

Actualización directa de base de datos con comandos For each, Delete y New

GeneXus 16 course

GeneXus® 16

Actualización a la Base de Datos

Usando Transacción

Attraction

SELECT

Id: 9

Name: Forbidden City

Country Id: 3

Country Name: China

Category Id: 2

Category Name: Monument

Photo:

CONFIRM CANCEL DELETE

Usando Business Component

Business Component: True

Attraction X

Structure Web Form Rules Events Variables Patterns

Name	Type	Description	For...	Nullable
Attraction	Attraction	Attraction		
AttractionId	Id	Attraction Id	No	
AttractionName	Name	Attraction Name	No	
CountryId	Id	Country Id	No	
CountryName	Name	Country Name		
CityId	Id	City Id	Yes	
CityName	Name	City Name		
CategoryId	Id	Category Id	Yes	
CategoryName	Name	Category Name		
AttractionPhoto	Image	Attraction Photo	No	

Event 'New Attraction'

```

&Attraction.AttractionName = "Forbidden City"
&Attraction.CountryId = find( CountryId, CountryName = "China")
&Attraction.CityId = find( CityId, CityName = "Beijing")
&Attraction.CategoryId = find( CategoryId, CategoryName = "Monument")
&Attraction.Save()
Commit
Endevent
  
```

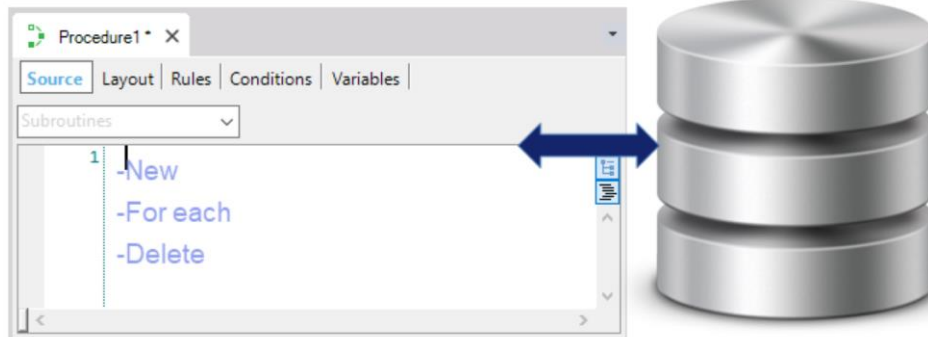
Hasta el momento, para actualizar los datos de la base de datos, hemos empleado las transacciones en sus 2 formas de uso:

- Ejecutando su pantalla e ingresando datos en forma interactiva
- Y ejecutadas como Business Component, a través de una variable, sin usar la pantalla

Por ejemplo, si quisiéramos agregar una nueva atracción, en un web panel podríamos haber colocado un botón y haber escrito para su evento asociado lo que vemos arriba.

Otra forma de actualizar la Base de Datos

Objeto procedure



Conoceremos ahora una alternativa más para realizar inserciones, modificaciones y eliminaciones en la base de datos.


Debemos saber que lo que veremos **solamente puede ser usado en objetos de tipo procedimiento**, a diferencia de la alternativa que vimos usando Business Component, que podía usarse desde cualquier objeto.

Solo válido en procedimientos

✓ INSERCIÓN

New Atributos de tabla

Endnew



Attraction id	AttractionName	Countryid	Cityid	Categoryid	AttractionPhoto
1	Louvre Museum	2	1	1	
2	Eiffel Tower	2	1	2	
3	The Great Wall	3	1	2	
4	Forbidden City	3	1	2	

Transaction integrity	
Commit on exit	Yes



Utilizando este comando podemos asignarle valores a los atributos de una *tabla física* para insertar *un registro*.

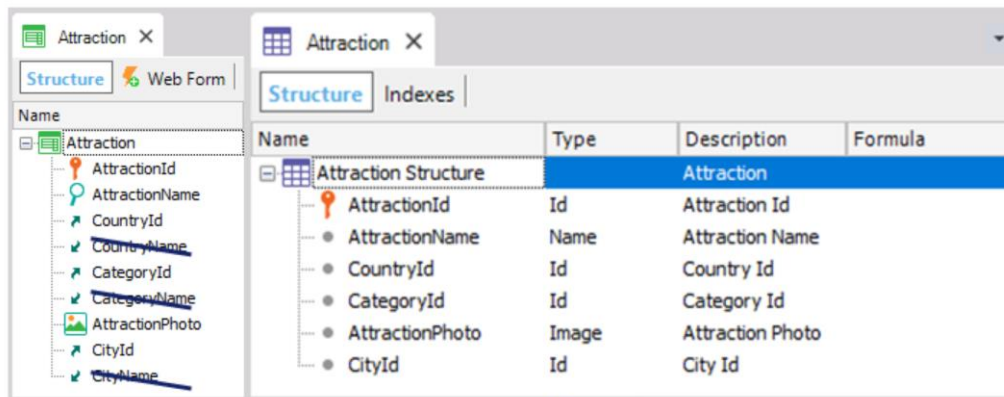
En los procedimientos contamos con un comando llamado New para insertar registros en una tabla.

Utilizando este comando podemos asignarle valores a los atributos de una **tabla física**.

Hablamos de una tabla y no de una transacción porque no todos los atributos que están en la estructura de una transacción están presentes en la tabla física.

Por ejemplo, si queremos insertar una atracción, en la transacción Attraction hay muchos atributos declarados en la estructura que no están físicamente en la tabla ATTRACTION sino que están **en la tabla extendida de la tabla ATTRACTION**. Los hemos incluido en la estructura para mostrarlos en el form o bien para usarlos en las reglas.

Atributos de la tabla Attraction y New



Name	Type	Description	Formula
Attraction Structure			
AttractionId	Id	Attraction Id	
AttractionName	Name	Attraction Name	
CountryId	Id	Country Id	
CategoryId	Id	Category Id	
AttractionPhoto	Image	Attraction Photo	
CityId	Id	City Id	

Comando NEW solo permite asignar
valor a estos atributos

Si hacemos View/Tables y editamos la tabla ATTRACTION, vemos estrictamente los atributos pertenecientes a la tabla ATTRACTION.

A estos atributos los podemos incluir dentro de un comando New, y asignarle valores. Es que el comando new insertará un registro y solo uno en esa tabla.

En los procedimientos...

✓ INSERCIÓN



GeneXus determinará cuál es la tabla física donde insertará el registro, analizando los atributos que están a la izquierda del signo de igual.

Si pertenecen todos a una misma tabla física, se realizará la inserción del registro en dicha tabla y si no, se nos informará que no es posible determinar la tabla en la cual realizar la inserción.

La tabla que encuentra GeneXus se llama **tabla base del New**. En este caso es: ATTRACTION.

Observemos que no estamos asignando una imagen al atributo *AttractionPhoto*. En este caso la inserción se realizará igual, solo que el atributo *AttractionPhoto* quedará sin asignar y la atracción será creada, pero sin foto.

También podríamos haber dejado sin asignar los atributos *CountryId*, *CityId* y *CategoryId*, que son claves foráneas. O incluso, en lugar de buscar con la fórmula *find* los ids correspondientes, podríamos escribir ids inexistentes, porque el comando **NEW** **no controla la integridad referencial**. Por este motivo hay que tener mucho cuidado al programar este comando.

Inserción New versus BC

Usando cláusula NEW en Procedimientos:

```

new
  AttractionName = "Forbidden City"
  CountryId = find(CountryId, CountryName = "China")
  CityId = find(CityId, CityName = "Beijing")
  CategoryId = find(CategoryId, CategoryName="Monument")
endnew

```

Tabla base: ATTRACTION

Transaction integrity

Commit on exit Yes

Usando Business Components:

```

Event 'New Attraction'
  &Attraction.AttractionName = "Forbidden City"
  &Attraction.CountryId = find( CountryId, CountryName = "China")
  &Attraction.CityId = find( CityId, CityName = "Beijing")
  &Attraction.CategoryId = find( CategoryId, CategoryName = "Monument")
  &Attraction.Save()
  Commit
Endevent

```

¿Control de duplicados?
 ¿Controles de IR?
 ¿Reglas?

En el momento de ejecutar la inserción, lo único que GeneXus verificará es que no se inserte un registro con una llave primaria ya existente, o, si hubiera un índice unique definido sobre un atributo, que no se ingrese un valor duplicado para este atributo. Si esto ocurriera, el new no hará nada, y no indicará nada al usuario tampoco. Luego veremos cómo detectar esta situación y tomar acciones al respecto. El **control de unicidad** es el único control que realiza y **NO** se realizan **controles de integridad referencial**, por lo que si asignáramos un CountryId inexistente, no fallará la integridad referencial, a diferencia de lo que ocurre cuando la inserción se realiza a través de un Business Component. Recordemos que en ese caso no solo se realiza el control de unicidad y de integridad referencial, sino que además se disparan las reglas definidas en la transacción.

En el caso de la inserción usando Business Component, como puede realizarse desde cualquier objeto, debemos especificar explícitamente el **Commit**, es decir, la orden a la base de datos de que deje persistentes los datos modificados desde el anterior Commit.

Los procedimientos cuentan con la propiedad **Commit on exit** que por defecto está en Yes. ¿Qué efecto tiene esto? Que si el procedimiento realiza algún acceso a la base de datos, entonces al final del código agregará un comando Commit sin que el desarrollador deba explicitarlo. De hecho no queda escrito en el objeto, pero se coloca en el programa generado. Es por ello que no necesitamos colocar explícitamente un Commit luego del comando new, a menos que el desarrollador desee commitear inmediatamente.

GeneXus reconoce que debe realizar un acceso a la base de datos dentro de un procedimiento cuando se utiliza new, for each para actualizar, delete (operaciones que veremos luego), o el método Load() de BC, así como el Delete luego de un Load. En esos casos, coloca el Commit implícito, en caso contrario, no entiende que hay acceso a la base de datos y por lo tanto no lo agrega.

Es por eso que cuando se hace:

```

&BC.elemento1 = ...
&BC.elemento2 = ...
&BC.Save()

```

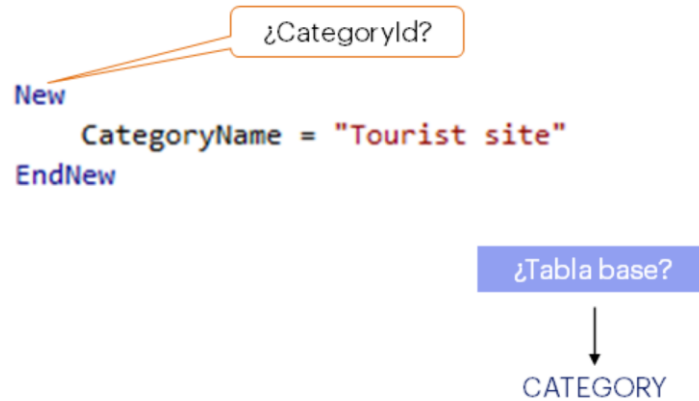
no agrega el Commit, debido a que no se da cuenta de que estamos queriendo hacer un Insert y trata al

Business Component como si fuera un SDT cualquiera. En este caso hay que escribir el Commit explícitamente.

Si dentro del código del procedimiento hubiera además un new, for each que actualiza, o incluso un Load() o Delete, en cualquiera de esos casos no habría que escribir explícitamente el commit, porque ya estableció una conexión a la BD. Si solo tenemos en nuestro procedimiento el código anterior, allí sí tendremos que agregar el comando Commit en forma explícita.

Solo válido en procedimientos

✓ INSERCIÓN



En una cláusula `New`, para crear un registro nuevo asignamos valores a los atributos que pertenecen a un registro de una tabla.

Si tuviéramos en un procedimiento escrito el `new` que vemos arriba, donde solamente asignamos valor al atributo `CategoryId`, evidentemente la tabla base será `CATEGORY`, porque es la tabla donde está almacenado físicamente el atributo `CategoryId`.

Otra vez, no asignamos valor al atributo `CategoryId` pues es autonumerado.

Solo válido en procedimientos

✓ ACTUALIZACIÓN

```
For each Attraction
  Where CityName = 'Beijing'
  Where CategoryName = 'Monument'
  CategoryId = find(CategoryId, CategoryName = 'Tourist site')
Endfor
```

Tabla base:
ATTRACTION

Los atributos de la tabla extendida de
ATTRACTION pueden ser actualizados
en el For each*



*salvo Clave Primaria

Vamos a ver ahora cómo podemos actualizar un valor existente en la base de datos.

Para cambiar un valor almacenado en un atributo --de un registro de una tabla-- por otro valor, nos posicionamos mediante un For each en ese registro y mediante una asignación damos el nuevo valor.

En el ejemplo, la tabla base del for each es ATTRACTION, dado que la transacción base es Attraction y estamos recorriendo todas las atracciones de Beijing que tienen categoría "Monument". A esas les estamos cambiando la categoría por "Tourist Site".

Notemos que en este ejemplo estamos actualizando muchos registros, todos los que cumplan con las condiciones de los where. Y estamos actualizando el valor de un único atributo, pero podrían ser varios. Incluso podrían actualizarse atributos de la **tabla extendida**, por ejemplo, el nombre de la categoría de la atracción, el nombre del país de la atracción, el nombre de la ciudad de la atracción. Lo veremos en la página siguiente.

Observar que en el new solamente se puede asignar valores a atributos físicamente presentes en la tabla, dado que allí se está queriendo insertar un nuevo registro y uno solo. En el for each, en cambio, estamos posicionados en cada momento en un registro existente y desde allí llegamos a editar todos registros asociados por tabla extendida.

Si tuviéramos una llave candidata (esto es, un índice unique definido sobre alguno de los atributos) y dentro del for each estuviéramos modificando su valor por uno ya existente para otro registro, la actualización **no** se realizará. Así como dijimos para el caso del new, aquí tampoco tendremos ningún aviso. Luego veremos cómo capturar este caso y hacer algo al respecto.

Como restricción, **no** es posible modificar dentro de un for each la llave primaria de la tabla que estamos recorriendo.

Solo válido en procedimientos

ACTUALIZACIÓN

La tabla **CUSTOMER** está incluida en la tabla extendida de **INVOICE**

Name	Type	Description	Formula	Nullable
Invoice	Invoice	Invoice		
InvoiceId	Id	Invoice Id		
InvoiceDate	Date	Invoice Date		
CustomerId	Numeric(4,0)	Customer Id		
CustomerName	Character(20)	Customer Name		
CustomerTotalPurchases	Price	Customer Total Purchases		
InvoiceTotalAmount	Price	Invoice Total Amount		
Flight	Flight	Flight		
FlightId	Id	Flight Id		
FlightAvailableSeats	Numeric(4,0)	Flight Available Seats		No
InvoiceFlightSeatQty	Numeric(4,0)	Invoice Flight Seat Qty		No
FlightFinalPrice	Price	Flight Final Price		No
InvoiceFlightAmount	Price	Invoice Flight Amount	FlightPrice*(1-Airline...	No

Customer	Customer
CustomerId	Id
CustomerName	Name
CustomerAddress	Address
CustomerPhone	Phone
CustomerEmail	Email
CustomerAddedDate	Date
CustomerVIP	Boolean

```

For each Invoice
Where InvoiceDate >= &LastDate
if count(InvoiceFlightSeatQty) >= 5
    CustomerVIP = True
endif
Endfor
  
```

La agencia de viajes quiere detectar aquellos clientes que en una misma factura a partir de una fecha dada tengan contratados más de cinco vuelos, marcando a esos clientes como clientes VIP. Para lograrlo agregamos el atributo CustomerVIP a la transacción Customer, atributo que estará almacenado en la tabla asociada.

Para detectar y marcar a estos clientes se deberá recorrer la tabla de facturas (INVOICE), contar los vuelos de cada una, y si esa cantidad supera los cinco vuelos, debemos actualizar el valor del atributo CustomerVIP, poniéndolo en True.

Aquí la tabla base del for each es INVOICE, por lo que la fórmula count contará únicamente los vuelos pertenecientes a esa factura. En caso de ser mayores a 5, observemos que estamos actualizando el valor de un atributo de la tabla CUSTOMER, que está en la tabla extendida de la tabla que estamos recorriendo.

Actualización For each versus BC

Usando un For Each en un Procedimiento:

```
For each Attraction
  where CityName = "Beijing" and CategoryName = "Monument"
  CategoryId = find( CategoryId, CategoryName = "Torist Site")
endfor
```

Transaction integrity

Commit on exit Yes

Usando Business Components:

```
For each Attraction
  Where CityName = "Beijing" and CategoryName = "Monument"
  &Attraction.Load(AttractionId)
  &Attraction.CategoryId = find( CategoryId, CategoryName = "Tourist site")
  &Attraction.Save()
Endfor
Commit
```

Control unicidad de
PK y CK

Control IR

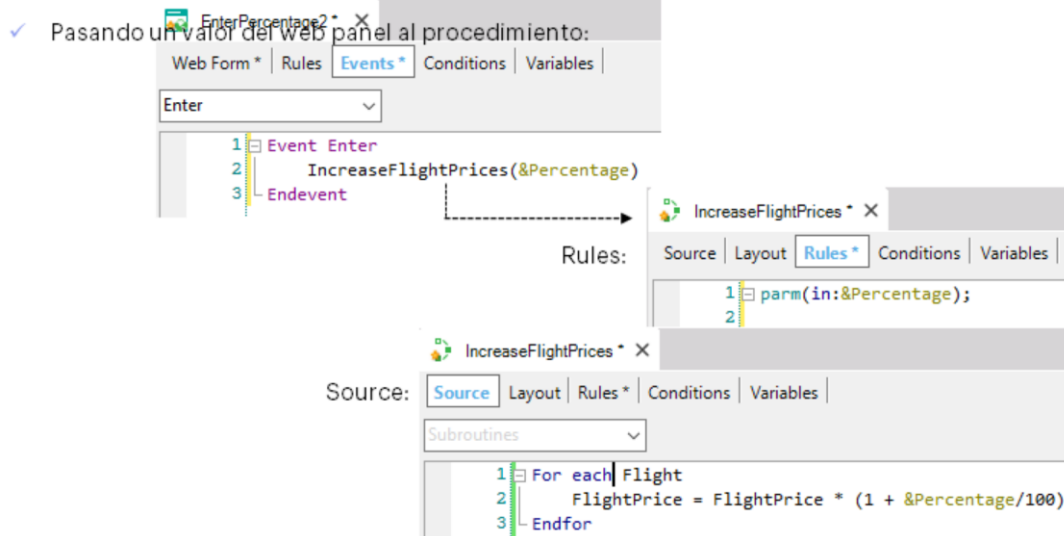
Reglas

Como dijimos, utilizando el for each dentro del procedimiento el único control que GeneXus realizará es el de unicidad de clave primaria y claves candidatas.

En cambio, la solución con Business Component controlará la integridad referencial, por lo que si le asignáramos a CategoryId un valor inexistente, no permitirá realizar la actualización. Además ejecutará las reglas de negocio.

Recordar lo dicho en new para **Commit on exit**.

Procedimiento para aumento de precios



Queremos pedirle al usuario que digite un porcentaje y recalculer el precio de todos los vuelos aplicándole ese porcentaje de aumento.

Para ello creamos un web panel para pedir este dato al usuario, y desde allí invocamos a un procedimiento que será quien realizará efectivamente la actualización de la tabla de vuelos, para modificar el valor del precio de acuerdo al porcentaje recibido por parámetro.

Para realizar esa actualización, entonces, utilizamos el comando For each, recorriendo la tabla FLIGHT y asignándole al atributo FlightPrice el valor que tenía por la expresión que vemos.

Comparando alternativas

Usando Business Components

```

Event Enter
  For each Flight
    $BCFlight.Load(FlightId)
    $BCFlight.FlightPrice = $BCFlight.FlightPrice * (1+ &Percentage/100)
    $BCFlight.Save()
    If $BCFlight.Success()
      Commit
    Else
      Rollback
    Endif
  Endfor
Endevent
  
```

Usando un Procedimiento

```

Subroutines
  1 For each Flight
  2   FlightPrice = FlightPrice * (1 + &Percentage/100)
  3 Endfor
  
```

Comparemos las dos alternativas que utilizamos para actualizar la base de datos.

En la primera alternativa que implementamos, en el evento enter del web panel EnterPercentage, escribimos el For each y actualizamos la base de datos empleando la transacción Flight como Business Component.

En la segunda solución, el evento enter del webpanel EnterPercentage2 solamente contiene una llamada a un procedimiento, quien recibe el valor del porcentaje, navega con For each todos los vuelos y a cada uno de ellos, le calcula y asigna el nuevo precio.

¿Cuáles son las diferencias, ventajas y desventajas entre resolverlo de una manera u otra?

Ejemplo Insert Category + cambio en Attractions

InsertCategoryUpdateAttractions

```

1 New
2   CategoryName = 'Tourist Site'
3 Endnew
4 For each Attraction
5   Where CityName = 'Beijing' and CategoryName = 'Monument'
6   CategoryId = find(CategoryId, CategoryName = 'Tourist Site')
7 Endfor

```

Exclusivamente en Procedure

CategoriesAndAttractions

Web Form Rules **Events** Conditions Variables

```

1 Event 'Do'
2   &category.CategoryName = "Tourist site"
3   &category.Save()
4   For each Attraction
5     Where CityName = "Beijing" and CategoryName = "Monument"
6     &Attraction.Load(AttractionId)
7     &attraction.CategoryId = &category.CategoryId
8     &attraction.Save()
9   Endfor
10  Commit
11 Endevent

```

Puede estar en cualquier objeto

Aquí vemos las dos soluciones para insertar una nueva categoría y modificar el valor de la categoría de ciertas atracciones por esta categoría.

Nótese que el código que está en el evento del web panel ejecuta en cualquier objeto GeneXus, mientras que el código que está en el procedimiento solo es válido dentro de un objeto procedimiento.

Analicemos el procedimiento:

Como CategoryId es autonumerado, el new nunca va a fallar.

Estamos utilizando una fórmula find para encontrar el Id de la categoría de nombre "Tourist Site" que acabamos de insertar con el new. Si nos hubiéramos equivocado al digitar el nombre de la categoría, entonces el find no encontrará registro y devolverá el valor empty, que para un numérico es 0. En este caso, estaríamos cambiándole el id de categoría a las atracciones de Beijing que eran monumentos, por el valor 0. Si el atributo CategoryId no admitiera nulos, entonces nos quedaría la base de datos en estado inconsistente. Recordemos que la actualización con for each no controla integridad referencial, como sí lo hace la actualización a través de business component. En nuestro caso CategoryId admite nulos, por lo que no tendremos problema de consistencia utilizando esta opción por procedimiento.

La actualización por business component es siempre más segura, puesto que controlará todo lo que controlan el new y el for each, y más: la integridad referencial, pero además las reglas definidas en la transacción asociada. Piense, además, que aunque usted haya tenido todas las precauciones al programar el procedimiento, siempre puede modificarse en algún momento posterior la realidad, y por ejemplo agregar reglas de error a la transacción, que usted no contempló al programar el procedimiento. La solución con Business Component le asegura que esas reglas se incluirán en los chequeos en todo momento.

Solo válido en procedimientos

ELIMINACIÓN:

```
For each Category
  where CategoryName = "Tourist Site"
  Delete
Endfor
```

```
For each Attraction
  Delete
Endfor
```

```
For each Category
  Delete
Endfor
```

Ya hemos visto cómo los procedimientos nos permiten insertar y actualizar registros en la base de datos. Vamos a ver ahora cómo eliminar registros.

Para eliminar registros contamos con el comando Delete que se usa dentro de un comando For each.

Básicamente como el comando Delete elimina el registro de la tabla en la que se está posicionado, se utiliza el comando for each para posicionarse en ese registro.

En el primer ejemplo vemos que estamos navegando la tabla Category, filtrando por la categoría "Tourist Site" y por lo tanto eliminando puntualmente un registro con el comando Delete.

Pero también podríamos haber eliminado **todas** las atracciones, y **todas** las categorías (si deseamos vaciar esas tablas).

Si eliminamos primero las categorías y luego las atracciones, como la base de datos sí realiza por defecto un control de integridad referencial, al querer eliminar la primera de las categorías, si tiene alguna atracción relacionada, fallará y el programa cancelará, como veremos a continuación.

Usando comandos especiales (NEW, FOR EACH, DELETE)

- ✓ Validaciones de datos: solo unicidad, no integridad referencial.
- ✓ Sólo válidos en procedimientos

Usando business components:

- ✓ Validaciones de datos: tanto unicidad como integridad referencial.
- ✓ Se disparan las reglas de las transacciones.
- ✓ Válidos en cualquier objeto.

Cuando empleamos business components se valida la consistencia de los datos que se intentan actualizar en la base de datos y se ejecutan las reglas definidas en la transacción que se está ejecutando como business component.

Las reglas que generan mensajes como msg y error, también se disparan y los mensajes correspondientes quedan guardados en una colección que se puede recorrer e imprimir.

En un procedimiento, ninguna de estas cosas se realiza.

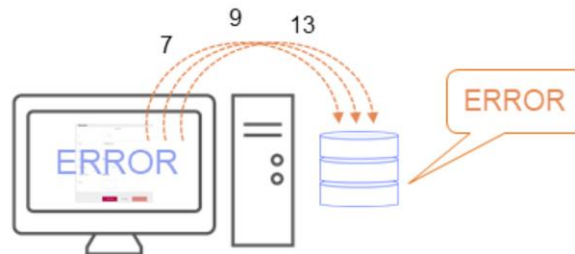
Algo que hemos visto y es oportuno repasar, es que si bien el comando For each puede ser usado en un web panel, no es posible utilizarlo para modificar la base de datos directamente asignándole valores a los atributos, y que esto solamente puede llevarse a cabo desde un objeto procedimiento. Tampoco es posible codificar un comando New en un web panel, ni incluir un comando delete dentro del For each. **Esto solo es válido en procedimientos.**

Sí en cambio es posible, en cualquier objeto, modificar la base de datos con business components.

Algo importante que hay que saber es que los procedimientos **no controlan que los datos que asignamos sean consistentes**. Únicamente controlan la **unicidad** de las claves primarias y candidatas.

Comandos New, For each, Delete (en procedimientos)





Es nuestra responsabilidad al usar estos comandos de actualización directa en procedimientos, asignar o insertar datos que sean consistentes con el resto de la información almacenada.

En el ejemplo que vimos con New, al CategoryName le podemos asignar cualquier valor, ya que no es un dato relacionado con otras tablas.

En el ejemplo de la actualización de la categoría de las atracciones, en cambio, estamos asignando un nuevo valor a un atributo que es llave foránea: CategoryId. Si el valor asignado no existe como identificador de categoría en la tabla CATEGORY, **el procedimiento no lo validaría, por lo que podríamos estar ingresando datos inconsistentes.**

Como las bases de datos controlan la consistencia de los datos interrelacionados, cuando el usuario ejecuta la aplicación y se intenta asignar un valor no consistente, la base de datos rechazará la operación y la grabación inconsistente no se llevará a cabo.

Sin embargo el programa cancelará su ejecución y esto no es nada amigable para el usuario.

Por lo tanto, si usamos procedimientos para actualizar la base de datos, **será nuestra responsabilidad asignar datos válidos y bien relacionados.**

Usando comandos especiales (NEW, FOR EACH, DELETE)

- ✓ Si respecto a validaciones de datos solo se valida la unicidad, ¿dónde programar acciones a ejecutarse frente a intentos de duplicación de llave primaria o de llave candidata?

```
new
  AttractionName = "Forbidden City"
  CountryId = find( CountryId, CountryName = "China")
  CityId = find( CityId, CityName = "Beijing")
  CategoryId = find( CategoryId, CategoryName = "Tourist Site")
when duplicate
endnew
```

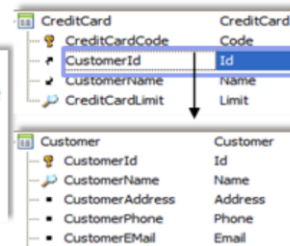
Si AttractionName es llave candidata, es decir, no puede repetirse, y ejecutamos un new donde estamos queriendo insertar un nuevo registro (de id autonumerado) asignándole a AttractionName el valor "Forbidden City" y ya existe un registro con ese valor, tenemos la posibilidad de hacer algo, programándolo dentro de la cláusula when duplicate del new.

Usando comandos especiales (NEW, FOR EACH, DELETE)

- ✓ Si respecto a validaciones de datos solo se valida la unicidad, ¿dónde programar acciones a ejecutarse frente a intentos de duplicación de llave primaria o de llave candidata?

Clave candidata

```
for each CreditCard
  where CreditCardCode = &creditCardCode
  CustomerId = &newCustomerId
  when duplicate
endfor
```



Análogamente, si tenemos dos transacciones Customer y CreditCard vinculadas por una relación 1 a 1: es decir, se ha especificado que CustomerId es una llave candidata (a través de un índice unique), y en un procedimiento que recibe por parámetro dos variables, el identificador de un cliente nuevo y un identificador de tarjeta de crédito existente, se programa un for each para cambiarle el cliente a esa tarjeta por el nuevo, si ya existía en la base de datos otra tarjeta para ese mismo cliente el for each no realizará la actualización. Si queremos tomar una acción en ese caso, contamos con la cláusula when duplicate.

Para adentrarse en esta cláusula, le recomendamos dirigirse a nuestro wiki: <http://wiki.genexus.com/commwiki/servlet/wiki?24843,When+duplicate+clause>.

Los atributos que se incluyan dentro de la cláusula when duplicate no serán tenidos en cuenta a la hora de determinar la tabla base del for each en el que se encuentran.

Sintaxis For each

```

For each BaseTransaction
  skip expression1, count expression2
  order att1, att2, ..., attn [when condition]
  order att1, att2, ..., attn [when condition]
  using DataSelector(parm1, parm2, ..., parmn)
  unique att1, att2, ..., attn
  where condition [when condition]
  where condition [when condition]
  where att IN DataSelector(parm1, parm2, ..., parmn)
  blocking N
    main code
  When duplicate
    ...
  When none
    ...
endfor

```

La cláusula blocking permite actualizar la base de datos en grupos de a N registros, para reducir los accesos y mejorar la performance. No lo abordamos en este curso. Por más información, dirigirse a:
<http://wiki.genexus.com/commwiki/servlet/wiki?4837,Blocking+clause+in+a+%27For+each%27+command>.

Por más información sobre la cláusula when duplicate, válida tanto para for each como para new, dirigirse al siguiente link:
<http://wiki.genexus.com/commwiki/servlet/wiki?24843,When+duplicate+clause>.

Para ver la sintaxis completa del comando new, dirigirse al siguiente link:
<http://wiki.genexus.com/commwiki/servlet/wiki?6714,New+command>.

Reiteremos que los atributos presentes en el código que se escriba dentro de la cláusula when duplicate no serán considerados a la hora de determinar la tabla base del for each en el que se encuentran. (Solo lo serán los subrayados en rojo)



Videos

training.genexus.com

Documentation

wiki.genexus.com

Certifications

training.genexus.com/certifications