# Actualización de la base de datos

Usando Business Component de un nivel (REPASO)

GeneXus

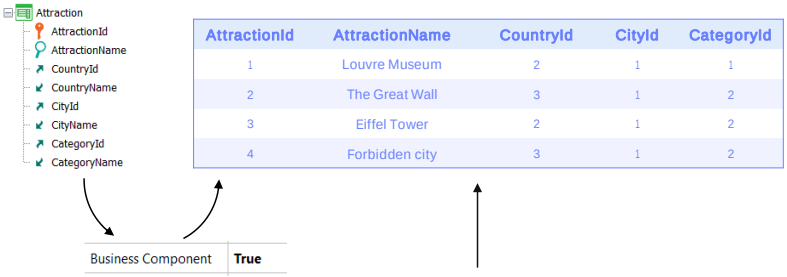
## Database update

Save()

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

InsertOrUpdate()

Insert, Update, Delete




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

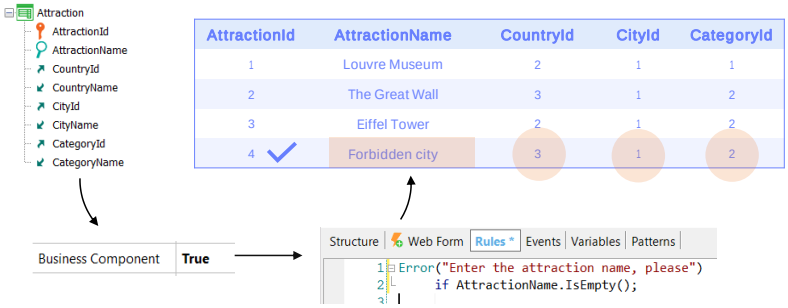
---

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 (o en lugar del Insert o del Update el Save, o incluso el InsertOrUpdate), o hacerlo exclusivamente dentro de un procedimiento, a través de los comandos New, For each con asignación directa de los atributos a ser modificados, y el comando Delete dentro de un for each para eliminar el registro en el que se esté posicionado.

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

Insert, Update, Delete




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

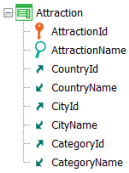
---

La enorme diferencia entre estas dos alternativas consistía en que mientras que la primera estaba fuertemente ligada a la lógica de la transacción, puesto que se disparaban las reglas, incluyendo el control de duplicados y de integridad referencial...

# Database update

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

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 | 1 | 100 |

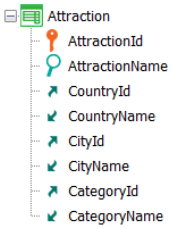
2. **Procedure**: New, For each, Delete

...en la segunda la actualización es independiente de la transacción, por lo que no se disparará regla alguna, y el único control que se realizará será el de duplicados: así podríamos asignar una categoría inexistente, que el programa no lo chequeará, aunque sí lo hará la base de datos y la ejecución se interrumpirá con una pantalla de error indeseable en el navegador del usuario).

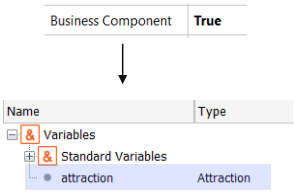
Sigamos profundizando, entonces, en la primera alternativa, la de los business components.

# BC: Insert

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 |



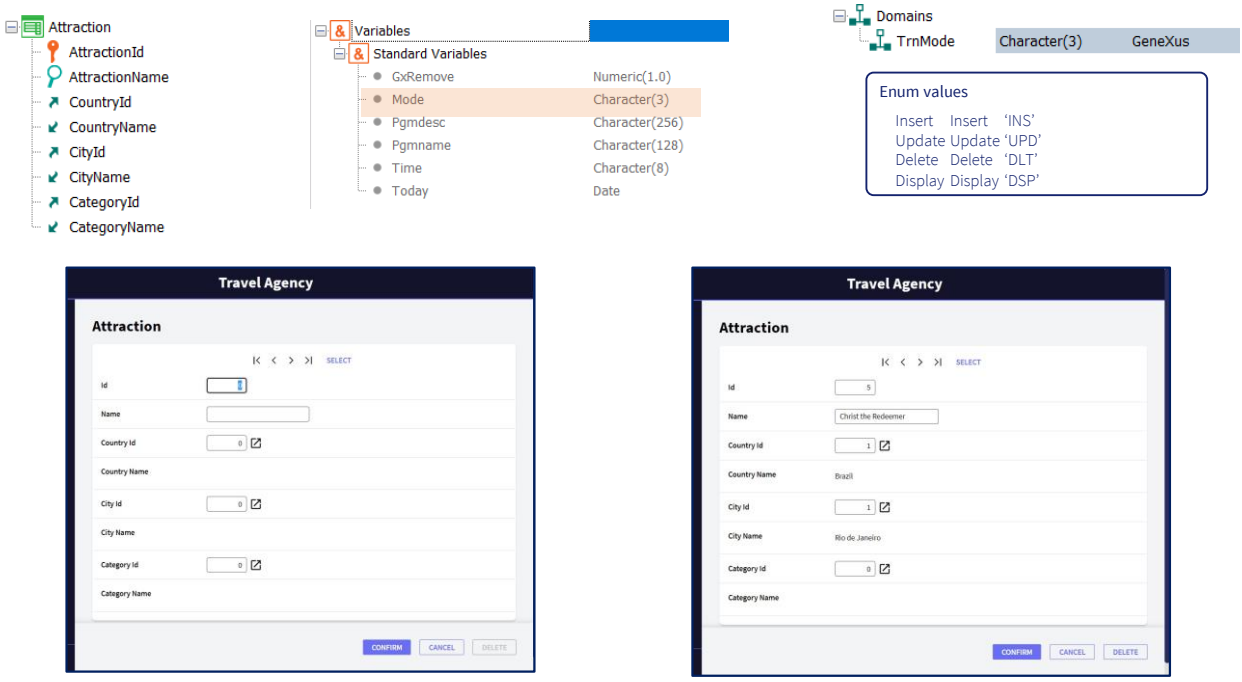

```
&attraction = new()

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

&attraction.Save()
if &attraction.Success()
  Commit
endif
```

Si queríamos insertar una atracción turística nueva utilizando el business component, alcanzaba con:

- crear una variable de ese tipo,
- asignarle un nuevo espacio de memoria (no es imprescindible pero es una buena práctica para asegurarnos de que, no importa qué haya pasado antes, la variable en este momento será completamente nueva).
- asignar valor a todos los elementos del Business Component correspondientes a atributos de la tabla:
    - si el identificador es autonumerado no necesitamos asignarle valor: lo hará la base de datos cuando insertemos,
    - si a algún atributo de la tabla no le asignamos valor quedará vacío o nulo. En general esto no es un problema, salvo para el caso de que el atributo sea clave foránea y no admita nulos; allí cuando intentemos insertar dará un error de falla de integridad referencial.
- Y finalmente, tras cargar la estructura del business component, solamente resta invocar al método Insert,
- Commiteando si así lo deseamos cuando la inserción es exitosa.

Esto es equivalente utilizar el método Save, ya que en este caso como la variable business component está en modo Insert intentará insertar (y no hacer un update).

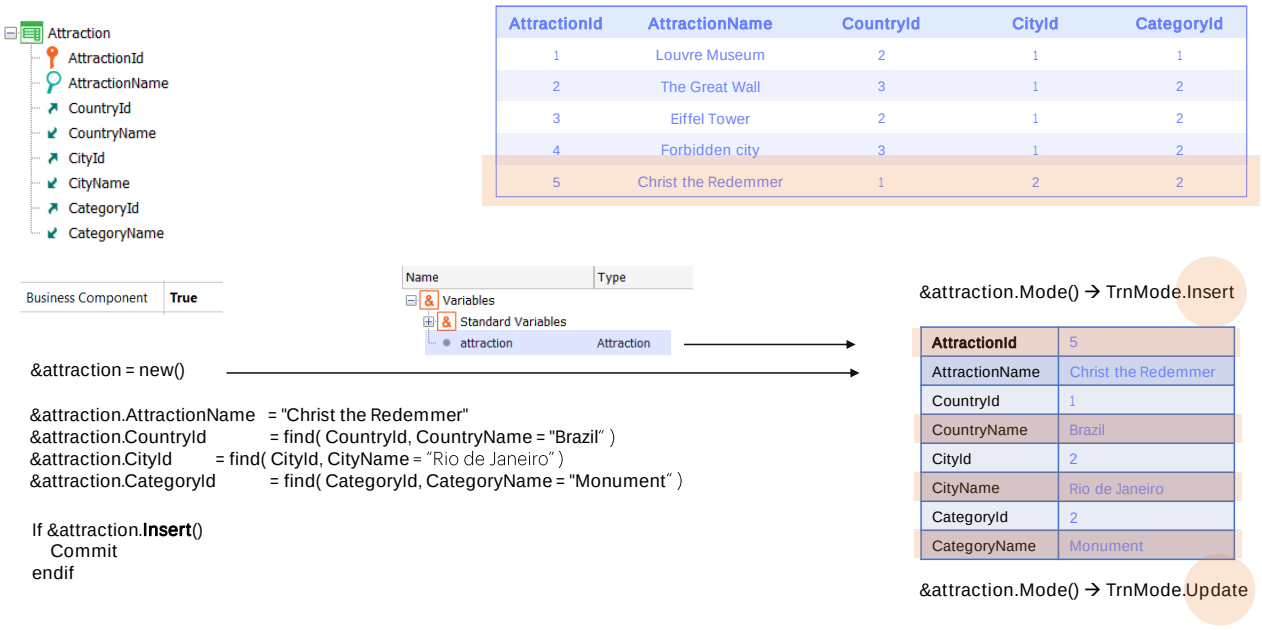
# Mode




Recordemos que entre las variables estándar de toda transacción, encontraremos la de nombre Mode. Esta variable contiene en todo momento el modo en el que se encuentra ejecutando la transacción.
Para poder manejar en alto nivel sus valores, el módulo GeneXus incorpora el dominio enumerado TrnMode, que puede asumir los 4 valores: Insert, Update, Delete, y Display -que indica que solamente se está visualizando la información pero no se hará nada con ella-.

Cuando la transacción se abre -si no le especificamos que reciba como parámetros modo e identificador- lo hace en modo Insert. Por eso los campos se encuentran vacíos. Cuando salimos del identificador, se busca si existe registro con el valor que le dejamos, en este caso 0, y como no encuentra, la transacción queda en modo Insert. Si tuviéramos reglas condicionadas con If Insert se irían disparando. Y grabamos. Al hacerlo nos informa que los datos se han insertado, pero además lo que podemos ver es que el formulario se ha vaciado nuevamente, lo que significa que la transacción ha vuelto a quedar en modo Insert. Como veremos esto no sucederá cuando insertemos a través del business component, que quedará en modo Update.
Si ahora elegimos un valor existente en la base de datos para el identificador, por ejemplo el 5 que es el que acabamos de insertar, al abandonar el campo la transacción trae cargados sus valores en los campos de la pantalla y queda automáticamente en modo Update. Podemos modificar algo, por ejemplo, quitarle la categoría, dejándola vacía (suponiendo que aceptamos nulos en esa clave foránea) y al confirmar se nos informa que el registo fue actualizado con éxito, pero además quedamos posicionados sobre ese mismo registro, en modo Update. Aquí sí, el business component se comportará de la misma manera.

# Mode()




Veamos qué pasa al trabajar con el business component.

Contamos con el método Mode() que nos permite consultar en qué modo se encuentra. Es de sólo lectura.

Si definimos una variable &attraction y consultamos antes de hacer nada en qué modo se encuentra, veremos que está en modo Insert.

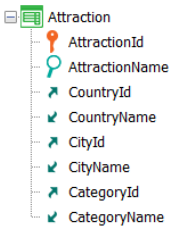
Toda vez que la reinicialicemos con new quedará en ese modo.

Luego ingresamos los valores que querremos dar a los atributos de la tabla asociada (solo a aquellos que no queramos que queden vacíos). Todo ese tiempo la variable business component seguirá en modo Insert.

¿Qué pasa una vez que se ejecuta el método Insert?
Se intentará realizar la inserción en la base de datos, ejecutando las reglas que correspondan. Si el proceso es exitoso, nos quedan cargados todos los elementos de la variable business component con los valores correspondientes. Como AttractionId es autonumerado obtenemos el valor que se le asignó en la base de datos, y los atributos que en la transacción son inferidos aquí también se cargan, de modo que podríamos consultarlos.
Y por otro lado, el modo de la variable business component pasa a ser Update.

# BC: Update

Insert, Update, Delete




| AttractionId | AttractionName | CountryId | CityId | CategoryId |
|---|---|---|---|---|
| 1 | Louvre Museum | 2 | 1 | 1 |
| 2 | The Great Wall | 3 | 1 | 3 |
| 3 | Eiffel Tower | 2 | 1 | 2 |
| 4 | Forbidden city | 3 | 1 | 2 |
| 5 | Christ the Redemmer | 1 | 2 | 2 |

| Business Component | True |
|---|---|

```
&attraction = Load(2)
&attraction.CategoryId  = find( CategoryId, CategoryName = "Tourist site" )

&attraction.Save()
if &attraction.Success()
  Commit
endif
```

&attraction.Mode() → TrnMode.Update

| AttractionId | 2 |
|---|---|
| AttractionName | The Great Wall |
| CountryId | 3 |
| CountryName | China |
| CityId | 1 |
| CityName | Beijing |
| CategoryId | 3 |
| CategoryName | Tourist site |

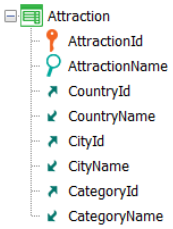
Por otro lado, si lo que queríamos era modificar la atracción turística de id 2, cambiándole la categoría por "Tourist site", alcanzaba con:

- cargar en la variable Business component &attraction los valores del registro de clave primaria 2 de la tabla Attraction (esto lo hacíamos con el método Load, que automáticamente, de existir ese registro, dejaba la variable en modo Update),
- asignar valor a todos los elementos del Business Component que queramos modificar (en este caso sólo CategoryId, pues los demás queremos que queden con el valor que tenían)
- Y finalmente invocar al método Update para realizar esa actualización en la base de datos,
- Commiteando si la operación fue exitosa.

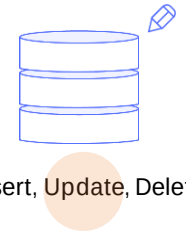
Una vez que se realiza el Update el elemento CategoryName, que es inferido en la transacción, queda en la variable con el valor correcto. La variable sigue quedando en modo Update.

También aquí es equivalente utilizar el método Save, ya que en este caso, como la variable business component luego del Load quedó en modo Update, el Save intentará actualizar y no insertar.

# BC: Update




| AttractionId | AttractionName | CountryId | CityId | CategoryId |
|---|---|---|---|---|
| 1 | Louvre Museum | 2 | 1 | 1 |
| 2 | The Great Wall | 3 | 1 | 3 |
| 3 | Eiffel Tower | 2 | 1 | 2 |
| 4 | Forbidden city | 3 | 1 | 3 |
| 5 | Christ the Redemmer | 1 | 2 | 2 |

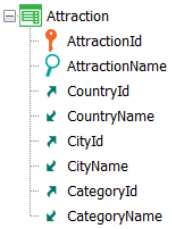


Insert, Update, Delete

```
For each Attraction
Where CountryName = "China" and CityName = "Beijing"
Where CategoryName = "Monument"

    &attraction.Load(AttractionId)
    &attraction.CategoryId = find( CategoryId, CategoryName = "Tourist site" )        st site" )

    If &attraction.Update()
       Commit
    endif

                endfor
```

Si lo que queríamos era modificar la categoría de todas las atracciones turísticas que siendo de Beijing correspondieran a monumentos, para asignarles ahora la categoría "Tourist Site", entonces alcanzaba con colocar el código anterior dentro de un for each que selecciona únicamente los registros deseados, y el Load se realiza por cada AttractionId de esos registros.

# BC: Update




| AttractionId | AttractionName | CountryId | CityId | CategoryId |
|---|---|---|---|---|
| 1 | Louvre Museum | 2 | 1 | 1 |
| 2 | The Great Wall | 3 | 1 | 3 |
| 3 | Eiffel Tower | 2 | 1 | 2 |
| 4 | Forbidden city | 3 | 1 | 2 |
| 5 | Christ the Redemmer | 1 | 2 | 2 |

Insert, Update, Delete

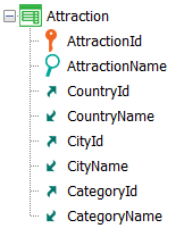
&attraction.Mode() → TrnMode.Update

```
For each Attraction
Where CountryName = "China" and CityName = "Beijing"
Where CategoryName = "Monument"

   &attraction.Load(AttractionId)
   &attraction.CategoryId = find( CategoryId, CategoryName = "Tourist site" )

   If &attraction.Update()
      Commit
   endif

endfor
```

| AttractionId | 2 |
|---|---|
| AttractionName | The Great Wall |
| CountryId | 3 |
| CountryName | China |
| CityId | 1 |
| CityName | Beijing |
| CategoryId | 3 |
| CategoryName | Tourist site |

Así, primero se hace el Load del de id 2, la variable queda en modo Update, luego se modifica su CategoryId, y al ejecutar el método Update se actualiza el registro en la tabla y la variable queda con el CategoryName correspondiente.

# BC: Update




| AttractionId | AttractionName | CountryId | CityId | CategoryId |
|---|---|---|---|---|
| 1 | Louvre Museum | 2 | 1 | 1 |
| 2 | The Great Wall | 3 | 1 | 3 |
| 3 | Eiffel Tower | 2 | 1 | 2 |
| 4 | Forbidden city | 3 | 1 | 3 |
| 5 | Christ the Redemmer | 1 | 2 | 2 |

Insert, Update, Delete

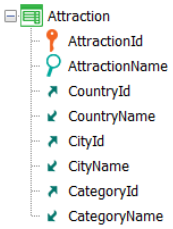
&attraction.Mode() → TrnMode.Update

```
For each Attraction
Where CountryName = "China" and CityName = "Beijing"
Where CategoryName = "Monument"

    &attraction.Load(AttractionId)
    &attraction.CategoryId = find( CategoryId, CategoryName = "Tourist site" )

    If &attraction.Update()
       Commit
    endif

                endfor
```

| AttractionId | 4 |
|---|---|
| AttractionName | Forbidden city |
| CountryId | 3 |
| CountryName | China |
| CityId | 1 |
| CityName | Beijing |
| CategoryId | 3 |
| CategoryName | Tourist site |

Y luego lo mismo para el siguiente registro que cumple las condiciones, esto es, el de id4.

# BC: Delete




| AttractionId | AttractionName | CountryId | CityId | CategoryId |
|---|---|---|---|---|
| 1 | Louvre Museum | 2 | 1 | 1 |
| 3 | Eiffel Tower | 2 | 1 | 2 |
| 4 | Forbidden city | 3 | 1 | 2 |
| 5 | Christ the Redemmer | 1 | 2 | 2 |
| 5 | Christ the Redemmer | 1 | 2 | 2 |

Insert, Update, Delete

```
&attraction.Load(2)
&attraction.Delete()

If &attraction.Success()
   Commit
endif
```

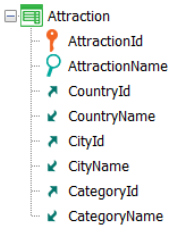
&attraction.Mode() → TrnMode.Update

| AttractionId | 2 |
|---|---|
| AttractionName | The Great Wall |
| CountryId | 3 |
| CountryName | China |
| CityId | 1 |
| CityName | Beijing |
| CategoryId | 2 |
| CategoryName | Monument |

&attraction.Mode() → TrnMode.Delete

Si en cambio queríamos eliminar una atracción turística, por ejemplo, la de identificador 2, primero teníamos que cargarla en la variable Business Component, tras lo cual queda en modo Update y luego utilizar el método Delete, que la borra de la tabla y deja la variable cargada con los datos pero en modo Delete. Luego preguntamos por Success para poder dar la orden de Commit.

# BC: Delete




| AttractionId | AttractionName | CountryId | CityId | CategoryId |
|---|---|---|---|---|
| 1 | Louvre Museum | 2 | 1 | 1 |
| 3 | Eiffel Tower | 2 | 1 | 2 |
| 5 | Christ the Redemmer | 1 | 2 | 2 |
| 4 | Forbidden city | 3 | 1 | 2 |
| 5 | Christ the Redemmer | 1 | 2 | 2 |

Insert, Update, Delete

```
For each Attraction
Where CountryName = "China" and CityName = "Beijing"
Where CategoryName = "Monument"

    &attraction.Load(AttractionId)
    &attraction.Delete()

    If &attraction.Success()
       Commit
    endif

endfor
```

Pero ¿y si lo que queríamos era eliminar todas las atracciones del tipo monumento de Beijing? Otra vez ubicábamos una a una las atracciones con el For each y las cargábamos con el load y luego utilizábamos el método Delete para eliminarlas.

GeneXus™

training.genexus.com
wiki.genexus.com