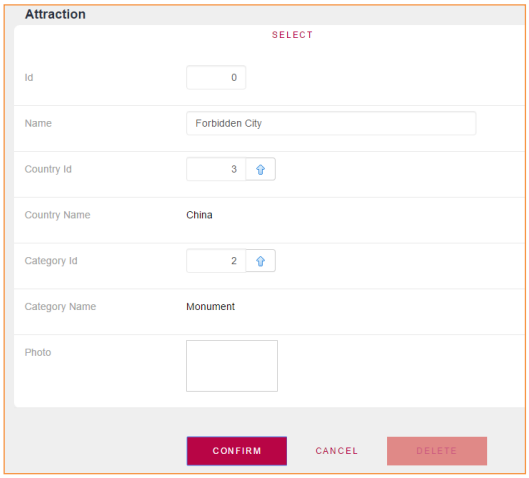
Atualização do banco de dados com comandos específicos de procedimentos

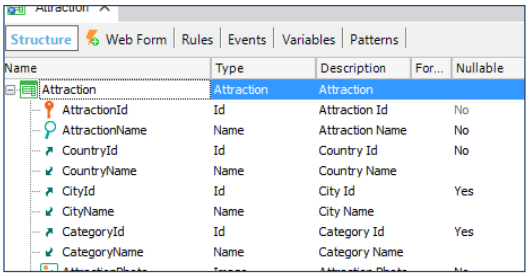
GeneXus™

Atualizando a base de dados

Usando a Transação



Usando um Business Component




```
Event 'New Attraction'
    &Attraction.AttractionName = "Forbidden City"
    &Attraction.CountryId = find( CountryId, CountryName = "China")
    &Attraction.CityId = find( CityId, CityName = "Beijing")
    &Attraction.CategoryId = find( CategoryId, CategoryName = "Monument")
    &Attraction.Save()
    Commit
Endevent
```

Até agora, para atualizar o banco de dados usamos transações das 2 maneiras disponíveis para fazê-lo:

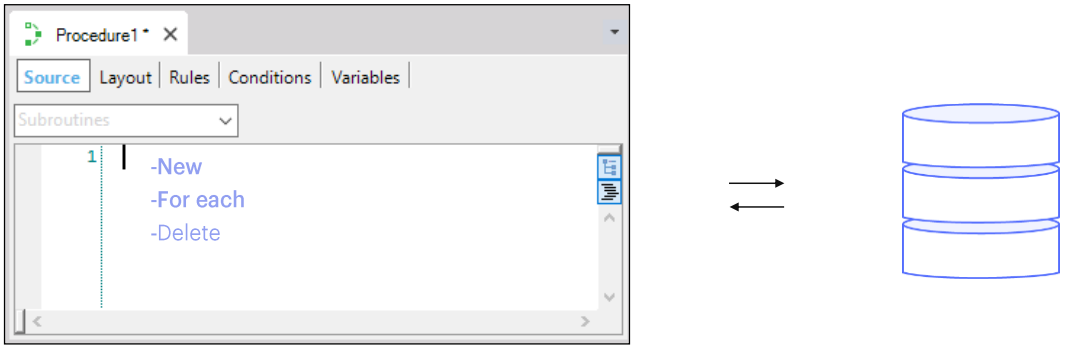
Executando suas telas e inserindo dados de forma interativa
Executando-as como Business Components, através de uma variável, sem o uso de suas telas.

Por exemplo, para adicionar uma nova atração, em um web panel poderíamos ter colocado um botão e escrito o que vemos acima para seu evento.

Outra maneira de atualizar o banco de dados

Objeto Procedure

DataBase



Agora vamos aprender sobre outra opção disponível para fazer inserções, alterações e exclusões no banco de dados.

Nós devemos levar em conta que o que vamos ver, só pode ser usado em objetos de tipo procedimento, ao contrário da opção em que usamos um Business Component, que pode ser usado em qualquer objeto.

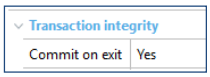
Válido apenas em procedures

INSERÇÃO

New Atributos da tabela

AttractionId	AttractionName	CountryId	CityId	CategoryId	AttractionPhoto
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	

EndNew



Usando este comando, nós podemos atribuir valores para os atributos de uma tabela física para inserir um novo registro.

Procedimentos fornecem um comando chamado New para inserir registros em uma tabela.

Usando este comando pode atribuir valores aos atributos de uma tabela física.

Estamos falando de uma tabela, não de uma transação, porque nem todos os atributos em uma estrutura de transação estão incluídos na tabela física.

Por exemplo, se queremos inserir uma atração, na transação Attraction muitos atributos são declarados na estrutura, mas não estão fisicamente presentes na tabela ATTRACTION. Em vez disso, eles estão na tabela estendida da tabela ATTRACTION . Nós os incluímos na estrutura para mostrá-los no form ou usá-los nas regras.

Atributos da tabela Attraction e comando New

Transaction Attraction

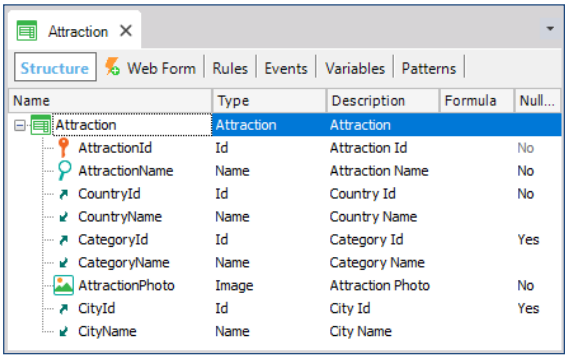
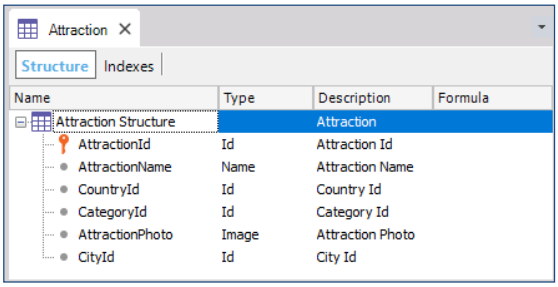


Table Attraction



O comando NEW apenas permite atribuir valores para estes atributos

Se selecionamos View/Tables e editamos a tabela ATTRACTION , veremos apenas os atributos que pertencem à tabela ATTRACTION .

Podemos incluir estes atributos em um comando New e atribuir valores a eles. O comando New irá inserir apenas um registro na tabela.

Em procedures...

INSERÇÃO

```
new
    AttractionName = "Forbidden City"
    CountryId = find( CountryId, CountryName = "China" )
    CityId = find( CityId, CityName = "Beijing" )
    CategoryId = find( CategoryId, CategoryName = "Monument" )
endnew
```

TABELA BASE

ATTRACTION

AttractionId: Nós não colocamos valor para o atributo AttractionId porque este é autonumber

AttractionPhoto: O registro será inserido aqui, sem o campo foto.

GeneXus irá determinar a tabela física para inserir o registro, analisando os atributos localizados à esquerda do sinal de igual.

Se todos eles pertencem à mesma tabela física, o registro será inserido na tabela. Caso contrário, seremos informados que não é possível selecionar uma tabela para fazer a inserção.

A tabela encontrada por GeneXus é a tabela base do comando New. Neste caso: ATTRACTION.

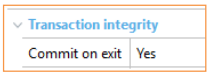
Note que não estamos atribuindo uma imagem para o atributo AttractionPhoto. Neste caso, a inserção será feita, mesmo que o atributo AttractionPhoto não seja atribuído e a atração será criada sem uma foto.

Além disso, poderíamos ter deixado os atributos CountryId, CityId e CategoryId, que são chaves estrangeiras, sem valores atribuídos. Ou, pelo contrário, em vez de usar a fórmula Find para procurar os IDs correspondentes, poderíamos digitar IDs inexistentes porque o New não verifica a integridade referencial. Por esta razão, nós deve ser muito cuidadosos quando programamos este comando .

Inserção com New versus BC

Usando a cláusula NEW em Procedures

```
new
    AttractionName = "Forbidden City"
    CountryId = find(CountryId, CountryName = "China")
    CityId = find(CityId, CityName = "Beijing")
    CategoryId = find(CategoryId, CategoryName="Monument")
endnew
```



Usando Business Components

```
Event 'New Attraction'
    &Attraction.AttractionName = "Forbidden City"
    &Attraction.CountryId = find( CountryId, CountryName = "China")
    &Attraction.CityId = find( CityId, CityName = "Beijing")
    &Attraction.CategoryId = find( CategoryId, CategoryName = "Monument")
    &Attraction.Save()
    Commit
Endevent
```

Ao executar a inserção, GeneXus só irá verificar que não há registros inseridos com chaves primárias que já existem. Além disso, se houver um índice unique definido sobre um atributo, ele verifica que não há valores duplicados inseridos para este atributo. Se isso acontecesse, o comando New não faria nada e não avisaria o usuário também. Em seguida, vamos ver como identificar esta situação e tomar as medidas correspondentes. O controle de unicidade é o único controle que faz, então controles de integridade referencial NÃO são feitos. Portanto, se atribuímos um CountryId inexistente, não haverá uma falha de integridade referencial, ao contrário do que acontece quando a inserção é feita através de um Business Component. Lembre-se que, neste caso, unicidade e controles de integridade referencial são feitos. Além disso, as regras definidas na transação são acionadas.

Quando a inserção é feita usando um Business Component, como ela pode ser feita de qualquer objeto, nós devemos explicitamente executar o Commit; isso quer dizer, dar a ordem para fazer o banco de dados a manter os dados alterados desde a última operação de Commit.

Os procedimentos têm a propriedade Commit on exit que é definida como Yes, por padrão. Que efeito isso tem? Se o procedimento acessa o banco de dados, um comando Commit será adicionado ao final do código sem necessidade para o desenvolvedor de indicá-lo. Na verdade, não é escrito no objeto e é em vez disso adicionado ao programa gerado. É por isso que não precisamos adicionar explicitamente um Commit após o comando New , a menos que queremos executar a operação de

confirmação imediatamente.

GeneXus percebe que ele tem que acessar o banco de dados dentro de um procedimento quando usando New, For Each para atualização, delete (operações que veremos mais adiante). Nestes casos, ele acrescenta o Commit implícito; caso contrário, ele não entende que o banco de dados precisa ser acessado e não o adiciona. Portanto, as operações de manipulação de banco de dados por meio de business components não resultarão em commit implícito.

Por esta razão, quando digitamos:
&BC.Element1 =...
&BC.Element2 =...
& BC. Save)

ele não adiciona o comando Commit, porque ele não sabe que nós queremos fazer uma inserção (nem mesmo se usássemos o método Insert diretamente). Como consequência, ele lida com o Business Component como se fosse qualquer SDT. Neste caso, o comando Commit tem que ser escrito explicitamente.

Se, dentro do código do procedimento, houver também um New, For Each que atualiza, ou Delete, em qualquer um dos casos não teríamos que escrever explicitamente o Commit , porque já estabeleceu uma conexão com o DB. Se só temos o código anterior no nosso procedimento, então teríamos que adicionar o comando Commit explicitamente.

Somente válido em procedures

INSERÇÃO

```
New
    CategoryName = "Tourist site"
EndNew
```

TABELA BASE

CATEGORY

CategoryId (¿?)

Em uma cláusula New, para criar um novo registro podemos atribuir valores para os atributos que pertencem a um registro da tabela.

Se tivéssemos a cláusula New acima, escrita em um procedimento, onde apenas atribuímos um valor para o atributo CategoryName, obviamente, a tabela base será CATEGORY, porque é a tabela onde o atributo CategoryName é armazenado fisicamente.

Mais uma vez, não atribuímos um valor para o atributo CategoryId porque é autonumerado.

Somente válido em procedures

ATUALIZAÇÃO

```
For each Attraction
    Where CityName = 'Beijing'
    Where CategoryName = 'Monument'
    CategoryId = find(CategoryId, CategoryName = 'Tourist site')
Endfor
```

TABELA BASE

ATTRACTION

Os atributos da tabela estendida de ATTRACTION podem ser atualizados no comando For Each (exceto a chave primária).

Agora vamos ver como podemos atualizar um valor existente no banco de dados.

Para substituir um valor armazenado em um atributo - de um registro da tabela - com outro valor, usamos um comando For Each para acessar o registro e dar-lhe o novo valor através de uma atribuição.

No exemplo, a tabela de base do comando For Each é ATTRACTION, porque a transação base é Attraction e estamos percorrendo as atrações em Beijing, que têm a categoria de “Monument” . Vamos mudar a categoria deste para " Tourist Site ".

Observe que neste exemplo estamos atualizando muitos registros, que são todos aqueles que atendem às condições da cláusula Where. Além disso, estamos atualizando o valor de um único atributo, mas poderiam ser muitos. Até mesmo atributos da tabela estendida podem ser atualizados; por exemplo, o nome da categoria da atração, o nome do país da atração, o nome da cidade da atração. Vamos ver isto na página seguinte .
Observe que o comando New só permite atribuir valores para atributos que estão fisicamente presentes na tabela, porque lá nós queremos inserir apenas um novo registro. No comando For Each , por outro lado, estamos sempre posicionados em um registro existente e a partir daí podemos editar todos os registros associados através de uma tabela estendida.

Se tivéssemos uma chave candidata (quer dizer, um índice unique

definido sobre um dos atributos) e dentro do comando For Each estivéssemos substituindo seu valor, com um já existente para outro registro, a atualização não seria feita. Então, como já dissemos para o comando New, aqui não vemos qualquer aviso também. Em seguida, veremos como capturar este caso e fazer algo a respeito.

Há uma restrição que não permite alterar, dentro de um comando For Each, a chave primária da tabela sendo executada.

Somente válido em procedures

ATUALIZAÇÃO

Trn Invoice

Invoice	Invoice
InvoiceId	Id
InvoiceDate	Date
CustomerId	Numeric(4.0)
CustomerName	Character(20)
CustomerTotalPurchases	Price
InvoiceTotalAmount	Price
Flight	Flight
FlightId	Id
FlightAvailableSeats	Numeric(4.0)
InvoiceFlightSeatQty	Numeric(4.0)
FlightFinalPrice	Price
InvoiceFlightAmount	Price

```
For each Invoice
    Where InvoiceDate >= &LastDate
    if count(InvoiceFlightSeatQty) >= 5
        CustomerVIP = True
    endif
Endfor
```

Trn Customer

Customer	Customer
CustomerId	Id
CustomerName	Name
CustomerAddress	Address
CustomerPhone	Phone
CustomerEMail	Email
CustomerAddedDate	Date
CustomerVIP	Boolean

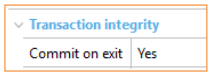
A agência de viagens quer identificar aqueles clientes que, a partir de uma determinada data, na mesma fatura compraram mais de cinco voos e marcar os clientes como VIP. Para fazer isso, nós adicionamos o atributo CustomerVIP na transação Customer; este atributo será armazenado na tabela associada.
Para identificar e marcar aqueles clientes, teremos de percorrer a tabela de Invoice e contar os voos em cada uma delas. Se esse número for superior a cinco voos, devemos atualizar o valor do atributo CustomerVIP, definindo-o como True.

Aqui a tabela base do comando For Each é INVOICE, então a fórmula Count só contará os voos que pertencem a esta fatura. Se eles são mais do que 5, note que estamos atualizando o valor de um atributo da tabela CUSTOMER , que é na tabela estendida da tabela que estamos percorrendo.

Atualizando usando o comando For Each vs BC

Usando o comando For Each em uma Procedure

```
For each Attraction
    where CityName = "Beijing" and CategoryName = "Monument"
    CategoryId = find( CategoryId, CategoryName = "Torist Site")
endfor
```



✓ Controle de unicidade de PK e CK

Usando Business Components

```
For each Attraction
    Where CityName = "Beijing" and CategoryName = "Monument"
    &Attraction.Load(AttractionId)
    &Attraction.CategoryId = find( CategoryId, CategoryName = "Tourist site")
    &Attraction.Save()
Endfor
Commit
```

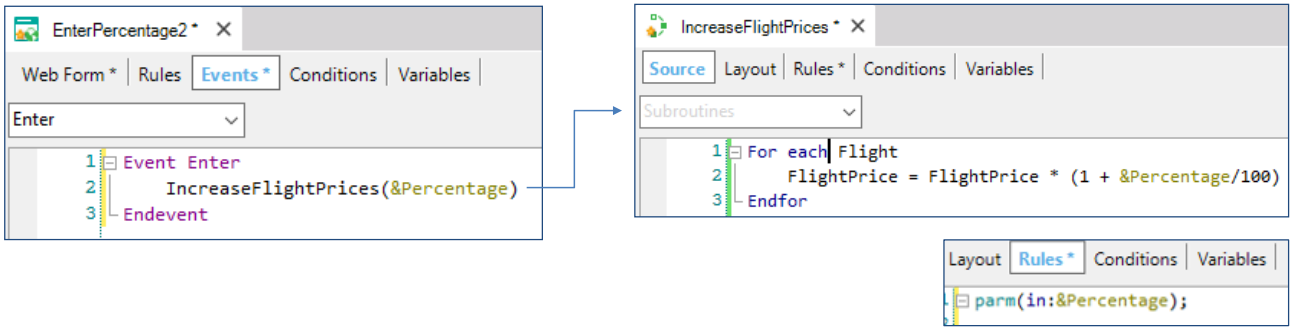
✓ Controle de unicidade de PK e CK
✓ Controle de IR
✓ Regras

Como já dissemos antes, quando um comando For Each é usado dentro do procedimento, o único controle feito pelo GeneXus será um controle de unicidade para a chave primária e chaves candidatas.

Por outro lado, a solução que usa Business Components também verificará a integridade referencial. Portanto, se atribuímos um valor inexistente para CategoryId, ele não permitiria fazer esta atualização. Além disso, ele irá executar as regras de negócios.

Lembre-se o que já dissemos sobre o comando New para Commit on exit..

Procedure para aumento de preços



Queremos solicitar ao usuário que insira uma porcentagem e recalcular o preço de todos os voos, aplicando este percentual de aumento.

Para isso, criamos um web panel para pedir esses dados ao usuário e de lá nós invocamos um procedimento que efetivamente irá atualizar a tabela de voos, para alterar o valor do preço de acordo com a porcentagem que recebeu em um parâmetro.

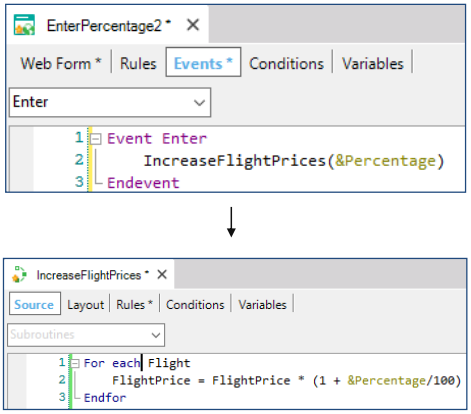
Para fazer essa atualização, nós usamos o comando For Each para percorrer a tabela FLIGHT e substituir o valor do atributo de FlightPrice com o resultado da expressão exibida.

Comparando alternativas

Usando Business Components

```
Event Enter
    For each Flight
        &BCFlight.Load(FlightId)
        &BCFlight.FlightPrice = &BCFlight.FlightPrice * (1+ &Percentage/100)
        &BCFlight.Save()
        If &BCFlight.Success()
            Commit
        Else
            Rollback
        Endif
    Endfor
Endevent
```

Usando a Procedure



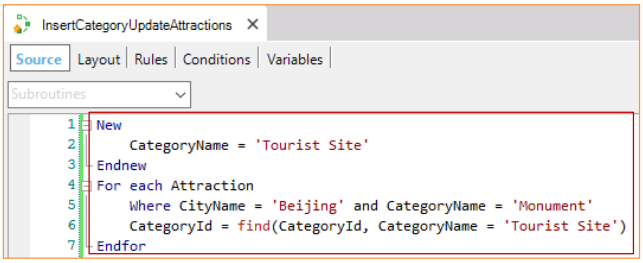
Vamos comparar as duas alternativas que podemos usar para atualizar o banco de dados.

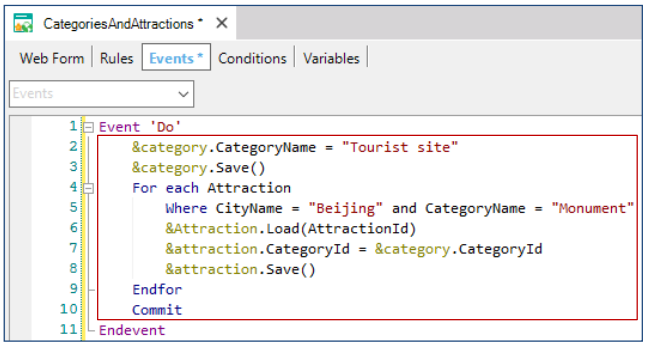
Na primeira alternativa implementamos, no evento Enter do web panel EnterPercentage, nós digitamos o comando For Each e atualizamos o banco de dados usando a transação Flight como um Business Component.

Na segunda solução, o evento Enter do web panel EnterPercentage2 contém apenas uma chamada para um procedimento, que recebe o valor da porcentagem, navega em todos os voos com um comando For Each e, para cada um deles, calcula e atribui o novo preço.

Quais são as diferenças, vantagens e desvantagens de resolver isto de uma maneira ou de outra?

Exemplo de Insert Category + mudanças em Attractions





Exclusivamente en Procedure

Puede estar en cualquier objeto

Aqui podemos ver as duas soluções para inserir uma nova categoria e substituir o valor de categoria de certas atrações com esta categoria.

Observe que o código no evento do web panel é executado em qualquer objeto GeneXus, e o código no procedimento só é válido dentro de um objeto procedimento.

Vamos examinar o procedimento:
Como CategoryId é autonumerado, o comando New nunca falhará. Estamos usando uma fórmula Find para localizar a ID da categoria chamada "Tourist Site" que acabamos de inserir com o comando New. Se tivéssemos cometido um erro ao digitar o nome da categoria, a fórmula Find não teria encontrado o registro e teria retornado um valor vazio, que é 0 para um valor numérico. Neste caso, mudaríamos o ID da categoria das atrações em Beijing que eram monumentos para valor 0. Se o atributo CategoryId não aceita valores nulos, uma inconsistência seria criada no banco de dados. Lembre-se que não são feitas verificações de integridade referencial quando o comando For Each é usado para fazer uma atualização, ao contrário do que acontece quando Business Component é usado, porque nesse caso seria verificada a integridade referencial. No nosso caso, CategoryId aceita valores nulos, então não tereríamos problemas de consistência ao usar esta opção através de um procedimento.

Atualização de valores através de Business Components é sempre a opção mais segura, porque eles farão os mesmos controles, como os

comando New e For Each e ainda mais: verifica integridade referencial, além das regras indicadas a transação associada. Considere que, mesmo se você teve cuidado, quando o programou o procedimento, sua realidade associada pode mudar a qualquer momento. Por exemplo, a necessidade de adicionar regras de erro para a transação que você não pensou quando programou o procedimento. O uso de um Business Component garante que essas regras serão incluídas em todas as verificações.

Somente válido em procedures

DELEÇÃO:

```
For each Category
    where CategoryName = "Tourist Site"
    Delete
Endfor
```

```
For each Attraction
    Delete
Endfor

For each Category
    Delete
Endfor
```

Já vimos como procedimentos nos permitem inserir e atualizar registros no banco de dados. Agora, vejamos como excluir registros.

Para excluir registros, temos o comando Delete que é usado dentro de um comando For Each .

Basicamente, como o comando Delete remove o registro da tabela onde estamos posicionados, o comando For Each é usado para acessar o registro.

No primeiro exemplo, vemos que estamos navegando a tabela Category , filtrando por categoria "Tourist Site"; Portanto, só está apagando um registro com o comando Delete.

Mas poderíamos ter também eliminado todas as atrações e todas as categorias (se queríamos esvaziar essas tabelas).

Se excluirmos as categorias primeiro e as atrações mais tarde, uma vez que o banco de dados verifica se há integridade referencial por padrão, quando tentamos excluir a primeira categoria, se tem uma atração relacionada falhará e o programa irá falhar também, como veremos em seguida.

Usando comandos especiais (NEW, FOR EACH e DELETE)

Validação de dados : apenas unicidade e não integridade referencial.

Apenas válido em procedures.

Usando business components:

Validação de dados : tanto unicidade quanto integridade referencial.

As regras das transações são disparadas.

Válido em qualquer objeto.

Quando usamos Business Components, os dados que serão atualizados no banco de dados são verificados para a consistência, e as regras declaradas na transação que está sendo executada como um Business Component são executadas.

Essas regras que geram mensagens, como msg e error, também são acionadas e as mensagens correspondentes são salvas em uma coleção que pode ser percorrida e impressa.

Em um procedimento, nenhuma dessas ações são executadas.

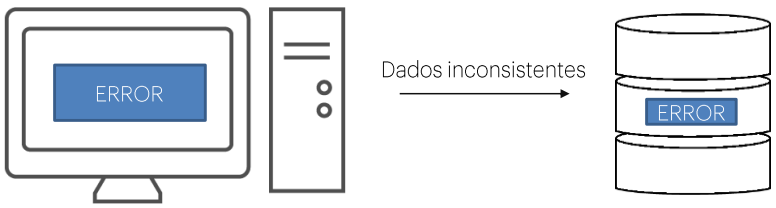
Algo que já falamos, e vale ressaltar novamente, é que mesmo se o comando For Each pode ser usado em um web panel, não pode ser usado para alterar o banco de dados diretamente, atribuindo valores para atributos, e que só pode ser feito a partir de um objeto procedimento. Não é possível codificar um comando New em um web panel também, ou incluir um comando Delete dentro do comando For Each. Isso só pode ser feito em procedimentos.

No entanto, em qualquer objeto, é possível alterar o banco de dados usando Business Components.

Devemos levar em conta que os procedimentos não verificam consistência nos dados que estão sendo atribuídos. Eles apenas verificam as chaves primárias e candidatas por unicidade.

Usando comandos especiais (NEW, FOR EACH e DELETE)

Nós seremos responsáveis por atribuir dados válidos e bem relacionados.



Ao usar esses comandos para fazer atualizações diretas nos procedimentos, é nossa responsabilidade atribuir ou inserir dados que são consistentes com o resto dos dados armazenados.

No exemplo anterior do comando New , nós podemos atribuir qualquer valor para CategoryName porque esses dados não estão relacionados com quaisquer outras tabelas.

No exemplo onde atualizamos a categoria de atrações, pelo contrário, estávamos atribuindo um novo valor para um atributo que era uma chave estrangeira: CategoryId. Se o valor atribuído não existia como um identificador de categoria na tabela CATEGORY , o procedimento não iria validá-lo, e por isso poderíamos estar incluindo dados inconsistentes.

Como bancos de dados verificam a consistência de dados inter-relacionados, quando o usuário executa a aplicação e tenta atribuir um valor incompatível, o banco de dados irá rejeitar a operação e os dados inconsistentes não serão salvos.

No entanto, o programa parará de funcionar, e isto não é amigável para o usuário.

Portanto, se usarmos procedimentos para atualizar o banco de dados, nós seremos responsáveis por atribuir dados válidos e bem relacionados.

Usando comandos especiais (NEW, FOR EACH e DELETE)

Já que somente é validado a unicidade dos dados, onde podemos programar ações caso seja tentado duplicar a chave primária ou chave candidata?

```
new
    AttractionName = "Forbidden City"
    CountryId = find( CountryId, CountryName = "China")
    CityId = find( CityId, CityName = "Beijing")
    CategoryId = find( CategoryId, CategoryName = "Tourist Site")
when duplicate

endnew
```

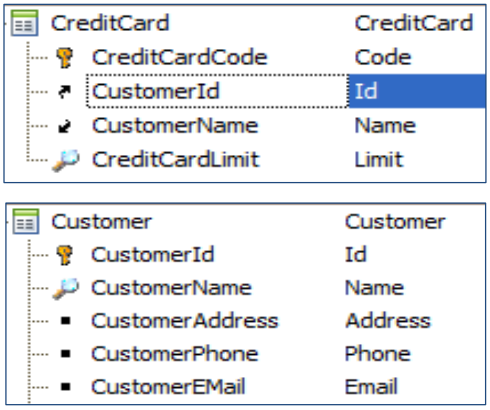
Se AttractionName é uma chave candidata, isso quer dizer, não pode ser repetida, e executamos um comando New para inserir um novo registro (com ID autonumerado) atribuindo o valor "Forbidden City" para AttractionName e já existe um registro com esse valor, podemos fazer algo a respeito, que é programar dentro do comando New a cláusula When duplicate.

Usando comandos especiais (NEW, FOR EACH e DELETE)

Duplicação de chave primária ou chave candidata

```
for each CreditCard
    where CreditCardCode = &creditCardcode
    CustomerId = &newCustomerId
when duplicate

endfor
```



Do mesmo modo, suponha que temos duas transações chamadas Customer e CreditCard que têm uma relação de 1 para 1: isso quer dizer, CustomerId foi definido como uma chave candidata (através de um unique index), e em um procedimento que recebe duas variáveis através de um parâmetro - o identificador de um novo cliente e o identificador de um cartão de crédito existente – um comando For Each é programado para alterar aquele cartão do cliente para um novo. Se o banco de dados já tiver outro cartão para o mesmo cliente, o comando For Each não fará a atualização. Se nós queremos executar uma ação, nesse caso, podemos usar a cláusula When duplicate .

Para saber mais sobre esta cláusula, por favor visite o nosso wiki: http://wiki.genexus.com/commwiki/servlet/wiki?24843,When+duplicate+clause.

Os atributos incluídos na cláusula When duplicate não serão levados em conta ao determinar a tabela base do comando For Each na qual estão situados.

Sintaxe do For each

```
For each   BaseTransaction
      skip expression1 count expression2
      order  att1, att2, ... , attn [when condition]
      order  att1, att2, ... , attn [when condition]
      using DataSelector( parm1, parm2, ... , parmn)
      unique att1, att2, ... , attn.
      where  condition [when condition]
      where  condition [when condition]
      where  att  IN DataSelector( parm1, parm2, ... , parmn)
      blocking N
        main code
When duplicate
      ...
When none
      ...
endfor
```

A cláusula blocking permite atualizar o banco de dados em grupos de N registros , para reduzir o número de acessos do banco de dados e melhorar o desempenho. Não falamos sobre isso neste curso. Par a obter mais informações, leia o artigo aqui:

http://wiki.genexus.com/commwiki/servlet/wiki?4837,Blocking+clause+in+a+%27For+each%27+command.

Para obter mais informações sobre a cláusula When duplicate, que é válida para ambos, comando For Each e New , clique neste link: http://wiki.genexus.com/commwiki/servlet/wiki?24843,When+duplicate+clause.

Para a sintaxe completa do comando New , clique neste link: http://wiki.genexus.com/commwiki/servlet/wiki?6714,New+command.

Lembre-se que os atributos incluídos no código escrito dentro da cláusula When duplicate não serão levados em conta ao determinar a tabela base do comando For Each , em que estão situados.

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