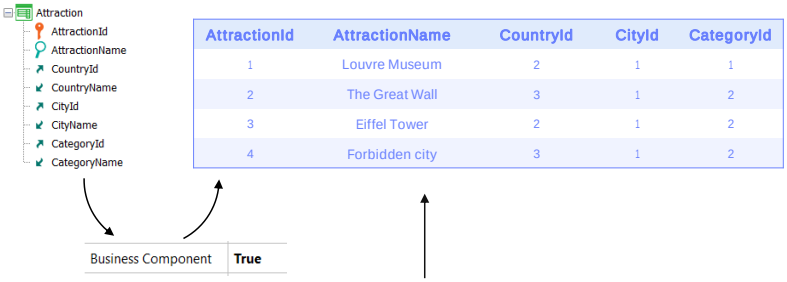
Atualização para a Base de Dados com comandos específicos de procedimentos.

Como inserir

GeneXus

1. **Business Compontent**: Insert(), Update(), Delete()

Insert, Update, Delete



2. **Procedure**: New, For each, Delete

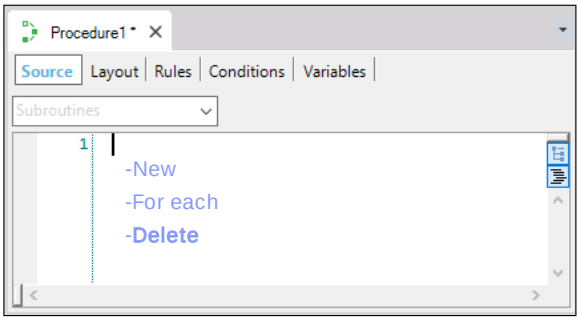
Para atualizar as informações da base de dados por código, tínhamos duas possibilidades:

Fazer utilizando o business component da transação, através de seus métodos Insert, Update ou Delete (o Save, e o InsertOrUpdate), ou fazer exclusivamente dentro de um procedimento, através dos comandos New, For each e Delete.

Em outros vídeos, estudamos detalhadamente o primeiro caso. Agora vamos nos aprofundar no segundo.

PROCEDURE ONLY

Insert, Update, Delete



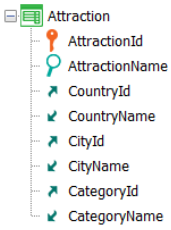
É muito importante ter em mente que esta segunda forma de atualização só pode ser feita no Source de um procedimento.
Os comandos que estudaremos não serão válidos em nenhum outro lugar, como eventos de Panels ou Web Panels, mas apenas aqui, em procedimentos.

Insert

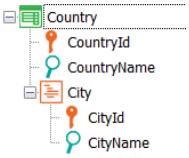
Vamos começar com o comando que permite inserir um registro em uma tabela.

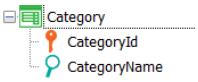
New Command

Insert, Update, Delete



AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2
5	Christ the Redemmer	1	2	2

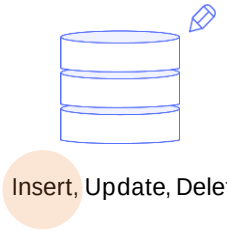




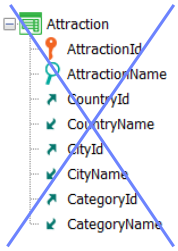
É literalmente UM registro e em UMA tabela.

Suponhamos que temos a transação Attraction, que registra atrações turísticas (onde também haverá uma transação Country que registra as informações de cada país e suas cidades e uma Category, que registra as categorias nas quais são classificadas as atrações turísticas), e que queremos inserir através de um procedimento, o Cristo Redentor na tabela associada a Attraction.

New Command



Insert, Update, Delete



AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2
5	Christ the Redemmer	1	2	2

New

```
AttractionId   = 5
AttractionName  = "Christ the Redemmer"
CountryId = find( CountryId, CountryName = "Brazil" )
CityId  = find( CityId, CityName = "Rio de Janeiro" )
CategoryId  = find( CategoryId, CategoryName = "Monument" )
```

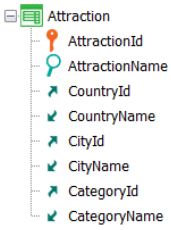
endnew

Para isso, pedimos que execute no Source do procedimento comando New, atribuindo valor a cada um dos atributos da tabela, para o registro que queremos inserir. Aqui estaremos trabalhando diretamente com a tabela, desvinculados completamente da transação que a cria. Ou seja, não importam as regras da transação, nem os eventos, nem nada. O comando new é completamente indiferente à transação. A única coisa com que se importa é a composição da tabela da base de dados na qual tentará inserir um registro.

Portanto, aqui os atributos inferidos ou fórmulas da transação não participam de forma alguma. Para o new, não existem. Os únicos que importam são os atributos da tabela. Por isso, neste caso, damos valor a todos eles.

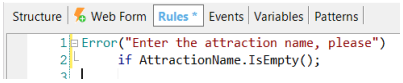
New Command

Insert, Update, Delete



AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2
5		1	2	2

Not assigned atributes?
Secondary attribute



New

AttractionId = 5
AttractionName = "Christ the Redemmer"
CountryId = find(CountryId, CountryName = "Brazil")
CityId = find(CityId, CityName = "Rio de Janeiro")
CategoryId = find(CategoryId, CategoryName = "Monument")

endnew

Isto é obrigatório? Claro que não. Se não atribuirmos valor a algum atributo, será considerado vazio.
Assim, se não atribuímos valor para AttractionName, será inserida a atração 5 sem nome. Existe na transação uma regra de erro que o impede. É que, como dissemos, a transação aqui não interfere para outra coisa senão para dar existência à tabela na base de dados. Esta é uma primeira grande diferença com a inserção por meio de Business Component.

New Command

Not assigned atributes?

Primary key attribute

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2

Uniqueness check

```
New
    AttractionId     = 5
    AttractionName  = "Christ the Redemmer"
    CountryId = find( CountryId, CountryName = "Brazil" )
    CityId  = find( CityId, CityName = "Rio de Janeiro" )
    CategoryId  = find( CategoryId, CategoryName = "Monument" )
endnew
```

AttractionId	AttractionName	CountryId	CityId	CategoryId
0	Louvre Museum	2	1	1
1	The Great Wall	3	1	2
2	Eiffel Tower	2	1	2
3	Forbidden city	3	1	2

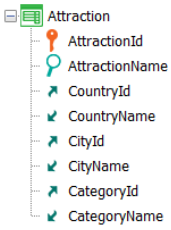
Mas, e o que acontece se o que não é atribuído for a chave primária?

Será feita uma tentativa de inserir um registro de chave vazia. Como se tivéssemos atribuído explicitamente 0.
Se a chave primária não é autonumerada, então, se não existe na tabela um registro com esse id zero, o insere.
E se existe? Não fará nada. Não veremos nenhuma diferença entre ter executado o New e não tê-lo feito.

Ou seja, o comando New controla a unicidade de registros. Não irá inserir registro de chave duplicada.

New Command

Insert, Update, Delete



AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2
5	Matisse Museum	2	2	1

New

Primary key attribute

Uniqueness check

```
AttractionId     = 5
AttractionName  = "Christ the Redemmer"
CountryId = find( CountryId, CountryName = "Brazil" )
CityId  = find( CityId, CityName = "Rio de Janeiro" )
CategoryId  = find( CategoryId, CategoryName = "Monument" )
```

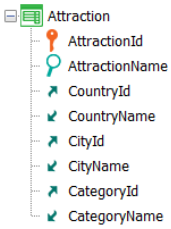
endnew

O mesmo que ocorreria se já existisse a atração com id 5 na tabela quando vamos executar o new. Aqui não fará absolutamente nada, devido ao controle de unicidade.

E isto vale tanto para chave primária quanto para chaves candidatas.

New Command

Insert, Update, Delete



Unique index

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2

Candidate key attribute

New

```
AttractionId     = 5
AttractionName  = "Eiffel Tower"
CountryId = find( CountryId, CountryName = "Brazil" )
CityId  = find( CityId, CityName = "Rio de Janeiro" )
CategoryId  = find( CategoryId, CategoryName = "Monument" )
```

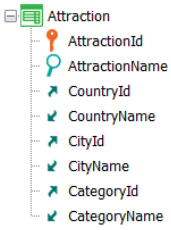
Uniqueness check

endnew

Imaginemos, por exemplo, que AttractionName seja chave candidata, ou seja, temos um índice unique definido neste atributo.
E suponhamos que o registro de Id 3 tivesse como valor de AttractionName o mesmo que estamos agora querendo inserir como registro 5.
Ao ser executado o new, não fará nada. Uma vez que encontrará que já existe um registro com o AttractionName idêntico ao do registro que estamos querendo inserir.

New Command

Insert, Update, Delete



AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2
5	Christ the Redemmer	1	2	2

Not assigned atributes?

Primary key attribute

autonumbered

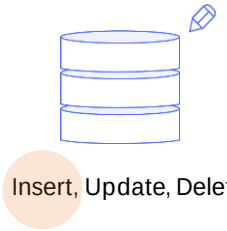
New

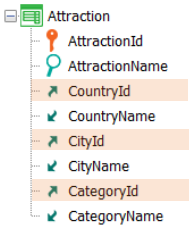
```
AttractionName  = "Christ the Redemmer"
CountryId = find( CountryId, CountryName = "Brazil" )
CityId  = find( CityId, CityName = "Rio de Janeiro" )
CategoryId  = find( CategoryId, CategoryName = "Monument
```

&AttractionId = AttractionId

Agora, voltando ao que nos perguntávamos antes, do que acontece quando deixamos sem especificar valor para a chave primária... se esta se autonumera, então nunca falhará por chave primária duplicada: sempre inserirá um registro com o número seguinte. Como sabemos qual número foi atribuído a ele? Permanece em memória, imediatamente após o new, o valor no atributo. Assim, podemos, por exemplo, atribuí-lo a uma variável para não perdê-lo no próximo uso do atributo.

New Command





AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2

Referential integrity checks?

NO

Not assigned atributes?

Foreign key attribute

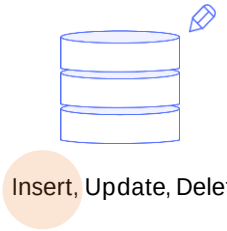
```
New
    AttractionId     = 5
    AttractionName  = "Christ the Redemmer"
    CountryId = find( CountryId, CountryName = "Brazil" )
    CityId  = find( CityId, CityName = "Rio de Janeiro" )
    CategoryId  = find( CategoryId, CategoryName = "Monument" )
endnew
```

E o que acontece com os atributos que são chaves estrangeiras? Como CountryId, CityId, CategoryId.

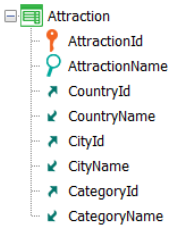
O comando new realiza controle de integridade referencial?

A resposta é não. Não realiza. É que a atualização por comando foi criada para ter uma forma de atualização rápida, com o melhor desempenho possível. Realizar essas verificações sempre retarda a operação. Quando se trata de um só registro, isto não importa, mas vamos pensar no que acontece se temos que inserir milhões de registros.

New Command



Insert, Update, Delete



AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2

New

Referential integrity checks?

NO

Not assigned atributes?

Foreign key attribute

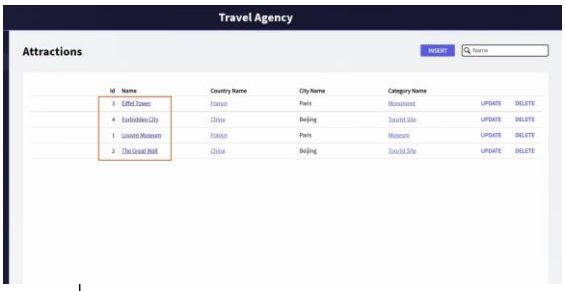
```
AttractionId     = 5
AttractionName  = "Christ the Redemmer"
CountryId = 15
CityId  = find( CityId, CityName = “Rio de Janeiro” )
CategoryId  = find( CategoryId, CategoryName = "Monument“ )
```

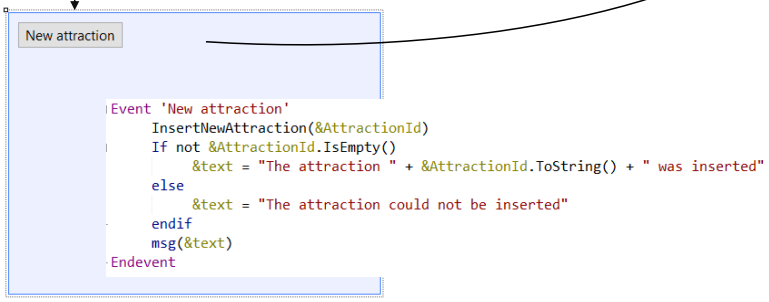
endnew

Assim, por exemplo, se para o novo registro queremos atribuir um país de id 15, o comando new não controlará que exista na tabela que armazena os países, um com esse valor. Irá inserir o registro sem problemas. E a base de dados ficará em estado inconsistente.

Vamos ao GeneXus para testá-lo.

New Command

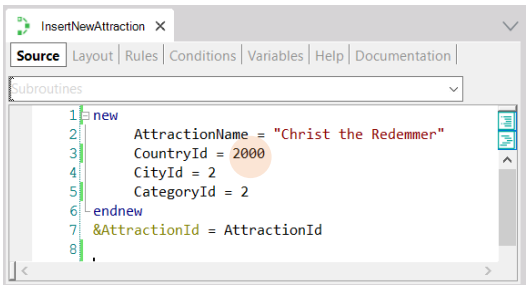


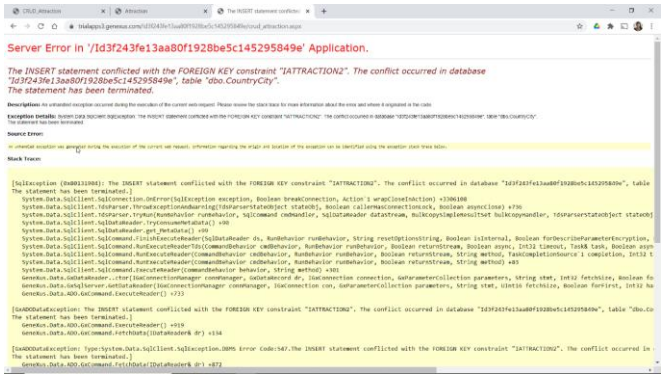


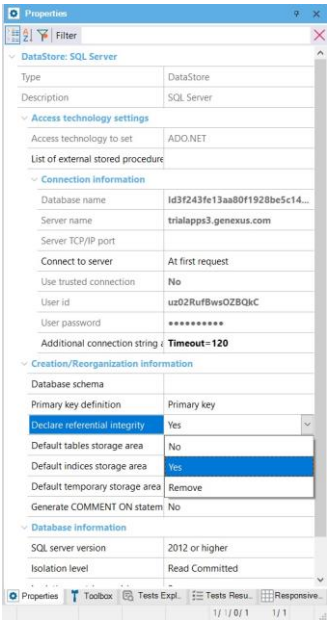
Temos as três transações com dados. Em particular, Attraction tem estes quatro registros.
AttractionId é autonumerado.

Criamos este web panel onde o usuário pressionará o botão "New attraction", que chamará um procedimento que tentará inserir um novo registro na tabela Attraction para o Cristo Redentor.
O procedimento retornará o AttractionId que a base de dados atribuiu ao registro inserido. Com isto montamos a mensagem que o usuário verá no painel.
Vamos testar.
The attraction 5 was inserted. Vamos ver... e aqui vemos, efetivamente, a atração 5 inserida na tabela.

New Command





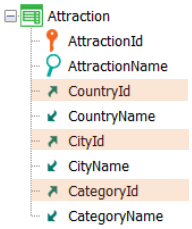


Agora, observemos o que acontece se para o novo registro quisermos atribuir um país inexistente. Por exemplo, este. Executemos.
Travamos o programa, quando o que esperávamos era que tivesse inserido o registro sem problema, já que dissemos que o new não controla a integridade referencial. O que aconteceu então?

É verdade que o programa não está controlando a integridade referencial, mas a base de dados está. Então o new tentou realizar a inserção, mas a base de dados não deixou, e lançou uma exceção.

Por padrão, a base de dados controla a integridade referencial. Podemos desligar este controle, por meio de uma propriedade.
Mas o que vemos claramente é que o new não está fazendo isso. Portanto, devemos ter muito cuidado ao utilizar este comando, pois um cancelamento de programa como este é inaceitável para o usuário final.

New Command



AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2

Insert, Update, Delete

Not assigned atributes?

Foreign key attribute

New

Referential integrity checks?

NO

AttractionId = 5
~~Attraction~~Name = "Christ the Redemmer"
CountryId = 15
CityId = find(CityId, CityName = “Rio de Janeiro”)
CategoryId = find(CategoryId, CategoryName = "Monument“)

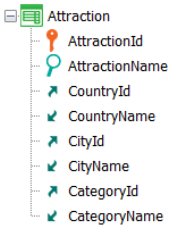
endnew

E o que acontece se deixamos sem atribuir uma chave estrangeira?

Novamente, o new não realizará nenhuma verificação e tentará inserir o registro. Se a base de dados fizer isso, então lançará uma exceção como a anterior, que interromperá o programa se não for capturada e manipulada por ele.
Poderíamos pensar que se o atributo aceita nulos, então, nesse caso, não falhará nunca, porque a base de dados permitirá esse nulo para a chave estrangeira. No entanto, deve-se ter cuidado sobre como a base de dados interpreta que se trata de um valor nulo e não de um valor vazio. Sobre isso falaremos em outro vídeo.

New Command

Insert, Update, Delete



AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2
5	Christ the Redemmer	1	2	2

COMMIT?

New

```
AttractionId     = 5
AttractionName  = "Christ the Redemmer"
CountryId = find( CountryId, CountryName = "Brazil" )
CityId  = find( CityId, CityName = "Rio de Janeiro" )
CategoryId  = find( CategoryId, CategoryName = "Monument"
```

endnew
....
Commit

Se tudo estiver bem, então o new insere um registro na tabela.
E o que acontece com o Commit?

Se vamos ver a navegação do procedimento que executamos há pouco, está nos mostrando um aviso que diz que o programa pode ser chamado por outro programa e que a propriedade Commit on Exit está configurada como Yes. Aqui a vemos.
Esta propriedade também se encontra no outro objeto que opera sobre a base de dados: o objeto transação. Aqui a vemos, sob o grupo Transaction integrity (em outra aula estudaremos este tema importante, que é como definir e obter a integridade transacional).

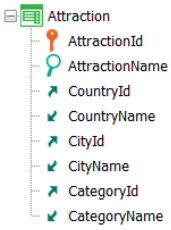
O importante é que se esta propriedade estiver configurada em Yes, isso significa que estará sendo adicionado um Commit automático no código-fonte do objeto (desde que o objeto realize alguma operação sobre a base de dados).

New Command

Na transação, no final da operação sobre o cabeçalho e suas linhas.

New Command

Insert, Update, Delete



AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2
5	Christ the Redemmer	1	2	2

COMMIT?

New

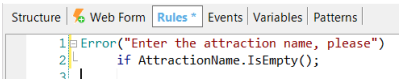
```
AttractionId     = 5
AttractionName  = "Christ the Redemmer"
CountryId = find( CountryId, CountryName = "Brazil" )
CityId  = find( CityId, CityName = "Rio de Janeiro" )
CategoryId  = find( CategoryId, CategoryName = "Monument" )
```

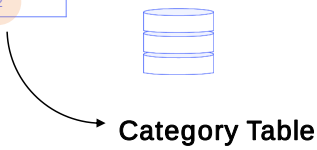
endnew
....
Commit

Em um objeto procedimento, ao final do Source. É por isso que não precisamos especificá-lo. Se a propriedade estivesse desativada, aí sim teríamos que escrever explicitamente o comando Commit, tal como fazíamos com os Business Components.

	Uniqueness check	Referential Integrity check	Rules/Events execution
New command	✓	✗	✗

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2
5	Christ the Redemmer	1	2	2





Resumindo o que vimos até aqui:

Ao tentar inserir um registro em uma tabela utilizando o comando new em um procedimento, o comando realiza controle de unicidade por chave primária e chaves candidatas, garantindo assim que não seja adicionado um registro à tabela que duplique a chave. Se encontra chave duplicada, então não faz nada.

Se algum atributo do registro a ser inserido for chave estrangeira, o comando new não realizará verificação de integridade referencial. Ou seja, não buscará na tabela que é referenciada a existência de um registro com o valor que estamos querendo usar, para apenas assim inserir o registro. MAS se a base de dados tem declarada a integridade referencial, então esta sim realizará a verificação e cancelará o programa se estiver sendo violada a integridade. Se não tem a integridade referencial declarada, então será permitida a inserção do registro, independentemente de violar a integridade.

E com relação às regras e eventos que se encontram na transação associada à tabela, estes claramente não têm aqui nada a fazer. Lembremos que, para o new, só importa a tabela, não de onde vem.

	Uniqueness check
New command	✓

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	3
4	Forbidden city	3	1	2

New

```
AttractionId   = 3
AttractionName = "Eiffel Tower"
CountryId = 2
CityId  = 1
CategoryId  = 3

endnew
```

New

```
AttractionId   = 3
AttractionName = "Eiffel Tower"
CountryId = 2
CityId  = 1
CategoryId  = 3

When duplicate
   for each Attraction
      CategoryId =
   3
   endfor
```

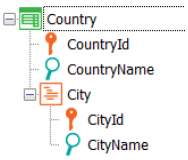
Agora, poderíamos querer realizar alguma ação se o registro se encontra duplicado por alguma chave. Por exemplo, em vez de inseri-lo, atualizá-lo. No exemplo, podemos querer, nesse caso, alterar o valor de CategoryId.

Para fazer isso, contamos com a cláusula when duplicate. O código que inclui só será executado quando for encontrada duplicada a chave primária ou alguma chave candidata. Se o que queremos nesse caso é atualizar o registro, então temos que fazer dentro de um for each [C], como vemos aqui. Não são atribuídos diretamente valores aos atributos que se deseja modificar, como se poderia pensar, mas isto deve ser feito escrevendo um for each, sem ter que filtrar por AttractionId, pois já subentende-se. É a maneira que o new entende que queremos atualizar esses atributos do registro que encontrou duplicado. Neste caso, atualizar apenas o atributo CategoryId. Aqui, poderiam ser atualizados atributos da tabela estendida.

New Command

Not assigned atributes → Depend on context are they instantiated?

```
For each Country.City
    where CountryName = "France"

    print cityInfo

    new

        AttractionName  = CityName + " attra
        CategoryId      = 2

    endnew

endfor
```



CountryId	CityId	CityName
1	1	Sao Paulo
1	2	Rio de Janeiro
2	1	Paris
2	2	Nice
3	1	Beijing

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Mattisse Museum	2	2	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2
5	Paris attraction	2	1	2
6	Nice attraction	2	2	2

Voltando ao assunto do que acontece com os atributos da tabela do new aos quais não se atribui valor. Dissemos que ficam vazios, mas na verdade isto depende do contexto. Ficam vazios se no contexto em que se encontra o new não estão instanciados, ou seja, não têm valor.

Pensemos por exemplo, o caso em que estamos percorrendo com um for each as cidades da França, e para cada uma: a imprimimos, por exemplo, e também, imediatamente, executamos o comando new que estamos vendo.

Primeira coisa: como GeneXus determina qual é a tabela base do new? Atendendo exclusivamente aos atributos aos quais está sendo atribuído valor. Deve encontrar uma tabela da base de dados que os contenha. Claramente será Attraction.

Este CityName participa? Não, não faz isso. Se está ali, é porque, no contexto em que escrevemos o new, sabemos que está instanciado. Esse CityName será o do For each. Em suma, estamos, para cada cidade, querendo inserir uma nova atração turística, com o nome da cidade e a categoria 2.

Bem, mas que identificador de atração, país e cidade será atribuído ao registro que é inserido? Neste caso, o único vazio será o AttractionId, que não está instanciado no contexto, pois não está na tabela estendida do for each. E também não é

recebido por parâmetro no atributo, que é a outra forma de instanciação que conhecemos.
Portanto, se AttractionId se autonumera, então quando o registro for inserido, a base de dados fornecerá o seguinte ao último número.
No entanto, CountryId e CityId estão instanciados no contexto. Portanto, será atribuído esse valor do contexto. Neste caso, o país e a cidade em que estamos posicionados no for each.

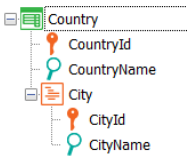
E então o mesmo ocorrerá para a próxima cidade do for each...

New Command

Not assigned atributes → Depend on context are they instantiated?

```
For each Country.City
    where CountryName = "France"

    print cityInfo

    new
        AttractionName =
        CategoryId     = 2
    endnew

endfor
```

WARNING
Base table: CATEGORY!!!!



CountryId	CityId	CityName
1	1	Sao Paulo
1	2	Rio de Janeiro
2	1	Paris
2	2	Nice
3	1	Beijing

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Mattisse Museum	2	2	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2
5		2	1	2
6		2	2	2

O que aconteceria se quiséssemos que fossem inseridos os mesmos dois registros em Attraction, mas com AttractionName vazio?

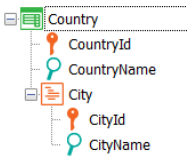
Pareceria que, simplesmente por não atribuir valor a AttractionName, estaríamos implementando-o. Porém, se observamos os atributos que GeneXus vai utilizar para determinar sua tabela base, a do new, só existe CategoryId. Portanto, não escolherá como tabela base Attraction, mas sim Category.

Então, como vamos fazemos para que escolha a tabela Attraction? Uma possibilidade é atribuir explicitamente o valor vazio para AttractionName. Pois desta forma, ao nomear esse atributo, participará da determinação da tabela base e já fará com que a tabela base, ao invés de Category, seja Attraction.

New Command

Not assigned atributes → Depend on context are they instantiated?

```
For each Country.City
    where CountryName = "France"

    print cityInfo

    new Defined by AttractionName

        CategoryId          = 2

    endnew

endfor
```



CountryId	CityId	CityName
1	1	Sao Paulo
1	2	Rio de Janeiro
2	1	Paris
2	2	Nice
3	1	Beijing

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Mattisse Museum	2	2	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2
5		2	1	2
6		2	2	2

A outra opção é utilizar a cláusula Defined by que permite agregar mais atributos a serem considerados junto com os demais na determinação da tabela base.

Summary

new
Defined by *Attribute*$_1$, *Attribute*$_2$, ..., *Attribute*$_N$
Blocking *NumericExpression*

Attribute$_1$ = *expression*$_1$
Attribute$_2$ = *expression*$_2$
...
Attribute$_N$ = *expression*$_N$

When duplicate
...
for each *BaseTransaction*
Attribute$_I$ = *expression*$_I$
Attribute$_K$ = *expression*$_K$
...
endfor
...

	Uniqueness check	Referential Integrity check
New command	✓	✗

COMMIT

Transaction integrity	
Commit on exit	Yes

Em resumo, o comando new é utilizado para inserir um registro em uma tabela. A tabela é determinada de acordo com os atributos que aparecem atribuídos. Se for adicionada cláusula Defined By, então os atributos que aparecem ali listados também participam. Aqui, o conceito de tabela estendida não tem nenhum sentido.

Por outro lado, vimos que o único controle programático que realiza o new é o controle de unicidade. E que se fosse encontrado um registro com a chave que estamos tentando inserir, então o new não fazia nada.... A menos que tivéssemos programado a cláusula when duplicate.

E havíamos dito que ali, entre outros comandos, podemos incluir um for each para atualizar atributos do registro que foi encontrado duplicado. Todos os atributos podem ser atualizados? Por exemplo, a chave primária pode ser atualizada ali? A resposta é não. Mas será possível atualizar os atributos da tabela estendida.

Por último: para que o registro seja commitado na base de dados, devemos nos certificar de que o comando Commit seja executado. Em um procedimento, por padrão, é colocado um Commit implícito ao final. Mas podemos escrever explicitamente Commits no Source, onde nos convenha.

Não veremos aqui, mas opcionalmente pode ser especificada uma cláusula de Blocking, que o que faz é permitir fazer inserções na base de dados em bloco, em vez de registro por registro, desde que o new esteja dentro de uma estrutura repetitiva. Isto, claramente, será por razões de eficiência, quando as inserções batch sejam muito grandes.

GeneXus™

training.genexus.com
wiki.genexus.com