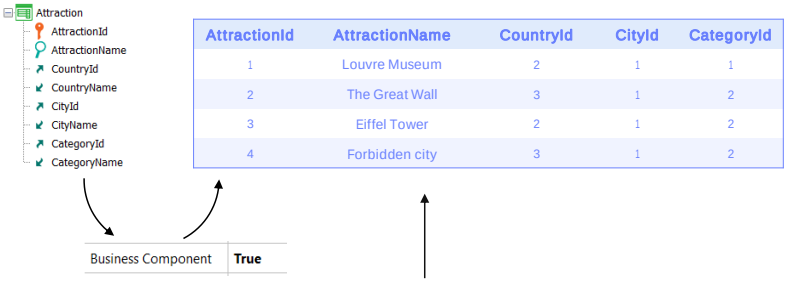
Actualización a la Base de datos con comandos específicos de procedimientos.

Cómo insertar

GeneXus

1. **Business Compontent**: Insert(), Update(), Delete()

Insert, Update, Delete



2. **Procedure**: New, For each, Delete

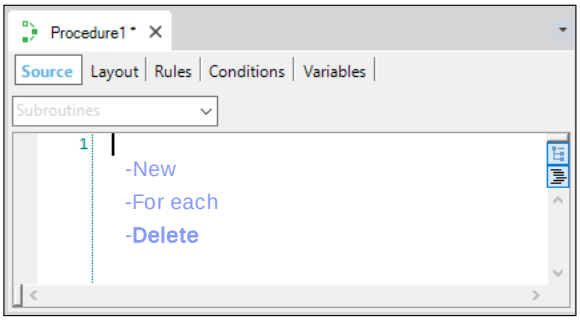
Para actualizar la información de la base de datos por código teníamos dos posibilidades:

Hacerlo utilizando el business component de la transacción, a través de sus métodos Insert, Update o Delete (el Save, y el InsertOrUpdate), o hacerlo exclusivamente dentro de un procedimiento, a través de los comandos New, For each, y Delete.

En otros videos estudiamos detalladamente el primer caso. Ahora profundizaremos en el segundo.

PROCEDURE ONLY

Insert, Update, Delete



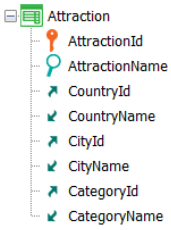
Es importantísimo tener en mente que esta segunda forma de actualización solo puede realizarse en el Source de un procedimiento.
Los comandos que estudiaremos no tendrán validez en ningún otro lado, como eventos de Panels o Web Panels, sino solamente aquí, en procedimientos.

Insert

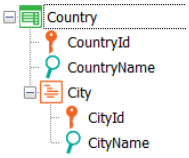
Empecemos por el comando que permite insertar un registro en una tabla.

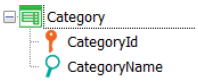
New Command

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	Forbidden city	3	1	2
5	Christ the Redemmer	1	2	2



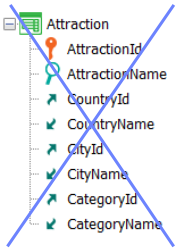


Es literalmente UN registro y en UNA tabla.

Supongamos que tenemos la transacción Attraction, que registra atracciones turísticas (donde también existirá una transacción Country que registra la información de cada país y sus ciudades y una Category, que registra las categorías en las que se clasifican las atracciones turísticas), y que queremos insertar a través de un procedimiento al Cristo Redentor en la tabla asociada a Attraction.

New Command

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	Forbidden city	3	1	2
5	Christ the Redemmer	1	2	2

New

```
AttractionId   = 5
AttractionName  = "Christ the Redemmer"
CountryId = find( CountryId, CountryName = "Brazil" )
CityId  = find( CityId, CityName = "Rio de Janeiro" )
CategoryId  = find( CategoryId, CategoryName = "Monument" )
```

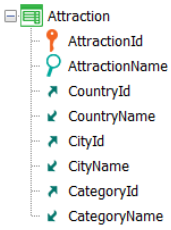
endnew

Para ello pedimos ejecutar en el Source del procedimiento el comando New, asignándole valor a cada uno de los atributos de la tabla, para el registro que queremos insertar. Aquí estaremos trabajando directamente con la tabla, desligados por completo de la transacción que la crea. Es decir, no importan las reglas de la transacción, ni los eventos, ni nada. El comando new es completamente indiferente a la transacción. Lo único que atiende es la composición de la tabla de la base de datos en la que intentará insertar un registro.

Entonces aquí los atributos inferidos o fórmulas de la transacción no participan en absoluto. Para el new no existen. Los únicos que importan son los atributos de la tabla. Por eso, en este caso, le hemos dado valor a todos ellos.

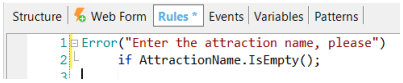
New Command

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	Forbidden city	3	1	2
5		1	2	2

Not assigned atributes?
Secondary attribute



New

AttractionId = 5
AttractionName = "Christ the Redemmer"
CountryId = find(CountryId, CountryName = "Brazil")
CityId = find(CityId, CityName = "Rio de Janeiro")
CategoryId = find(CategoryId, CategoryName = "Monument")

endnew

¿Esto es obligatorio? Claro que no. Si no asignamos valor a algún atributo, se considerará vacío.
Así, si no asignáramos valor para AttractionName, se insertará la atracción 5 sin nombre. Así exista en la transacción una regla de error que lo impida. Es que, como dijimos, la transacción aquí no interviene para nada más que para dar existencia a la tabla en la base de datos. Esta es una primera gran diferencia con la inserción a través de Business Component.

New Command

Not assigned atributes?

Primary key attribute

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

Uniqueness check

New

```
AttractionId     = 5
AttractionName  = "Christ the Redemmer"
CountryId = find( CountryId, CountryName = "Brazil" )
CityId  = find( CityId, CityName = "Rio de Janeiro" )
CategoryId  = find( CategoryId, CategoryName = "Monument" )
```

endnew

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

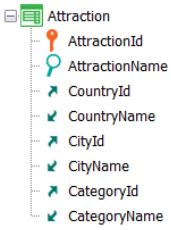
Pero, ¿y qué pasa si lo que no se asigna es la clave primaria?

Se intentará insertar un registro de clave vacía. Como si hubiéramos asignado explícitamente el 0.
Si la clave primaria no es autonumerada, entonces si no existe en la tabla ya un registro con ese id cero lo inserta.
¿Y si existe? No hará nada. No veremos ninguna diferencia entre haber ejecutado el New y no haberlo hecho.

Es decir, el comando New controla la unicidad de registros. No va a insertar registro de clave duplicada.

New Command

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	Forbidden city	3	1	2
5	Matisse Museum	2	2	1

New

Primary key attribute

Uniqueness check

```
AttractionId     = 5
AttractionName  = "Christ the Redemmer"
CountryId = find( CountryId, CountryName = "Brazil" )
CityId  = find( CityId, CityName = "Rio de Janeiro" )
CategoryId  = find( CategoryId, CategoryName = "Monument" )
```

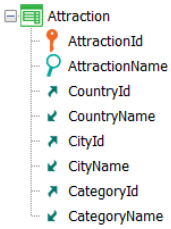
endnew

Lo mismo que ocurriría si ya existía la atracción con id 5 en la tabla cuando vamos a ejecutar el new. Aquí no hará absolutamente nada, debido al control de unicidad.

Y esto vale tanto para clave primaria como para claves candidatas.

New Command

Insert, Update, Delete



Unique index

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

Candidate key attribute

New

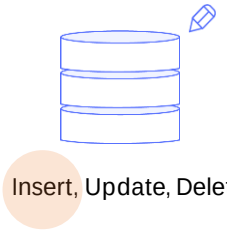
AttractionId = 5
AttractionName = "Eiffel Tower"
CountryId = find(CountryId, CountryName = "Brazil")
CityId = find(CityId, CityName = "Rio de Janeiro")
CategoryId = find(CategoryId, CategoryName = "Monument")

Uniqueness check

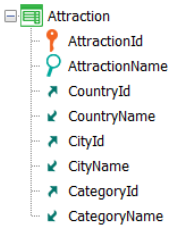
endnew

Imaginemos, por ejemplo, que AttractionName es clave candidata, es decir, tenemos un índice unique definido sobre este atributo.
Y supongamos que el registro de Id 3 tenía como valor de AttractionName el mismo que estamos ahora queriendo insertar como registro 5.
Al ejecutarse el new, no hará nada. Puesto que encontrará que ya existe un registro con el AttractionName idéntico al del registro que estamos queriendo insertar.

New Command



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	Forbidden city	3	1	2
5	Christ the Redemmer	1	2	2

Not assigned atributes?

Primary key attribute

autonumbered

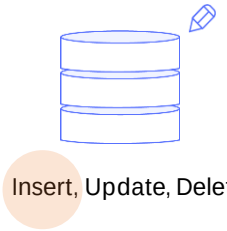
New

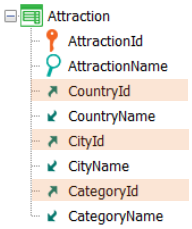
AttractionName = "Christ the Redemmer"
CountryId = find(CountryId, CountryName = "Brazil")
CityId = find(CityId, CityName = "Rio de Janeiro")
CategoryId = find(CategoryId, CategoryName = "Monument

&AttractionId = AttractionId

Ahora bien, volviendo a lo que nos preguntábamos antes de qué pasa cuando dejamos sin especificar valor para la clave primaria... si ésta se autonumera, entonces nunca fallará por clave primaria duplicada: siempre insertará un registro con el siguiente número. ¿Cómo sabemos qué número se le asignó? Queda en memoria inmediatamente después del new el valor en el atributo. Así, podemos, por ejemplo, asignárselo a una variable para no perderlo en el siguiente uso del atributo.

New Command





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

Referential integrity checks?

NO

Not assigned atributes?

Foreign key attribute

New

AttractionId = 5
AttractionName = "Christ the Redemmer"
CountryId = find(CountryId, CountryName = "Brazil")
CityId = find(CityId, CityName = "Rio de Janeiro")
CategoryId = find(CategoryId, CategoryName = "Monument")

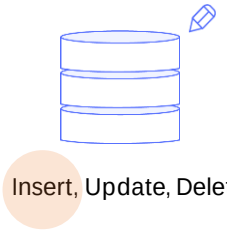
endnew

¿Y qué pasa con los atributos que son claves foráneas? Como CountryId, CityId, CategoryId.

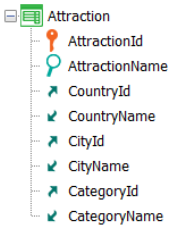
¿El comando new realiza control de integridad referencial?

La respuesta es no. No lo realiza. Es que la actualización por comando fue creada para tener una forma de actualización rápida, con la mejor performance posible. Realizar esos chequeos siempre enlentece la operación. Cuando se trata de un solo registro esto no importa, pero pensemos qué pasa si tenemos que insertar millones de registros.

New Command



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	Forbidden city	3	1	2

New

Referential integrity checks?

NO

Not assigned atributes?

Foreign key attribute

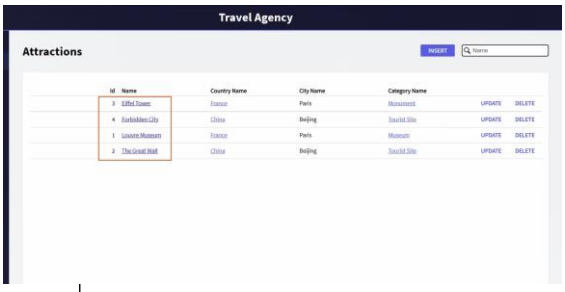
```
AttractionId     = 5
AttractionName  = "Christ the Redemmer"
CountryId = 15
CityId  = find( CityId, CityName = “Rio de Janeiro” )
CategoryId  = find( CategoryId, CategoryName = "Monument“ )
```

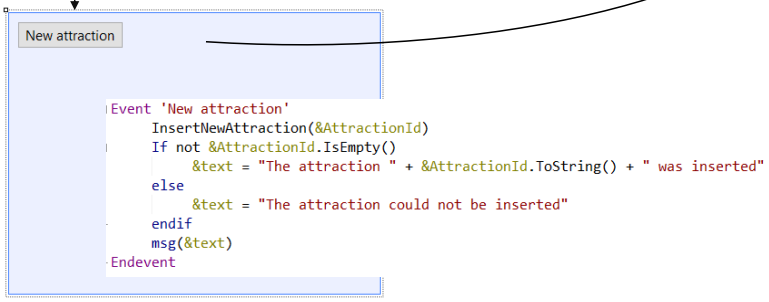
endnew

Así, por ejemplo, si para el nuevo registro queremos asignar un país de id 15, el comando new no controlará que exista en la tabla que almacena los países uno con ese valor. Insertará el registro sin problema. Y la base de datos nos quedará en estado inconsistente.

Vamos a GeneXus a probarlo.

New Command

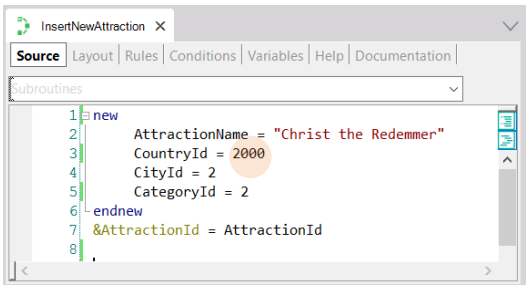


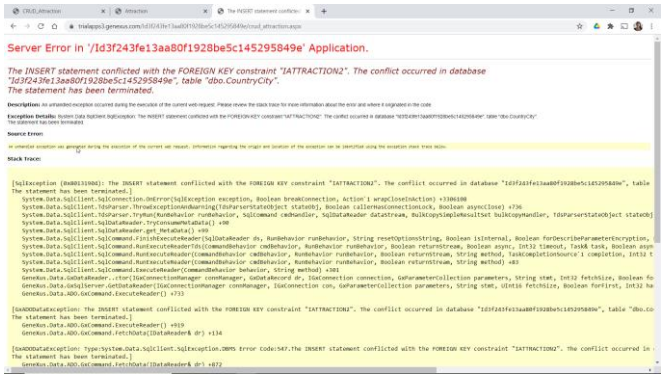


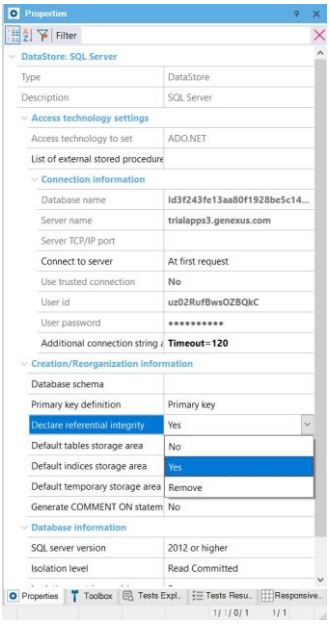
Tenemos las tres transacciones con datos. En particular, Attraction tiene estos cuatro registros.
AttractionId es autonumerado.

Hemos creado este web panel donde el usuario presionará el botón “New attraction”, que llamará a un procedimiento que intentará insertar un nuevo registro en la tabla Attraction para el Cristo Redentor.
El procedimiento devolverá el AttractionId que la base de datos le asignó al registro insertado. Con esto armamos el mensaje que el usuario verá en el panel.
Probemos.
The attraction 5 was inserted. Vamos a ver... y aquí vemos, efectivamente, la atracción 5 insertada en la tabla.

New Command





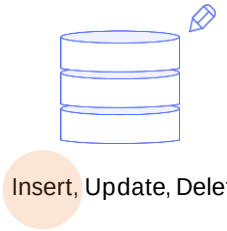


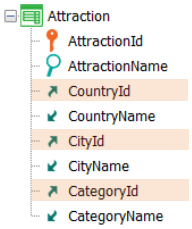
Ahora, observemos qué pasa si para el nuevo registro queremos asignar un país inexistente. Por ejemplo este. Ejecutemos.
Se nos cayó el programa, cuando lo que esperábamos era que hubiera insertado el registro sin problema, dado que dijimos que el new no controla la integridad referencial. ¿Qué pasó entonces?

Es verdad que el programa no está controlando la integridad referencial, pero la base de datos sí lo está haciendo. Entonces el new intentó realizar la inserción, pero la base de datos no lo dejó, y arrojó una excepción.

Por defecto la base de datos controla la integridad referencial. Podemos apagar este control, mediante una propiedad.
Pero lo que vemos claramente es que el new no lo está haciendo. Por eso tenemos que tener mucho cuidado al utilizar este comando, porque una cancelación de programa como esta es inaceptable para el usuario final.

New Command





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

Referential integrity checks?

NO

Not assigned atributes?

Foreign key attribute

New

AttractionId = 5
~~Attraction~~Name = "Christ the Redemmer"
CountryId = 15
CityId = find(CityId, CityName = “Rio de Janeiro”)
CategoryId = find(CategoryId, CategoryName = "Monument“)

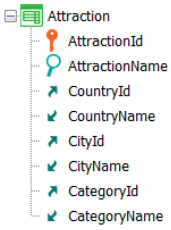
endnew

¿Y qué pasa si dejamos sin asignar una clave foránea?

Otra vez, el new no realizará ningún chequeo e intentará insertar el registro. Si la base de datos sí los realiza, entonces arrojará una excepción como la anterior que detendrá el programa si no es capturada y manejada por éste.
Podríamos pensar que si el atributo acepta nulos, entonces, en ese caso, no fallará nunca, porque la base de datos permitirá ese nulo para la clave foránea. Sin embargo hay que tener cuidado sobre cómo interpreta la base de datos que se trata de un nulo y no de un valor vacío. Sobre eso hablaremos en otro video.

New Command

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	Forbidden city	3	1	2
5	Christ the Redemmer	1	2	2

COMMIT?

New

```
AttractionId     = 5
AttractionName  = "Christ the Redemmer"
CountryId = find( CountryId, CountryName = "Brazil" )
CityId  = find( CityId, CityName = "Rio de Janeiro" )
CategoryId  = find( CategoryId, CategoryName = "Monument"
```

endnew
....
Commit

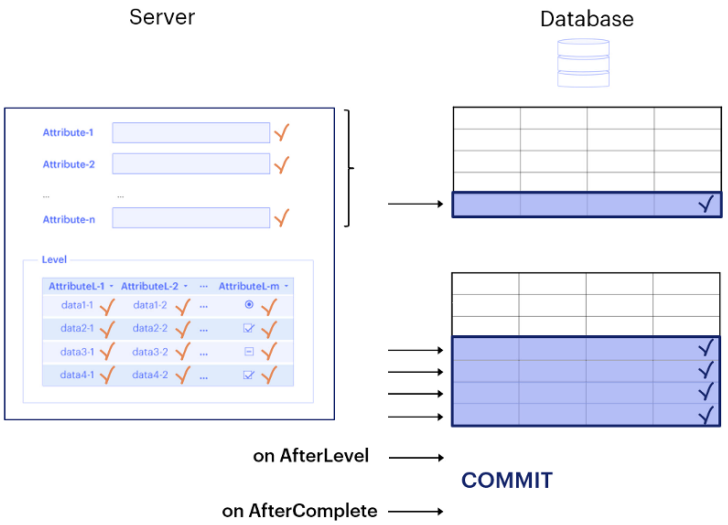
Si todo está bien, entonces el new inserta un registro en la tabla. ¿Y qué pasa con el Commit?

Si vamos a ver la navegación del procedimiento que ejecutamos hace un rato, nos está mostrando una advertencia que dice que el programa podría ser llamado por otro programa y que la propiedad Commit on Exit está seteada en Yes. Aquí la vemos.
Esta propiedad también se encuentra en el otro objeto que opera sobre la base de datos: el objeto transacción. Aquí la vemos, bajo el grupo Transaction integrity (en otra clase estudiaremos este tema importante, que es el de cómo definir y conseguir la integridad transaccional).

Lo importante es que si esta propiedad está en Yes, eso significa que se estará agregando un Commit automático en el código fuente del objeto (siempre y cuando el objeto realice alguna operación sobre la base de datos).

New Command

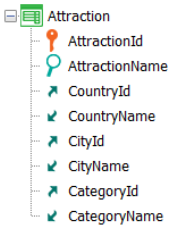
Insert, Update, Delete



En la transacción, al final de la operación sobre el cabezal y sus líneas.

New Command

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	Forbidden city	3	1	2
5	Christ the Redemmer	1	2	2

COMMIT?

New

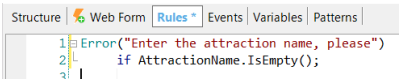
```
AttractionId     = 5
AttractionName  = "Christ the Redemmer"
CountryId = find( CountryId, CountryName = "Brazil" )
CityId  = find( CityId, CityName = "Rio de Janeiro" )
CategoryId  = find( CategoryId, CategoryName = "Monument"
```

endnew
....
Commit

En un objeto procedimiento, al final del Source. Es por ello que no tuvimos que especificarlo. Si la propiedad estuviera apagada, ahí sí tendríamos que escribir explícitamente el comando Commit, tal como lo hacíamos con los Business Components.

	Uniqueness check	Referential Integrity check	Rules/Events execution
New command	✓	✗	✗

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
5	Christ the Redemmer	1	2	2



Category Table

Resumiendo lo que hemos visto hasta aquí:

Al intentar insertar un registro en una tabla utilizando el comando new en un procedimiento, el comando realiza control de unicidad por clave primaria y claves candidatas, asegurando, así, que no se agregue un registro a la tabla que duplique la clave. Si encuentra clave duplicada entonces no hace nada.

Si algún atributo del registro a ser insertado es clave foránea, el comando new no realizará chequeo de integridad referencial. Es decir, no irá a buscar a la tabla que es referenciada la existencia de un registro con el valor que estamos queriendo usar, para solo así insertar el registro. PERO si la base de datos tiene declarada la integridad referencial, entonces ésta sí realizará el chequeo y cancelará el programa si se está violando la integridad. Si no tiene la integridad referencial declarada, entonces sí se permitirá la inserción del registro independientemente de que viole la integridad.

Y respecto a las reglas y eventos que se encuentran en la transacción asociada a la tabla, éstos, claramente, no tienen aquí nada que hacer. Recordemos que para el new solo importa la tabla, no de dónde viene.

	Uniqueness check
New command	✓

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

New

```
AttractionId   = 3
AttractionName = "Eiffel Tower"
CountryId = 2
CityId  = 1
CategoryId  = 3

endnew
```

New

```
AttractionId   = 3
AttractionName = "Eiffel Tower"
CountryId = 2
CityId  = 1
CategoryId  = 3

When duplicate
   for each Attraction
      CategoryId =
   3
   endfor
```

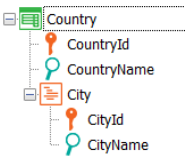
Ahora bien, podríamos querer realizar alguna acción si el registro se encuentra duplicado por alguna clave. Por ejemplo, en vez de insertarlo, actualizarlo. En el ejemplo, podemos querer en ese caso cambiar el valor de CategoryId.

Para ello, contamos con la cláusula when duplicate. El código que encierra solo se ejecutará cuando se encuentre duplicada la clave primaria o alguna clave candidata. Si lo que queremos en ese caso es actualizar el registro, entonces tenemos que hacerlo dentro de un for each [C], como vemos aquí. No se asignan directamente valores a los atributos que se desean modificar, como podría pensarse, sino que esto hay que hacerlo escribiendo un for each, sin tener que filtar por AttractionId, porque ya se sobreentiende. Es la manera en que el new entiende que queremos actualizar esos atributos del registro que encontró duplicado. En este caso actualizar únicamente el atributo CategoryId. Aquí sí podrían actualizarse atributos de la tabla extendida.

New Command

Not assigned atributes → Depend on context are they instantiated?

```
For each Country.City
    where CountryName = "France"

    print cityInfo

    new

        AttractionName  = CityName + " attra
        CategoryId      = 2

    endnew

endfor
```



CountryId	CityId	CityName
1	1	Sao Paulo
1	2	Rio de Janeiro
2	1	Paris
2	2	Nice
3	1	Beijing

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Mattisse Museum	2	2	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2
5	Paris attraction	2	1	2
6	Nice attraction	2	2	2

Volviendo al asunto de qué sucede con los atributos de la tabla del new a los que no se les asigna valor. Habíamos dicho que quedan vacíos, pero en verdad esto depende del contexto. Quedan vacíos si en el contexto en el que se encuentra el new no están instanciados, es decir, no tienen valor.

Pensemos por ejemplo el caso en el que estamos recorriendo con un for each las ciudades de Francia, y para cada una: la imprimimos, por ejemplo, y además, inmediatamente, ejecutamos el comando new que estamos viendo.

Primera cosa: ¿cómo determina GeneXus cuál es la tabla base del new? Atendiendo exclusivamente a los atributos a los que se les está asignando valor. Tiene que encontrar una tabla de la base de datos que los contenga. Claramente será Attraction.

¿Este CityName participa? No, no lo hace. Si está allí es porque en el contexto en el que hemos escrito el new, sabemos que está instanciado. Ese CityName será el del For each. En definitiva estamos, para cada ciudad, queriendo insertar una atracción turística nueva, con el nombre de la ciudad y la categoría 2.

Bien, pero, ¿y qué identificador de atracción, país y ciudad se les va a asignar al registro que se inserte? En este caso el único vacío será el AttractionId, que no está instanciado en el

contexto, porque no está en la tabla extendida del for each. Y tampoco es recibido por parámetro en el atributo, que es la otra forma de instanciación que conocemos.
Así que si AttractionId se autonumera, entonces cuando se inserte el registro la base de datos le dará el siguiente al último número.
Sin embargo, CountryId y CityId sí están instanciados en el contexto. Por lo que se les asignará ese valor del contexto. En este caso, el país y ciudad en el que estamos posicionados en el for each.

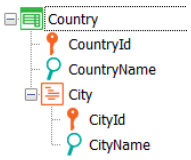
Y luego lo mismo ocurrirá para la siguiente ciudad del for each...

New Command

Not assigned atributes → Depend on context are they instantiated?

```
For each Country.City
    where CountryName = "France"

    print cityInfo

    new
        AttractionName =
        CategoryId     = 2

    endnew

endfor
```

WARNING
Base table: CATEGORY!!!!



CountryId	CityId	CityName
1	1	Sao Paulo
1	2	Rio de Janeiro
2	1	Paris
2	2	Nice
3	1	Beijing

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Mattisse Museum	2	2	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2
5		2	1	2
6		2	2	2

¿Qué pasaría si quisiéramos que se inserten los mismos dos registros en Attraction, pero con AttractionName vacío?

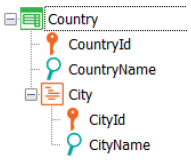
Parecería que simplemente con no asignarle valor a AttractionName estaríamos implementándolo. Sin embargo, si observamos los atributos que va a utilizar GeneXus para determinar su tabla base, la del new, está únicamente CategoryId. Por tanto no elegirá como tabla base Attraction, sino Category.

¿Entonces, cómo hacemos para que elija la tabla Attraction? Una posibilidad es asignar explícitamente el valor vacío para AttractionName. Porque de este modo, al estar nombrando a ese atributo, va a participar en la determinación de la tabla base y ya va a hacer que la tabla base en lugar de Category, sea Attraction.

New Command

Not assigned atributes → Depend on context are they instantiated?

```
For each Country.City
    where CountryName = "France"

    print cityInfo

    new Defined by AttractionName

        CategoryId        = 2

    endnew

endfor
```



CountryId	CityId	CityName
1	1	Sao Paulo
1	2	Rio de Janeiro
2	1	Paris
2	2	Nice
3	1	Beijing

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Mattisse Museum	2	2	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden city	3	1	2
5		2	1	2
6		2	2	2

La otra opción es utilizar la cláusula Defined by que permite agregar más atributos para ser contemplados junto con los demás en la determinación de la tabla base.

Summary

new
Defined by *Attribute*$_1$, *Attribute*$_2$, ..., *Attribute*$_N$
Blocking *NumericExpression*

Attribute$_1$ = *expression*$_1$
Attribute$_2$ = *expression*$_2$
...
Attribute$_N$ = *expression*$_N$

When duplicate
...
for each *BaseTransaction*
Attribute$_I$ = *expression*$_I$
Attribute$_K$ = *expression*$_K$
...
endfor
...

	Uniqueness check	Referential Integrity check
New command	✓	✗

COMMIT

Transaction integrity

Commit on exit	Yes

En definitiva, el comando new se utiliza para insertar un registro en una tabla. La tabla se determina atendiendo a los atributos que aparecen asignados. Si se agrega cláusula Defined By, entonces los atributos que allí aparezcan listados también participan. Aquí el concepto de tabla extendida no tiene ningún sentido.

Por otro lado, habíamos visto que el único control programático que realiza el new es el control de unicidad. Y que si se encontraba un registro con la clave que estamos intentando insertar, entonces el new no hacía nada.... Salvo que hubiéramos programado la cláusula when duplicate.

Y habíamos dicho que allí, entre otros comandos, podemos incluir un for each para actualizar atributos del registro que se encontró duplicado. ¿Todos los atributos pueden ser actualizados? Por ejemplo, la clave primaria ¿puede actualizarse allí? La respuesta es no. Pero sí podrán actualizarse atributos de la tabla extendida.

Por último: para que el registro quede commiteado en la base de datos debemos asegurarnos de que el comando Commit se ejecute. En un procedimiento, por defecto, se coloca un Commit implícito al final. Pero podemos escribir explícitamente Commits en el Source, donde nos convenga.

No lo veremos aquí, pero opcionalmente puede especificarse una cláusula Blocking, que lo que hace es permitir hacer inserciones en la

base de datos en bloque, en lugar de registro a registro, siempre que el new esté dentro de una estructura repetitiva. Esto, claramente, será por motivos de eficiencia, cuando las inserciones batch sean muy cuantiosas.

GeneXus™

training.genexus.com
wiki.genexus.com