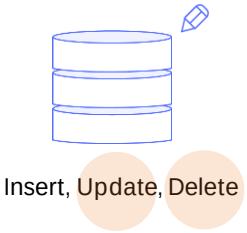
Atualização para a Base de dados com comandos específicos de procedimentos.

Como excluir (delete)

GeneXus

For each Command

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



For each Attraction
Where AttractionName = "Eiffel Tower"

AttractionId = 5
CategoryId = 3

endfor

For each Attraction
Where AttractionName = "Eiffel Tower"

new
AttractionId = 5
CategoryId = 3
endnew

Delete

endfor

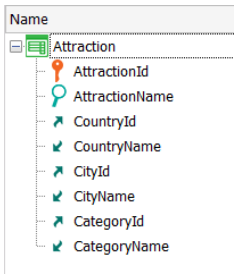
No vídeo sobre atualização com For each em um procedimento, vimos um caso em que precisávamos modificar o valor da chave primária de um registro, para o qual tivemos que criar um novo, com o novo valor de chave e excluir o antigo.

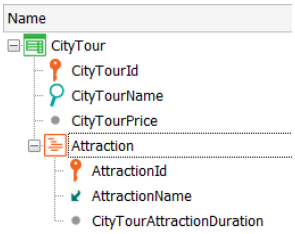
E fizemos isto nos posicionando sobre o registro em questão, criando um novo com new; e imediatamente executando o comando Delete para excluir o registro do for each no qual estávamos posicionados.

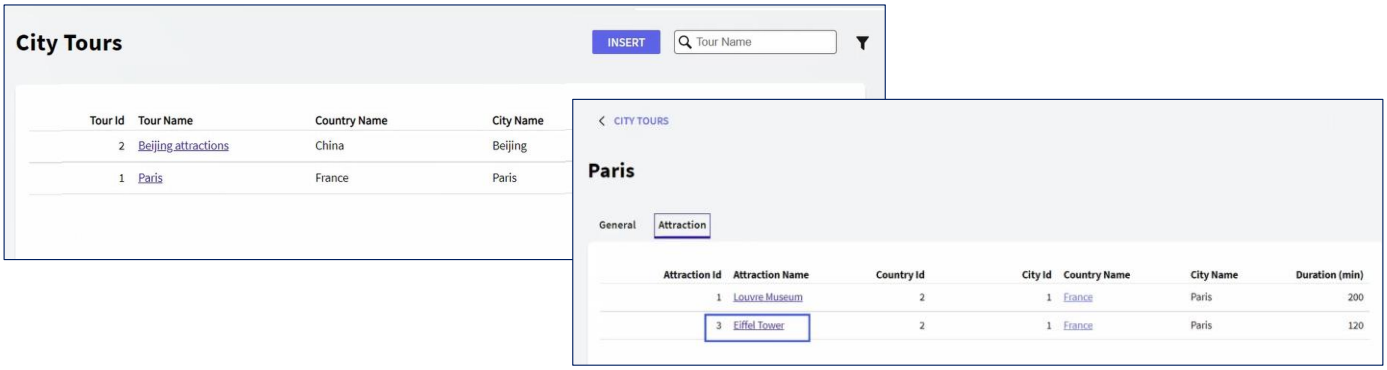
Assim será geralmente a eliminação. Utilizando for each para escolher registro e executando Delete para excluí-lo.

Delete

Vamos estudar os detalhes da exclusão.

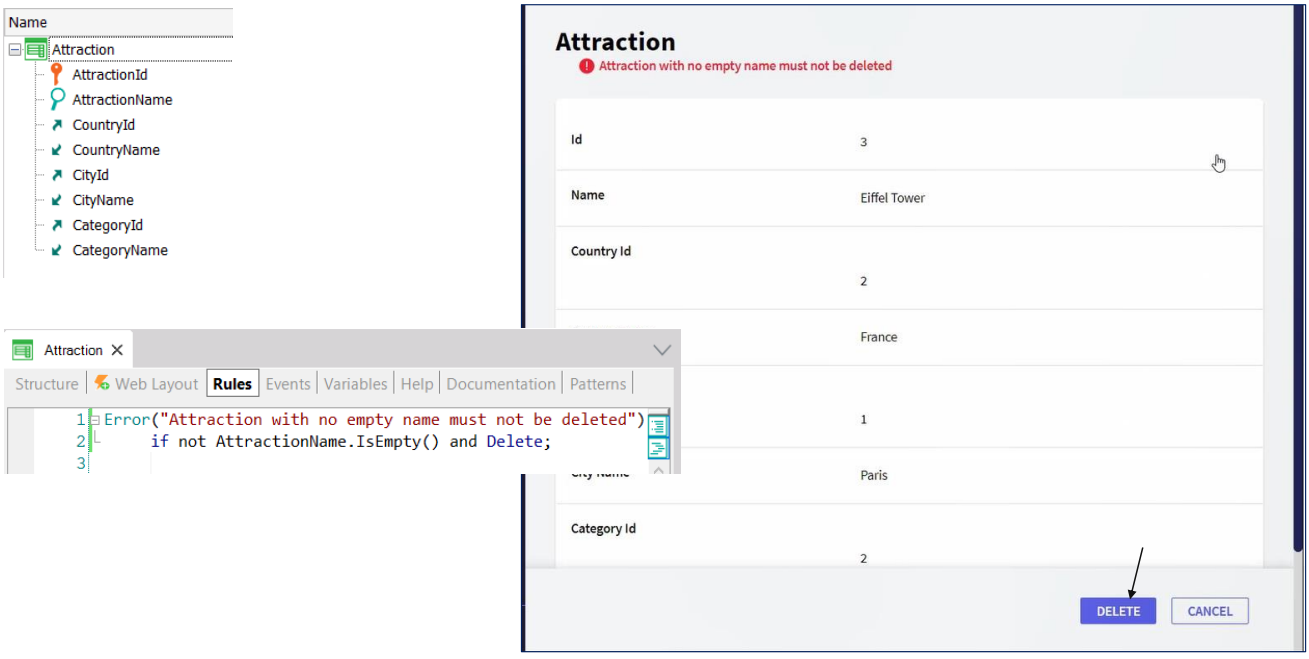






Lembremos as transações que vínhamos utilizando para estudar a inserção e atualização da base de dados por procedimentos.

Aqui tínhamos CityTour, onde podíamos especificar por quais atrações turísticas seria realizado o tour atual. Por exemplo, se vemos em execução, tínhamos este tour por Paris, que visitaria as atrações museu do Louvre e Torre Eiffel.



Por outro lado, tínhamos a transação que registra as atrações. E adicionamos esta regra de error para não permitir a exclusão de uma atração com nome inserido.

Então, vamos ver que, se tentamos excluir a atração Torre Eiffel por meio da transação, não será permitido.

Exclusão por Procedimento?

Agora, o que acontece se tentarmos removê-la por meio de um procedimento?

Sabemos que esta atração, Torre Eiffel, faz parte de um city tour, e também tem um nome atribuído, que é justamente Eiffel Tower. Então, nos permitirá removê-la?

	Uniqueness check	Referential Integrity check	Rules/Events execution
Delete in For each	✕	✕	✕

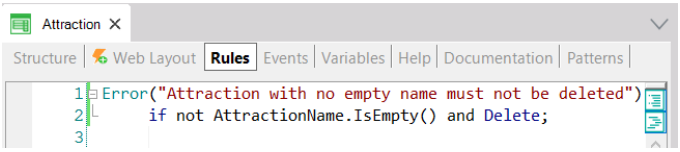
CityTourId	AttractionId	CityTourAttractionDuration
1	1	200
1	3	120
2	2	240
2	4	240

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

For each Attraction
Where AttractionName = "Eiffel Tower"

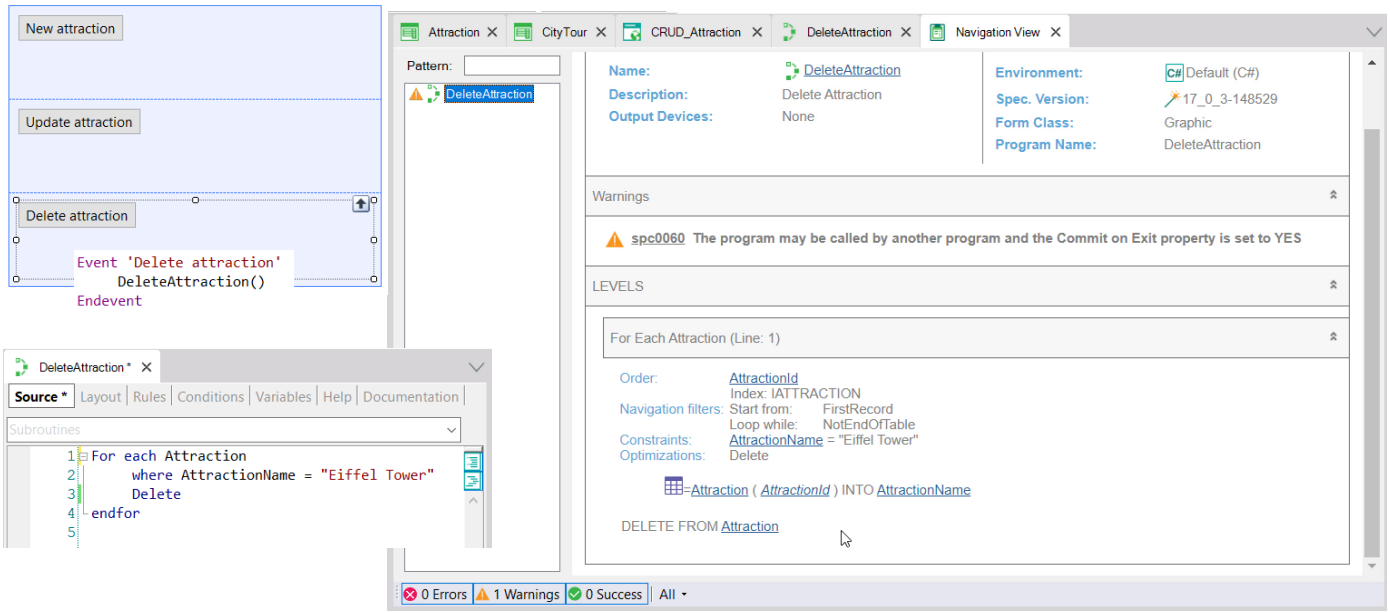
Delete

endfor

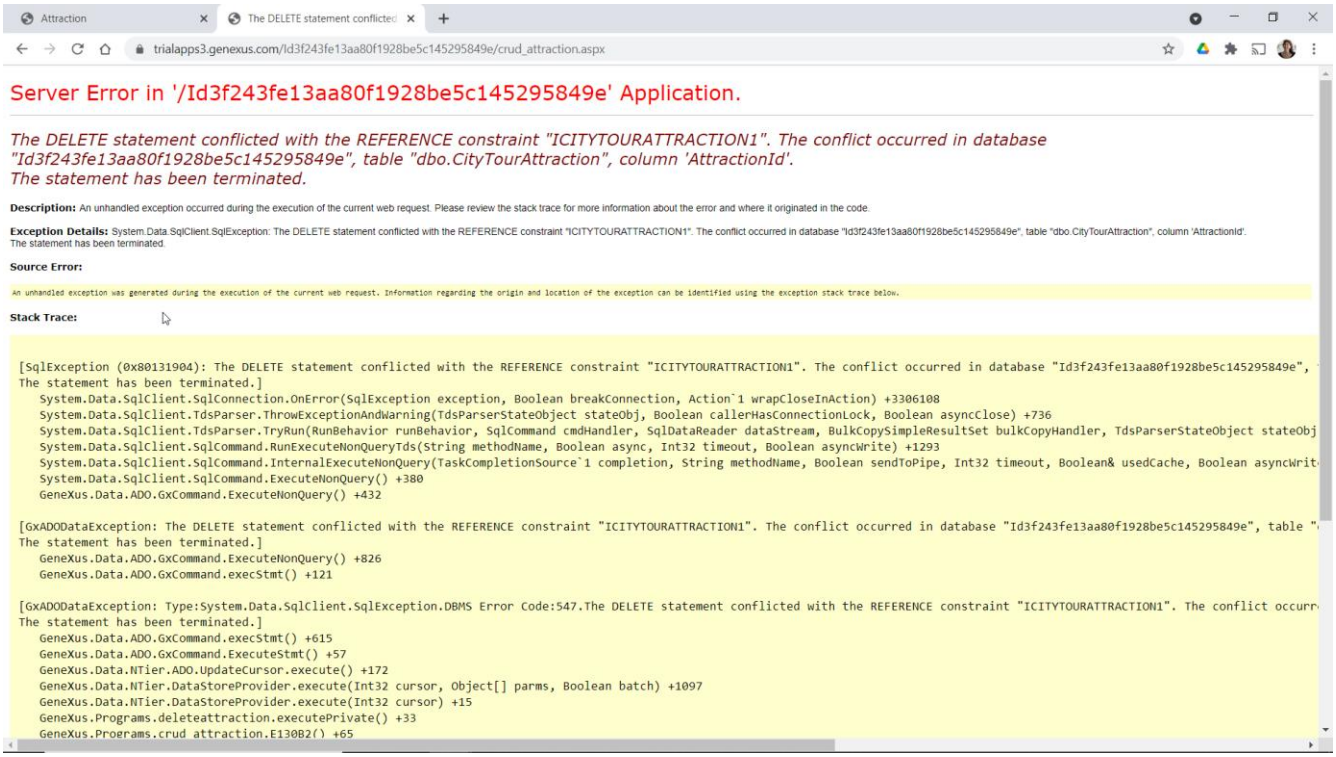


Se pensarmos em tudo o que vimos até agora sobre os comandos de atualização da base de dados por procedimentos, podemos responder que SIM, que nos permitirá eliminar a atração, porque:

- **NÃO** são realizadas verificações de integridade referencial programáticas, ou seja, não será verificado que não exista um city tour que esteja referenciando a atração a ser eliminada.
- Além disso, **não** são executadas regras da transação associada à tabela da qual está sendo removido um registro.



Portanto se formos ao GeneXus e observarmos que programamos este botão que invoca este procedimento... que percorre com um for each as atrações, filtrando pela de nome “Eiffel Tower” e para os registros encontrados (no nosso caso será apenas um), os elimina com o comando Delete...



Se executamos... cai o programa. Por quê?

Pelo mesmo motivo que vimos antes.

	Uniqueness check	Referential Integrity check	Rules/Events execution
Delete in For each	✕	✕	

CityTourId	AttractionId	CityTourAttractionDuration
1	1	200
1	3	120
2	2	240
2	4	240

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

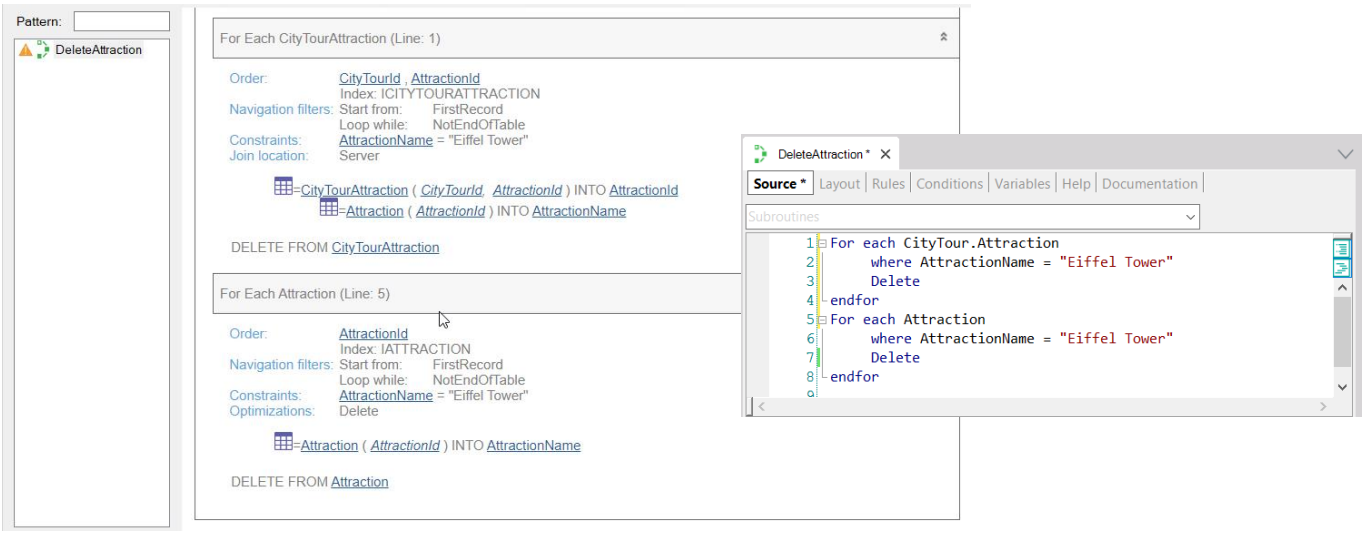
```
For each Attraction
    Where AttractionName = "Eiffel Tower"

    Delete

endfor
```

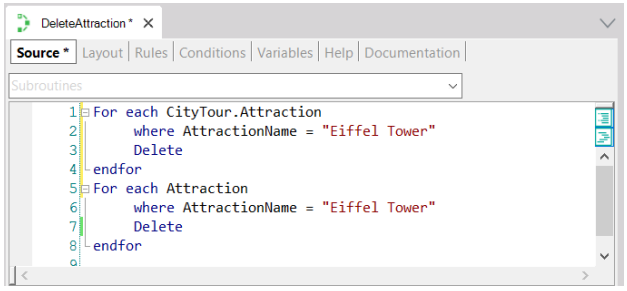
exception

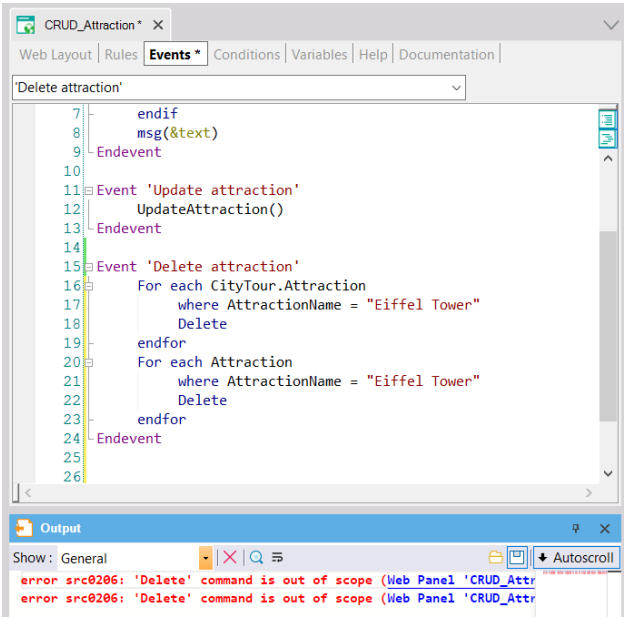
O Delete não verifica a integridade referencial, mas por padrão o base de dados sim, e não estamos capturando a exceção que produz.



Portanto, se quisermos remover essa atração, antes devemos removê-la de todos city tours nos quais ela se encontre. Assim... Executemos.

Como observação óbvia, o comando Delete apenas exclui o registro da tabela base do For each em que estamos posicionados. Não exclui registros da tabela estendida. Neste sentido, funciona como o new.





E outra coisa que já destacamos muitas vezes, mas que é importante enfatizar novamente: o comando Delete só pode ser utilizado dentro de um for each e em um procedimento. Não poderíamos ter programado a eliminação diretamente dentro do evento, por exemplo.

Ao contrário do que acontece quando excluímos por meio de Business Component.

	Uniqueness check	Referential Integrity check	Rules/Events execution
Delete in For each	✕	✕	✕

CityTourId	AttractionId	CityTourAttractionDuration
1	1	200
1	3	120
2	2	240
2	4	240

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

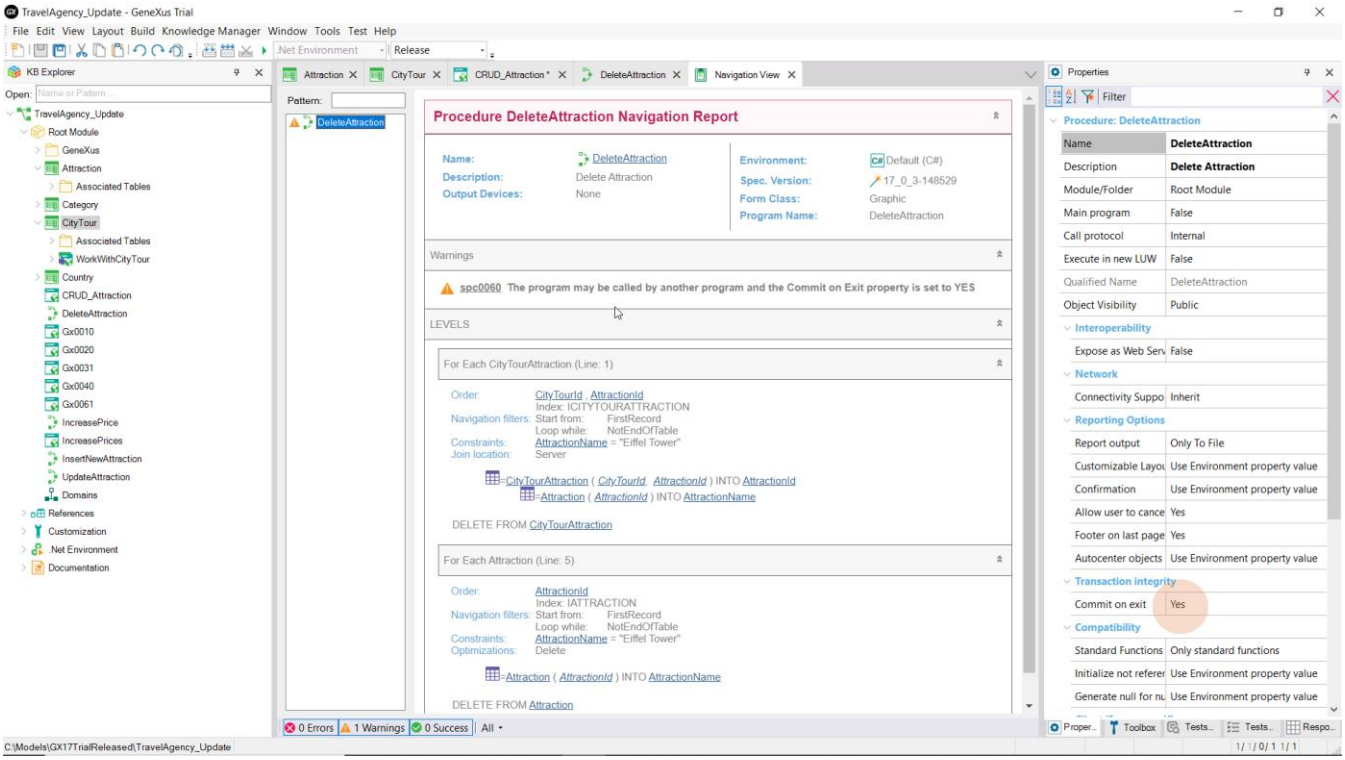
```
For each CityTour.Attraction
      Where AttractionName = "Eiffel Tower"
      Delete
endfor
For each Attraction
      Where AttractionName = "Eiffel Tower"
      Delete
endfor
Commit
```

COMMIT?

Transaction integrity	
Commit on exit	Yes

Por último, neste procedimento estamos eliminando dois registros. Quando esta operação fica commitada na base de dados?

Acontece a mesma coisa que vimos com o new e a atualização com for each. Se a propriedade Commit on exit permanece com seu valor default, que é Yes, como GeneXus descobre que se deseja realizar uma exclusão na base de dados, automaticamente adiciona ao final do Source do procedimento um comando Commit.



E é por isso que a lista de navegação nos mostra um aviso que o indica, para que saibamos que mesmo que não tenhamos especificado explicitamente um commit, GeneXus o adicionará.

	Uniqueness check	Referential Integrity check	Rules/Events execution
Delete in For each	✕	✕	✕

CityTourId	AttractionId	CityTourAttractionDuration
1	1	200
1	3	120
2	2	240
2	4	240

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

For each CityTour.Attraction
Where AttractionName = "Eiffel Tower"
Delete
Endfor
Commit
For each Attraction
Where AttractionName = "Eiffel Tower"
Delete
Endfor
Commit

For each CityTour.Attraction
Where AttractionName = "Eiffel Tower"
Delete
Commit
Endfor
For each Attraction
Where AttractionName = "Eiffel Tower"
Delete
Commit
Endfor

Claro, poderíamos programar um commit depois de cada for each.

Ou mesmo depois de cada Delete (porque no nosso caso o for each irá recuperar um único registro, mas poderiam ser vários).

Summary

For each *BaseTransaction*
skip $expression_1$ **count** $expression_2$
order att_{11}, att_{12}, ... att_{1n} **[when** *condition***]**
order att_{21}, att_{22}, ... att_{2n} **[when** *condition* | **otherwise]**
using *DataSelector*(*parm*$_1$, ..., *parm*$_n$)
unique att_1, ..., att_n
where *condition* **[when** *condition***]**
where *condition* **[when** *condition***]**
where *att* **in** *DataSelector*(*parm*$_1$,..., *parm*$_n$)
blocking *NumericExpression*
...
Delete
...
When duplicate
...
When none
...
endfor

	Uniqueness check	Referential Integrity check
Delete	✕	✕

COMMIT

Resumidamente, para excluir registros especificamente por procedimento, temos o comando Delete que deve ser utilizado dentro do comando For each.

A eliminação é do registro da tabela base do for each no qual este se encontre posicionado em cada iteração.

No caso da exclusão, não faz sentido um controle de unicidade de chaves, porque nada está sendo inserido ou atualizado. E assim como vimos com a inserção e a atualização, o Delete não realiza programaticamente controle de integridade referencial algum. Isto, novamente, é por motivos de desempenho. No entanto, as bases de dados em geral o realizam, a menos que desativemos essa funcionalidade; portanto, se não for desativada e falhar a integridade, uma exceção será produzida.

Por último: para que o registro fique commitado na base de dados, isto é, seja excluído de forma permanente, é necessário garantir que o comando Commit seja executado. Em um procedimento, por padrão, é colocado um Commit implícito no final (desde que seja entendido que no Source está sendo acessada em algum lugar a base de dados para atualizá-la). Mas podemos escrever explicitamente Commits no Source, onde for conveniente para nós.

Novamente, não veremos isso aqui, mas opcionalmente pode ser especificada uma cláusula Blocking, que o que faz é permitir fazer

exclusões em bloco, em vez de registro por registro. Ou seja, vai processar os registros em blocos de a N para reduzir os acessos e melhorar o desempenho.

Tanto para a inserção, quanto para a atualização, quanto para a eliminação, as redundâncias não são gerenciadas automaticamente quando são feitas por procedimento. É responsabilidade do desenvolvedor gerenciá-las.

Sobre tudo isto podemos ver mais em nossa wiki.

GeneXus™

training.genexus.com
wiki.genexus.com