

# Exercícios práticos

**GeneXus™ 16**

Março 2019

*Copyright © GeneXus S.A. 1988-2019.*

*All rights reserved. This document may not be reproduced by any means without the express permission of GeneXus S.A. The information contained herein is intended for personal use only.*

**Registered Trademarks:**

*GeneXus is trademark or registered trademark of GeneXus S.A. All other trademarks mentioned herein are the property of their respective owners.*

## CONTEÚDO

CONTEÚDO .....	2
O PROBLEMA.....	4
NOVO PROJETO, NOVA BASE DE CONHECIMENTO .....	4
PRIMEIRAS TRANSAÇÕES.....	5
Transação “Employee” .....	5
Transações “AmusementPARK” e “country”, relacionadas.....	7
Dados relacionados: como é mantida a integridade? .....	10
Transação ‘Game’.....	11
Transação ‘Category’ .....	11
Transações “Employee” e “AmusementPark”, relacionadas .....	13
Adicionemos as cidades à transação ‘Country’ .....	14
Transação “AmusementPark”: adicionemos a cidade.....	16
ADICIONEMOS COMPORTAMENTO ÀS TRANSAÇÕES (RULES).....	17
PATTERNS: MELHORARANDO A INTERFACE PARA TRABALHAR COM INFORMAÇÃO .....	18
TRANSAÇÕES “REPAIR” E “TECHNICIAN” E NECESSIDADE DE DEFINIR SUBTIPOS.....	21
FÓRMULAS .....	22
CRIAÇÃO DE SEGUNDO NÍVEL.....	23
LISTAS PDF.....	25

PASSAGEM DE PARÂMETROS.....	30
Lista de parques em um intervalo determinado .....	30
<b>BUSINESS COMPONENTS .....</b>	<b>31</b>
Aumento de preço dos reparos.....	31
Tela para eliminação de todos os reparos .....	33
<b>PROCEDIMENTOS PARA ATUALIZAR REGISTROS.....</b>	<b>34</b>
Aumento de preços dos reparos .....	34
Eliminação de todos os reparos.....	35
Inicialização da informação da base dde dados [opcional] .....	38
<b>WEB PANELS .....</b>	<b>39</b>
<b>EXTENDED CONTROLS .....</b>	<b>40</b>
<b>OBJETO QUERY.....</b>	<b>40</b>
<b>PARTE PARA SMART DEVICES .....</b>	<b>41</b>
<b>GENEXUS SERVER.....</b>	<b>42</b>

## O PROBLEMA

Uma multinacional responsável por gerenciar parques de diversões o contrata para desenvolver um sistema para armazenar e manipular as informações com as quais trabalha. Imagine que o sistema é composto por dois módulos:

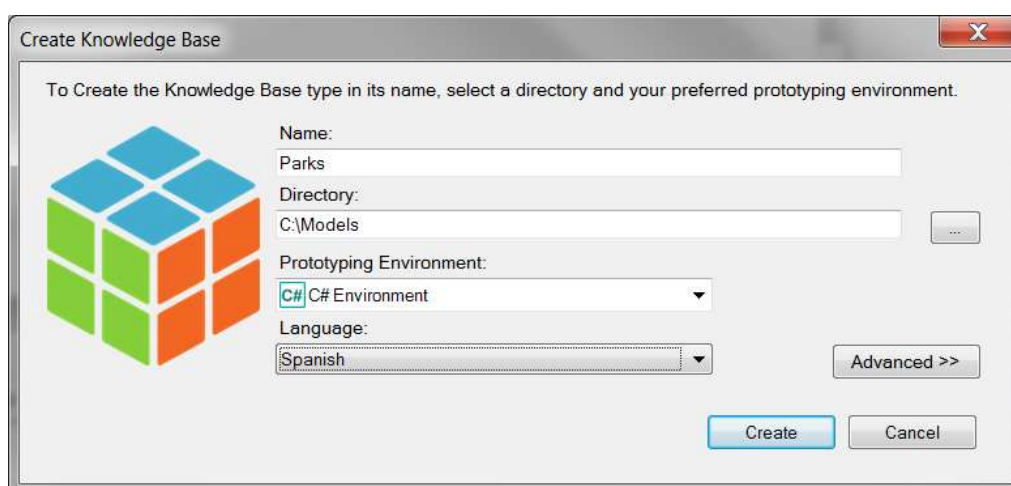
- **Backend:** parte da aplicação que deverá ser executada em um servidor web, de forma que os funcionários da empresa possam manipular as informações a partir de qualquer lugar com uma conexão à Internet.
- **Aplicação simples para dispositivos móveis:** parte da aplicação que será destinada ao download pelos clientes da empresa, a qual lhes permitirá consultar os parques disponíveis, bem como os principais parques de diversões que cada cidade oferece e suas atrações.

## NOVO PROJETO, NOVA BASE DE CONHECIMENTO

Entrar no GeneXus e criar uma base de conhecimento chamada *Parks* para iniciar o desenvolvimento da aplicação.

Sugerimos:

- Escolher como ambiente de desenvolvimento C#. Certifique-se de ter instalado tudo o que você precisa (incluindo o SQL Server). Se usa GeneXus Trial, o ambiente de geração com C# e SQL Server já está predefinido, prototipando na nuvem da Amazon.
- Não criar a base de conhecimento na pasta "Meus Documentos" ou qualquer outra pasta que esteja sob "Documents and Settings", porque essas pastas têm permissões especiais concedidas pelo Windows.



Reserve alguns minutos para **se familiarizar** com o **IDE (ambiente de desenvolvimento integrado do GeneXus)**. Tente **mover janelas, visualizar janelas específicas que você deseja** (View e View/Other Tool Windows) e observe atentamente o conteúdo da janela **KBExplorer** (Knowledge Base Explorer). Você verá que aparecem já inicializados **domínios**, alguns **objetos, imagens**, etc.

Sugestão: mantenha a janela de propriedades aberta (**F4**), pois você a usará continuamente. Dentro da janela **'Preferences'**, onde se configura o **'Environment'**.

## PRIMEIRAS TRANSAÇÕES

Nas reuniões com a empresa, lhe transmitem o seguinte:

“Registramos os dados dos parques de diversões, para gerenciar tanto seus funcionários quanto suas atrações e atividades oferecidas aos visitantes”.

Para começar a construir a aplicação, precisamos começar identificando os atores da realidade e representá-los por meio de **transações**. Quais transações devemos criar então na base de conhecimento (KB)?

### TRANSAÇÃO “EMPLOYEE”

Perguntamos: quais dados registram dos funcionários da empresa? A resposta é a seguinte:

O **nome** (que não excede 20 caracteres), **sobrenome** (que também não os excede), **endereço**, **telefone** e o **e-mail**.

Com estes dados já pode criar a transação *Employee*.

#### Lembrar que:

- Existem várias alternativas para criar objetos:
  - Fazê-lo a partir do menu: File/New/Object
  - Ctrl+N
  - Ícone da barra de ferramentas
- Digitando ponto (“.”) quando inserir um novo atributo, este é iniciado com o nome da transação.
- Necessitará de um atributo que identifique cada empregado (EmployeeId).

A estrutura da transação deveria ter ficado conforme mostrado:

Name	Type
Employee	Employee
EmployeeId	Numeric(4,0)
EmployeeName	Character(20)
EmployeeLastName	Character(20)
EmployeeAddress	Address, GeneXus
EmployeePhone	Phone, GeneXus
EmployeeEmail	Email, GeneXus

#### Lembrar que:

- Address, Phone e Email** são **domínios semânticos** que são atribuídos automaticamente aos atributos que são definidos contendo em seus nomes os textos Address, Phone ou Email respectivamente.
- Quando está definindo o tipo de dados do atributo identificador, em vez de usar diretamente Numeric(4,0), defina o **domínio Id** com esse tipo de dados. Configure a propriedade **Autonumber** desse domínio como **True**, para que todos os atributos baseados nele sejam numerados automaticamente, sem que o usuário precise se preocupar.

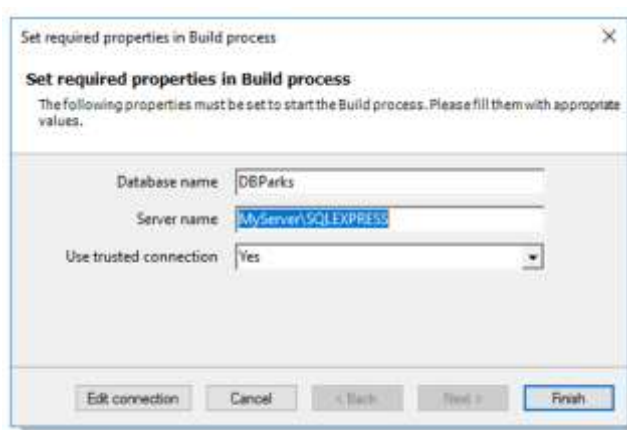
O próximo passo é testar a aplicação em execução. Certifique-se de ter a janela *Output* do GeneXus ativada e visível. (*View/Other Tool Windows /Output*).

Agora sim, teste a aplicação em execução pressionando **F5**.

O que acontecerá?

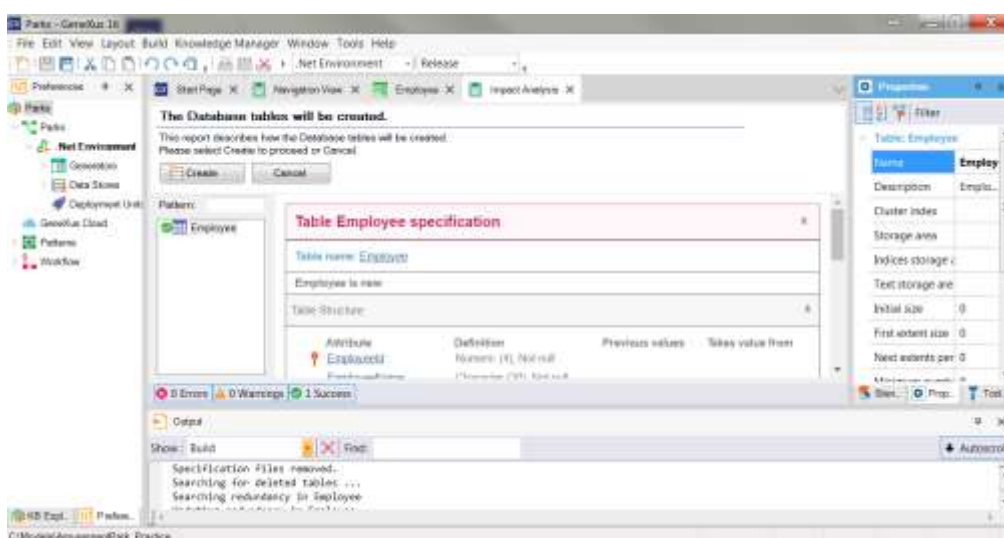
### Solução

Se você decidir criar a base de dados e os programas **localmente**, uma janela como a seguinte será aberta para você inserir as informações da Base de Dados, Servidor e método de conexão. Lembre-se que, se não houver uma base de dados com o nome indicado nesse servidor, GeneXus a **cria**.



Se, em vez disso, a base de dados e os programas forem criados na nuvem, o diálogo anterior não aparece porque GeneXus conhece os dados do servidor na nuvem e configura automaticamente o nome da base de dados e todas as informações de conexão para ela.

Em seguida, é exibida uma **Análise de Impacto** detalhando que será criada a base de dados e a tabela *EMPLOYEE* dentro dela:



Se pressionar o botão **Create**, GeneXus continuará a executar o programa que realizará a criação. No final do processo, será aberto no navegador que foi configurado como padrão, o menu com links para executar os objetos definidos. Neste caso, apenas um: a transação *Employee*.

**Insira** alguns funcionários no sistema. Em seguida, **modifique** algum dado de algum dos funcionários inseridos anteriormente e **elimine** algum funcionário.

Além disso, teste o uso das setas oferecidas para passar de registro em registro de funcionários e a opção *SELECT*, que oferece uma “lista de seleção” para visualizar a lista de funcionários registrados e selecionar um.

Agora vamos identificar e criar a transação seguinte. Lembremos do que nos haviam dito, para o qual foram feitas algumas adições:

“Registramos os dados dos **parques de diversões** de diferentes **idades** em diferentes **países**, para gerenciar tanto seus **funcionários** quanto suas **atrações** e **atividades** oferecidas aos visitantes”.

#### TRANSAÇÕES “AMUSEMENTPARK” E “COUNTRY”, RELACIONADAS

Perguntamos aos funcionários da empresa: que dados registram dos parques de diversões com os quais trabalham? A resposta é a seguinte:

O **nome** (que não excede os 40 caracteres), **site web** (que não excede os 60 caracteres), **endereço** e **foto representativa**.

Com estes dados, já pode criar a transação *AmusementPark*.

Name	Type
AmusementPark	AmusementPark
AmusementParkId	Numeric(4.0)
AmusementParkName	Character(20)
AmusementParkWebsite	Character(60)
AmusementParkAddress	Address, GeneXus
AmusementParkPhoto	Image

#### Lembrar que:

- **Address**, é um **domínio semântico** que é automaticamente atribuído aos atributos definidos contendo em seus nomes o texto Address.
- Quando estiver definindo o tipo de dados do atributo identificador, em vez de utilizar diretamente Numeric(4.0), defina o **domínio Id** com esse tipo de dados.
- Configure a propriedade **Autonumber** desse domínio como **True**, para que todos os atributos baseados nele sejam numerados automaticamente, sem que o usuário tenha que se preocupar.

Vamos criar uma transação para registrar os países aos quais os parques de diversões pertencem.

**Lembre-se** que:

- pressionando ponto (".") quando está prestes a nomear um atributo na estrutura da transação, aparece iniciado com o nome da transação.
- Necessitará um atributo identificador, *CountryId*.

Quando estiver definindo o tipo de dados do atributo identificador, em vez de utilizar diretamente *Numeric(4,0)*, defina o **domínio Id** com esse tipo de dados.

Configure a propriedade *Autonumber* desse domínio como *True*, para que todos os atributos baseados nele sejam numerados automaticamente, sem que o usuário tenha que se preocupar.

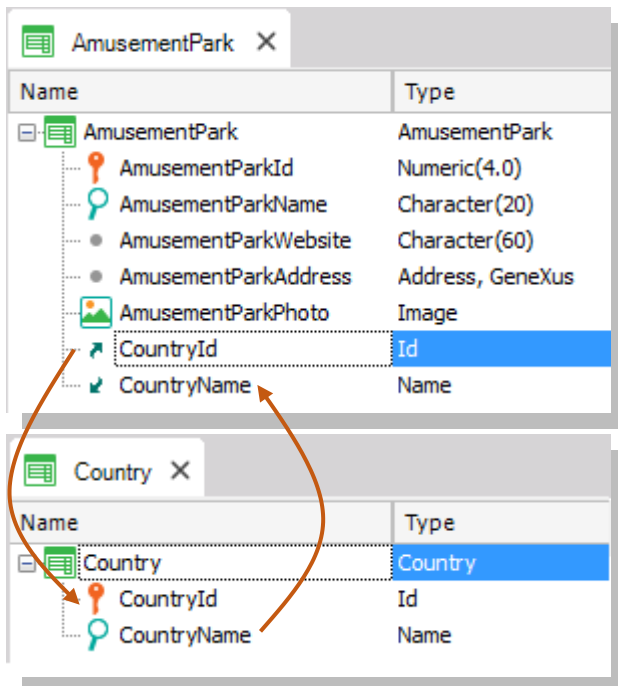
Defina o atributo *CountryName*, criando e utilizando um novo domínio: *Name = Character (50)*.

Name	Type
Country	Country
CountryId	Id
CountryName	Name

Agora, retornaremos à transação *AmusementPark*, para adicionar os atributos *CountryId* e *CountryName*.

Vamos aproveitar para alterar o tipo de dados de *AmusementId*, atribuindo a ele o domínio Id criado anteriormente, se você ainda não o tiver configurado.

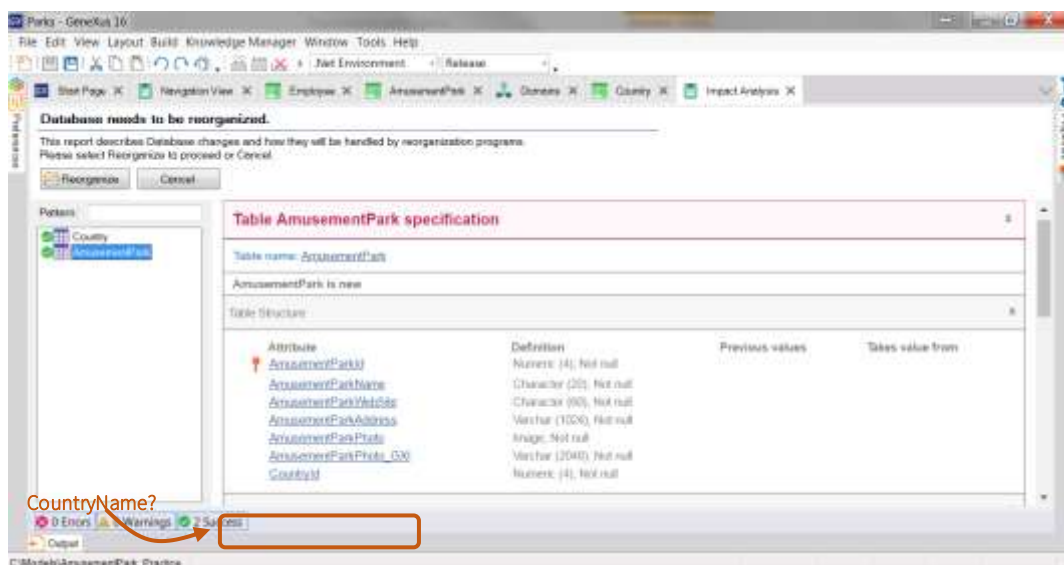




Por que colocou além de *CountryId*, o atributo *CountryName* em *AmusementPark*?

**Resposta:** E O atributo *CountryName* é importante que apareça na tela da transação, para nos mostrar o nome do país, que é o dado que mais lembramos do país, em vez de ver apenas seu identificador. Também é necessário adicionar o atributo na estrutura da transação, se quisermos utilizá-lo, por exemplo, dentro de uma regra.

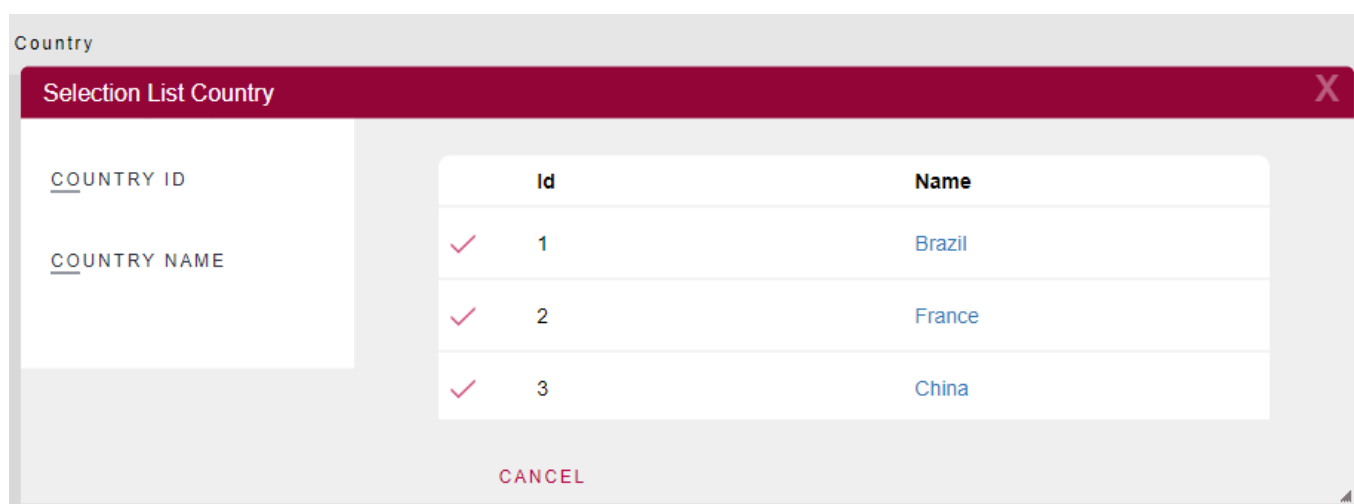
Execute para testar (F5) e o seguinte relatório será exibido:



Por que na tabela *AmusementPark*, que GeneXus informa que deve ser alterada na Base de Dados, o atributo *CountryName* não aparece? Ou seja, por que a tabela física não o conterá, quando está na estrutura da transação?

Depois de estudar o relatório, se concordarmos, pressionaremos *Reorganize* para realizar efetivamente o que é informado. O navegador será aberto com o menu com links para os 3 programas que correspondem a cada uma das transações (*AmusementPark*, *Country* e *Employee*).

Inserir como **países**: Brasil, França e China. Note que, deixando o valor 0 como valor no identificador, ao gravar lhe é atribuído automaticamente o número posterior ao último atribuído (efetivamente, se está autonumerando).



Inserir como parque de diversões: “Beto Carrero World”, que fica no Brasil. Se não se lembra do identificador do Brasil no sistema, como insere o país? É oferecido um ícone com uma seta ao lado de *CountryId*, para abrir uma “Lista de seleção” de países, criada automaticamente pelo GeneXus. Isso ocorre porque *CountryId* tem o papel de chave estrangeira (foreign key) nesta transação (ou seja, está “apontando” para outra tabela).

#### DADOS RELACIONADOS: COMO É MANTIDA A INTEGRIDADE?

*AmusementPark* e *Country* estão relacionados. Ao colocar *CountryId* na estrutura de *AmusementPark*, por ter exatamente o mesmo nome do atributo que é chave primária na transação *Country*, GeneXus entende que em *AmusementPark* o atributo *CountryId* é uma chave estrangeira e mantém automaticamente a integridade da informação. Então, por exemplo:

- Tente inserir um parque de diversões com um Id de país que não exista. Permite que grave esse parque?
- Escolha um parque previamente inserido (por exemplo, 'Beto Carrero World') e altere o país, para um que não exista. Foi possível gravar a modificação?
- Tente eliminar um país (usando a transação *Country*) que tenha algum parque associado (por exemplo, Brasil). Permite isso?

*Conclusão: os programas correspondentes às transações garantem a integridade dos dados.*

### TRANSAÇÃO 'GAME'

Como inicialmente solicitado, o sistema deve oferecer a possibilidade de inserir as atrações disponíveis em cada parque, portanto, devemos criar uma transação que contenha seu nome e o parque de diversões ao qual pertence.

Name	Type
Game	Game
GameId	Id
GameName	Name
AmusementParkId	Id
AmusementParkName	Character(40)

### TRANSAÇÃO 'CATEGORY'

Nos falta completar as informações da transação *Game*. Os funcionários descreveram que também registram de cada atração, a **categoria** (infantil, radical, recreativo, etc.) à qual pertence. Portanto, precisaremos criar uma transação para registrar esta informação e adicionar a categoria à transação *Game*.

Mas, além disso, nos informaram que não é obrigatório registrar sempre a categoria à qual pertence um determinada atração que está sendo manipulada. Pode ser deixada vazia. Se sabemos que o GeneXus controla automaticamente a integridade, como obtemos isso?

Solução:

Name	Type	Description	Formula	Nullable
Game	Game	Game		
GameId	Id	Game Id		No
GameName	Name	Game Name		No
AmusementParkId	Id	Amusement Park Id		No
AmusementParkName	Character(40)	Amusement Park Name		
CategoryId	Id	Category Id		Yes
CategoryName	Name	Category Name		

Name	Type
Category	Category
CategoryId	Id
CategoryName	Name

Para finalizar a definição da transação Game, adicionemos o dado que está faltando: a foto.

Para fazer isso, crie o atributo *GamePhoto* do tipo de dados *Image*.

Peça ao GeneXus para construir a aplicação, então pode testá-la em execução (F5).

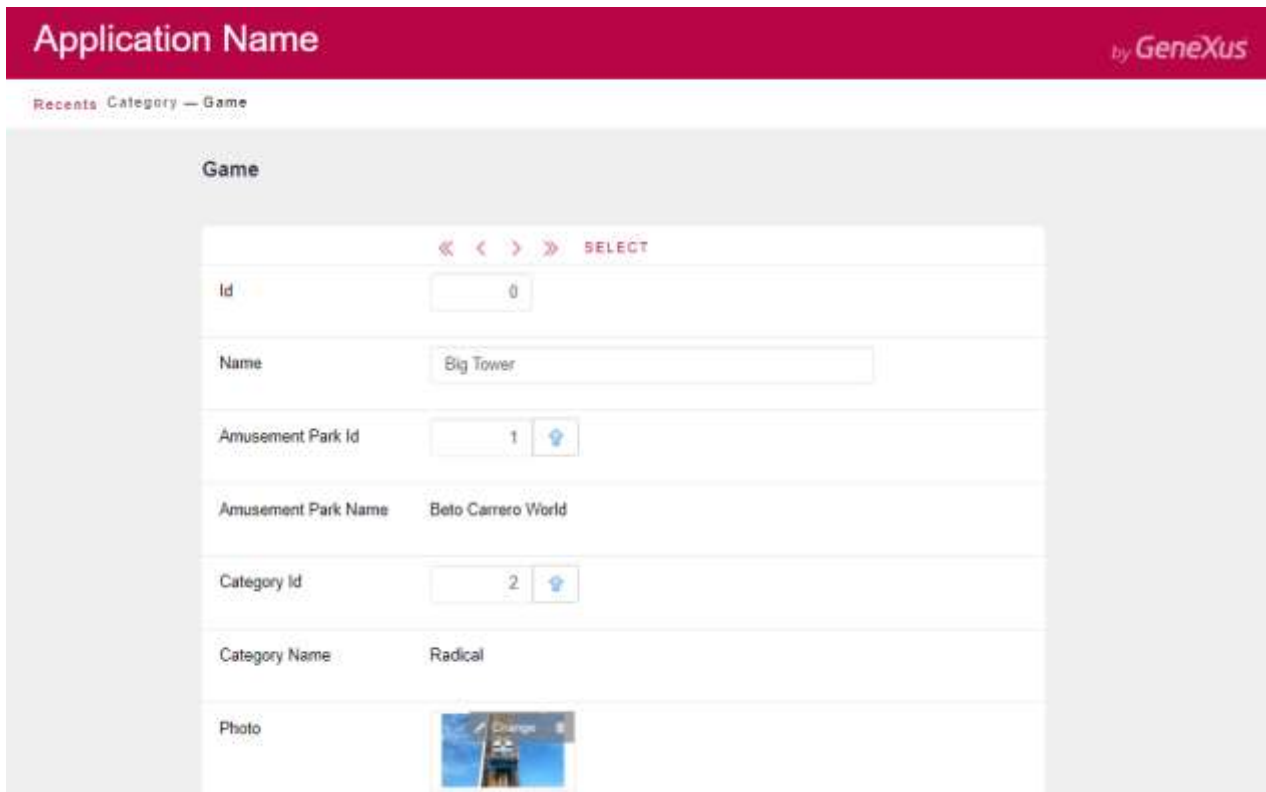
**Observe** o que informa o relatório de Análise de Impacto. Deverão ser criadas as tabelas *Category* e *Game* (não se preocupe em entender por que precisa armazenar dois valores por imagem).

Reorganize e execute.

Insira categorias (como infantil e radical) e atrações (como montanhas-russas e carrosséis).

Observe que, neste caso, pode deixar a categoria vazia (porque configurou a propriedade **Nullable** como **Yes** na estrutura da transação).

No entanto, se tentar colocar um valor inexistente como valor de *CategoryId* para a atração, não permitirá a gravação.



### TRANSAÇÕES “EMPLOYEE” E “AMUSEMENTPARK”, RELACIONADAS

Como nos disseram no início do desenvolvimento da aplicação, o sistema gerenciará os parques de diversões e seus funcionários. Portanto, precisamos vincular os funcionários que são registrados com o parque no qual trabalham.

Para isso, vamos para a transação *Employee* e adicionamos os atributos *AmusementParkId* e *AmusementParkName*.

Aproveitamos para alterar o tipo de dados de *EmployeeId*, atribuindo a ele o domínio *Id* criado anteriormente, caso você ainda não o tenha configurado.

Name	Type
Employee	Employee
EmployeeId	Id
EmployeeName	Character(20)
EmployeeLastName	Character(20)
EmployeeAddress	Address, GeneXus
EmployeePhone	Phone, GeneXus
EmployeeEmail	Email, GeneXus
AmusementParkId	Id
AmusementParkName	Character(40)

Fomos informados de que não é obrigatório registrar no momento, o parque onde o funcionário trabalha, ou seja, pode ficar vazio. O que devemos fazer?

Solução:

Name	Type	Description	Formula	Nullable
Employee	Employee	Employee		
EmployeeId	Id	Employee Id		No
EmployeeName	Character(20)	Employee Name		No
EmployeeLastName	Character(20)	Employee Last Name		No
EmployeeAddress	Address, GeneXus	Employee Address		No
EmployeePhone	Phone, GeneXus	Employee Phone		No
EmployeeEmail	Email, GeneXus	Employee Email		No
AmusementParkId	Id	Amusement Park Id		<b>Yes</b>
AmusementParkName	Character(40)	Amusement Park Name		

Execute para testar (F5) e aparecerá o seguinte relatório indicando que o atributo AmusementParkId agora permite deixar um valor não especificado:

**Database needs to be reorganized.**

This report describes Database changes and how they will be handled by reorganization programs. Please select Reorganize to proceed or Cancel.

Reorganize Cancel

Pattern: Employee

**Table Employee specification**

Table name: Employee

Employee needs conversion

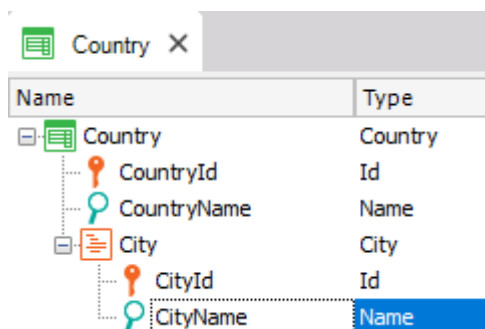
Table Structure

Attribute	Definition	Previous values	Takes value from
EmployeeId	Numeric (4), Not null, Autonumber		Employee EmployeeId
EmployeeName	Character (20), Not null		Employee EmployeeName
EmployeeLastName	Character (20), Not null		Employee EmployeeLastName
EmployeeAddress	Varchar (1024), Not null		Employee EmployeeAddress
EmployeePhone	Character (20), Not null		Employee EmployeePhone
EmployeeEmail	Varchar (100), Not null		Employee EmployeeEmail
New AmusementParkId	Numeric (4)		Null

0 Errors 0 Warnings 1 Success

### ADICIONEMOS AS CIDADES À TRANSAÇÃO 'COUNTRY'

Além dos países, precisamos registrar a informação de suas cidades. Portanto, devemos adicionar um segundo nível à transação *Country*, com o identificador e o nome da cidade.



**Lembre** que:

- Posicionado no atributo *CountryName*, com o botão direito, **Insert Level**, adiciona o subnível.
- Depois de dar um nome ao novo nível, digitando aspas (") em vez de ponto, o atributo que define se iniciará com o nome do nível.
- As cidades serão identificadas por seu próprio Id em combinação com o do país. Ou seja, não poderá identificar uma cidade sem primeiro fornecer a informação do país em questão. Assim, poderia haver uma cidade 1 Rosario tanto para Uruguai como para Argentina:  
País: 1 (Uruguai) – Cidade: 1 (Rosario)  
País: 2 (Argentina) – Cidade: 1 (Rosario)
- Ou inclusive poderia ser que Rosario para Argentina se identificava com outro número:  
País: 2 (Argentina) – Cidade: 4 (Rosario)

Reorganize e execute (F5).

**Database needs to be reorganized.**

This report describes Database changes and how they will be handled by reorganization programs.  
Please select Reorganize to proceed or Cancel.

Reorganize Cancel

Pattern: CountryCity

**Table CountryCity specification**

Table name: CountryCity

CountryCity is new

Warnings

rgz0009 AutoNumber=True ignored. Attribute CityId is not table CountryCity's primary key.

Attribute	Definition	Previous values	Takes value from
CountryId	Numeric (4), Not null		
CityId	Numeric (4), Not null		
CityName	Character (50), Not null		

Indexes

Name	Definition	Composition
ICOUNTRYCITY	primary key Clustered	CountryId CityId

**Observe** que o Mapa de Navegação irá informá-lo que:

- A propriedade **Autonumber** para o caso **CityId** será ignorada. Isto significa que em execução o usuário deverá inserir manualmente os identificadores de cidade. A explicação é que a propriedade *Autonumber* somente autonumera chaves primárias simples e, neste caso, *CityId* é o segundo componente de uma chave composta.
- Será criada uma nova tabela **CountryCity** para armazenar a informação correspondente às cidades.

Insira cidades para os países que já tinha registrados.

TRANSAÇÃO “AMUSEMENTPARK”: ADICIONEMOS A CIDADE.

Na transação *AmusementPark* adicionemos a cidade do país ao qual o parque pertence. O que deve fazer se a empresa nos informa que esse valor pode não ser conhecido ou relevante para um determinado parque em determinado momento?

Construa a aplicação e teste-a (F5 e Reorganize).

*Solução:*

Name	Type	Description	For...	Nullable
AmusementPark	AmusementPark	Amusement Park		
AmusementParkId	Id	Amusement Park Id		No
AmusementParkName	Character(40)	Amusement Park Name		No
AmusementParkWebSite	Character(60)	Amusement Park Web Site		No
AmusementParkAddress	Address, GeneXus	Amusement Park Address		No
AmusementParkPhoto	Image	Amusement Park Photo		No
CountryId	Id	Country Id		No
CountryName	Name	Country Name		No
CityId	Id	City Id		<b>Yes</b>
CityName	Name	City Name		No

Name	Type
Country	Country
CountryId	Id
CountryName	Name
City	City
CityId	Id
CityName	Name



## ADICIONEMOS COMPORTAMENTO ÀS TRANSAÇÕES (RULES)

Depois de testar conosco a aplicação que estamos desenvolvendo, na empresa nos dizem que, para os funcionários, há algum comportamento específico que devemos fazer cumprir ao manipular as informações por meio do programa (transação Employee).

Qual é este comportamento?

Nos dizem:

- “O sistema não deve permitir inserir funcionários sem nome ou sem sobrenome”.
- “O usuário deve ser avisado se está deixando o telefone sem atribuir, caso seja um descuido”.
- “Deve ser registrada a data de inserção do funcionário no sistema (**EmployeeAddedDate**) e deve ser proposta como valor padrão para esse atributo, a data de hoje”.

Especifique esse comportamento e teste-o (F5 e Reorganize).

**Lembre** que:

- As regras terminam com ponto e vírgula “;”.
- O método **IsEmpty()** aplicado a um atributo devolve True quando o atributo está vazio e False em caso contrário.
- A variável **&Today** é do sistema e tem carregado o valor da data do dia.

Para escrever uma variável dentro da tela **Rules**, quando digita “&” são exibidas todas as variáveis definidas até o momento, para que selecione aquela que precisa. A outra possibilidade é utilizar **Insert / Variable**.

Tente inserir um novo funcionário deixando o nome vazio. Permite gravar ou passar o próximo campo?

Idem com o sobrenome. Acontece o mesmo com o telefone?

Se então lhe informam que a data de inserção no sistema não deveria ser manipulada pelo usuário, mas apenas visualizada, como você estabelece este comportamento?

Especifique e teste-o em execução.

## PATTERNS: MELHORARANDO A INTERFACE PARA TRABALHAR COM INFORMAÇÃO

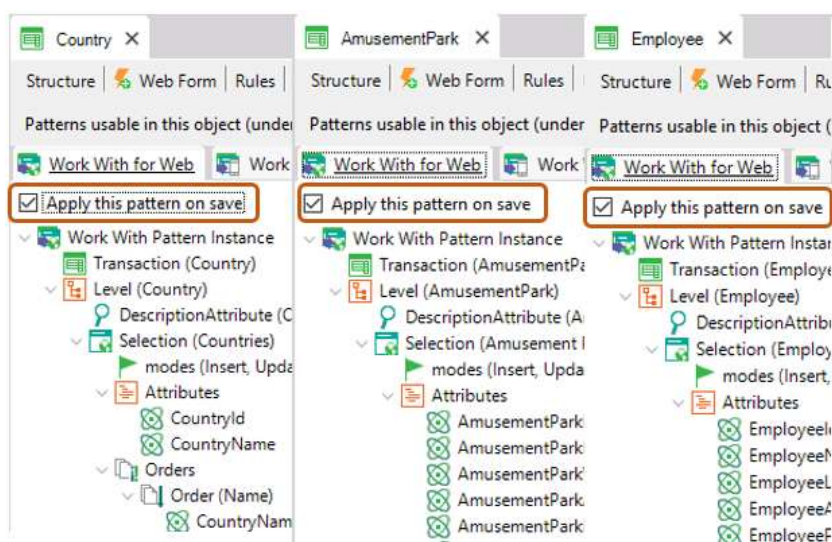
Ao mostrar ao cliente o que foi feito até agora, ele nos diz que gostaria de poder manipular as informações de países, parques de diversões, funcionários, categorias e atrações de uma forma mais potente e atraente (que ofereça consulta, possibilidade de filtragem, bem como inserir, modificar e excluir dados, etc.).

Para fazer isso, deverá aplicar o padrão *Work With for Web* nas três transações. Teste e veja em execução.

### Observar que:

- existe também um *Work With* para Smart Devices. Mas o que você deverá aplicar é o correspondente à aplicação web que está construindo.
- GeneXus criará automaticamente vários objetos por transação, para implementar o "Trabalhar com" essa entidade.

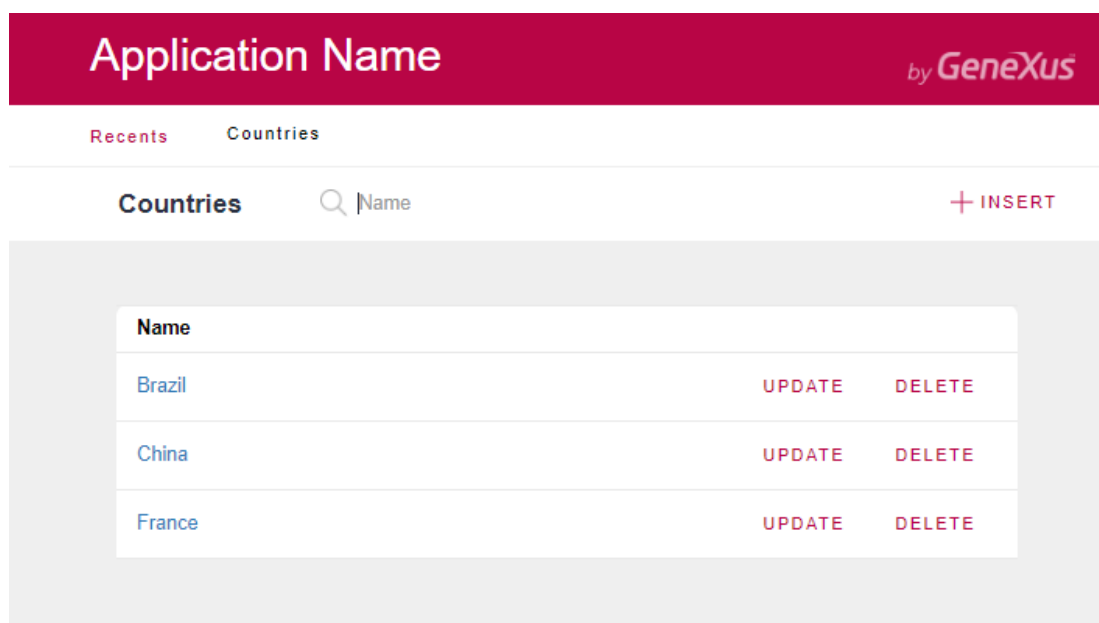
### Solução:



Por que não aparecem mais no *Developer Menu* as transações *Country*, *AmusementPark* e *Employee*?

Tente fazer o seguinte:

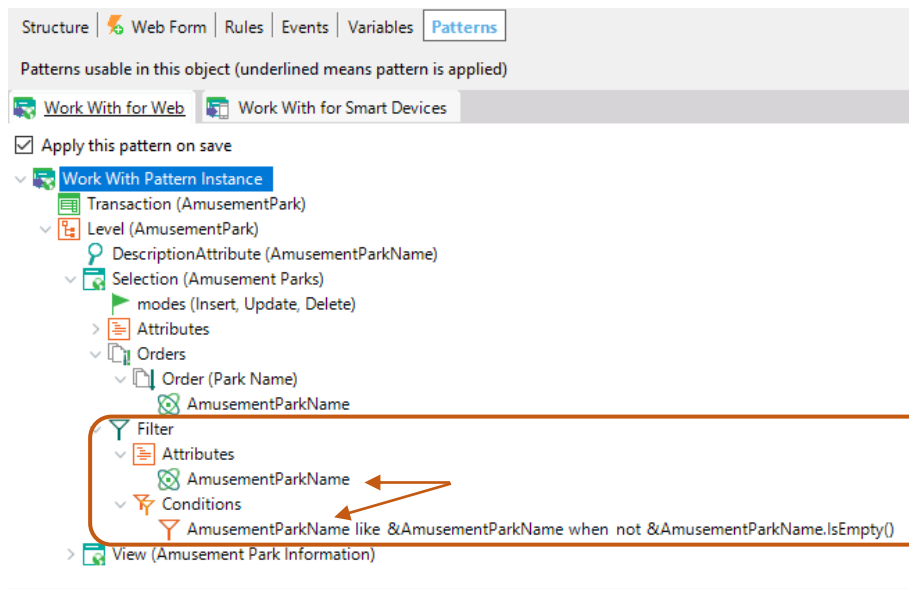
1. Inserir um novo país.
2. Modificar um país existente (por exemplo, lhe adicionando uma cidade).
3. Eliminar um país existente.
4. Visualizar a informação um país.
5. Realizar uma pesquisa por nome de país.



6. Insira alguns parques de diversões (Ex: Shangai Disney Resort, da China/Shangai, Parc Du Bocasse de Francia/Normandia, Happy Valley, da China/Beijing).
7. Filtre os parques de diversões cujo nome comece com P. E se agora quiser poder visualizar todos os parques da China? Esta possibilidade não está incluída, portanto, deveremos personalizar o pattern *Work With* desta transação, para incluí-la. Faça **em GeneXus** e teste em **execução**.

**Solução:**

Para isso, primeiro observe como está especificado o filtro que existe, por nome do parque de diversões:



8. Agora, remova os identificadores de país e cidade da tela do *Work With* e teste-o em execução.

Application Name by GeneXus

Recente Amusement Parks

Amusement Parks  + INSERT

Park Id	Park Name	Park Website	Park Address	Park Photo	Country Id	Country Name	City Id	City Name	UPDATE	DELETE
1	Beto Carrus World	www.betocarrus.com.br	Rua Inácio Francisco de Sousa, 1597		1	Brazil	2	São Paulo	UPDATE	DELETE
4	Happy Valley	bj.happyvalley.cn	Beijing Olympic Tower		3	China	1	Beijing	UPDATE	DELETE
3	Parc Du Bocasse	www.parcduboocasse.fr	225 Route de Clèves, 76690		2	France	2	Normandia	UPDATE	DELETE
2	Shanghai Disney Resort	www.shanghaidisneyresort.com	Shanghai Disney Resort, Pudong, Xangai		3	China	2	Shanghai	UPDATE	DELETE

9. Se agora deseja oferecer a possibilidade de que o usuário escolha se deseja ver os parques de diversões ordenados pelo nome do parque ou pelo nome do país, **implemente-o e teste.**

The screenshot shows a web application interface for 'Amusement Parks'. The interface includes a search bar for 'Park Name' and a dropdown menu for 'Ordered By: Park Name'. The dropdown menu has two options: 'Park Name' (selected) and 'Country Name'. The table below shows the following data:

Park Id	Park Name	Park Website	Park Address	Park Photo	Country Name	City Name	UPDATE	DELETE
1	Beta Casino World	www.betocasino.com.br	Rua Inácio Francisco de Souza, 1557		Brazil	São Paulo	UPDATE	DELETE
4	Happy Valley	bj.happyvalley.cn	Beijing Olympic Tower		China	Beijing	UPDATE	DELETE
3	Parc Du Bocasse	www.parcdubo-casse.fr	225 Route de Clères, 75650		France	Normandie	UPDATE	DELETE
2	Shanghai Disney Resort	www.shanghaidisneyresort.com	Shanghai Disney Resort, Pudong Xingui		China	Shanghai	UPDATE	DELETE

## TRANSAÇÕES “REPAIR” E “TECHNICIAN” E NECESSIDADE DE DEFINIR SUBTIPOS

Agora é necessário registrar as atrações que entram em estado de reparo. Cada reparo tem um identificador, uma data a partir da qual a atração deixa de estar disponível para uso, o número estimado de dias para seu conserto, o identificador da atração, seu nome, o técnico titular e o técnico substituto. Além disso, cada reparo tem um custo. Para o custo, crie um domínio chamado *Cost*, do tipo Numeric(8,2).

Crie uma transação para registrar os técnicos que trabalharão nos reparos. Cada técnico tem um identificador, um nome e sobrenome, um telefone, um país e uma cidade na qual se encontra.

Como é definido que cada reparo tem um técnico titular e outro substituto?

### **Lembre-se**

- 1) Que na estrutura da transação:
  - um ícone representando uma seta para cima ↗ informa que o atributo é chave estrangeira (Foreign Key), isto é, que aponta para outra tabela.
  - Um ícone representado por uma seta apontando para baixo ↘ informa que o atributo é inferido de outra tabela.
  - Um ícone representando um **S** indica que o atributo é um subtipo.
- 2) Sobre os grupos de subtipos:
  - São definidos da mesma maneira que qualquer tipo de objeto.
  - Cada grupo de subtipos deve conter obrigatoriamente um subtipo de um atributo primário (que é chave primária de uma tabela) ou conjunto de atributos que formam uma chave primária.
  - Em cada grupo de subtipos, é necessário incluir todos os atributos subtipos que precisam ser conhecidos, pertencentes à tabela base e/ou estendida da chave primária do grupo.

**Execute e verifique** que, ao tentar inserir um reparo, um erro será disparado se o técnico titular que está querendo atribuir ao reparo não existir. Idem para o técnico substituto.

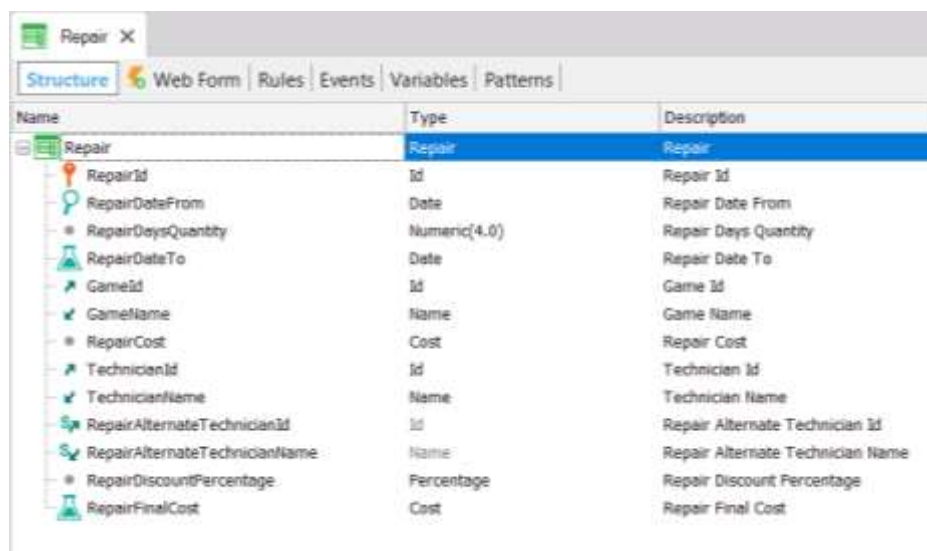
Não deve ser permitido inserir um reparo cujo técnico titular coincida com o técnico substituto. Implemente esse comportamento e teste-o em execução.

## FÓRMULAS

É necessário poder registrar o **desconto atual que tem cada reparo**. Defina um novo atributo na transação *Repair* para armazenar este dado. Dar ao novo atributo o nome: *RepairDiscountPercentage* e que seu tipo de dados seja um domínio *Percentage*, numérico de 3.

Deseja-se visualizar o preço final do conserto com o desconto aplicado. Para resolver isto, defina outro atributo, chamado *RepairFinalCost*, que seja uma **fórmula global** e que calcule automaticamente o preço final do reparo.

Adicione um novo campo chamado *RepairDateTo*, que será uma soma entre a data de início do reparo e a quantidade de dias que levará o mesmo.



Name	Type	Description
Repair	Repair	Repair
RepairId	Id	Repair Id
RepairDateFrom	Date	Repair Date From
RepairDaysQuantity	Numeric(4,0)	Repair Days Quantity
RepairDateTo	Date	Repair Date To
GameId	Id	Game Id
GameName	Name	Game Name
RepairCost	Cost	Repair Cost
TechnicianId	Id	Technician Id
TechnicianName	Name	Technician Name
RepairAlternateTechnicianId	Id	Repair Alternate Technician Id
RepairAlternateTechnicianName	Name	Repair Alternate Technician Name
RepairDiscountPercentage	Percentage	Repair Discount Percentage
RepairFinalCost	Cost	Repair Final Cost

Pressione F5, observe na **Análise de Impacto** qual atributo será criado fisicamente e qual não será, reorganize e teste a aplicação em operação.

## CRIAÇÃO DE SEGUNDO NÍVEL

Deseja-se criar um segundo nível na transação *Repair* para guardar um detalhe do tipo de problema encontrado para reparar.

Para fazer isso, primeiro deverá ser criado um domínio chamado *KindName*, *Character(1)*. Restrinja os valores possíveis para o domínio: que sejam válidos os valores “E”, “M” e “R” (editando a propriedade **Enum Values** do mesmo, como mostrado a seguir).



Crie um segundo nível na transação *Repair* chamada *Kind* para registrar o tipo de reparo. Este nível terá estes três atributos:

- *RepairKindId* – *Numeric(4)* (será chave neste segundo nível)
- *RepairKindName* – Baseado no domínio *KindName* (Genexus o sugerirá automaticamente).
- *RepairKindRemarks* – *Character(120)*, Conterá observações sobre o problema, um pequeno detalhe do problema encontrado ou a peça a ser substituída, por exemplo.

**Lembre-se que** para definir que determinado atributo faz parte da chave primária, deve pressionar o botão **direito do mouse** no atributo e o menu contextual oferecerá a opção **Toggle Key**. Neste caso, não será necessário, pois somente o primeiro atributo faz parte da chave primária e este já aparece com o ícone da chave.

Para saber a quantidade de tipos de problemas envolvidos em um reparo, crie um novo atributo no primeiro nível da transação *Repair*, chamado *RepairProblems*, *Numeric(1)* e defina-o como uma fórmula global, deverá contar os tipos de problemas encontrados.

Um reparo poderia envolver problemas de eletricidade, mecânicos ou a substituição de alguma peça. Poderia até ter mais de um problema do mesmo tipo, dois registros de eletricidade, por exemplo. Se houver muitos, pode detalhar algo mais com o uso do atributo de observações, que permite escrever um texto pequeno.

Se deseja ver a quantidade de problemas no formulário web e deve ser controlado que sejam inseridas entre 1 e 3 linhas de tipos de problemas.

Este controle deverá ser feito **quando terminar de inserir dados no segundo nível** e depois de ter pressionado o botão *Confirm*.

Name	Type	Description
Repair	Repair	Repair
RepairId	Id	Repair Id
RepairDateFrom	Date	Repair Date From
RepairDaysQuantity	Numeric(4.0)	Repair Days Quantity
RepairDateTo	Date	Repair Date To
GameId	Id	Game Id
GameName	Name	Game Name
RepairCost	Cost	Repair Cost
TechnicianId	Id	Technician Id
TechnicianName	Name	Technician Name
RepairAlternateTechnicianId	Id	Repair Alternate Technician Id
RepairAlternateTechnicianName	Name	Repair Alternate Technician Name
RepairDiscountPercentage	Percentage	Repair Discount Percentage
RepairFinalCost	Cost	Repair Final Cost
RepairProblems	Numeric(1.0)	Repair Problems
Kind	Kind	Kind
RepairKindId	Numeric(1.0)	Repair Kind Id
RepairKindName	KindName	Repair Kind Name
RepairKindRemarks	Character(120)	Repair Kind Remarks

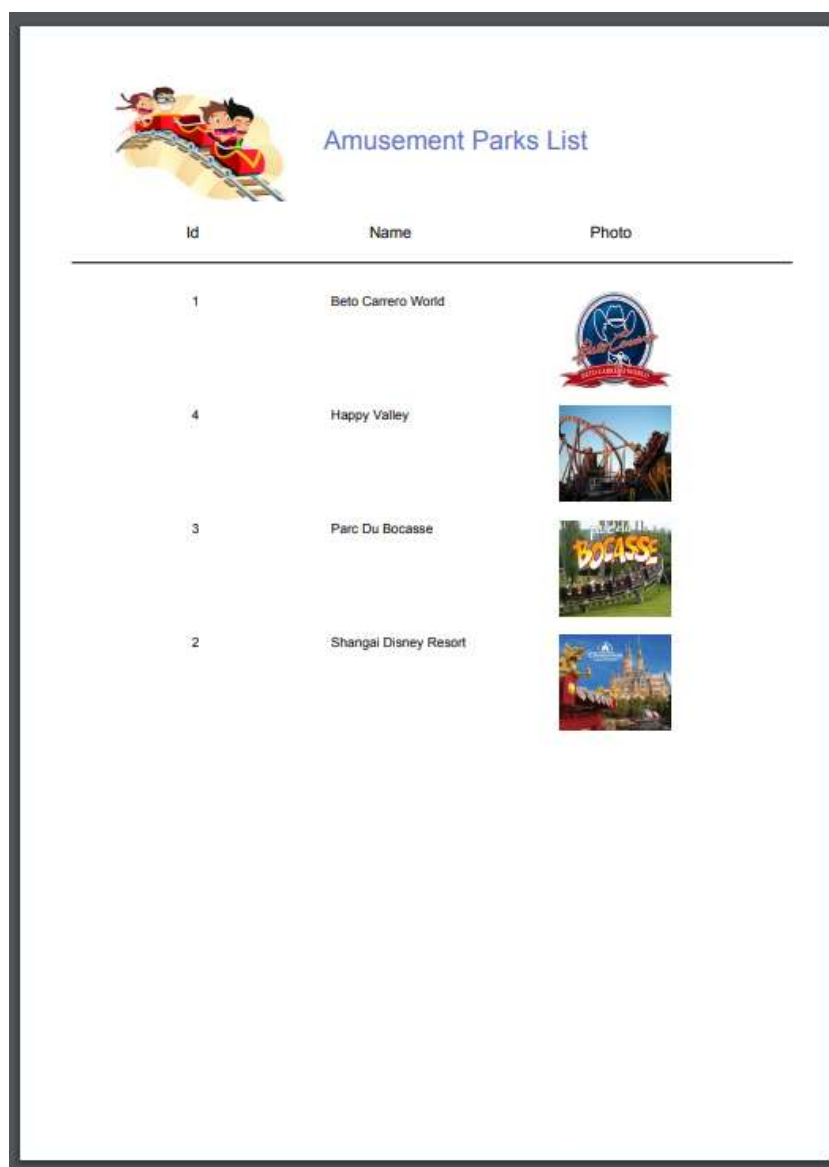
O valor de *RepairKindId* vai sendo inserido manualmente, o normal seria ir dando valores a partir de 1. Lembremos de que **não é possível autonumerar** utilizando a propriedade *Autonumber*.







## LISTAS PDF

Agora, suponhamos que, como parte da aplicação, deverá ser implementada a possibilidade de que, a pedido do usuário, sejam exibidas listagens *PDF* com a informação solicitada. Por exemplo, suponha que precise de uma lista que mostre em ordem alfabética os parques de diversões armazenados na base de dados.

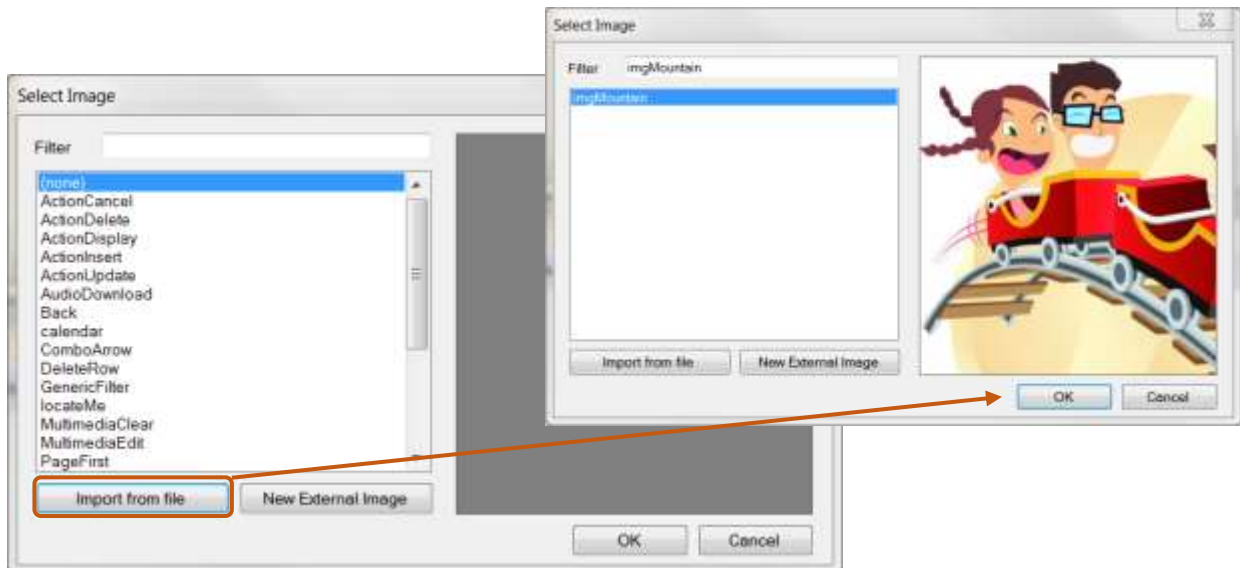
Sabe-se que deve parecer mais ou menos como segue:



Id	Name	Photo
1	Beto Carrero World	
4	Happy Valley	
3	Parc Du Bocasse	
2	Shangai Disney Resort	

**Sugestão para colocar uma imagem no título:** Utilizar o controle *Image* presente na *Toolbox* para exibir uma imagem junto ao título da lista.

Tal imagem deverá ser integrada à base de conhecimento. Para isso selecionar a opção *Import from File*, procurar a imagem e atribuir a ela um nome.



Implemente em GeneXus.

**Lembrar** que para poder visualizar uma lista diretamente a partir do browser, a mesma deve ser gerada como PDF. Para isto, deve configurar as seguintes propriedades do objeto procedimento:

- Main program = 'True'
- Call Protocol = HTTP
- Report output = Only to File

E a seguinte regra:


- Output\_file('nome-arquivo.pdf' , 'PDF')

Para executar a lista, na guia do objeto, botão direito / **Run with this only**


Notou no que lhe informa o mapa de navegação do procedimento?

E se agora necessita que a lista saia ordenada pelo nome do país? Implemente-o, observe o que é informado no mapa de navegação e teste-o.

E se agora precisa somente listar os parques de diversões da China? Teste-o (observando o mapa de navegação).

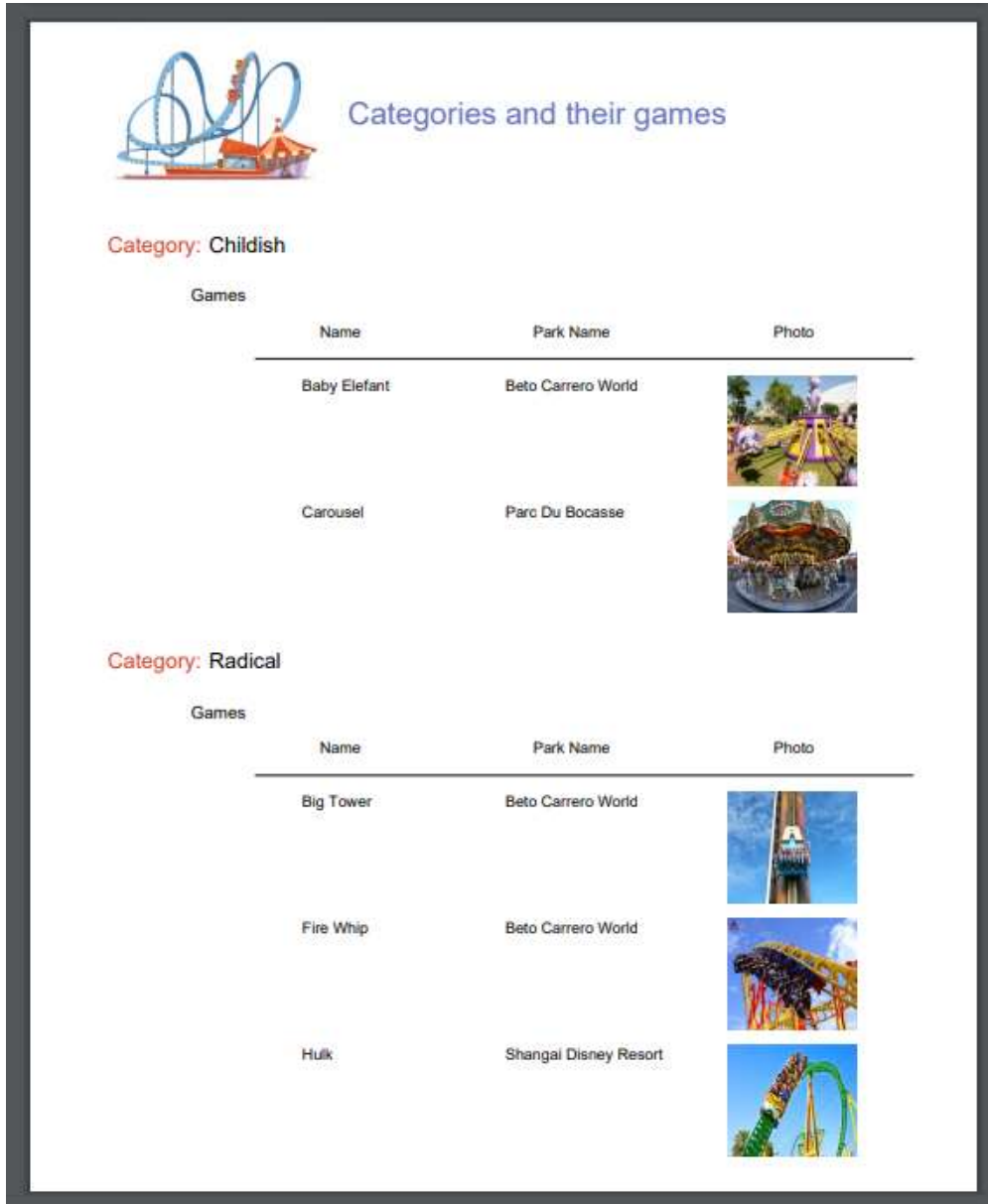


### Amusement Parks List

Id	Name	Country	Photo
2	Shangai Disney Resort	China	
4	Happy Valley	China	


Em cada caso, descubra qual tabela da base de dados se está percorrendo para realizar a consulta do *for each*.

Também é necessária uma lista como a que segue (que mostre cada categoria, e para cada uma delas, suas atrações). Implemente-a e teste o que foi feito.






The screenshot shows a web application interface with the title "Categories and their games" and a roller coaster icon. It displays two tables of amusement rides, each with a category header, a table of ride details, and a photo of the ride.

**Category: Childish**

Games	Name	Park Name	Photo
	Baby Elefant	Beto Carrero World	
	Carousel	Paro Du Bocasse	

**Category: Radical**

Games	Name	Park Name	Photo
	Big Tower	Beto Carrero World	
	Fire Whip	Beto Carrero World	
	Hulk	Shangai Disney Resort	

Adicione uma nova categoria ao sistema, por exemplo, *Aquatic* (aquáticos). Execute novamente a lista anterior. A categoria saiu listada?

Modifique a lista anterior para que não sejam listadas categorias que não possuem atrações relacionadas.

Quais modificações você encontrou no mapa de navegação?

**Resposta:** O surgimento da palavra *break* que indica que está realizando um **corte de controle**, mostrando também que a tabela base do *for each* externo é igual àquela do *for each* aninhado.

Outro pedido da empresa é uma lista que mostre todos os nomes de países e para cada país, a quantidade de parques de diversões que oferece:



Country	Quantity
Brazil	2
France	1
China	2
United States	3

E nos pedem outra lista também que mostre todos os países que têm mais de 2 parques para visitar:



Country	Quantity
United States	3

## PASSAGEM DE PARÂMETROS

Muitas vezes necessitamos que um objeto receba parâmetros para que, com base neles, execute sua lógica. Por exemplo, uma coisa é fazer uma lista *PDF* de todos os parques de diversões da base de dados, e outra é fazer uma lista daqueles cujos nomes estão dentro de um determinado intervalo.

### LISTA DE PARQUES EM UM INTERVALO DETERMINADO

Grave com outro nome o procedimento que criou anteriormente para listar os parques de diversões e modifique-o para que agora liste apenas aqueles cujo nome esteja dentro de um intervalo recebido por parâmetro (o valor inicial e o final do intervalo serão os parâmetros recebidos).

Implemente uma tela que solicite os valores desse intervalo ao usuário e chame este objeto, passando-lhe esses valores por parâmetro. Teste em execução.

#### **Lembre** que:

- Se define uma variável baseada em (**based on**) um atributo, ficará vinculada ao tipo de dados do atributo, ou seja, se este for modificado, o da variável também, de acordo com a alteração.
- As variáveis usadas para fazer uma chamada no objeto que chama e aquelas usadas para declarar os parâmetros que são recebidos no objeto chamado não precisam coincidir em nome, mas devem ter tipos de dados compatíveis.

#### *Solução:*

Para o procedimento (vamos chamá-lo de *AmusementParksFromTo*, adicionamos a regra *parm* (e definimos ambas as variáveis no procedimento):

```
parm(in: &fromName, in: &toName);
```

E em seu Source:

```
print Title
for each AmusementPark
  where AmusementParkName >= &fromName
  where AmusementParkName <= &toName
  print Data
endfor
```

E então criamos um web panel, no qual definimos duas variáveis, que podem ser chamadas de qualquer forma, por exemplo &A e &B, mas que devem ter como tipo de

dados um compatível com o das variáveis &fromName e &toName. Por quê? Porque será nestas variáveis que o usuário irá inserir valores que no evento que programemos serão enviadas por parâmetro para o procedimento, assim:

```
AmusementParksFromTo(&A, &B)
```

## BUSINESS COMPONENTS

Realizaremos algumas operações na base de dados, através de *Business Components*.

### AUMENTO DE PREÇO DOS REPAROS

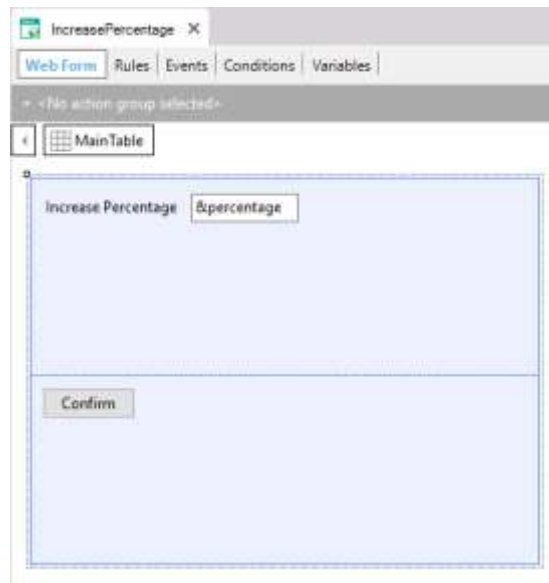
A empresa se reserva o direito de aumentar os preços dos reparos em um determinado percentual a partir de determinado momento. Para fazer uma gravação massiva, testaremos esta funcionalidade. Para fazer isso, deveremos implementar uma **tela** que permita ao usuário especificar esse **percentual de aumento** e dar a **ordem** para aplicá-lo a todos os reparos da base de dados. **Implemente e teste.**

#### **Lembre** que:

- O objeto web panel permite implementar telas flexíveis para entrada e saída de informações.
- Para a inserção de informações (que o usuário possa inserir valores na tela), pode inserir a partir da *Toolbar* um controle *Attribute/Variable* no form e atribuir a ele uma variável
- Se quiser editar barras de menu, posicionando-se em GeneXus acima, na barra, com botão direito poderá inserir, por exemplo, a barra *Formatting*.
- Para que o web panel execute uma ação determinada, podem ser inseridos botões e programar o "evento" associado.
- Os *Business Components* são tipos de dados que são criados ao configurar a propriedade de nome *Business Component* do objeto transação com valor *Yes*. Ao fazer isso, para inserir, modificar ou excluir registros das tabelas correspondentes, poderá ser utilizada além da transação, uma variável do tipo de dados *Business Component* em qualquer outro objeto (por exemplo, um web panel) e realizar as mesmas operações por meio de dos métodos *Load()*, *Save()* e *Delete()*.
- Para que as operações realizadas através do *Business Component* sejam efetuadas permanentemente, deverá executar em seguida o comando *Commit*.
- Se para um valor X quiser aumentá-lo em 20%, consegue-se fazendo  $X = X * (1 + 20/100) = X * 1,20$

#### **Solução:**

Uma solução possível (tente implementar você o que foi solicitado sem olhar o que segue): criamos um Web panel, com uma variável &percentage, do tipo de dados: Numeric(3.0)



Clicando duas vezes no botão, nos leva à seção Events, editando o evento associado ao momento de definir o botão. Em nosso exemplo, é o evento Confirm.

Lá programaremos a lógica que queremos que seja executada quando o usuário pressione o botão.

Necessitamos percorrer todos os reparos e sobrescrever o preço que tinham, aumentando-o nesse percentual indicado pelo usuário na variável de tela.

Para percorrer todos os reparos da base de dados, usamos o comando For each.

E para cada reparo, como o editamos para modificar o valor de RepairCost?

Precisaremos de uma variável para isso, que tenha toda a estrutura da transação Repair, e também nos permita gravar na base de dados. Que tipo de dados terá então essa variável? O Business Component Repair (observe que o tipo de dados Business Component tem o mesmo nome que a transação). Mas GeneXus não cria esse tipo de dados automaticamente para cada transação. Temos que solicitar isso. Como o fazíamos? Definindo a propriedade correspondente da transação.

Feito isto, então:

```

Event 'Confirm'
  For each Repair
    &Repair.Load(RepairId)
    &Repair.RepairCost = &Repair.RepairCost * (1 + &percentage/100)
    &Repair.Save()
  Endfor
  commit
Endevent

```



Como as operações de gravação ou exclusão da base de dados (por meio dos métodos Save() e Delete()) podem causar erros, é importante saber o que aconteceu. Para isso temos o método Success() que retornará True se não houver erros, e False caso contrário.

```

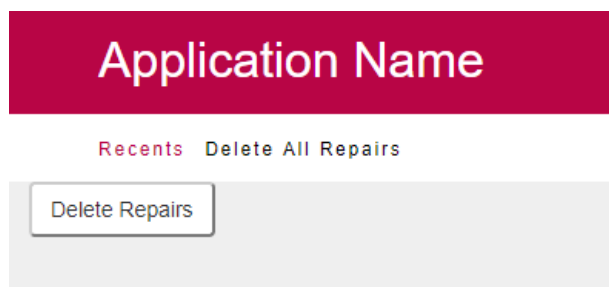
Event 'Confirm'
  For each Repair
    &Repair.Load(RepairId)
    &Repair.RepairCost = &Repair.RepairCost * (1 + &percentage/100)
    &Repair.Save()
    If &Repair.Success()
      commit
    else
      rollback
    endif
  Endfor
Endevent

```

Ao inserir, modificar ou excluir registros de uma tabela da base de dados, enquanto não seja dito: “tudo que foi feito, que fique permanente”, essas modificações serão provisórias. O que quer dizer? Que por exemplo, se houver um blecaute, essas modificações serão perdidas. A maneira de dizer “o que foi feito, que fique permanente” é executar o comando Commit. Se, por outro lado, queremos que o que fizemos seja desfeito, executamos o comando Rollback.

## TELA PARA ELIMINAÇÃO DE TODOS OS REPAROS

Salve o web panel anterior com outro nome (para isto basta posicionar-se na aba e com o botão direito, fazer *Save as*) e modifique o Form para que contenha apenas um botão com o texto *Delete Repairs*, de forma que em execução este web panel seja visto assim:



Quando o usuário pressione o botão, deverão ser eliminados todos os reparos da base de dados.

O que deve modificar do evento *Confirm* que tinha programado?

**Nota:** Se posiciona-se sobre o botão e vê suas propriedades, na de nome *Caption* pode modificar o texto do botão.

Solução:

```
Event 'Confirm'  
  For each Repair  
    &Repair.Load(RepairId)  
    &Repair.Delete()  
    If &Repair.Success()  
      commit  
      msg("Successful deletion")  
    else  
      msg("Deletion failed")  
      rollback  
    endif  
  Endfor  
Endevent
```

Com *Msg*, obterá que seja exibida essa mensagem na tela do web panel.

## PROCEDIMENTOS PARA ATUALIZAR REGISTROS

### AUMENTO DE PREÇOS DOS REPAROS

Suponha que os reparos para os quais deve aumentar o preço em uma determinada porcentagem sejam milhares. Sabendo que o aumento de preço é um procedimento simples que não fará falhar a integridade de forma alguma, exercite resolver com um procedimento, sem utilizar *Business Components*.

**Lembre** que:

- Com o comando *for each* dentro de um procedimento, pode atualizar os atributos de sua tabela estendida através de atribuições simples.
- A atualização "direta" através de procedimentos não controla a integridade das informações.
- Todo objeto deve declarar os parâmetros que recebe e os parâmetros que retorna. Se não os declarar, não receberá nem retornará valores.
- Os parâmetros são declarados através da regra **parm**.
- As variáveis são locais para o objeto onde são utilizadas. Isto significa que se eu quiser receber um valor como parâmetro em uma variável &X, devo declará-la no objeto.

## ELIMINAÇÃO DE TODOS OS REPAROS

E se agora quiser eliminar todos os reparos, como foi feito anteriormente no prático, mas desta vez através de um procedimento?

---

### Solução:

Crie um procedimento *RepairsDeletion* que não receba parâmetros, com o seguinte código:

```
for each Repair
  delete
endfor
```

Faça um *Save as* do web panel que foi implementado para fazer a eliminação através do *Business Component*) e altere o evento *Enter*:

```
Event Enter
  RepairsDeletion()
EndEvent
```

---

E se agora deseja excluir todas as informações da base de dados?

### Solução

Criar um procedimento DeleteAll com o seguinte Source:

```

1  for each Repair
2      for each Repair.Kind
3          delete
4      endfor
5      delete
6  endfor
7
8  for each Technician
9      delete
10 endfor
11
12 for each Game
13     delete
14 endfor
15
16 for each Category
17     delete
18 endfor
19
20 for each Employee
21     delete
22 endfor
23
24 for each AmusementPark
25     delete
26 endfor
27
28 for each Country
29     for each Country.City
30         delete
31     endfor
32     delete
33 endfor

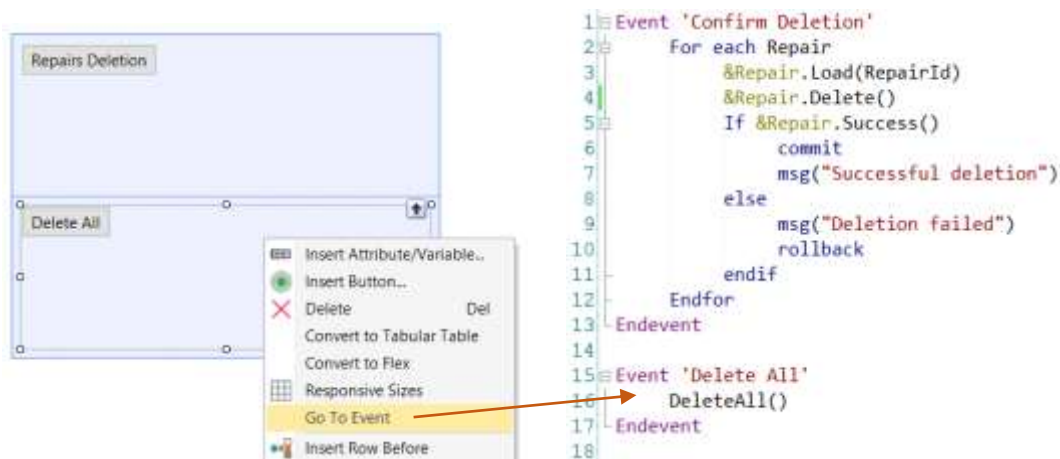
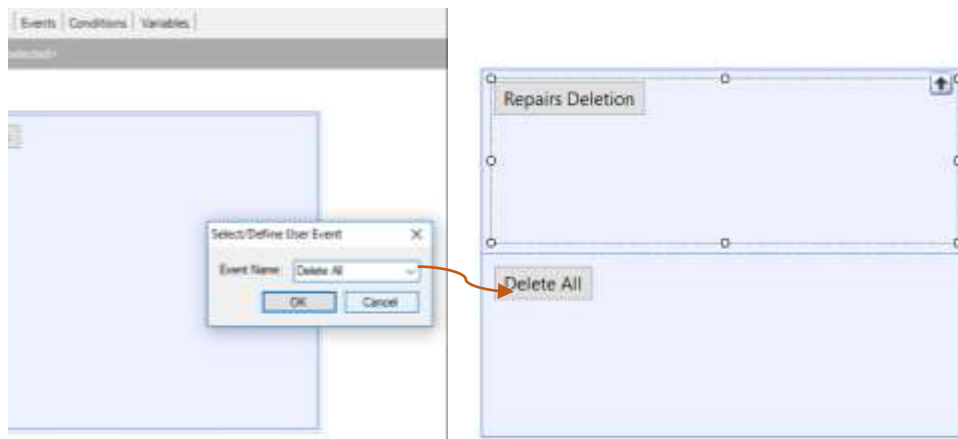
```

Lembre-se de que os procedimentos não validam a consistência dos dados, mas a base de dados o faz. Ou seja, a base de dados valida a consistência dos dados inter-relacionados, portanto, a ordem na qual você tenta eliminar os dados é importante.

Por exemplo, se você tentar eliminar os países antes dos parques, a base de dados o impedirá, o programa será cancelado e não será amigável para o usuário.

No mesmo web panel que foi utilizado para excluir todos os registros de Repair utilizando Business Components, adicione um novo botão, ao qual associaremos um “evento de usuário” e faremos a chamada ao procedimento.

Arraste o controle button sobre o form e defina o novo evento.



Então, pressionando o botão direito do mouse sobre o botão e selecionando Go To Event, nos posicionamos no evento associado ao botão e definimos dentro dele a chamada ao procedimento.

## INICIALIZAÇÃO DA INFORMAÇÃO DA BASE DDE DADOS [OPCIONAL]

Quando a aplicação que você está desenvolvendo é colocada em produção (ou seja, comece a ser utilizada pela empresa) deverão ser carregados todos os dados de países, parques, categorias, atrações, técnicos, reparos. Utilize as facilidades que o objeto *Transação* oferece para isto e inicialize essas tabelas com informações fornecidas pela empresa, e teste.

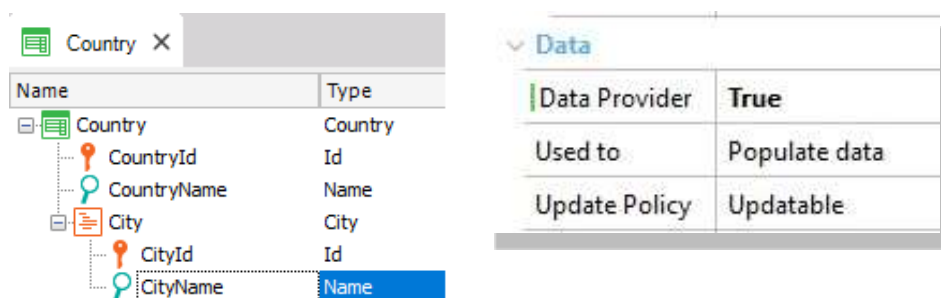
### **Levar em conta que:**

- As transações têm uma propriedade **Data Provider** que nos permite ter um data provider associado à transação, de modo que quando a tabela é criada na base de dados, seja inicializada com os valores que definamos no referido data provider. Para isso, também deverá configurar a propriedade **Used to** e **Update Policy**.
- Deve inserir as imagens na KB para poder utilizá-las. Uma maneira de fazer isso é na janela *KB Explorer* ir para a pasta *Customization* e escolhendo *Images*, onde são listadas todas as imagens atualmente armazenadas na *KB*, inserir uma nova (a partir de um arquivo) e dar um nome a ela.

## Solução

Como forma de testar diferentes opções, popularemos os países e as categorias utilizando a propriedade Data Provider das transações.

- Abra a transação Country e configure o valor True para a propriedade Data Provider:



- Carregue os dados desejados em Country\_DataProvider (pode vê-lo abaixo da transação Country na janela KBExplorer). No momento da criação, a tabela será carregada com os dados indicados.

```

Source* Rules Variables Help Documentation
1 CountryCollection
2 {
3   Country
4   {
5     CountryName = "Brazil"
6     City
7     {
8       City
9       {
10        CityId = 1
11        CityName = "Rio de Janeiro"
12      }
13    }
14  }
15  Country
16  {
17    CountryName = "France"
18    City
19    {
20      City
21      {
22        CityId = 1
23        CityName = "Paris"
24      }
25      City
26      {
27        CityId = 2
28        CityName = "Versailles"
29      }
30    }
31  }
}

```

**Lembrar:**

O DP associado à transação voltará a disparar

- Cada vez que a tabela correspondente seja reorganizada.
- Cada vez que seja editado o conteúdo do DP.

Portanto, é necessário controlar que as informações não sejam duplicadas. Isso pode ser conseguido:

- Se a chave primária for autonumerada, pode ser definido um índice *unique* no atributo descritor.
- Deixando a chave primária sem autonumeração. Desta forma, seu valor será atribuído de forma fixa ao definir o DP e não se duplicará.

- De maneira análoga, carregue as categorias.

**WEB PANELS**

Solicita-se uma página que mostre todos os países e para cada um, a quantidade de parques de diversões que oferece visitar.

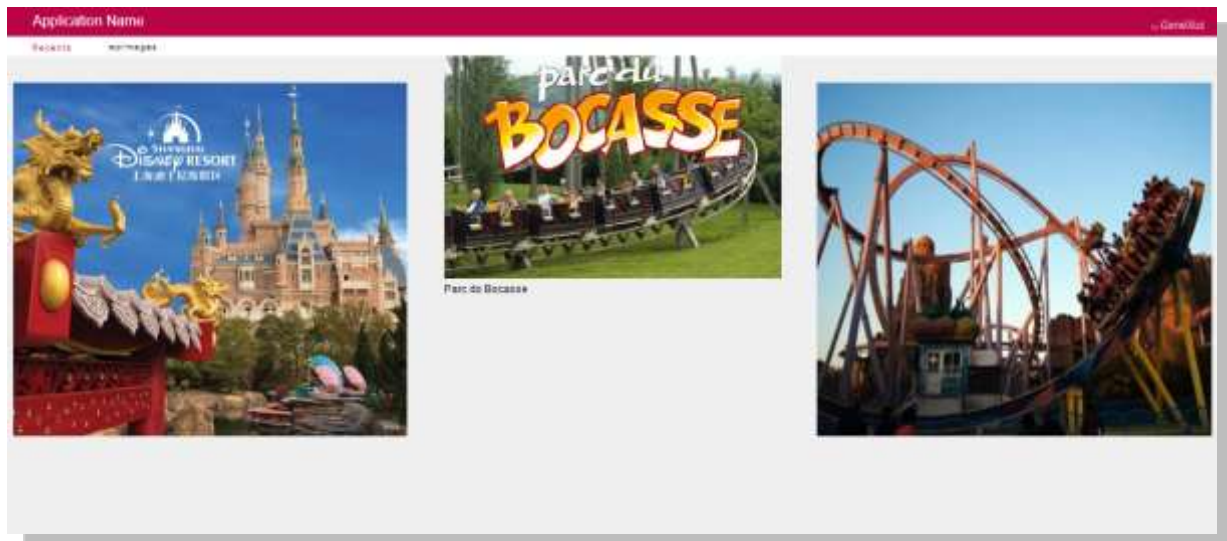
**Lembre** que o **evento Load em um web panel com tabela base que possui um grid** é executado antes de carregar cada linha no grid. Esse evento é adequado para atribuir à variável o cálculo que retorna a conta das cidades de cada país que está sendo navegado e prestes a carregar em uma linha de grid.

Agora, adicione duas variáveis ao web panel definido (&CountryNameFrom e &CountryNameTo) e defina as condições necessárias para filtrar os países incluídos no referido intervalo.

## EXTENDED CONTROLS

Utilizando o *extended control ImageGallery*, crie um web panel que mostre a galeria de fotos de todos os parques de diversões.

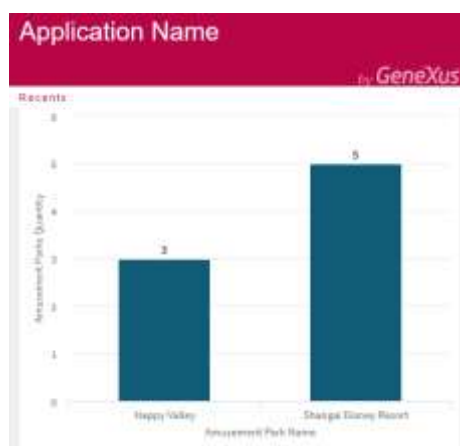
Personalize as propriedades do *extended control*, estabelecendo Width=1000, Height= 500, e Type=Slider:



## OBJETO QUERY

Defina um objeto *Query* que retorne apenas os parques de diversão da China, organizados em ordem alfabética, cada um deles com sua respectiva quantidade de atrações.

Defina um web panel e, utilizando o *extended control QueryViewer*, visualize a consulta anterior como um gráfico.



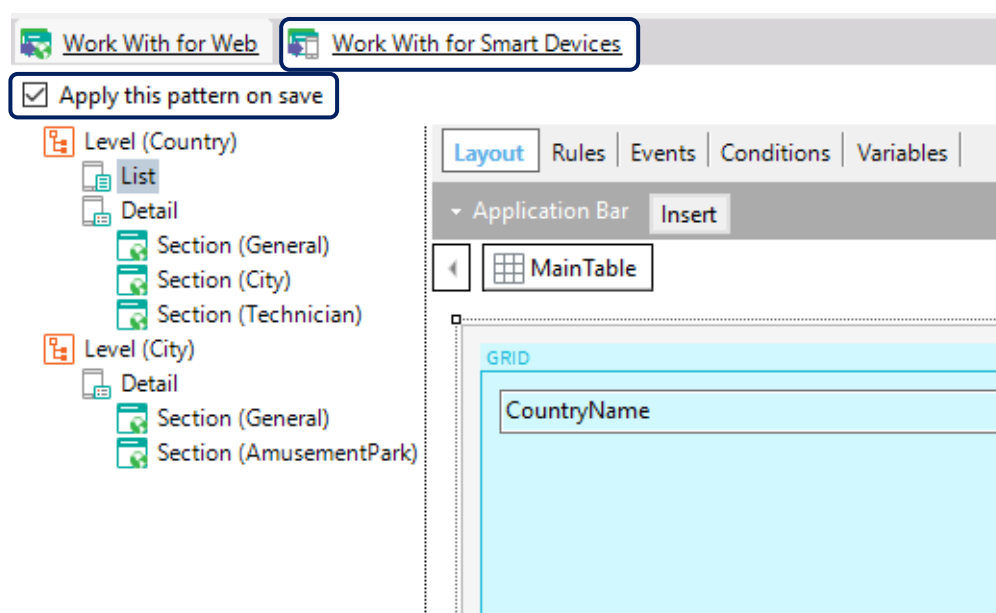


## PARTE PARA SMART DEVICES

A empresa também quer oferecer uma pequena aplicação para dispositivos inteligentes para ser utilizada pelos usuários finais.

O objetivo é que qualquer pessoa possa consultar a partir de seu smart device, todos os países que pode visitar, e para cada um deles seus parques de diversões e atrações.

Para fazer isso, deve aplicar o padrão *Work With for Smart Devices* à transação *Country*:



E simplesmente gravar, para depois criar um objeto *MenuforSmartDevices* e adicionar o objeto chamado: *WorkWithDevicesCountry* gerado pelo padrão.

**Lembre** que pelo fato de ter criado dentro de nossa base de conhecimento, objetos próprios para Smart Devices, ao pressionar F5 automaticamente se executará o emulador para *Android*, podendo também acessar nossa aplicação web a partir do *Developer menu*.

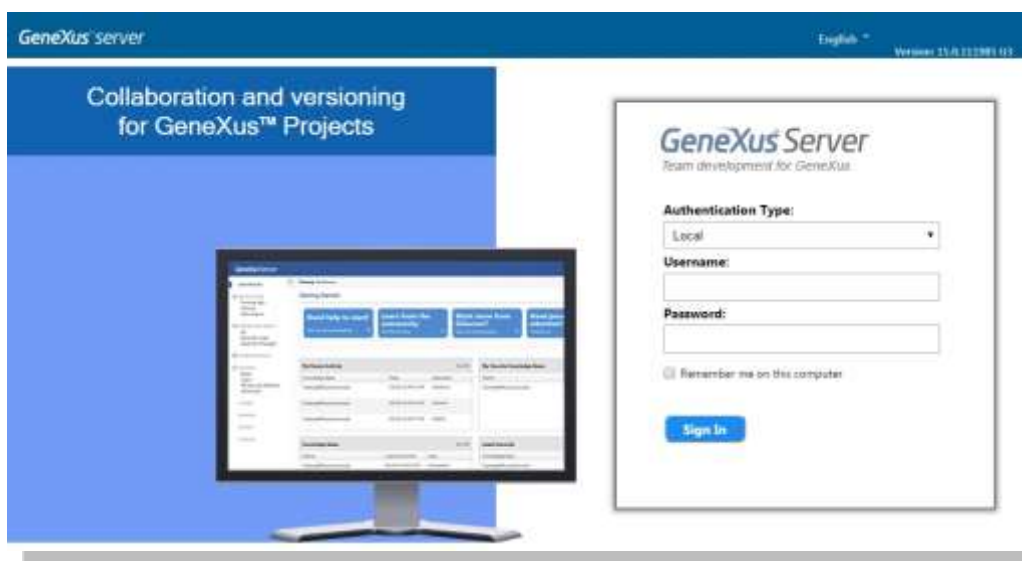
## GENEXUS SERVER

Publique a base de conhecimento no servidor <http://sandbox.genexusserver.com/v16>

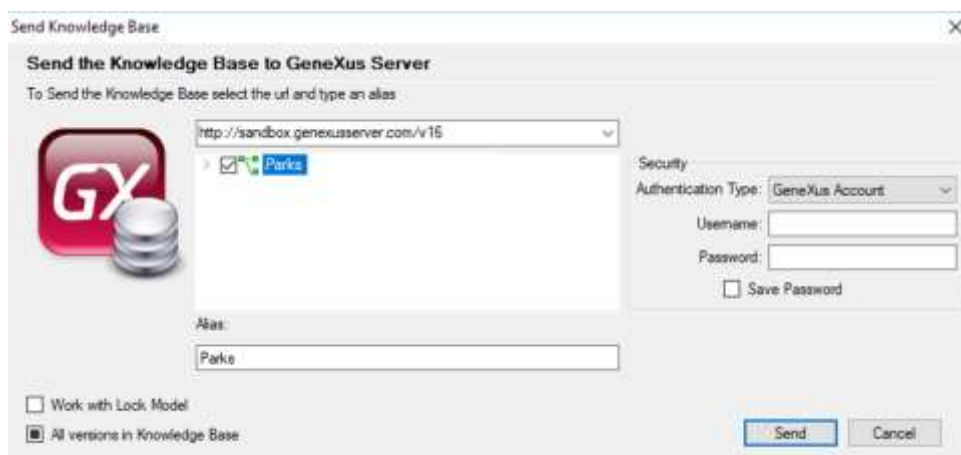
Crie um novo web panel que mostre a lista de parques de diversões registrados.

Envie este novo objeto ao servidor para que seja integrado à base de conhecimento centralizada.

Entre no console web e verifique o estado final da KB.



Feche a base de conhecimento anterior e crie uma nova sincronizando-se com a KB publicada anteriormente. Desta forma, se recebe localmente uma cópia da KB administrada pelo GeneXus Server.



Nesta nova cópia local, edite a transação *Country* e defina o novo atributo *CountryFlagImage*, do tipo *Image*. Envie esta alteração para o servidor.

Feche esta *KB* e abra novamente a *KB* inicial. Execute a operação *Update* para receber a alteração realizada anteriormente.

---



MONTEVIDEO - URUGUAY  
CIUDAD DE MÉXICO - MÉXICO  
MIAMI - USA  
SÃO PAULO - BRASIL  
TOKYO - JAPAN

Av. Italia 6201- Edif. Los Pinos, P1  
Hegel N° 221, Piso 2, Polanco V Secc.  
7300 N Kendall Drive, Suite 470  
Rua Samuel Morse 120 Conj. 141  
2-27-3, Nishi-Gotanda  
Shinagawa-ku, Tokyo, 141-0031

(598) 2601 2082  
(52) 55 5255 4733  
(1) 201 603 2022  
(55) 11 4858 0300  
(81) 3 6303 9381  
(81) 3 6303 9980