Thursday
    for &n = 0 to 1
        &officeVisit = new()
        &officeVisit.OfficeVisitDate = &FirstDayOfWeek + ((2*&n)+1)
        &officeVisit.PhysicianId = &PhysicianId
        &officeVisit.ShiftId = AnotherShift(&nextShift)
        &officeVisit.OfficeId = &usualOfficeId
        &officeVisit.Insert()
        Do "GetMessages"
    endfor
```

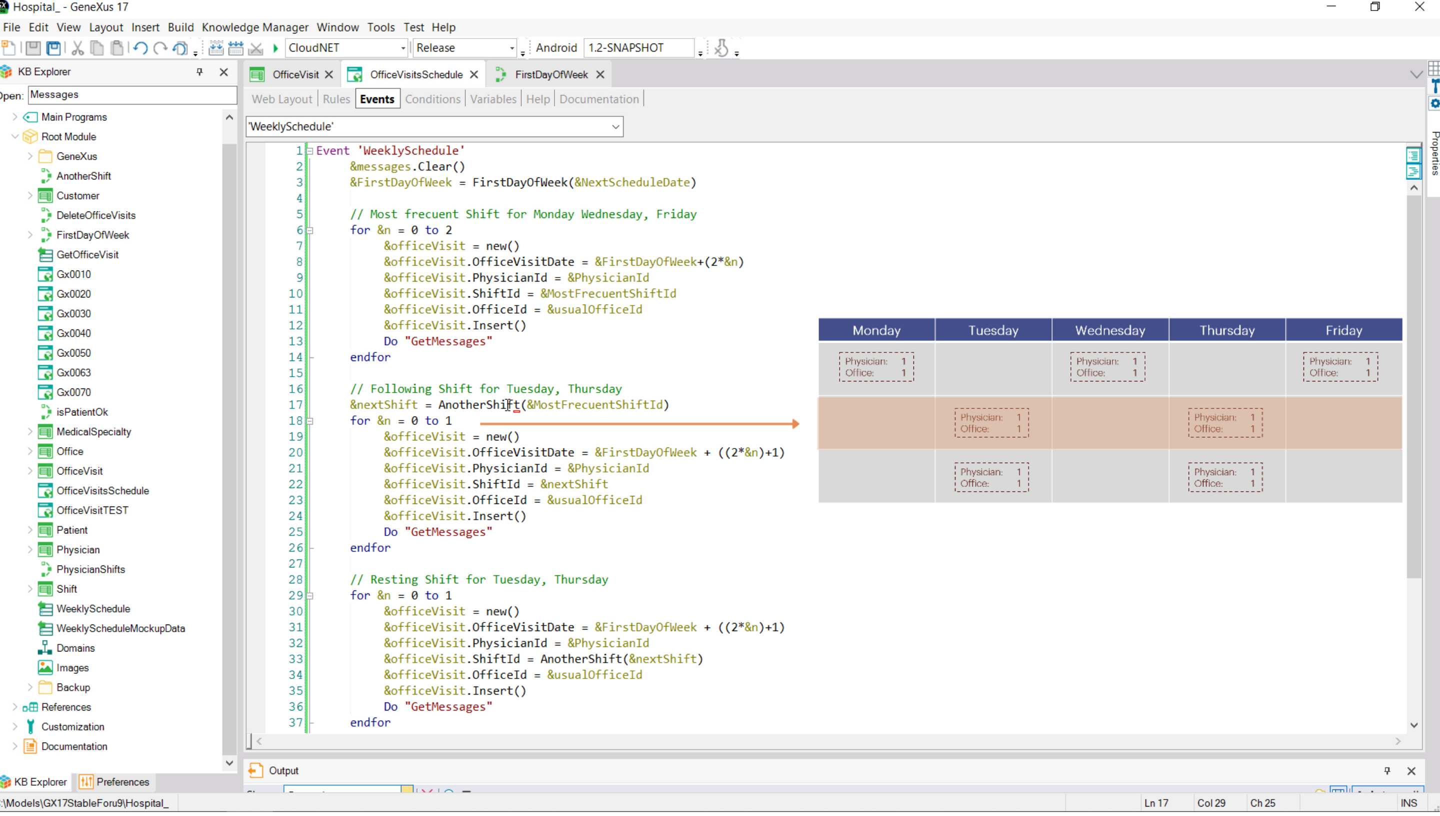
Here we implement the requirement with a Web Panel where we ask the user for the data (here we will show the resulting message collection) and in the event... we first clear the message collection that would have been shown in a previous execution, call this proc that returns for this date, the date of the following Monday.



```
Event 'WeeklySchedule'
    &messages.Clear()
    &FirstDayOfWeek = FirstDayOfWeek(&NextScheduleDate)

    // Most frecuent Shift for Monday Wednesday, Friday
    for &n = 0 to 2
        &officeVisit = new()
        &officeVisit.OfficeVisitDate = &FirstDayOfWeek+(2*&n)
        &officeVisit.PhysicianId = &PhysicianId
        &officeVisit.ShiftId = &MostFrecuentShiftId
        &officeVisit.OfficeId = &usualOfficeId
        &officeVisit.Insert()
        Do "GetMessages"
    endfor

    // Following Shift for Tuesday, Thursday
    &nextShift = AnotherShift(&MostFrecuentShiftId)
    for &n = 0 to 1
        &officeVisit = new()
        &officeVisit.OfficeVisitDate = &FirstDayOfWeek + ((2*&n)+1)
        &officeVisit.PhysicianId = &PhysicianId
        &officeVisit.ShiftId = &nextShift
        &officeVisit.OfficeId = &usualOfficeId
        &officeVisit.Insert()
        Do "GetMessages"
    endfor

    // Resting Shift for Tuesday, Thursday
    for &n = 0 to 1
        &officeVisit = new()
        &officeVisit.OfficeVisitDate = &FirstDayOfWeek + ((2*&n)+1)
        &officeVisit.PhysicianId = &PhysicianId
        &officeVisit.ShiftId = AnotherShift(&nextShift)
        &officeVisit.OfficeId = &usualOfficeId
        &officeVisit.Insert()
        Do "GetMessages"
    endfor
```

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

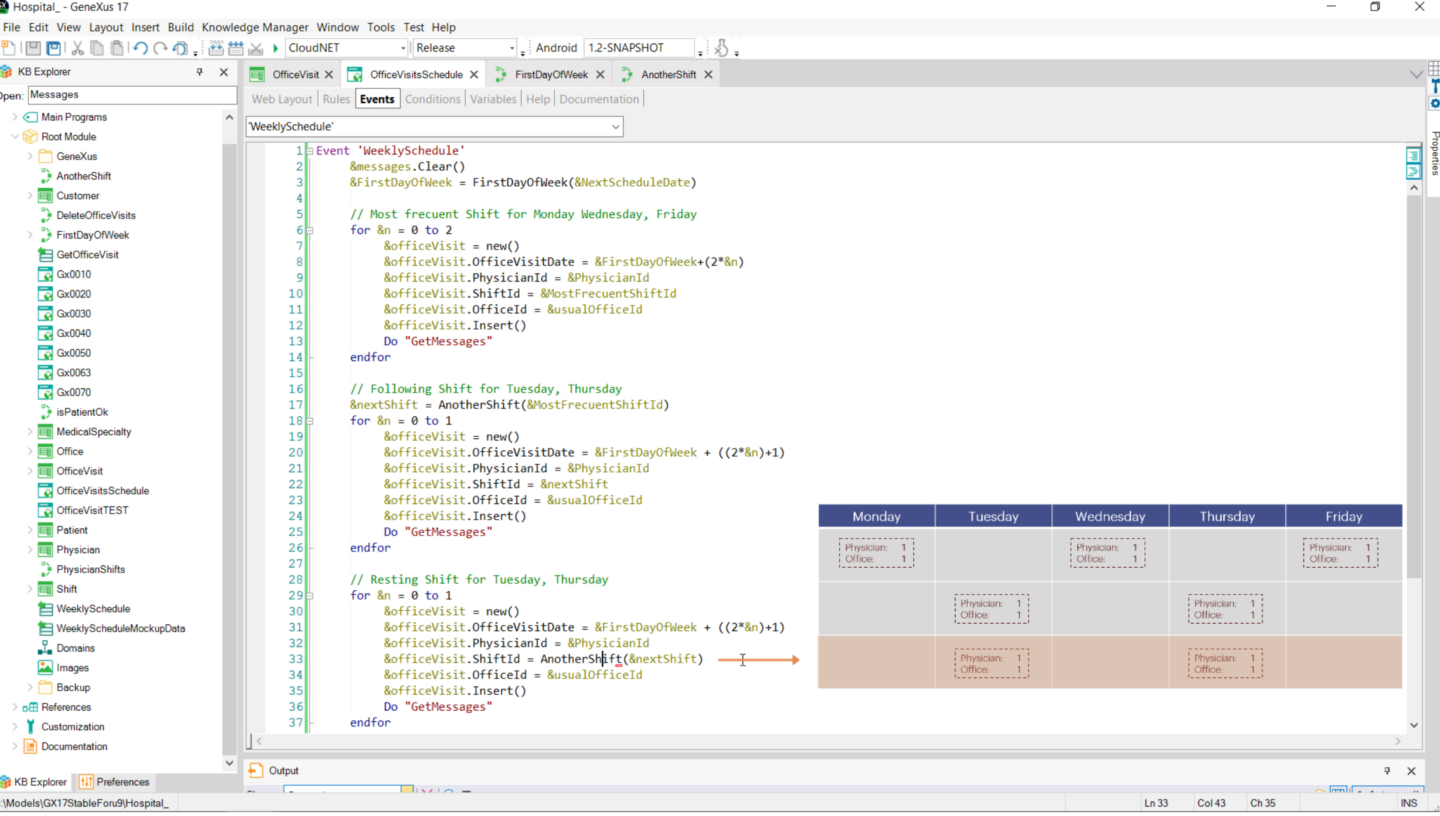
And instead of asking 7 times for memory for the variable, assigning the corresponding elements for each one, and doing Insert, we choose to use the repetitive structure “for” to write the insertion only once for Monday, Wednesday and Friday, where only the day changes, and everything else is kept...



```
Event 'WeeklySchedule'
	&messages.Clear()
	&FirstDayOfWeek = FirstDayOfWeek(&NextScheduleDate)

	// Most frecuent Shift for Monday Wednesday, Friday
	for &n = 0 to 2
		&officeVisit = new()
		&officeVisit.OfficeVisitDate = &FirstDayOfWeek+(2*&n)
		&officeVisit.PhysicianId = &PhysicianId
		&officeVisit.ShiftId = &MostFrecuentShiftId
		&officeVisit.OfficeId = &usualOfficeId
		&officeVisit.Insert()
		Do "GetMessages"
	endfor

	// Following Shift for Tuesday, Thursday
	&nextShift = AnotherShift(&MostFrecuentShiftId)
	for &n = 0 to 1
		&officeVisit = new()
		&officeVisit.OfficeVisitDate = &FirstDayOfWeek + ((2*&n)+1)
		&officeVisit.PhysicianId = &PhysicianId
		&officeVisit.ShiftId = &nextShift
		&officeVisit.OfficeId = &usualOfficeId
		&officeVisit.Insert()
		Do "GetMessages"
	endfor

	// Resting Shift for Tuesday, Thursday
	for &n = 0 to 1
		&officeVisit = new()
		&officeVisit.OfficeVisitDate = &FirstDayOfWeek + ((2*&n)+1)
		&officeVisit.PhysicianId = &PhysicianId
		&officeVisit.ShiftId = AnotherShift(&nextShift)
		&officeVisit.OfficeId = &usualOfficeId
		&officeVisit.Insert()
		Do "GetMessages"
	endfor
```

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

...Then another one for Tuesday and Thursday, which is the following shift (calculated with this proc), and here we assign it...

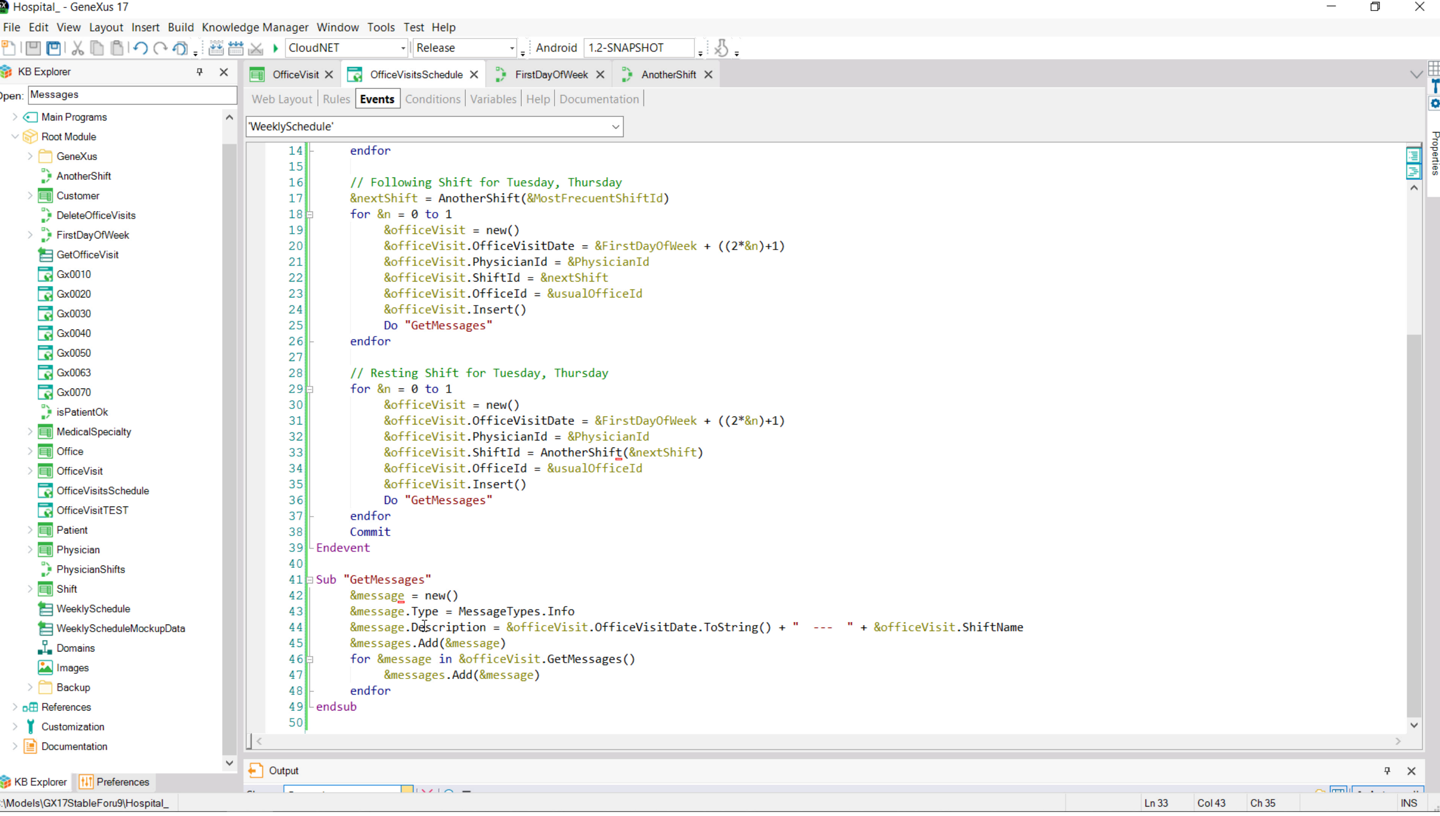


```
Event 'WeeklySchedule'
	&messages.Clear()
	&FirstDayOfWeek = FirstDayOfWeek(&NextScheduleDate)

	// Most frecuent Shift for Monday Wednesday, Friday
	for &n = 0 to 2
		&officeVisit = new()
		&officeVisit.OfficeVisitDate = &FirstDayOfWeek+(2*&n)
		&officeVisit.PhysicianId = &PhysicianId
		&officeVisit.ShiftId = &MostFrecuentShiftId
		&officeVisit.OfficeId = &usualOfficeId
		&officeVisit.Insert()
		Do "GetMessages"
	endfor

	// Following Shift for Tuesday, Thursday
	&nextShift = AnotherShift(&MostFrecuentShiftId)
	for &n = 0 to 1
		&officeVisit = new()
		&officeVisit.OfficeVisitDate = &FirstDayOfWeek + ((2*&n)+1)
		&officeVisit.PhysicianId = &PhysicianId
		&officeVisit.ShiftId = &nextShift
		&officeVisit.OfficeId = &usualOfficeId
		&officeVisit.Insert()
		Do "GetMessages"
	endfor

	// Resting Shift for Tuesday, Thursday
	for &n = 0 to 1
		&officeVisit = new()
		&officeVisit.OfficeVisitDate = &FirstDayOfWeek + ((2*&n)+1)
		&officeVisit.PhysicianId = &PhysicianId
		&officeVisit.ShiftId = AnotherShift(&nextShift)
		&officeVisit.OfficeId = &usualOfficeId
		&officeVisit.Insert()
		Do "GetMessages"
	endfor
```

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

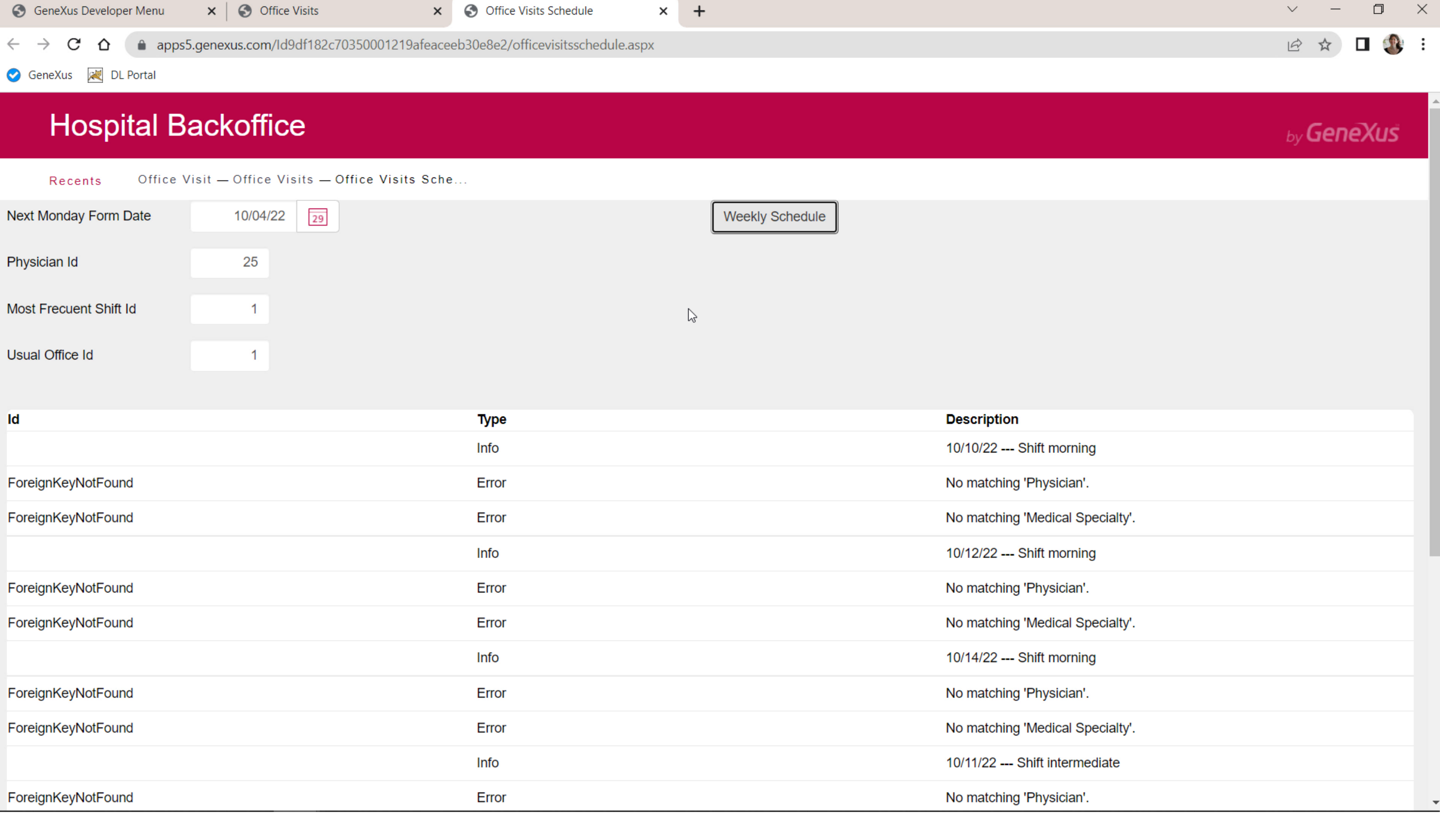
…and then another “for,” also for Tuesday and Thursday, but in the remaining shift.

Note that in addition to improving the code, in short what we are doing is use a BC variable, assigning it the header values, and using the Insert method to try to insert in the corresponding table.



```
    endfor

    // Following Shift for Tuesday, Thursday
    &nextShift = AnotherShift(&MostFrecuentShiftId)
    for &n = 0 to 1
        &officeVisit = new()
        &officeVisit.OfficeVisitDate = &FirstDayOfWeek + ((2*&n)+1)
        &officeVisit.PhysicianId = &PhysicianId
        &officeVisit.ShiftId = &nextShift
        &officeVisit.OfficeId = &usualOfficeId
        &officeVisit.Insert()
        Do "GetMessages"
    endfor

    // Resting Shift for Tuesday, Thursday
    for &n = 0 to 1
        &officeVisit = new()
        &officeVisit.OfficeVisitDate = &FirstDayOfWeek + ((2*&n)+1)
        &officeVisit.PhysicianId = &PhysicianId
        &officeVisit.ShiftId = AnotherShift(&nextShift)
        &officeVisit.OfficeId = &usualOfficeId
        &officeVisit.Insert()
        Do "GetMessages"
    endfor
    Commit
Endevent

Sub "GetMessages"
    &message = new()
    &message.Type = MessageTypes.Info
    &message.Description = &officeVisit.OfficeVisitDate.ToString() + "  ---  " + &officeVisit.ShiftName
    &messages.Add(&message)
    for &message in &officeVisit.GetMessages()
        &messages.Add(&message)
    endfor
endsub
```

Then we process the messages obtained.

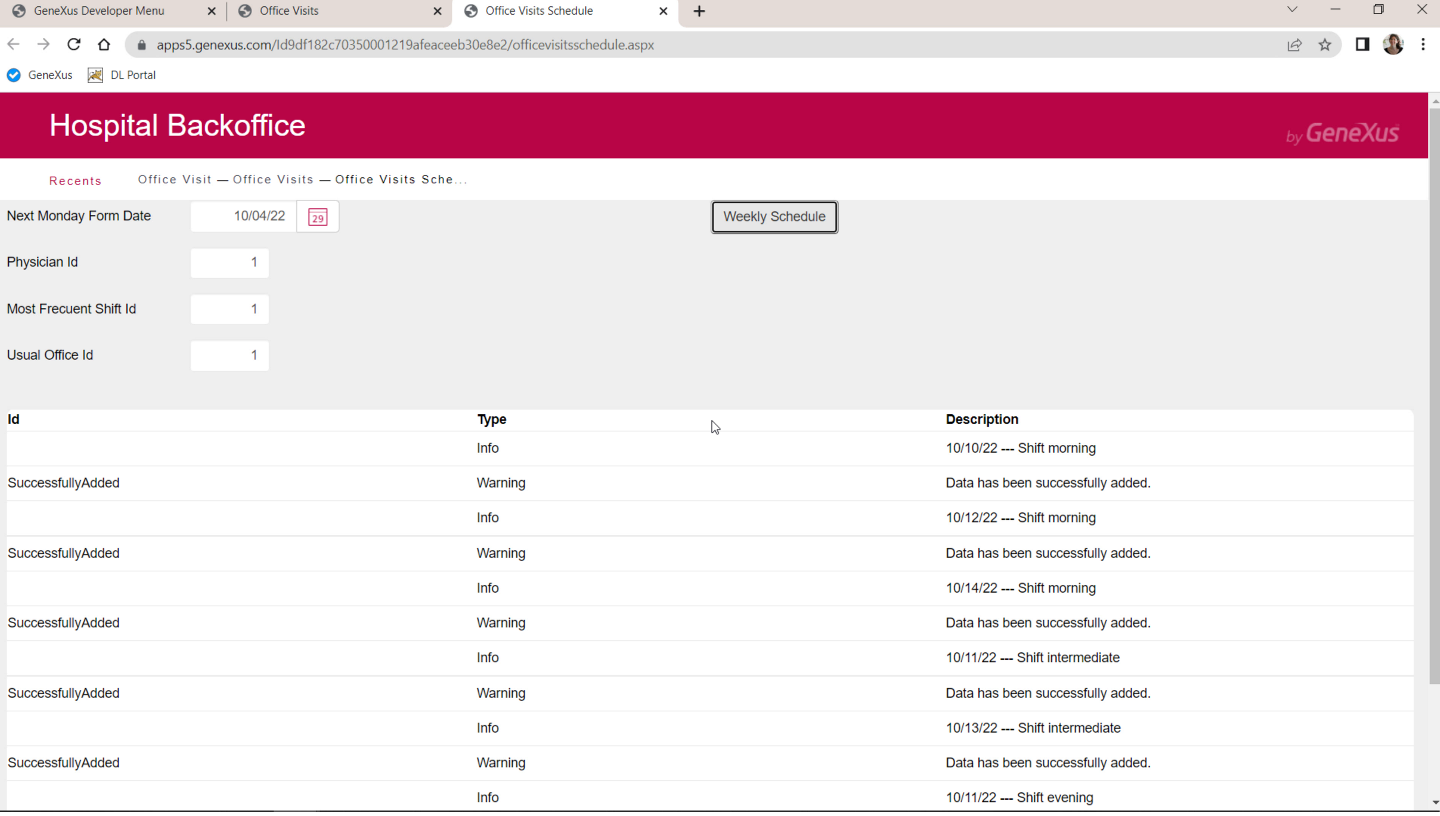

GeneXus Developer Menu
Office Visits
Office Visits Schedule
apps5.genexus.com/Id9df182c70350001219afeaceeb30e8e2/officevisitsschedule.aspx
GeneXus
DL Portal
Hospital Backoffice
by GeneXus
Recents
Office Visit — Office Visits — Office Visits Sche...
Weekly Schedule

Next Monday Form Date	10/04/22
Physician Id	25
Most Frecuent Shift Id	1
Usual Office Id	1

Id	Type	Description
	Info	10/10/22 --- Shift morning
ForeignKeyNotFound	Error	No matching 'Physician'.
ForeignKeyNotFound	Error	No matching 'Medical Specialty'.
	Info	10/12/22 --- Shift morning
ForeignKeyNotFound	Error	No matching 'Physician'.
ForeignKeyNotFound	Error	No matching 'Medical Specialty'.
	Info	10/14/22 --- Shift morning
ForeignKeyNotFound	Error	No matching 'Physician'.
ForeignKeyNotFound	Error	No matching 'Medical Specialty'.
	Info	10/11/22 --- Shift intermediate
ForeignKeyNotFound	Error	No matching 'Physician'.

Let's try this when there are no office visits yet in the table.

Of course, if we enter nonexistent foreign key values—for example, a nonexistent doctor—the insertion attempts will all fail. The insertion will be attempted by doing the same checks as through the transaction.

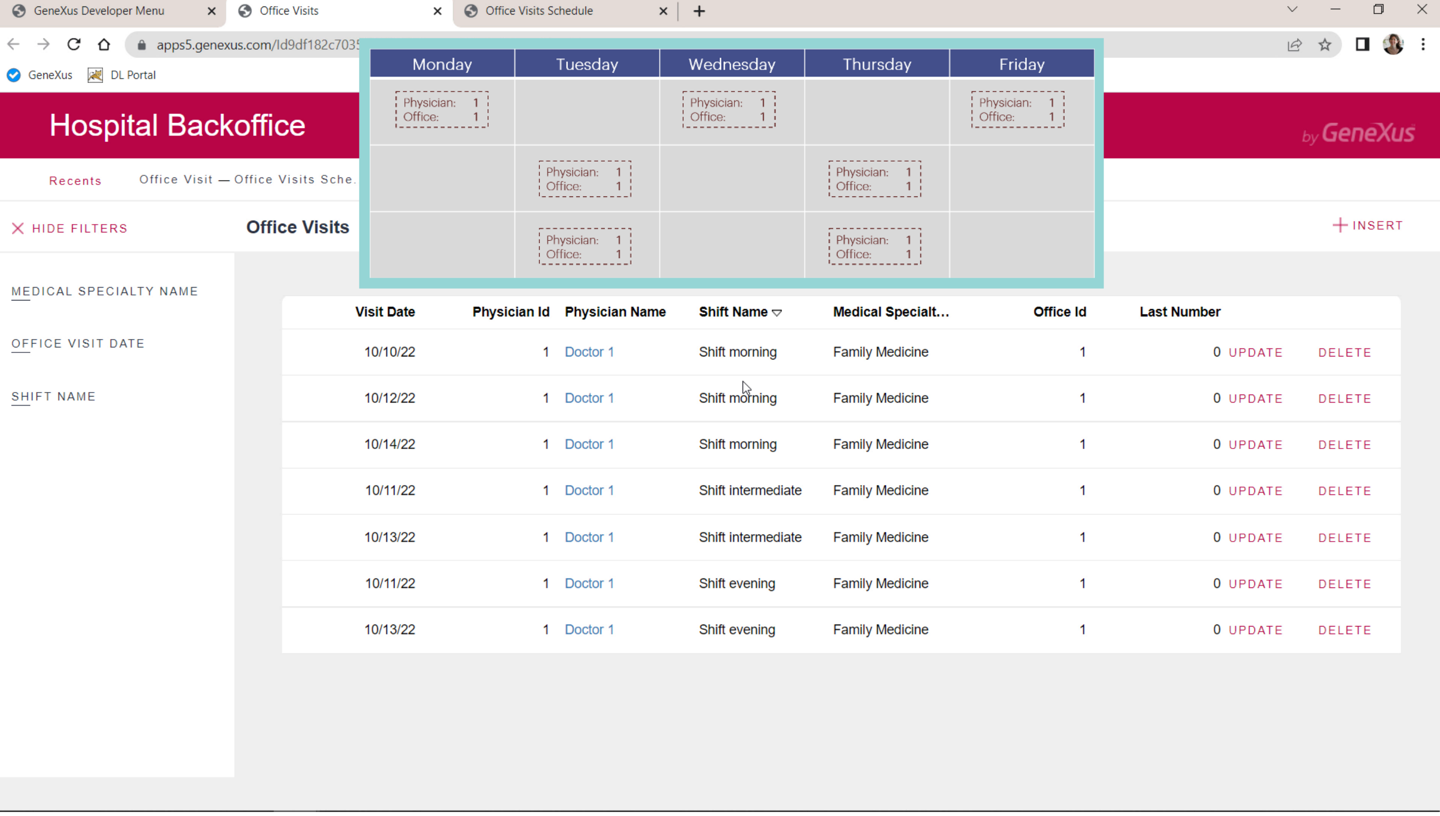


GeneXus Developer Menu
Office Visits
Office Visits Schedule
apps5.genexus.com/Id9df182c70350001219afeaceeb30e8e2/officevisitsschedule.aspx
GeneXus
DL Portal
Hospital Backoffice
by GeneXus
Recents
Office Visit — Office Visits — Office Visits Sche...
Weekly Schedule

Next Monday Form Date	10/04/22
Physician Id	1
Most Frecuent Shift Id	1
Usual Office Id	1

Id	Type	Description
	Info	10/10/22 --- Shift morning
SuccessfullyAdded	Warning	Data has been successfully added.
	Info	10/12/22 --- Shift morning
SuccessfullyAdded	Warning	Data has been successfully added.
	Info	10/14/22 --- Shift morning
SuccessfullyAdded	Warning	Data has been successfully added.
	Info	10/11/22 --- Shift intermediate
SuccessfullyAdded	Warning	Data has been successfully added.
	Info	10/13/22 --- Shift intermediate
SuccessfullyAdded	Warning	Data has been successfully added.
	Info	10/11/22 --- Shift evening

Now let's see the successful path.



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

Hospital Backoffice

Recents   Office Visit — Office Visits Sche.

HIDE FILTERS

MEDICAL SPECIALTY NAME

OFFICE VISIT DATE

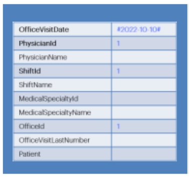
SHIFT NAME

Office Visits

INSERT

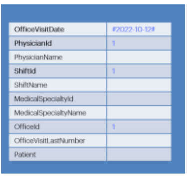
Visit Date	Physician Id	Physician Name	Shift Name	Medical Specialt...	Office Id	Last Number		
10/10/22	1	Doctor 1	Shift morning	Family Medicine	1	0	UPDATE	DELETE
10/12/22	1	Doctor 1	Shift morning	Family Medicine	1	0	UPDATE	DELETE
10/14/22	1	Doctor 1	Shift morning	Family Medicine	1	0	UPDATE	DELETE
10/11/22	1	Doctor 1	Shift intermediate	Family Medicine	1	0	UPDATE	DELETE
10/13/22	1	Doctor 1	Shift intermediate	Family Medicine	1	0	UPDATE	DELETE
10/11/22	1	Doctor 1	Shift evening	Family Medicine	1	0	UPDATE	DELETE
10/13/22	1	Doctor 1	Shift evening	Family Medicine	1	0	UPDATE	DELETE

&officeVisits



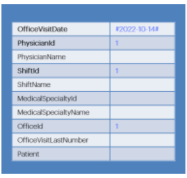
OfficeVisitDate	#2022-10-10#
PhysicianId	1
PhysicianName	
ShiftId	1
ShiftName	
MedicalSpecialtyId	
MedicalSpecialtyName	
OfficeId	1
OfficeVisitLastNumber	
Patient	

(1)



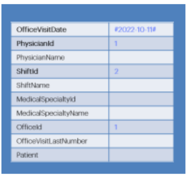
OfficeVisitDate	#2022-10-12#
PhysicianId	1
PhysicianName	
ShiftId	1
ShiftName	
MedicalSpecialtyId	
MedicalSpecialtyName	
OfficeId	1
OfficeVisitLastNumber	
Patient	

(2)



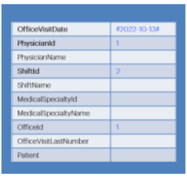
OfficeVisitDate	#2022-10-14#
PhysicianId	1
PhysicianName	
ShiftId	1
ShiftName	
MedicalSpecialtyId	
MedicalSpecialtyName	
OfficeId	1
OfficeVisitLastNumber	
Patient	

(3)



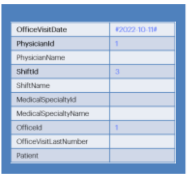
OfficeVisitDate	#2022-10-11#
PhysicianId	1
PhysicianName	
ShiftId	2
ShiftName	
MedicalSpecialtyId	
MedicalSpecialtyName	
OfficeId	1
OfficeVisitLastNumber	
Patient	

(4)



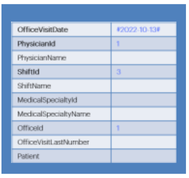
OfficeVisitDate	#2022-10-13#
PhysicianId	1
PhysicianName	
ShiftId	2
ShiftName	
MedicalSpecialtyId	
MedicalSpecialtyName	
OfficeId	1
OfficeVisitLastNumber	
Patient	

(5)



OfficeVisitDate	#2022-10-11#
PhysicianId	1
PhysicianName	
ShiftId	3
ShiftName	
MedicalSpecialtyId	
MedicalSpecialtyName	
OfficeId	1
OfficeVisitLastNumber	
Patient	

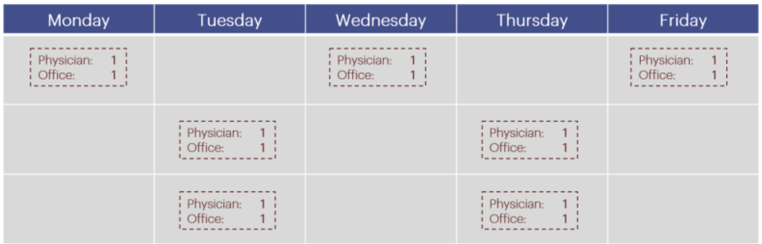
(6)



OfficeVisitDate	#2022-10-13#
PhysicianId	1
PhysicianName	
ShiftId	3
ShiftName	
MedicalSpecialtyId	
MedicalSpecialtyName	
OfficeId	1
OfficeVisitLastNumber	
Patient	

(7)

&officeVisits.Insert()

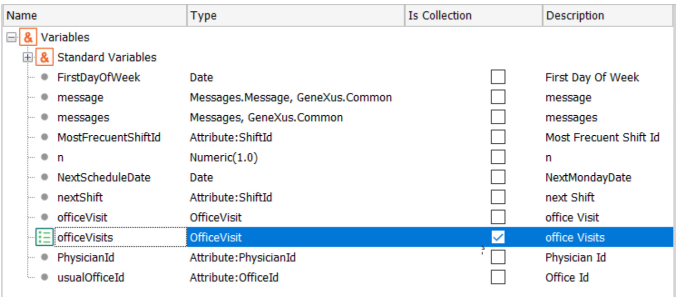


Monday	Tuesday	Wednesday	Thursday	Friday
Physician: 1 Office: 1		Physician: 1 Office: 1		Physician: 1 Office: 1
	Physician: 1 Office: 1		Physician: 1 Office: 1	
	Physician: 1 Office: 1		Physician: 1 Office: 1	

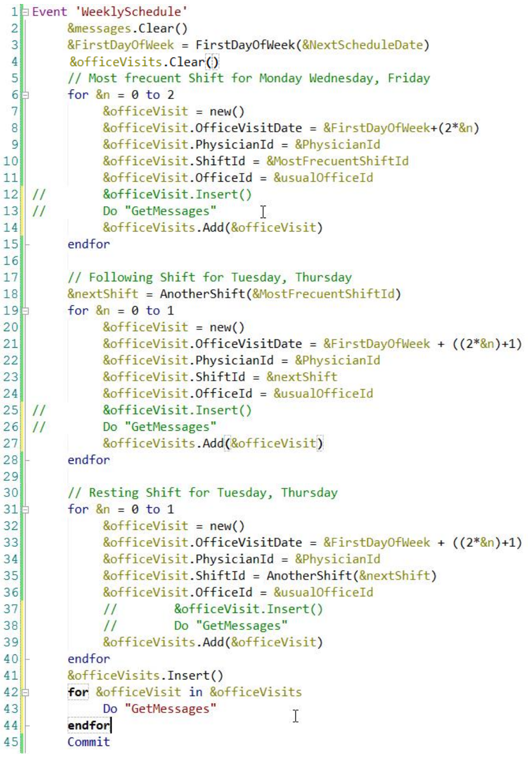
10-10-2022 10-11-2022 10-12-2022 10-13-2022 10-14-2022

Another way to program this would be to load each BC into an item in a collection, and then give the Insert order to the entire collection.

We know that internally each item will be accessed and an Insert of that item will be attempted.



Name	Type	Is Collection	Description
Variables			
Standard Variables			
FirstDayOfWeek	Date	☐	First Day Of Week
message	Messages.Message, GeneXus.Common	☐	message
messages	Messages, GeneXus.Common	☐	messages
MostFrecuentShiftId	Attribute:ShiftId	☐	Most Frecuent Shift Id
n	Numeric(1.0)	☐	n
NextScheduleDate	Date	☐	NextMondayDate
nextShift	Attribute:ShiftId	☐	next Shift
officeVisit	OfficeVisit	☐	office Visit
officeVisits	OfficeVisit	☑	office Visits
PhysicianId	Attribute:PhysicianId	☐	Physician Id
usualOfficeId	Attribute:OfficeId	☐	Office Id



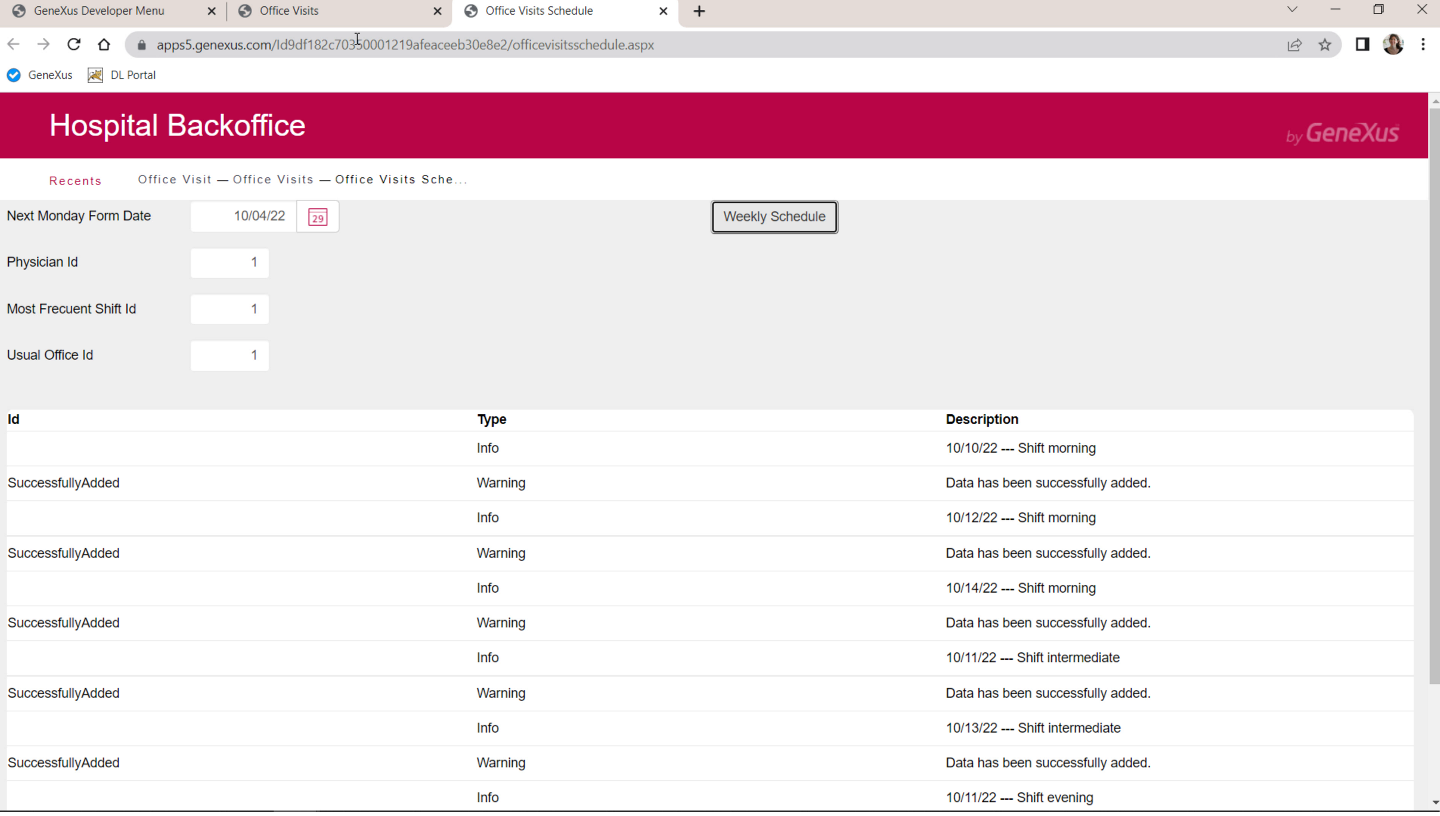
```
Event 'WeeklySchedule'
     &messages.Clear()
     &FirstDayOfWeek = FirstDayOfWeek(&NextScheduleDate)
      &officeVisits.Clear()
     // Most frecuent Shift for Monday Wednesday, Friday
     for &n = 0 to 2
          &officeVisit = new()
          &officeVisit.OfficeVisitDate = &FirstDayOfWeek+(2*&n)
          &officeVisit.PhysicianId = &PhysicianId
          &officeVisit.ShiftId = &MostFrecuentShiftId
          &officeVisit.OfficeId = &usualOfficeId
//        &officeVisit.Insert()
//        Do "GetMessages"
          &officeVisits.Add(&officeVisit)
     endfor

     // Following Shift for Tuesday, Thursday
     &nextShift = AnotherShift(&MostFrecuentShiftId)
     for &n = 0 to 1
          &officeVisit = new()
          &officeVisit.OfficeVisitDate = &FirstDayOfWeek + ((2*&n)+1)
          &officeVisit.PhysicianId = &PhysicianId
          &officeVisit.ShiftId = &nextShift
          &officeVisit.OfficeId = &usualOfficeId
//        &officeVisit.Insert()
//        Do "GetMessages"
          &officeVisits.Add(&officeVisit)
     endfor

     // Resting Shift for Tuesday, Thursday
     for &n = 0 to 1
          &officeVisit = new()
          &officeVisit.OfficeVisitDate = &FirstDayOfWeek + ((2*&n)+1)
          &officeVisit.PhysicianId = &PhysicianId
          &officeVisit.ShiftId = AnotherShift(&nextShift)
          &officeVisit.OfficeId = &usualOfficeId
          //        &officeVisit.Insert()
          //        Do "GetMessages"
          &officeVisits.Add(&officeVisit)
     endfor
     &officeVisits.Insert()
     for &officeVisit in &officeVisits
          Do "GetMessages"
     endfor
     Commit
```

That is to say, we would define an officeVisits variable, collection of the OfficeVisit BC.

And we would load the information in the &officeVisit variable, but instead of already inserting in the database, we would ask to add the BC to the collection, for all the BC instances.
And only later we would ask for the insertion of the entire collection.

Of course, here if we want to get the messages of each Insert (that of each item) we will have to run through the collection and call the subroutine.

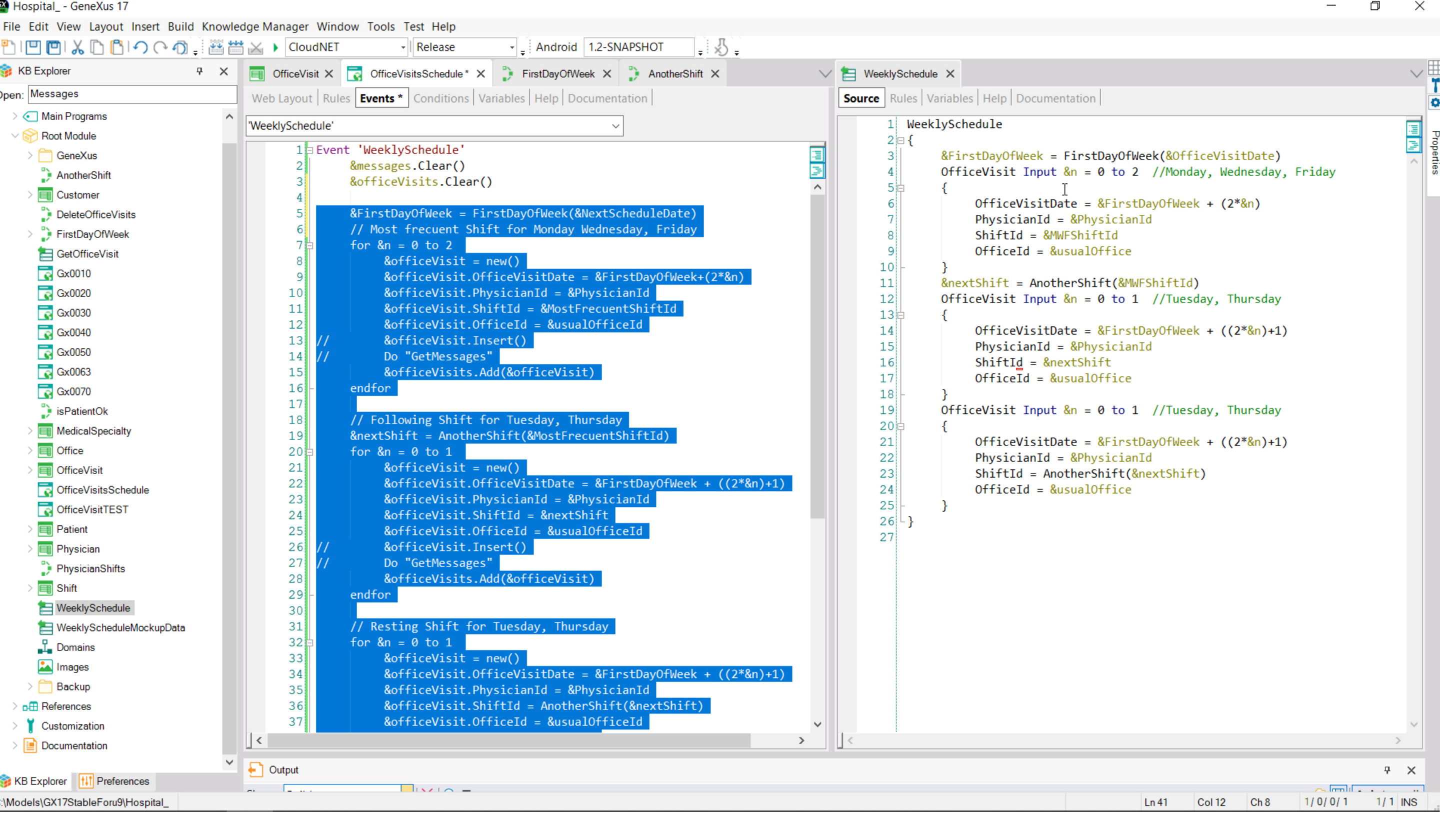
And we should clear the BC collection before we start.


GeneXus Developer Menu
Office Visits
Office Visits Schedule
apps5.genexus.com/Id9df182c70350001219afeaceeb30e8e2/officevisitsschedule.aspx
GeneXus
DL Portal
Hospital Backoffice
by GeneXus
Recents
Office Visit — Office Visits — Office Visits Sche...
Weekly Schedule

Next Monday Form Date	10/04/22
Physician Id	1
Most Frecuent Shift Id	1
Usual Office Id	1

Id	Type	Description
	Info	10/10/22 --- Shift morning
SuccessfullyAdded	Warning	Data has been successfully added.
	Info	10/12/22 --- Shift morning
SuccessfullyAdded	Warning	Data has been successfully added.
	Info	10/14/22 --- Shift morning
SuccessfullyAdded	Warning	Data has been successfully added.
	Info	10/11/22 --- Shift intermediate
SuccessfullyAdded	Warning	Data has been successfully added.
	Info	10/13/22 --- Shift intermediate
SuccessfullyAdded	Warning	Data has been successfully added.
	Info	10/11/22 --- Shift evening

If we try now with the empty table as before, it should give exactly the same result...

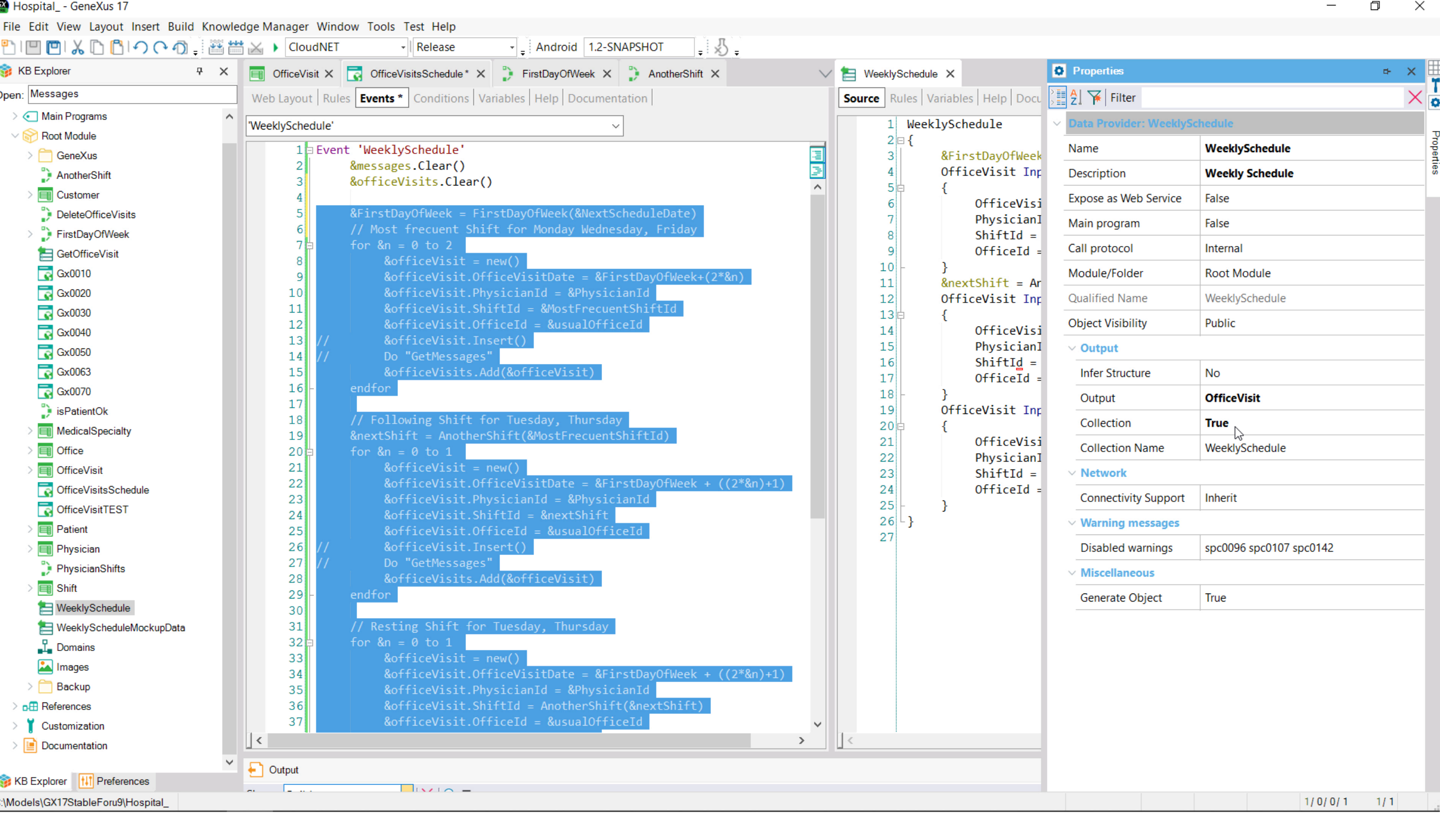


```
Event 'WeeklySchedule'
	&messages.Clear()
	&officeVisits.Clear()

	&FirstDayOfWeek = FirstDayOfWeek(&NextScheduleDate)
	// Most frecuent Shift for Monday Wednesday, Friday
	for &n = 0 to 2
		&officeVisit = new()
		&officeVisit.OfficeVisitDate = &FirstDayOfWeek+(2*&n)
		&officeVisit.PhysicianId = &PhysicianId
		&officeVisit.ShiftId = &MostFrecuentShiftId
		&officeVisit.OfficeId = &usualOfficeId
//		&officeVisit.Insert()
//		Do "GetMessages"
		&officeVisits.Add(&officeVisit)
	endfor

	// Following Shift for Tuesday, Thursday
	&nextShift = AnotherShift(&MostFrecuentShiftId)
	for &n = 0 to 1
		&officeVisit = new()
		&officeVisit.OfficeVisitDate = &FirstDayOfWeek + ((2*&n)+1)
		&officeVisit.PhysicianId = &PhysicianId
		&officeVisit.ShiftId = &nextShift
		&officeVisit.OfficeId = &usualOfficeId
//		&officeVisit.Insert()
//		Do "GetMessages"
		&officeVisits.Add(&officeVisit)
	endfor

	// Resting Shift for Tuesday, Thursday
	for &n = 0 to 1
		&officeVisit = new()
		&officeVisit.OfficeVisitDate = &FirstDayOfWeek + ((2*&n)+1)
		&officeVisit.PhysicianId = &PhysicianId
		&officeVisit.ShiftId = AnotherShift(&nextShift)
		&officeVisit.OfficeId = &usualOfficeId
```

```
WeeklySchedule
{
	&FirstDayOfWeek = FirstDayOfWeek(&OfficeVisitDate)
	OfficeVisit Input &n = 0 to 2  //Monday, Wednesday, Friday
	{
		OfficeVisitDate = &FirstDayOfWeek + (2*&n)
		PhysicianId = &PhysicianId
		ShiftId = &MWFShiftId
		OfficeId = &usualOffice
	}
	&nextShift = AnotherShift(&MWFShiftId)
	OfficeVisit Input &n = 0 to 1  //Tuesday, Thursday
	{
		OfficeVisitDate = &FirstDayOfWeek + ((2*&n)+1)
		PhysicianId = &PhysicianId
		ShiftId = &nextShift
		OfficeId = &usualOffice
	}
	OfficeVisit Input &n = 0 to 1  //Tuesday, Thursday
	{
		OfficeVisitDate = &FirstDayOfWeek + ((2*&n)+1)
		PhysicianId = &PhysicianId
		ShiftId = AnotherShift(&nextShift)
		OfficeId = &usualOffice
	}
}
```

Wouldn't it be simpler to do it using a Data Provider instead of loading the collection manually?

That is to say, replace all this...



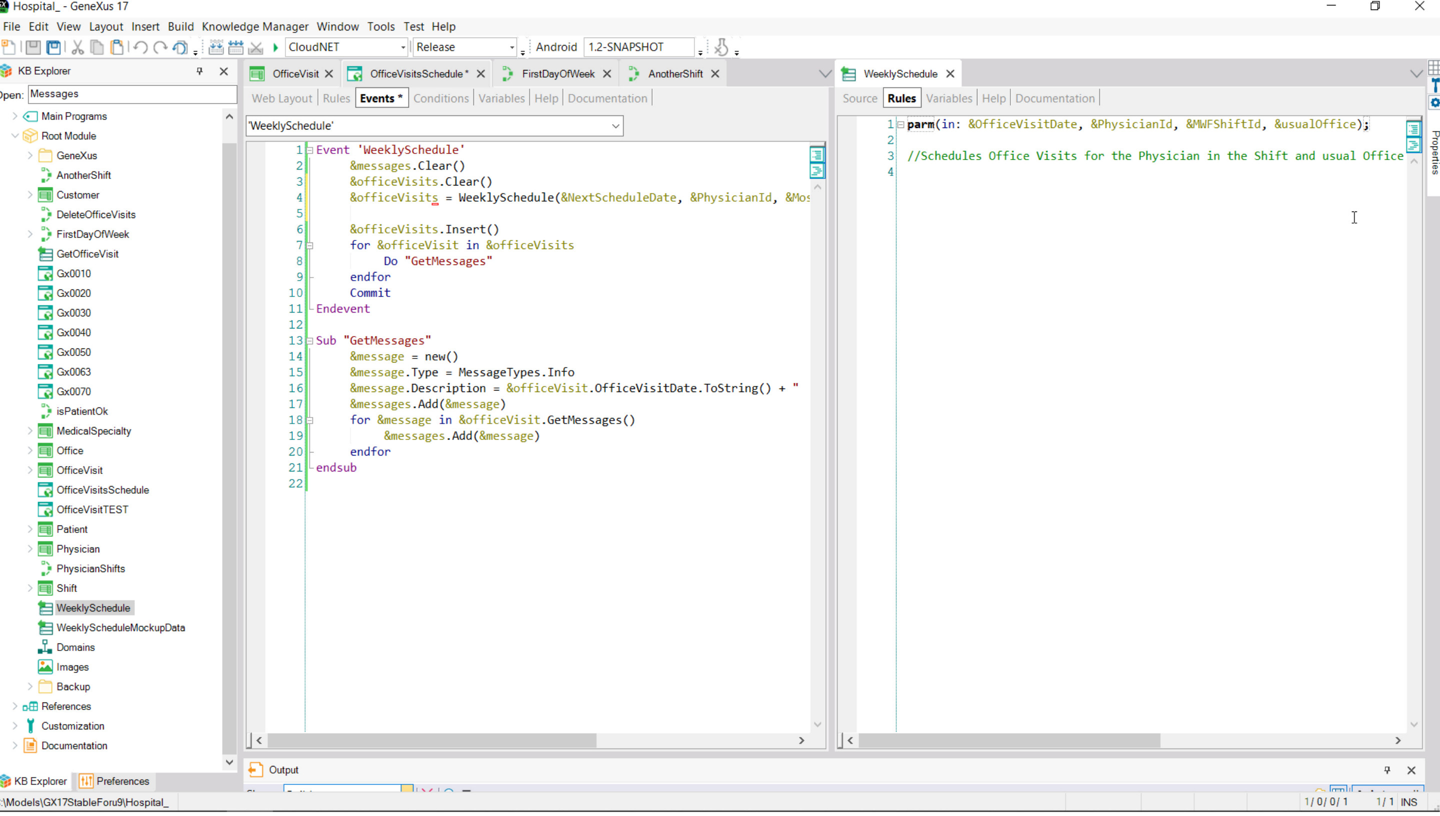
```
Event 'WeeklySchedule'
      &messages.Clear()
      &officeVisits.Clear()

      &FirstDayOfWeek = FirstDayOfWeek(&NextScheduleDate)
      // Most frecuent Shift for Monday Wednesday, Friday
      for &n = 0 to 2
            &officeVisit = new()
            &officeVisit.OfficeVisitDate = &FirstDayOfWeek+(2*&n)
            &officeVisit.PhysicianId = &PhysicianId
            &officeVisit.ShiftId = &MostFrecuentShiftId
            &officeVisit.OfficeId = &usualOfficeId
//          &officeVisit.Insert()
//          Do "GetMessages"
            &officeVisits.Add(&officeVisit)
      endfor

      // Following Shift for Tuesday, Thursday
      &nextShift = AnotherShift(&MostFrecuentShiftId)
      for &n = 0 to 1
            &officeVisit = new()
            &officeVisit.OfficeVisitDate = &FirstDayOfWeek + ((2*&n)+1)
            &officeVisit.PhysicianId = &PhysicianId
            &officeVisit.ShiftId = &nextShift
            &officeVisit.OfficeId = &usualOfficeId
//          &officeVisit.Insert()
//          Do "GetMessages"
            &officeVisits.Add(&officeVisit)
      endfor

      // Resting Shift for Tuesday, Thursday
      for &n = 0 to 1
            &officeVisit = new()
            &officeVisit.OfficeVisitDate = &FirstDayOfWeek + ((2*&n)+1)
            &officeVisit.PhysicianId = &PhysicianId
            &officeVisit.ShiftId = AnotherShift(&nextShift)
            &officeVisit.OfficeId = &usualOfficeId
```

```
WeeklySchedule
{
      &FirstDayOfWeek
      OfficeVisit Inp
      {
            OfficeVisi
            PhysicianI
            ShiftId =
            OfficeId =
      }
      &nextShift = Ar
      OfficeVisit Inp
      {
            OfficeVisi
            PhysicianI
            ShiftId =
            OfficeId =
      }
      OfficeVisit Inp
      {
            OfficeVisi
            PhysicianI
            ShiftId =
            OfficeId =
      }
}
```

Data Provider: WeeklySchedule

Name	**WeeklySchedule**
Description	**Weekly Schedule**
Expose as Web Service	False
Main program	False
Call protocol	Internal
Module/Folder	Root Module
Qualified Name	WeeklySchedule
Object Visibility	Public
Output	
Infer Structure	No
Output	**OfficeVisit**
Collection	**True**
Collection Name	WeeklySchedule
Network	
Connectivity Support	Inherit
Warning messages	
Disabled warnings	spc0096 spc0107 spc0142
Miscellaneous	
Generate Object	True

... with an invocation to this Data Provider, which returns a collection of the OfficeVisit BC...

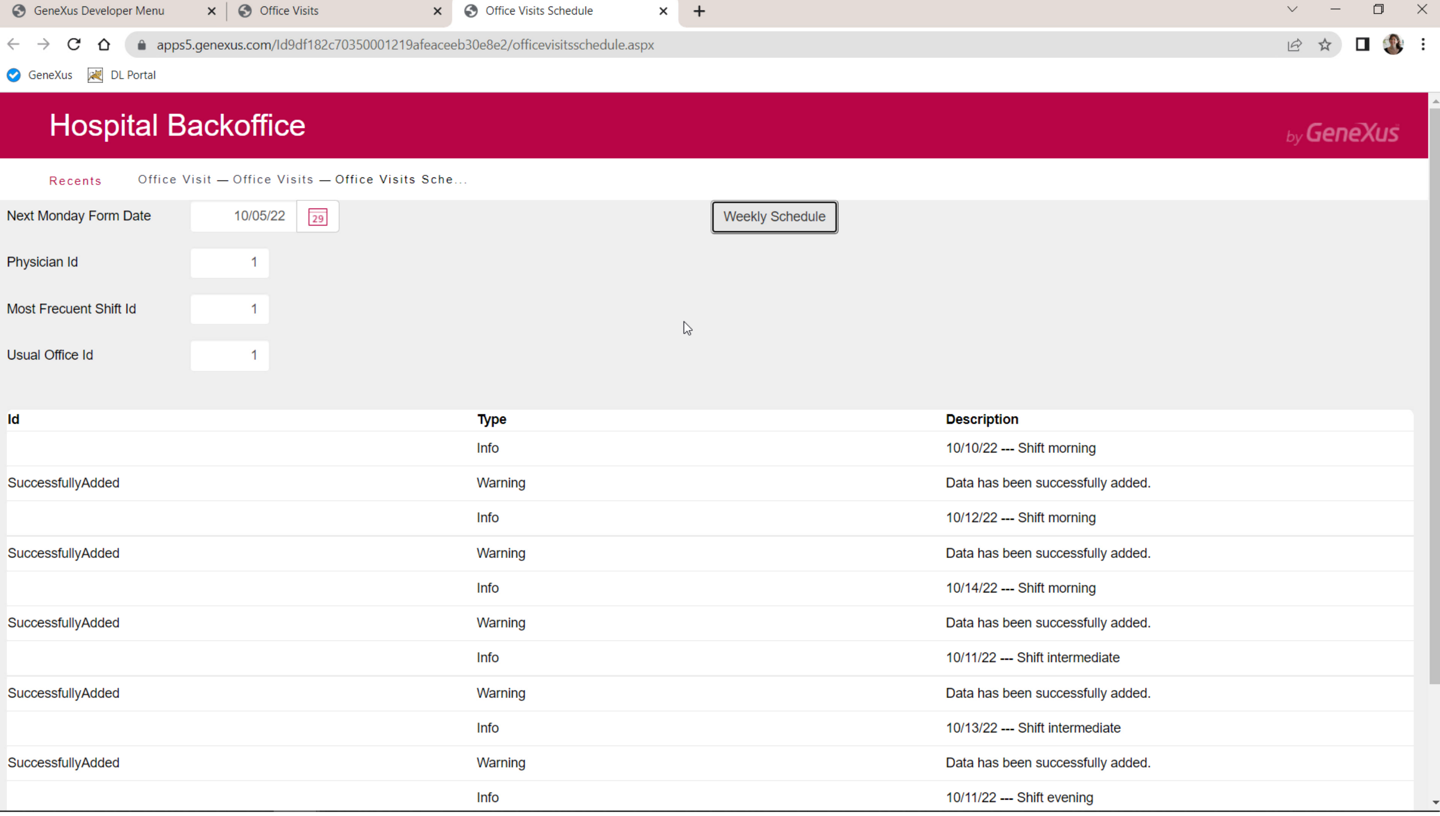


```
Event 'WeeklySchedule'
	&messages.Clear()
	&officeVisits.Clear()
	&officeVisits = WeeklySchedule(&NextScheduleDate, &PhysicianId, &Mos

	&officeVisits.Insert()
	for &officeVisit in &officeVisits
		Do "GetMessages"
	endfor
	Commit
Endevent

Sub "GetMessages"
	&message = new()
	&message.Type = MessageTypes.Info
	&message.Description = &officeVisit.OfficeVisitDate.ToString() + "
	&messages.Add(&message)
	for &message in &officeVisit.GetMessages()
		&messages.Add(&message)
	endfor
endsub
```

```
parm(in: &OfficeVisitDate, &PhysicianId, &MWFShiftId, &usualOffice);

//Schedules Office Visits for the Physician in the Shift and usual Office
```

...loaded in the same way as in the web panel event, but at a higher level.

Then we load the collection variable with the result of the invocation to the Data Provider, to which we must pass all the parameters.



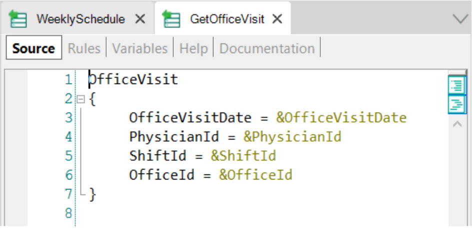
GeneXus Developer Menu
Office Visits
Office Visits Schedule
apps5.genexus.com/Id9df182c70350001219afeaceeb30e8e2/officevisitsschedule.aspx
GeneXus
DL Portal
Hospital Backoffice
by GeneXus
Recents
Office Visit — Office Visits — Office Visits Sche...
Weekly Schedule

Next Monday Form Date	10/05/22
Physician Id	1
Most Frecuent Shift Id	1
Usual Office Id	1

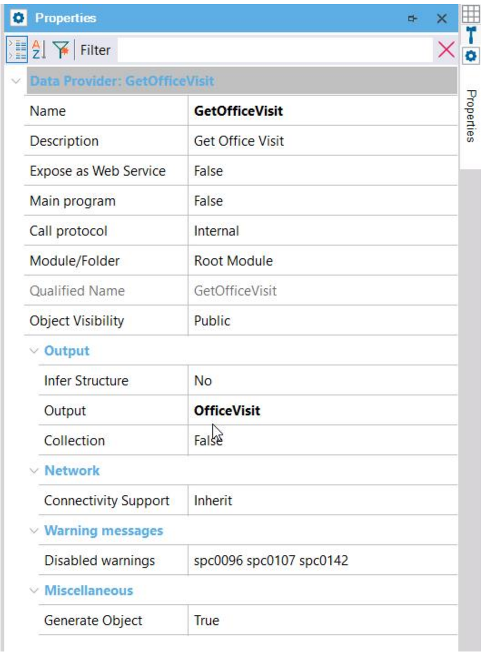
Id	Type	Description
	Info	10/10/22 --- Shift morning
SuccessfullyAdded	Warning	Data has been successfully added.
	Info	10/12/22 --- Shift morning
SuccessfullyAdded	Warning	Data has been successfully added.
	Info	10/14/22 --- Shift morning
SuccessfullyAdded	Warning	Data has been successfully added.
	Info	10/11/22 --- Shift intermediate
SuccessfullyAdded	Warning	Data has been successfully added.
	Info	10/13/22 --- Shift intermediate
SuccessfullyAdded	Warning	Data has been successfully added.
	Info	10/11/22 --- Shift evening

Let’s try it. Again, having cleared the table, we see that it gives exactly the same result.

```
parm(in: &OfficeVisitDate, in: &PhysicianId, in: &ShiftId, in: &OfficeId);
```



```
OfficeVisit
{
      OfficeVisitDate = &OfficeVisitDate
      PhysicianId = &PhysicianId
      ShiftId = &ShiftId
      OfficeId = &OfficeId
}
```

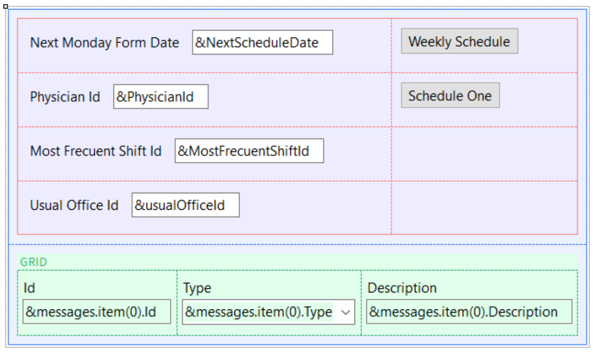


Data Provider: GetOfficeVisit

Name	GetOfficeVisit
Description	Get Office Visit
Expose as Web Service	False
Main program	False
Call protocol	Internal
Module/Folder	Root Module
Qualified Name	GetOfficeVisit
Object Visibility	Public
Output	
Infer Structure	No
Output	OfficeVisit
Collection	False
Network	
Connectivity Support	Inherit
Warning messages	
Disabled warnings	spc0096 spc0107 spc0142
Miscellaneous	
Generate Object	True

Of course, we can use the Data Provider to return a collection of Business Components as we did here, which we then assign to this collection variable, or to return a single one and do the Insert on that one.

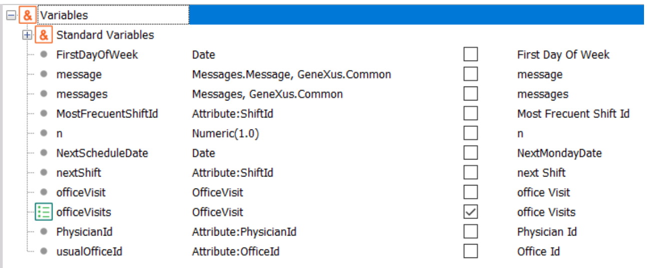
For example, here we have this other Data Provider that will return a single BC, this one, OfficeVisit. Note that collection is set to False, and only one header is loaded according to the values received in the parameters.



Next Monday Form Date	&NextScheduleDate	Weekly Schedule
Physician Id	&PhysicianId	Schedule One
Most Frecuent Shift Id	&MostFrecuentShiftId	
Usual Office Id	&usualOfficeId	

GRID

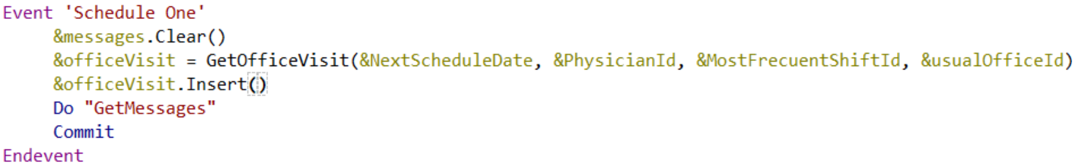
Id	Type	Description
&messages.item(0).Id	&messages.item(0).Type	&messages.item(0).Description



Variables

Standard Variables

FirstDayOfWeek	Date	□	First Day Of Week
message	Messages.Message, GeneXus.Common	□	message
messages	Messages, GeneXus.Common	□	messages
MostFrecuentShiftId	Attribute:ShiftId	□	Most Frecuent Shift Id
n	Numeric(1.0)	□	n
NextScheduleDate	Date	□	NextMondayDate
nextShift	Attribute:ShiftId	□	next Shift
officeVisit	OfficeVisit	□	office Visit
officeVisits	OfficeVisit	☑	office Visits
PhysicianId	Attribute:PhysicianId	□	Physician Id
usualOfficeId	Attribute:OfficeId	□	Office Id

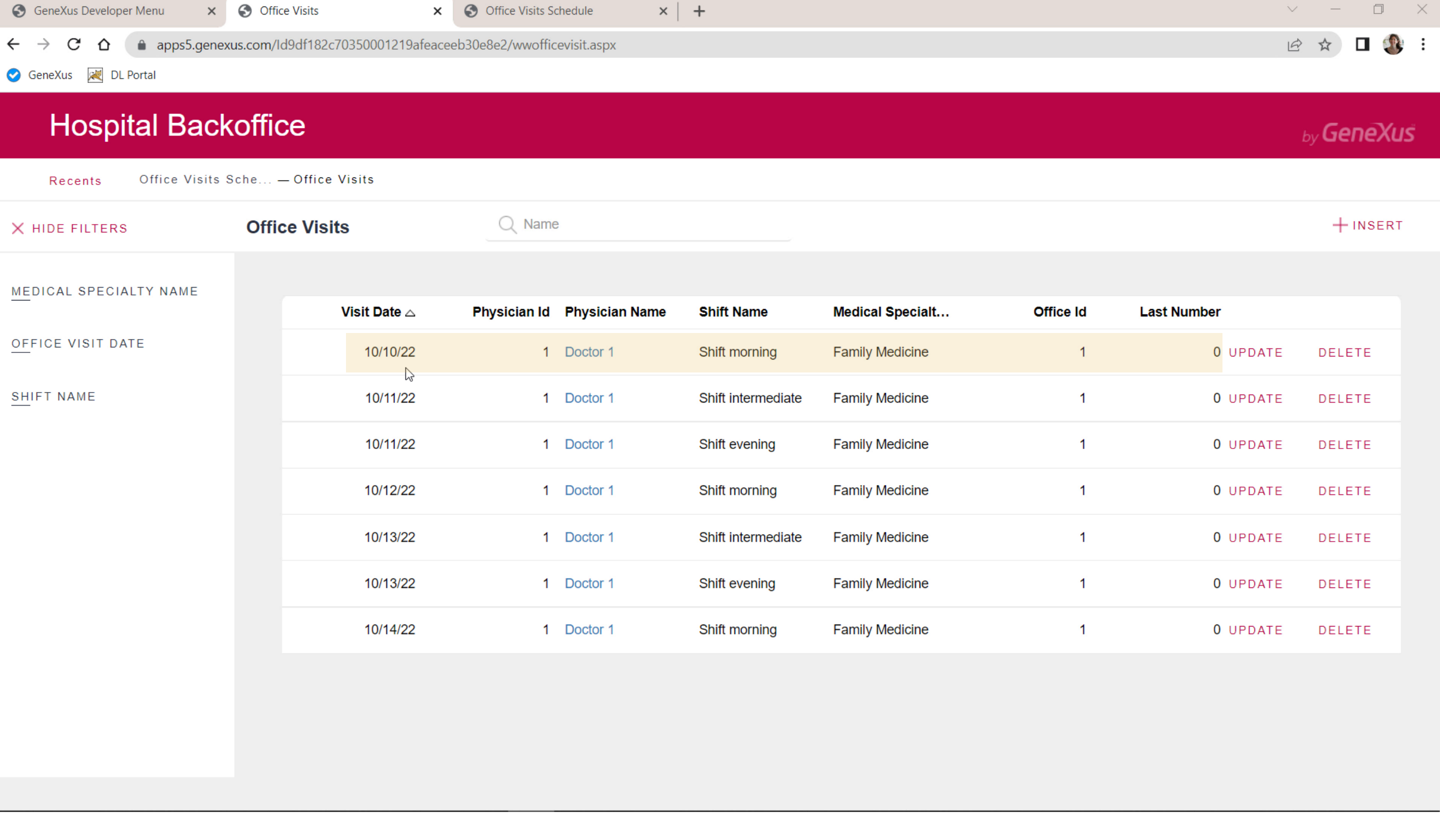


```
Event 'Schedule One'
	&messages.Clear()
	&officeVisit = GetOfficeVisit(&NextScheduleDate, &PhysicianId, &MostFrecuentShiftId, &usualOfficeId)
	&officeVisit.Insert()
	Do "GetMessages"
	Commit
Endevent
```

Let's add a button that allows inserting a single office visit with the screen variables, with those values. Then we use the non-collection BC variable, &officeVisit, that we already have declared... and load it with what is returned by this Data Provider, to which we pass the screen variables as parameters.

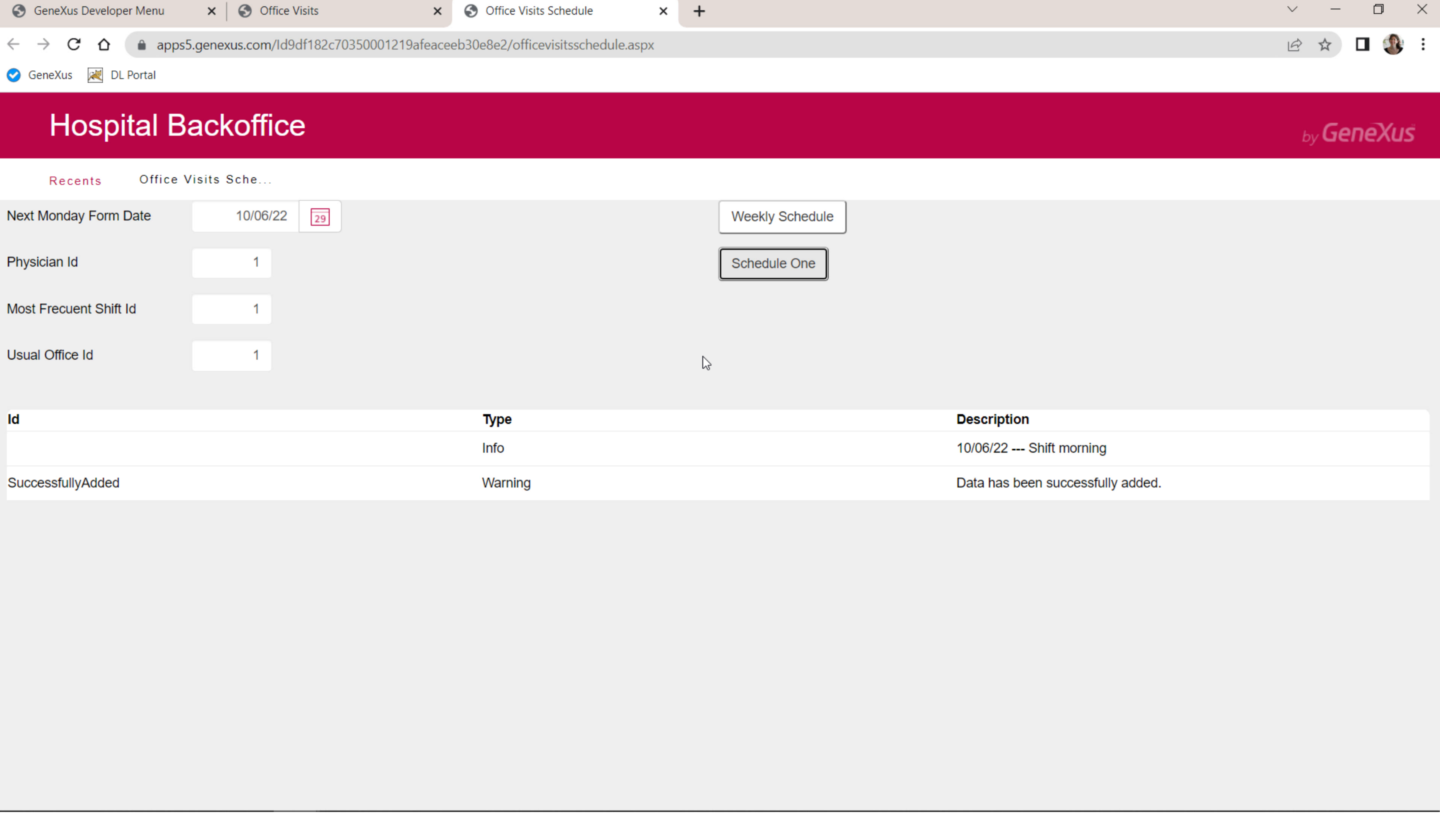
The variable is indeed of the BC data type, not collection.

Then we simply ask it to insert in the database and process the messages, adding the Commit, of course.



Visit Date	Physician Id	Physician Name	Shift Name	Medical Specialt...	Office Id	Last Number		
10/10/22	1	Doctor 1	Shift morning	Family Medicine	1	0	UPDATE	DELETE
10/11/22	1	Doctor 1	Shift intermediate	Family Medicine	1	0	UPDATE	DELETE
10/11/22	1	Doctor 1	Shift evening	Family Medicine	1	0	UPDATE	DELETE
10/12/22	1	Doctor 1	Shift morning	Family Medicine	1	0	UPDATE	DELETE
10/13/22	1	Doctor 1	Shift intermediate	Family Medicine	1	0	UPDATE	DELETE
10/13/22	1	Doctor 1	Shift evening	Family Medicine	1	0	UPDATE	DELETE
10/14/22	1	Doctor 1	Shift morning	Family Medicine	1	0	UPDATE	DELETE

Let’s try it. For example, for a date prior to this first one…



Next Monday Form Date 10/06/22

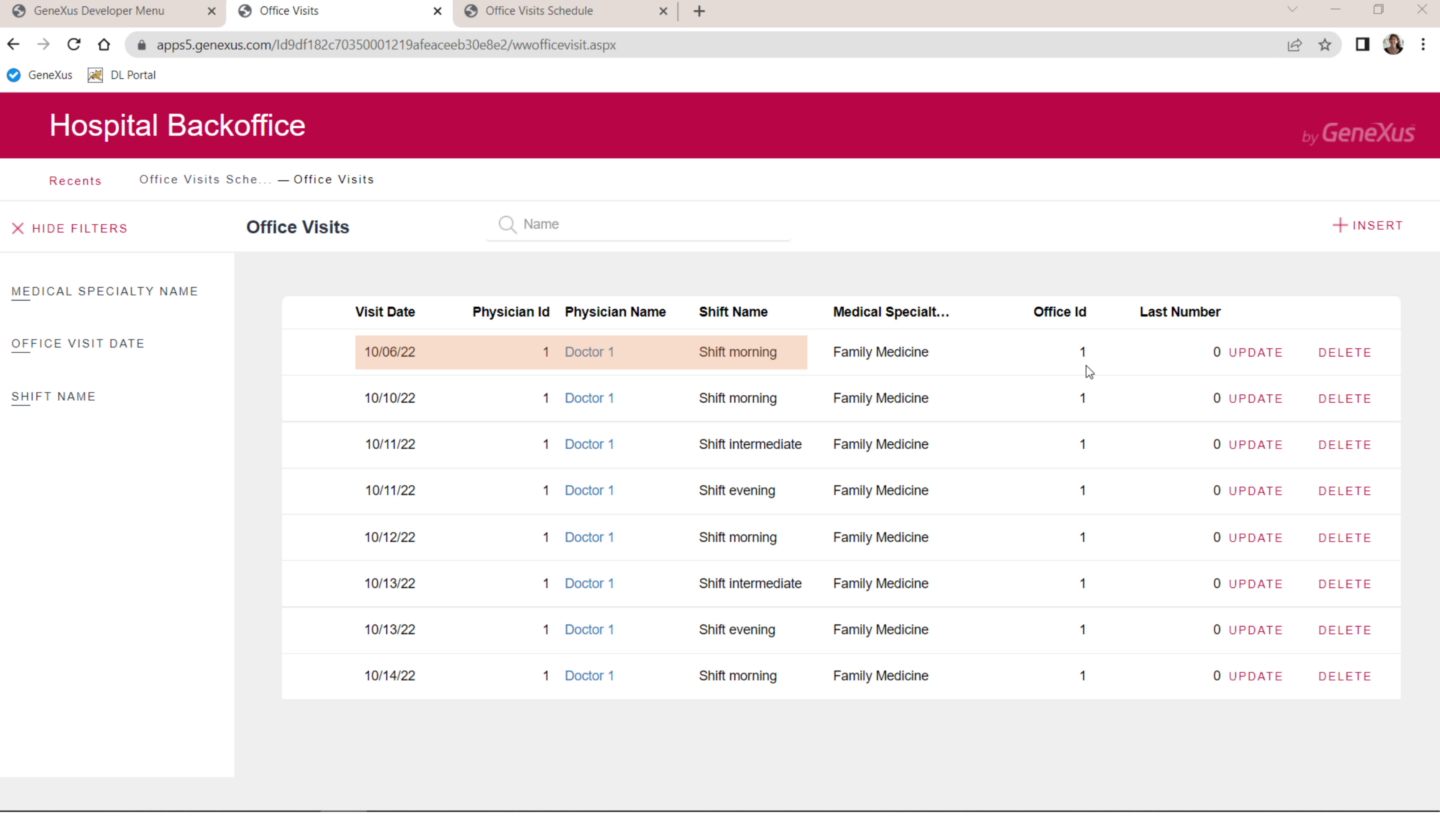
Physician Id 1

Most Frecuent Shift Id 1

Usual Office Id 1

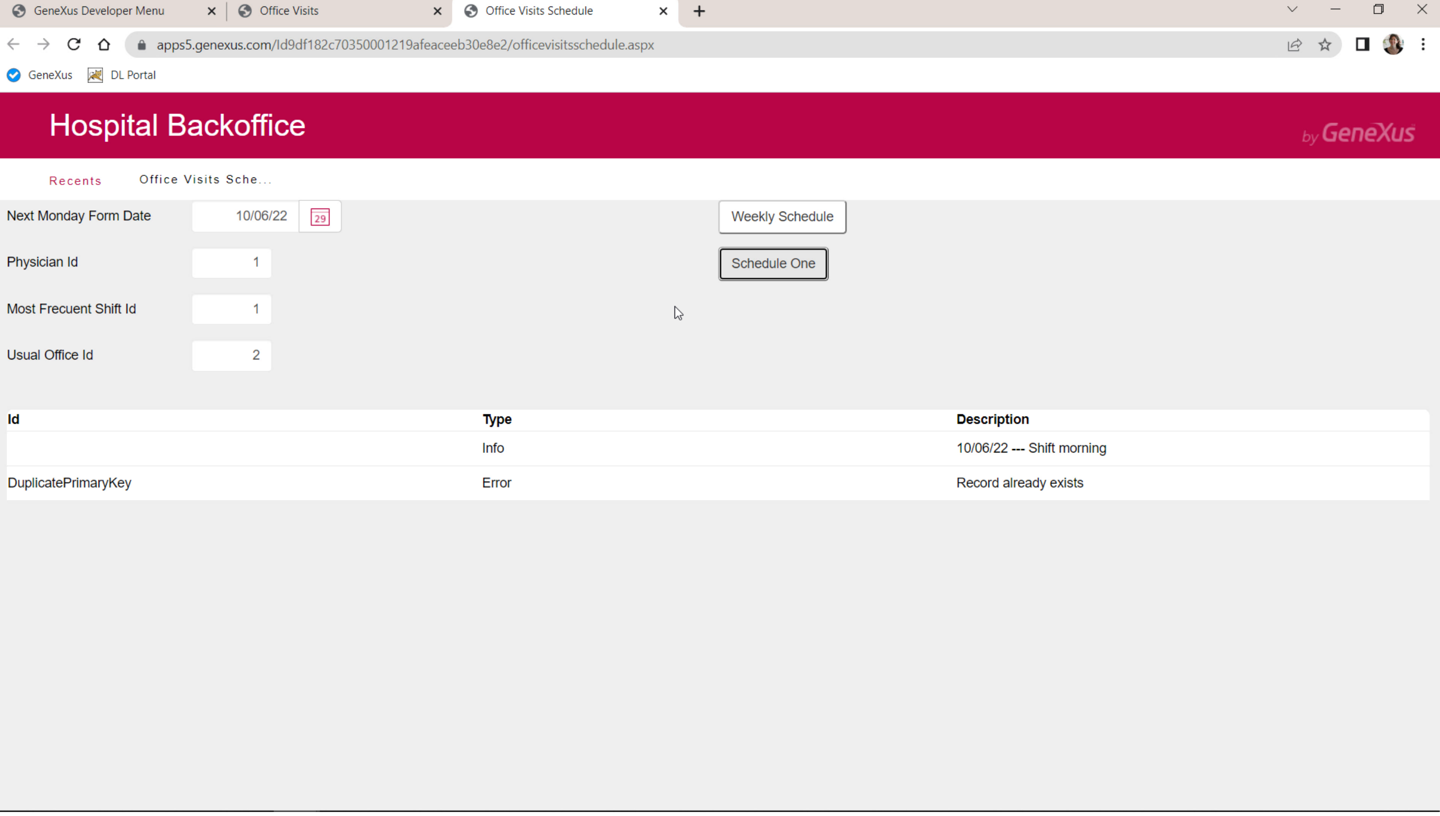
Id	Type	Description
	Info	10/06/22 --- Shift morning
SuccessfullyAdded	Warning	Data has been successfully added.

...we insert successfully.



Visit Date	Physician Id	Physician Name	Shift Name	Medical Specialt...	Office Id	Last Number		
10/06/22	1	Doctor 1	Shift morning	Family Medicine	1	0	UPDATE	DELETE
10/10/22	1	Doctor 1	Shift morning	Family Medicine	1	0	UPDATE	DELETE
10/11/22	1	Doctor 1	Shift intermediate	Family Medicine	1	0	UPDATE	DELETE
10/11/22	1	Doctor 1	Shift evening	Family Medicine	1	0	UPDATE	DELETE
10/12/22	1	Doctor 1	Shift morning	Family Medicine	1	0	UPDATE	DELETE
10/13/22	1	Doctor 1	Shift intermediate	Family Medicine	1	0	UPDATE	DELETE
10/13/22	1	Doctor 1	Shift evening	Family Medicine	1	0	UPDATE	DELETE
10/14/22	1	Doctor 1	Shift morning	Family Medicine	1	0	UPDATE	DELETE

Now, suppose that in case the record exists we would like to update it; for example, we place the same primary key, but we want to change the office.



Next Monday Form Date: 10/06/22

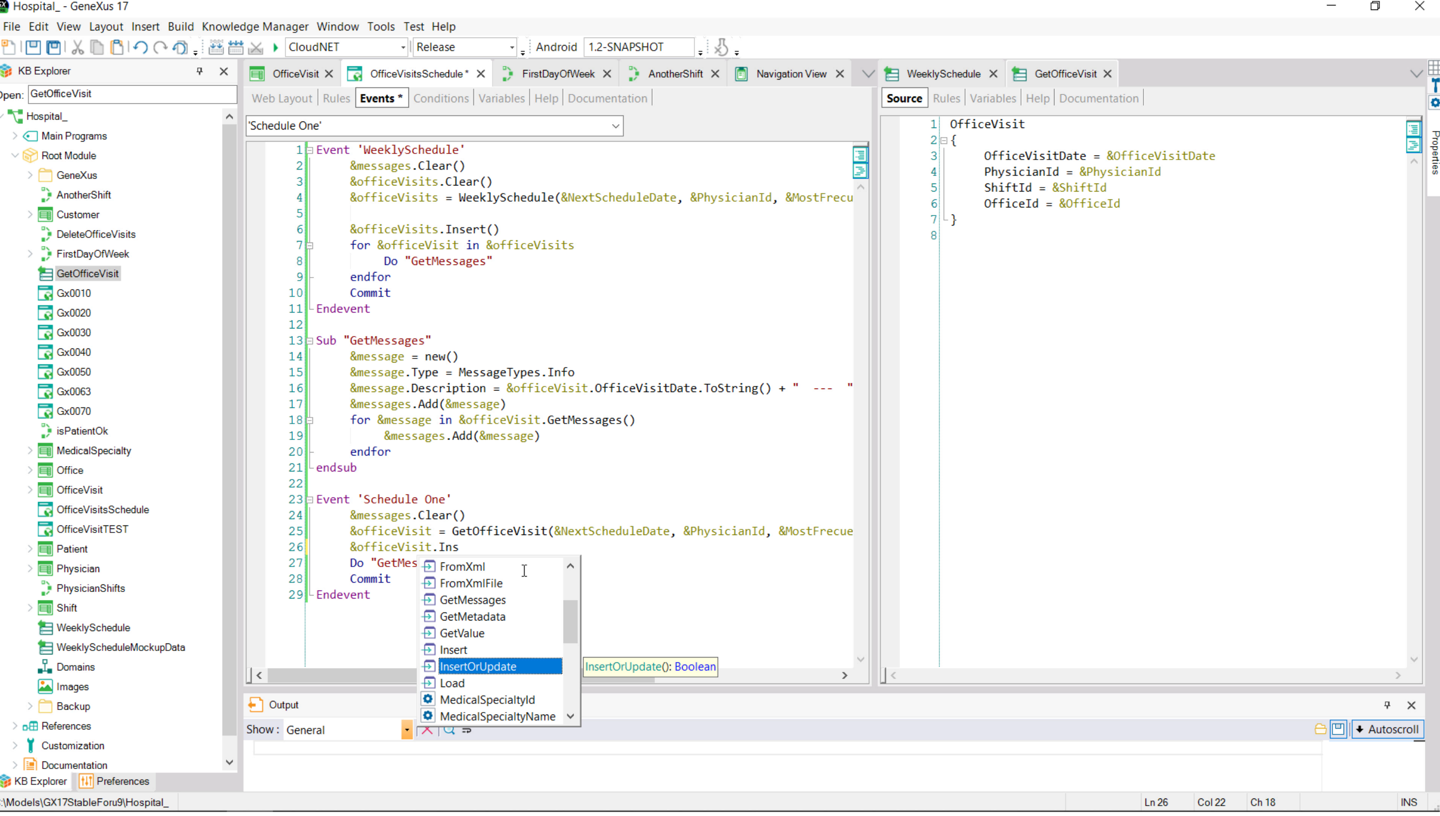
Physician Id: 1

Most Frecuent Shift Id: 1

Usual Office Id: 2

Id	Type	Description
	Info	10/06/22 --- Shift morning
DuplicatePrimaryKey	Error	Record already exists

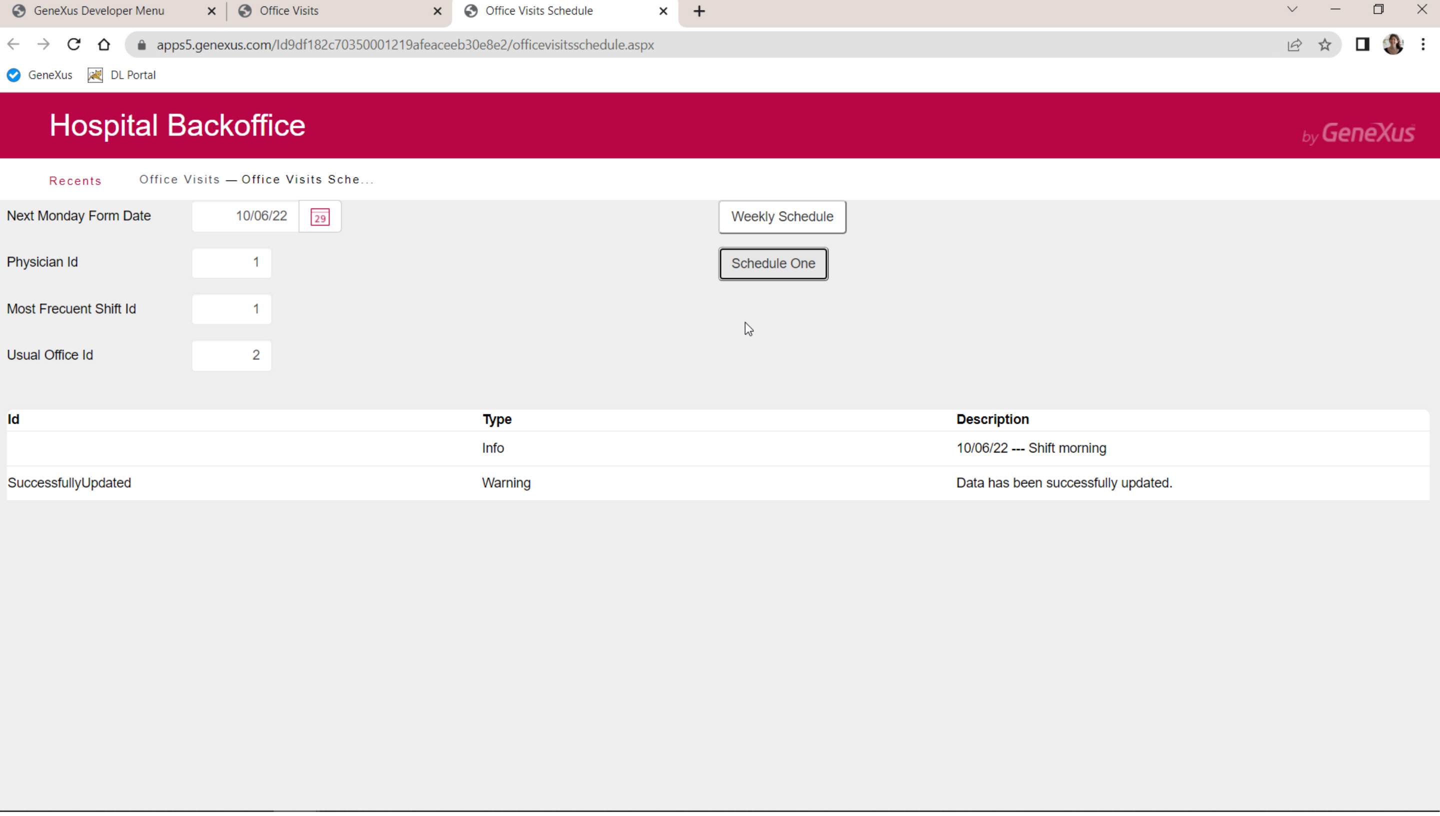
Of course, a duplicate primary key error will occur, because we tried an Insert and the record already existed.



```
Event 'WeeklySchedule'
     &messages.Clear()
     &officeVisits.Clear()
     &officeVisits = WeeklySchedule(&NextScheduleDate, &PhysicianId, &MostFrecu

     &officeVisits.Insert()
     for &officeVisit in &officeVisits
          Do "GetMessages"
     endfor
     Commit
Endevent

Sub "GetMessages"
     &message = new()
     &message.Type = MessageTypes.Info
     &message.Description = &officeVisit.OfficeVisitDate.ToString() + "  ---  "
     &messages.Add(&message)
     for &message in &officeVisit.GetMessages()
          &messages.Add(&message)
     endfor
endsub

Event 'Schedule One'
     &messages.Clear()
     &officeVisit = GetOfficeVisit(&NextScheduleDate, &PhysicianId, &MostFrecue
     &officeVisit.Ins
     Do "GetMes
     Commit
Endevent
```

```
OfficeVisit
{
     OfficeVisitDate = &OfficeVisitDate
     PhysicianId = &PhysicianId
     ShiftId = &ShiftId
     OfficeId = &OfficeId
}
```

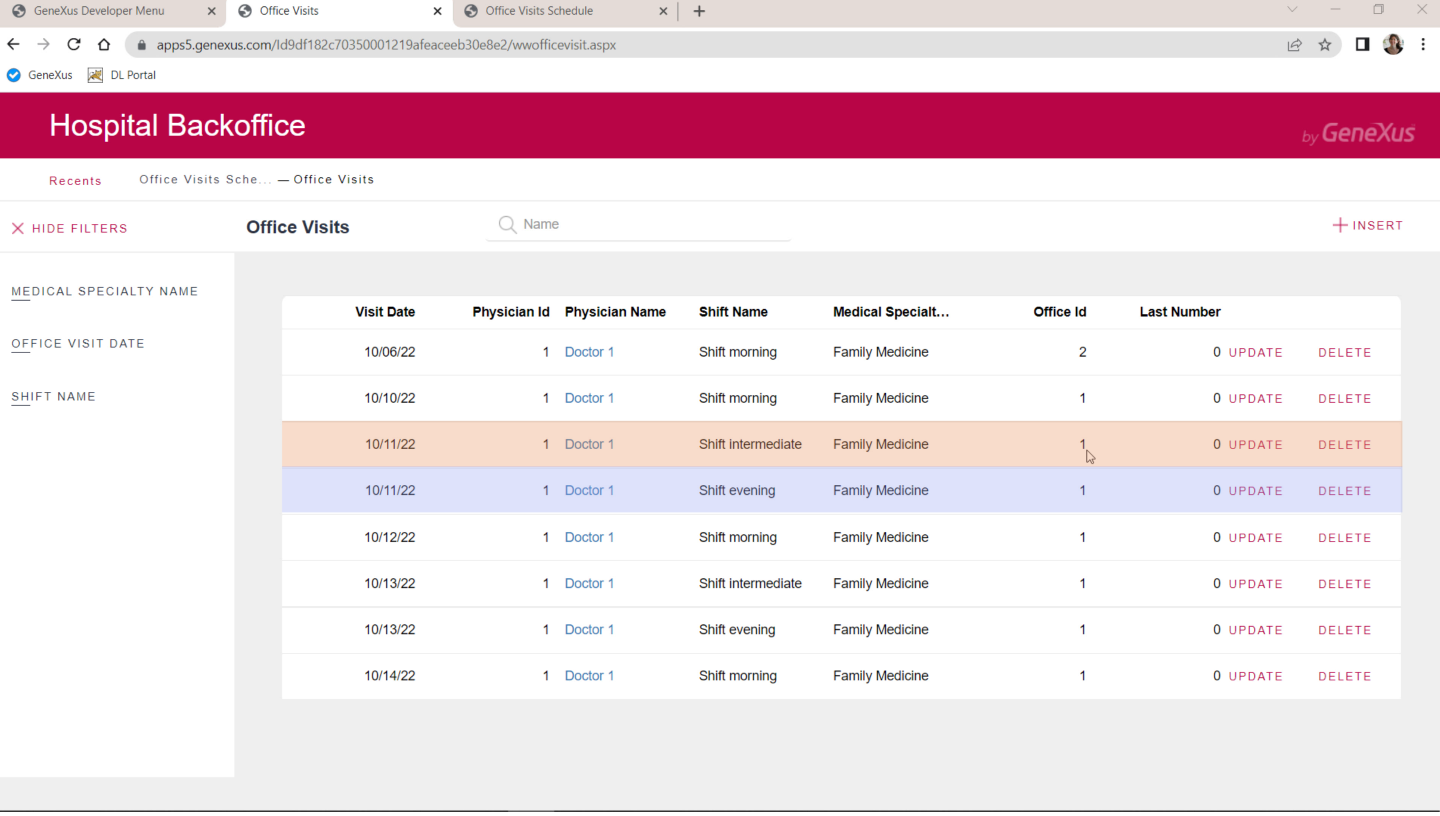
We can think, then, that to solve this we can just modify the method that we are invoking, changing it to InsertOrUpdate, so that it first tries to insert, and if it fails due to a duplicate key, then it tries to update.



Next Monday Form Date	10/06/22
Physician Id	1
Most Frecuent Shift Id	1
Usual Office Id	2

Id	Type	Description
	Info	10/06/22 --- Shift morning
SuccessfullyUpdated	Warning	Data has been successfully updated.

If now we try again to change the office of this doctor’s visit to number 2... we see that indeed it did what we wanted.



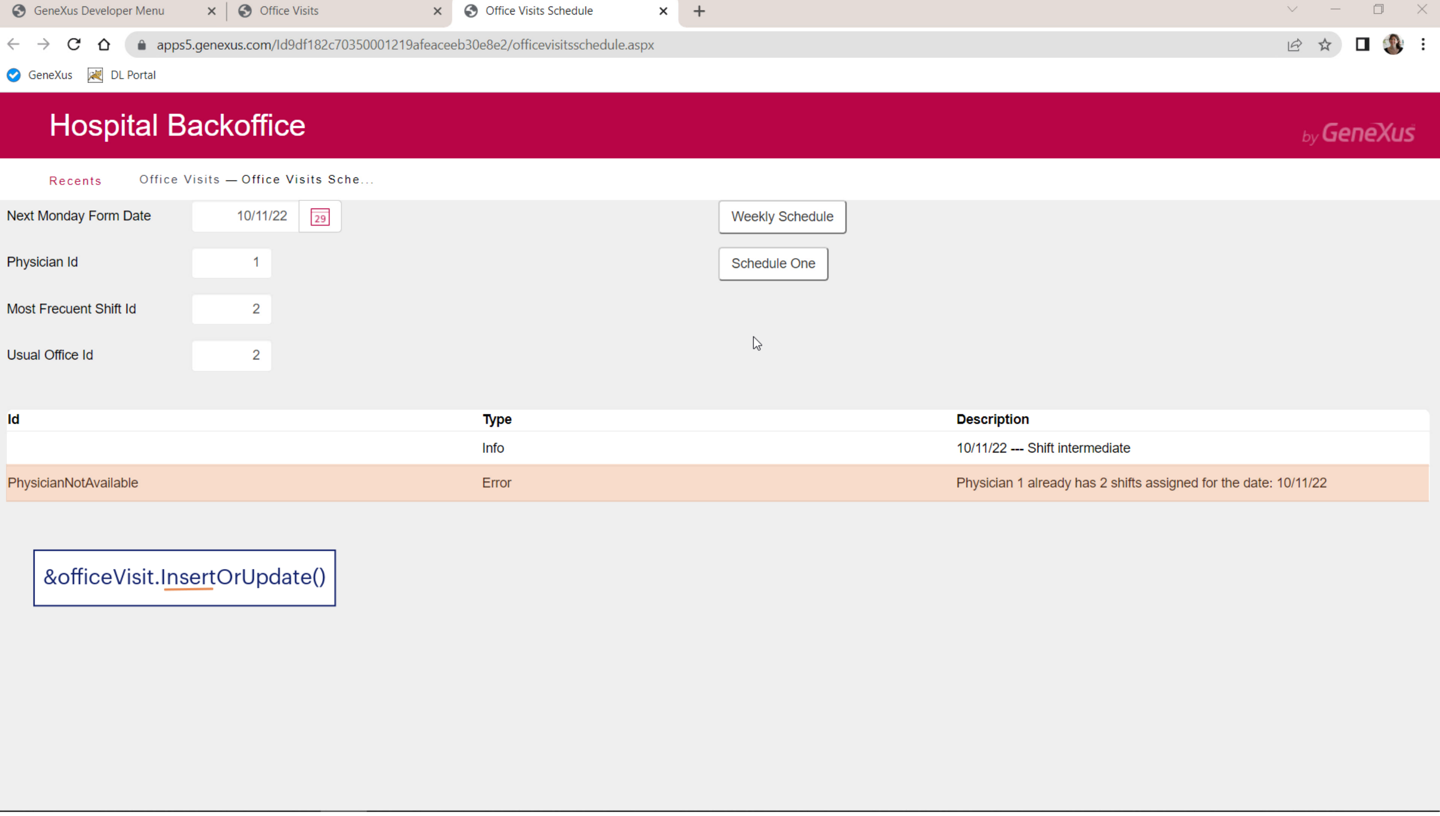
Hospital Backoffice

Recents   Office Visits Sche... — Office Visits

Office Visits

Visit Date	Physician Id	Physician Name	Shift Name	Medical Specialt...	Office Id	Last Number		
10/06/22	1	Doctor 1	Shift morning	Family Medicine	2	0	UPDATE	DELETE
10/10/22	1	Doctor 1	Shift morning	Family Medicine	1	0	UPDATE	DELETE
10/11/22	1	Doctor 1	Shift intermediate	Family Medicine	1	0	UPDATE	DELETE
10/11/22	1	Doctor 1	Shift evening	Family Medicine	1	0	UPDATE	DELETE
10/12/22	1	Doctor 1	Shift morning	Family Medicine	1	0	UPDATE	DELETE
10/13/22	1	Doctor 1	Shift intermediate	Family Medicine	1	0	UPDATE	DELETE
10/13/22	1	Doctor 1	Shift evening	Family Medicine	1	0	UPDATE	DELETE
10/14/22	1	Doctor 1	Shift morning	Family Medicine	1	0	UPDATE	DELETE

However, let's see what happens if we want to do the same but this time, for example, for this office visit, for this day in the intermediate shift, and doctor 1, who has another visit for the same day, in another shift...



Next Monday Form Date	10/11/22
Physician Id	1
Most Frecuent Shift Id	2
Usual Office Id	2

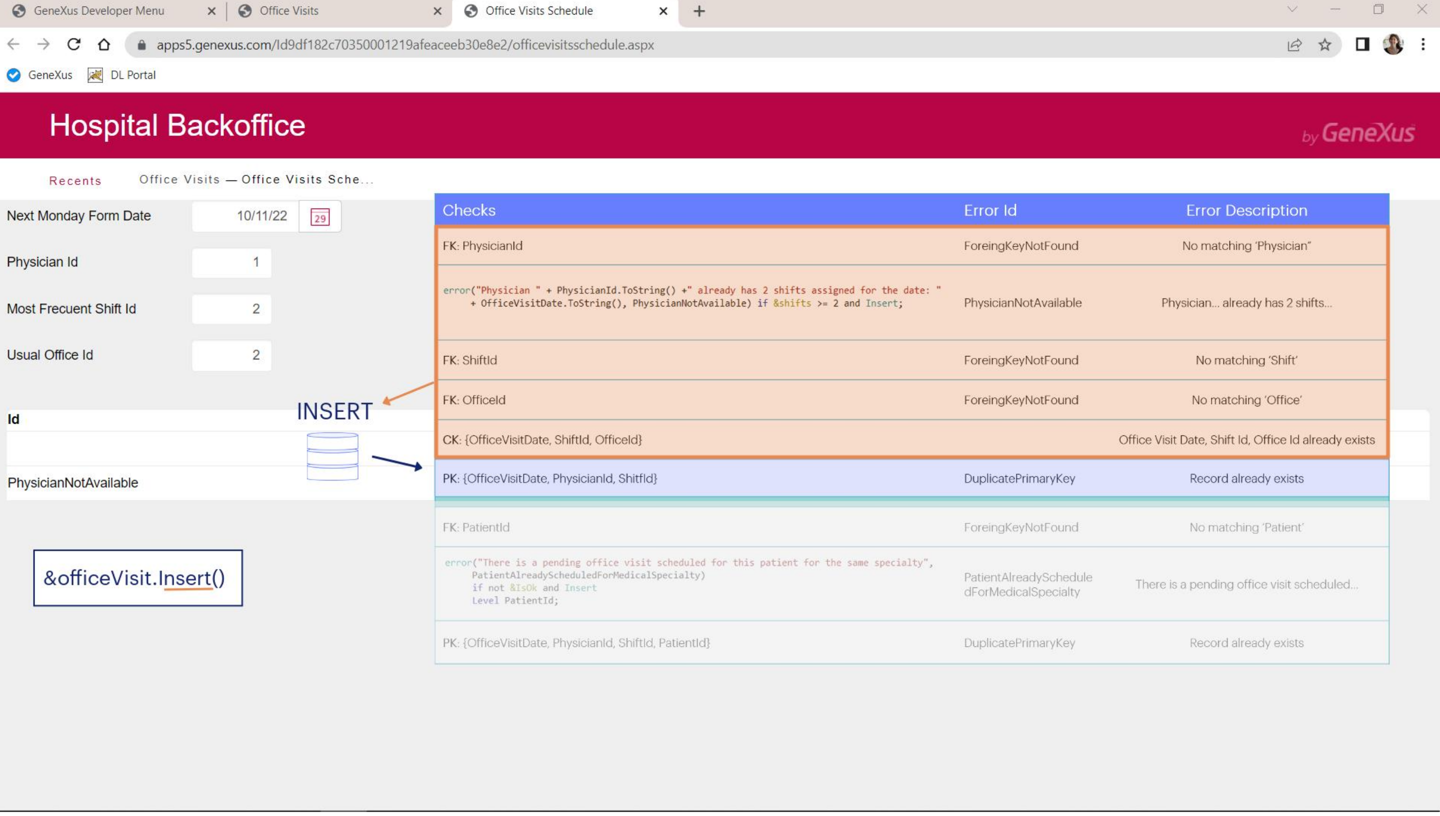
Id	Type	Description
	Info	10/11/22 --- Shift intermediate
PhysicianNotAvailable	Error	Physician 1 already has 2 shifts assigned for the date: 10/11/22

```
&officeVisit.InsertOrUpdate()
```

Then we place that day, doctor 1, shift 2 which is the intermediate shift and we want to change the office from 1 to 2.

Oops, why did we get this error indicating that doctor 1 already has 2 office visits assigned for that date?

It's because the Insert is tried first, and that's where this error is going to occur.


GeneXus Developer Menu
Office Visits
Office Visits Schedule
apps5.genexus.com/Id9df182c70350001219afeaceeb30e8e2/officevisitsschedule.aspx
GeneXus
DL Portal
Hospital Backoffice
by GeneXus
Recents
Office Visits — Office Visits Sche...
INSERT

Next Monday Form Date	10/11/22
Physician Id	1
Most Frecuent Shift Id	2
Usual Office Id	2

Id

PhysicianNotAvailable

&officeVisit.Insert()

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, ShitfId}	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;`
	PatientAlreadyScheduledForMedicalSpecialty	There is a pending office visit scheduled...

PK: {OfficeVisitDate, PhysicianId, ShiftId, PatientId}	DuplicatePrimaryKey	Record already exists

Let's remember the order in which the checks are triggered: before checking the uniqueness of the primary key, it will first trigger the procedure that will calculate the number of shifts for that doctor, and if the result is 2 or more, the error rule will be triggered.

What proves interesting here is that the uniqueness of the record will never be checked. Why?
And there is something very important that should be known: due to performance reasons, and assuming an optimistic strategy, in most cases, the insertion is not supposed to fail due to dupe PK. Therefore, all checks will take place first, and then the database INSERT command will be sent to the DBMS. This means that the INSERT is sent only when these previous checks were successful. And upon trying the Insert, the DBMS may return the dupe key error. And only then we will know whether another record with that key already existed or not.
So, if any of these previous checks fails, for example, due to this error rule, there is no attempt for the Insert, and we cannot know, **either**, whether a record with that key already existed that would also cause the insertion to fail.

In this case, the Update is not attempted, because it is only executed if the Insert failed because of a duplicate key.

InsertOrUpdate()

&officeVisit.**InsertOrUpdate**()

```
If not &officeVisit.Insert()
   for &message in &officeVisit.GetMessages()
            if &message.Id = “DuplicatePrimaryKey”
               &officeVisit.Update()
            endif
 endfor
```

UpdateOrInsert()

If we translate the logic of the InsertOrUpdate method to what happens in the background, it would be something like this:
The Insert is tried, and if it works, it continues with the code that comes after. If it doesn't work, only if it is caused by a duplicate primary key, the Update is tried, which may or may not work, of course.

But we have to bear in mind the behavior we have just described. That is, if the Insert fails due to something else prior to the actual attempt for insertion, since it is never known whether the record already existed, then it will never execute the Update.

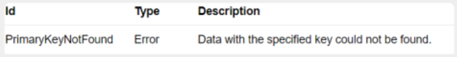
The reverse method has not yet been implemented in GeneXus.

UpdateOrInsert()

```
&officeVisit.UpdateOrInsert()
```

```
If not &officeVisit.Update()
   &officeVisit.Insert()
endif
```

```
If not &officeVisit.Update()
   for &message in &officeVisit.GetMessages()
           if &message.Id = “PrimaryKeyNotFound”
              &officeVisit.Insert()
           endif
 endfor
```

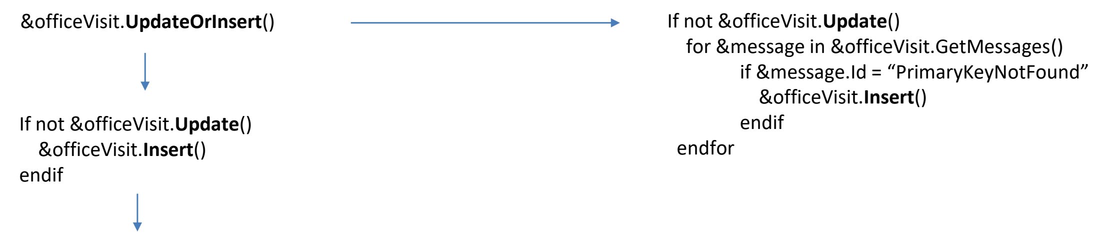


Id	Type	Description
PrimaryKeyNotFound	Error	Data with the specified key could not be found.

But it should be analogous, in the sense that the Insert should be tried only if the Update cannot be done because the record does not exist and not because of anything else.

Since we don't have this method yet, we can try the Update and if it fails for whatever reason, try the Insert anyway, knowing that if the failure was not because the key did not exist, then the Insert will inevitably fail.

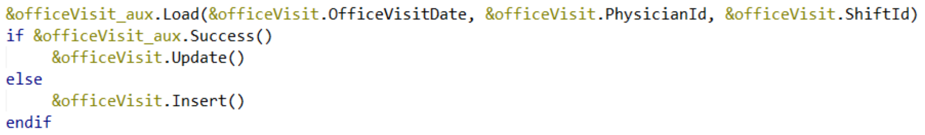
UpdateOrInsert()



```
&officeVisit.UpdateOrInsert()
```

```
If not &officeVisit.Update()
   &officeVisit.Insert()
endif
```

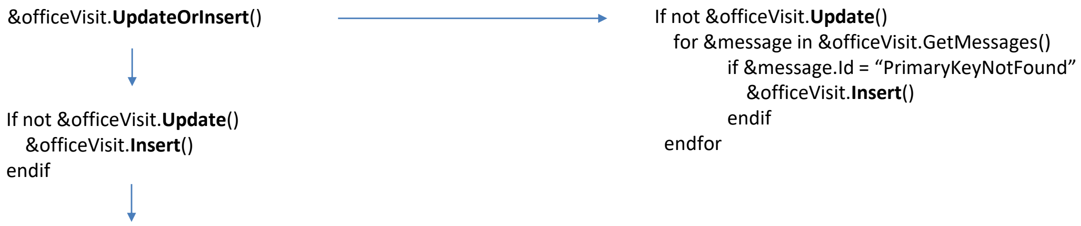
```
If not &officeVisit.Update()
   for &message in &officeVisit.GetMessages()
            if &message.Id = “PrimaryKeyNotFound”
               &officeVisit.Insert()
            endif
 endfor
```



```
&officeVisit_aux.Load(&officeVisit.OfficeVisitDate, &officeVisit.PhysicianId, &officeVisit.ShiftId)
if &officeVisit_aux.Success()
	&officeVisit.Update()
else
	&officeVisit.Insert()
endif
```

Of course, we could also be more specific and try the Load method, and ask if it was successful or not, and then order an Update or an Insert, respectively.

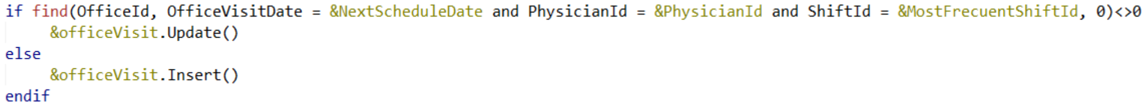
UpdateOrInsert()



```
&officeVisit.UpdateOrInsert()
```

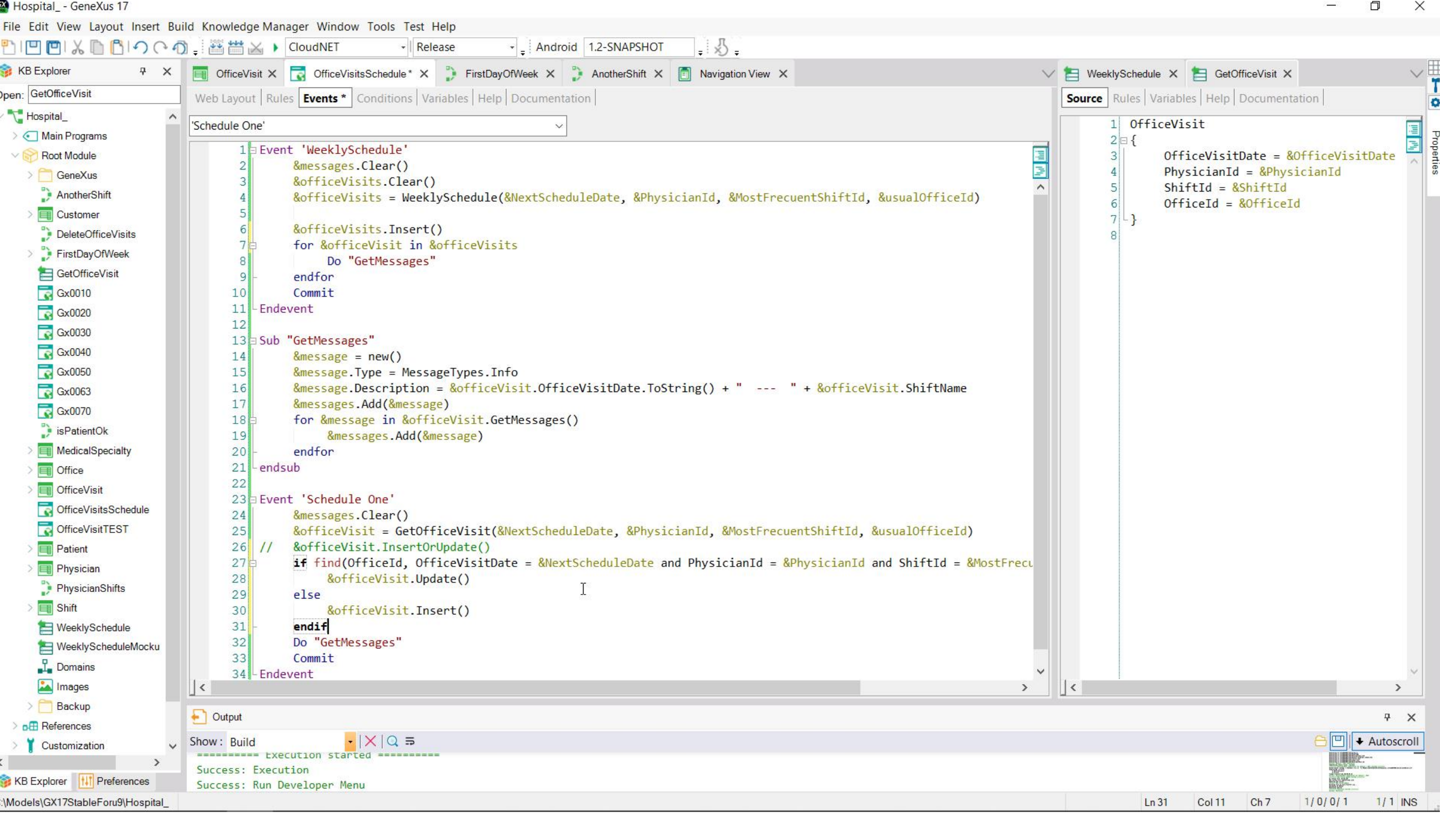
```
If not &officeVisit.Update()
   &officeVisit.Insert()
endif
```

```
If not &officeVisit.Update()
   for &message in &officeVisit.GetMessages()
            if &message.Id = “PrimaryKeyNotFound”
               &officeVisit.Insert()
            endif
 endfor
```



```
if find(OfficeId, OfficeVisitDate = &NextScheduleDate and PhysicianId = &PhysicianId and ShiftId = &MostFrecuentShiftId, 0)<>0
       &officeVisit.Update()
else
       &officeVisit.Insert()
endif
```

If we want to be more efficient, we can search for the existence of the record with a Find formula, for example.

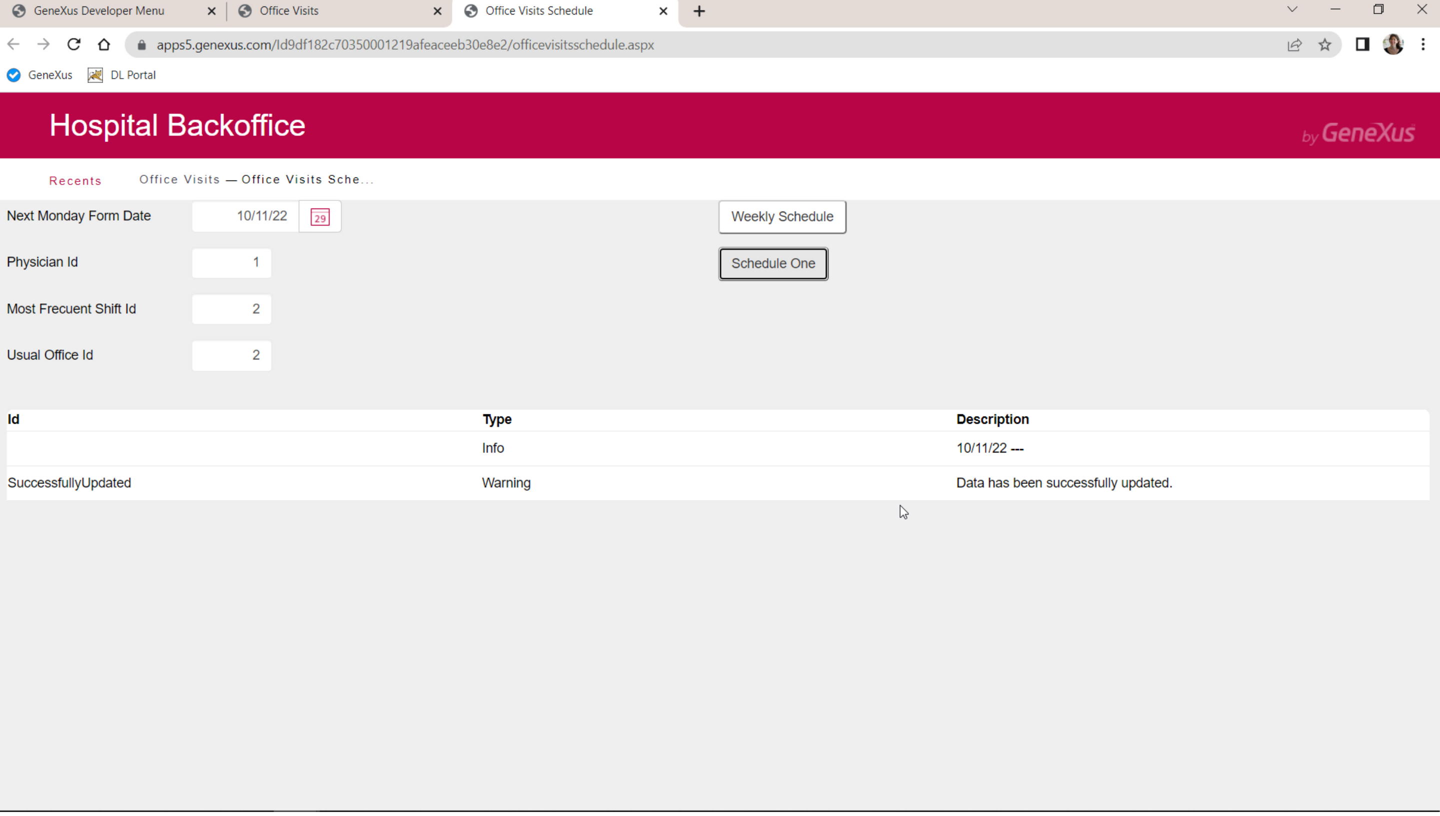


```
Event 'WeeklySchedule'
	&messages.Clear()
	&officeVisits.Clear()
	&officeVisits = WeeklySchedule(&NextScheduleDate, &PhysicianId, &MostFrecuentShiftId, &usualOfficeId)

	&officeVisits.Insert()
	for &officeVisit in &officeVisits
		Do "GetMessages"
	endfor
	Commit
Endevent

Sub "GetMessages"
	&message = new()
	&message.Type = MessageTypes.Info
	&message.Description = &officeVisit.OfficeVisitDate.ToString() + "  ---  " + &officeVisit.ShiftName
	&messages.Add(&message)
	for &message in &officeVisit.GetMessages()
		&messages.Add(&message)
	endfor
endsub

Event 'Schedule One'
	&messages.Clear()
	&officeVisit = GetOfficeVisit(&NextScheduleDate, &PhysicianId, &MostFrecuentShiftId, &usualOfficeId)
//	&officeVisit.InsertOrUpdate()
	if find(OfficeId, OfficeVisitDate = &NextScheduleDate and PhysicianId = &PhysicianId and ShiftId = &MostFrecu
		&officeVisit.Update()
	else
		&officeVisit.Insert()
	endif
	Do "GetMessages"
	Commit
Endevent
```

```
OfficeVisit
{
	OfficeVisitDate = &OfficeVisitDate
	PhysicianId = &PhysicianId
	ShiftId = &ShiftId
	OfficeId = &OfficeId
}
```

```
========== Execution started ==========
Success: Execution
Success: Run Developer Menu
```

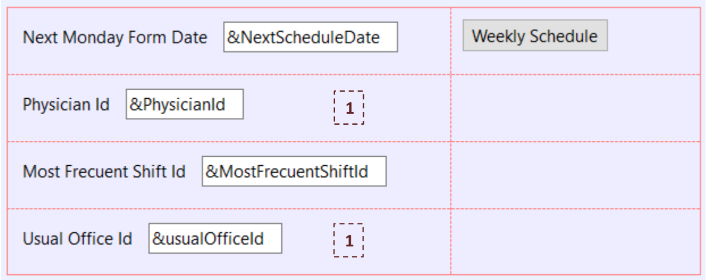
Let's try this solution.



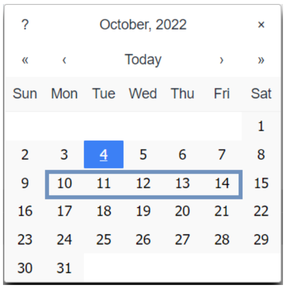
Next Monday Form Date	10/11/22
Physician Id	1
Most Frecuent Shift Id	2
Usual Office Id	2

Id	Type	Description
	Info	10/11/22 ---
SuccessfullyUpdated	Warning	Data has been successfully updated.

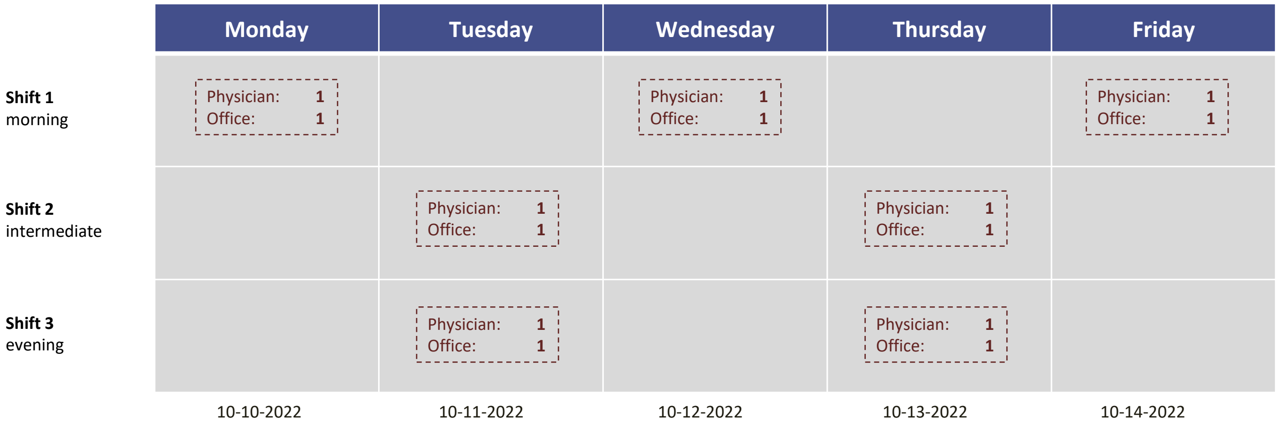
Now we were able to change the office.



Next Monday Form Date	&NextScheduleDate	Weekly Schedule
Physician Id	&PhysicianId 1	
Most Frecuent Shift Id	&MostFrecuentShiftId	
Usual Office Id	&usualOfficeId 1	

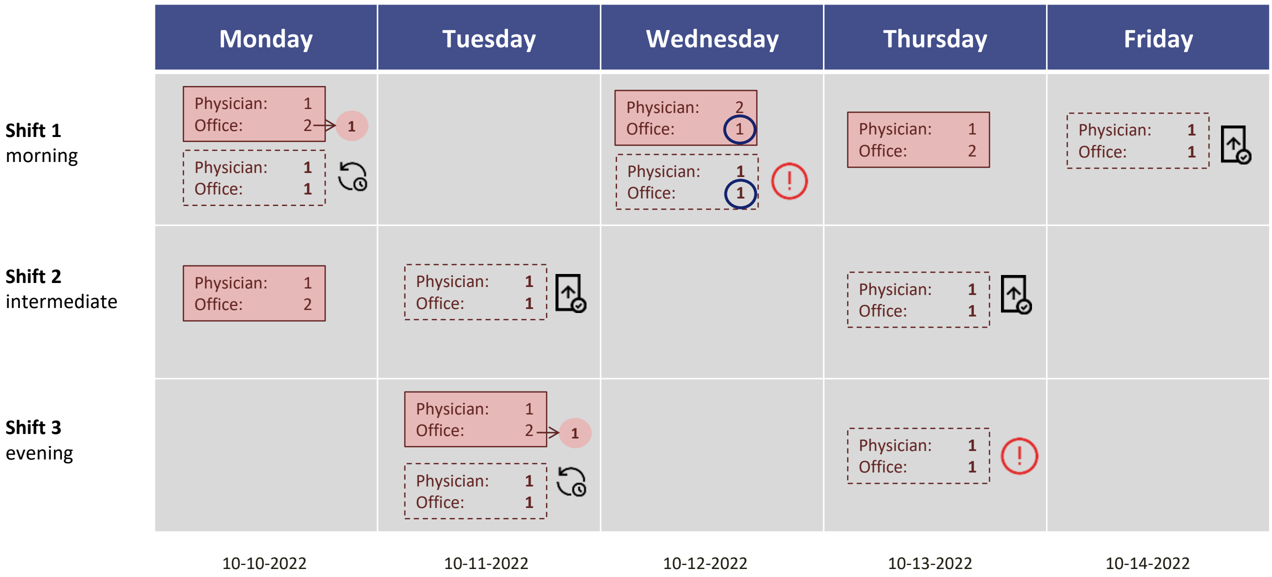


Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					



	Monday	Tuesday	Wednesday	Thursday	Friday
Shift 1 morning	Physician: 1 Office: 1		Physician: 1 Office: 1		Physician: 1 Office: 1
Shift 2 intermediate		Physician: 1 Office: 1		Physician: 1 Office: 1	
Shift 3 evening		Physician: 1 Office: 1		Physician: 1 Office: 1	
	10-10-2022	10-11-2022	10-12-2022	10-13-2022	10-14-2022

Let's go back to the batch insertion. Previously we had inserted all these records without any problems because there were no other records for that week in the database. But what if they existed?

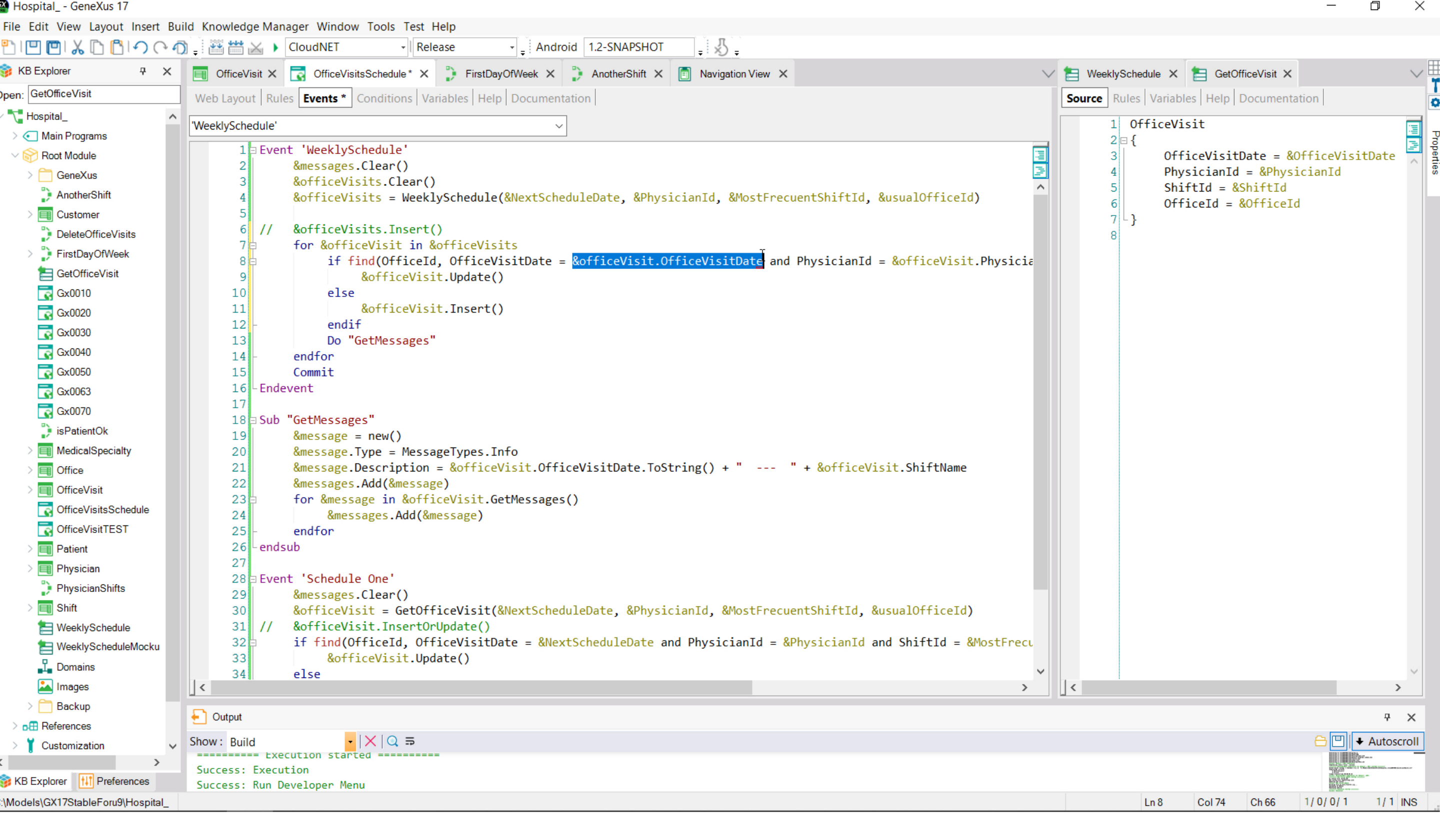


	Monday	Tuesday	Wednesday	Thursday	Friday
Shift 1 morning	Physician: 1 Office: 2 → 1 Physician: 1 Office: 1		Physician: 2 Office: 1 Physician: 1 Office: 1 !	Physician: 1 Office: 2	Physician: 1 Office: 1
Shift 2 intermediate	Physician: 1 Office: 2	Physician: 1 Office: 1		Physician: 1 Office: 1	
Shift 3 evening		Physician: 1 Office: 2 → 1 Physician: 1 Office: 1		Physician: 1 Office: 1 !	
	10-10-2022	10-11-2022	10-12-2022	10-13-2022	10-14-2022

For example, let's suppose that these records already existed for that week, when we want to launch the batch process.

We will want to do the same as in the previous individual case: that is, insert if there is no record with that key and otherwise update.

So, what should happen with each one? For the first one, there already exists a record with that primary key, this one, so it will try the Update, which will work, changing the office for this one, number 1.
For the second one, it checks if there is a record with that key, and since it doesn't exist, it tries the Insert, but it will fail because of a duplicate candidate key. The reason is that office 1 is already used.
For the third one, it checks if there is a record with that key; it does not exist, so it tries the Insert, which will be successful.
For the fourth one, it tries the Insert, which will count how many office visits are already entered for that day for the doctor, and since it will give 1, no error will be triggered and it will be possible to insert without any problem.
For the fifth one the same thing will happen, it will insert without any problem.
Now comes the sixth one and since there is already a record with that key, it tries the Update, which will be successful, modifying the office of the existing record.
For the last one, since there is no record with that key, it will try the Insert, but it will fail. Why? Because it will find that doctor 1 already has 2 office visits scheduled that day.



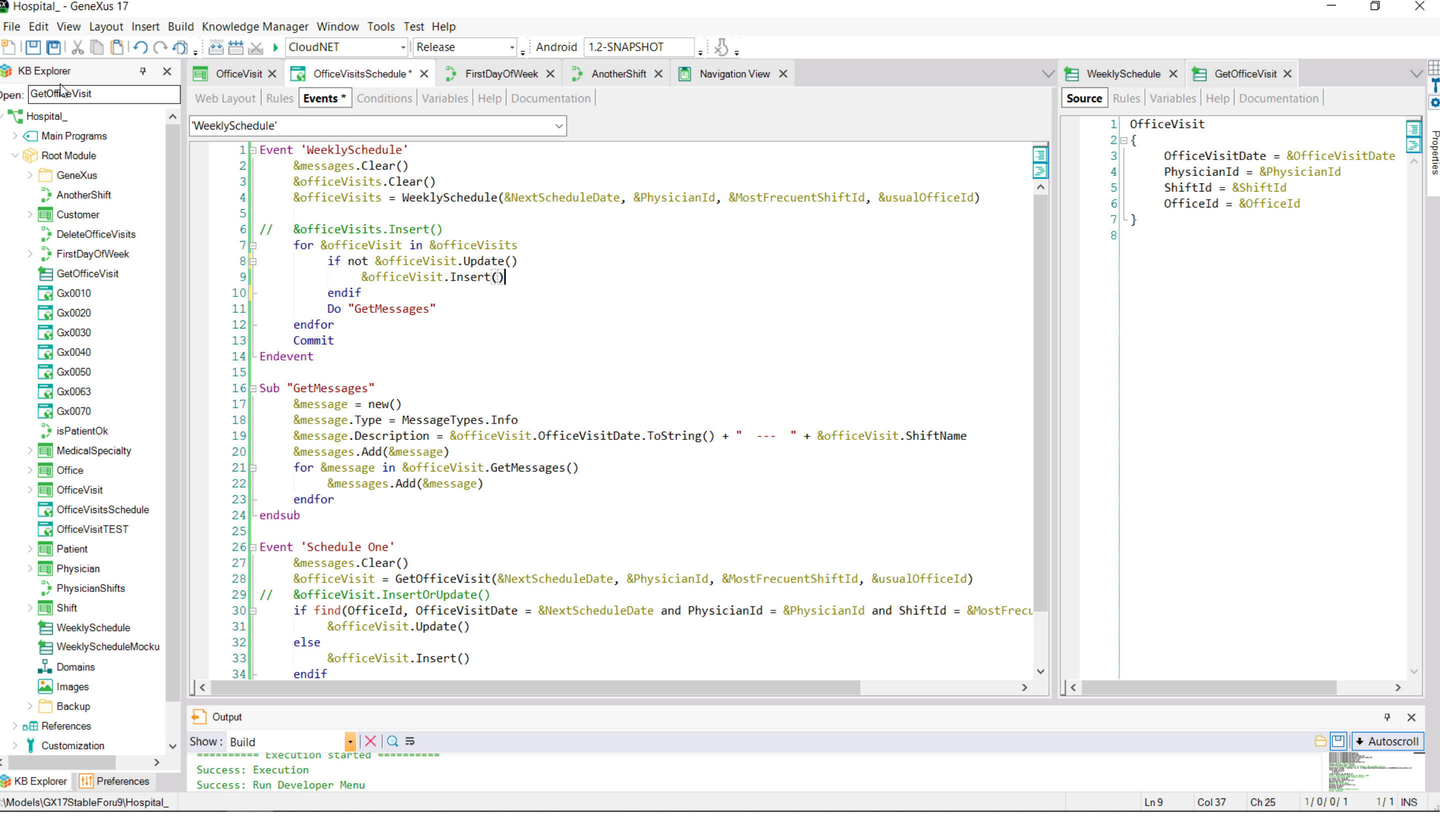
```
Event 'WeeklySchedule'
     &messages.Clear()
     &officeVisits.Clear()
     &officeVisits = WeeklySchedule(&NextScheduleDate, &PhysicianId, &MostFrecuentShiftId, &usualOfficeId)

//   &officeVisits.Insert()
     for &officeVisit in &officeVisits
          if find(OfficeId, OfficeVisitDate = &officeVisit.OfficeVisitDate and PhysicianId = &officeVisit.Physicia
               &officeVisit.Update()
          else
               &officeVisit.Insert()
          endif
          Do "GetMessages"
     endfor
     Commit
Endevent

Sub "GetMessages"
     &message = new()
     &message.Type = MessageTypes.Info
     &message.Description = &officeVisit.OfficeVisitDate.ToString() + "  ---  " + &officeVisit.ShiftName
     &messages.Add(&message)
     for &message in &officeVisit.GetMessages()
          &messages.Add(&message)
     endfor
endsub

Event 'Schedule One'
     &messages.Clear()
     &officeVisit = GetOfficeVisit(&NextScheduleDate, &PhysicianId, &MostFrecuentShiftId, &usualOfficeId)
//   &officeVisit.InsertOrUpdate()
     if find(OfficeId, OfficeVisitDate = &NextScheduleDate and PhysicianId = &PhysicianId and ShiftId = &MostFrecu
          &officeVisit.Update()
     else
```

```
OfficeVisit
{
     OfficeVisitDate = &OfficeVisitDate
     PhysicianId = &PhysicianId
     ShiftId = &ShiftId
     OfficeId = &OfficeId
}
```

```
========== Execution started ==========
Success: Execution
Success: Run Developer Menu
```

How do we change the code we had to make it work like this?

It is no longer useful to launch a massive operation on the collection of the 7 items returned by the Data Provider. We will have to work individually, running through the entire collection and in each case checking for the existence of the record with that key. Here we must use the data of each BC.



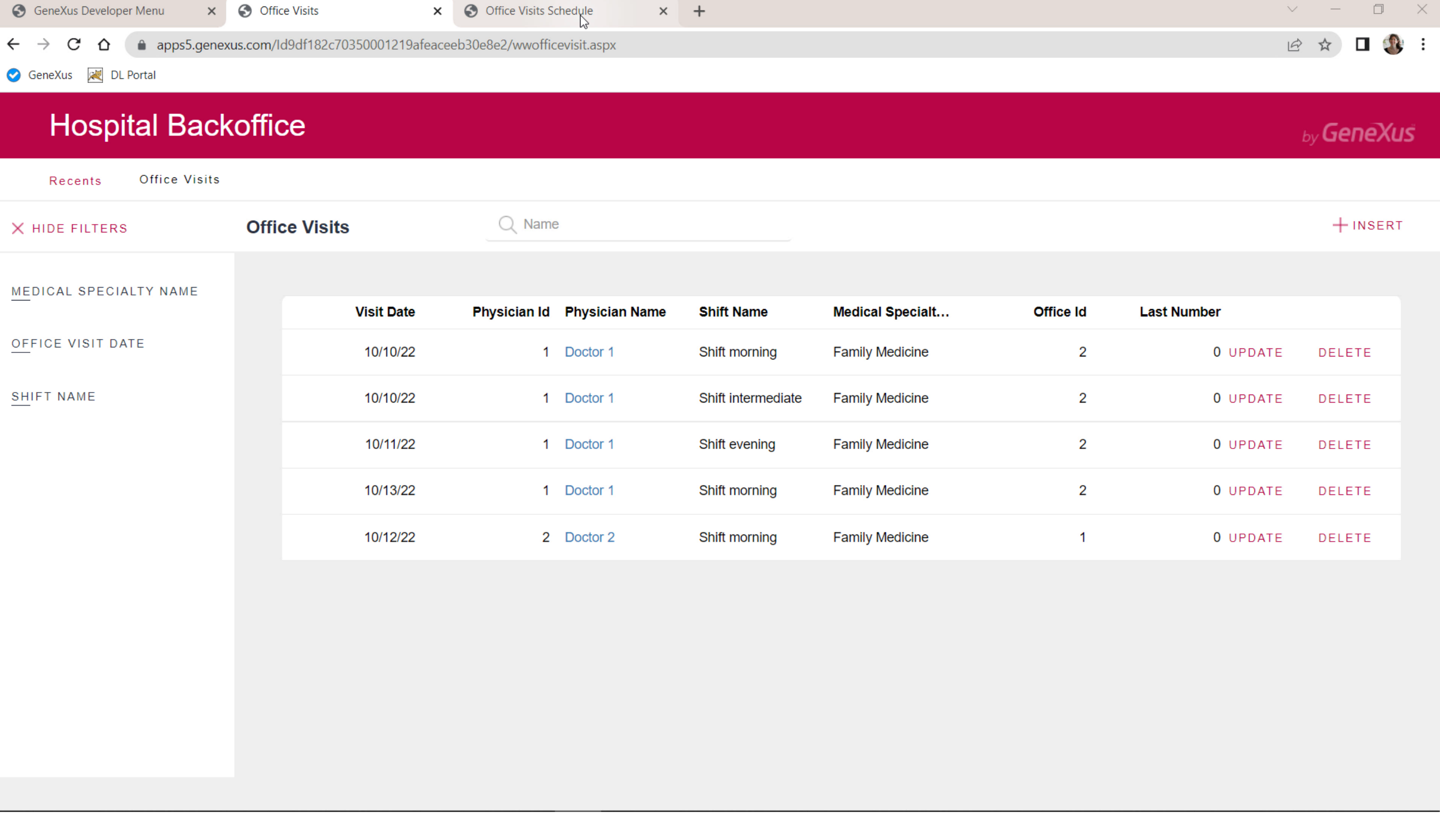
```
Event 'WeeklySchedule'
    &messages.Clear()
    &officeVisits.Clear()
    &officeVisits = WeeklySchedule(&NextScheduleDate, &PhysicianId, &MostFrecuentShiftId, &usualOfficeId)

//  &officeVisits.Insert()
    for &officeVisit in &officeVisits
        if not &officeVisit.Update()
            &officeVisit.Insert()
        endif
        Do "GetMessages"
    endfor
    Commit
Endevent

Sub "GetMessages"
    &message = new()
    &message.Type = MessageTypes.Info
    &message.Description = &officeVisit.OfficeVisitDate.ToString() + "  ---  " + &officeVisit.ShiftName
    &messages.Add(&message)
    for &message in &officeVisit.GetMessages()
        &messages.Add(&message)
    endfor
endsub

Event 'Schedule One'
    &messages.Clear()
    &officeVisit = GetOfficeVisit(&NextScheduleDate, &PhysicianId, &MostFrecuentShiftId, &usualOfficeId)
//  &officeVisit.InsertOrUpdate()
    if find(OfficeId, OfficeVisitDate = &NextScheduleDate and PhysicianId = &PhysicianId and ShiftId = &MostFrecu
        &officeVisit.Update()
    else
        &officeVisit.Insert()
    endif
```

```
OfficeVisit
{
    OfficeVisitDate = &OfficeVisitDate
    PhysicianId = &PhysicianId
    ShiftId = &ShiftId
    OfficeId = &OfficeId
}
```

```
========== Execution started ==========
Success: Execution
Success: Run Developer Menu
```

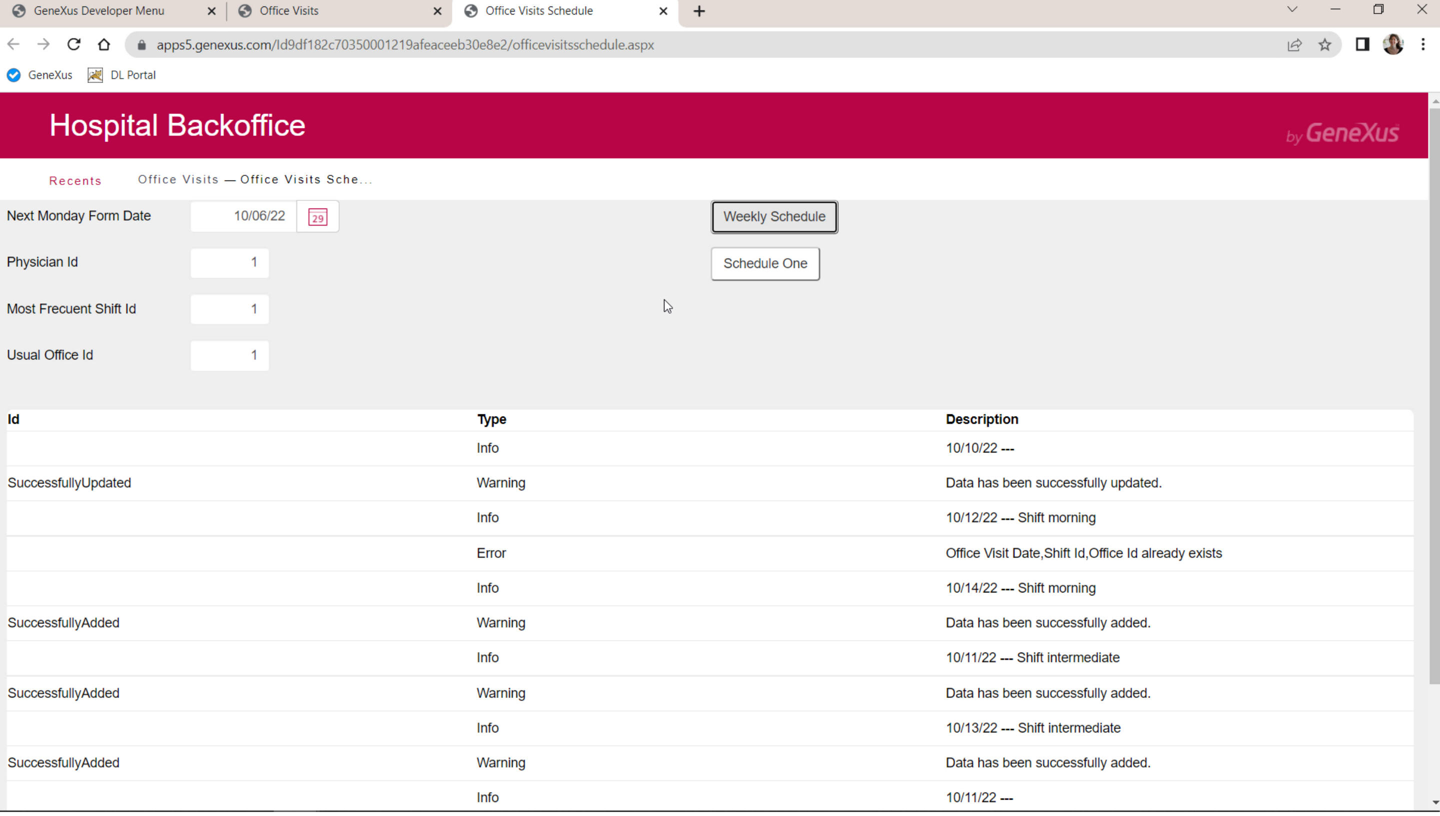
Or, if we want to do it in a simpler way, we try the Update and if it doesn't work, we try the Insert...

Let's do so.



Visit Date	Physician Id	Physician Name	Shift Name	Medical Specialt...	Office Id	Last Number		
10/10/22	1	Doctor 1	Shift morning	Family Medicine	2	0	UPDATE	DELETE
10/10/22	1	Doctor 1	Shift intermediate	Family Medicine	2	0	UPDATE	DELETE
10/11/22	1	Doctor 1	Shift evening	Family Medicine	2	0	UPDATE	DELETE
10/13/22	1	Doctor 1	Shift morning	Family Medicine	2	0	UPDATE	DELETE
10/12/22	2	Doctor 2	Shift morning	Family Medicine	1	0	UPDATE	DELETE

Here we have the initial data...

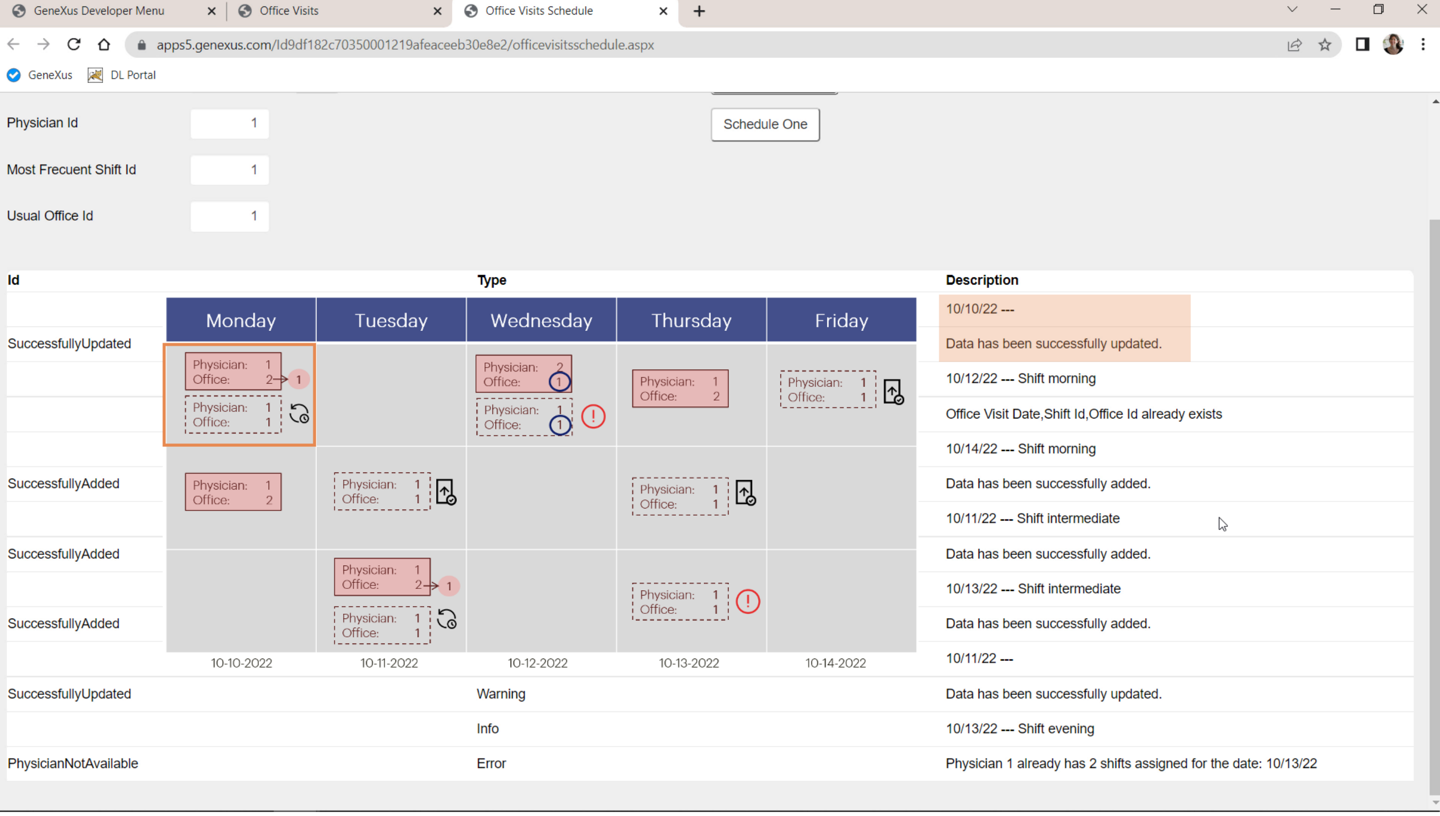


Next Monday Form Date	10/06/22
Physician Id	1
Most Frecuent Shift Id	1
Usual Office Id	1

Id	Type	Description
	Info	10/10/22 ---
SuccessfullyUpdated	Warning	Data has been successfully updated.
	Info	10/12/22 --- Shift morning
	Error	Office Visit Date,Shift Id,Office Id already exists
	Info	10/14/22 --- Shift morning
SuccessfullyAdded	Warning	Data has been successfully added.
	Info	10/11/22 --- Shift intermediate
SuccessfullyAdded	Warning	Data has been successfully added.
	Info	10/13/22 --- Shift intermediate
SuccessfullyAdded	Warning	Data has been successfully added.
	Info	10/11/22 ---

And now let's try to launch the schedule in batch mode, for the following week, doctor 1, shift 1.

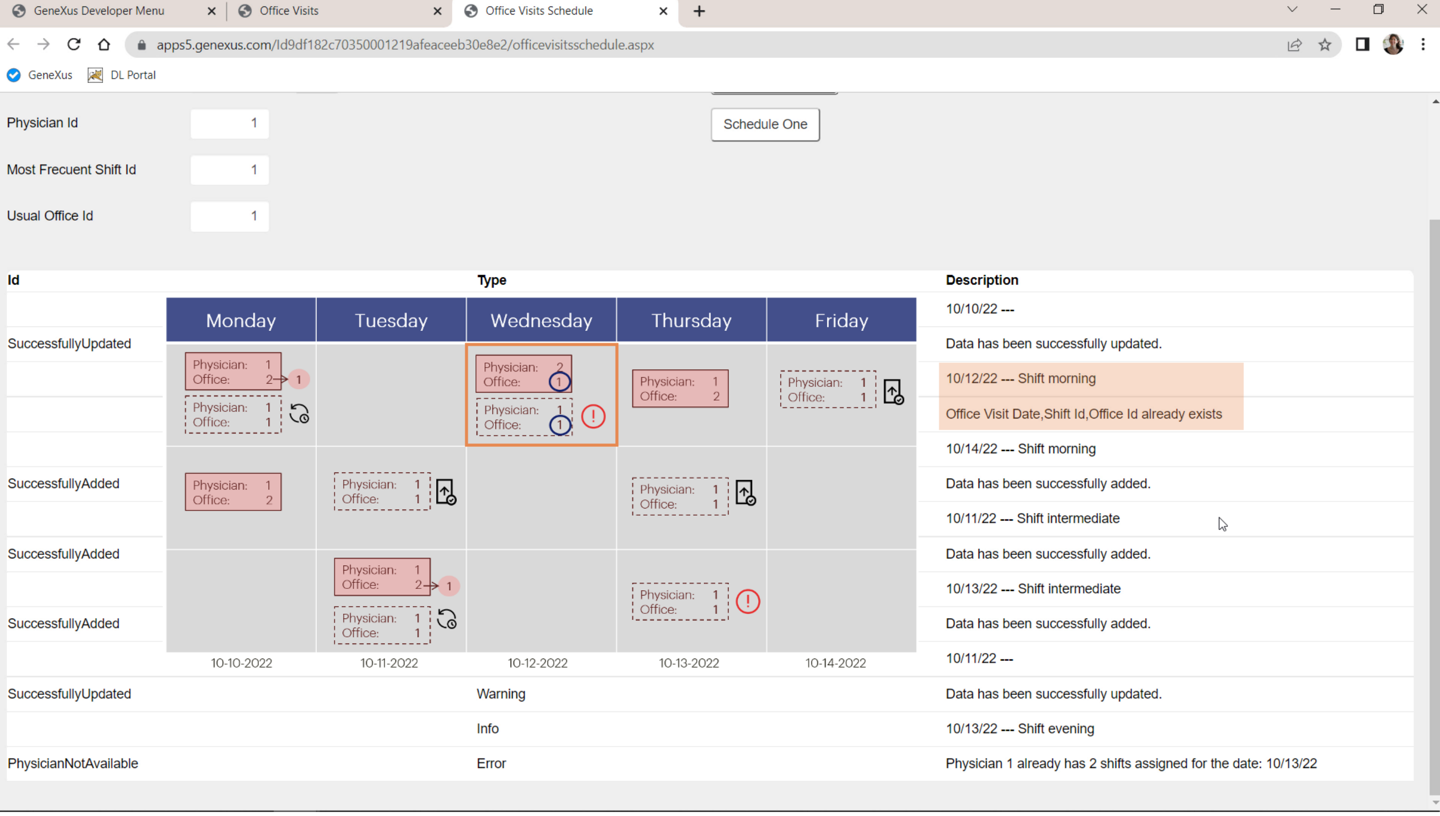
Let's compare the messages generated with what we expected...



Physician Id	1
Most Frecuent Shift Id	1
Usual Office Id	1

Id	Type	Description
		10/10/22 ---
SuccessfullyUpdated		Data has been successfully updated.
		10/12/22 --- Shift morning
		Office Visit Date,Shift Id,Office Id already exists
		10/14/22 --- Shift morning
SuccessfullyAdded		Data has been successfully added.
		10/11/22 --- Shift intermediate
SuccessfullyAdded		Data has been successfully added.
		10/13/22 --- Shift intermediate
SuccessfullyAdded		Data has been successfully added.
		10/11/22 ---
SuccessfullyUpdated	Warning	Data has been successfully updated.
	Info	10/13/22 --- Shift evening
PhysicianNotAvailable	Error	Physician 1 already has 2 shifts assigned for the date: 10/13/22

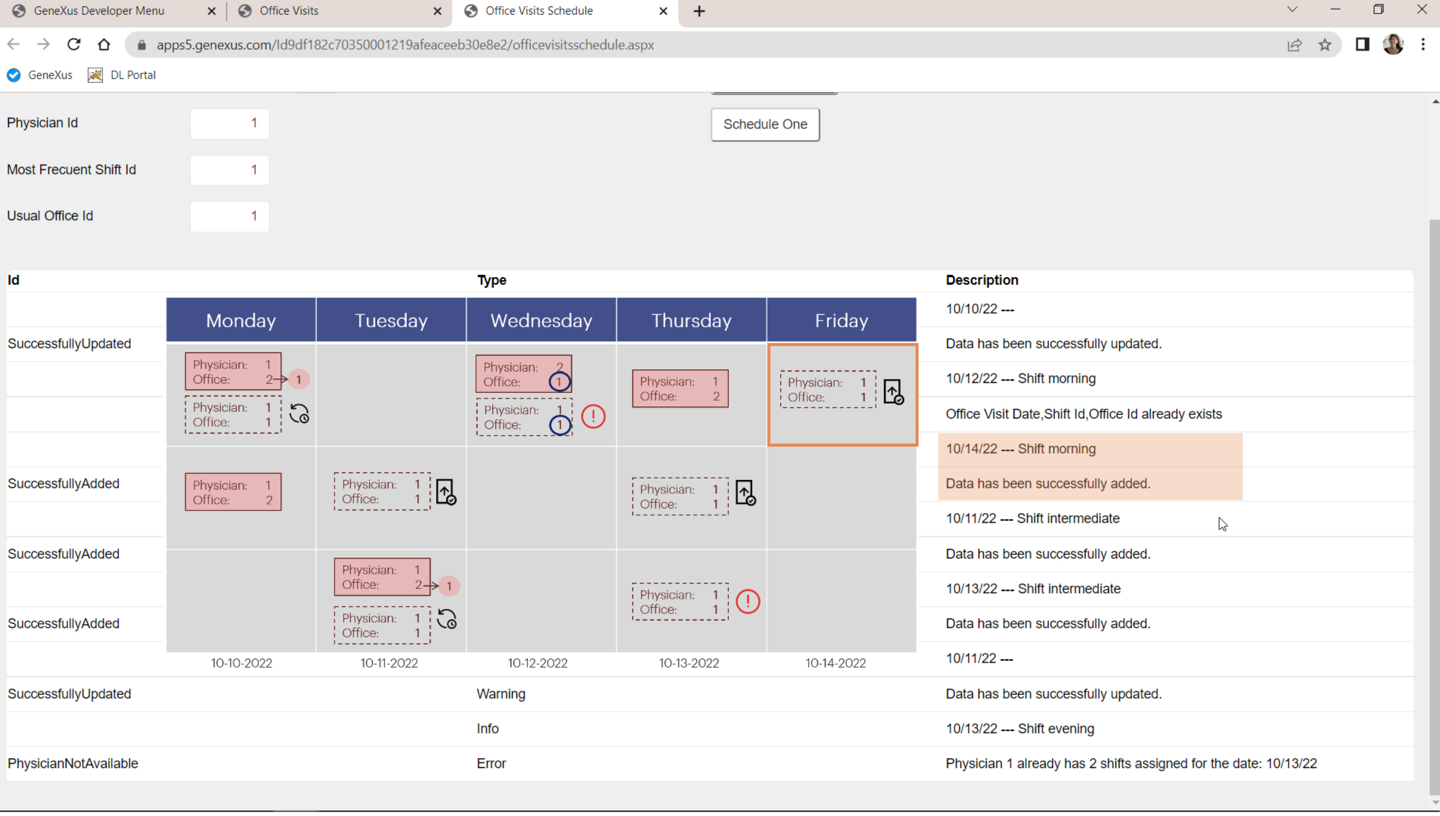
The first one was updated successfully, fine.



Physician Id	1
Most Frecuent Shift Id	1
Usual Office Id	1

Id	Type	Description
		10/10/22 ---
SuccessfullyUpdated		Data has been successfully updated.
		10/12/22 --- Shift morning
		Office Visit Date,Shift Id,Office Id already exists
		10/14/22 --- Shift morning
SuccessfullyAdded		Data has been successfully added.
		10/11/22 --- Shift intermediate
SuccessfullyAdded		Data has been successfully added.
		10/13/22 --- Shift intermediate
SuccessfullyAdded		Data has been successfully added.
		10/11/22 ---
SuccessfullyUpdated	Warning	Data has been successfully updated.
	Info	10/13/22 --- Shift evening
PhysicianNotAvailable	Error	Physician 1 already has 2 shifts assigned for the date: 10/13/22

The second one failed due to duplicate candidate key, good.

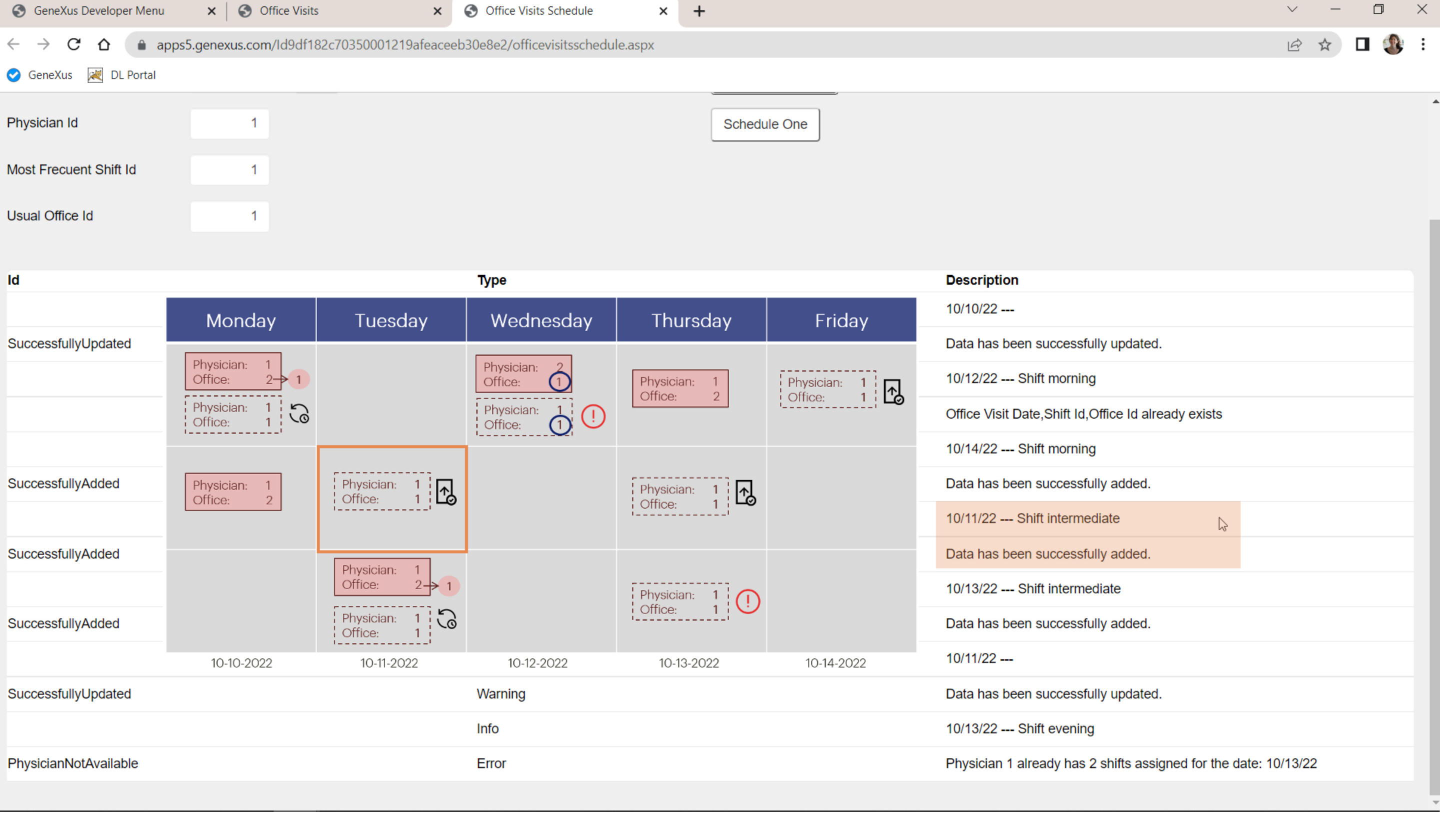


Physician Id	1
Most Frecuent Shift Id	1
Usual Office Id	1

Monday	Tuesday	Wednesday	Thursday	Friday
Physician: 1 Office: 2 → 1 Physician: 1 Office: 1		Physician: 2 Office: 1 Physician: 1 Office: 1	Physician: 1 Office: 2	Physician: 1 Office: 1
Physician: 1 Office: 2	Physician: 1 Office: 1		Physician: 1 Office: 1	
	Physician: 1 Office: 2 → 1 Physician: 1 Office: 1		Physician: 1 Office: 1	
10-10-2022	10-11-2022	10-12-2022	10-13-2022	10-14-2022

Id	Type	Description
		10/10/22 ---
SuccessfullyUpdated		Data has been successfully updated.
		10/12/22 --- Shift morning
		Office Visit Date,Shift Id,Office Id already exists
		10/14/22 --- Shift morning
SuccessfullyAdded		Data has been successfully added.
		10/11/22 --- Shift intermediate
SuccessfullyAdded		Data has been successfully added.
		10/13/22 --- Shift intermediate
SuccessfullyAdded		Data has been successfully added.
		10/11/22 ---
SuccessfullyUpdated	Warning	Data has been successfully updated.
	Info	10/13/22 --- Shift evening
PhysicianNotAvailable	Error	Physician 1 already has 2 shifts assigned for the date: 10/13/22

The third one was successfully inserted, good.



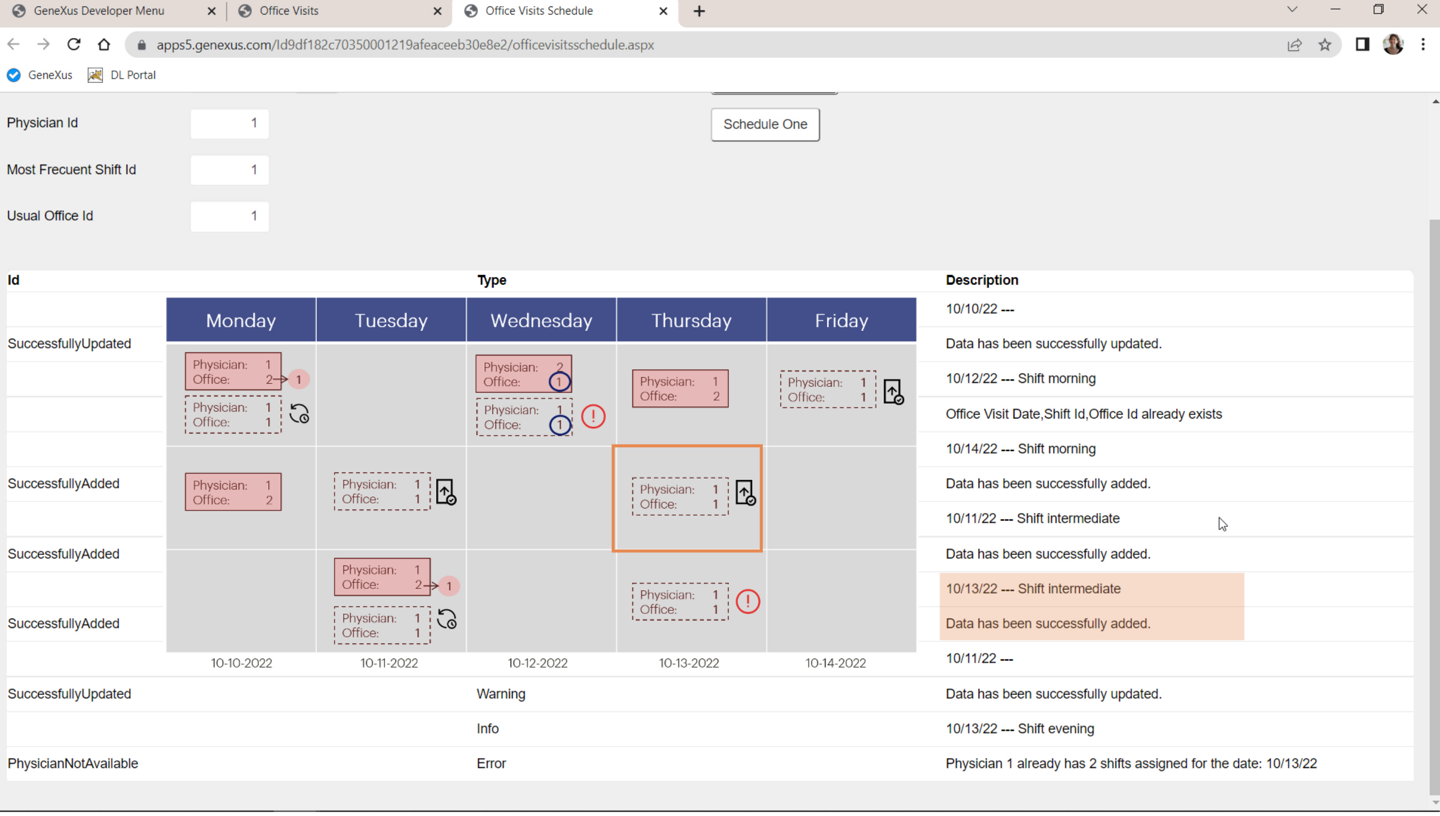
Physician Id 1

Most Frecuent Shift Id 1

Usual Office Id 1

Id	Type	Description
		10/10/22 ---
SuccessfullyUpdated		Data has been successfully updated.
		10/12/22 --- Shift morning
		Office Visit Date,Shift Id,Office Id already exists
		10/14/22 --- Shift morning
SuccessfullyAdded		Data has been successfully added.
		10/11/22 --- Shift intermediate
SuccessfullyAdded		Data has been successfully added.
		10/13/22 --- Shift intermediate
SuccessfullyAdded		Data has been successfully added.
		10/11/22 ---
SuccessfullyUpdated	Warning	Data has been successfully updated.
	Info	10/13/22 --- Shift evening
PhysicianNotAvailable	Error	Physician 1 already has 2 shifts assigned for the date: 10/13/22

The fourth one was successfully inserted, good.



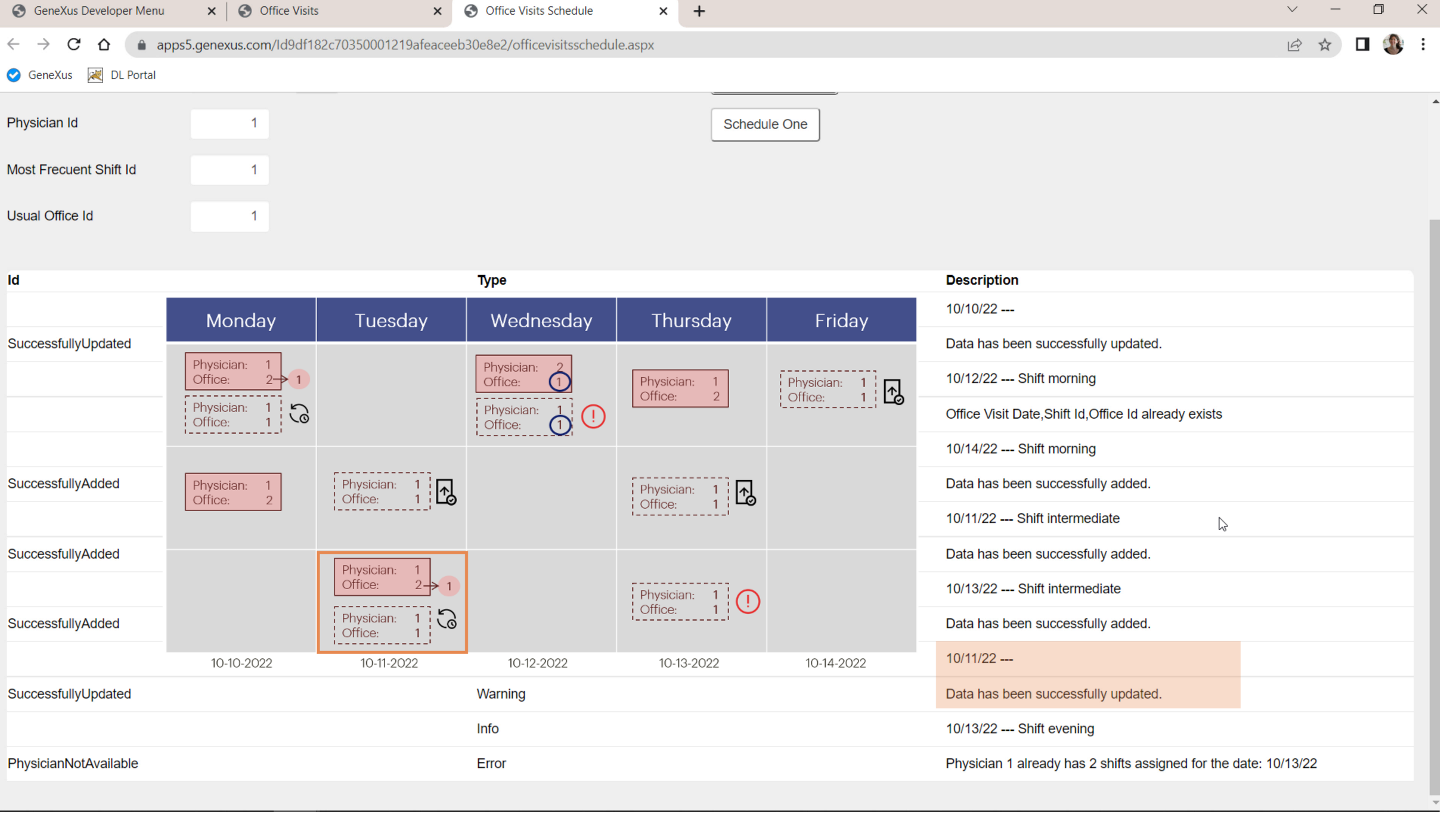
Physician Id 1

Most Frecuent Shift Id 1

Usual Office Id 1

Id	Type	Description
		10/10/22 ---
SuccessfullyUpdated		Data has been successfully updated.
		10/12/22 --- Shift morning
		Office Visit Date,Shift Id,Office Id already exists
		10/14/22 --- Shift morning
SuccessfullyAdded		Data has been successfully added.
		10/11/22 --- Shift intermediate
SuccessfullyAdded		Data has been successfully added.
		10/13/22 --- Shift intermediate
SuccessfullyAdded		Data has been successfully added.
		10/11/22 ---
SuccessfullyUpdated	Warning	Data has been successfully updated.
	Info	10/13/22 --- Shift evening
PhysicianNotAvailable	Error	Physician 1 already has 2 shifts assigned for the date: 10/13/22

The fifth one was also successfully inserted, OK.



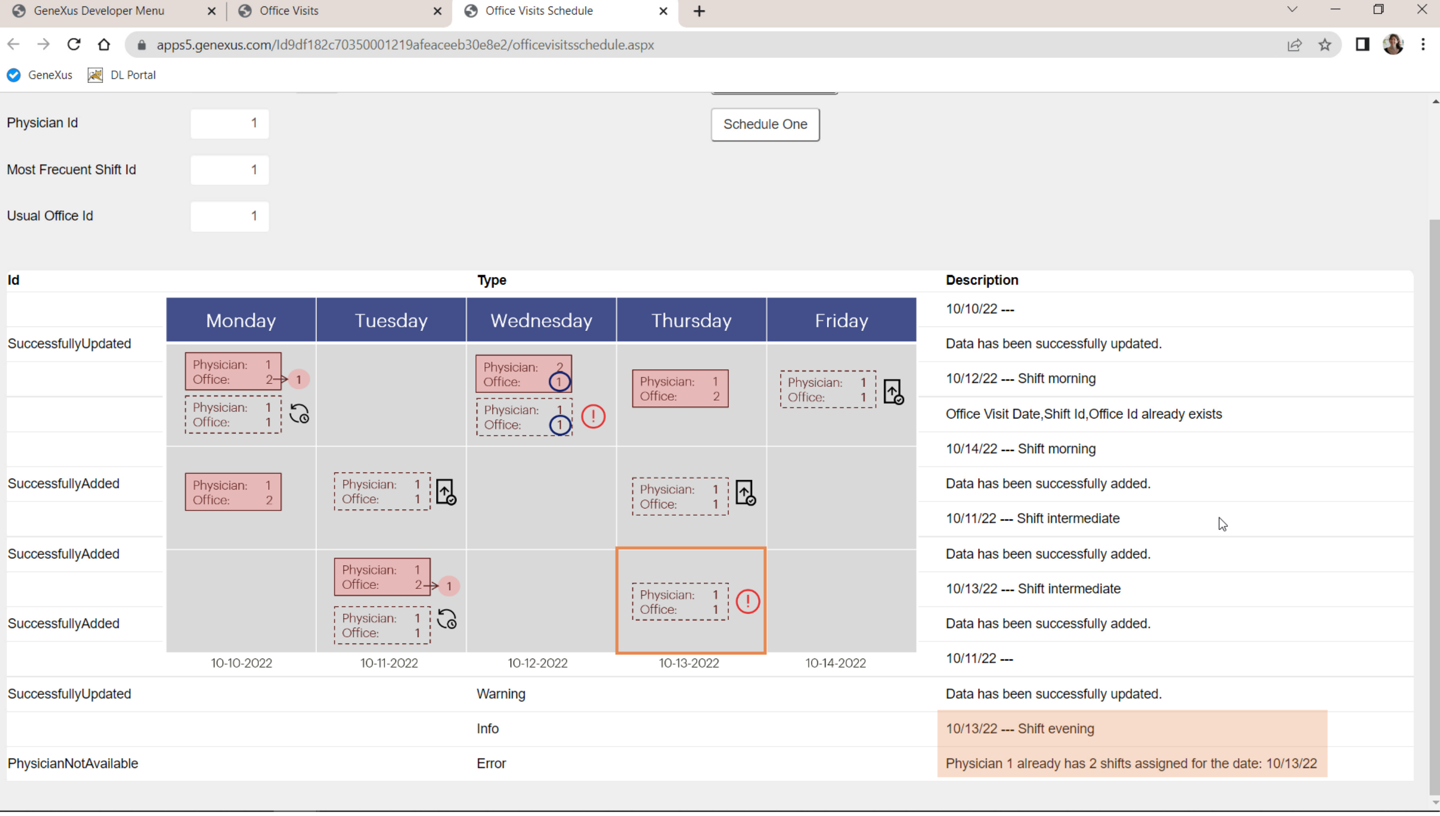
Physician Id 1

Most Frecuent Shift Id 1

Usual Office Id 1

Id	Type	Description
		10/10/22 ---
SuccessfullyUpdated		Data has been successfully updated.
		10/12/22 --- Shift morning
		Office Visit Date,Shift Id,Office Id already exists
		10/14/22 --- Shift morning
SuccessfullyAdded		Data has been successfully added.
		10/11/22 --- Shift intermediate
SuccessfullyAdded		Data has been successfully added.
		10/13/22 --- Shift intermediate
SuccessfullyAdded		Data has been successfully added.
		10/11/22 ---
SuccessfullyUpdated	Warning	Data has been successfully updated.
	Info	10/13/22 --- Shift evening
PhysicianNotAvailable	Error	Physician 1 already has 2 shifts assigned for the date: 10/13/22

The sixth one was successfully updated, OK.



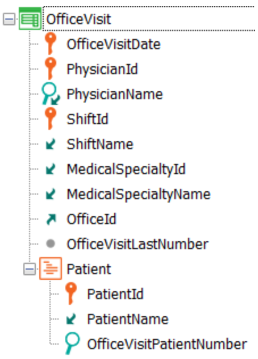
Physician Id	1
Most Frecuent Shift Id	1
Usual Office Id	1

Monday	Tuesday	Wednesday	Thursday	Friday
Physician: 1 Office: 2→1 Physician: 1 Office: 1		Physician: 2 Office: 1 Physician: 1 Office: 1	Physician: 1 Office: 2	Physician: 1 Office: 1
Physician: 1 Office: 2	Physician: 1 Office: 1		Physician: 1 Office: 1	
	Physician: 1 Office: 2→1 Physician: 1 Office: 1		Physician: 1 Office: 1	
10-10-2022	10-11-2022	10-12-2022	10-13-2022	10-14-2022

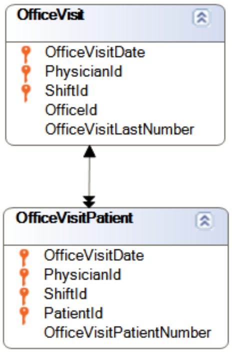
Id	Type	Description
		10/10/22 ---
SuccessfullyUpdated		Data has been successfully updated.
		10/12/22 --- Shift morning
		Office Visit Date,Shift Id,Office Id already exists
		10/14/22 --- Shift morning
SuccessfullyAdded		Data has been successfully added.
		10/11/22 --- Shift intermediate
SuccessfullyAdded		Data has been successfully added.
		10/13/22 --- Shift intermediate
SuccessfullyAdded		Data has been successfully added.
		10/11/22 ---
SuccessfullyUpdated	Warning	Data has been successfully updated.
	Info	10/13/22 --- Shift evening
PhysicianNotAvailable	Error	Physician 1 already has 2 shifts assigned for the date: 10/13/22

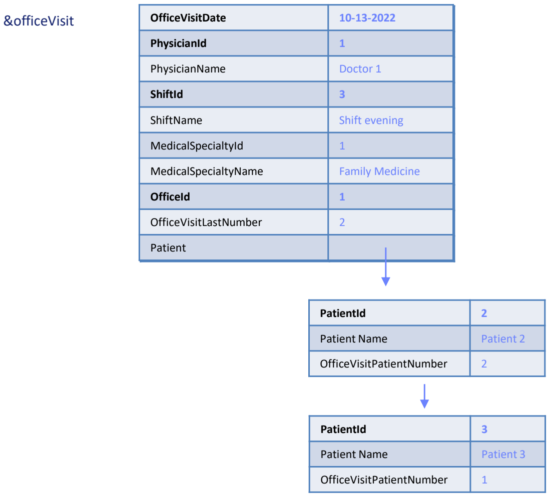
And the last one failed because the doctor already had 2 office visits for that day, OK.

Let's see that the office of this visit has indeed changed... OK.
And this one... fine too.



- OfficeVisit
  - OfficeVisitDate
  - PhysicianId
  - PhysicianName
  - ShiftId
  - ShiftName
  - MedicalSpecialtyId
  - MedicalSpecialtyName
  - OfficeId
  - OfficeVisitLastNumber
  - Patient
    - PatientId
    - PatientName
    - OfficeVisitPatientNumber





&officeVisit

OfficeVisitDate	10-13-2022
PhysicianId	1
PhysicianName	Doctor 1
ShiftId	3
ShiftName	Shift evening
MedicalSpecialtyId	1
MedicalSpecialtyName	Family Medicine
OfficeId	1
OfficeVisitLastNumber	2
Patient	

PatientId	2
Patient Name	Patient 2
OfficeVisitPatientNumber	2

PatientId	3
Patient Name	Patient 3
OfficeVisitPatientNumber	1

So far, we have worked with two-level BCs, but their second level was empty. What we are studying here in relation to the Insert or Update will have no difference with the case in which there is a patient collection that is not empty.

Next, after addressing the differences between business component methods, we will show how to perform the Insert or Update operations with a two-level BC where we will work with the lines, but we will do it at the highest level, using a Data Provider.

GeneXus™

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