

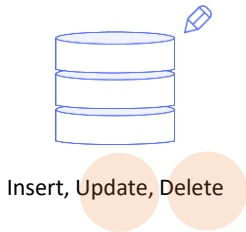
Atualização para a Base de dados com comandos específicos de procedimentos.

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.

The image displays two database models and a corresponding user interface. The first model, 'Attraction', lists fields: AttractionId, AttractionName, CountryId, CountryName, CityId, CityName, CategoryId, and CategoryName. The second model, 'CityTour', lists fields: CityTourId, CityTourName, CityTourPrice, and an 'Attraction' relationship with sub-fields AttractionId, AttractionName, and CityTourAttractionDuration.

The user interface shows a 'General' tab for 'Attraction' with 'UPDATE' and 'DELETE' buttons. Below, a 'City Tour Information' section displays a table for 'Paris' with columns for 'Attraction Id', 'Attraction Name', and 'Attraction Duration'.

Attraction Id	Attraction Name	Attraction Duration
1	Louvre Museum	200
3	Eiffel Tower	120

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.

The image shows a screenshot of the GeneXus IDE. On the left, a tree view shows the 'Attraction' entity with fields: AttractionId, AttractionName, CountryId, CountryName, CityId, CityName, CategoryId, and CategoryName. In the center, a 'Rules' editor window is open, displaying the following code:

```
1 Error("Attraction with no empty name must not be deleted")
2   if not AttractionName.IsEmpty() and Delete;
3
```

On the right, a data form for 'Attraction' is shown. The form has fields for Id, Name, Country Id, Country Name, City Id, City Name, Category Id, and Category Name. The 'Name' field contains 'Eiffel Tower' and is highlighted with a red border. A yellow tooltip message reads: 'Attraction with no empty name must not be deleted'. At the bottom of the form, there are three buttons: 'CONFIRM', 'CANCEL', and 'DELETE'. An arrow points to the 'DELETE' button.

Por outro lado, tínhamos a transação que registra as atrações. E adicionamos esta regra de erro 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

```

1 Error("Attraction with no empty name must not be deleted")
2   if not AttractionName.IsEmpty() and Delete;
3

```

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.

The screenshot displays the GeneXus IDE interface for developing a program. On the left, a design view shows a form with three buttons: "New attraction", "Update attraction", and "Delete attraction". The "Delete attraction" button is connected to an event named "Delete attraction", which is configured to call the "DeleteAttraction()" subroutine. Below the design view, the source code for the "DeleteAttraction" subroutine is visible, showing a "For each Attraction" loop with a "where AttractionName = 'Eiffel Tower'" condition and a "Delete" statement. The main workspace shows the "DeleteAttraction" program details, including its name, description, and environment. Below this, a "Warnings" section displays a warning: "spc0060 The program may be called by another program and the Commit on Exit property is set to YES". The "LEVELS" section shows a "For Each Attraction (Line: 1)" loop with the following code: "Order: AttractionId", "Index: IATTRACTION", "Navigation filters: Start from: FirstRecord, Loop while: NotEndOfTable", "Constraints: AttractionName = 'Eiffel Tower'", "Optimizations: Delete", and a table operation: "=Attraction (AttractionId) INTO AttractionName". The final line of code is "DELETE FROM Attraction". At the bottom, a status bar indicates "0 Errors", "1 Warnings", and "0 Success".

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...

Server Error in '/Id3f243fe13aa80f1928be5c145295849e' Application.

The DELETE statement conflicted with the REFERENCE constraint "ICITYTOURATTRACTION1". The conflict occurred in database "Id3f243fe13aa80f1928be5c145295849e", table "dbo.CityTourAttraction", column 'AttractionId'. The statement has been terminated.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: The DELETE statement conflicted with the REFERENCE constraint "ICITYTOURATTRACTION1". The conflict occurred in database "Id3f243fe13aa80f1928be5c145295849e", table "dbo.CityTourAttraction", column 'AttractionId'. The statement has been terminated.

Source Error:

An unhandled exception was generated during the execution of the current web request. Information regarding the origin and location of the exception can be identified using the exception stack trace below.

Stack Trace:

```
[SqlException (0x80131904): The DELETE statement conflicted with the REFERENCE constraint "ICITYTOURATTRACTION1". The conflict occurred in database "Id3f243fe13aa80f1928be5c145295849e", The statement has been terminated.]
System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction) +3306108
System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Boolean callerHasConnectionLock, Boolean asyncClose) +736
System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopyHandler, TdsParserStateObject stateObj)
System.Data.SqlClient.SqlCommand.RunExecuteNonQueryTds(String methodName, Boolean async, Int32 timeout, Boolean asyncWrite) +1293
System.Data.SqlClient.SqlCommand.InternalExecuteNonQuery(TaskCompletionSource`1 completion, String methodName, Boolean sendToPipe, Int32 timeout, Boolean& usedCache, Boolean asyncWrite)
System.Data.SqlClient.SqlCommand.ExecuteNonQuery() +380
GeneXus.Data.ADO.GxCommand.ExecuteNonQuery() +432

[GxADODataException: The DELETE statement conflicted with the REFERENCE constraint "ICITYTOURATTRACTION1". The conflict occurred in database "Id3f243fe13aa80f1928be5c145295849e", table " The statement has been terminated.]
GeneXus.Data.ADO.GxCommand.ExecuteNonQuery() +826
GeneXus.Data.ADO.GxCommand.execStm() +121

[GxADODataException: Type:System.Data.SqlClient.SqlException.DBMS Error Code:547.The DELETE statement conflicted with the REFERENCE constraint "ICITYTOURATTRACTION1". The conflict occur The statement has been terminated.]
GeneXus.Data.ADO.GxCommand.execStm() +615
GeneXus.Data.ADO.GxCommand.ExecuteNonQuery() +57
GeneXus.Data.NTier.ADO.UpdateCursor.execute() +172
GeneXus.Data.NTier.DataStoreProvider.execute(Int32 cursor, Object[] parms, Boolean batch) +1097
GeneXus.Data.NTier.DataStoreProvider.execute(Int32 cursor) +15
GeneXus.Programs.deleteattraction.executePrivate() +33
GeneXus.Programs.crud_attraction.E130B2() +65
```

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.

Pattern:

DeleteAttraction

For Each CityTourAttraction (Line: 1)

Order: [CityTourId](#), [AttractionId](#)
 Index: ICITYTOURATTRACTION
 Navigation filters: Start from: FirstRecord
 Loop while: NotEndOfTable
 Constraints: [AttractionName](#) = "Eiffel Tower"
 Join location: Server

```

  [table] = CityTourAttraction ( CityTourId, AttractionId ) INTO AttractionId
  [table] = Attraction ( AttractionId ) INTO AttractionName

```

DELETE FROM [CityTourAttraction](#)

For Each Attraction (Line: 5)

Order: [AttractionId](#)
 Index: IATTRACTION
 Navigation filters: Start from: FirstRecord
 Loop while: NotEndOfTable
 Constraints: [AttractionName](#) = "Eiffel Tower"
 Optimizations: Delete

```

  [table] = Attraction ( AttractionId ) INTO AttractionName

```

DELETE FROM [Attraction](#)

DeleteAttraction * X

Source * | Layout | Rules | Conditions | Variables | Help | Documentation

Subroutines

```

1 For each CityTour.Attraction
2   where AttractionName = "Eiffel Tower"
3   Delete
4 -endfor
5 For each Attraction
6   where AttractionName = "Eiffel Tower"
7   Delete
8 -endfor
9

```

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

```

DeleteAttraction * X
Source * Layout Rules Conditions Variables Help Documentation
Subroutines
1 For each CityTour.Attraction
2   where AttractionName = "Eiffel Tower"
3   Delete
4   endfor
5 For each Attraction
6   where AttractionName = "Eiffel Tower"
7   Delete
8   endfor
9

```

```

CRUD_Attraction * X
Web Layout Rules Events * Conditions Variables Help Documentation
'Delete attraction'
7   endif
8   msg(&text)
9   Endevent
10
11 Event 'Update attraction'
12   UpdateAttraction()
13   Endevent
14
15 Event 'Delete attraction'
16   For each CityTour.Attraction
17     where AttractionName = "Eiffel Tower"
18     Delete
19   endfor
20   For each Attraction
21     where AttractionName = "Eiffel Tower"
22     Delete
23   endfor
24   Endevent
25
26

```

Output

Show: General

```

error src0206: 'Delete' command is out of scope (Web Panel 'CRUD_Attr
error src0206: 'Delete' command is out of scope (Web Panel 'CRUD_Attr

```

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 excluimos 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

```

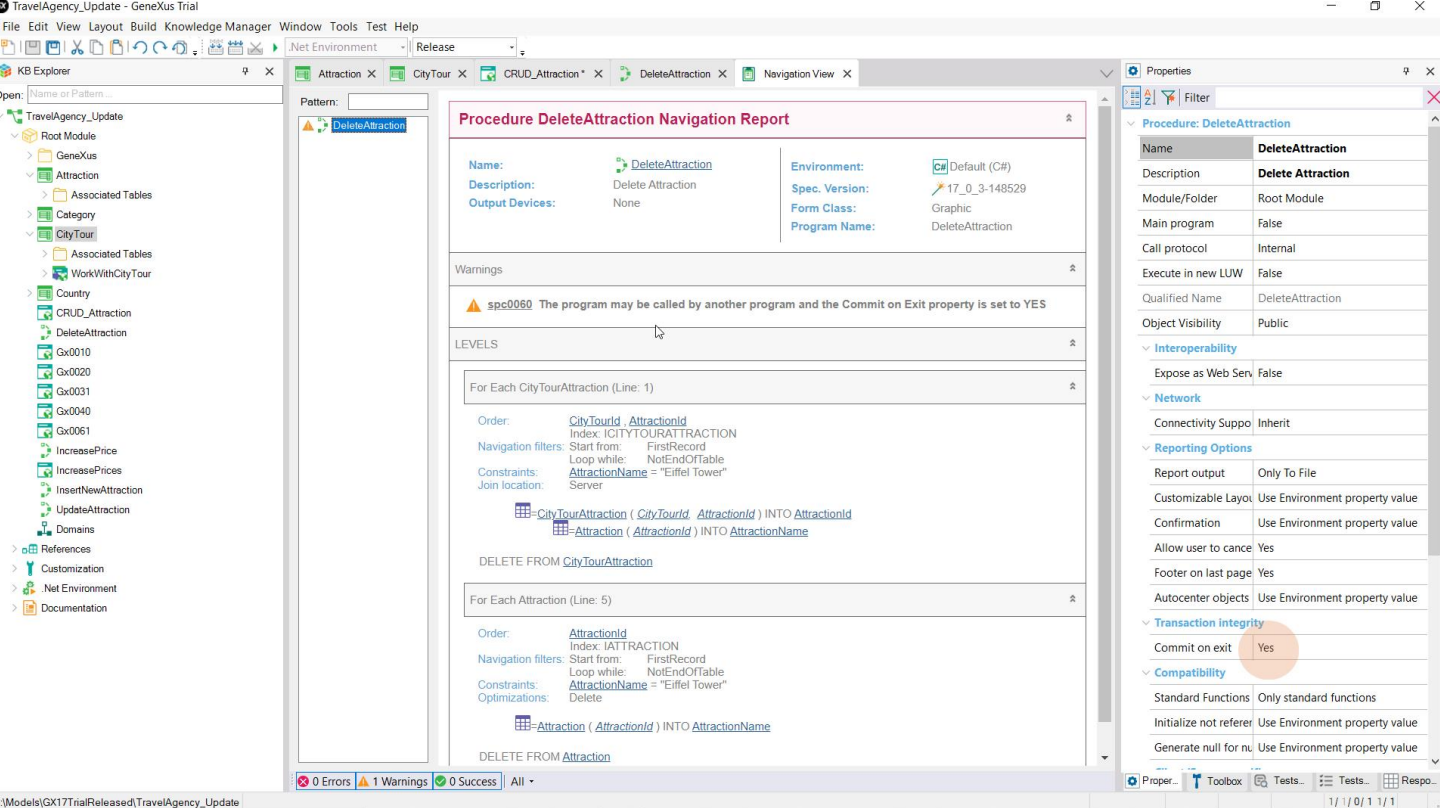
COMMIT?

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.



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 expression1 count expression2
order att11, att12, ... att1n [when condition]
order att21, att22, ... att2n [when condition | otherwise]
using DataSelector(parm1, ..., parmn)
unique att1, ..., attn
where condition [when condition]
where condition [when condition]
where att in DataSelector(parm1, ..., parmn)
blocking NumericExpression

```

...

Delete

...

When duplicate

...

When none

...

endfor

	Uniqueness check	Referential Integrity check
Delete	✗	✗

COMMIT

Transaction integrity

Commit on exit Yes

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 encontra 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 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

training.genexus.com/certifications