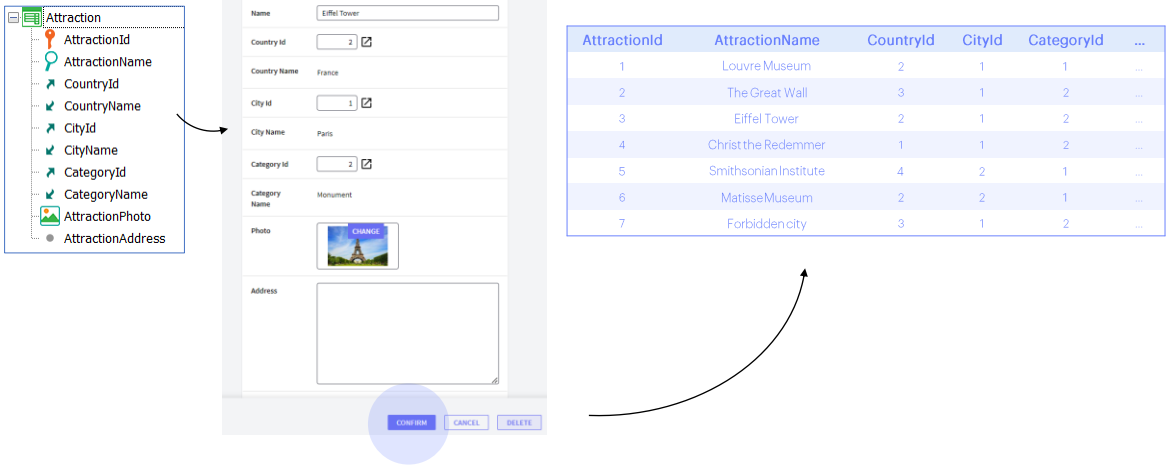
Atualização da base de dados

Usando Business Components. Justificativa

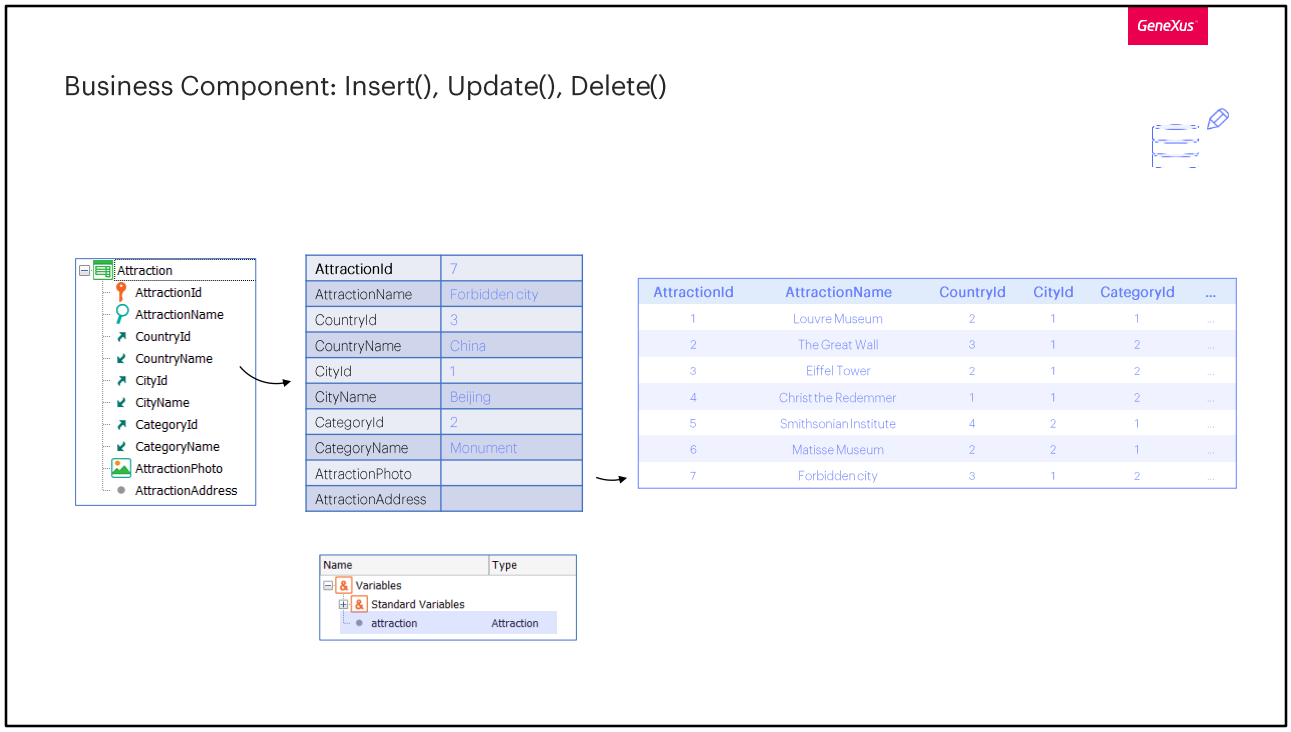
GeneXus™

GeneXus

Transaction: Insert, Update, Delete



Até agora, apenas atualizamos informações da base de dados através das transações, ou seja, de forma interativa, através de uma interface gráfica.



No que segue, vamos estudar como atualizar a informação da base de dados por código.

Vamos privilegiar uma das duas maneiras: a atualização através de business components, já entenderemos o porquê.
Trabalharemos com a estrutura da transação como se fosse uma variável SDT, respeitando também as regras da transação. Através dessa variável é que inseriremos, modificaremos ou excluiremos da base de dados. Será como utilizar a transação, mas sem sua tela.

GeneXus

Procedure: New, For each, Delete

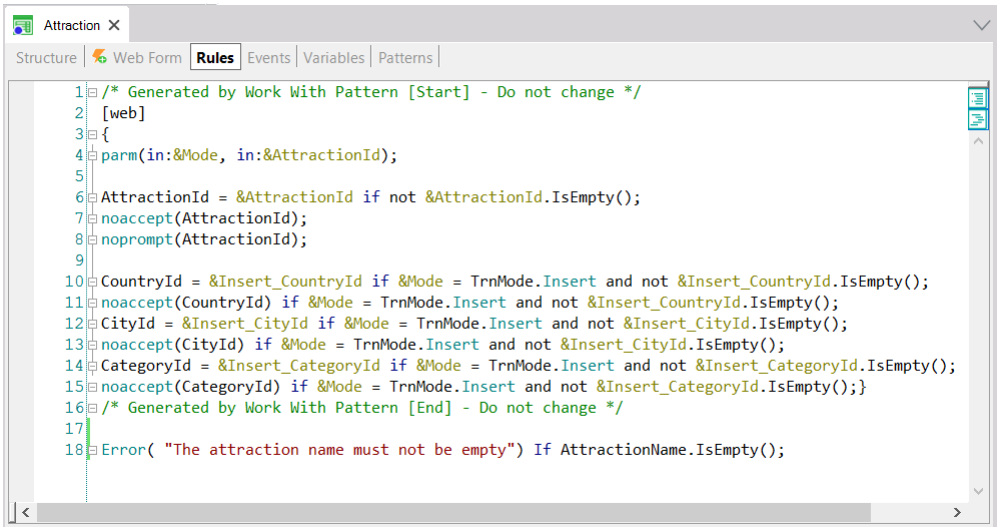
AttractionId	AttractionName	CountryId	CityId	CategoryId	...
1	Louvre Museum	2	1	1	...
2	The Great Wall	3	1	2	...
3	Eiffel Tower	2	1	2	...
4	Christ the Redemmer	1	1	2	...
5	Smithsonian Institute	4	2	1	...
6	Matisse Museum	2	2	1	...
7	Forbidden city	3	1	2	...

A outra maneira, é fazê-lo através de comandos especiais que só podem ser utilizados dentro de objetos do tipo procedimento. Os veremos mais adiante, em outro vídeo. Mas, neste caso, nos separamos da transação. Trabalhamos diretamente sobre as tabelas, o que tem suas desvantagens.

Então, estudemos esta primeira opção, a de mais alto nível e, portanto, a que em princípio nos será mais relevante.

GeneXus

Insert through the transaction



Observaremos primeiro, com certa atenção, o que acontece quando queremos inserir uma atração turística nova utilizando a transação.

Modificamos a ordem dos atributos para que a cidade fique logo após o país, o que influencia no form.

Vamos adicionar uma regra de erro para evitar que se deixe o nome da atração vazio.
Se agora pressionarmos a tecla Control mais a barra de espaço, nos abre esta janelinha que nos oferece todas as coisas que poderíamos colocar nesta parte do código. Se digitarmos “N”, já encontramos o atributo que procurávamos. Esta é uma forma de não cometer erros de digitação. A outra é com Insert > Attribute, cujo shortcut é Control+Shift+A, e ali escolhemos Attraction Name.

A esta transação tínhamos aplicado o pattern Work With e, por isso, estão aparecendo todas estas outras regras, adicionadas automaticamente pelo pattern para, por exemplo, permitir invocar a transação a partir do Work With, passando-lhe o modo (insert, update, delete) e o identificador de atração.

Para entender de forma mais simples o que nos propomos, preferimos ter

a transação invocável diretamente, sem parâmetros, tal como estava antes de lhe aplicar o pattern. Podemos desaplicá-lo, excluindo, recordemos, daqui a instância... ou, uma vez que vamos precisar dele para mais tarde, gravar esta transação com outro nome, por exemplo este.

GeneXus

Parallel transactions

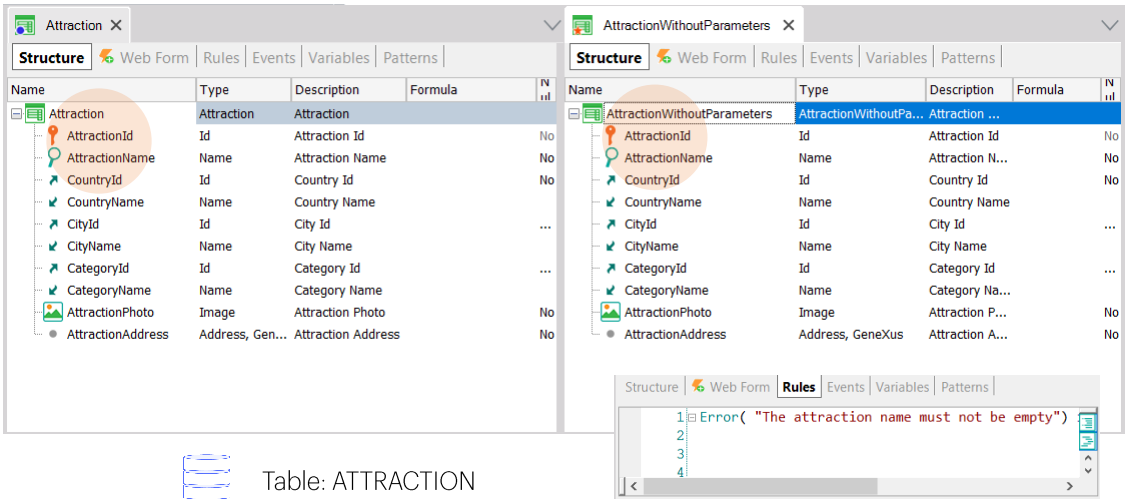


Table: ATTRACTION

Esta transação será idêntica à outra, exceto pelo nome e porque não tem o pattern aplicado, embora as regras que vieram do pattern e os eventos ficaram, assim simplesmente os excluímos. Deixamos a regra do erro e eliminamos todos os eventos que tinha adicionado o padrão.
A tabela sobre a qual irá inserir, modificar e excluir registros desta transação, vemos que é exatamente a mesma que a tabela da transação Attraction. Por quê? Porque o identificador é o mesmo. Estas transações recebem o nome de transações paralelas. Compartilham a tabela, mas os programas são independentes.
Para esta, por exemplo, poderíamos inclusive remover a regra de erro e, portanto, permitiria inserir registros que a outra não.

GeneXus

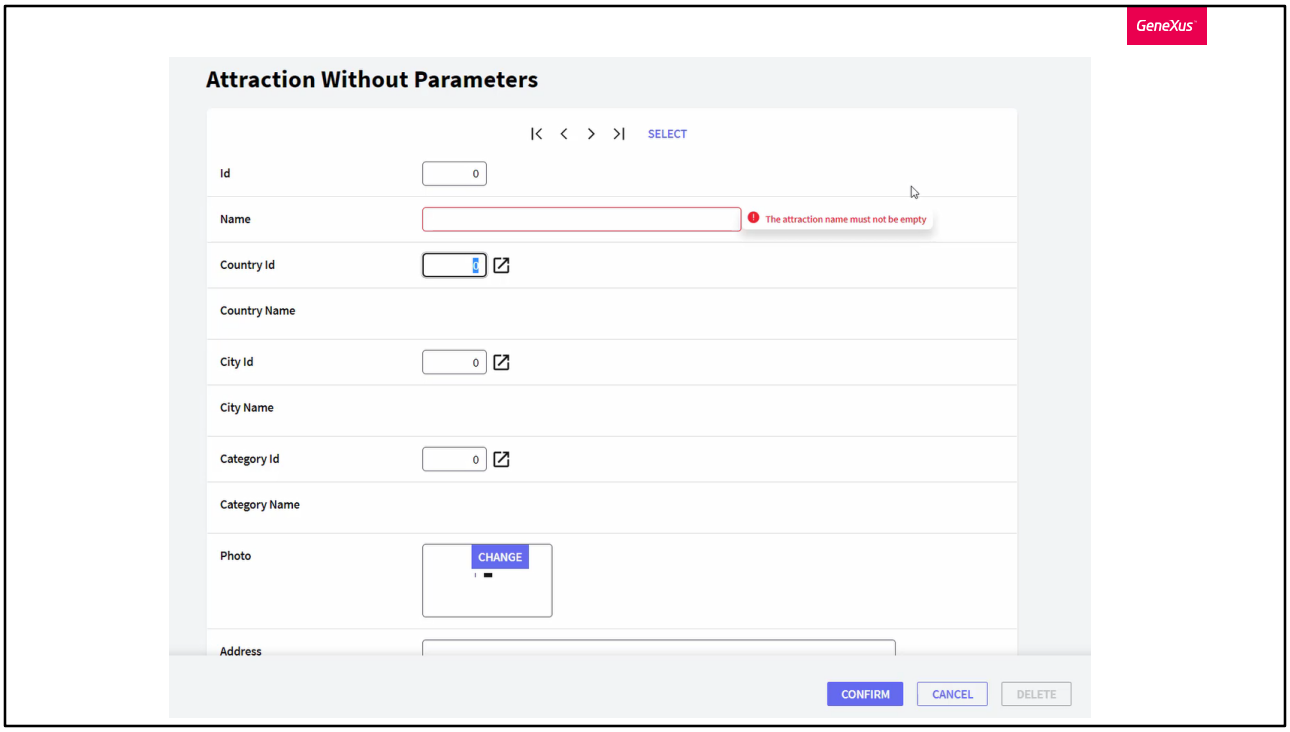
Transaction: Insert, Update, Delete

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	Christ the Redemmer	1	1	2	...
5	Smithsonian Institute	4	2	1	...
6	Matisse Museum	2	2	1	...

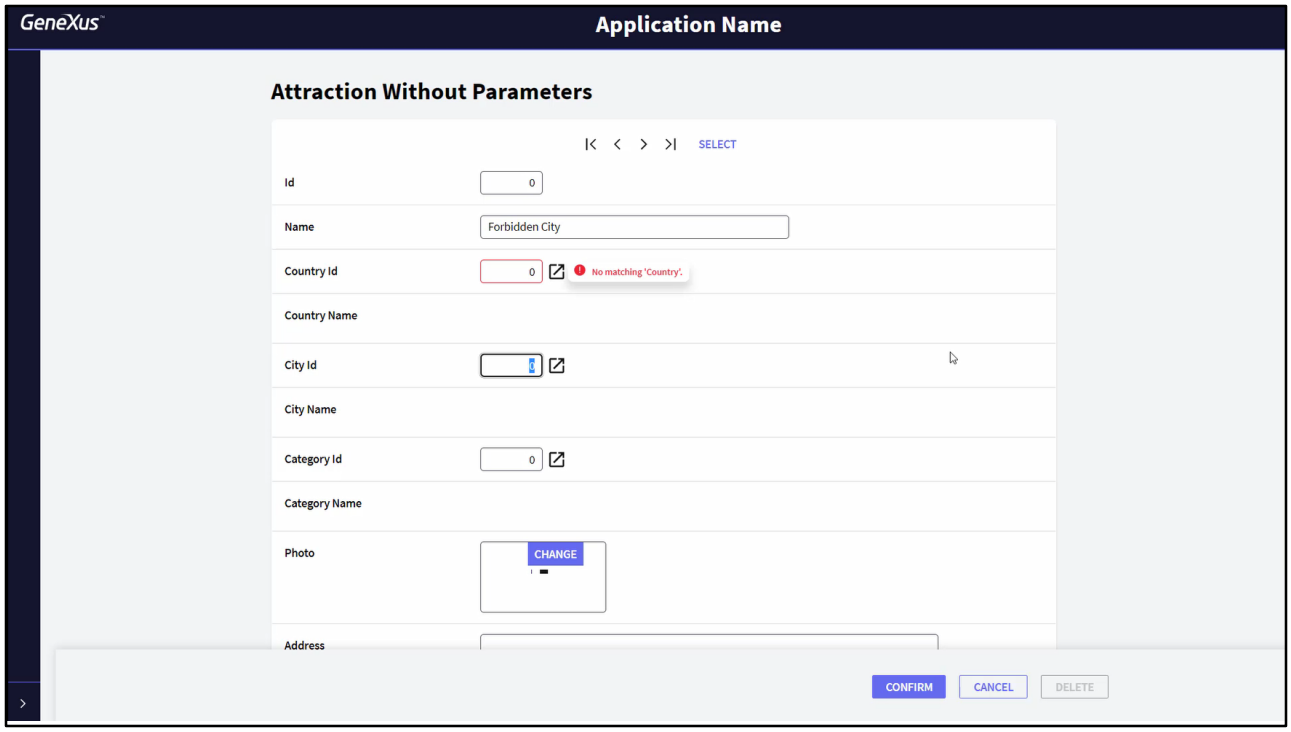
Pressionamos F5 para executá-la.

Vemos que se abre com os campos vazios, e ativo o do identificador, na espera de que o usuário insira ali um valor, para poder inferir depois, de que operação se tratará. Se uma inserção ou, pelo contrário, uma atualização ou eliminação.

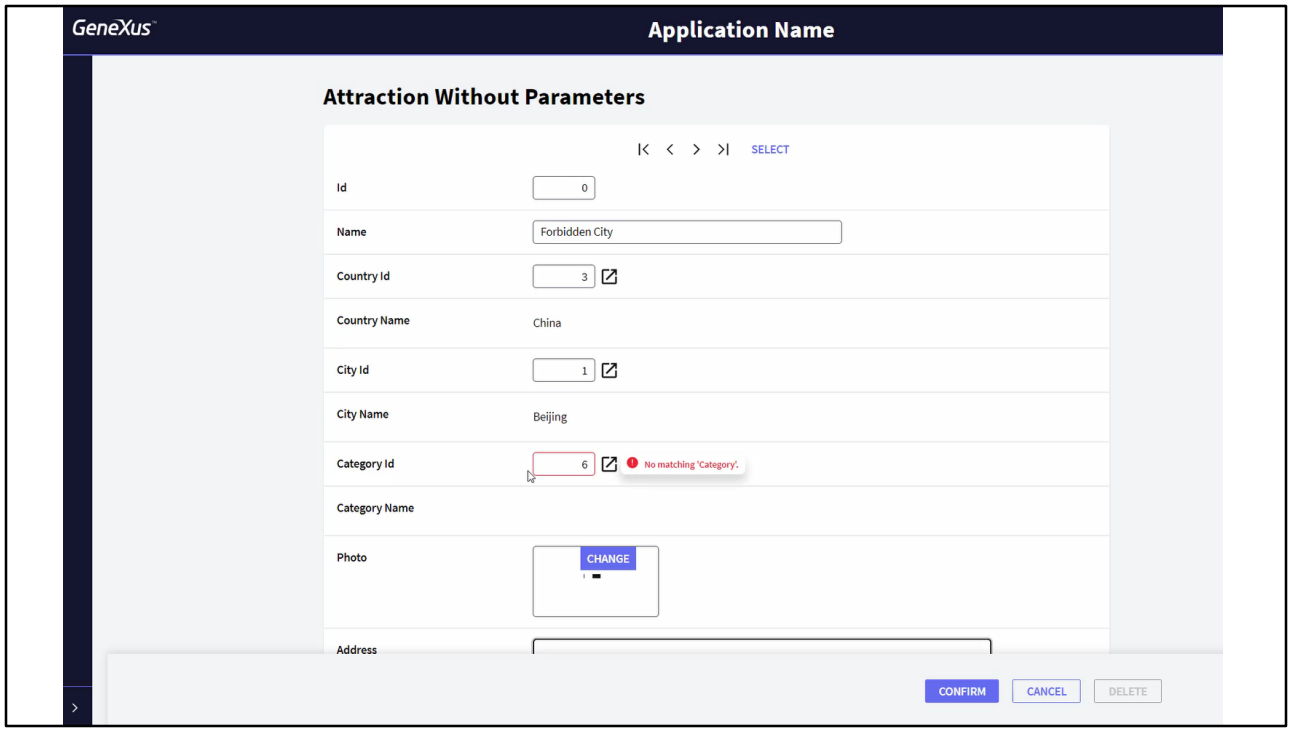
Assim, ao sair do campo, GeneXus buscará na tabela se existe um registro com o valor indicado. Caso exista, a transação ficará em modo Update, e com os campos com os valores correspondentes. Em vez disso, se não existir, a transação ficará em modo Insert e todos os campos estarão vazios (a menos que haja alguma regra Default que dependa apenas do modo, ou alguma atribuição que só esteja condicionada com If Insert. Por exemplo, AttractionName igual a algo If Insert. Neste caso não há nenhuma).



Como AttractionId é autonumerado, o usuário provavelmente deixa o valor 0 quando quer inserir uma nova atração. A transação então, ficará em modo Insert. Se sai agora do campo seguinte, AttractionName, sem inserir um valor, será exibida uma mensagem de erro, pela regra que tínhamos programada. Mas mesmo assim, nos permite continuar inserindo os outros pois, de qualquer forma, ao Confirmar não nos permitirá gravar, continuará nos mostrando a mensagem. Então, inserimos a cidade proibida.



Quando se trata de chaves estrangeiras, ao sair do campo verifica-se que um registro com esse valor exista na tabela associada. Caso contrário, lança o erro "No matching" da integridade referencial. Também podemos continuar, mas, tal como no outro erro, também não nos deixará gravar. A cidade proibida está na China, e vemos que ao escolher sua chave, já nos infere seu nome. O mesmo ocorre para a cidade.

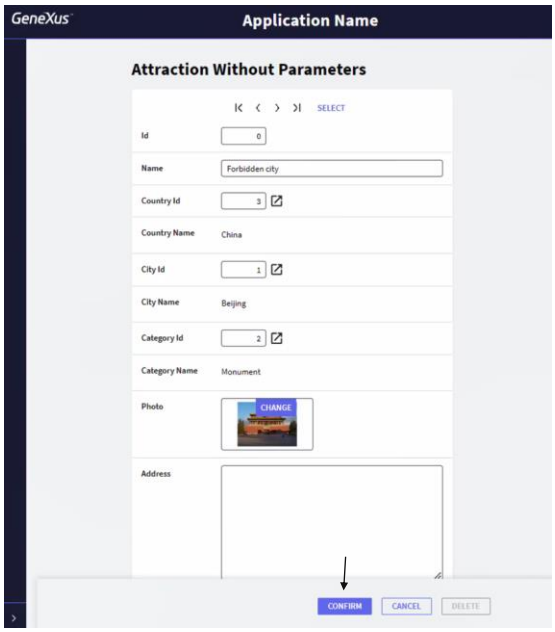


Aqui temos outra chave estrangeira, a categoria, que não nos permite deixar com um valor inexistente, mas sim vazia. Recordemos que indicamos que o atributo aceitaria nulos. Mas na realidade, sabemos sua categoria, será Monument.

Observemos que para realizar estes controles de integridade referencial e trazer os nomes inferidos, não houve outra escolha, a não ser ir ao servidor, que é o que realmente controla a base de dados. Todas as regras e controles, que vão sendo disparadas campo a campo no navegador do usuário, estão ali para dar agilidade à experiência de usuário, fazendo-o sentir que não há períodos ociosos. A isto, chamamos Client Side Validation, ou seja, validação do lado do cliente. Mas tudo isto terá que ser repetido no servidor quando o usuário confirmar.

Claro que podemos deixar vazios os campos que não sejam chaves estrangeiras default, ou seja, que não aceitem nulos, e os que não tenham regras explícitas que o impeçam. Vamos colocar uma foto, mas sem endereço.

E agora, sim, vamos confirmar. Nos informa que a inserção foi realizada com sucesso. Mas o que aconteceu por trás?



GeneXus

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	Christ the Redemmer	1	1	2	...
5	Smithsonian Institute	4	2	1	...
6	Matisse Museum	2	2	1	...
7	Forbidden city	3	1	2	

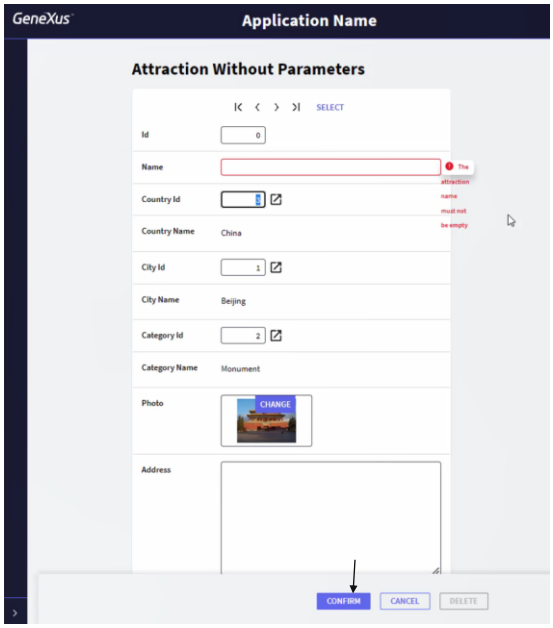
AttractionId	0
AttractionName	Forbidden city
CountryId	3
CountryName	
CityId	1
CityName	
CategoryId	2
CategoryName	
AttractionPhoto	
AttractionAddress	

Ao pressionar Confirm todas as informações inseridas pelo usuário nos campos deverá viajar ao servidor, que voltará a começar do zero, porque é ele quem assegura para que não haja violações de segurança. O browser é sempre um ambiente hostil. É o servidor o encarregado de que o programa faça o que tem codificado. E em tempo imperceptível para o usuário. E é ele o único que tem permissão de operar sobre a base de dados.

Podemos imaginar, apenas para fins práticos, que é como se fossem tomados todos os atributos da estrutura da transação e montado com eles um SDT que é carregado com os valores que, interativamente, o usuário lhes deu no form (os que importam são os não virtuais, ou seja, os que estarão fisicamente na tabela; aqui são estes).

Com esta estrutura carregada no server, se executa tudo do zero: as validações, regras e fórmulas, passando por cada elemento, como o que o usuário fez interativamente.

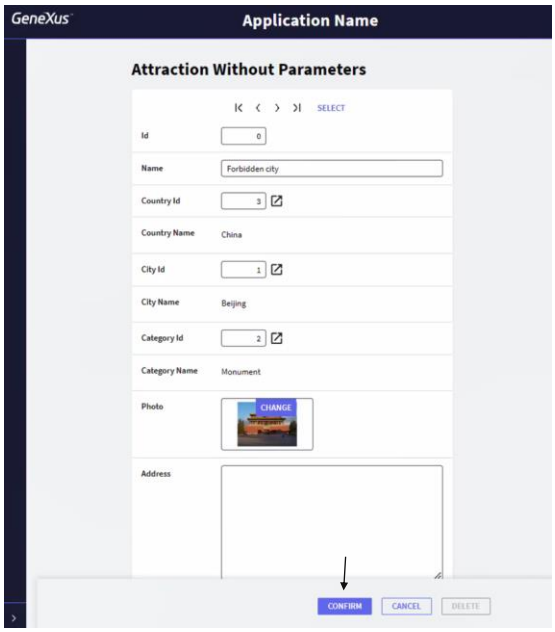
Se consegue terminar sem erros, então insere o registro na base de dados.



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	Christ the Redemmer	1	1	2	...
5	Smithsonian Institute	4	2	1	...
6	Matisse Museum	2	2	1	...

AttractionId	0
AttractionName	
CountryId	3
CountryName	
CityId	1
CityName	
CategoryId	2
CategoryName	
AttractionPhoto	
AttractionAddress	

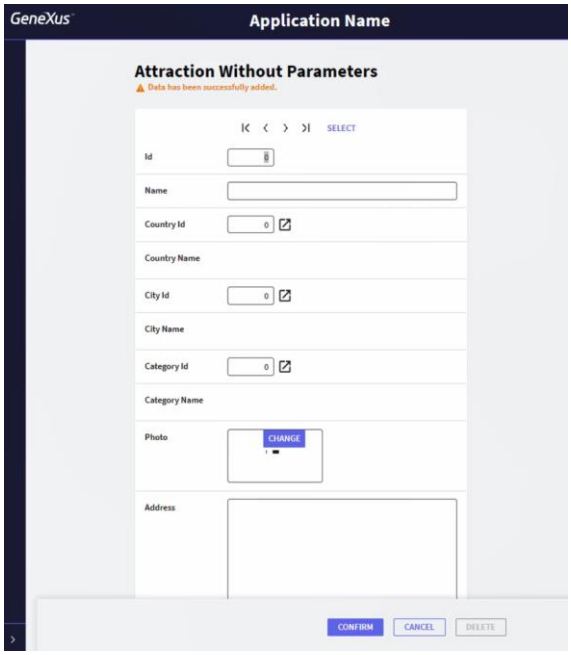
Se ao pressionar Confirm, tivéssemos deixado o AttractionName vazio, saltará a regra de erro e não será permitida a inserção na base de dados, mostrando o erro ao usuário no browser. O mesmo acontecerá se deixamos uma valor de chave estrangeira inexistente.



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	Christ the Redemmer	1	1	2	...
5	Smithsonian Institute	4	2	1	...
6	Matisse Museum	2	2	1	...
7	Forbidden city	3	1	2	

AttractionId	7
AttractionName	Forbidden city
CountryId	3
CountryName	China
CityId	1
CityName	Beijing
CategoryId	2
CategoryName	Monument
AttractionPhoto	
AttractionAddress	

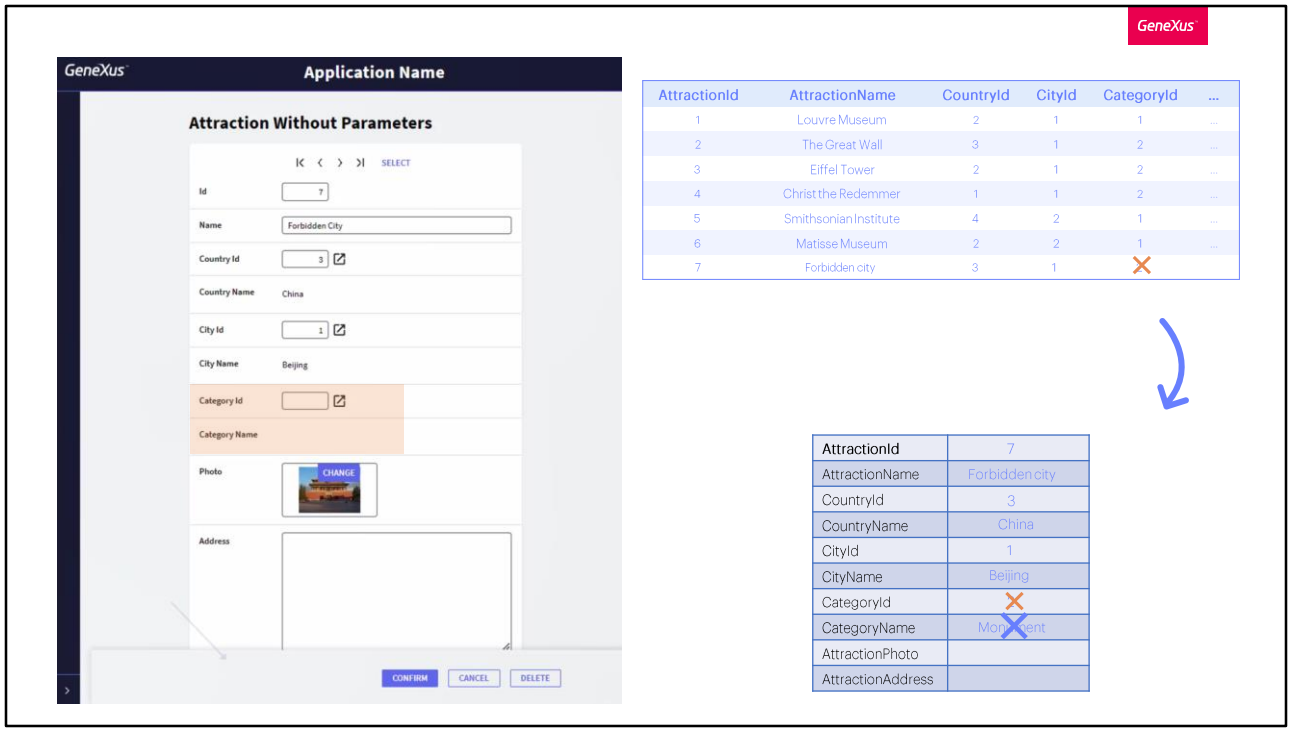
Mas se tudo correu bem, neste caso, como AttractionId é autonumerado, ao inseri-lo na tabela lhe será dado o número correspondente e o SDT também ficará atualizado, com esse dado e com os dados inferidos e fórmulas que correspondam (aqui não há fórmulas) caso fosse necessário fazer algo mais com ele (se se tratasse de uma transação de dois níveis, ainda faltaria trabalhar com as linhas).



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	Christ the Redemmer	1	1	2	...
5	Smithsonian Institute	4	2	1	...
6	Matisse Museum	2	2	1	...
7	Forbidden city	3	1	2	

AttractionId	
AttractionName	
CountryId	
CountryName	
CityId	
CityName	
CategoryId	
CategoryName	
AttractionPhoto	
AttractionAddress	

O que vemos em execução é que após a inserção, a tela fica vazia, com a mensagem de que os dados foram inseridos com sucesso. A transação volta a ficar em modo Insert, ou seja, volta a estar pronta para que o usuário insira uma nova atração. Podemos pensar também que é excluída essa estrutura no servidor, para ser novamente criada quando o processo seja reiniciado.



Se agora inserirmos na chave este valor, 7, e deixamos o campo... o que vemos no browser será isto. A transação terá ido ao servidor, que irá à base de dados para consultar a existência de um registro com esse valor. O encontrará. Podemos, outra vez, pensar que carrega todos os seus valores (os físicos e os inferidos e fórmulas) em uma estrutura como a anterior, e a envia ao cliente, com a informação de que então, agora, o modo da transação será Update.

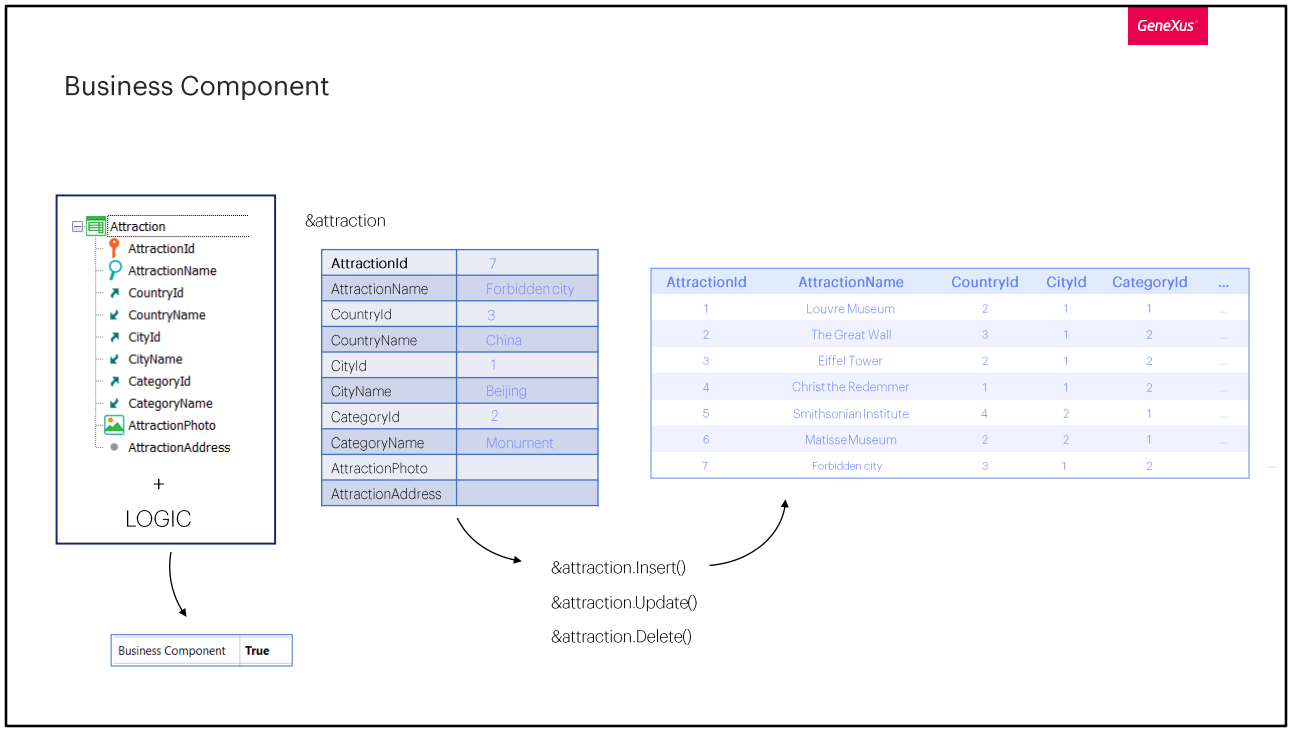
Outra vez, o usuário interativamente fará as modificações que deseja, por exemplo, exclui a categoria (que como aceita nulos, será permitido). Ao confirmar, tudo volta a se realizar no servidor: carrega-se o registro da base de dados, realizam-se as modificações feitas no cliente, e vai-se validando campo a campo, disparando as regras que correspondam. Se nada falhar, será atualizado o registro da tabela, neste caso, excluindo a categoria, e atualizará a estrutura, removendo também o nome de categoria, que era inferido.

Se nenhuma outra regra da transação o impede, então no browser fica a informação atualizada, e uma mensagem que o indica. Observemos que a transação fica em modo Update. Poderíamos voltar a modificar este registro, por exemplo, adicionando outra vez a categoria.

Se agora quiséssemos removê-la, bastaria pressionar o botão Delete, e no server, com a estrutura carregada será fácil ir procurar na tabela a atração 7 para

removê-la.

Com isto já fica bastante claro que se conseguíssemos trabalhar com uma variável estruturada como a que imaginamos, utiliza internamente a transação no servidor, que encapsule as regras da transação e além disso nos permita realizar as operações sobre a tabela, poderíamos inserir, modificar e apagar da base de dados por código, respeitando a lógica declarada na transação.



Isto é nada mais, nem menos do que se conhece como Business Component.

Da estrutura da transação com sua lógica (e por lógica entendemos os controles de duplicados –não só os de chave primária, mas também de chaves candidatas-, de integridade referencial, suas regras e alguns de seus eventos), obtém-se uma espécie de tipo de dados semelhante a um SDT, mas muito mais potente.

Então, conseguirá definir, em quase qualquer programa, uma variável desse tipo de dados e manipulá-la, que é o que veremos a seguir.

GeneXus™

training.genexus.com
wiki.genexus.com