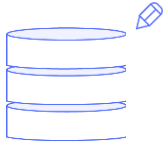


Atualização da base de dados

Business Components. Regras, eventos e verificações. Revisão

GeneXus™



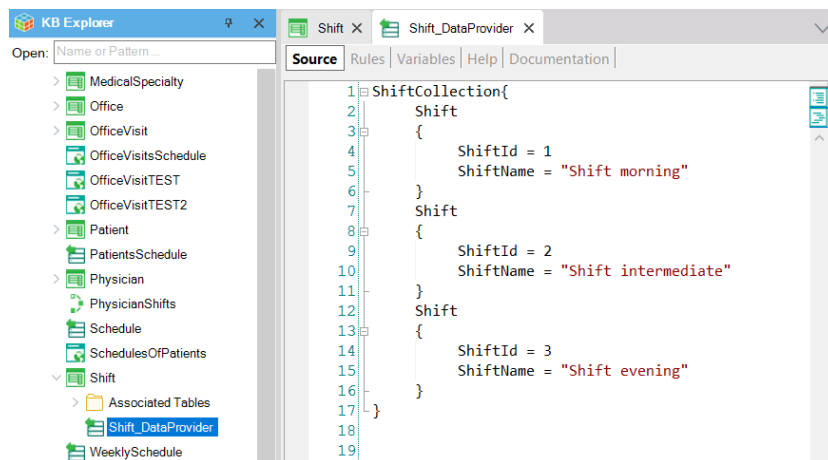
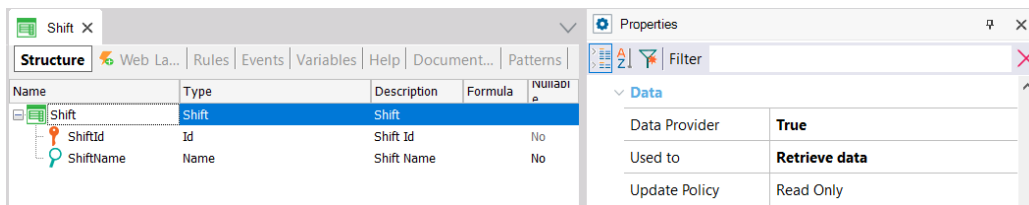
Insert, Update, Delete

Business Component

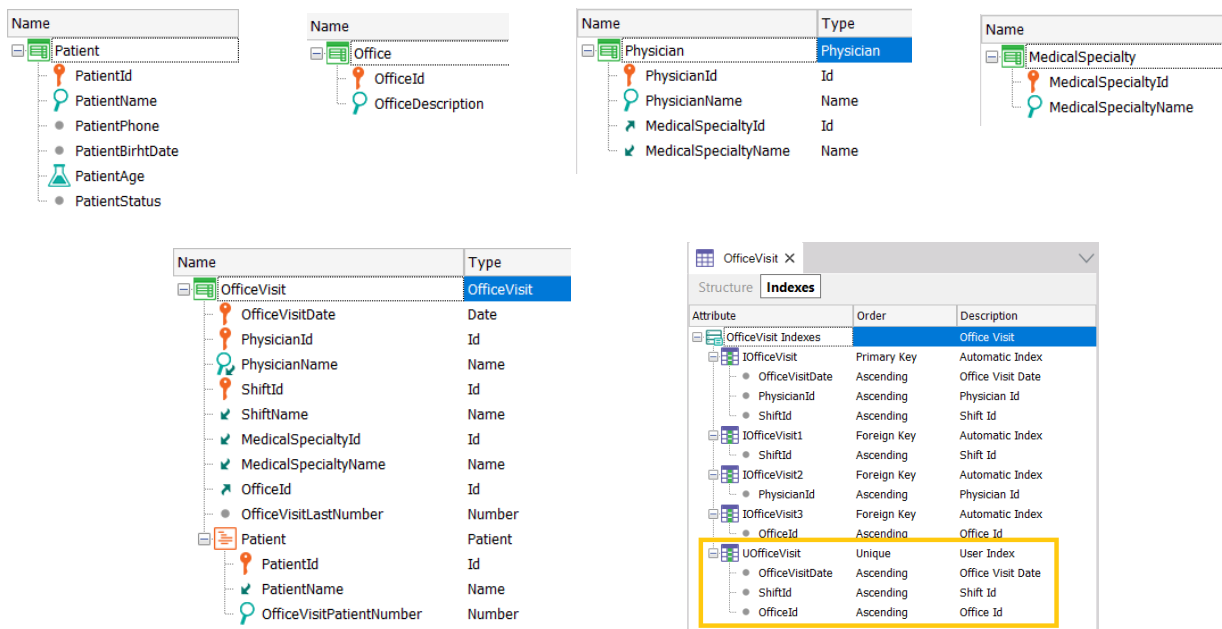
KB Hospital

Vamos integrar o que sabemos sobre atualização da base de dados através de Business Components e agregar algum conhecimento um pouco mais avançado.

Para isso vamos focar em uma KB que estamos desenvolvendo para um serviço de saúde...



... que oferece aos seus pacientes a possibilidade de realizar consultas médicas em um dos três turnos possíveis: matutino, intermediário ou vespertino. Para representar esses turnos temos a transação Shift, que é dinâmica (vemos que ativamos a propriedade Data Provider e dissemos que era para recuperar dados, e apenas recuperá-los, pois vemos que deixamos aqui Read Only. Aqui indicamos os 3 turnos.



Por outro lado, temos a transação Patient para registrar os pacientes, Office para registrar os consultórios em que serão realizadas essas consultas médicas, Physician para registrar a informação de cada médico, que possui uma (e apenas uma) especialidade médica, e por último, e nesta vamos focar, a transação OfficeVisit, que é aquela que registra cada consulta em que um médico atende em uma data e turno (matutino, intermediário ou vespertino) uma lista de pacientes.

Como aos pacientes é desejado atribuir um número para ser atendidos, é adicionado no primeiro nível um atributo que registra o último número fornecido, de forma a ser possível utilizar uma regra serial cada vez que for inserido um novo paciente através da transação.

Também é necessário indicar em qual consultório será realizada a consulta. Já podemos observar que embora definimos que o trio “data de consulta, médico e turno” não pode ser repetido (constitui a chave primária), também não deve ser repetido data de consulta, turno e consultório (ou seja, não pode haver duas consultas em uma mesma data, turno e consultório). Portanto, estes três atributos formam uma chave candidata. Por esse motivo, criamos um índice unique para representá-lo (e garanti-lo).

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

```

Também queremos acrescentar duas regras de negócio: um mesmo médico pode atender até em dois turnos no mesmo dia, mas não três. E, por outro lado, não deve ser permitida a inserção de um paciente que já tenha uma consulta pendente para um médico da mesma especialidade.

As verificamos assim:

Aqui estamos chamando este procedimento somente se estivermos em modo Insert; nele é contada a quantidade de consultas do médico na data em que estamos querendo atribuir uma nova consulta. E dispararemos um erro se retornar 2 ou mais.

Aqui temos a regra serial para que quando estiver sendo inserido um novo paciente para a consulta, seja atribuído o próximo número para ser atendido. E aqui desabilitamos o campo, pois não queremos que o usuário o modifique.

E aqui estamos chamando outro procedimento que verifica se o paciente tem alguma consulta pendente para a mesma especialidade da consulta para a qual está tentando agendar. Nesse caso, é disparado um erro.

Vemos que os procedimentos e as regras de erro só são executados quando está querendo **inserir**: cabeçalho para um caso e linha para o outro.

Além disso, observemos que, como aplicamos o pattern WorkWith à transação, ele automaticamente adicionou estas regras (entre elas, a parm

para receber chave e modo).

Nosso objetivo será revisar e aprofundar em que de tudo isto é executado quando são realizadas as operações através do 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			

Vamos supor que temos estes dados na base de dados para a próxima semana, sem pacientes ainda agendados, exceto para a consulta de quinta-feira, que tem o paciente 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

Recents Office Visit — Office Visits

SHOW FILTERS Office Visits Name + 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

Aqui os vemos no Work With.

Queremos inserir uma nova para segunda-feira e testaremos se funciona conforme o esperado em todos os casos.

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

Vamos resumir tudo o que sabemos que irá verificar a transação em execução quando tentarmos inserir e então veremos o que verifica o Business Component.

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

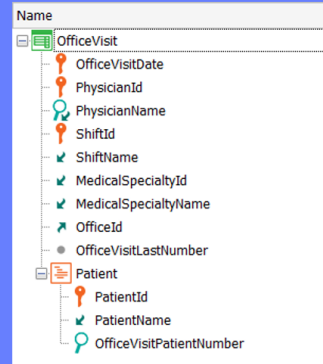
CK: {OfficeVisitDate, ShiftId, Officeld}

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}



Pela ordem dos atributos na estrutura da transação, a ordem das verificações será a que mostramos.

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}

The screenshot shows a web browser window displaying the 'Hospital Backoffice' application. The page title is 'Office Visit'. The form contains the following fields:

- Visit Date:** 141122 (highlighted as 'Invalid date')
- Physician Id:** (empty)
- Physician Name:** (empty)
- Shift Id:** 0
- Shift Name:** (empty)
- Medical Specialty Id:** 0
- Medical Specialty Name:** (empty)
- Office Id:** 0

Não adicionamos as verificações dos tipos de dados. Por exemplo, para o primeiro campo claramente verificará se a data é válida.

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

A próxima verificação será que existe um médico com esse Id na tabela de médicos. Caso contrário, veremos este erro de integridade referencial. Vamos colocar o médico 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

A seguinte coisa que será feita é invocar o procedimento que conta a quantidade de consultas que esse médico já tem nesse dia e se der 2 ou mais, será disparada a regra de erro, que mostra esta mensagem em tela.

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

Com o médico 2 não deu erro, mas se colocarmos no lugar o médico 1, que sabemos que já tem 2 turnos nesse dia... vemos o erro.

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

Este código interno do erro corresponde ao segundo parâmetro da regra de erro. Se não o colocarmos, o Id do erro ficará vazio (o que não afeta em nada o funcionamento).

Voltemos a colocar o médico 2 para seguir.

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

A próxima será controlar a integridade referencial com 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	

Vamos atribuir agora o 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}

The screenshot shows a web browser window displaying the 'Office Visit' form in the GeneXus application. The form contains several input fields and a dropdown menu. The 'Office Id' field is highlighted with a red border and a yellow tooltip that reads 'No matching 'Office''. Below the form, a table lists patient information.

Patient Id	Patient Name	Patient Number
<input type="text" value=""/>		0

Em seguida, controlará que o consultório exista. Por exemplo, colocamos o 25.

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/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

E agora este, que existe.

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: Officeld
CK: {OfficeVisitDate, ShiftId, Officeld}
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

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

Ao sair do campo Officeld, já é preciso verificar a chave candidata. Neste caso tínhamos colocado consultório 3, mas observemos o que aconteceria se colocássemos o 2, que sabemos que já está ocupado (pelo médico 1). Nos mostra esta mensagem de erro. Para verificar é que ele utiliza o índice unique.

Voltamos a colocar o consultório 3, que sabemos que não está ocupado.

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/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

Neste momento, é controlada a unicidade do registro que está querendo inserir. Neste caso, não encontrou que o registro já existia, portanto não vemos nada.

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

Se tivéssemos tentado inserir um registro que já existia... por exemplo, observemos que para este médico e este turno, para o dia quarta-feira já tínhamos uma consulta agendada, então vamos ver o que aconteceria se mudássemos a data para a nova consulta que estamos querendo atribuir, e colocamos esta da quarta-feira.

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

Não está mostrando para nós interativamente, mas se pressionarmos confirm veremos a mensagem "Record already exists".

Vamos voltar a colocar a data de segunda-feira.

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

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

Com isto, as verificações do primeiro nível terminaram.

Agora passa a verificar cada linha do segundo nível. Se colocarmos um id de paciente inexistente, falhará a integridade referencial. Vamos colocar um existente, o 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 Id	Patient Name	Patient Number
1	Patient1	1
2	Patient2	0

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


A próxima coisa a ser verificada é o erro. Vamos tentar colocar o paciente 2. A regra de erro que tínhamos declarado está sendo disparada. Por quê?

Porque se lembramos, já tínhamos o paciente 2 agendado para a consulta da quinta-feira, para o médico 1 que é da mesma especialidade que o médico 2 (Family Medicine). Então não podemos agendá-lo.

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: Officeld	
CK: {OfficeVisitDate, ShiftId, Officeld}	
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}	

Então, será verificada a unicidade da chave primária deste segundo nível. Como não temos nenhum outro paciente inserido, não falhará. Mas vamos testar com outra linha. Voltemos a colocar o 1. Aqui vemos que está controlando a unicidade, efetivamente.

Se agora pressionamos confirm, veremos inserida a consulta.

	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; margin-bottom: 5px;">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>			

Agora vamos tentar repetir a mesma coisa, mas inserindo através do Business Component. Vamos então excluir a consulta recém-inserida para tentar novamente.

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

```

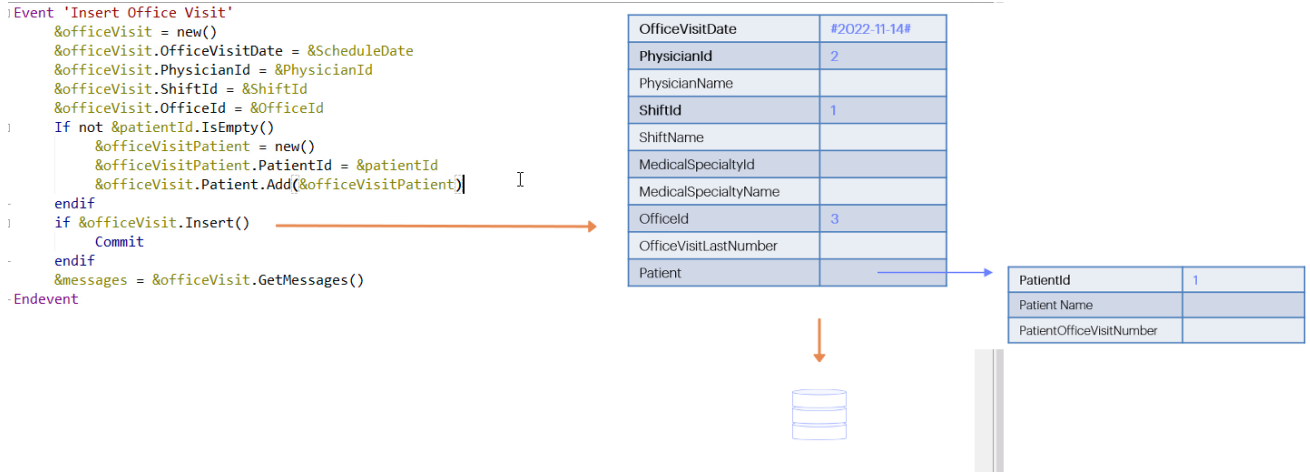
Type Definition

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

Para facilitar, criamos um web panel com 5 variáveis de entrada: para os 4 atributos relevantes do cabeçalho da consulta e para inserir um paciente. Primeiro tentaremos inserir através do BC. Para fazer isso, é claro, ativamos a propriedade Business Component da transação.

A partir daí é que conseguimos criar esta variável do tipo de dados Business Component, através da qual iremos inserir.

Como boa prática, solicitamos primeiramente espaço de memória novo para a variável, para garantir que ela esteja limpa. Atribuímos valores aos 4 elementos do cabeçalho, a partir das variáveis da tela. Então, se o campo na tela para Patient Id não foi deixado vazio, tentaremos adicionar um paciente. Temos esta variável do tipo de dados de cada linha, a limpamos com new, atribuímos valor apenas ao elemento PatientId e o adicionamos à coleção de patients do Business Component.



Com isso carregamos a estrutura da variável &officeVisit. Agora temos que dar a ordem para tentar a inserção...

```

)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

E se for bem-sucedida, commitar.

The screenshot shows the GeneXus IDE interface. The main editor displays the following code for the 'Insert Office Visit' event:

```

1 Event 'Insert Office Visit'
2   &officeVisit = new()
3   &officeVisit.OfficeVisitDate = &ScheduleDate
4   &officeVisit.PhysicianId = &PhysicianId
5   &officeVisit.ShiftId = &ShiftId
6   &officeVisit.OfficeId = &OfficeId
7   If not &patientId.IsEmpty()
8     &officeVisitPatient = new()
9     &officeVisitPatient.PatientId = &patientId
10    &officeVisit.Patient.Add(&officeVisitPatient)
11  endif
12  If &officeVisit.Insert()
13    Commit
14  endif
15  &messages = &officeVisit.GetMessages()
16 Endevent

```

The code is annotated with a red circle around the `&officeVisit.Insert()` call and a red box around the `&messages = &officeVisit.GetMessages()` assignment. Below the code, a table displays the structure of the Messages data table:

Name	Type	Description	Is Collection
Messages		Messages	<input checked="" type="checkbox"/>
Message			
Id	VarChar(128)	Id	<input type="checkbox"/>
Type	MessageTypes, GeneXus	Type	<input type="checkbox"/>
Description	VarChar(256)	Description	<input type="checkbox"/>

The right-hand pane shows the Properties window for the variable `&messages`, with the following details:

- Variable: &messages**
- Name:** messages
- Description:** messages
- Column title:** messages
- Class:** Attribute
- Type Definition:** Based on (none); Data Type: Messages, GeneXus.Common; Collection: False
- Validation:** Value range: ; Validation Failed Me: ;
- Control Info:** Control Type: Edit; Input Type: Values; Notify Context Chan: False
- Behavior:** Input History: False; Is Password: False
- Appearance:** Auto Resize: True; Width: 4chr; Height: 1row; Fill: True

The Output window at the bottom shows the following messages:

```

Show: General
Saving Web Component 'OfficeVisitPatientW...' skipped (no significant changes).
Saving Transaction 'OfficeVisit'... skipped (no significant changes).
Saving Procedure 'ListPrograms'... skipped (no significant changes).
Success: Pattern generation (WorkWithOfficeVisit)

```

Na variável `&messages` do tipo de dados SDT que vem predefinido no módulo GeneXus, obtemos todas as mensagens que resultaram da execução do método `Insert`. E a inserimos na tela para poder ver todas essas mensagens.

Então vamos executar para repetir tudo o que fizemos através da transação, mas agora por esta via.

Hospital Backoffice

Recents Office Visit TEST

Schedule Date: 11/14/22

Physician Id:

Shift Id:

Office Id:

Patient Id:

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

Vamos escolher como data a da segunda-feira que queríamos. Vamos colocar um médico inexistente, um turno inexistente, um consultório inexistente e até um paciente inexistente, para ver todas as mensagens que obteremos ao tentar inserir. Vemos todos os erros de integridade referencial. Para Patient nem chegou porque isso só aconteceria se conseguisse inserir o cabeçalho.

Office Visits x Office Visit TEST x +

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...

Agora vamos colocar o médico 1, no turno 3, no consultório 3, sem paciente. Sabemos que tem que falhar pela regra de erro.

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...

Vemos que o id da mensagem de erro é aquele que colocamos na regra.

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'.

Podemos violar mais verificações e veremos isso na coleção de mensagens. Por exemplo, vamos colocar o shift 25, que não existe, e vemos as duas mensagens de erro.

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'.

Agora vamos testar a verificação de unicidade de chave primária. Vamos tentar inserir um registro para a quarta-feira, médico 2, turno 1, consultório 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

Bem, era o que esperávamos.

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: 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			

by GeneXus

Id	Type	Description
DuplicatePrimaryKey	Error	Record already exists

Agora vamos testar inserir a consulta para a segunda-feira no turno 1, mas para consultório já ocupado, ou seja, o 2.

Hospital Backoffice

Recents Office Visit TEST

Schedule Date

Physician Id

Shift Id

Office Id

Patient Id

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(), PhysicianId) 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>	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

Vemos que verificou unicidade de chave candidata.

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:

Id	Type	Description
SuccessfullyAdded	Warning	Data has been successfully added.

Agora sim, vamos colocar o consultório correto. Se deixamos vazio Patient Id, inserirá o cabeçalho com sucesso. Vemos que a mensagem é do tipo Warning e é a que aparece na transação quando não é aplicado o pattern, ou seja, quando não retorna ao objeto chamador.

Office Visits x Office Visit TEST x +

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

Checks	Error Id	Error Description
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'.

Eliminemos o registro e tentemos novamente, desta vez com paciente. Colocamos paciente inexistente e vemos o erro de integridade referencial. Portanto o cabeçalho não é inserido.

Office Visits x Office Visit TEST x +

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 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;		
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;		
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

Se agora tentamos colocar o paciente 2, que já tem uma consulta pendente para a mesma especialidade... também nos lança o erro e também não grava.

Hospital Backoffice



Recents Office Visit TEST

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

Physician Id: 2

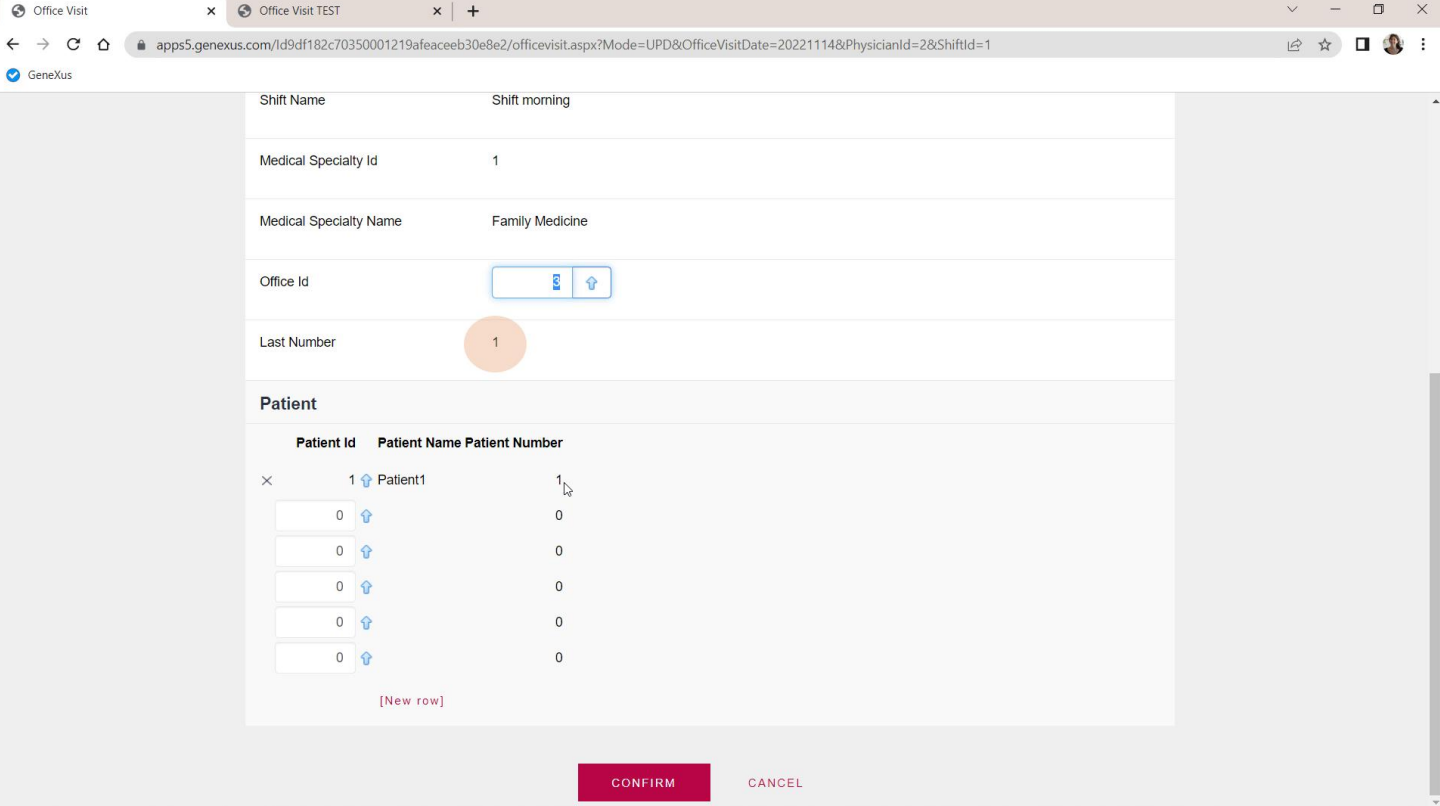
Shift Id: 1

Office Id: 3

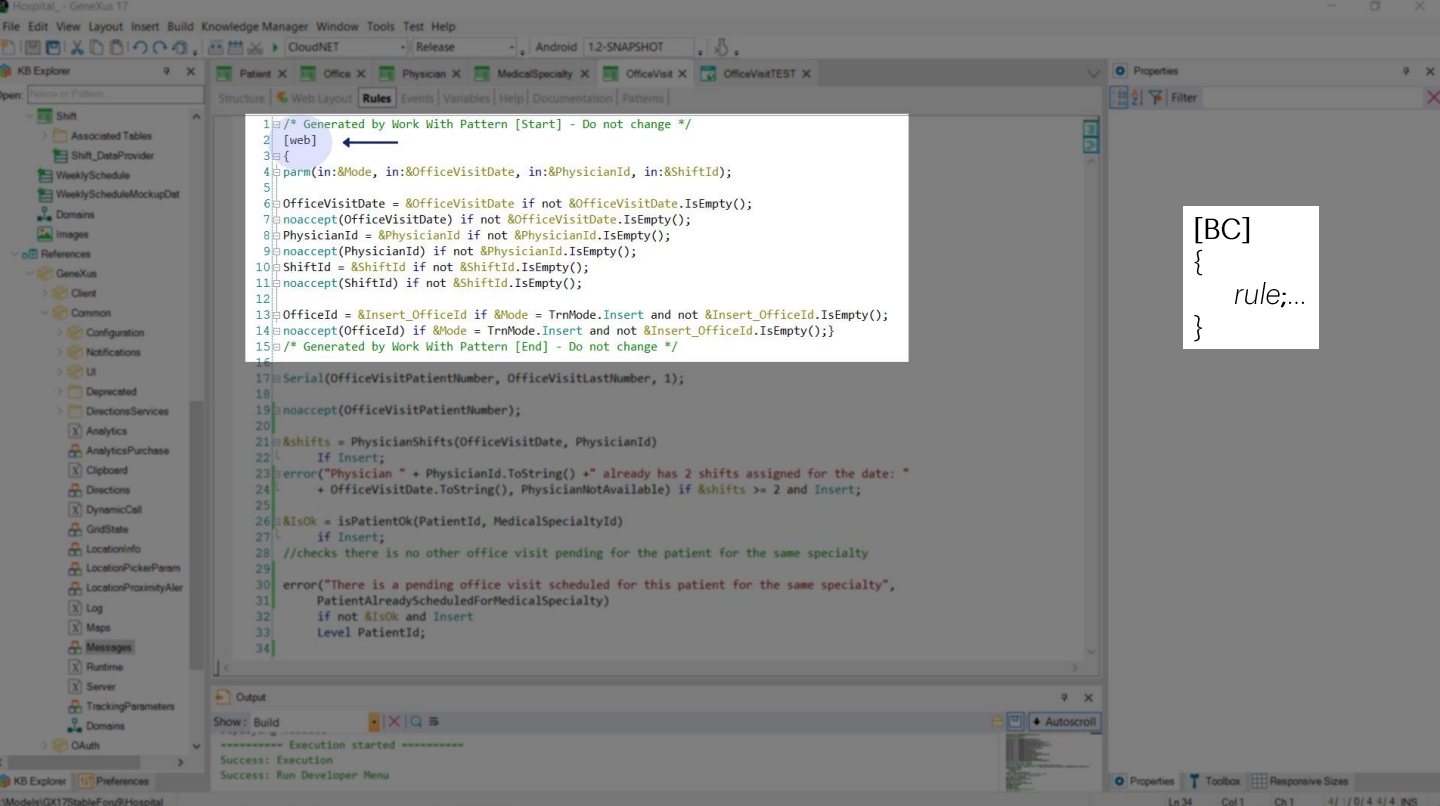
Patient Id: 1

Id	Type	Description
SuccessfullyAdded	Warning	Data has been successfully added.

Agora sim, vamos colocar o paciente 1. Success!



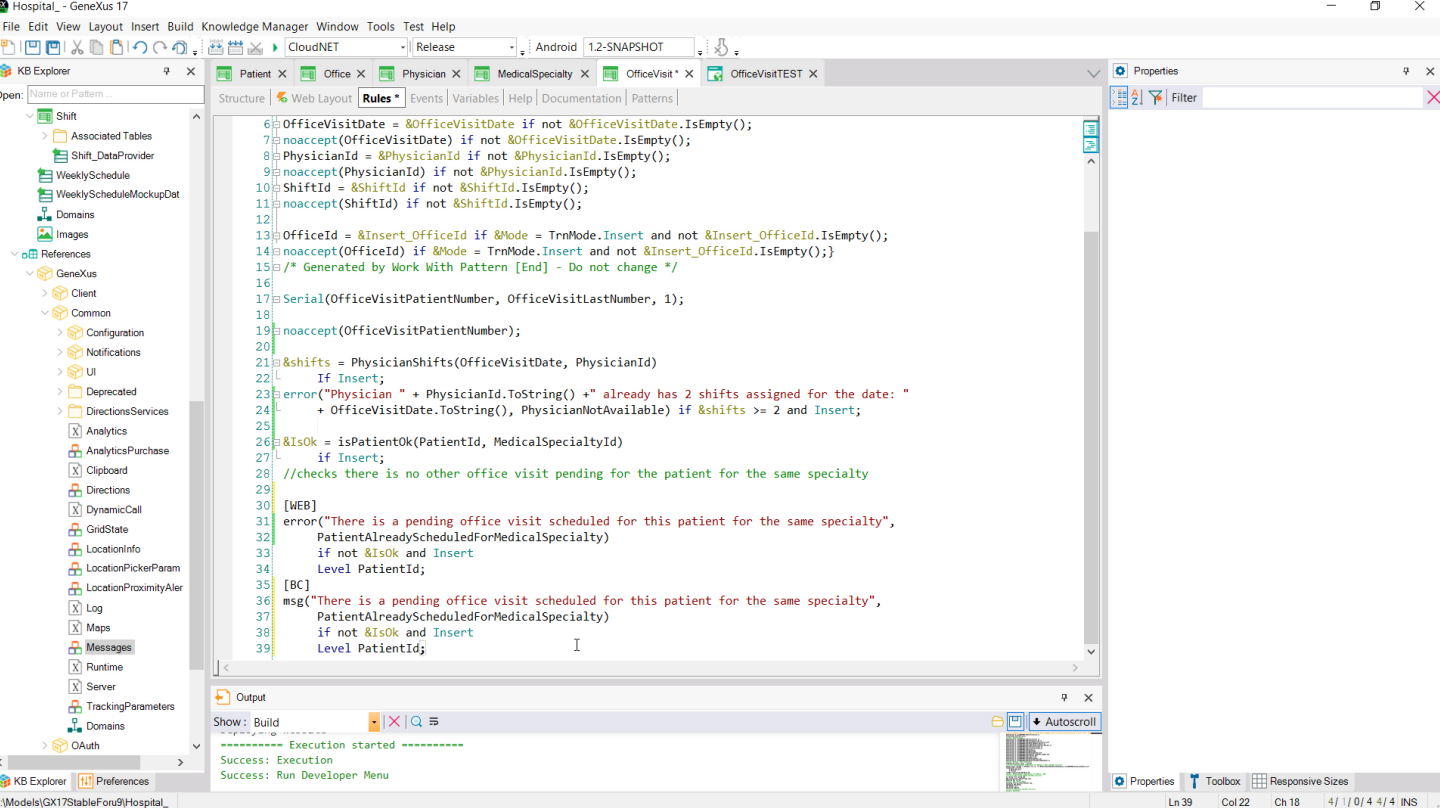
Se formos ver como ficou gravada essa consulta, é indistinguível de quando a inserimos através da transação. Podemos observar que também foi disparada a regra serial, dando o número 1 ao paciente e atualizando o atributo de último número dado.



De todas as regras da transação, quais foram disparadas ao inserir através do Business Component?

Como já sabemos, são disparadas todas aquelas que não dependem da interface e não fazem sentido neste contexto, como a regra parm, logicamente, pois o BC não é um objeto que pode ser chamado como a transação, mas um tipo de dados especial, com métodos especiais.

No nosso caso, a regra parm não só não foi disparada porque não faz sentido, mas também porque observamos que todo o bloco de regras adicionadas pelo pattern foram qualificadas com isto, que é conhecido como atributo de environment. Assim se indica que apenas sejam incluídas estas regras quando está sendo executada a transação Web, e não quando é executado como Business component. Se, por outro lado, quiséssemos que alguma regra apenas seja executada no contexto do Business Component, então podemos qualificá-la assim. As chaves são necessárias se são várias regras as qualificadas e não apenas uma.



Por exemplo, suponhamos que se for inserido por Business Component não queremos impedir que seja inserido um paciente que tenha uma consulta pendente. De qualquer forma, apenas queremos lançar uma mensagem neste caso. Então qualificamos a regra de erro para que seja executada apenas em environment Web e adicionamos uma regra message para o caso de BC.

Hospital Backoffice



Recents Office Visits — Office Visit TEST

Schedule Date: 11/21/22

Physician Id:

Shift Id:

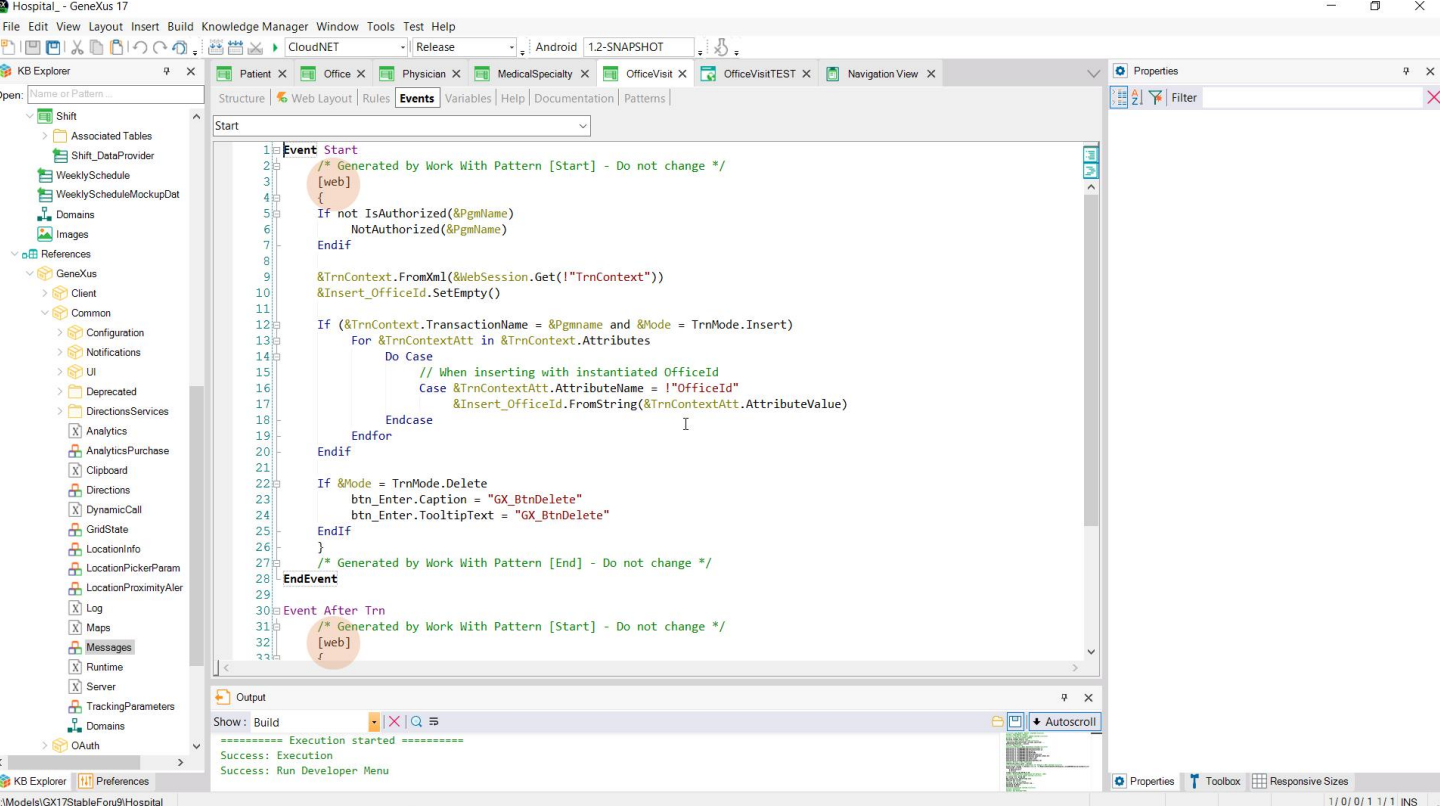
Office Id:

Patient Id:

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.

Vamos testar. Se tentamos pela transação, nos lança um erro, assim como esperávamos. Se, em vez disso, tentamos pelo BC, vemos que o tipo de mensagem é Warning e inseriu sem problemas.

Voltemos a deixá-lo como estava.



Também os eventos podem ser qualificados.

De todos os eventos definidos em uma transação, quais são executados quando é utilizado o business component? Apenas o Start e o After Trn. Se esses eventos incluem referências a objetos com interface de usuário, essas referências são ignoradas.

Coincidentemente, são os dois eventos que aqui adicionou o pattern. Mas observemos que estão qualificados assim, ou seja, que só serão executados no contexto da transação Web e não quando for executado o Business Component.

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

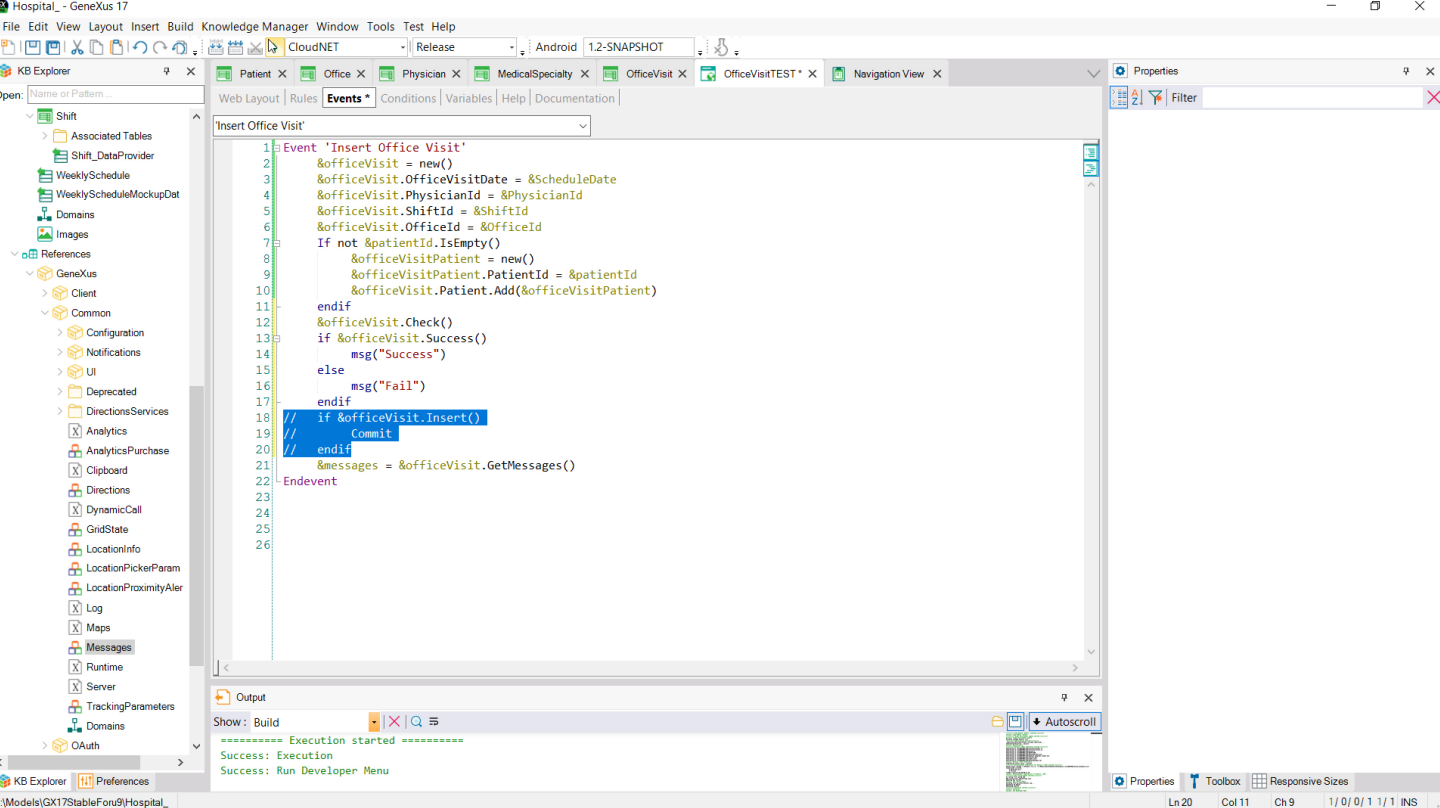
```
[BC]
{
  Event After Trn
  .....
  EndEvent

  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.

Se quiséssemos definir outras opções para esses eventos que valham apenas para Business Component, então as escreveríamos precedendo-os com colchetes BC.

Pode ser assim, englobando-os, ou dentro de cada um, diferenciando o que é web do que é bc.



Vamos voltar para a inserção com o BC. Poderíamos querer fazer as mesmas validações que fará o Insert, mas sem atualizar a base de dados. Para isso temos o método Check.

O invocamos e depois verificamos o resultado da operação, através do método Success, que retornará true se foi bem-sucedida e false caso contrário. Vamos parar aqui por um momento: o resultado da aplicação deste método a um BC se referirá ao resultado da última operação que tenha sido realizada sobre o BC. Aqui é Check, mas poderia ter sido Load, ou Delete, ou Save, ou mesmo a de Insert ou Update ou InsertOrUpdate. Em particular, estes últimos métodos já retornam seu resultado, portanto não precisamos do Success para conhecê-lo e, portanto, realizamos a operação e perguntamos seu resultado na mesma sentença. Embora não seja imprescindível, também temos o método oposto, Fail.

Indicaremos com uma mensagem ao usuário se a verificação foi ou não bem-sucedida. Para testar vamos deixar comentada a atualização da base de dados. Executemos.

Fail

Schedule Date: 11/22/22 29

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'.

A última data de consulta agendada é esta. Vamos tentar verificar uma inserção para o dia seguinte. Vamos colocar médico inexistente, turno inexistente, consultório inexistente. Vemos as mensagens de erro e a mensagem Fail.

Hospital Backoffice



Recents Office Visit TEST

Fail

Schedule Date

Physician Id

Shift Id

Office Id

Patient Id

Id	Type	Description
ForeignKeyNotFound	Error	No matching 'Patient'.

Agora vamos colocar tudo correto para o primeiro nível e um paciente inexistente. Vemos a mensagem de erro e a mensagem Fail.

Hospital Backoffice



Recents Office Visit TEST

Fail

Schedule Date

Physician Id

Shift Id

Office Id

Patient Id

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

E com o paciente 1 sabemos que tem que ser disparada a regra de erro.
Bem.

Success

Schedule Date: 11/22/22

Physician Id:

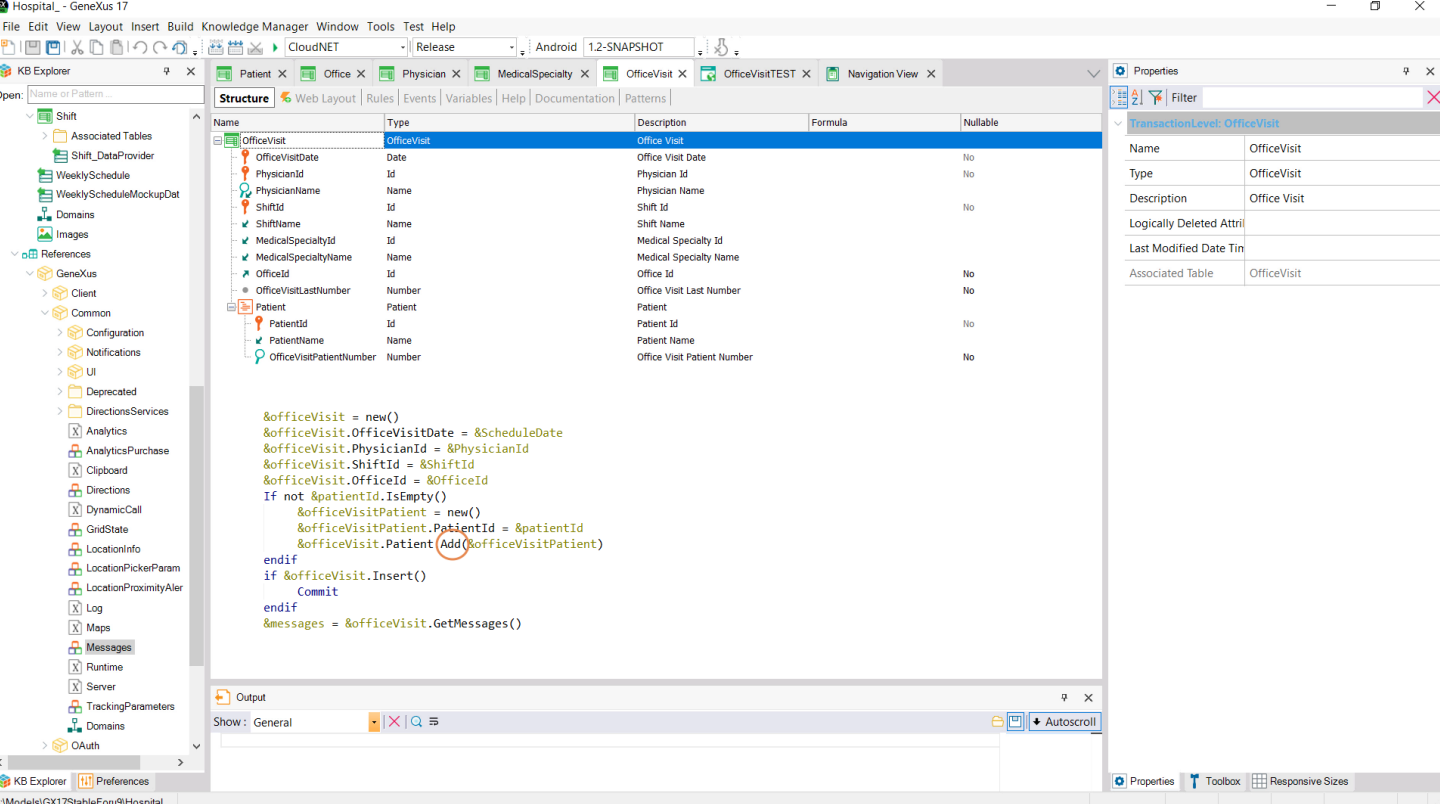
Shift Id:

Office Id:

Patient Id:

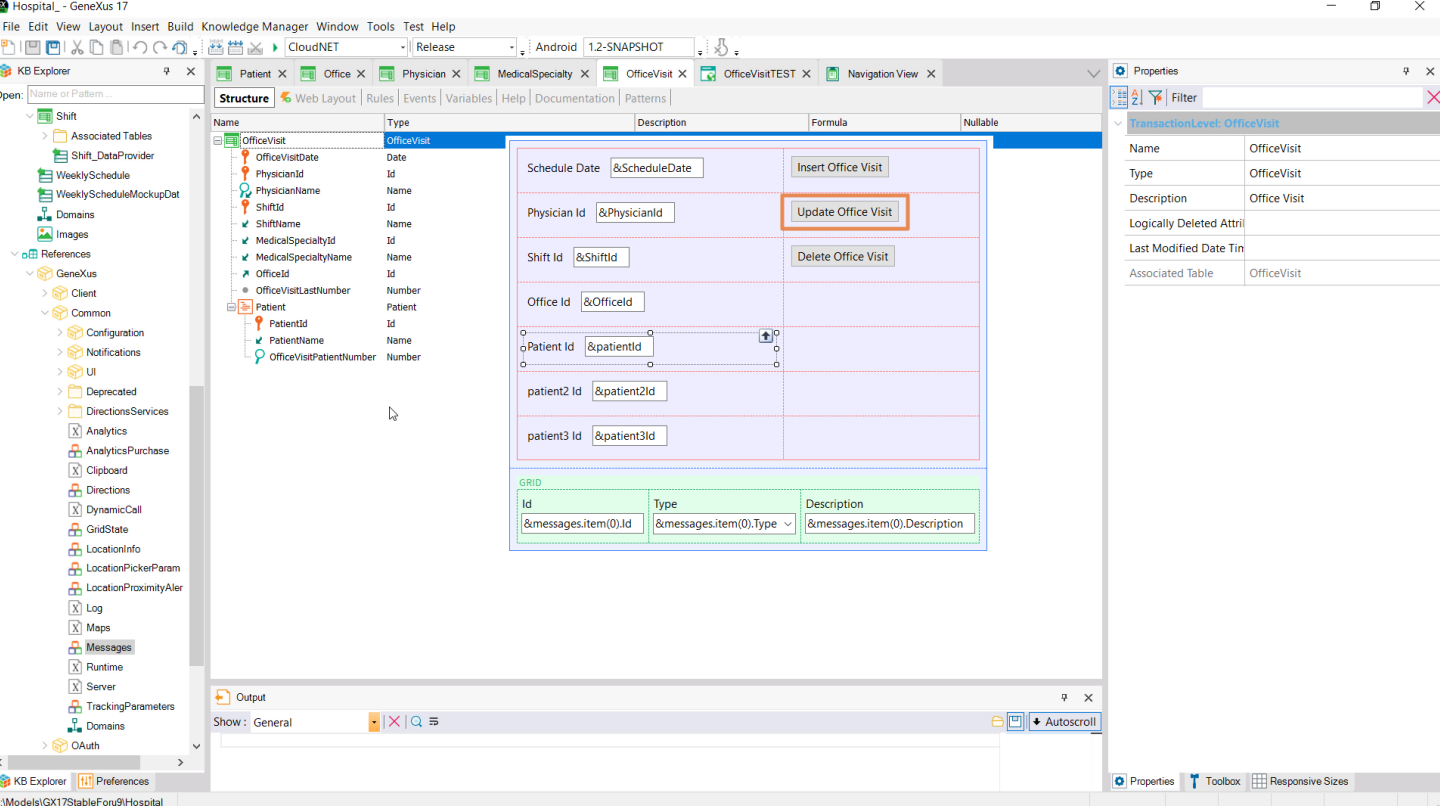
Id	Type	Description
----	------	-------------

E se agora escolhermos um correto? Vemos a mensagem Success.
Logicamente nada foi inserido na base de dados.



Bem, voltemos a deixar tudo como estava.

Até aqui vimos que quando se trata das verificações e execução de regras de negócios e eventos, quase não há diferença entre inserir um cabeçalho e suas linhas através da transação e fazê-lo através do business component. Também revisamos como se trabalha com o business component e, em particular, usamos a forma de adicionar linhas com o método Add da coleção. Depois, revisaremos outra, com Data Provider.



Um esclarecimento: aqui testamos o funcionamento do Business Component manualmente, através de um web panel que construímos especialmente e vamos testando em execução. Mas a forma de testar isto de modo que sempre possa ser testado novamente diante de qualquer alteração ou até mesmo em tempo de integração, é com os testes unitários. Não é recomendado fazer testes manuais como os que fizemos aqui.

No próximo vídeo vamos querer modificar uma consulta, não já inseri-la. Modificá-la implica modificar dados do cabeçalho e também das linhas: inserir novas, modificar existentes e excluir outras. E veremos, como fizemos aqui, que são disparadas as regras. Também analisaremos a eliminação.

Você pode pular esse vídeo se achar que domina estes casos e passar para o seguinte.

GeneXus™

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