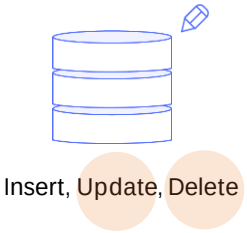
Actualización a la Base de datos con comandos específicos de procedimientos.

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

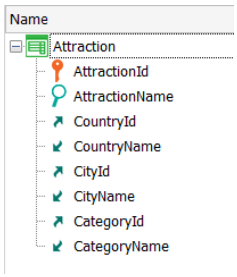
En el video sobre actualización con For each en un procedimiento vimos un caso en el que necesitábamos modificar el valor de la clave primaria de un registro, para lo cuál teníamos que crear uno nuevo con el nuevo valor de clave y eliminar el viejo.

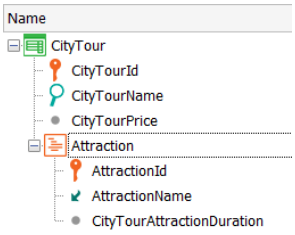
Y esto lo hacíamos posicionándonos sobre el registro en cuestión, creando uno nuevo con new; e inmediatamente ejecutando el comando Delete para eliminar el registro del for each en el que estábamos posicionados.

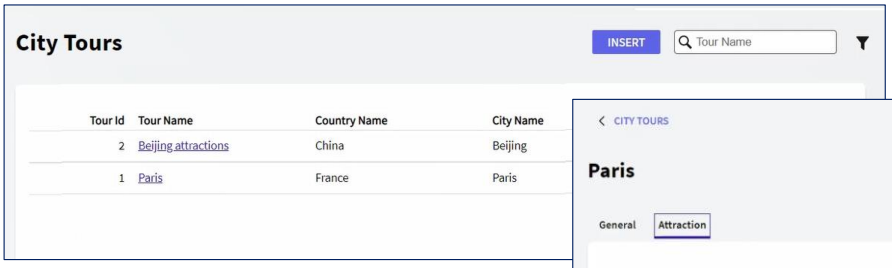
Así será en general la eliminación. Utilizando for each para elegir registro y ejecutando Delete para eliminarlo.

Delete

Estudiemos los detalles de la eliminación.

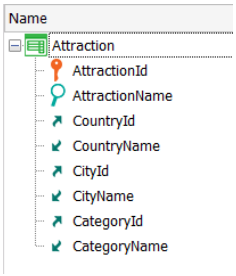


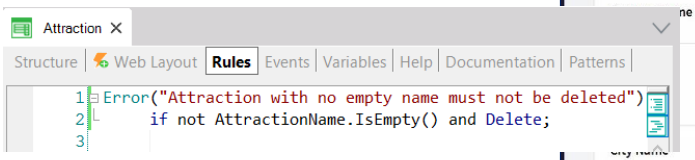


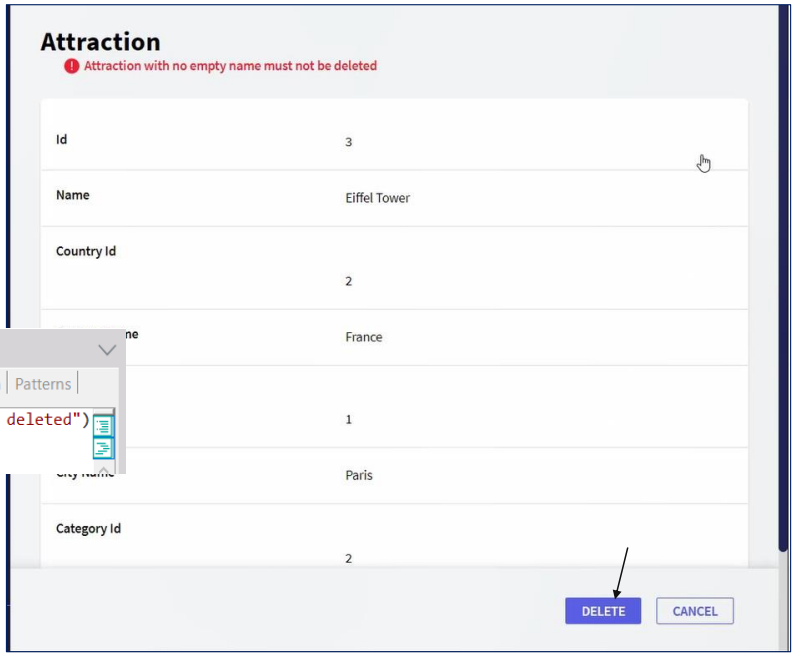


Recordemos las transacciones que veníamos utilizando para estudiar la inserción y actualización de la base de datos por procedimientos.

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.







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.

¿Eliminación por Procedimiento?

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	✕	✕	✕

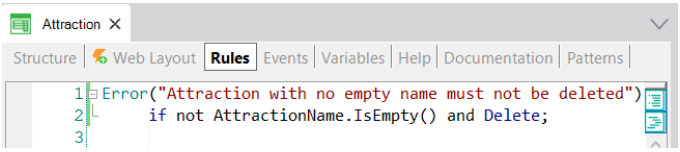
CityTourId	AttractionId	CityTourAttractionDuration
1	1	200
1	3	120
2	2	240
2	4	240

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"

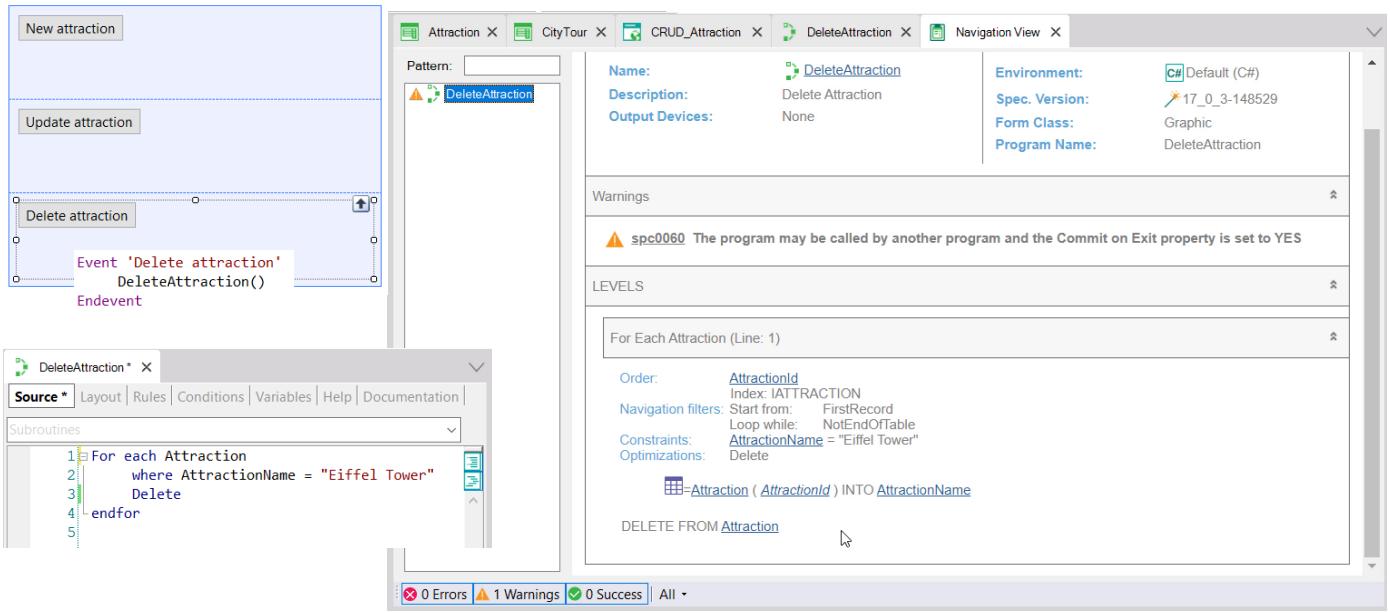
Delete

endfor

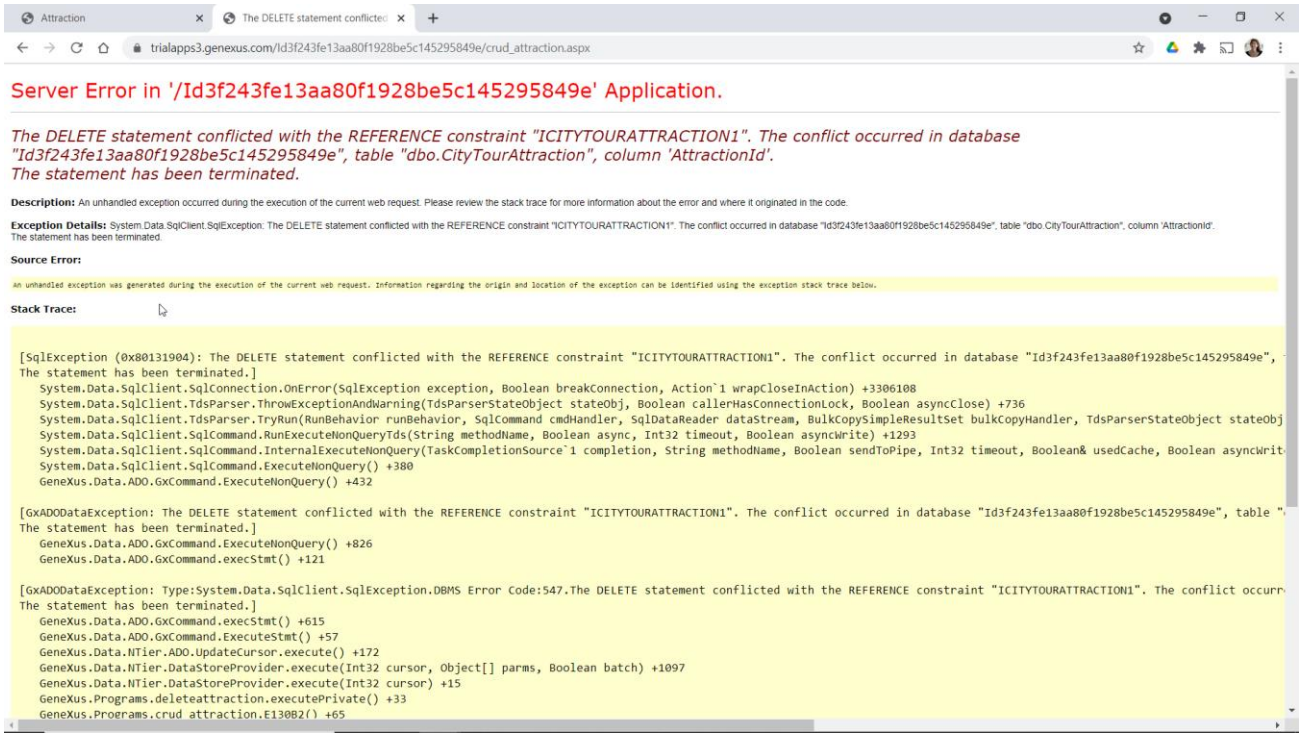


Si pensamos en todo lo que vimos hasta el momento relativo a los comandos de actualización de la base de datos por procedimientos, podemos contestar que SÍ, que nos permitirá eliminar la atracción, porque:

- **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.



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...



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

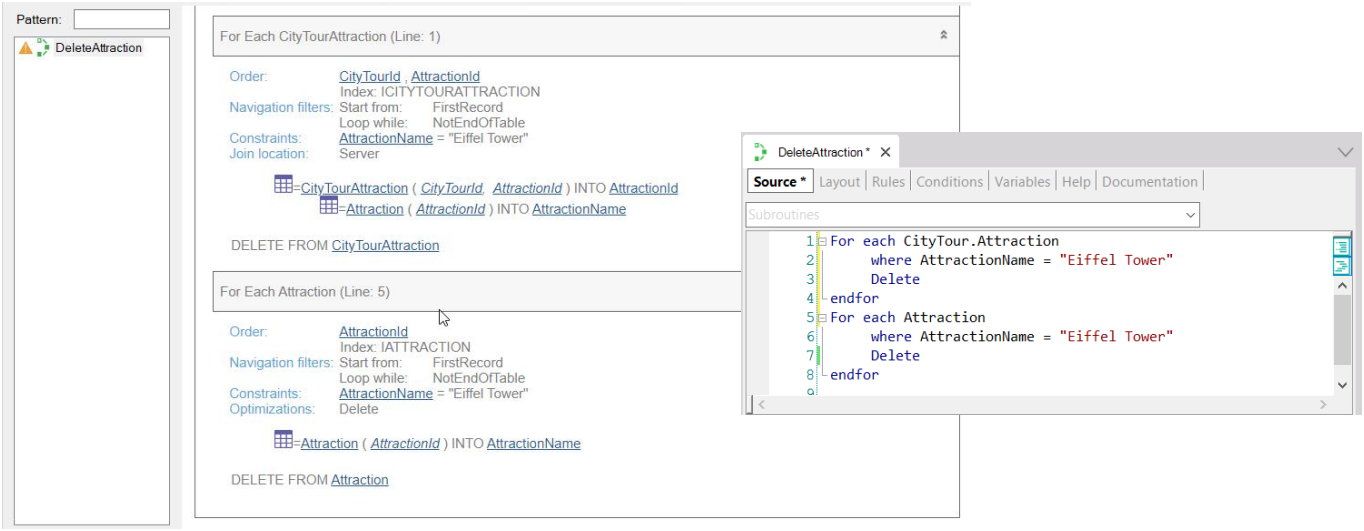
```
For each Attraction
    Where AttractionName = "Eiffel Tower"

    Delete

endfor
```

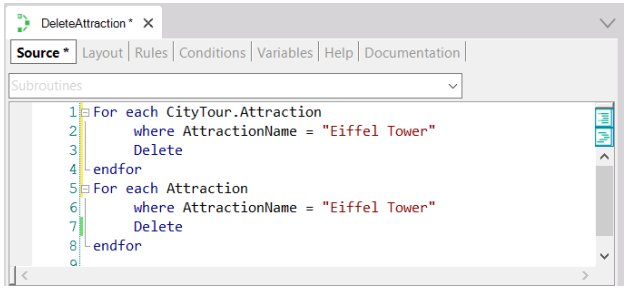
exception

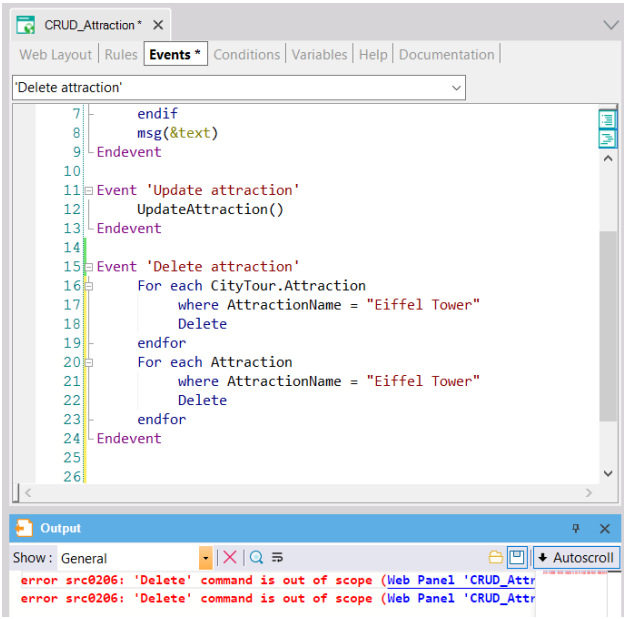
El Delete no chequea integridad referencial pero por defecto la base de datos sí, y no estamos capturando la excepción que arroja.



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.





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

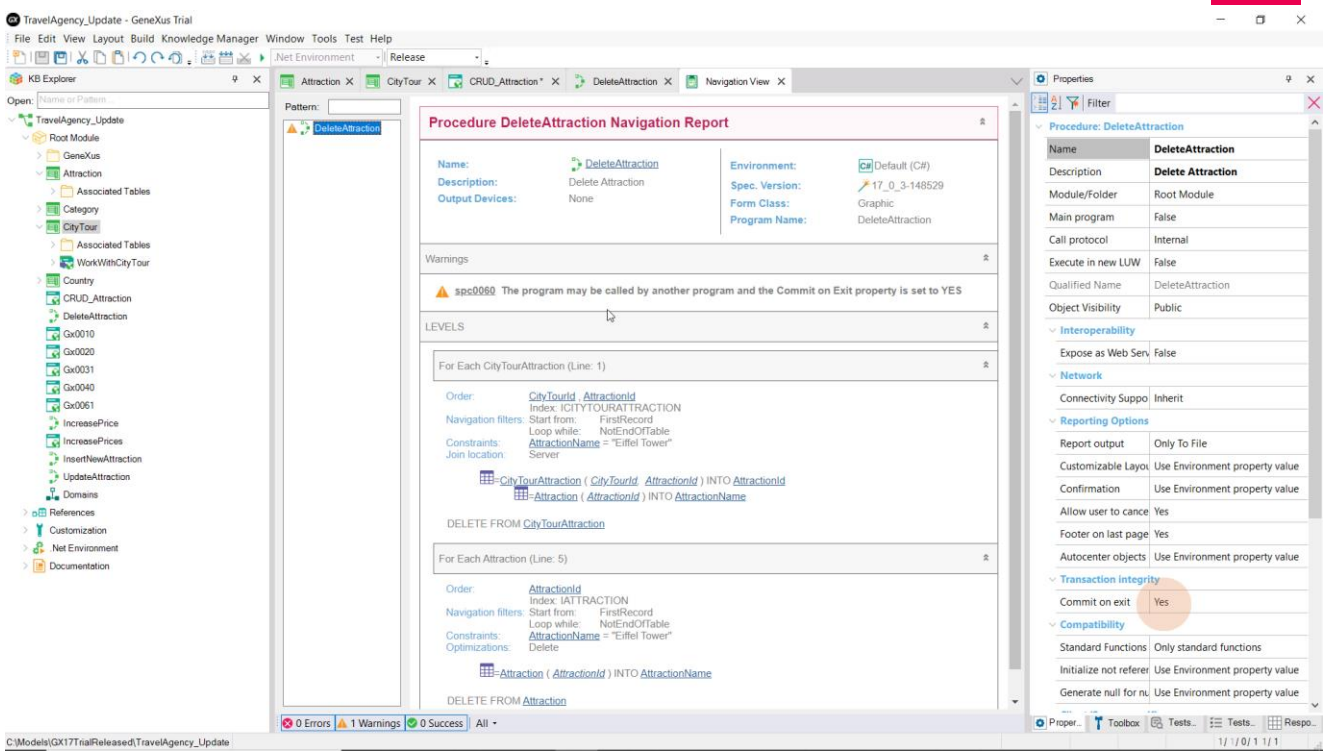
AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2

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

COMMIT?

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.



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 agregará.

	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 $expression_1$ **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

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.

GeneXus™

training.genexus.com
wiki.genexus.com