Más acerca de fórmulas

Globales e Inline

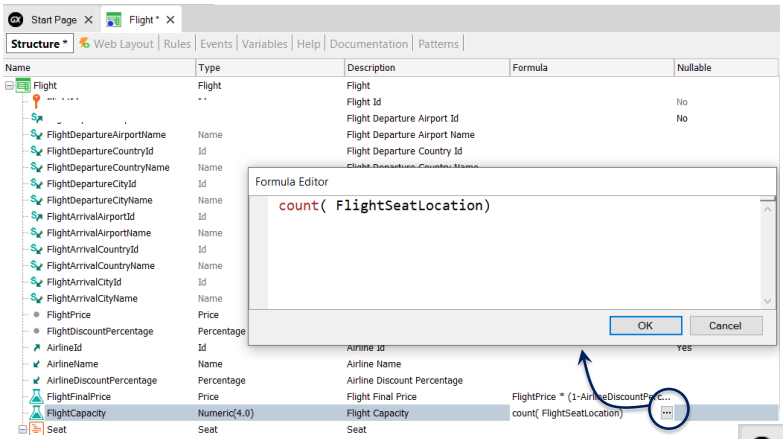
GeneXus™

A continuación, profundizaremos sobre algunos conceptos de las fórmulas, tanto globales como inline.

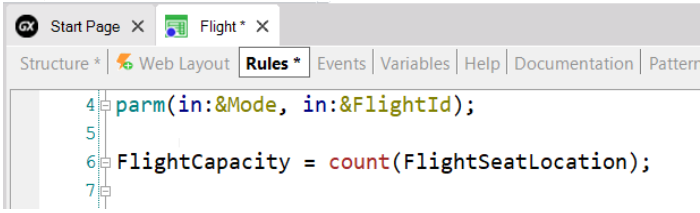
Fórmula versus Regla para definir un atributo

Veamos la diferencia entre definir un atributo mediante un cálculo en una fórmula o en una regla de asignación.

Consideraciones al definir un atributo en base a un cálculo: ¿regla o fórmula?



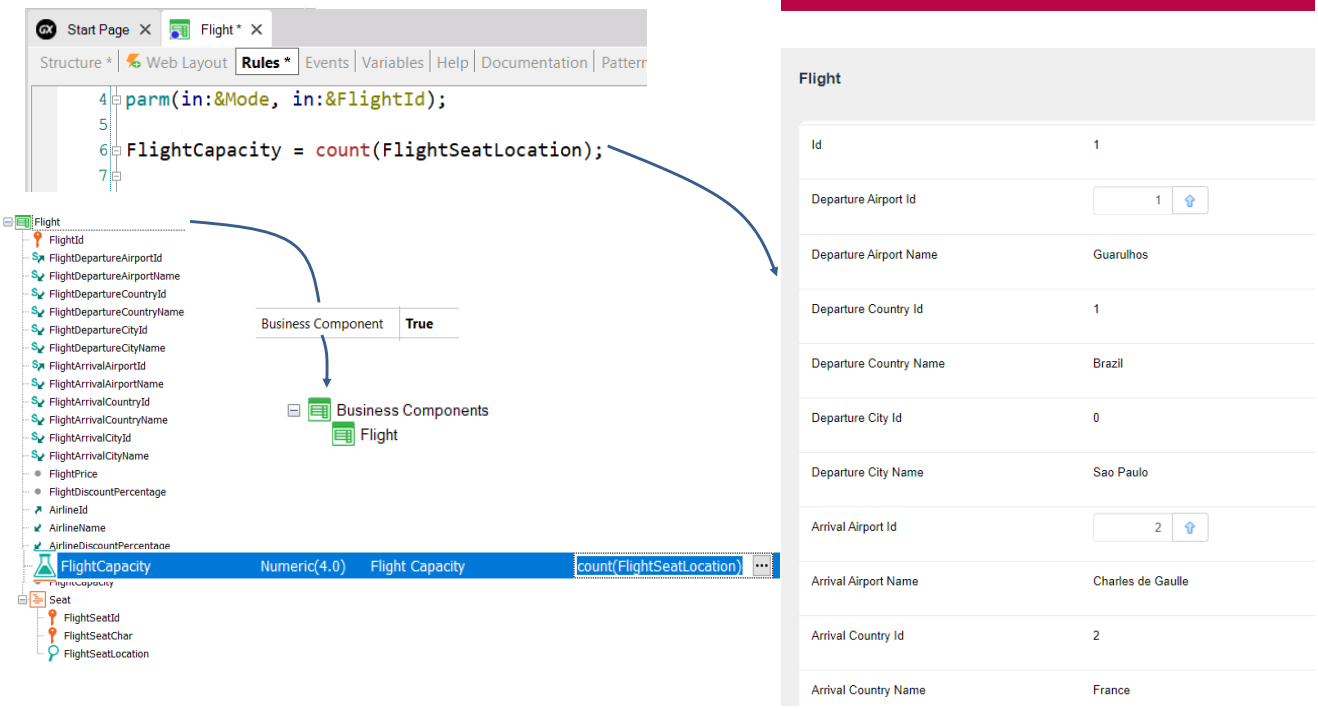
Vs.



Cuando el valor de un atributo puede obtenerse mediante un cálculo, por lo general lo definimos como fórmula en la estructura de la transacción. Sin embargo, notemos que también podríamos asignar el cálculo al atributo mediante una regla.

¿Qué consideraciones debemos tomar en cuenta para decidir si asignamos el cálculo mediante una regla o si lo hacemos definiendo el atributo como fórmula? Ya vimos anteriormente que si el atributo es fórmula, GeneXus no crea en su tabla asociada (es decir la tabla a la que pertenecería el atributo fórmula si estuviera almacenado), un campo para almacenar el valor, ya que entiende que su valor puede obtenerse a partir del cálculo. Debido a esto, decimos que consideramos al atributo como "virtual" ya que sigue presente en la estructura de la transacción, pero no en la tabla asociada.
Vemos que en la definición de la tabla aparece como atributo "lógico".

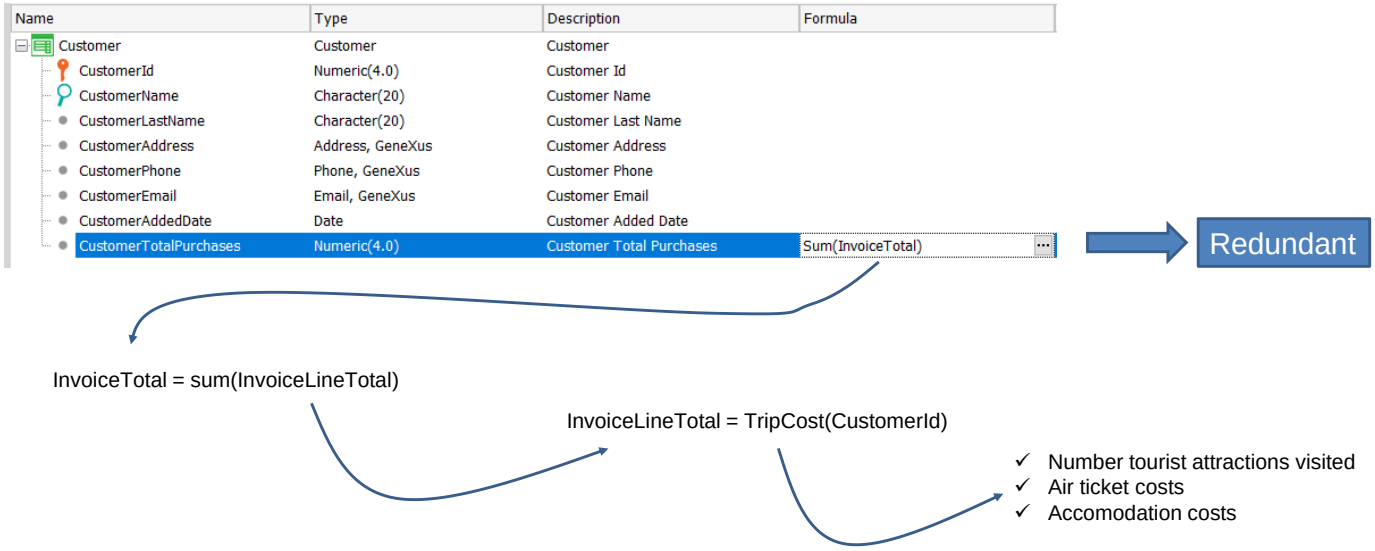
En cambio, si el valor del atributo lo hubiéramos asignado mediante una regla, no deja de estar presente en la tabla por el mero hecho de asignarle un valor, por lo que no se convierte en un atributo virtual, sino que sigue siendo un atributo almacenado.



¿Entonces es mejor usar una regla de asignación que una fórmula? No necesariamente, debemos tomar en cuenta que esa regla al se disparará únicamente cuando se ejecuta el objeto transacción o se utiliza un business component de la transacción. Por lo tanto, la definición del atributo mediante una regla es una definición de alcance **local** y esto implica que el valor del cálculo no se actualizará si se utiliza el atributo en cualquier otro objeto de la base de conocimiento y si se consulta su valor, se obtendrá el valor almacenado en la tabla en el último cálculo realizado.

Si ese cálculo debe mantenerse actualizado porque el atributo es utilizado frecuentemente en otros objetos, para ser mostrado en pantallas o listados, o bien como base de otros cálculos, entonces deberíamos definir el atributo como fórmula en la estructura de la transacción. Esta definición es de alcance **global** a la base de conocimiento y cada vez que se requiera el valor del atributo, el mismo se recalculará en el momento.

Consideraciones al definir un atributo en base a un cálculo



Sin embargo, que el atributo deba calcularse cada vez, no siempre es una ventaja. Como un atributo fórmula puede a su vez ser calculado a partir de otros atributos fórmula, puede pasar que la cantidad de cálculos que deban realizarse para obtener el atributo actualizado, atente contra la performance de la aplicación, en contextos donde se requiere repuestas en tiempo real.

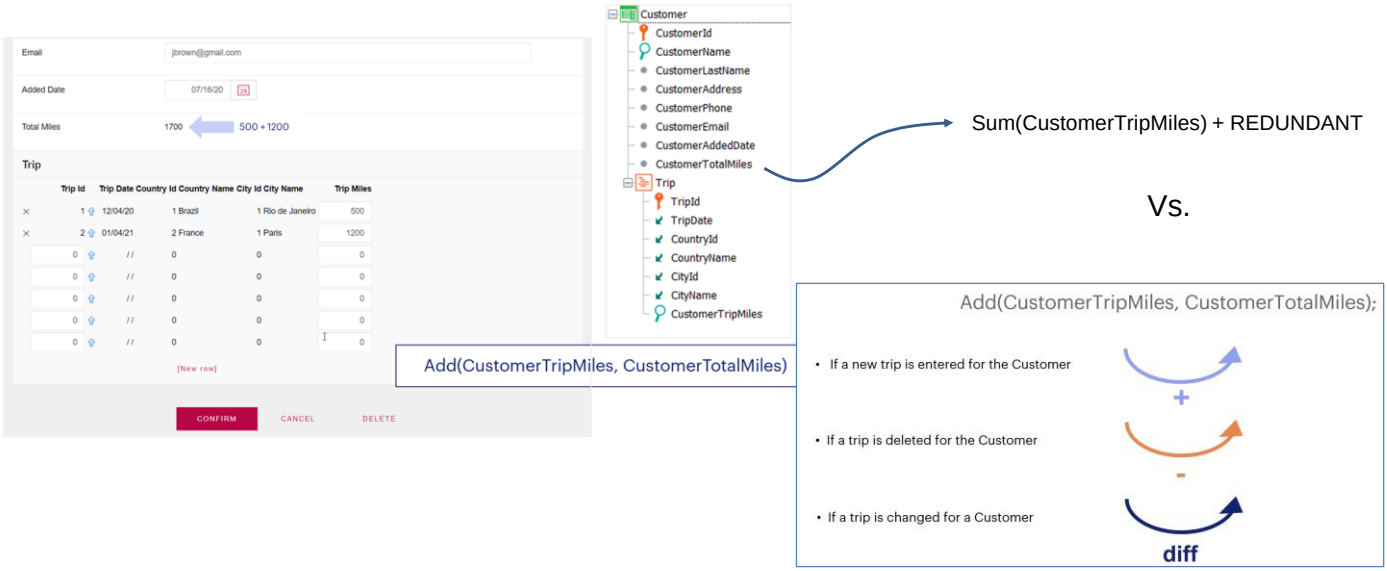
Supongamos por ejemplo que estamos calculando el total de compras de un cliente de la agencia de viajes, como la suma de las facturas realizadas al cliente y el total de cada factura se calcula como la suma de sus viajes y a su vez el costo de cada viaje es calculado mediante un procedimiento que toma en cuenta la cantidad de atracciones visitadas, el costo de los pasajes aéreos, el costo de los alojamientos, etc.
Si se desea listar el total de compras de todos los clientes de la agencia en los últimos 5 años, seguramente deban recorrerse muchos registros y realizarse muchos cálculos, que pueden afectar el tiempo de respuesta del sistema para obtener el resultado.

Ante esta situación, sería bueno que el resultado del total de compras del cliente se almacene en una tabla, de modo que si cambia algo que afecte el resultado se recalcule nuevamente y se almacene el nuevo resultado, pero si no cambia nada, se pueda utilizar el valor almacenado en lugar de que se efectúe siempre el cálculo.
Esto lo hacemos definiendo al atributo fórmula como redundante, es decir, que si bien puede obtenerse su valor mediante un cálculo, el resultado del mismo se almacenará en la base de datos y el valor será recuperado desde allí en el futuro.

También por razones similares, podemos definir a un atributo inferido como redundante.

La definición de atributos redundantes lo veremos más adelante en otro video.

Actualización de un atributo mediante una fórmula o una regla



Vimos las posibles formas de definir un atributo como fórmula o como regla, pero también debemos considerar que la actualización de un atributo también puede realizarse mediante una fórmula o una regla.

Recordemos el uso de la regla Add (o Subtract), que nos permitía mantener actualizado el valor de un atributo de la tabla extendida, realizando la operación adecuada dependiendo de si se estaba insertando, eliminando o modificando un registro.

El atributo del ejemplo CustomerTotalMiles, actualizado por la regla Add, es un atributo almacenado y por lo tanto la recuperación del valor es inmediata.
Sin embargo, debido a que la regla Add es local a la transacción Customer, como vimos antes, solamente se actualizará el atributo CustomerTotalMiles si se ejecuta la pantalla de la transacción Customer o un business component de la transacción.

Si queremos que el valor del atributo del total de millas del cliente se mantenga siempre actualizado, deberíamos definirlo como fórmula, en este caso una fórmula Sum que sume las millas de cada viaje al total de millas del cliente.

Si bien al definir el atributo como fórmula aseguramos su constante actualización, perdemos la posibilidad de que esté almacenado y para obtener su valor podemos incurrir en los problemas de performance que mencionamos. Para solucionarlo, podríamos definir al atributo fórmula CustomerTotalMiles como redundante y pasará a ser un atributo

almacenado. ¿Pero cuándo se actualiza el valor del atributo en la tabla?

Actualización de un atributo mediante una fórmula o una regla (continuación)

Cuando definimos un atributo fórmula como redundante, GeneXus crea automáticamente procedimientos encargados de actualizar su valor y almacenarlo en la base de datos.

Cuando se inserta, modifica o elimina registros mediante la ejecución de una transacción que contiene un atributo definido como fórmula redundante, la fórmula se dispara efectuándose el cálculo y el resultado se almacena en el campo físico de la base de datos.
Esto también es válido cuando las operaciones de CRUD se realizan mediante el Business Component de la transacción.

Cuando cambia el valor de algunos de los atributos que forman parte del cálculo de la fórmula, GeneXus también disparará los procedimientos de actualización del atributo redundante. Estos procedimientos tienen el conocimiento de cómo calcular el nuevo valor y el resultado será almacenado en la base de datos.

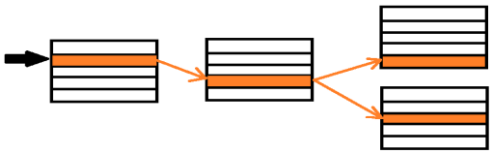
Por lo tanto, desde el punto de vista de la actualización y que el atributo esté almacenado, actualizar el atributo mediante una regla ADD y definirlo como fórmula redundante son **soluciones equivalentes**.

Fórmulas horizontales

Recordemos lo que vimos de las fórmulas horizontales

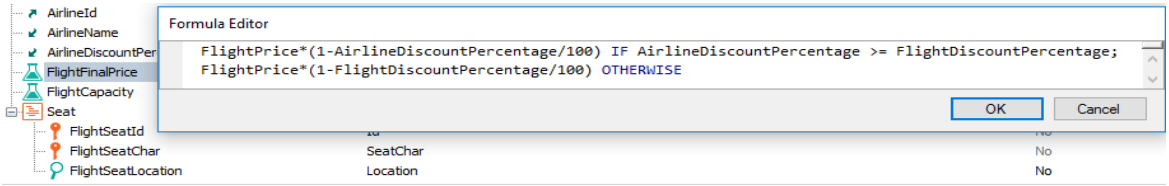
Repaso de fórmulas horizontales

- Permiten definir una o varias expresiones condicionales
- Se llaman horizontales porque acceden a un único registro, involucrando atributos de la tabla extendida.
- Siempre tienen un contexto



Attributo = $expresión_1$ **if condición**$_1$**;**
$expresión_2$ **if condición**$_2$**;**
…
$expresión_n$ **if condición**$_n$**;**
$expresión_o$ **otherwise;**

- Ejemplo:



Las fórmulas horizontales nos permiten definir expresiones aritméticas o de otro tipo, que incluyen condiciones que son cualquier expresión lógica válida.
Al atributo que definimos como fórmula le podemos asignar varias expresiones con condiciones excluyentes y se ejecutará aquella para la que se cumpla la condición. También podemos con "otherwise" asignar una expresión por defecto, que se ejecutará si no se cumple ninguna de las otras condiciones definidas.

Los atributos involucrados en la definición de la fórmula deberán pertenecer a la tabla extendida de la tabla asociada al atributo fórmula.

Estas fórmulas se llaman horizontales porque en su ejecución acceden a un único registro de la tabla base y eventualmente a aquellos relacionados pertenecientes a la tabla extendida.

Más sobre fórmulas horizontales

Attributo =

expresión$_1$ **if condición**$_1$**;**
expresión$_2$ **if condición**$_2$**;**
…
expresión$_n$ **if condición**$_n$**;**
expresión$_o$ **otherwise;**

Qué podemos incluir:

Expresión:

- Atributos (de la tabla extendida de la tabla asociada al atributo fórmula)
- Constantes, Funciones y Operadores (aritméticos, de strings y de fechas)
- Invocación a un procedimiento que devuelva un valor

Condición:

- Atributos (de la tabla extendida de la tabla asociada al atributo fórmula)
- Constantes, Funciones, Operadores lógicos: OR, AND, NOT y relacionales: (>, >=, <, <=, =, like).

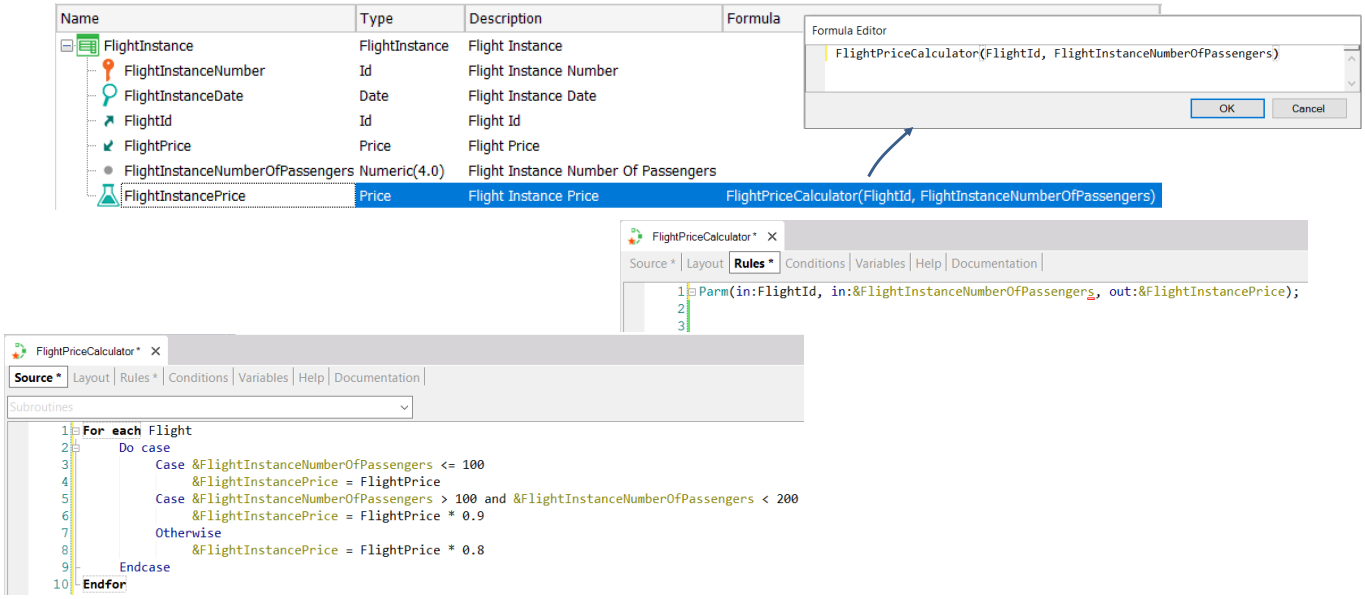
Vemos aquí la sintaxis general de la fórmula.

Las *expresiones* pueden contener **atributos pertenecientes a la tabla extendida de la tabla asociada al atributo que se está definiendo como fórmula,** constantes, funciones y operadores aritméticos (como la suma, resta, multiplicación, división, o potencia), operadores de strings (como el +) y operadores de fechas (como el + y el -). También pueden contener una invocación a un objeto procedimiento que retornen un valor. El resultado, ya sea de las expresiones o el retornado por el procedimiento invocado, deberá ser del mismo tipo de datos que el del atributo que se está definiendo como fórmula.

Las *condiciones* son cualquier expresión lógica válida, pudiendo contener **atributos pertenecientes a la tabla extendida de la tabla asociada al atributo que se está definiendo como fórmula,** constantes, funciones y operadores lógicos (and, or, not) y operadores de comparación (como >, >=, <, <=, = y like). La primera condición que al ser evaluada sea True, provocará que el resultado de la fórmula sea el de la expresión de la izquierda de esa condición y las demás no se seguirán evaluando.

Más sobre fórmulas horizontales

Invocación a un objeto procedimiento:



Como dijimos, las fórmulas horizontales pueden incluir en sus expresiones la invocación a un **objeto procedimiento** que devuelva un valor, del tipo de datos del atributo fórmula.

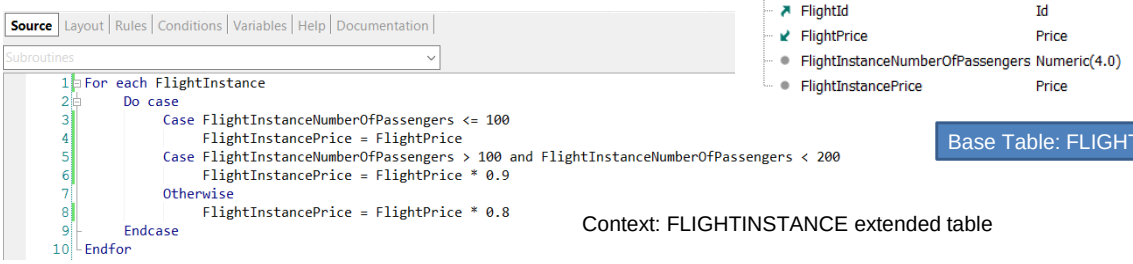
Por ejemplo, consideremos al atributo FlightInstancePrice, perteneciente a la transacción FlightInstance, que podría definirse como un atributo fórmula que invoque a un procedimiento que devuelva el valor del precio de la instancia del vuelo, calculando los descuentos correspondientes en función del precio del vuelo y de la cantidad de pasajeros.

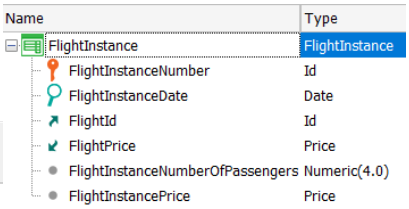
En la regla Parm del procedimiento vemos los parámetros recibidos: FlightId y &FlightInstanceNumberOfPassengers y el valor devuelto en la variable &FlightInstancePrice. En el source, podemos ver cómo se realiza el cálculo del precio.

Vemos que la definición de la fórmula contiene la invocación al procedimiento FlightPriceCalculator, pasándole como parámetros al identificador del vuelo y la cantidad de pasajeros. Notemos que en este caso la expresión de la fórmula horizontal sólo contiene la llamada al procedimiento, pero podría incluir otras expresiones y condiciones.

Las fórmulas horizontales deben tener un contexto

Fórmulas horizontales inline

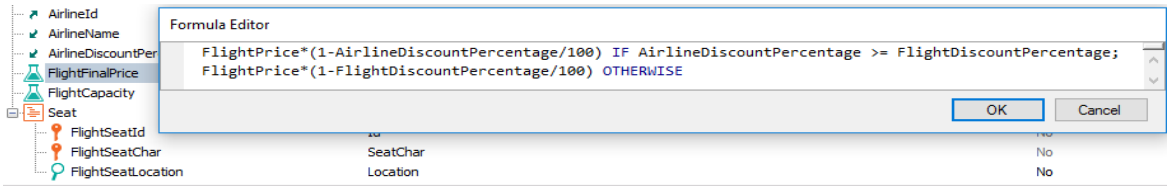




Base Table: FLIGHTINSTANCE

Context: FLIGHTINSTANCE extended table

Context: FLIGHT extended table



Las fórmulas horizontales pueden ser globales como vimos, pero también inline. Por ejemplo, podemos definir una fórmula como parte del código de un procedimiento, como vemos aquí. El atributo que recibe el cálculo, FlightInstancePrice no es un atributo fórmula, sino que se le asigna el cálculo únicamente en el source del procedimiento.

Las fórmulas horizontales, tanto globales como inline, para que tengan un sentido, siempre tienen un **contexto**.

En el caso de las fórmulas horizontales inline, el contexto es dado por el lugar donde fue definida la fórmula, por ejemplo dentro de un comando For Each. En este caso el contexto sería la tabla base del For each (FlightInstance) y su tabla extendida.

En el caso de las fórmulas globales, el contexto está dado por la transacción donde fue definida la fórmula, concretamente la tabla extendida de su tabla base. En el ejemplo que vimos, mostrado abajo, es la tabla extendida de la tabla Flight.

El contexto asegura que puedan instanciarse todos los atributos que forman parte de la fórmula.

Fórmulas de agregación

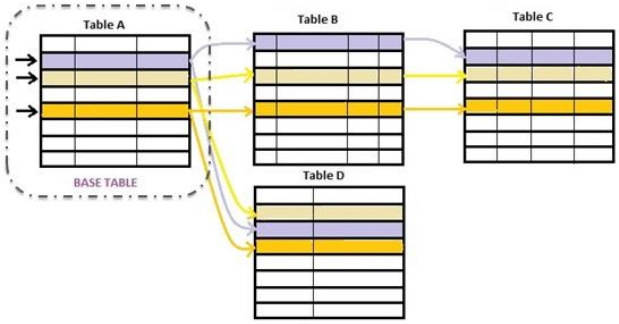
Repasemos algunos conceptos de las fórmulas aggregate.

Repaso de fórmulas aggregate

- Realizan cálculos y búsquedas sobre muchos registros, en cualquier tabla del modelo y su tabla extendida
- No necesitan un contexto como las fórmulas horizontales

Fórmulas aggregate:

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



Sintaxis:

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

opcional opcional

Las fórmulas aggregate permiten realizar cálculos o búsquedas que involucran muchos registros de una tabla y valores relacionados de la tabla extendida de la tabla recorrida.

Estas fórmulas nos permiten contar (con Count), sumar (con Sum) o hacer el promedio (con Average) de varios registros, realizar búsquedas como por ejemplo encontrar el valor máximo (con Max) o mínimo (con Min) de un atributo en un conjunto de registros, o dado muchos registros de una tabla, encontrar un valor de un atributo (usando Find) cuyo registro sea el primero encontrado que cumpla determinadas condiciones.

En la sintaxis, la **expresión de agregación** es la expresión que será buscada, maximizada, minimizada, sumada o promediada, entre los registros que cumplen la condición de agregación. Puede contener atributos (incluso atributos fórmula), constantes y variables (las variables creadas por el usuario, sólo se permiten en fórmulas inline). Solo para el caso Count, no es una expresión sino un atributo. Para Sum y Average, el resultado de la expresión de agregación debe ser un valor numérico.

La condición de agregación, es la condición que los registros deben verificar para ser considerados en la agregación. Puede contener atributos, constantes y variables (las de usuario sólo en fórmulas inline). Es opcional y puede no incluirse.

El valor por defecto, es el valor devuelto cuando ningún registro coincide con la condición de agregación. Es una constante y también es opcional.

Repaso de fórmulas aggregate (continuación)

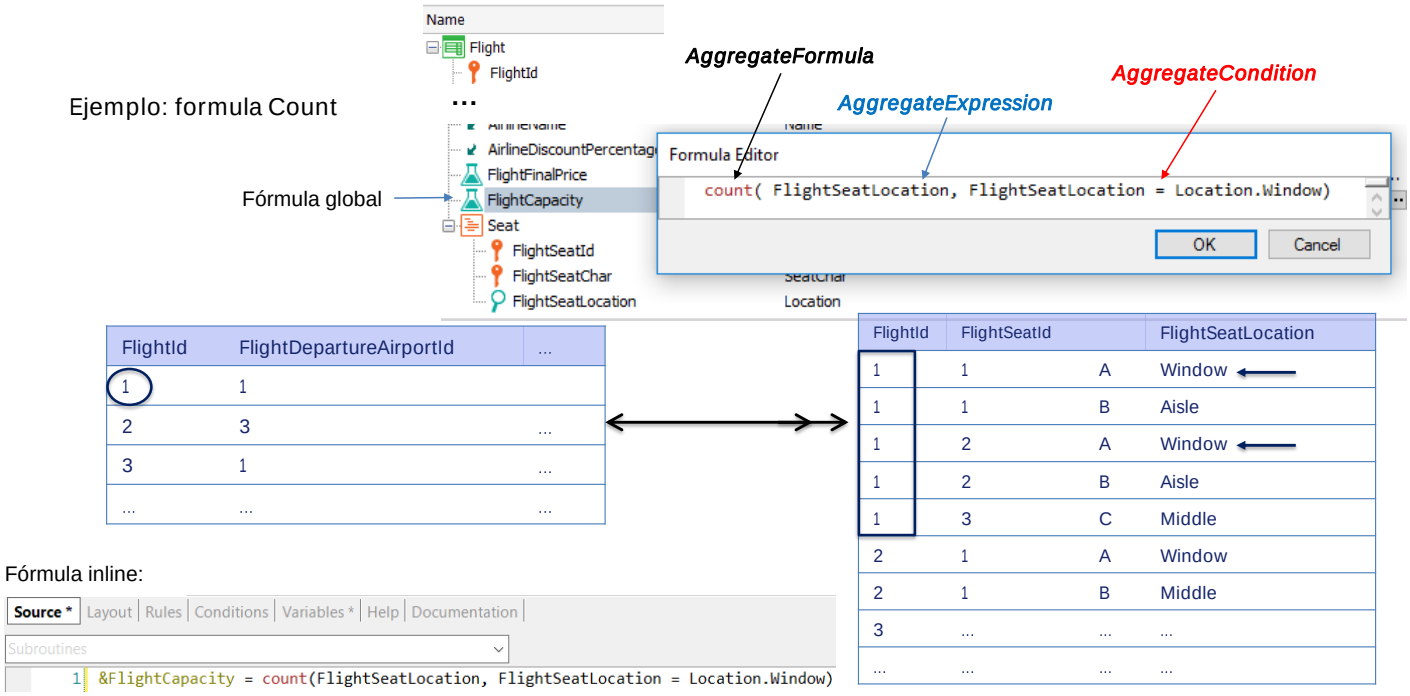
El valor retornado, es el atributo cuyo valor es devuelto por la fórmula cuando encuentra registros que cumplen la condición de agregación. Su inclusión es opcional.

La condición de disparo, es la condición que determina si la fórmula debe dispararse o no. Es opcional y los únicos atributos permitidos son los que pertenecen a la tabla asociada y a su extendida. Las condiciones de disparo solamente se pueden utilizar en fórmulas aggregate globales ya que en las fórmulas inline no forma parte de la definición de la fórmula, sino que su disparo se condiciona en el código mediante cláusulas condicionales (por ejemplo if, else, do case, etc.) .

Mientras que una fórmula horizontal necesita un contexto para ser evaluada, una fórmula aggregate no necesariamente lo necesita. Sin embargo, si existe una tabla de contexto, no se considerarán todos los registros que la fórmula establece explícitamente, sino solo aquellos que también coincidan con las condiciones implícitas que surgen de ese contexto.

Repaso de fórmulas aggregate

Ejemplo: formula Count



FlightId	FlightDepartureAirportId	...
1	1	
2	3	...
3	1	...
...	...	...

FlightId	FlightSeatId		FlightSeatLocation
1	1	A	Window
1	1	B	Aisle
1	2	A	Window
1	2	B	Aisle
1	3	C	Middle
2	1	A	Window
2	1	B	Middle
3	...	...	...
...	...	...	...

Fórmula inline:

Por ejemplo el atributo FlightCapacity de la transacción Flight, es una fórmula count que recorre la tabla FLIGHTSEAT y cuenta la cantidad de asientos del vuelo.

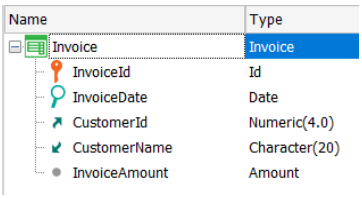
Como la tabla asociada al atributo es la tabla, Flight, que tiene una relación de 1 a N con la tabla FLIGHTSEAT navegada por el Count, se contarán únicamente los registros relacionados, es decir los asientos correspondientes al vuelo instanciado en Flight. Si no hubiera relación, se contarían todos los registros del tabla FLIGHTSEAT.

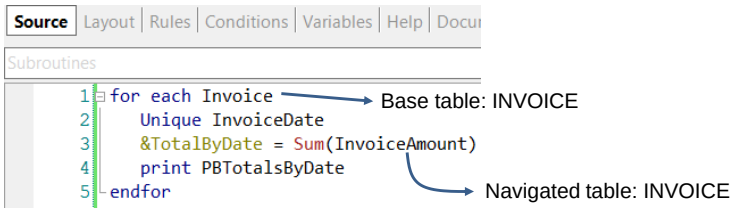
Además como indicamos que solamente se cuenten los asientos que sean ventana, del conjunto de los registros relacionados se contarán únicamente los que cumplan la condición de filtro.

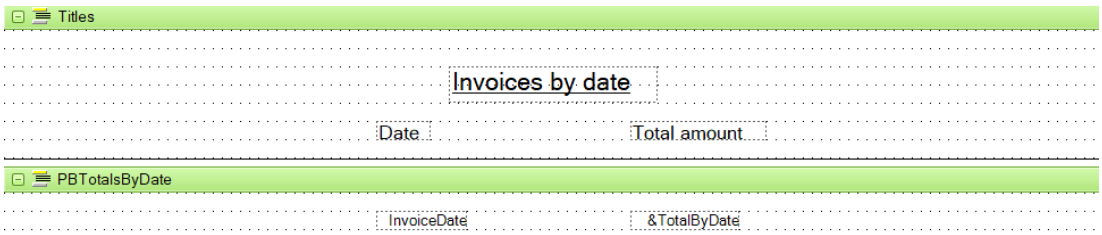
Vemos que como es una fórmula Count, la expresión de agregación es únicamente un atributo que determina la tabla donde se contarán los registros, en este caso FlightSeatLocation, la condición de agregación es el filtro que deben cumplir los asientos (que sean ventana), como no es una fórmula Max o Min no tiene valor por defecto, ni valor retornado y tampoco hay condición de disparo definida.

Aquí vemos a la misma definición como fórmula inline en el source de un procedimiento. En este caso la fórmula no está dentro de un For Each por lo que no será filtrado su resultado en el caso de que hubiera relación entre la tabla base del For Each y la tabla FLIGHTSEAT navegada por la fórmula.

Ejemplo : tabla asociada = tabla navegada







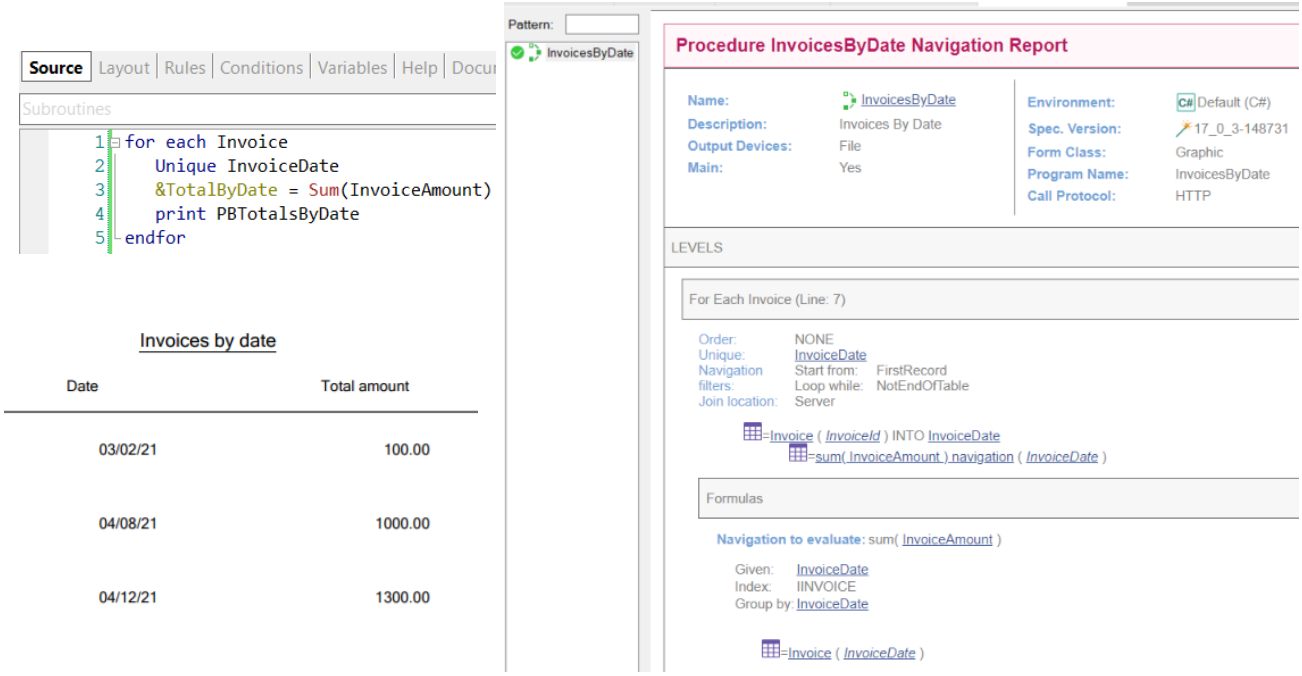
Analicemos un caso particular que es cuando la fórmula está en un cierto contexto y la tabla navegada por la fórmula coincide con esa tabla de contexto.

En este ejemplo queremos imprimir para cada fecha de factura, la suma total de todas las facturas con esa fecha de factura.

Vemos que la tabla navegada por la fórmula Sum es Invoice, y coincide con la tabla base del For Each, que también es Invoice.

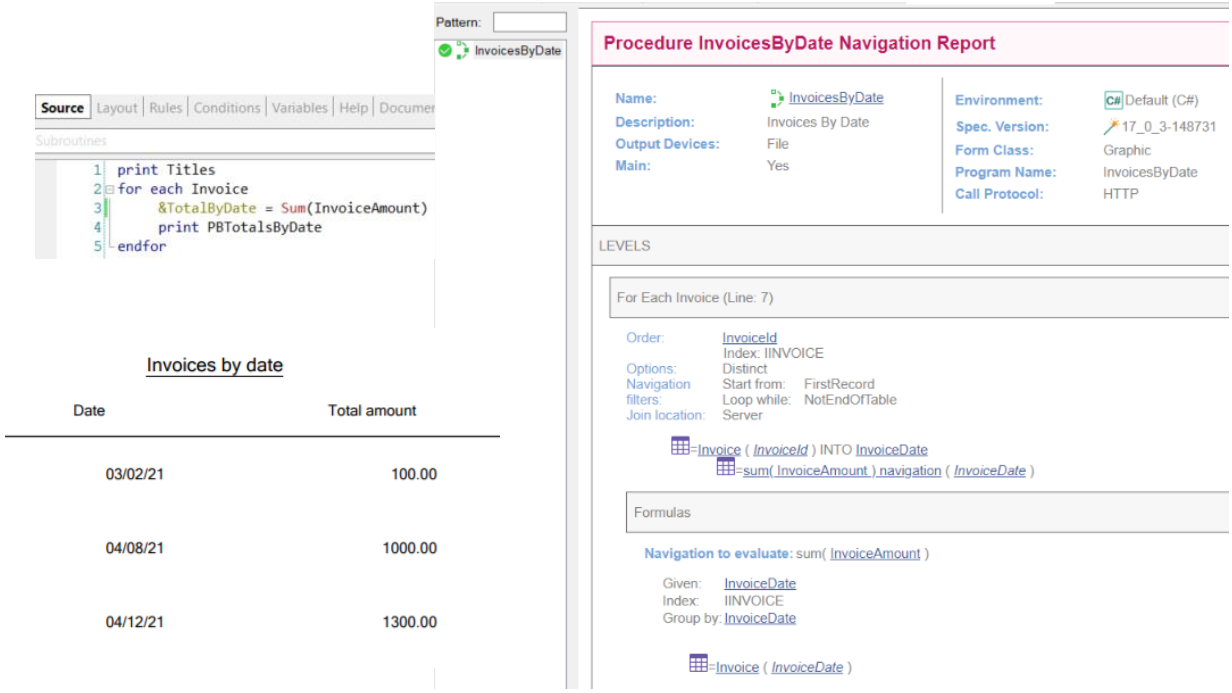
En este caso, debido a la cláusula Unique por InvoiceDate, GeneXus agrupará la información por la fecha de la factura, tanto en el For Each como en la fórmula Sum. Es decir, la fórmula Sum tendrá una condición implícita por igualdad por parte del atributo InvoiceDate, que se considerará dado.

Es como si fuera un corte de control, cortando por InvoiceDate. Veamos el listado de navegación.



Comprobamos que la tabla base del for each es Invoice y que el Unique es por InvoiceDate. Vemos que la tabla accedida es Invoice y que dice INTO InvoiceDate, lo que nos indica que está agrupando por lo que especifica la sentencia Unique.

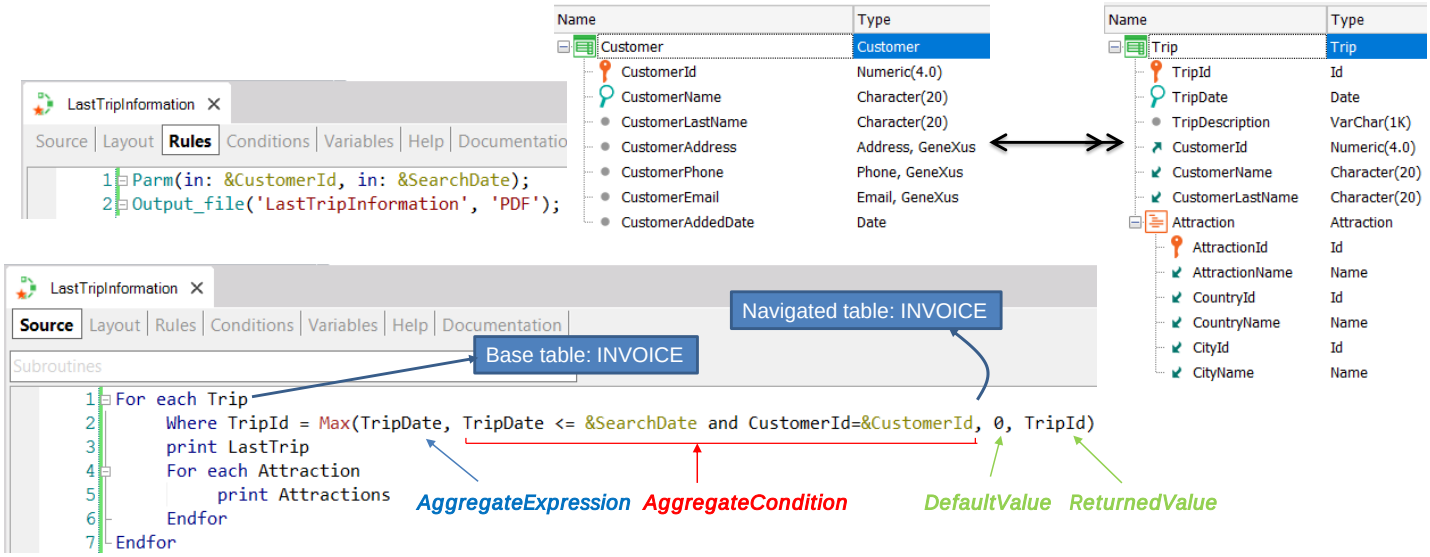
Más abajo vemos la fórmula que también está agrupando por InvoiceDate (observemos el Group by) y que además InvoiceDate es Given.



¿Qué pasaría si quitamos la cláusula Unique? Veamos el listado de navegación de este caso.
Observemos que no aparece más la cláusula Unique, pero GeneXus agrega automáticamente una cláusula DISTINCT, por lo que el resultado sigue siendo el mismo. GeneXus tuvo la inteligencia de detectar que se deseaba agrupar por fecha de factura y reflejó eso en la generación del código, haciendo que el listado funcionara como nosotros queríamos.

Si bien GeneXus, dependiendo del caso, es capaz de detectar automáticamente lo que desea hacer el desarrollador, lo correcto es que seamos nosotros los que incluyamos la cláusula Unique, para que quede clara nuestra intención en la programación del código.

Ejemplo 2: tabla asociada = tabla navegada



Veamos ahora un ejemplo de uso de la fórmula Max. 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.

Supongamos que se desea que la información salga en un listado que imprima los datos del viaje. El procedimiento recibe por parámetro el identificador del cliente que desea la información y la fecha de búsqueda.

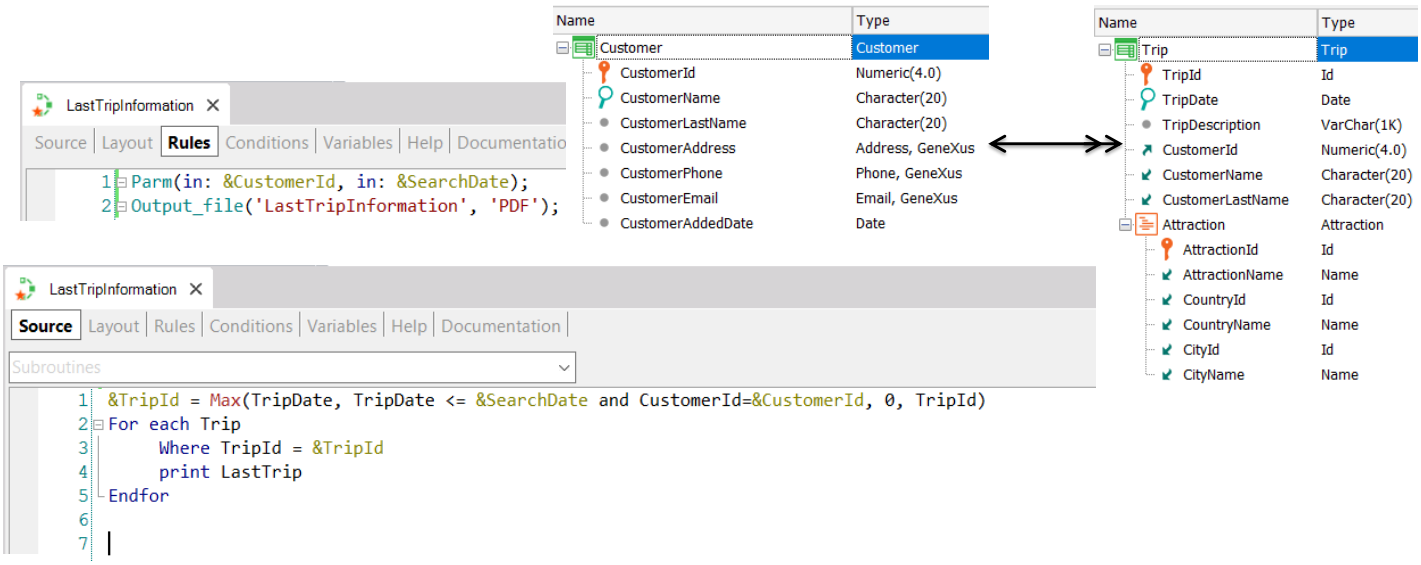
En el source del procedimiento hay un for each que recorre la tabla Trip y el where filtrará por el TripId devuelto por la fórmula Max. Luego se imprimirán los datos correspondientes a ese viaje.

En la fórmula Max, el atributo a maximizar es TripDate, ya que queremos el viaje que haya sido inmediatamente anterior a una fecha dada, por lo tanto será el viaje con la mayor fecha posible, pero menor o igual a la fecha de búsqueda.
La condición de agregación especifica que el viaje debe tener una fecha menor o igual a la fecha dada y que además sea del cliente que busca la información. El valor por defecto devuelto si no se encuentra ningún viaje que cumpla con esas condiciones es el 0 y en caso de que se encuentre será el identificador del viaje encontrado.

Ahora analicemos lo que hemos programado. El For Each iterará en 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, debido a 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.

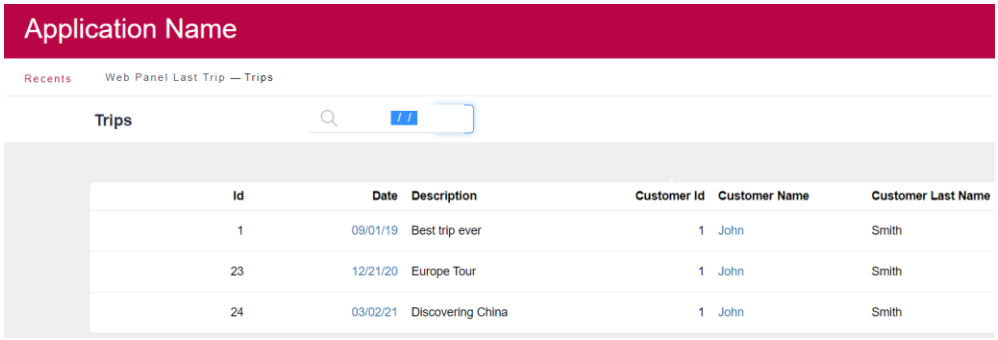
Entonces, no tiene sentido aquí la fórmula porque no podrá encontrar de entre todos los viajes, aquel cuya fecha sea la mayor posible, pero que no sobrepase la fecha de búsqueda, ya que solamente podrá itererar sobre un único registro.

Ejemplo 2: tabla asociada = tabla navegada

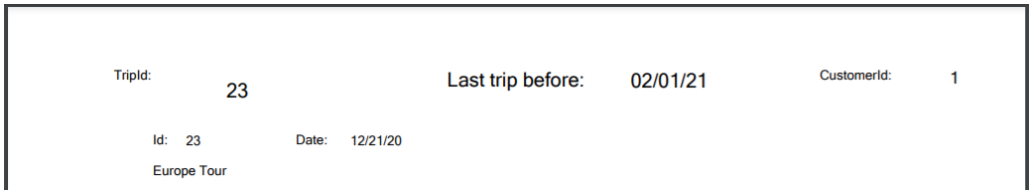


La solución para esto es ejecutar primero la fórmula para busque el identificador del viaje que cumple las condiciones deseadas y luego filtramos al For Each por ese valor de identificador.

Ejemplo 2: tabla asociada = tabla navegada (cont.)



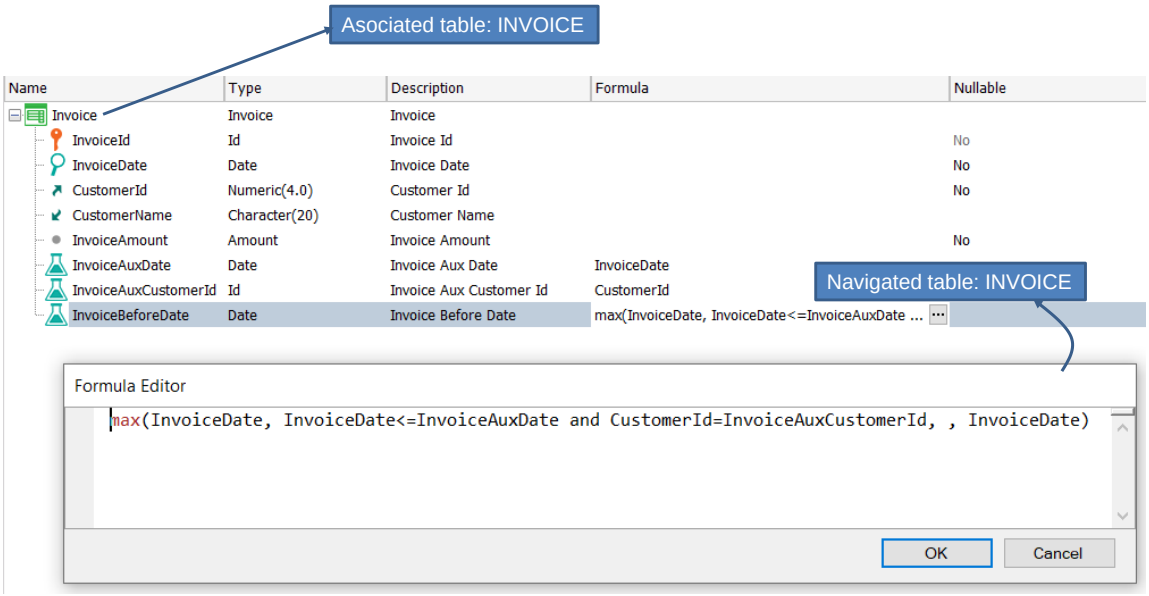
CustomerId=1
&SearchDate=01/02/21



De los 3 viajes que tenía realizados John Smith, el inmediatamente anterior a la fecha de búsqueda (el 1° de febrero del 2021) fue el Tour a Europa, realizado el 21 de diciembre del 2020.

Por lo que vemos que la fórmula Max funcionó como deseábamos

Ejemplo 3: tabla asociada = tabla navegada



Ahora veamos un caso similar al anterior en una fórmula global 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 similar a la que definimos antes, pero definiendo un atributo como fórmula global.

Para eso agregamos a la transacción Invoice un atributo InvoiceBeforeDate definido como Max. 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.

Luego definimos la fórmula Max de la siguiente manera.

Nuestra intención es que al momento de dispararse la fórmula Max correspondiente al atributo InvoiceBeforeDate, por referenciar ésta a los atributos fórmula InvoiceAuxCustomerId y InvoiceAuxDate, sus fórmulas horizontales se disparen y en los atributos auxiliares InvoiceAuxCustomerId y InvoiceAuxDate queden los valores de los atributos CustomerId e InvoiceDate correspondientes al registro de la tabla asociada y estos valores se comparan con los valores de los atributos CustomerId e InvoiceDate de cada uno de los registros de la tabla navegada por la fórmula (la tabla Invoice).

Ejemplo 3: tabla asociada = tabla navegada (continuación)

De esa forma deseamos lograr una búsqueda en la misma tabla asociada a la fórmula que queremos definir.
Sin embargo, estamos en el mismo caso anterior, ya que esta fórmula Max, al estar definida en la transacción Invoice, la tabla asociada al atributo fórmula será INVOICE y coincidirá con la tabla navegada que también es INVOICE..

Por lo tanto, se filtrará por el identificador del registro donde se esté posicionando y no se podrá maximizar el valor de la fecha entre todas las facturas.

Debido a este filtro implícito, el atributo InvoiceBeforeDate tendrá como resultado la misma fecha de la factura dada.

En resumen, esto verifica lo que dijimos antes, de que las fórmulas aggregate si bien no necesitan un contexto para ser evaluadas (como sí lo necesitan las horizontales), en caso de que ese contexto exista, no se considerarán todos los registros que la fórmula establece explícitamente, sino solo aquellos que también coincidan con las condiciones implícitas que surgen de ese contexto.

Fórmulas compuestas

Veamos ahora qué son las fórmulas compuestas.

Fórmulas compuestas

Attributo = *expresión*$_1$ **if condición$_1$;**
expresión$_2$ **if condición$_2$;**
...
expresión$_n$ **if condición$_n$;**
expresión$_o$ **otherwise;**

Ejemplo:

Atributo = ***Count***(*Atributo, condición,* ***Valor*** *por defecto)* **if** condición;

Sum(*Expresión, condición,* ***Valor*** *por defecto*) **if** condición;

Find(*Expresión, condición,* ***Valor*** *por defecto*) **if** condición;

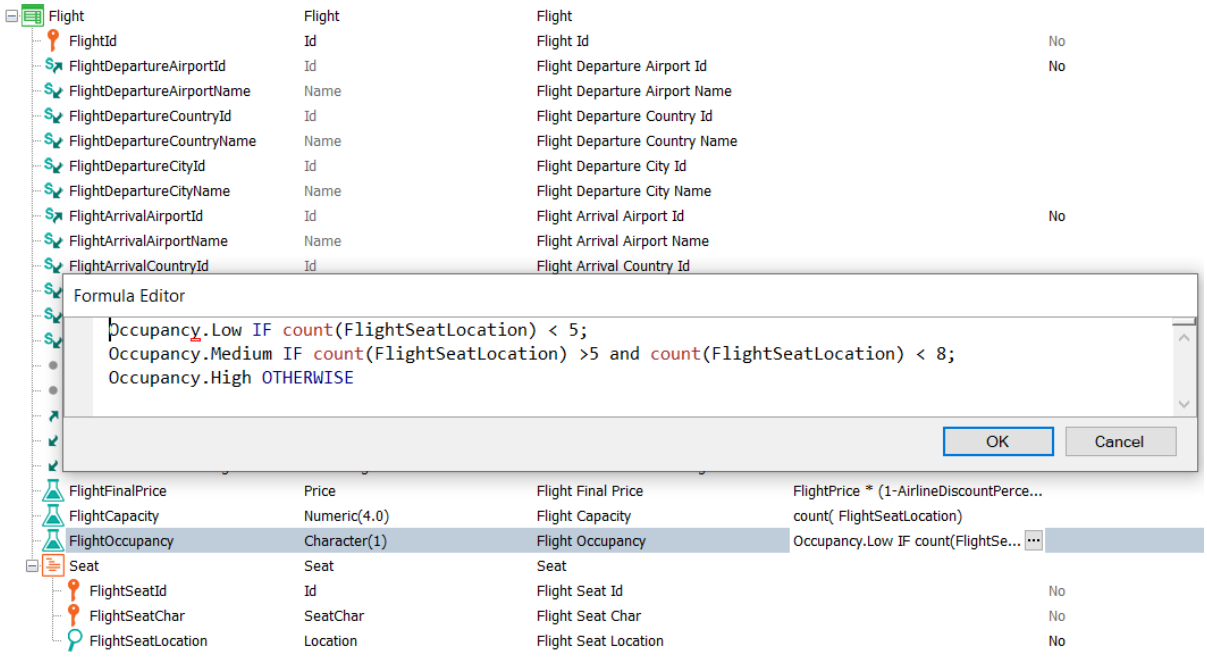
Las fórmulas compuestas son fórmulas que integran varias fórmulas aggregate condicionales, pudiendo también contener expresiones horizontales.

En este caso cada expresión puede ser una fórmula aggregate o una fórmula horizontal. Si todas las expresiones incluidas son fórmulas horizontales, entonces la fórmula definida no es compuesta sino horizontal.

Las condiciones son cualquier expresión lógica válida, pudiendo contener la misma atributos pertenecientes a la tabla extendida de la tabla asociada al atributo que se está definiendo como fórmula, constantes, funciones, operadores lógicos (and, or, not) y operadores relacionales (>, >=, <, <=, =, like). 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 (las demás no se seguirán evaluando).

Cuando ninguna de las condiciones evaluadas dan True, si existe una *expresión* con cláusula otherwise, el resultado de la fórmula será el de la expresión que anteceda a esta cláusula.

Ejemplo



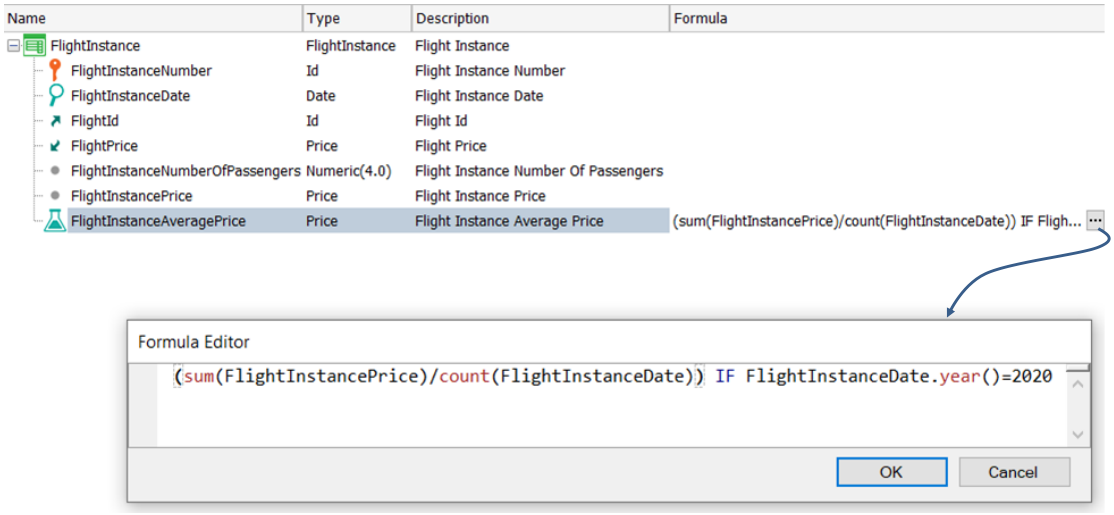
Veamos un ejemplo de este tipo de fórmulas compuestas en nuestra realidad de la agencia de viajes.

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

Otro ejemplo de fórmula compuesta



En este otro ejemplo necesitamos calcular el promedio del precio de la instancias del vuelo . Para esto, debemos sumar todos los precios de las instancias de los vuelos y dividirlos entre la cantidad de instancias.

Definimos al atributo FlightInstanceAveragePrice como al cociente de una fórmula aggregate sum, que suma el atributo FlightInstancePrice y el resultado lo divide entre la cantidad de instancias de vuelo calculado como una fórmula aggregate Count que utiliza al atributo FlightInstanceDate para contar las instancias de vuelo.

Ejemplos de otras fórmulas compuestas

```
Atributo =  Max(...)   if condicion1;
            (2 * atr_x) + 100   if condicion2;
            Sum(atr_y)    otherwise

Atributo = Find(...)    if condicion1;
           1        otherwise

Atributo = procedure(...)       if condicion1;
           Min(...)   if condicion2;
           10         if condicion3

Atributo = 2 + Count(Atributo, condición, Valor por defecto) *
               Sum(Expresión, condición, Valor por defecto) if condición;
               Atr1 + Atr2 * Atr3  otherwise;
```

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

En este video vimos la conveniencia de utilizar fórmulas para ahorrar código, con la facilidad que nos brinda la programación declarativa.

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