Fórmulas

Uma visão integradora

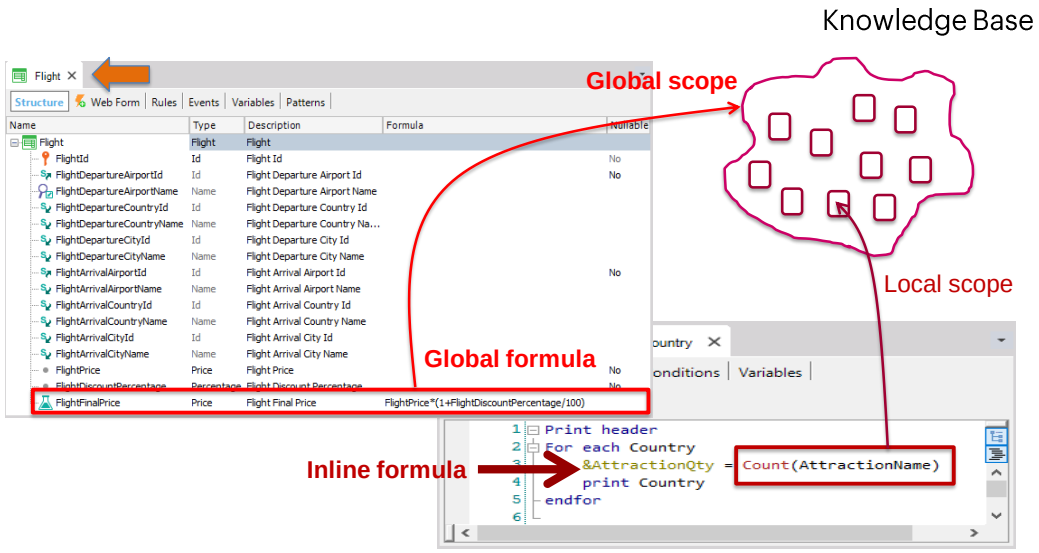
GeneXus™

Em outros vídeos explicamos o que são as fórmulas globais e as fórmulas inline, analisando os diferentes tipos de acordo com sua navegação, como as fórmulas horizontais, fórmulas de agregação, fórmulas compostas e seus respectivos casos de uso.

Neste vídeo vamos tentar dar uma visão mais em perspectiva e não tão em detalhe, para descobrir quando é conveniente usar as fórmulas de acordo com seu tipo e quando podemos usar uma solução alternativa, em quais casos temos restrições e como podemos ultrapassá-las, que custo implica adicionar redundância e outras observações que nos ajudem a integrar os conceitos sobre este tema.

Se tem alguma dúvida sobre os conceitos das fórmulas sobre os quais se baseia este vídeo, ou deseja relembrá-los antes de ver esta síntese, sugerimos que veja os vídeos de fórmulas do Curso GeneXus Avançado.

Global formulas vs. Inline formulas (local formulas)



Uma **fórmula global** também é conhecida como "atributo fórmula" e é um cálculo atribuído a um atributo na estrutura de uma transação.
São chamadas de globais porque quando é definido o cálculo em um atributo, sua definição fica no nível da base de conhecimento e, portanto, todos os objetos terão acesso ao atributo com o referido cálculo.

A partir do momento em que associamos um cálculo a um atributo, ele não será armazenado como campo de uma tabela na base de dados e por essa razão também é chamado de "atributo virtual". No entanto, o atributo fórmula global fica associado à tabela à qual pertenceria se não tivesse sido definido como fórmula. Esta tabela associada representa o contexto da fórmula, ou seja, no momento em que a fórmula é disparada, está posicionado em um determinado registro dessa tabela.

Somente os atributos podem ser definidos como fórmulas globais e as variáveis não, pois os atributos têm alcance global na base de conhecimento, enquanto o alcance das variáveis é restrito ao objeto onde foram definidas e não podem ser acessadas a partir de outro objeto.

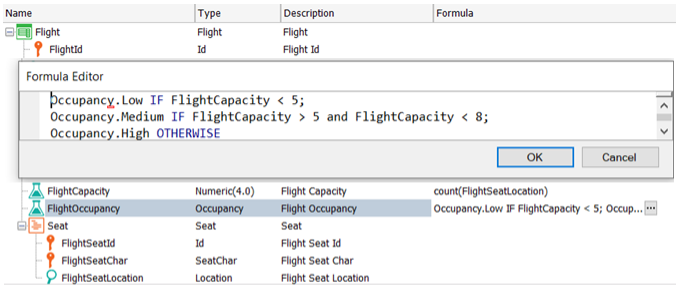
Se usamos uma fórmula em um cálculo que implementamos quando escrevemos código em um objeto, como por exemplo no source de um procedimento ou eventos de um panel ou web panel, ou em um data provider, ou na tab Conditions de um objeto, estamos definindo uma **fórmula inline**.

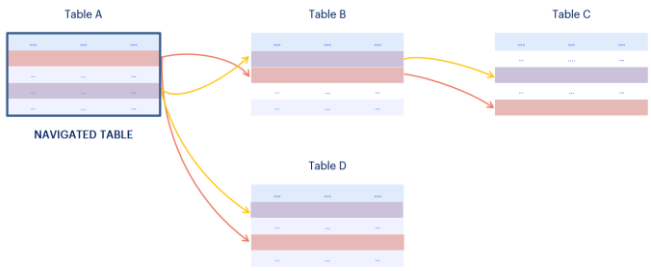
Como podemos definir fórmulas inline em qualquer objeto no qual possamos escrever código, na maioria das vezes atribuímos este cálculo a uma variável, pois somente poderíamos atribuir o valor retornado por uma fórmula a um atributo se estivéssemos atualizando a tabela por meio de um For Each em um objeto procedimento.

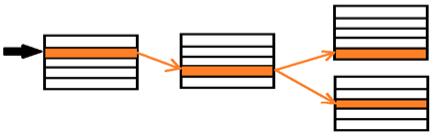
Como as variáveis têm alcance local para o objeto, as fórmulas inline são conhecidas também como fórmulas locais.

No caso dos data providers, os elementos do lado esquerdo das atribuições fazem parte de uma estrutura hierárquica que somente tem valor dentro do data provider, portanto seu alcance também é local. O mesmo acontece no caso em que são usadas fórmulas inline na tab Conditions de um 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

A navegação da fórmula determina se a fórmula é horizontal ou de agregação.

Quando temos uma fórmula que navega em um único registro de uma tabela (acessando eventualmente atributos de sua tabela estendida), falamos de uma **fórmula horizontal**.

Se, em vez disso, a fórmula acessa muitos registros de uma tabela, então é uma **fórmula de agregação**.

Portanto, se o que queremos é recuperar um valor a partir de um cálculo que pode ser obtido com dados de um único registro e os registros associados de sua tabela estendida, então escrevemos uma fórmula horizontal. Este tipo de fórmula nos permite atribuir diferentes valores dependendo de determinadas condições e também assumir um valor padrão caso não se cumpra nenhuma das condições estabelecidas.

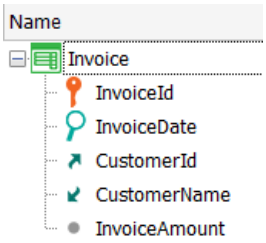
Se, em vez disso, quisermos um valor que depende da pesquisa e/ou processamento de múltiplos registros (e seus registros associados da tabela estendida), então utilizaremos algumas das fórmulas de agregação, como: count, sum, max, min, find ou average.

As fórmulas de agregação podem ser globais (atribuídas a atributos no nível da estrutura da transação, ou locais (inline), atribuídas a atributos ou variáveis, elementos de um data provider, ou sendo parte de filtros **(como tab conditions ou diretamente em for eachs, grupos de Data Providers, etc.)** que são avaliados em tempo de execução.

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.

Vejamos os parâmetros de uma fórmula de agregação e qual papel desempenha cada um.

As fórmulas de agregação têm um primeiro parâmetro obrigatório que é a expressão de agregação e que é um atributo que estabelece qual será a tabela percorrida pela fórmula. Recomenda-se que este atributo não seja chave, para evitar possíveis ambiguidades quando GeneXus determina a tabela a ser navegada, principalmente se nas demais partes da fórmula não aparecem outros atributos que permitam determinar com unicidade a tabela, por exemplo quando a fórmula não tem parâmetros adicionais ao primeiro.
Em particular, este atributo pode ser um atributo fórmula, não precisa ser um atributo armazenado, pois todo atributo fórmula possui uma tabela associada.

Além do primeiro parâmetro obrigatório, uma fórmula de agregação pode ter outros parâmetros, todos opcionais.

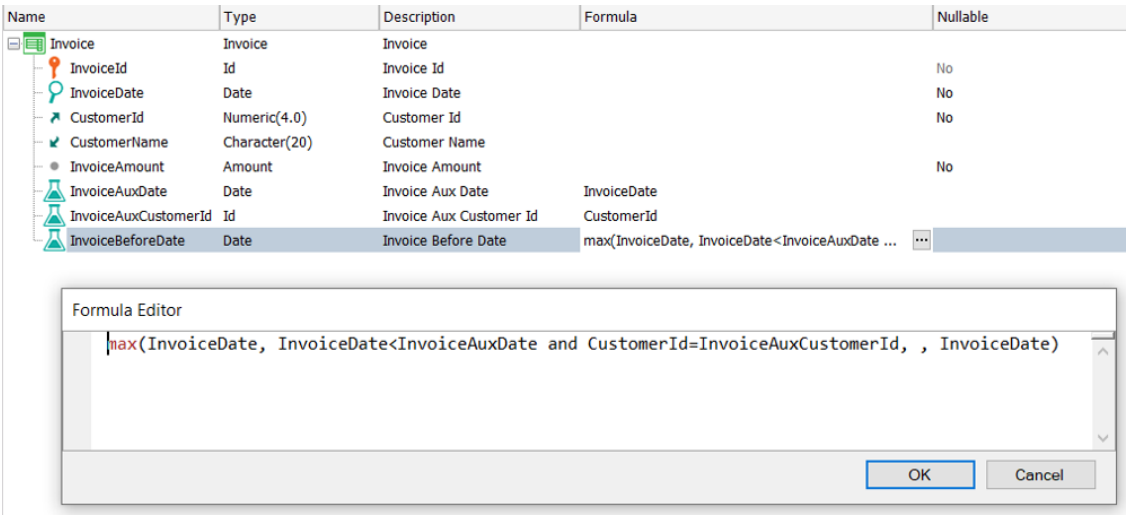
O primeiro após a expressão de agregação é a condição de agregação, ou seja, a condição que devem cumprir os registros que serão contados, somados, calculados a média, etc.

O próximo parâmetro é o valor padrão retornado pela fórmula se nenhum registro for encontrado que atenda à condição de agregação.

O último parâmetro será o atributo que contém o valor que deve ser retornado no caso em que tenham sido processados registros.

Após o if, a condição de disparo estabelece qual condição deve ser atendida para que a fórmula seja executada. Esta condição é uma expressão lógica tão complexa quanto seja necessário.

Restrictions on global aggregation formulas



Como a escolha do atributo da expressão de agregação é arbitrária, a fórmula pode, em princípio, navegar por qualquer tabela da base de dados. Portanto, poderíamos fazer com que a tabela a ser percorrida fosse a mesma que a tabela associada ao atributo fórmula.

Vamos analisar o que aconteceria neste caso.

Vejamos um caso em que, dada uma fatura de um cliente, é desejado encontrar a data da fatura anterior do mesmo cliente. Para resolver isto, utilizaremos uma fórmula Max que encontre a data de fatura máxima que seja menor ou igual à data de fatura dada.

Percebemos que devemos diferenciar os atributos que são do registro da tabela associada, dos atributos que são dos registros da tabela onde será realizada a busca.
Para isso adicionamos os seguintes atributos auxiliares: InvoiceAuxCustomerId definido como fórmula horizontal que assume o valor de CustomerId e o atributo InvoiceAuxDate que assume o valor de InvoiceDate.

Notemos que estamos tentando definir uma fórmula que navegará sobre a mesma tabela que está associada à fórmula. A fórmula está definida na transação Invice, portanto sua tabela associada é INVOICE, e como o atributo da expressão de agregação é InvoiceDate, a tabela navegada também é INVOICE.

Como vimos antes, a tabela associada representa o contexto da fórmula, portanto, quando a fórmula é disparada, está posicionado em um determinado registro dessa tabela. Portanto, a fórmula será filtrada pelo identificador do registro onde esteja posicionado e apenas navegará por um único registro!

Ou seja, que não poderemos ter uma fórmula aggregate global que navegue na mesma tabela onde está definida, pois não poderá percorrê-la como precisávamos neste caso para encontrar o máximo.

Se precisarmos desta funcionalidade, poderíamos definir a fórmula invocando um objeto procedimento que nos retorne o cálculo, como veremos a seguir.

Global formulas that call procedures

Name	Type	Description	Formula	Nullable
Invoice	Invoice	Invoice		
InvoiceId	Id	Invoice Id		No
InvoiceDate	Date	Invoice Date		No
CustomerId	Numeric(4.0)	Customer Id		No
CustomerName	Character(20)	Customer Name		
CustomerTotalPurchases	Amount	Customer Total Purchases		
InvoiceAmount	Amount	Invoice Amount		No
InvoiceBeforeDate	Date	Invoice Before Date	GetInvoiceBeforeDate(InvoiceId)	

GetInvoiceBeforeDate *

Source * | Layout | Rules | Conditions | Variables | Help | Documentation

```
1  Parm(in:&InvoiceId, out:&InvoiceBeforeDate);
2
```

GetInvoiceBeforeDate

Source | Layout | Rules | Conditions | Variables | Help | Documentation

'GetPreviousInvoiceDate'

```
1  For each Invoice
2       Where InvoiceId=&InvoiceId
3       &InvoiceDate = InvoiceDate
4       &CustomerId = CustomerId
5       Do 'GetPreviousInvoiceDate'
6  Endfor
7
8  Sub 'GetPreviousInvoiceDate'
9       For each Invoice order (InvoiceDate)
10           Where CustomerId = &CustomerId
11           Where InvoiceDate < &InvoiceDate
12           &InvoiceBeforeDate = InvoiceDate
13           Exit
14      Endfor
15 EndSub
```

Vamos criar um procedimento que recebe a fatura de um cliente e nos devolva a data da fatura anterior do mesmo cliente.

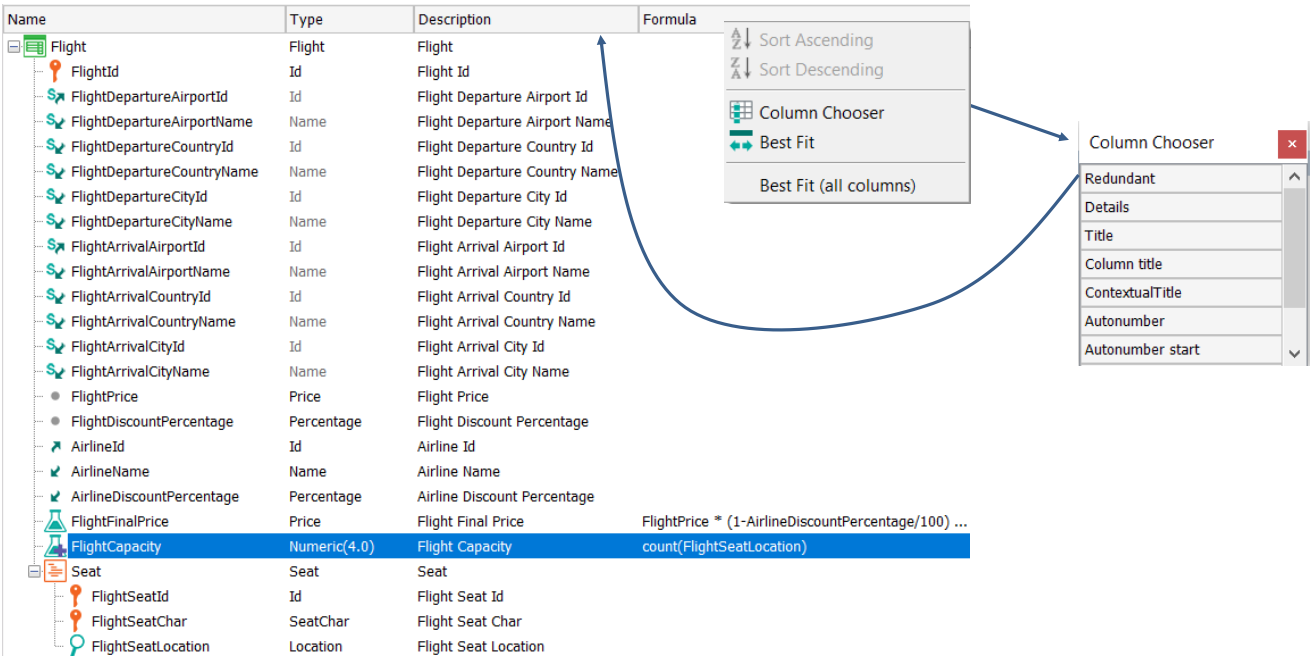
Assim definimos o atributo InvoiceBeforeDate como fórmula e no form editor invocamos o procedimento GetInvoiceBeforeDate, passando como parâmetro o InvoiceId da fatura de interesse.

No source implementamos um For Each que percorre a tabela INVOICE filtrando pelo InvoiceId recebido por parâmetro e recuperamos a data e o cliente da fatura recebida. Então chamamos uma sub-rotina que percorre novamente a tabela de faturas ordenada por fatura em ordem decrescente e ficamos com a fatura do mesmo cliente cuja data seja a maior possível, mas que seja menor que a fatura recebida por parâmetro.

Com isto estamos fazendo o mesmo que pretendíamos com a fórmula max, ou seja, maximizar a data da fatura do mesmo cliente mas que seja menor que a de interesse, em outras palavras, a data da fatura anterior do mesmo cliente.

O fato de podermos definir uma fórmula que invoque um procedimento para realizar o cálculo necessário abre muitas possibilidades, pois este cálculo pode ser tão complexo quanto quisermos e então poderemos simplesmente utilizar o atributo fórmula a partir de qualquer objeto de nossa base de conhecimento.

How to define a formula attribute as redundant

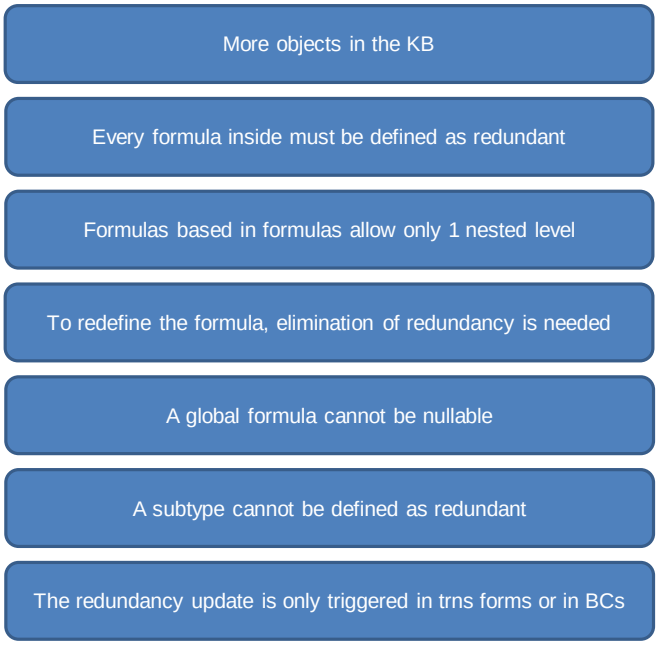


Embora as fórmulas redundantes sejam abordadas amplamente em outro vídeo, revisaremos aqui alguns conceitos.

Em alguns casos em que a fórmula é disparada muitas vezes, por motivos de desempenho, nos convém definir o atributo fórmula global como redundante. Isto fará com que o atributo passe a ser armazenado na base de dados e, portanto, o tempo que levará para recuperar o valor será menor que o tempo necessário para realizar os cálculos.

Para definir um atributo como redundante fazemos isso a partir da estrutura da transação, clicando com o botão direito do mouse na barra de colunas, clicamos em Column Chooser e adicionamos a coluna Redundant arrastando-a para a barra de colunas. Em seguida, na coluna Redundant, marcamos o checkbox para definir o atributo como redundante. No símbolo da fórmula é adicionado um sinal de “+” para indicar que é redundante.

Implications of defining a global formula as redundant



Definir um atributo como redundante tem um certo custo que devemos considerar.

Em primeiro lugar, para que o valor da base de dados esteja sempre atualizado, GeneXus criará objetos procedimentos que serão disparados cada vez que mudarem os dados envolvidos no cálculo da fórmula. Isto implica que para cada atributo fórmula que definimos como redundante, aumentará a quantidade de objetos na base de conhecimento.

Mas também, se o atributo fórmula que queremos que seja redundante foi definido com base em outros atributos que também são fórmula, deveremos também definir esses outros atributos como redundantes.

Outra limitação é que em fórmulas de agregação que se tornam redundantes, definidas com base em atributos que também são fórmulas redundantes, somente é permitido um nível de aninhamento. Se este limite for excedido, suas redundâncias não serão corretamente mantidas.

Para alterar a definição de uma fórmula que é redundante, primeiro deve remover a redundância, fazer a alteração e definir a fórmula como redundante novamente.

As fórmulas globais definidas como redundantes não permitem valores nulos, portanto, não poderemos definir a propriedade Nullable como Yes.

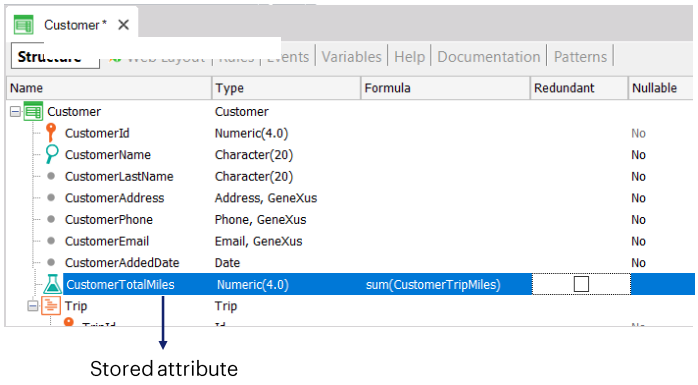
Um subtipo que seja fórmula não pode ser definido como redundante.

Por último, os procedimentos encarregados de atualizar o valor armazenado serão disparados somente quando for editado o registro através do form da transação ou por meio de um business component dela. Isto implica que se estão sendo alterados os dados envolvidos na fórmula a partir de um objeto procedimento, não serão disparados os procedimentos de atualização de forma automática e deveremos forçar esta atualização de forma explícita, invocando um utilitário criado por GeneXus para este fim.

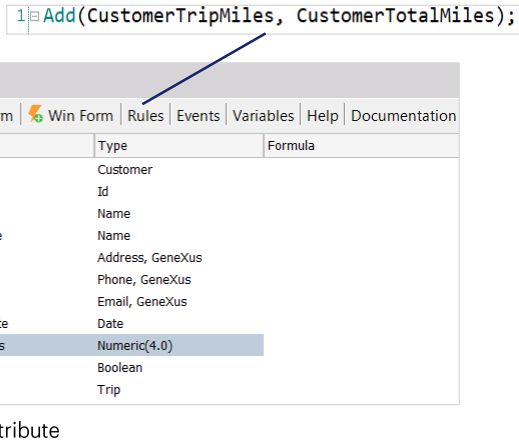
Concluindo, devemos pensar com cuidado quando decidimos definir um atributo fórmula como redundante, pois é possível que estas restrições tornem inviável nossa implementação.

Alternative solutions to a redundant formula

Vs.



Stored attribute



Stored attribute

Quando o valor de um atributo pode ser obtido através de um cálculo, geralmente o definimos como fórmula na estrutura da transação e este atributo deixa de estar armazenado na base de dados. Se precisarmos que o atributo fique sempre armazenado, podemos defini-lo como redundante, com o que GeneXus cria automaticamente procedimentos encarregados de atualizar seu valor e armazená-lo na base de dados.

No entanto, também poderíamos atribuir o cálculo ao atributo através de uma regra. O atributo não deixa de estar presente na tabela pelo simples fato de lhe atribuir um valor, portanto não se torna um atributo virtual, mas continua sendo um atributo armazenado.

Portanto, do ponto de vista da atualização e que o atributo esteja armazenado, atualizar o atributo através de uma regra e defini-lo como fórmula redundante poderiam ser consideradas soluções equivalentes.

A diferença fundamental entre uma solução e outra é o alcance do cálculo, se o definirmos como atributo fórmula redundante seu alcance será global, mas se definirmos o cálculo através de uma regra, o alcance será local e somente poderá ser acessado o cálculo a partir da transação onde foi definida a regra.

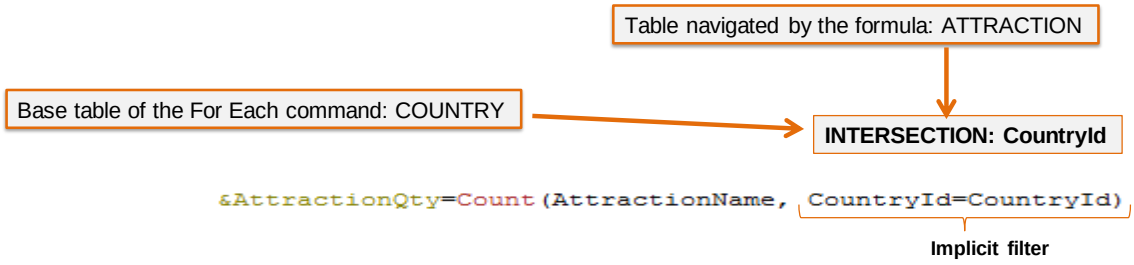
Como vantagem adicional, com a regra, o atributo continuará sendo editável no form da transação, embora como desvantagem, não seja possível forçar o disparo da regra sob demanda, portanto, não pode ser forçada a atualização do atributo, o que é possível ser feito com um atributo fórmula redundante.

Portanto, quando usar uma solução ou outra, depende em última análise, do uso que daremos ao atributo em nossa aplicação.

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
```

Name	Type	Is Collection	Description
Variables			
Standard Variables			
AttractionQty	Numeric(4.0)	☐	Attraction Qty



Vejamos o caso de fórmulas inline definidas no corpo de um comando For each.

Como já vimos, as fórmulas determinam a tabela a ser navegada, pelo atributo ou atributos referenciados dentro dos parênteses. Neste caso, incluímos o atributo AttractionName, portanto, a tabela em que navegará a fórmula é ATTRACTION.

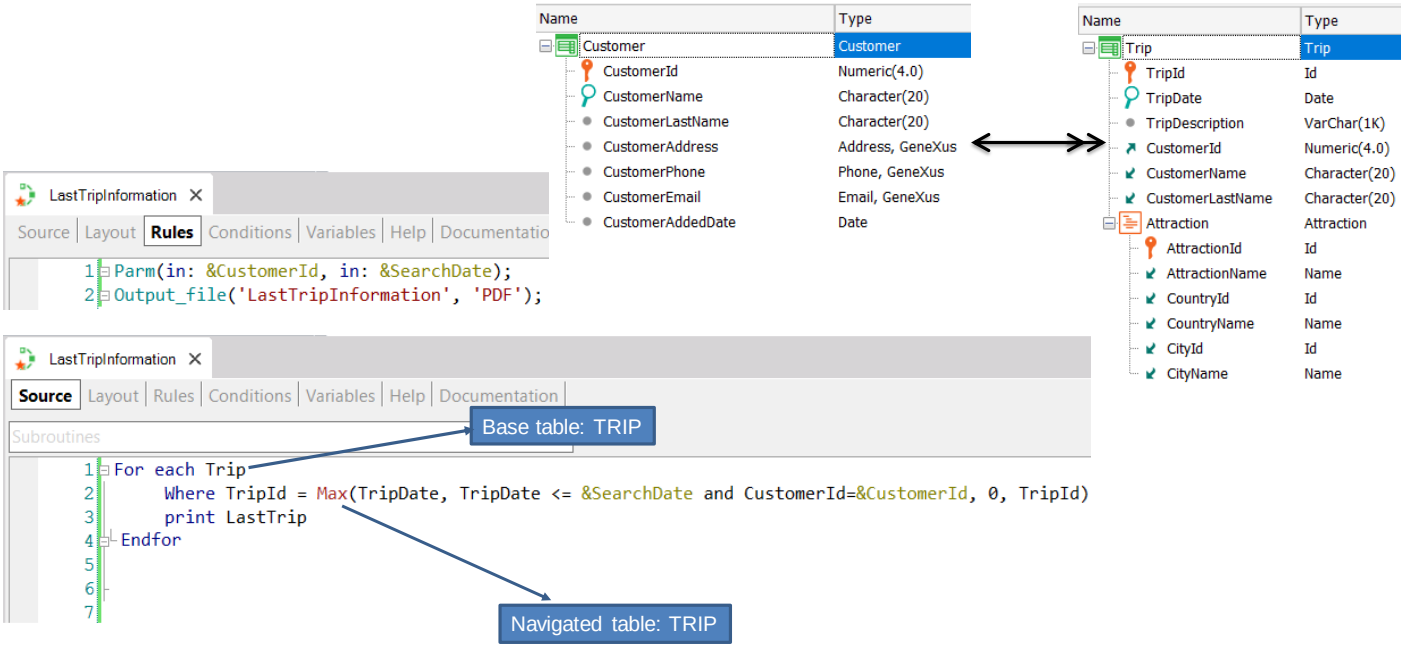
No exemplo não queremos contar todas as atrações da tabela ATTRACTION, mas sim aquelas que correspondem ao país em que estamos posicionados em cada iteração do For each que percorrerá a tabela COUNTRY.

Como a fórmula está definida dentro de um comando For each, está em um contexto em que está percorrendo uma tabela, neste caso, a de países. Vemos que há um atributo em comum entre as duas navegações, CountryId, e é por isso que GeneXus determina um filtro implícito pelo atributo em comum, de forma que para cada país que o For each encontre, são contadas somente as atrações que são desse país.

Outro requisito solicitado é que apenas sejam listados os países que possuem mais de duas atrações turísticas.
Para solucionar isto, podemos utilizar uma fórmula count como parte da expressão de filtro na cláusula Where do For Each.

Esta fórmula count, definida da mesma forma que a anterior, nos retornará a quantidade de atrações do país em que está posicionado o For Each em cada iteração (devido ao filtro implícito mencionado anteriormente) e utilizamos isso para filtrar aqueles países para os quais a quantidade de atrações seja maior que dois.

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



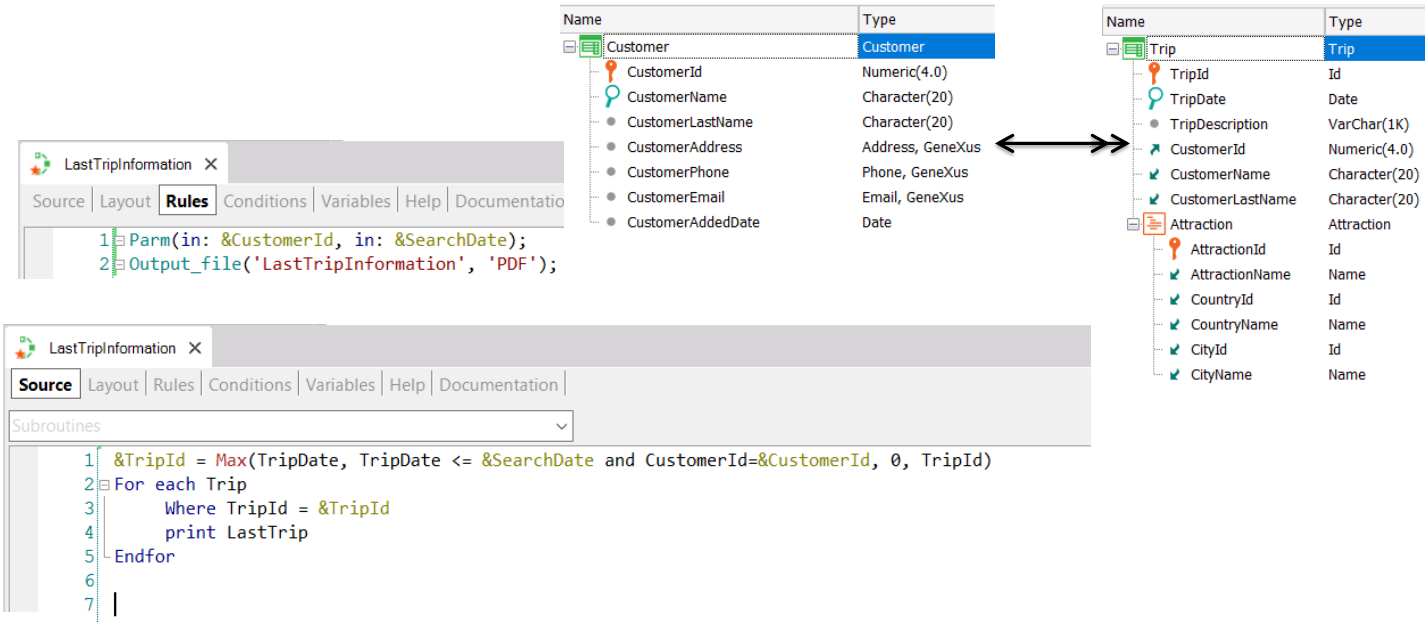
Vejamos agora um caso particular, em que coincide a tabela navegada pela fórmula com a tabela base do For Each.

Suponhamos que dado um cliente que tem muitas viagens, deseja recuperar os dados de uma viagem realizada, que tenha sido imediatamente anterior a uma data dada, ou seja, a última viagem anterior a essa data. O procedimento recebe por parâmetro o identificador do cliente que deseja a informação e a data de busca.

No source do procedimento existe um for each que percorre a tabela Trip e o where vai filtrar pelo TripId retornado pela fórmula Max. Em seguida, serão impressos os dados correspondentes a essa viagem.

Se analisarmos o que implementamos, o For Each irá iterar sobre a tabela TRIP e definirá esse contexto para a fórmula Max.
A fórmula Max também irá iterar sobre a tabela TRIP, mas devido ao contexto, será estabelecido um filtro automático, pois ambas as tabelas estão relacionadas. Neste caso é a mesma tabela, então a fórmula será filtrada pelo TripId que está posicionado o For Each. Portanto, retornará sempre esse TripId em que o For Each se encontra, portanto, a fórmula nunca percorrerá a tabela TRIP, pois somente acessará um registro.

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

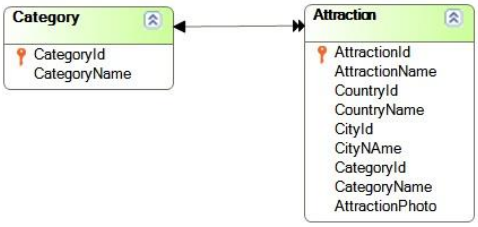


A solução neste caso é executar primeiro a fórmula para encontrar o identificador da viagem que atenda as condições desejadas e depois filtramos o For Each por esse valor de identificador.

Vemos que neste exemplo se definirmos uma fórmula inline dentro de um contexto (a tabela base do for each), acontece o mesmo que vimos no exemplo da fórmula global, cuja tabela associada coincidia com a tabela navegada, sendo a tabela associada a que fornecia o contexto.

Ao contrário das fórmulas horizontais que precisam de um contexto (já que só podem utilizar atributos da tabela associada e sua estendida), as fórmulas de agregação não precisam disso, mas se forem definidas dentro de um contexto, ele funcionará como filtro da 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.

Base table of the For each: ATTRACTION

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

Table navigated by the formula: ATTRACTION

Vejamos agora outro caso de uma fórmula inline definida dentro de um For Each, em que coincide a tabela navegada pela fórmula com a tabela base do For Each. Observemos que em ambos os casos, a tabela é ATTRACTION.

No entanto, neste exemplo adicionamos uma cláusula Unique por CategoryId e por isso GeneXus agrupará as atrações por categoria, tanto no For each quanto na fórmula, assim, a fórmula Count adicionará do contexto uma condição implícita em sua avaliação: contará todas as atrações para o atributo declarado na cláusula unique.

É como se fosse um corte de controle, cortando por CategoryId.

Graças a essa condição adicional proporcionada pela cláusula Unique, GeneXus pode resolver o problema de que a tabela navegada pela fórmula coincide com a tabela base do For each, já que só serão contadas as atrações da categoria dada na 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 condición2;
              Find(…) if condición3;
```

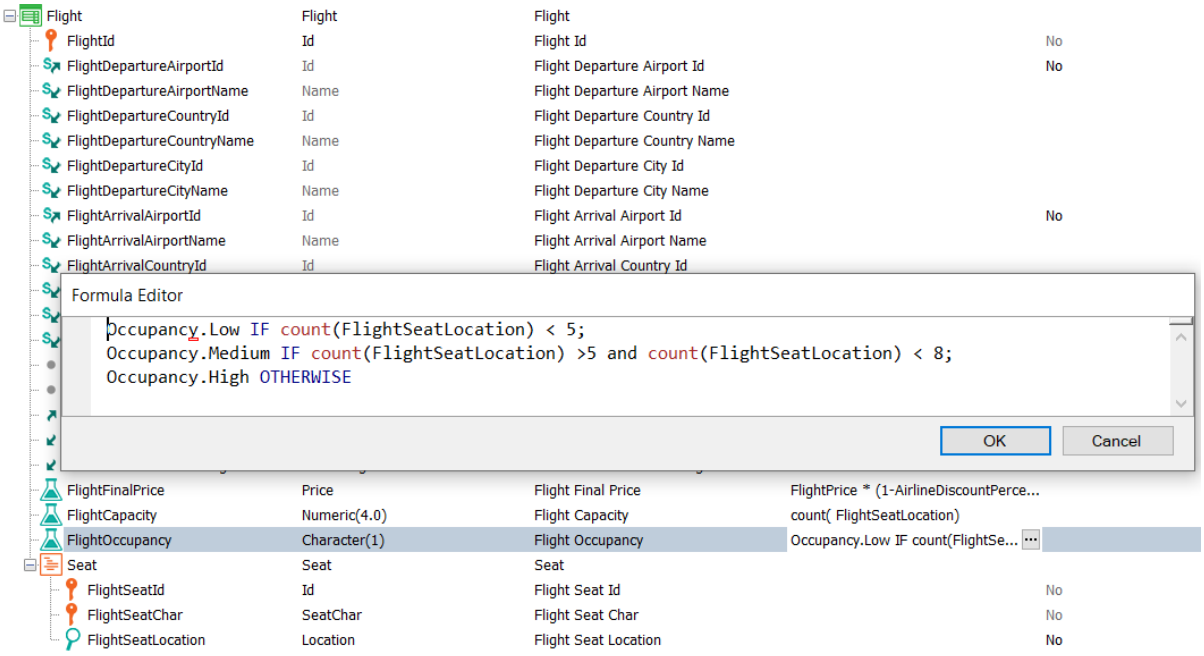
Em GeneXus podemos definir fórmulas complexas, usando fórmulas compostas.

As fórmulas compostas são fórmulas que integram várias fórmulas aggregate condicionais, podendo também conter expressões horizontais com condições de disparo.

As condições são qualquer expressão lógica válida, que pode conter atributos pertencentes à tabela estendida, da tabela associada ao atributo que está sendo definido como fórmula.

A primeira condição que ao ser avaliada dê True, fará com que o resultado da fórmula seja o da expressão da esquerda dessa condição e as demais expressões não continuarão a ser avaliadas.

Example



Vejamos um exemplo deste tipo de fórmulas compostas em nossa realidade da agência de viagens.

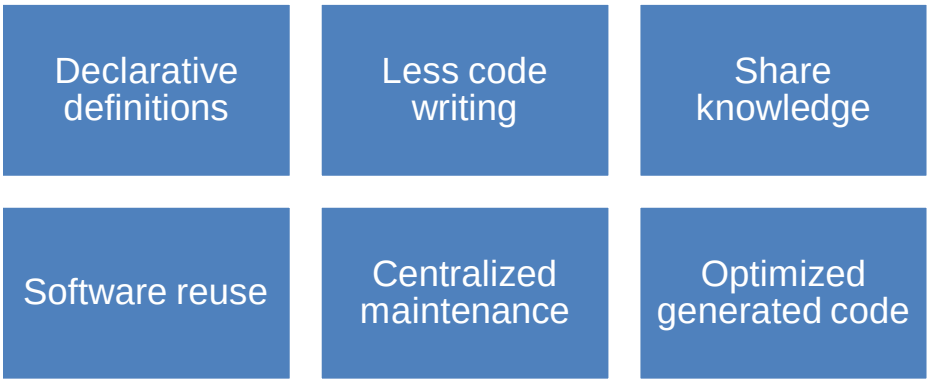
Aqui vemos que o atributo FlightOccupancy foi definido com base em expressões horizontais que atribuem o valor correspondente do domínio Occupancy (Low, Medium ou High), dependendo da quantidade de assentos do voo, que são calculados com fórmulas aggregate count.

Em particular, no nosso caso, poderíamos ter substituído as fórmulas aggregate pelo atributo FlightCapacity, mas é perfeitamente válido deixá-lo como está definido.

Nesta implementação, a estrutura é a de uma fórmula horizontal e as aggregate foram incluídas nas condições de disparo.

As fórmulas compostas nos permitem uma grande flexibilidade na definição de cálculos, podendo ser modelada uma grande quantidade de situações.

Conclusions



Depois de ver esta revisão conceitual do uso das fórmulas em GeneXus, concluímos que são muito úteis em muitos casos e fundamentalmente nos proporcionam as seguintes vantagens:

- Definição declarativa em vez de código procedural para fórmulas globais
- Nos permitem economizar código, principalmente nas funcionalidades das fórmulas aggregate que processam muitos registros, o que evita que tenhamos que iterar sobre os registros e implementar a lógica por código para contar, somar, maximizar, etc.
- São uma forma de compartilhar conhecimento, por exemplo no caso dos atributos fórmulas que podem ser usados em qualquer objeto da base de conhecimento
- Nos permitem reutilizar o código, pois as definições que fazemos através de uma fórmula (por exemplo em uma fórmula local), poderemos reutilizá-la em vários objetos que precisem do mesmo cálculo
- A manutenção é centralizada, pois no caso de uma fórmula global, alteramos a definição em um único local, na estrutura da transação onde foi definido o atributo.
- Definir fórmulas é ainda melhor do que escrever procedimentos e invocá-los. Quando é definida uma fórmula, GeneXus tem conhecimento de sua definição e é capaz de gerar sentenças otimizadas combinando a consulta da fórmula com a consulta na qual está presente a fórmula.

Em resumo, as fórmulas facilitam muito o desenvolvimento de nossas aplicações e é altamente aconselhável aprender a usá-las.

Se você quiser saber mais sobre este tema, convidamos você a ver mais conteúdo na wiki.

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