Fórmulas

Una mirada integradora

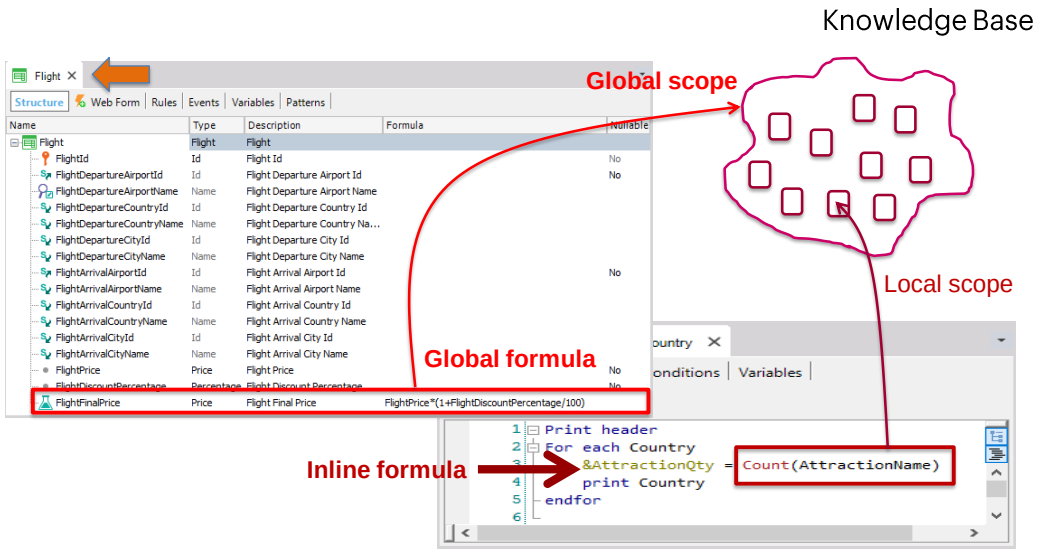
GeneXus™

En otros videos hemos explicado qué son las fórmulas globales y las fórmulas inline, analizando los distintos tipos según su navegación, como las fórmulas horizontales, fórmulas de agregación, fórmulas compuestas y sus respectivos casos de uso.

En este video trataremos de aportar una mirada más en perspectiva y no tan en detalle, que descubra cuándo conviene usar las fórmulas según su tipo y cuándo podemos usar una solución alternativa, en qué casos tenemos restricciones y cómo podemos levantarlas, qué costo implica agregarles redundancia y otras observaciones que nos ayuden a integrar los conceptos sobre este tema.

Si tiene alguna duda sobre los conceptos de las fórmulas sobre los que se basa este video, o desea refrescarlos antes de ver esta síntesis, le sugerimos ver los videos de fórmulas del Curso GeneXus Avanzado.

Global formulas vs. Inline formulas (local formulas)



Una **fórmula global** se conoce también como “atributo fórmula” y es un cálculo que se asigna a un atributo en la estructura de una transacción.
Se llaman globales debido a que al definirse el cálculo en un atributo, su definición queda a nivel de la base de conocimiento, y por lo tanto todos los objetos tendrán acceso al atributo con dicho cálculo.

A partir de que asociamos un cálculo a un atributo, éste no se almacenará como campo de una tabla en la base de datos y por esa razón se les llama también llamado “atributo virtual”. Sin embargo, el atributo formula global queda asociado a la tabla a la que hubiera pertenecido si no se hubiera definido como fórmula. Esta tabla asociada representa el contexto de la fórmula, es decir que en el momento que la formula se dispara, se está posicionado en un determinado registro de esa tabla.

Solamente los atributos pueden ser definidos como fórmulas globales y no las variables, ya que los atributos tienen alcance global en la base de conocimiento, mientras que el alcance de las variables se restringe al objeto donde fueron definidas y no pueden ser accedidas desde otro objeto.

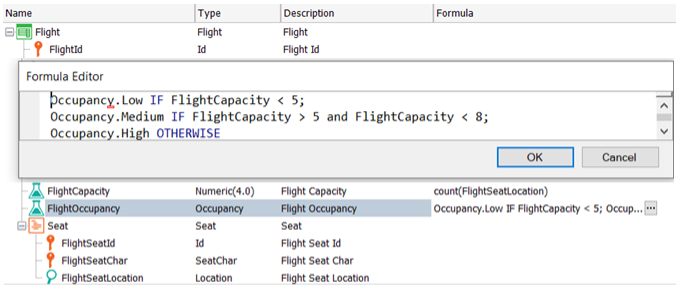
Si usamos una formula en un cálculo que implementamos cuando escribimos código en un objeto, como por ejemplo en el source de un procedimiento o eventos de un panel o web panel, o en un data provider, o en el tab Conditions de un objeto, estamos definiendo una **fórmula inline**.

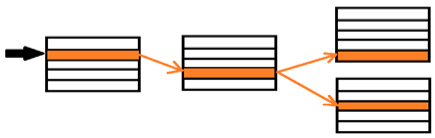
Como podemos escribir formulas inline en cualquier objeto en el que podamos escribir código, la mayoría de las veces asignamos este cálculo a una variable, ya que solamente podríamos asignar el valor devuelto por una fórmula a un atributo si estamos actualizando la tabla mediante un For Each en un objeto procedimiento.

Como las variables tienen alcance local al objeto, a las formulas inline se las conoce también como fórmulas locales.

En el caso de los data providers, los elementos del lado izquierdo de las asignaciones forman parte de una estructura jerárquica que solamente tiene valor dentro del data provider, por lo que su alcance es también local. Lo mismo sucede en el caso de que se usen formulas inline en el tab Conditions de un objeto.

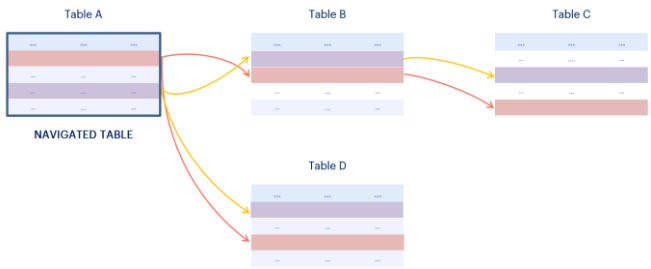
When to use horizontal formulas and when to use aggregation formulas





Attribute =

$expression_1$ **if** $condition_1$;
$expression_2$ **if** $condition_2$;
...
$expression_n$ **if** $condition_n$;
$expression_o$ **otherwise**;



Aggregate formulas:

- Count
- Sum
- Average
- Max
- Min
- Find

La navegación de la fórmula determina si la fórmula es horizontal o de agregación.

Cuando tenemos una fórmula que navega un único registro de una tabla (accediendo eventualmente a atributos de su tabla extendida), hablamos de una **fórmula horizontal**.

Si en cambio la fórmula accede a muchos registros de una tabla, entonces es una **fórmula de agregación**.

Por lo tanto si lo que queremos es recuperar un valor a partir de un cálculo que puede obtenerse con datos de un solo registro y los registros asociados de su tabla extendida, entonces escribimos una fórmula horizontal. Este tipo de fórmulas nos permite asignar distintos valores dependiendo de ciertas condiciones y también tomar un valor por defecto en caso de que no se cumpla ninguna de las condiciones establecidas.

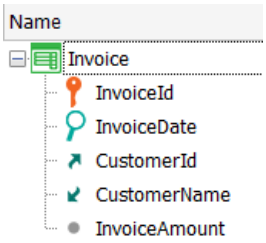
Si en cambio queremos un valor que depende de la búsqueda y/o procesamiento de múltiples registros (y sus registros asociados de la tabla extendida), entonces utilizaremos algunas de las fórmulas de agregación, como: count, sum, max, min, find o average.

Las fórmulas de agregación pueden ser globales (asignadas a atributos a nivel de la estructura de la transacción, o locales (inline), asignadas a atributos o variables, elementos de un data provider, o siendo parte de filtros **(como tab conditions o directamente en for eachs, grupos de Data Providers, etc.)** que son evaluados en tiempo de ejecución.

Optional parameters in Aggregation Formulas

AggregateFormula(AggregateExpression, AggregateCondition, DefaultValue, ReturnedValue) if TriggeringCondition

optional optional optional optional



Sum(InvoiceAmount) : summarize InvoiceAmount from all records of the INVOICE table

Sum(InvoiceAmount. InvoiceDate=&today) : summarize the InvoiceAmount values from the invoices issued today

Max(InvoiceAmount. InvoiceDate=&today, 0, InvoiceAmount) : Retrieves the máximum value of InvoiceAmount from the invoices issued today. If there is no invoices issued today, the returned value will be 0.

Max(InvoiceAmount. InvoiceDate=&today, 0, InvoiceAmount) if CustomerId = 4 : Only for issues of customer with Id=4, it retrieves the máximum value of InvoiceAmount from the invoices issued today. If there is no invoices issued today, the returned value will be 0.

Veamos los parámetros de una fórmula de agregación y qué rol cumple cada uno.

Las fórmulas de agregación tienen un primer parámetro obligatorio que es la expresión de agregación, que establece cuál será la tabla recorrida por la fórmula.
Este atributo es recomendable que no sea clave, para evitar posibles ambigüedades a la hora de que GeneXus determine la tabla a navegar, fundamentalmente si en las demás partes de la fórmula no aparecen otros atributos que permitan determinar con unicidad la tabla, por ejemplo cuando la fórmula no tiene más parámetros que el primero.
En particular este atributo puede ser un atributo fórmula, no tiene porqué ser un atributo almacenado, ya que todo atributo fórmula tiene una tabla asociada.

Además del primer parámetro obligatorio, una fórmula de agregación puede tener otros parámetros, todos opcionales.

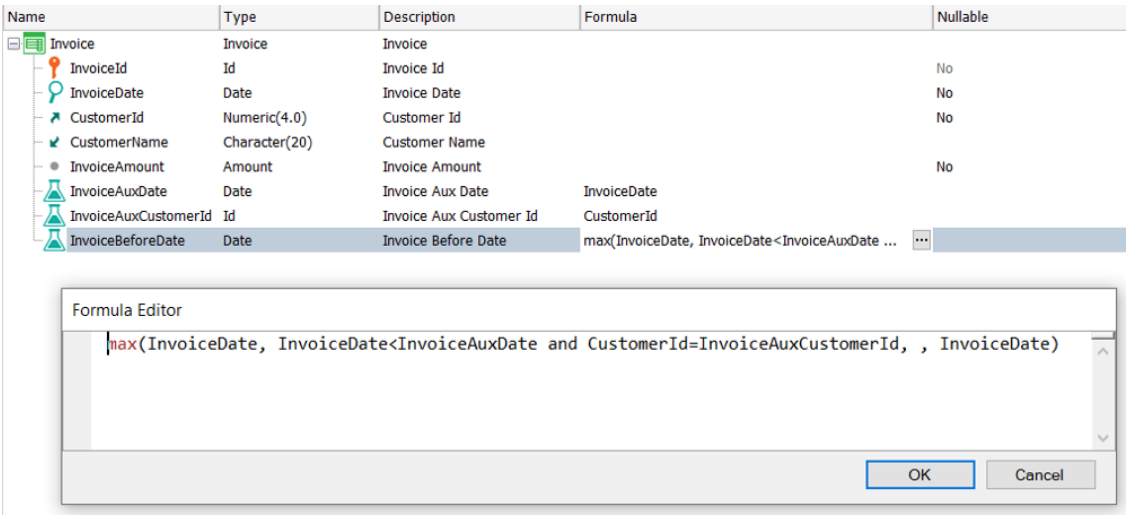
El primero luego de la expresión de agregación es la condición de agregación, es decir la condición que deben cumplir los registros que serán contados, sumados, promediados, etc.

El siguiente parámetro es el valor por defecto devuelto por la fórmula si no encuentra ningún registro que cumpla la condición de agregación.

El último parámetro será el atributo que contenga el valor que debe devolverse en caso de que se hayan procesado registros.

Luego del if, la condición de disparo establece qué condición debe cumplirse para que la fórmula se ejecute. Esta condición es una expresión lógica tan compleja como se necesite.

Restrictions on global aggregation formulas



Como la elección del atributo de la expresión de agregación es arbitraria, la formula puede en principio navegar sobre cualquier tabla de la base de datos. Por lo tanto, podríamos hacer que la tabla a recorrerse sea la misma que la tabla asociada al atributo fórmula.

Analicemos lo que pasaría en este caso.

Veamos un caso en el que dada una factura de un cliente, se desea encontrar la fecha de la factura anterior del mismo cliente. Para resolver esto utilizaremos una fórmula Max que encuentre la fecha de factura máxima, que sea menor o igual a la fecha de la factura dada.
Nos damos cuenta que debemos diferenciar a los atributos que son del registro de la tabla asociada, de los atributos que son de registros de la tabla navegada donde se realizará la búsqueda
Para eso agregamos los siguientes atributos auxiliares: InvoiceAuxCustomerId definido como fórmula horizontal que toma el valor de CustomerId y el atributo InvoiceAuxDate que toma el valor de InvoiceDate.

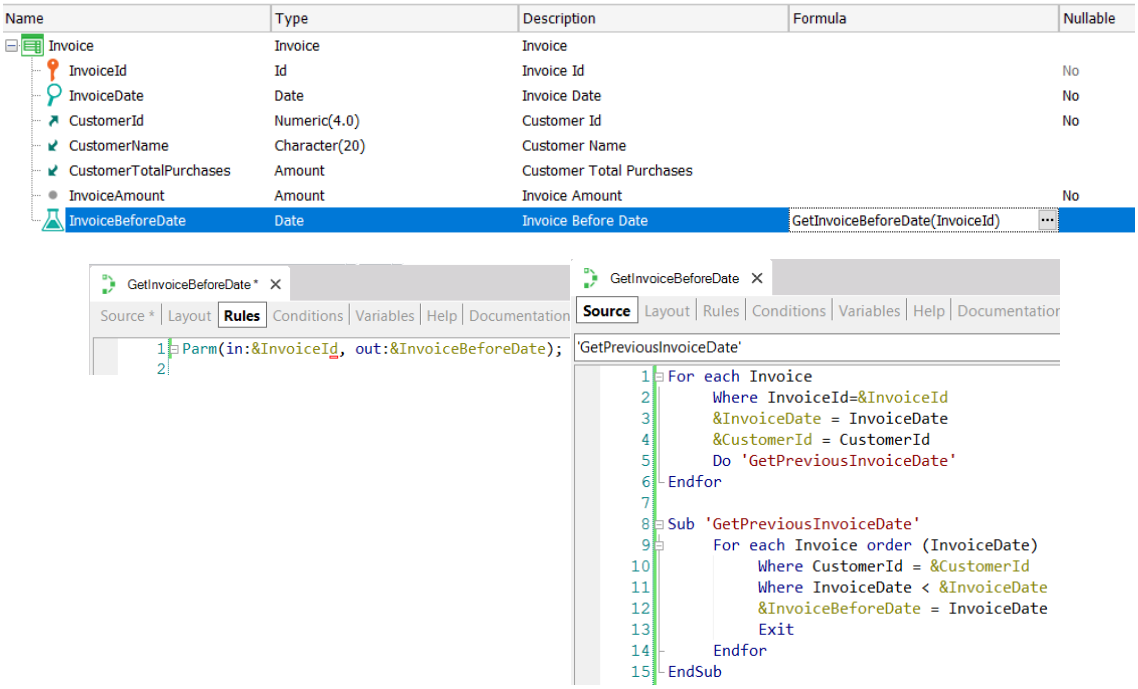
Notemos que estamos intentando definir una fórmula que navegará sobre la misma tabla que está asociada a la fórmula. La fórmula está definida en la transacción Invoice, por lo que su tabla asociada es INVOICE y la tabla navegada también es INVOICE.

Como vimos antes, la tabla asociada representa el contexto de la fórmula, por lo que cuando la formula se dispara, se está posicionado en un determinado registro de esa tabla. Por lo tanto, la fórmula quedará filtrada por el identificador del registro donde se esté posicionando y solamente navegará por un único registro!

Es decir que no podremos tener una fórmula aggregate global que navegue la misma tabla donde está definida, ya que no podrá recorrerla como necesitábamos en este caso para hallar el máximo.

Si necesitamos esta funcionalidad, podríamos definir la fórmula invocando un objeto procedimiento que nos devuelva el cálculo, como veremos a continuación.

Global formulas that call procedures



Vamos a crear un procedimiento que recibe la factura de un cliente y nos devuelva la fecha de la factura anterior del mismo cliente.

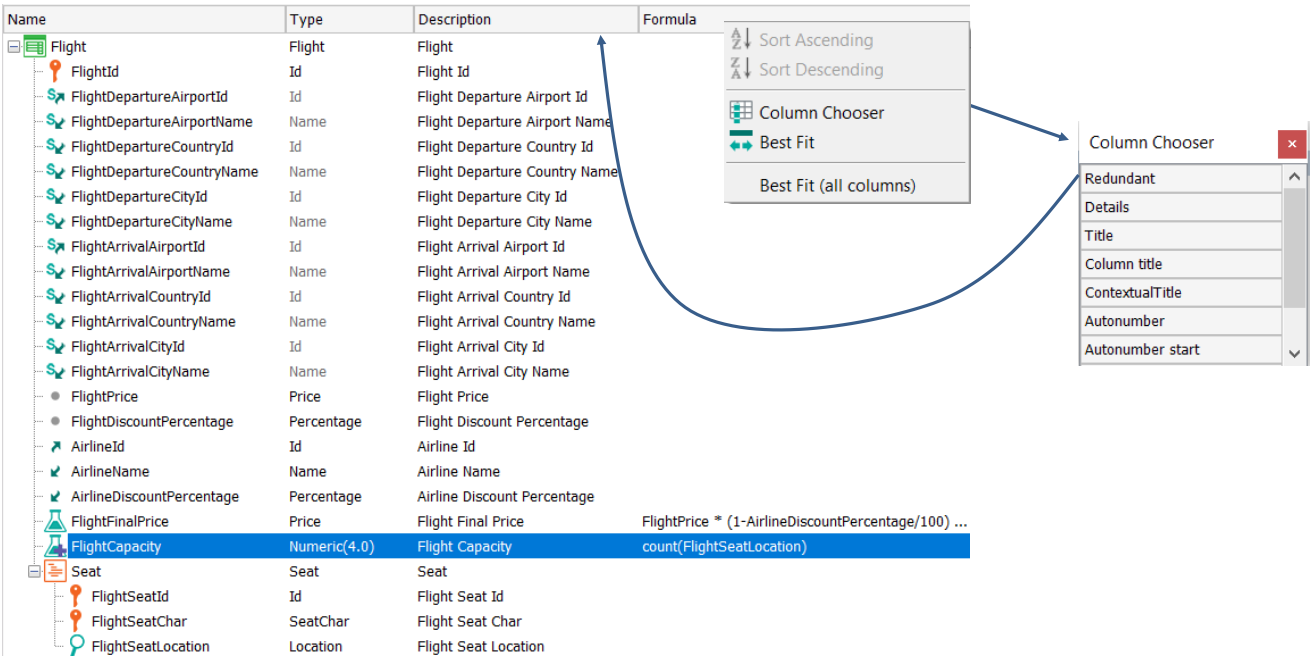
Así que definimos al atributo InvoiceBeforeDate como fórmula y en el form editor invocamos al procedimiento GetInvoiceBeforeDate, pasándole como parámetro el InvoiceId de la factura de interés.

En el source implementamos un For Each que recorre la tabla INVOICE filtrando por el InvoiceId recibido por parámetro y recuperamos la fecha y el cliente de la factura recibida. Luego invocamos a una subrutina que recorre nuevamente la tabla de facturas ordenada por factura en orden descendente y nos quedamos con la factura del mismo cliente cuya fecha sea la mayor posible, pero que sea menor a la factura recibida por parámetro.

Con esto estamos haciendo lo mismo que pretendíamos con la fórmula max, es decir, maximizar la fecha de la factura del mismo cliente pero que sea menor a la de interés, en otras palabras la fecha de la factura anterior del mismo cliente.

El hecho de que podamos definir una fórmula que invoque un procedimiento para que realice el cálculo necesario, nos abre muchas posibilidades ya que este calculo puede ser tan complejo como queramos y luego podremos simplemente utilizar el atributo fórmula desde cualquier objeto de nuestra base de conocimiento.

How to define a formula attribute as redundant

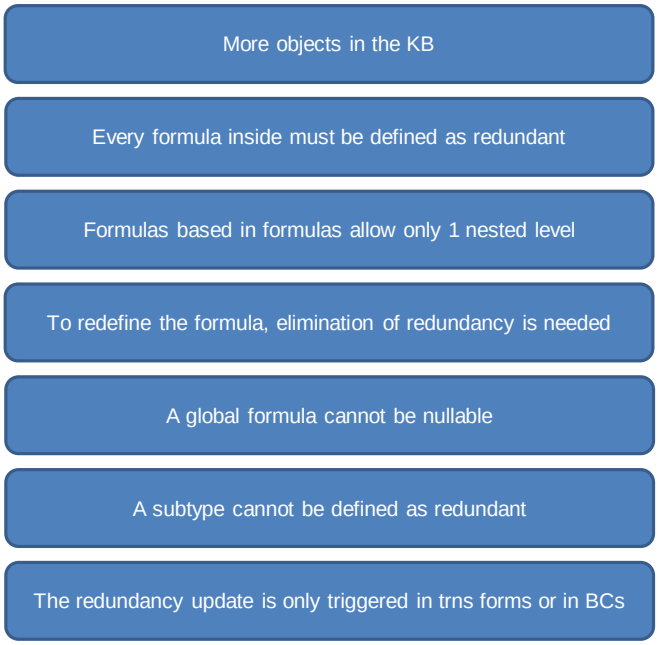


Si bien las fórmulas redundantes se abordan exhaustivamente en otro video, repasaremos aquí algunos conceptos.

En algunos casos en los que la fórmula se dispara muchas veces, por razones de performance nos conviene definir al atributo fórmula global como redundante. Esto hará que el atributo pase a estar almacenado en la base de datos y por lo tanto el tiempo que insumirá recuperar el valor será menor que el necesario para realizar los cálculos.

Para definir un atributo como redundante lo hacemos desde la estructura de la transacción, dando botón derecho sobre la barra de columnas, hacemos clic sobre Column Chooser y agregamos la columna Redundant arrastrándola hasta la barra de columnas. Luego en la columna Redundant marcamos el checkbox para definir al atributo como redundante. En el símbolo de la fórmula se le agrega un signo de “+” para indicar que es redundante.

Implications of defining a global formula as redundant



Definir un atributo como redundante tiene cierto costo que debemos considerar.

En primer lugar, para que el valor de la base de datos está siempre actualizado, GeneXus creará objetos procedimientos que se dispararán cada vez que cambien los datos involucrados en el cálculo de la fórmula. Esto implica que por cada atributo fórmula que definamos como redundante, aumentará la cantidad de objetos en la base de conocimiento.

Pero además, si el atributo fórmula que queremos sea redundante fue definido en base a otros atributos que también son fórmula, deberemos también definir a esos otros atributos también como redundantes.

Otra limitación es que en fórmulas de agregación que se hacen redundantes, definidas en base a atributos que también son fórmulas redundantes, sólo se permite un nivel de anidación. Si se supera este límite, sus redundancias no serán correctamente mantenidas.

Para cambiar la definición de una fórmula que es redundante, primero hay que eliminar la redundancia, hacer el cambio y definir la fórmula como redundante nuevamente.

Las fórmulas globales definidas como redundantes, no admiten valores nulos, por lo que no podremos setear la propiedad Nullable en Yes.

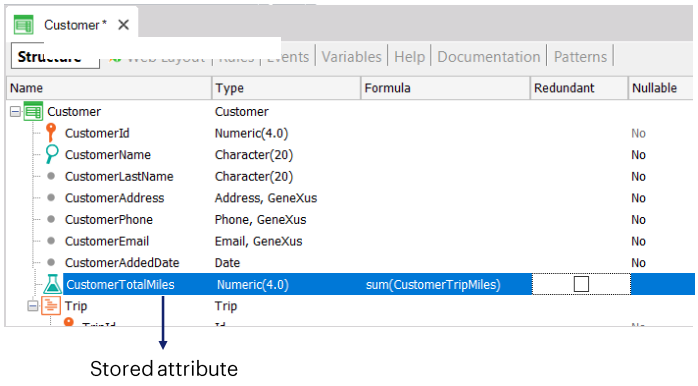
Un subtipo que sea fórmula, no puede ser definido como redundante.

Por último, los procedimientos encargados de actualizar el valor almacenado, se dispararán únicamente cuando se edita el registro mediante el form de la transacción o mediante un business component de la misma. Esto implica que si se está cambiando los datos involucrados en la fórmula desde un objeto procedimiento, no se dispararán los procedimientos de actualización en forma automática y deberemos forzar esta actualización en forma explícita, invocando a un utilitario creado por GeneXus para tal fin.

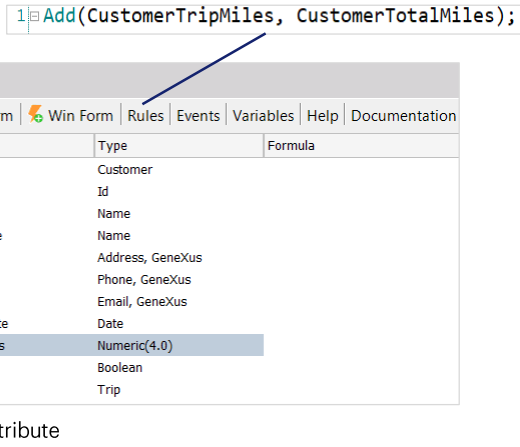
En conclusión, debemos pensar con cuidado cuando decidamos definir un atributo fórmula como redundante, ya que es posible que estas restricciones hagan inviable nuestra implementación.

Alternative solutions to a redundant formula

Vs.



Stored attribute



Stored attribute

Cuando el valor de un atributo puede obtenerse mediante un cálculo, por lo general lo definimos como fórmula en la estructura de la transacción y este atributo deja de estar almacenado en la base de datos. Si necesitamos que el atributo quede siempre almacenado, podemos definirlo como redundante, con lo cual GeneXus crea automáticamente procedimientos encargados de actualizar su valor y almacenarlo en la base de datos.

Sin embargo, también podríamos asignar el cálculo al atributo mediante una regla. El atributo no deja de estar presente en la tabla por el mero hecho de asignarle un valor, por lo que no se convierte en un atributo virtual, sino que sigue siendo un atributo almacenado.

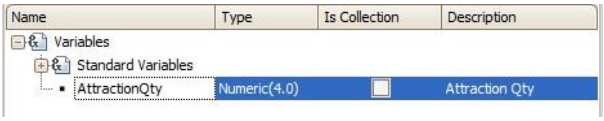
Por lo tanto, desde el punto de vista de la actualización y que el atributo esté almacenado, actualizar el atributo mediante una regla y definirlo como fórmula redundante podrían considerarse soluciones equivalentes.

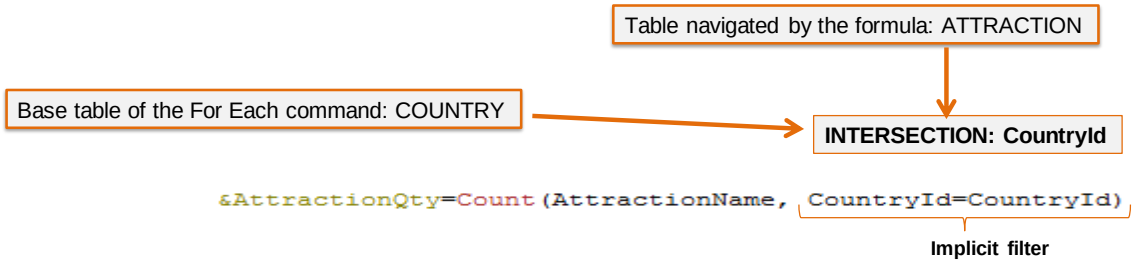
Como ventaja adicional con la regla el atributo seguirá siendo editable en el form de la transacción, aunque como desventaja no se puede forzar el disparo de la regla a demanda, por lo que no se puede forzar la actualización del atributo, cosa que sí se puede hacer con un atributo fórmula redundante.

Por lo tanto, cuándo usar una solución o la otra, depende en definitiva del uso que le daremos al atributo en nuestra aplicación.

Inline formulas in a For Each: implicit filters and edge cases

```
Print header
For each Country
    Where Count(AttractionName) > 2
    &AttractionQty = Count(AttractionName)
    Print Country
Endfor
```





Veamos el caso de fórmulas inline definidas en el cuerpo de un comando For each.

Como ya vimos, las fórmulas determinan la tabla a navegar, por el o los atributos referenciados dentro del paréntesis. En este caso hemos incluido el atributo AttractionName, por lo que la tabla que navegará la fórmula es ATTRACTION.

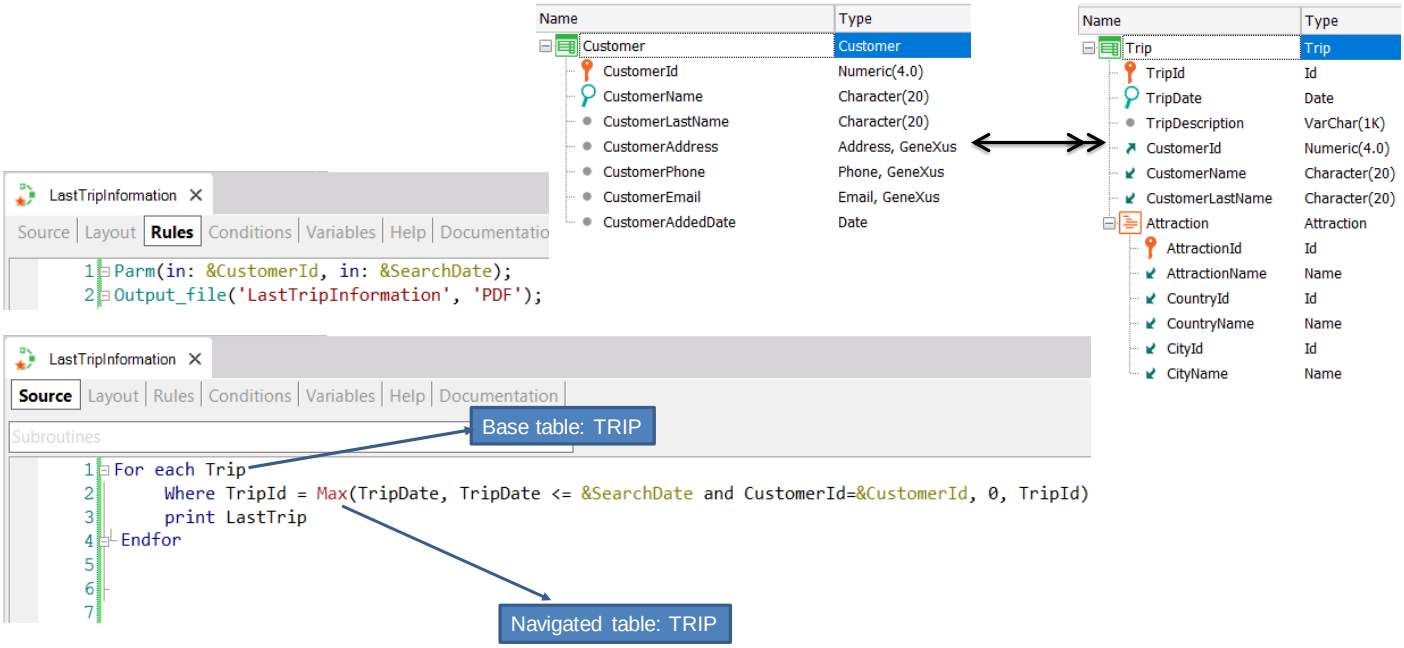
En el ejemplo no queremos contar todas las de la tabla ATTRACTION, sino las que corresponden al país en el que estamos posicionados en cada iteración del For each que recorrerá la tabla COUNTRY.

Como la fórmula está definida dentro de un comando For each, se encuentra en un contexto en el que se está recorriendo una tabla, en este caso, la de países. Vemos que hay un atributo en común entre las dos navegaciones, CountryId, y es por eso que GeneXus determina un filtro implícito por el atributo en común, de modo que para cada país que el For each encuentre, se cuentan únicamente las atracciones que son de ese país.

Otro requerimiento solicitado es que solamente se listen los países es que tengan más de dos atracciones turísticas.
Para solucionar esto, podemos utilizar una fórmula count como parte de la expresión de filtro en la cláusula Where del For Each.

Esta fórmula count, definida igual que la anterior, nos retornará la cantidad de atracciones del país en el que está posicionado el For Each en cada iteración (debido al filtro implícito mencionado antes) y utilizamos eso para filtrar aquellos países para los cuales la cantidad de atracciones sea mayor que dos.

Inline formulas in a For Each: implicit filters and edge cases



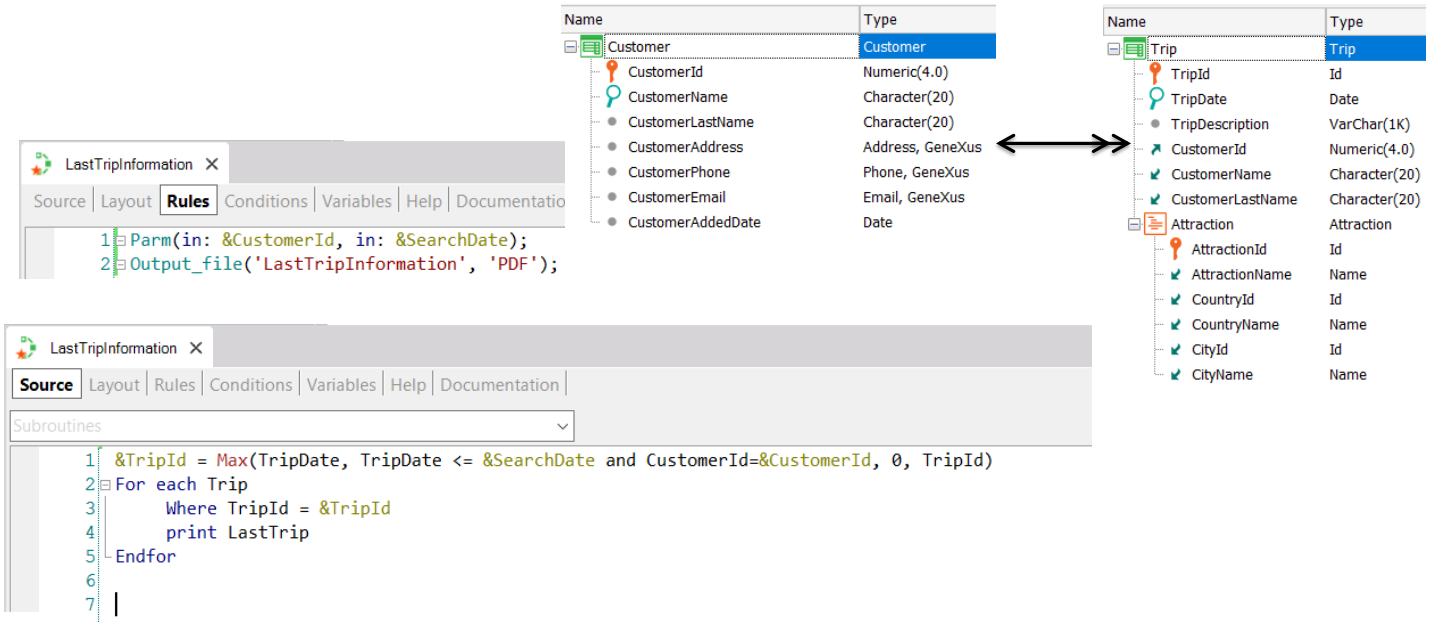
Veamos ahora un caso particular, en el que coincide la tabla navegada por la formula con la table base del For Each.

Supongamos que dado un cliente que tiene muchos viajes, quiere recuperar los datos de un viaje realizado, que haya sido inmediatamente anterior a una fecha dada, es decir el último viaje antes de esa fecha. El procedimiento recibe por parámetro el identificador del cliente que desea la información y la fecha de búsqueda.

En el source del procedimiento hay un for each que recorre la tabla Trip y el where filtrará por el TripId devuelto por la fórmula Max. Luego se imprimirán los datos correspondientes a ese viaje.

Si analizamos lo que hemos implementado, el For Each iterará sobre la tabla TRIP y establecerá ese contexto para la fórmula Max.
La fórmula Max también iterará en la tabla TRIP, pero debido al contexto, se establecerá un filtro automático, dado que ambas tablas están relacionadas. En este caso es la misma tabla, por lo que la fórmula quedará filtrada por el TripId que esté posicionado el For Each. Por tanto, va a devolver siempre ese TripId en el que el For Each se encuentra, por lo que la fórmula nunca se recorrerá la tabla TRIP ya que solamente accederá a un registro.

Inline formulas in a For Each: implicit filters and edge cases

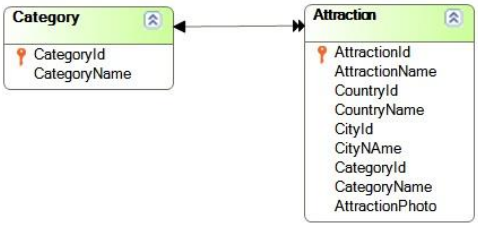


La solución en este caso, es ejecutar primero la fórmula para busque el identificador del viaje que cumple las condiciones deseadas y luego filtramos al For Each por ese valor de identificador.

Vemos que en este ejemplo si definimos una fórmula inline dentro de un contexto (la tabla base del for each), sucede lo mismo que vimos en el ejemplo de la fórmula global, cuya tabla asociada coincidía con la tabla navegada, siendo la tabla asociada la que brindaba el contexto.

A diferencia de las fórmulas horizontales que necesitan un contexto (ya que solamente pueden utilizar atributos de la tabla asociada y su extendida), las fórmulas de agregación no lo necesitan, pero si se definen dentro de un contexto, éste actuará como filtro de la fórmula.

Inline formulas in a For Each: implicit filters and edge cases



Returns the list (without repeating elements) of all categories that have attractions, each with the corresponding number of registered attractions.

```
Print Title
For each Attraction
      Unique CategoryId
      &Quantity = Count(AttractionId)
      Print Categories
Endfor
```

Base table of the For each: ATTRACTION

Table navigated by the formula: ATTRACTION

Veamos ahora otro caso de una formula inline definida dentro de un For Each, en el que coincide la tabla navegada por la fórmula con la table base del For Each. Observemos que en ambos casos, la table es ATTRACTION.

Sin embargo, en este ejemplo hemos agregado una clausula Unique por CategoryId y debido a esto, GeneXus agrupará las atracciones por categoría, tanto en el For each como en la fórmula, por lo que la fórmula Count agregará del contexto una condición implícita en su evaluación: contará todas las atracciones para el atributo declarado en la cláusula unique.

Es como si fuera un corte de control, cortando por CategoryId.

Gracias a esa condición adicional brindada por la cláusula Unique, GeneXus puede resolver el problema de que la tabla navegada por la fórmula coincide con la tabla base del For each, ya que únicamente se contarán las atracciones de la categoría dada en la cláusula Unique..

Compound formulas

```
Attribute =  Max(...) if condition1;
             (2 * attrX) + 100  if condition2;
             Sum(attrY) otherwise

Attribute = procedure(...) if condition1;
            Min(...)  if condition2;
            10   if condition3

Attribute  =  2 + Count(…) *   Sum(…) if condition;
              Attr1 + Attr2 * Attr3  otherwise


Attribute  =  Count(…) if condition1;
              Sum(…) if condition2;
              Find(…) if condition3;
```

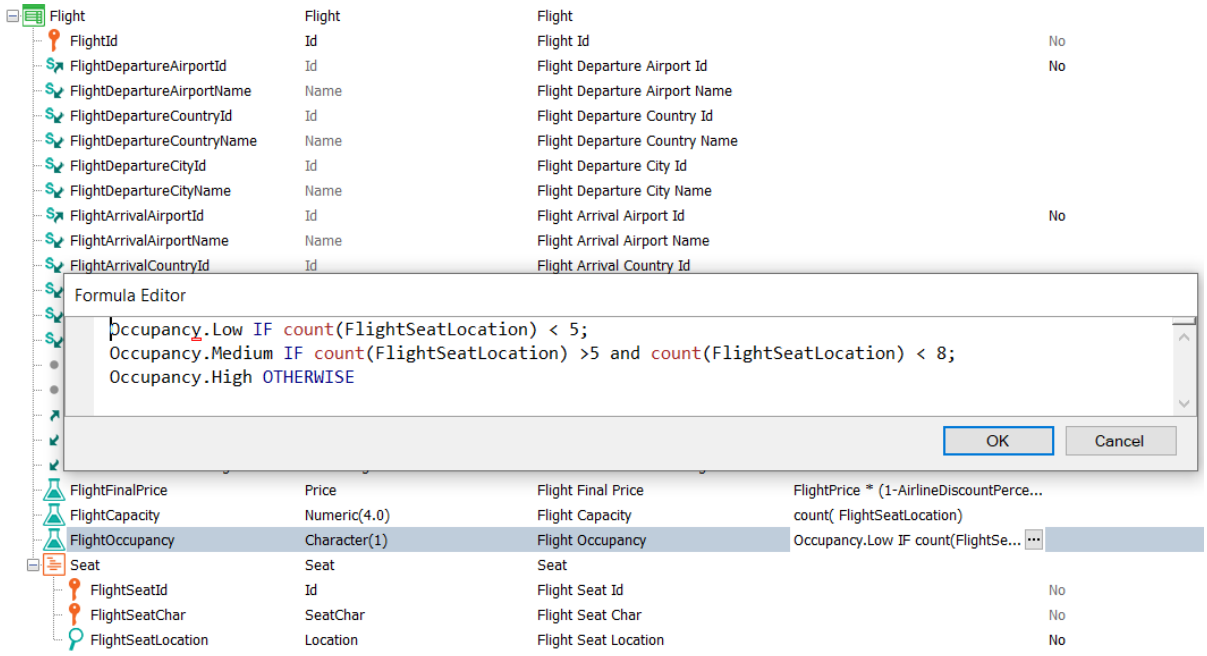
En GeneXus podemos definir fórmulas complejas, usando fórmulas compuestas.

Las fórmulas compuestas son fórmulas que integran varias fórmulas aggregate condicionales, pudiendo también contener expresiones horizontales con condiciones de disparo.

Las condiciones son cualquier expresión lógica válida, que puede contener atributos pertenecientes a la tabla extendida, de la tabla asociada al atributo que se está definiendo como fórmula.

La primera condición que al ser evaluada de True, provocará que el resultado de la fórmula sea el de la expresión de la izquierda de esa condición y las demás expresiones no se seguirán evaluando.

Example



Veamos un ejemplo de este tipo de fórmulas compuestas en nuestra realidad de la agencia de viajes.

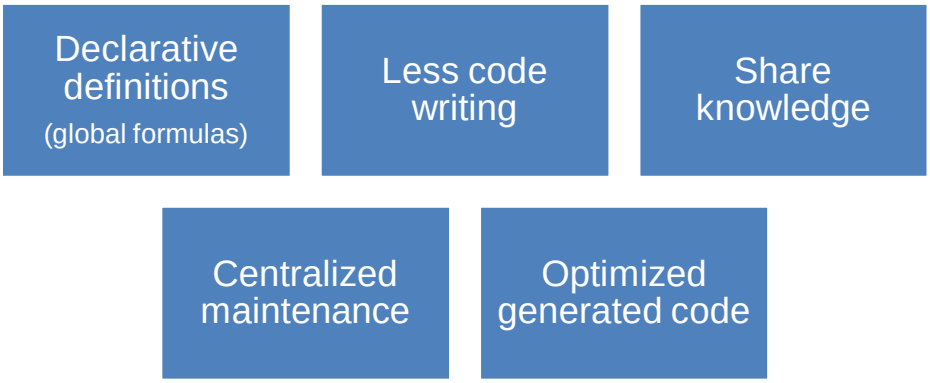
Aquí vemos que el atributo FlightOccupancy se definió en base a expresiones horizontales que asignan el valor correspondiente del dominio Occupancy (Low, Medium o High), dependiendo de la cantidad de asientos del vuelo, que se calculan con fórmulas aggregate count.

En particular, en nuestro caso, podríamos haber sustituido las fórmulas aggregate por el atributo FlightCapacity, pero es perfectamente válido dejarlo así como está definido.

En esta implementación, la estructura es la de una fórmula horizontal y las aggregate se incluyeron en las condiciones de disparo.

Las fórmulas compuestas nos permiten una gran flexibilidad en la definición de cálculos, pudiendo modelarse una gran cantidad de situaciones.

Conclusions



Después de haber visto este repaso conceptual del uso de las fórmulas en GeneXus, concluimos que son muy útiles en muchos casos, y fundamentalmente nos aportan las siguientes ventajas:

- Definición declarativa en lugar de código procedural para fórmulas globales
- Nos permiten ahorrar código, especialmente en las funcionalidades de las fórmulas aggregate que procesan muchos registros, que nos evita tener que iterar sobre los registros e implementar la lógica por código para contar, sumar, maximizar, etc.
- Son una forma de compartir conocimiento, por ejemplo en el caso de los atributos fórmulas que pueden ser usados en cualquier objeto de la base de conocimiento
- El mantenimiento es centralizado, ya que en el caso de una fórmula global, cambiamos la definición en un único lugar, en la estructura de la transacción donde fue definido el atributo.
- Definir fórmulas es incluso mejor que escribir procedimientos e invocarlos. Cuando se define una fórmula, GeneXus tiene conocimiento de su definición y es capaz de generar sentencias optimizadas combinando la consulta de la fórmula, con la consulta en la que está presente la fórmula.

En resumen, las fórmulas nos facilitan mucho el desarrollo de nuestras aplicaciones y es altamente aconsejable aprender a usarlas.

Si quiere saber más sobre este tema, lo invitamos a ver más contenido en el wiki.

training.genexus.com
wiki.genexus.com
training.genexus.com/certifications