

Actualización a la Base de datos con comandos específicos de procedimientos.

Cómo eliminar (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

En el video sobre actualización con For each en un procedimiento vimos un caso en el que necesitábamos modificar el valor de la clave primaria de un registro, para lo cuál teníamos que crear uno nuevo con el nuevo valor de clave y eliminar el viejo.

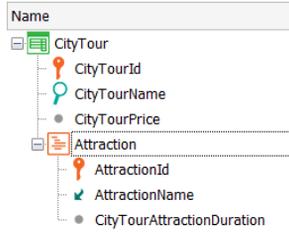
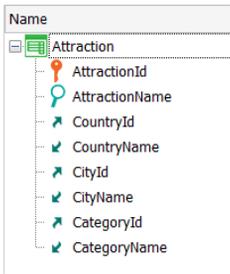
Y esto lo hacíamos posicionándonos sobre el registro en cuestión, creando uno nuevo con new; e inmediatamente ejecutando el comando Delete para eliminar el registro del for each en el que estábamos posicionados.

Así será en general la eliminación. Utilizando for each para elegir registro y ejecutando Delete para eliminarlo.

DELETE

Estudiemos los detalles de la eliminación.

Demo



General **Attraction**

UPDATE DELETE

Tour Id	1
Tour Name	Paris
Tour Price	220

City Tour Information ← CITY TOURS

Tour Name Paris

General Attraction

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

Recordemos las transacciones que veníamos utilizando para estudiar la inserción y actualización de la base de datos por procedimientos.

Aquí teníamos CityTour, donde podíamos especificar por qué atracciones turísticas iba a realizarse el tour actual. Por ejemplo, si vemos en ejecución, teníamos este tour por París, que recorrería las atracciones museo Louvre y Torre Eiffel.

The image shows a GeneXus application interface. On the left, a tree view under 'Name' lists the fields: AttractionId, AttractionName, CountryId, CountryName, CityId, CityName, CategoryId, and CategoryName. In the center, a 'Rules' 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, the 'Attraction' form is shown. 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'. The form also shows 'Country Id' (2), 'Country Name' (France), 'City Id' (1), 'City Name' (Paris), and 'Category Id' (2), with 'Category Name' (Monument). At the bottom, there are three buttons: 'CONFIRM' (red), 'CANCEL' (grey), and 'DELETE' (grey). An arrow points to the 'DELETE' button.

Por otro lado, teníamos la transacción que registra las atracciones. Y hemos agregado esta regla de error para no permitir la eliminación de una atracción con nombre ingresado.

Entonces, veamos que si intentamos eliminar la atracción Torre Eiffel a través de la transacción, no se nos permitirá.

Delete by Procedure?

Ahora, ¿qué pasa si intentamos eliminarla a través de un procedimiento?

Sabemos que esa atracción, Torre Eiffel, es parte de un city tour, y además tiene un nombre asignado, que es, justamente, Eiffel Tower. Entonces, ¿nos permitirá eliminarla?

	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

```

```

Attraction X
Structure | Web Layout | Rules | Events | Variables | Help | Documentation | Patterns
1 Error("Attraction with no empty name must not be deleted")
2 if not AttractionName.IsEmpty() and Delete;
3

```

Si pensamos en todo lo que vimos hasta el momento relativo a los comandos de actualización de la base de datos por procedimientos, podemos contestar que Sí, que nos permitirá eliminar la atracción, porque:

- **No** se realizan chequeos de integridad referencial programáticos, es decir, no chequeará que no exista ningún city tour que esté referenciando a la atracción a ser eliminada.
- Y además, **no** se ejecutan reglas de la transacción asociada a la tabla de la que se está eliminando un registro.

The screenshot displays the GeneXus IDE interface for developing a 'DeleteAttraction' program. On the left, a design view shows three buttons: 'New attraction', 'Update attraction', and 'Delete attraction'. The 'Delete attraction' button is connected to an event 'Delete attraction' which triggers the 'DeleteAttraction()' subroutine. The main workspace shows the 'DeleteAttraction' program details, including its name, description, and environment. A warning message 'spc0060 The program may be called by another program and the Commit on Exit property is set to YES' is displayed. The 'LEVELS' section shows a 'For Each Attraction' loop with the following code: 'Order: AttractionId', 'Index: IATTRACTION', 'Navigation filters: Start from: FirstRecord, Loop while: NotEndOfTable', 'Constraints: AttractionName = "Eiffel Tower"', 'Optimizations: Delete', and 'DELETE FROM Attraction'. The status bar at the bottom indicates '0 Errors', '1 Warnings', and '0 Success'.

Entonces si vamos a GeneXus y vemos que programamos este botón que invoca a este procedimiento... que recorre con un for each las atracciones filtrando por la de nombre "Eiffel Tower" y para los registros encontrados (en nuestro caso será uno solo), los elimina con el comando Delete...

```
Attraction x The DELETE statement conflicted with the REFERENCE constraint "ICITYTOURATTRACTION1". The conflict occurred in database "Id3f243fe13aa80f1928be5c145295849e/crud_attraction.aspx"
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
```

Si ejecutamos... se nos cae el programa. ¿Por qué?

Por lo mismo que ya 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

El Delete no chequea integridad referencial pero por defecto la base de datos sí, y no estamos capturando la excepción que arroja.

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

[CityTourAttraction](#) ([CityTourId](#), [AttractionId](#)) INTO [AttractionId](#)
[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

[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

```

Entonces, si queremos eliminar esa atracción, antes deberíamos eliminarla de todos los city tours en los que se encuentre. Así... Ejecutemos.

Como observación obvia, el comando Delete solo elimina el registro de la **tabla base** del For each en el que nos encontremos posicionados. No elimina registros de la tabla extendida. En este sentido funciona como el 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

Y otra cosa que ya hemos resaltado muchas veces, pero que es importante volver a destacar: el comando Delete solamente puede utilizarse dentro de un for each y en un procedimiento. No podríamos haber programado la eliminación directamente dentro del evento, por ejemplo.

A diferencia de lo que ocurre cuando eliminamos a través 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, en este procedimiento estamos eliminado dos registros.
¿Cuándo queda esta operación commiteada en la base de datos?

Sucede lo mismo que vimos con el new y la actualización con for each. Si la propiedad Commit on exit queda con su valor default, que es Yes, como GeneXus encuentra que se está queriendo realizar una eliminación sobre la base de datos, automáticamente agrega al final del Source del procedimiento un comando Commit.

TravelAgency_Update - GeneXus Trial

File Edit View Layout Build Knowledge Manager Window Tools Test Help

KB Explorer

TravelAgency_Update

- Root Module
- GeneXus
- Attraction
 - Associated Tables
 - Category
 - CityTour
 - Associated Tables
 - WorkWithCityTour
 - Country
 - CRUD_Attraction
 - DeleteAttraction
 - Gx0010
 - Gx0020
 - Gx0031
 - Gx0040
 - Gx0061
 - IncreasePrice
 - IncreasePrices
 - InsertNewAttraction
 - UpdateAttraction
 - Domains
 - References
 - Customization
 - .Net Environment
 - Documentation

Attraction x CityTour x CRUD_Attraction x DeleteAttraction x Navigation View x

Pattern: DeleteAttraction

Procedure DeleteAttraction Navigation Report

Name:	DeleteAttraction	Environment:	Default (C#)
Description:	Delete Attraction	Spec. Version:	17_0_3-148529
Output Devices:	None	Form Class:	Graphic
		Program Name:	DeleteAttraction

Warnings

▲ spc0060 The program may be called by another program and the Commit on Exit property is set to YES

LEVELS

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

CityTourAttraction (CityTourId , AttractionId) INTO AttractionId
 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

Attraction (AttractionId) INTO AttractionName

DELETE FROM Attraction

0 Errors 1 Warnings 0 Success All

Properties

Filter

Procedure: DeleteAttraction

Name	DeleteAttraction
Description	Delete Attraction
Module/Folder	Root Module
Main program	False
Call protocol	Internal
Execute in new LUW	False
Qualified Name	DeleteAttraction
Object Visibility	Public
Interoperability	
Expose as Web Serv	False
Network	
Connectivity Suppo	Inherit
Reporting Options	
Report output	Only To File
Customizable Layout	Use Environment property value
Confirmation	Use Environment property value
Allow user to cancel	Yes
Footer on last page	Yes
Autocenter objects	Use Environment property value
Transaction integrity	
Commit on exit	Yes
Compatibility	
Standard Functions	Only standard functions
Initialize not referer	Use Environment property value
Generate null for n	Use Environment property value

1/1/0/1/1/1

Y es por ello que el listado de navegación nos muestra una advertencia que nos lo indica, para que sepamos que aunque no hayamos especificado explícitamente un commit, GeneXus lo agregó.

	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

```

Por supuesto, podríamos programar nosotros un commit luego de cada for each.

O incluso, luego de cada Delete (porque en nuestro caso el for each recuperará un solo registro, pero podría tratarse de varios).

Summary

```

For each BaseTransaction
  skip expression1, count expression2
  order att1, att2, ... attn [when condition]
  order att2, 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

En suma, para eliminar registros específicamente por procedimiento contamos con el comando Delete que debe utilizarse dentro del comando For each.

La eliminación es del registro de la tabla base de for each en el que éste se encuentre posicionado en cada iteración.

En el caso de la eliminación no tiene sentido un control de unicidad de claves, porque no se está insertando ni actualizando nada. Y al igual que vimos con la inserción y la actualización, el Delete no realiza programáticamente control de integridad referencial alguno. Esto, otra vez, es por motivos de performance. Sin embargo las bases de datos en general sí lo realizan, a menos que apaguemos esa funcionalidad; por lo que, de no apagarla y fallar la integridad, arrojarán una excepción.

Por último: para que el registro quede commiteado en la base de datos, es decir, quede eliminado de forma permanente, debemos asegurarnos de que el comando Commit se ejecute. En un procedimiento, por defecto, se coloca un Commit implícito al final (siempre y cuando se entienda que en el Source se está accediendo en algún lado a la base de datos para actualizarla). Pero podemos escribir explícitamente Commits en el Source, donde nos convenga.

Otra vez, no lo veremos aquí, pero opcionalmente puede especificarse una cláusula Blocking, que lo que hace es permitir hacer eliminaciones en bloque, en lugar de registro a registro. Es decir, procesará los registros en

bloques de a N para reducir los accesos y mejorar la performance.

Tanto para la inserción, como para la actualización como para la eliminación **las redundancias** no son mantenidas automáticamente cuando se hacen por procedimiento. Es responsabilidad del desarrollador mantenerlas.

Sobre todo esto podemos ver más en nuestro wiki.

CRUD by procedure or by Business Components?

Nos queda por ver una breve comparación entre realizar las operaciones de Create/Update/Delete así, con estos comandos específicos de procedimientos, o hacerlo utilizando Business Components.