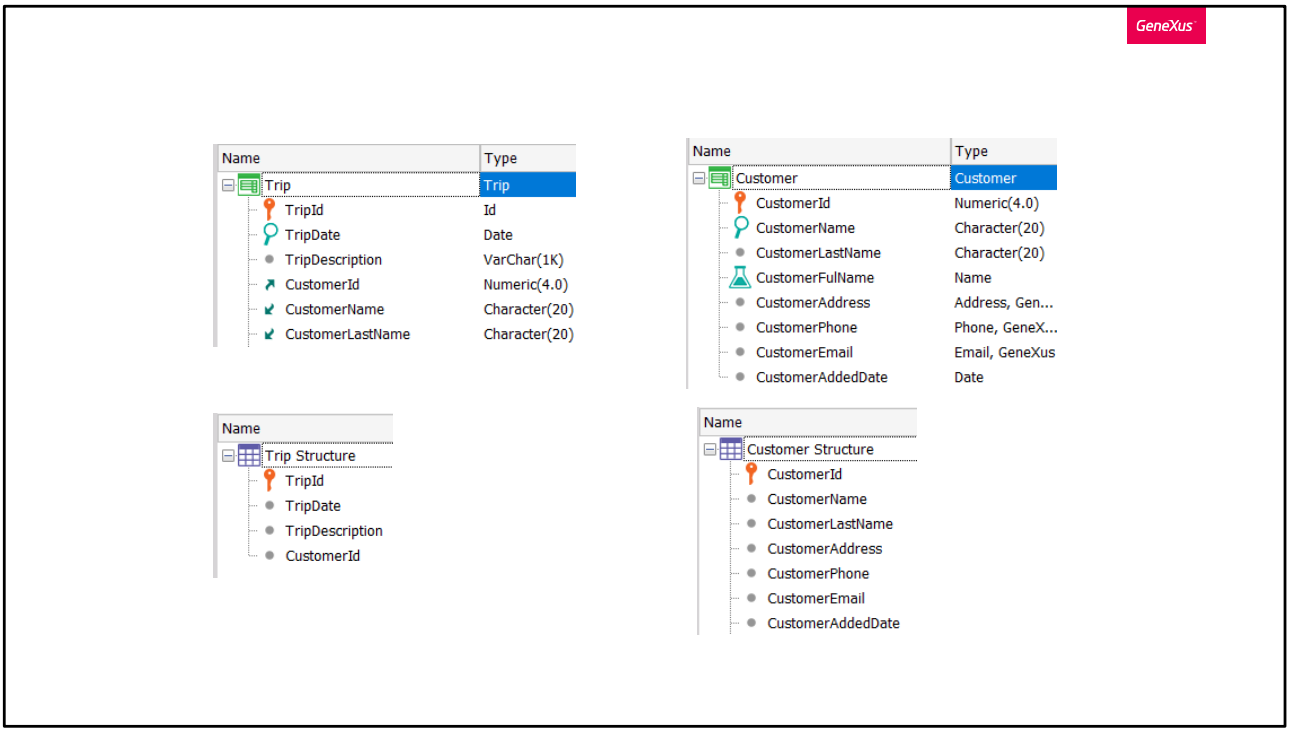
Atributos redundantes e sua manutenção

GeneXus

Neste vídeo veremos como definir atributos inferidos ou fórmulas, que por definição não são armazenados, como redundantes e que passem a ser atributos de uma tabela da base de dados.

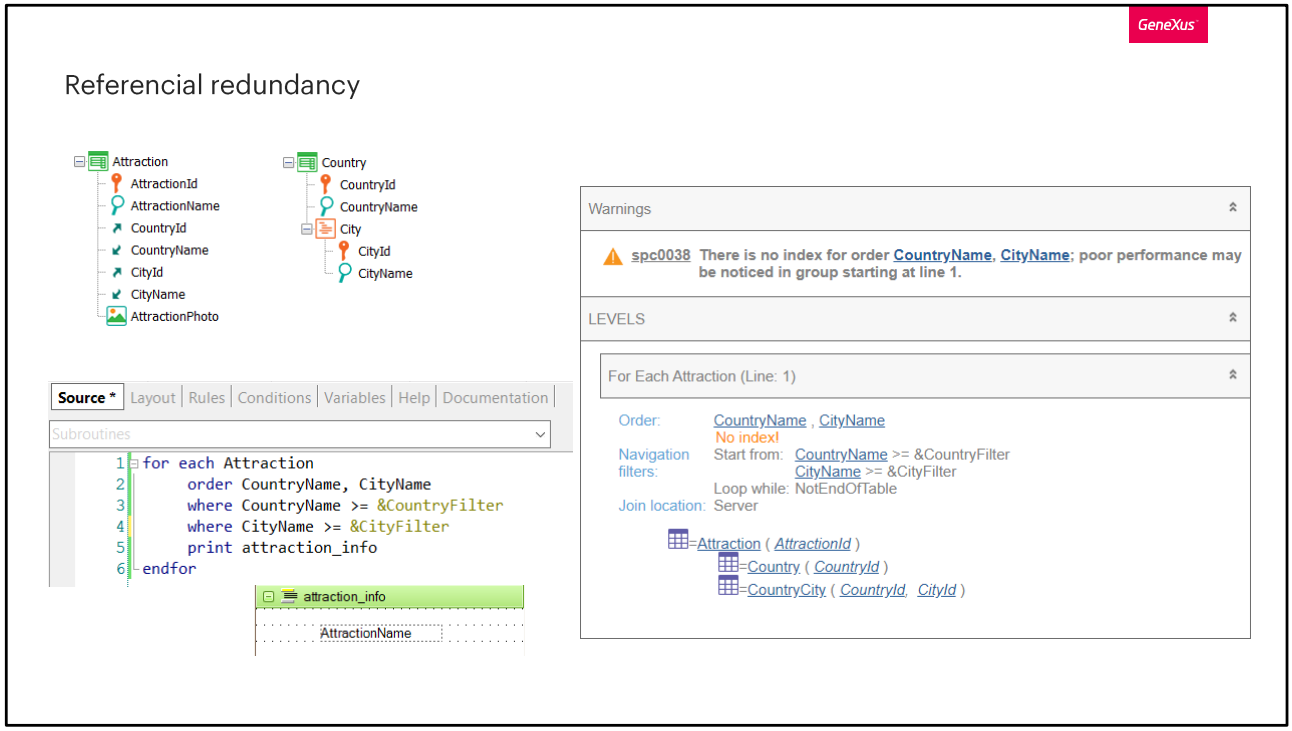


Como sabemos, GeneXus normaliza automaticamente a base de dados na Terceira Forma Normal, o que significa que os únicos atributos que podem estar em mais de uma tabela são os atributos que são chave primária, cumprindo funções de chave estrangeira.

O restante dos atributos, que chamamos de secundários, são armazenados em uma única tabela, determinados pela chave primária e no caso de serem adicionados a uma transação diferente, GeneXus se encarrega de inferi-los, recuperando através da chave estrangeira, seu valor a partir da tabela onde estão armazenados.
Além disso, quando definimos um atributo como fórmula em uma transação, ele deixa de ser armazenado e se torna um atributo virtual.

No entanto, muitas vezes por questões de desempenho, em alguns casos queremos permitir que um atributo inferido seja armazenado na tabela associada à transação onde é inferido, ou que um atributo fórmula que deve realizar muitos cálculos e consome muito tempo cada vez que é obtido seu valor, seja armazenado em sua tabela associada para poder obter o valor mais rapidamente.

GeneXus permite que possamos armazenar um atributo que por padrão não está armazenado em uma tabela, definindo-o como redundante.



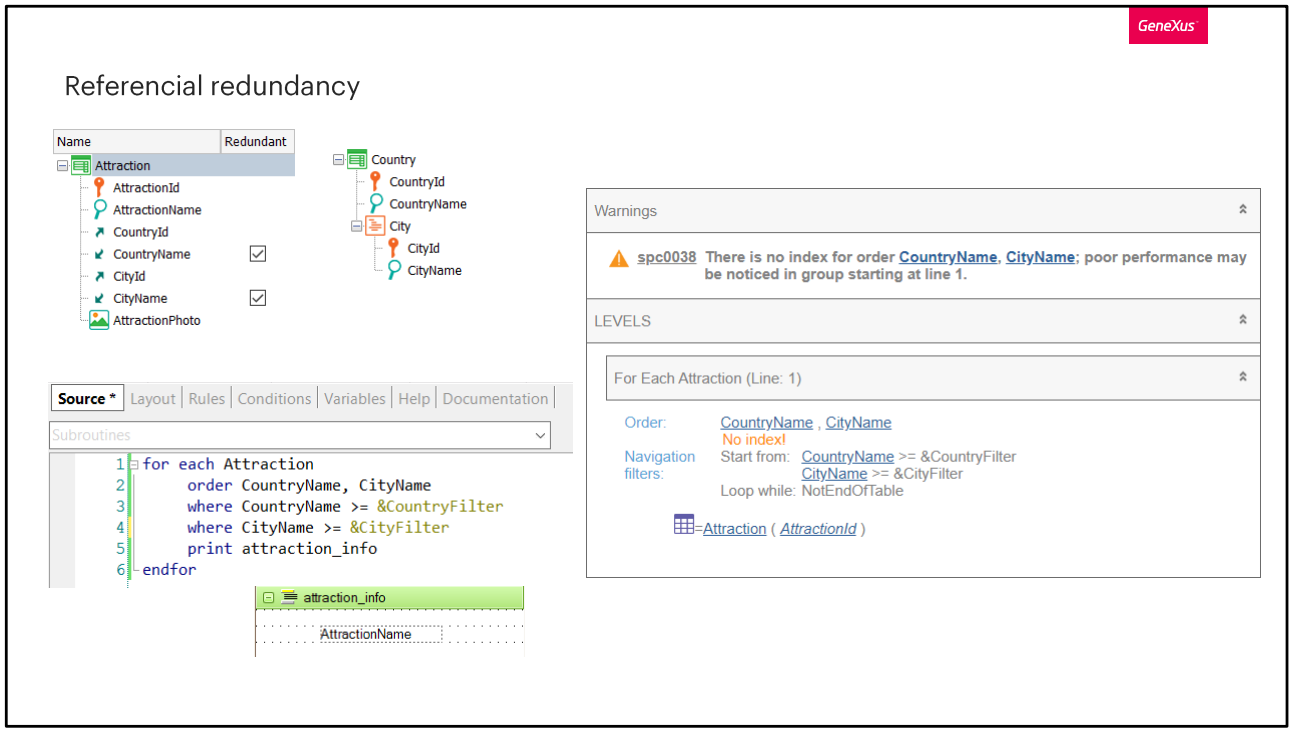
Quando redundamos um atributo inferido, é denominada redundância referencial.

A razão principal pela qual surgiu esta alternativa de redundar um atributo foi melhorar o desempenho.

Imaginemos que a tabela de atrações turísticas tivesse milhões de registros, e que quiséssemos recuperar aqueles que correspondem a países cujo nome seja alfabeticamente posterior a um valor determinado, e cujo nome de cidade seja posterior a outro.

Para otimizar sabemos que é conveniente ordenar pelos atributos dos filtros.

Se observamos a lista de navegação, por um lado vemos que a pesquisa está otimizada em termos de filtros de navegação, mas observamos que terão que ser feitos dois joins, pois CountryName e CityName não estão na tabela Attraction que é percorrida. Em vez disso, se estivessem na tabela...



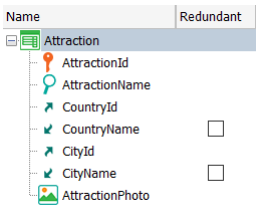
...não haveria join a ser realizado, portanto, o desempenho melhorará.

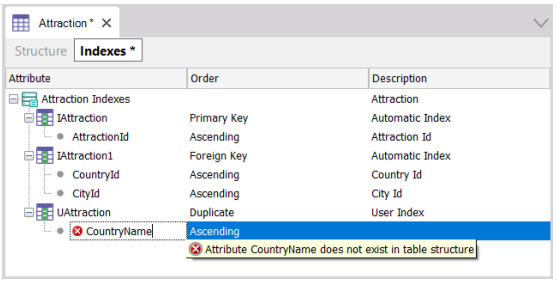
Em ambos os casos somos informados que devido ao fato de não existir um índice para a ordem que definimos, poderíamos notar problemas de performance. Dependendo do DBMS que estamos utilizando, sua inteligência e capacidades. Sabemos que hoje em dia os DBMSs são muito mais inteligentes do que antigamente e possuem estratégias para otimizar as buscas.
No entanto, em alguns casos, para resolver a consulta, será necessário criar um índice temporário que é eliminado após a consulta. E assim cada vez que a consulta é executada.

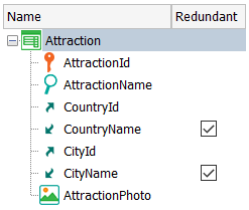
Se fosse o caso, e necessitássemos, justamente para evitar essa criação contínua de índice e sua posterior eliminação, criar um índice de usuário por esses atributos...

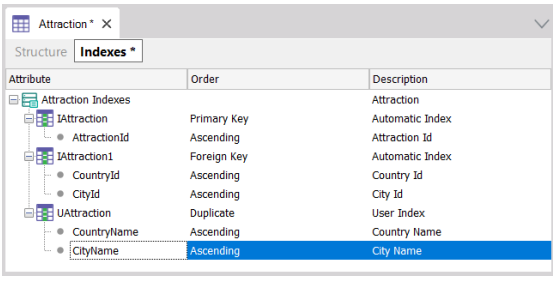
GeneXus

Referencial redundancy

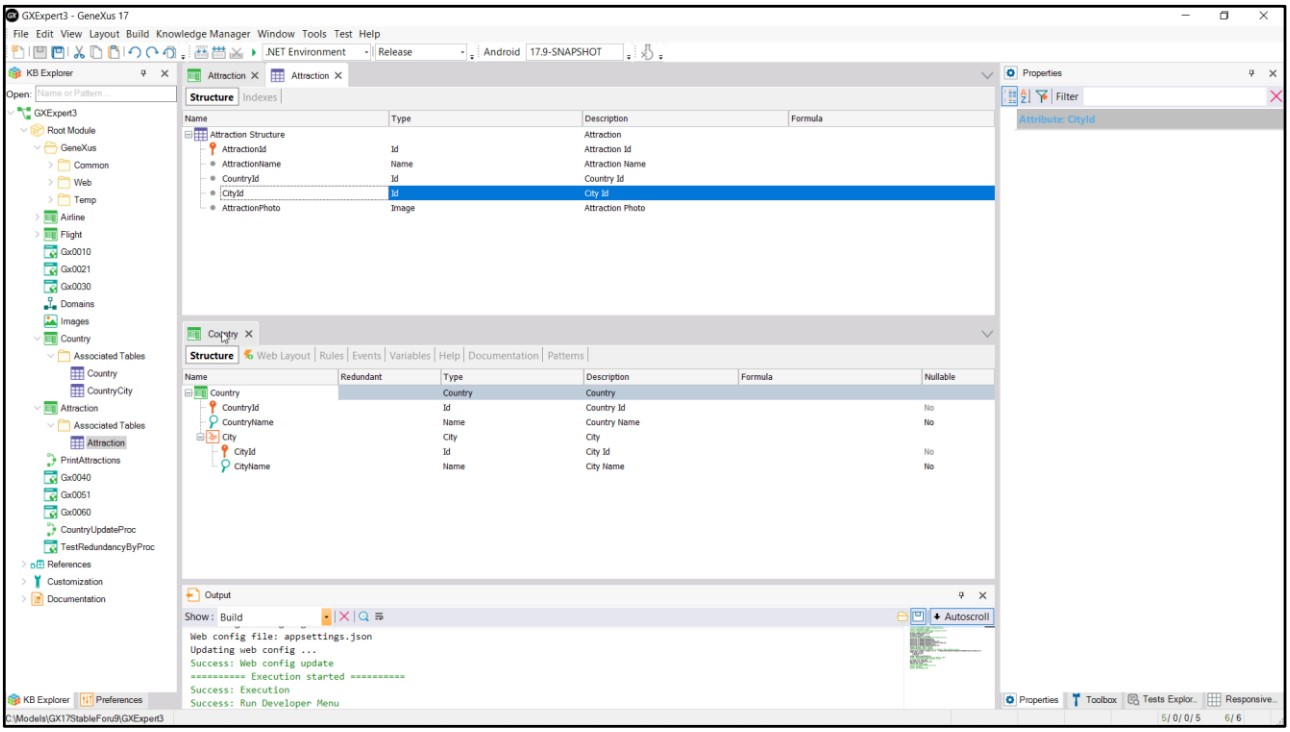






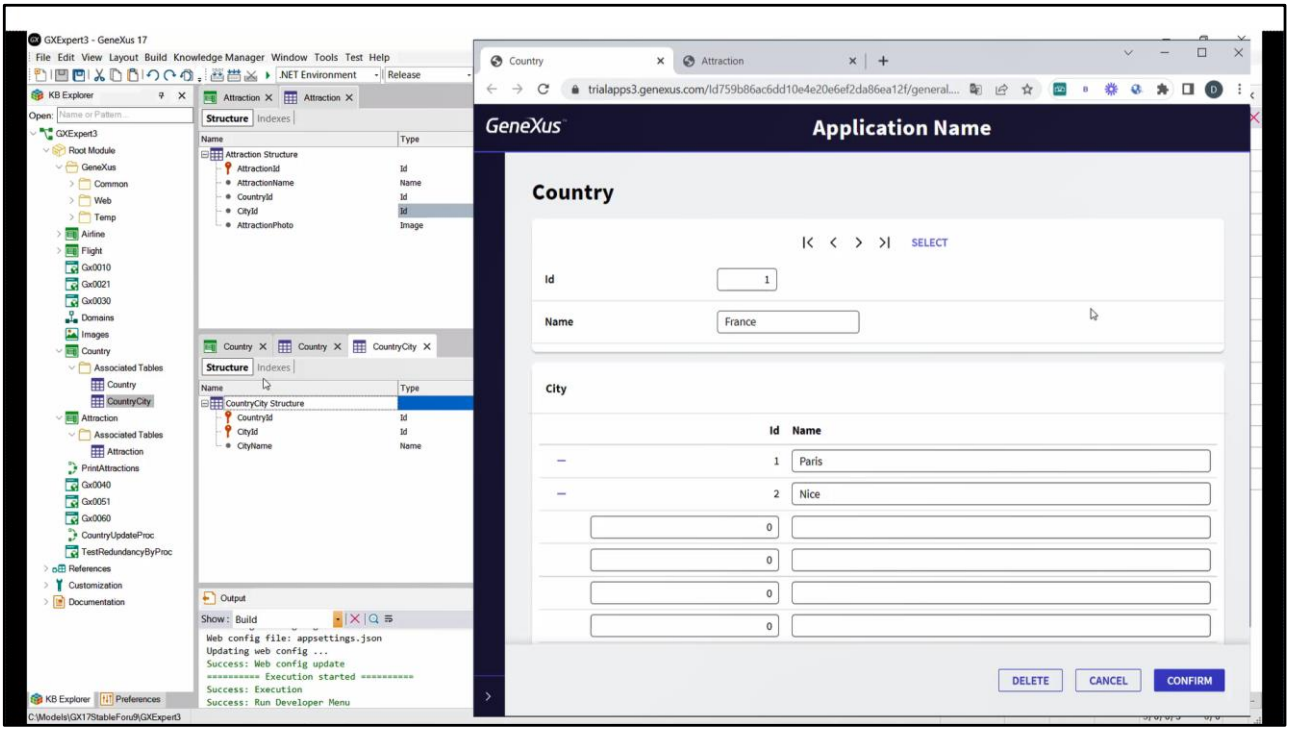


...só podemos fazer isso se ambos estiverem na tabela Attraction.

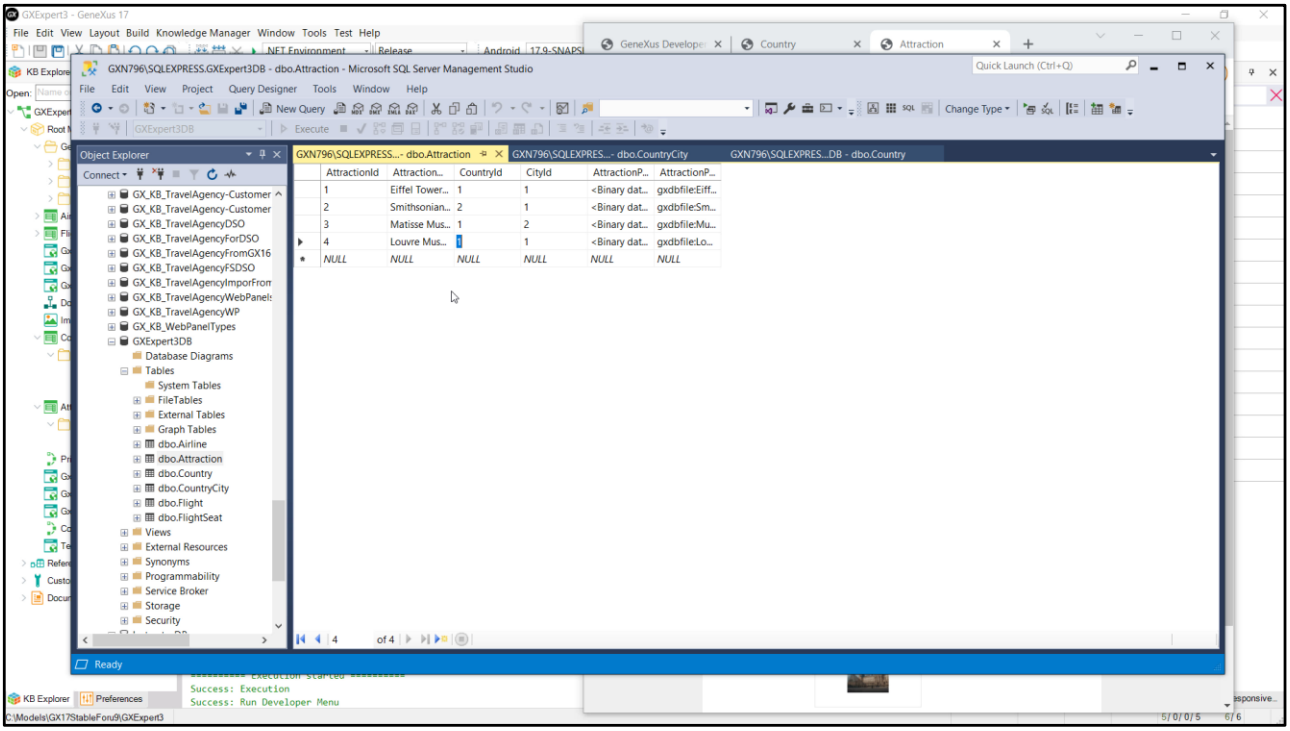


Vejamos em GeneXus como declarar a redundância e seus efeitos.

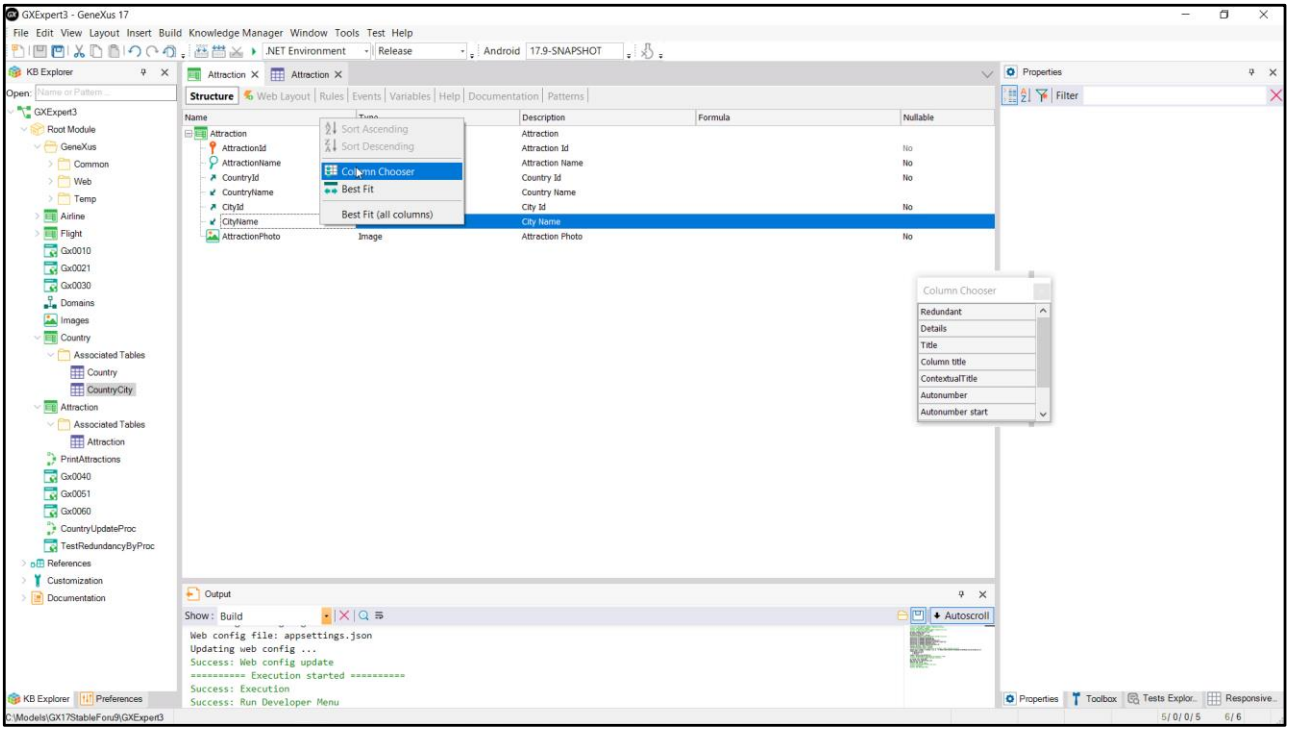
Temos a transação de Atrações inferindo, como de costume, os valores de CountryName e CityName a partir das chaves estrangeiras. De fato, se observamos a estrutura da tabela de atrações, vemos que não existem tais atributos, apenas as chaves estrangeiras.



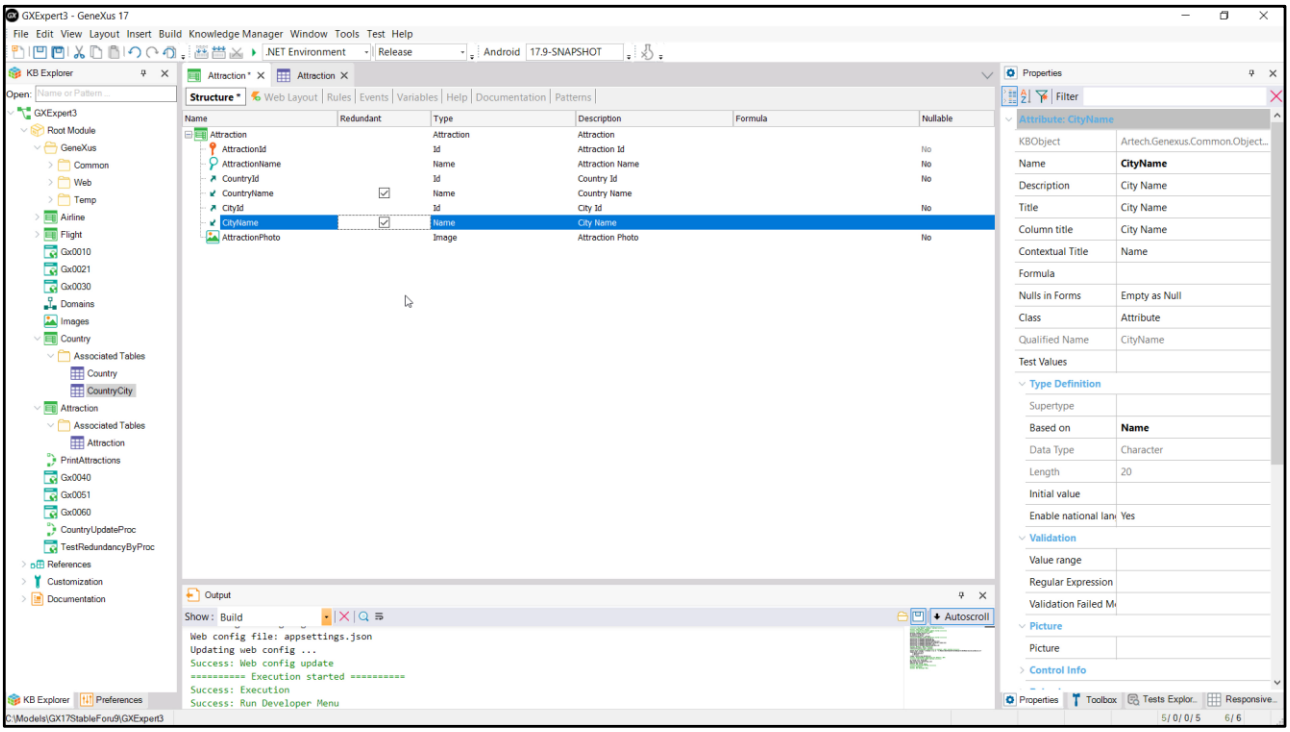
Já temos alguns dados carregados em execução. Temos dois países com suas cidades. Vamos prestar atenção especial ao 1, France, com suas duas cidades Paris e Nice. Então vemos que temos 3 atrações da França: duas de Paris e uma de Nice.



Se buscamos os dados das tabelas no SQLServer... vemos que na de atrações existem apenas as chaves estrangeiras.

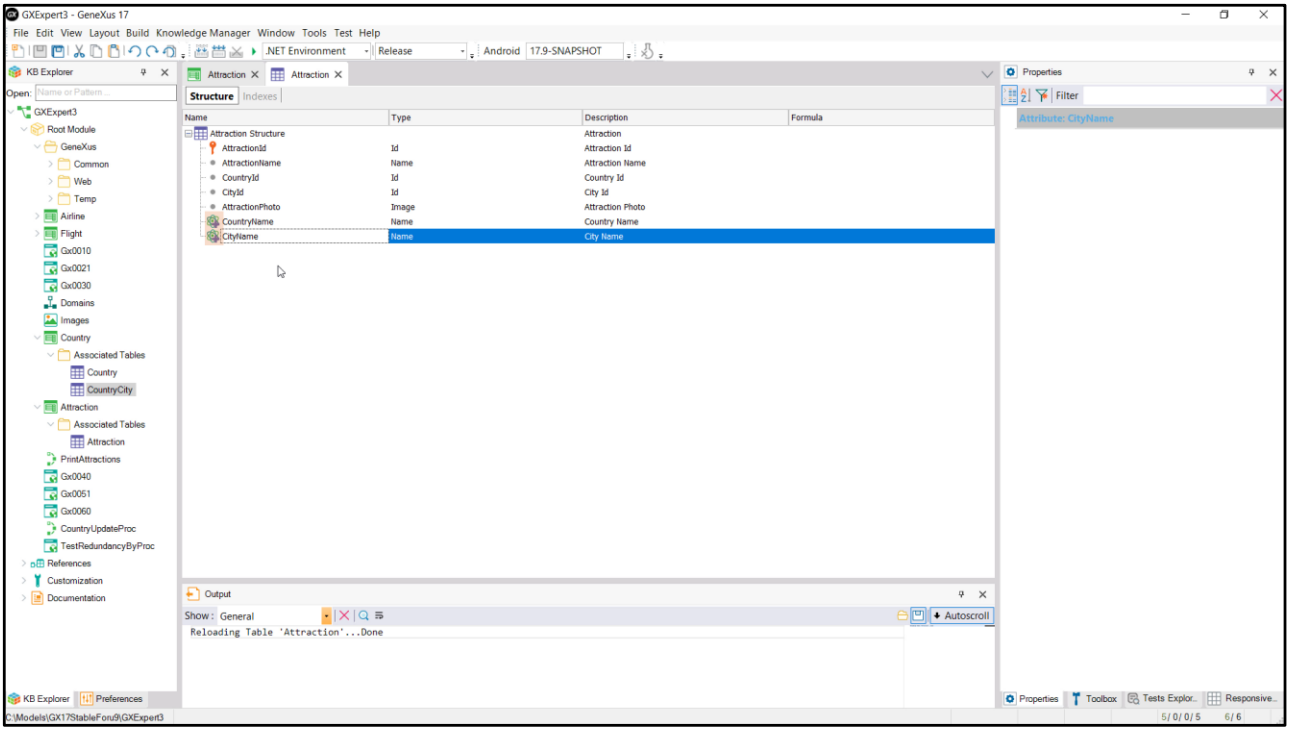


Bem, agora queremos definir em Attraction os dois atributos inferidos como redundantes. Para fazer isso, adicionamos no editor da estrutura da transação esta coluna...

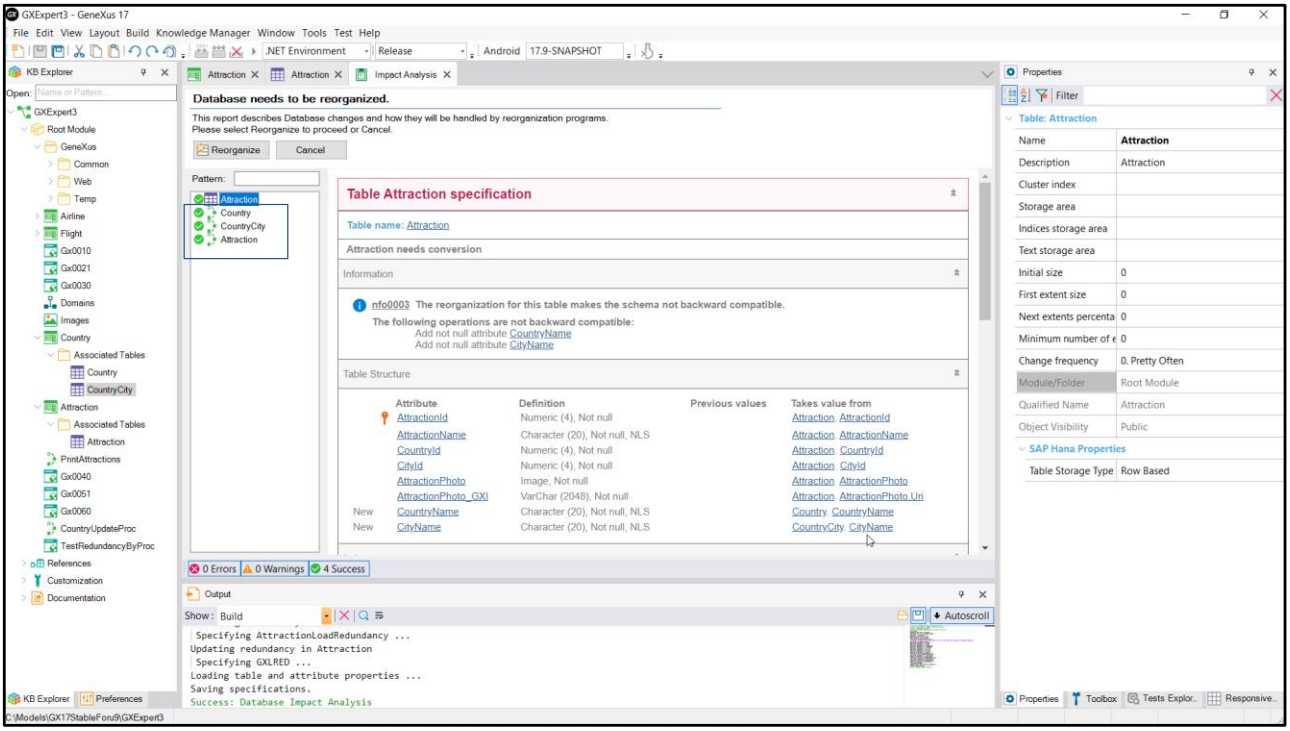


... que nos oferece check boxes para indicar quais atributos da estrutura da transação queremos redundar. Nos oferece apenas redundar os atributos inferidos. Se houvesse atributos fórmula, também nos ofereceria redundá-los, como veremos.

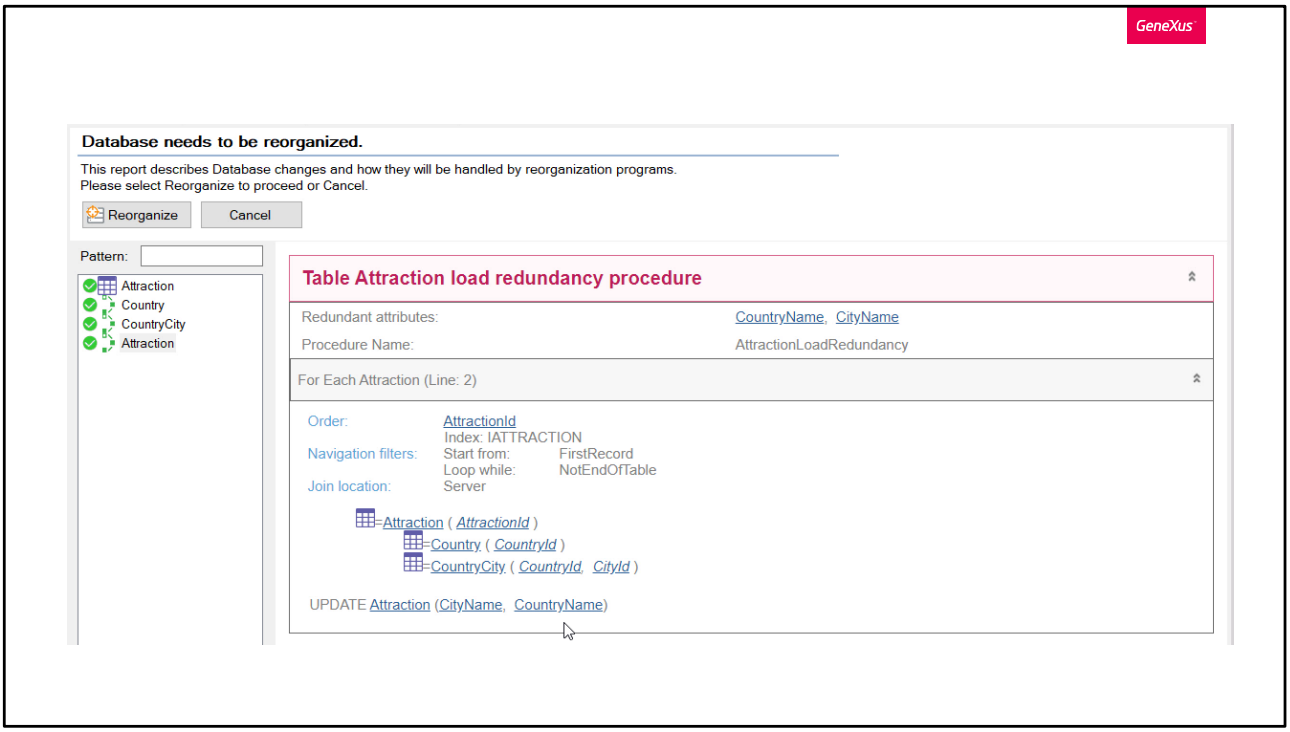
Vamos marcar ambos, porque queremos que ambos sejam redundados na tabela Attraction.



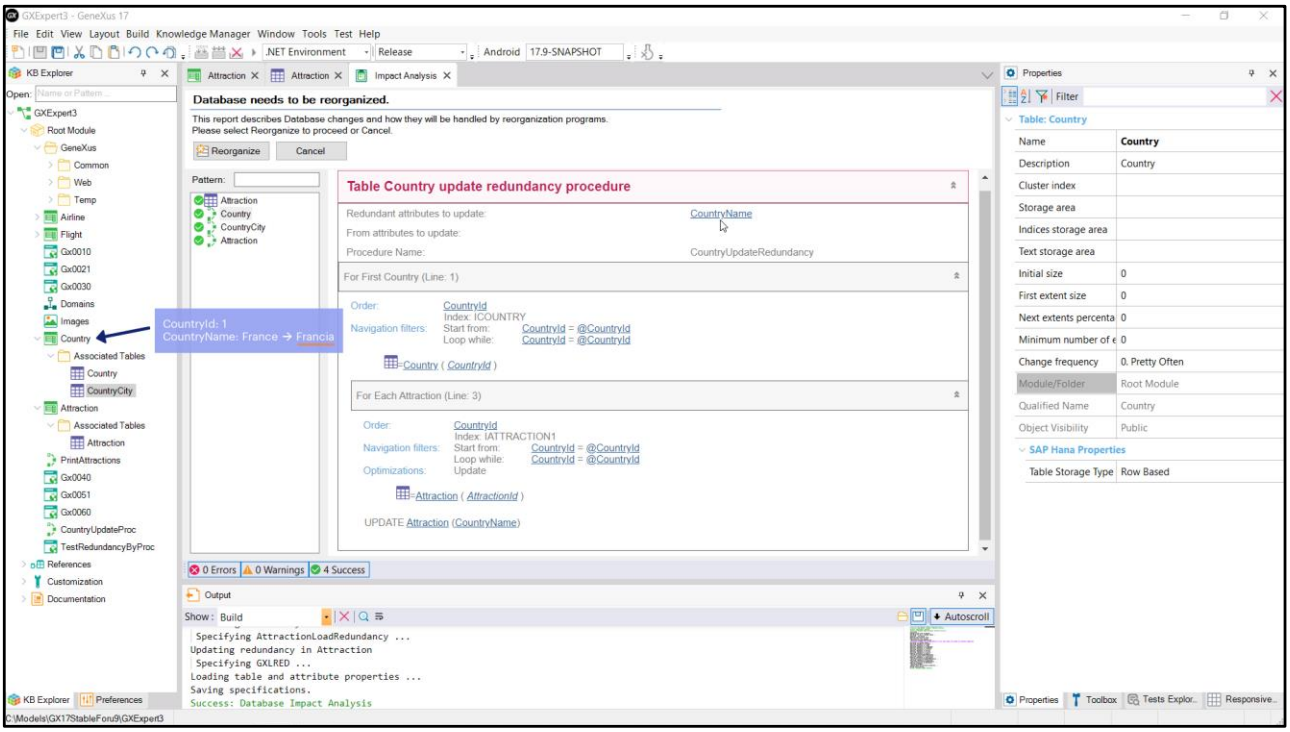
Vamos gravar. Vemos que foram adicionados os atributos à estrutura da tabela, e indica-se desta forma que são redundantes.



Se agora pedimos para executar, claramente terá que reorganizar a tabela Attraction para adicionar esses atributos que até o momento eram inferidos, mas agora queremos que sejam armazenados.
Observemos que é indicado de onde serão obtidos seus valores.
Bem, isto nós esperávamos. Mas, o que são estes três procedimentos internos que criará GeneXus?

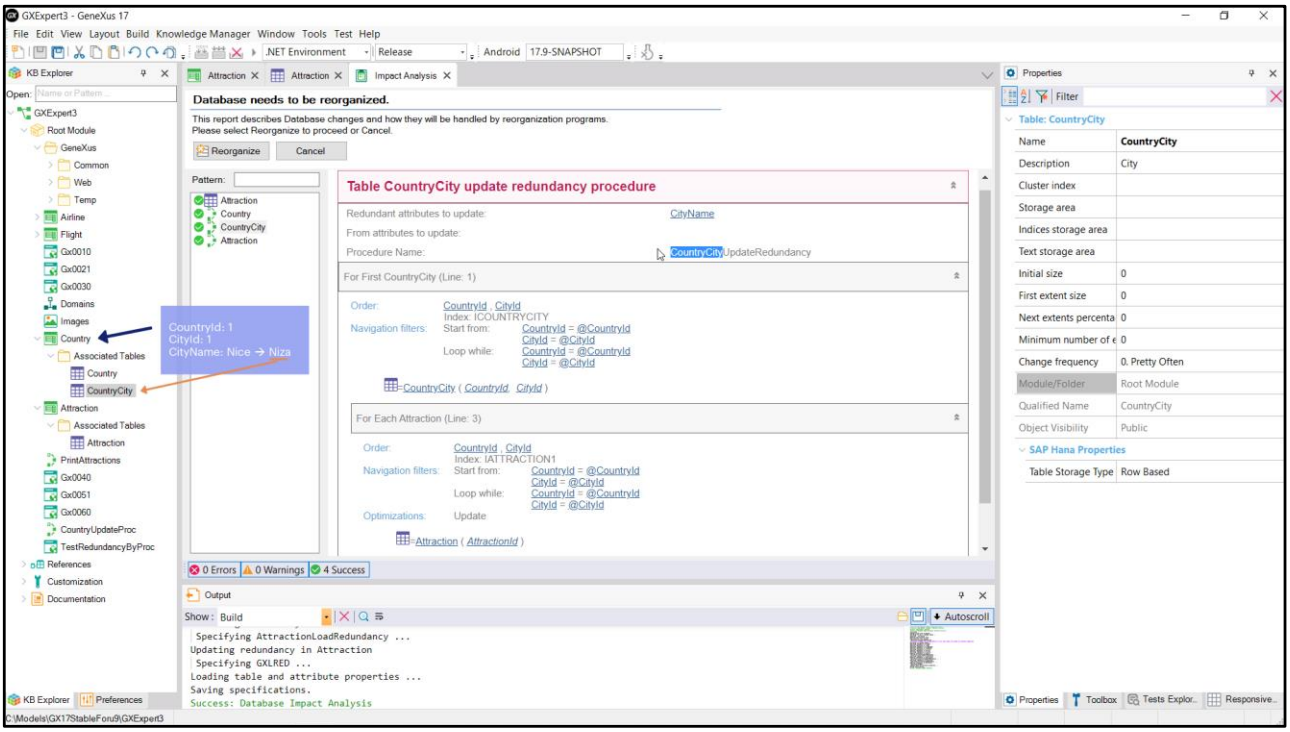


Vamos começar analisando o último. Vemos que o nome é AttractionLoadRedundancy. Aqui nos diz que é o procedimento de carga das redundâncias da tabela Attraction. E nos indica quais são os atributos redundantes que deve carregar. O que este procedimento fará é percorrer toda a tabela de atrações e para cada uma ir buscar em Country o valor do atributo CountryName para armazená-lo no atributo redundante desta tabela Attraction, e ir buscar na tabela de cidades o valor do atributo CityName para armazená-lo no novo atributo CityName redundante nesta tabela Attraction. Resumindo, deverá executar automaticamente este procedimento após reorganizar a tabela Attraction para que os atributos redundantes sejam carregados com os valores apropriados.



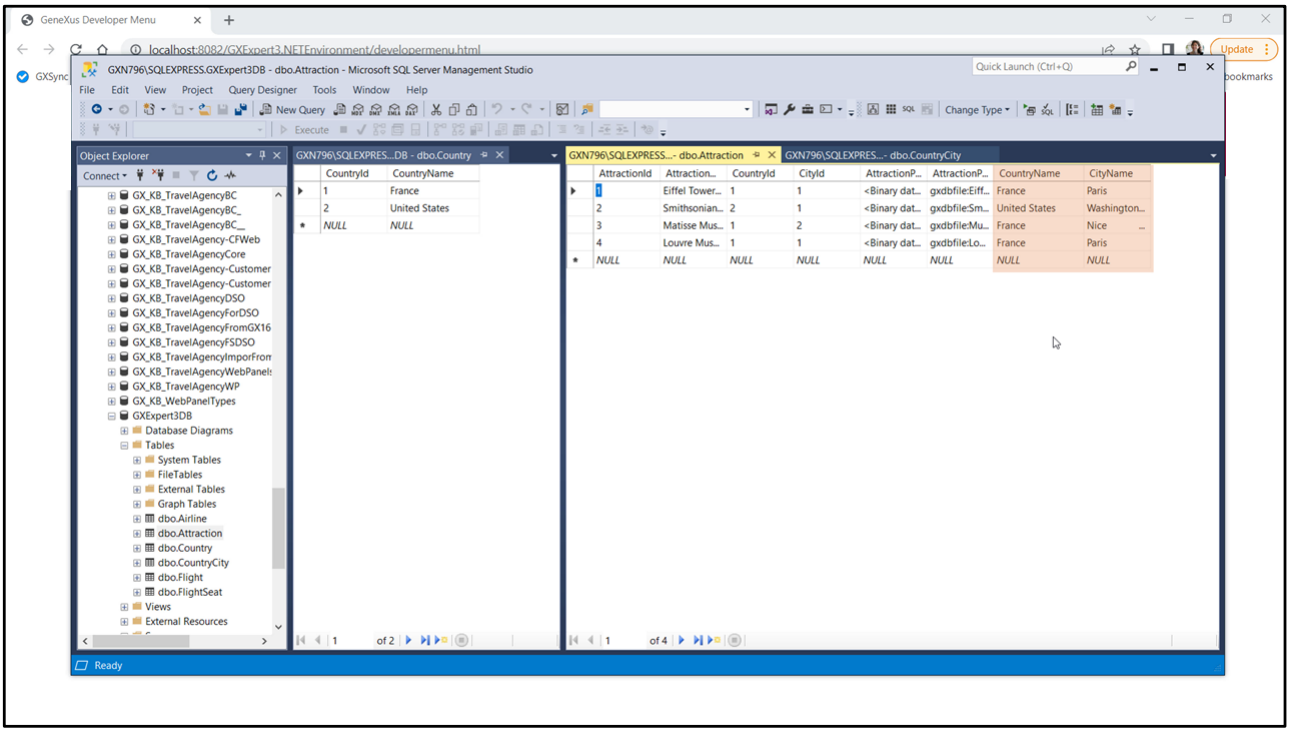
Bem, mas e estes dois procedimentos?

Se analisamos o primeiro, vemos que é um procedimento de atualização das redundâncias da tabela Country. Sabemos que o atributo CountryName desta tabela é redundante em Attraction. Isto significa que se executamos a transação e alteramos para um país seu nome, essa alteração também deverá ser feita para todas as atrações que correspondem a esse país, para manter a redundância atualizada. É o que fará este procedimento, que será invocado de forma transparente cada vez que o usuário modificar através da transação Country (ou de seu business component) o valor de CountryName para um CountryId. Aqui podemos ver que o procedimento recebe como parâmetro o valor da chave primária para instanciar esse registro, e então percorre a tabela de atrações filtrando por esse valor e para cada atração atualiza o CountryName nessa tabela.



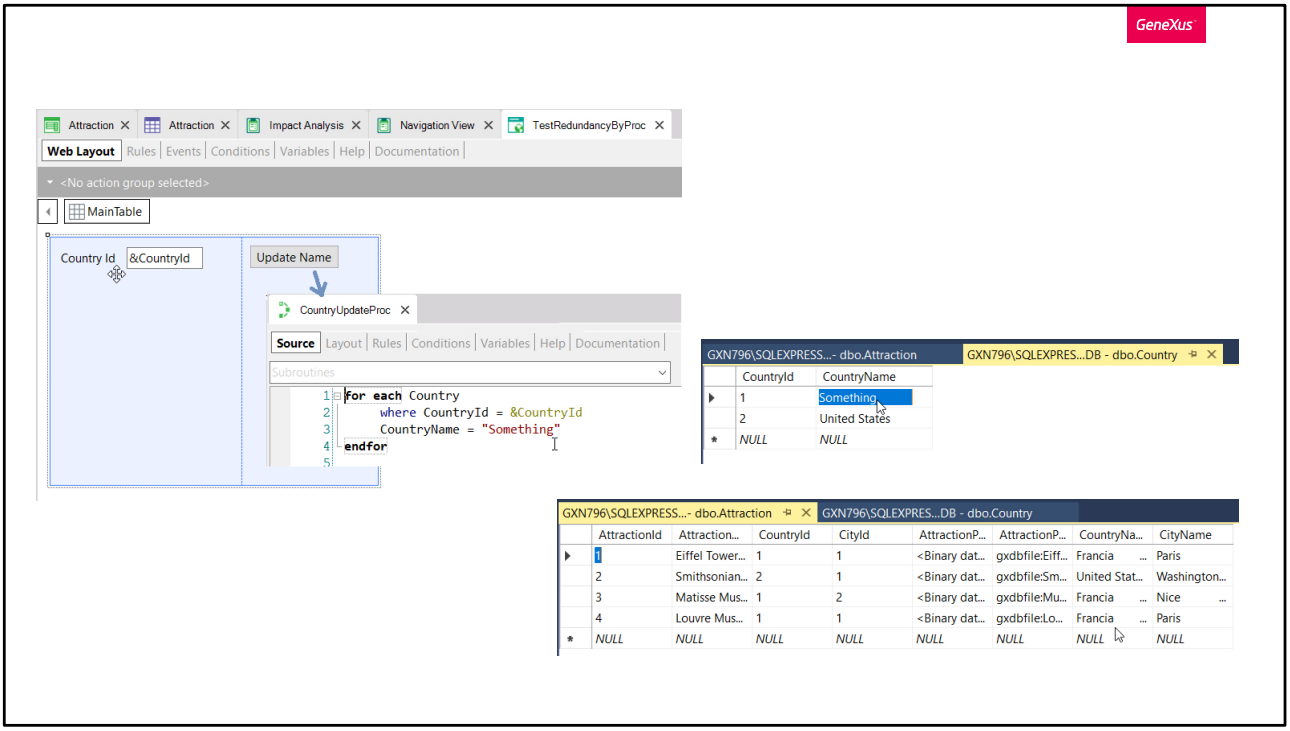
E o que acontece com este outro procedimento? O mesmo, mas para manter atualizadas as redundâncias relativas à tabela de cidades. Ou seja, quando a partir da transação Country for modificado o nome de uma cidade de um país, automaticamente e de forma transparente será executado este procedimento, para o qual serão enviados por parâmetro o id de país e o id de cidade, será acessado esse registro da tabela e para cada registro de Attraction que corresponda, será modificado em Attraction o valor do atributo redundante CityName.

Estes procedimentos serão criados então nesta reorganização, e será adicionada às transações a lógica que acabamos de explicar. Reorganizamos.



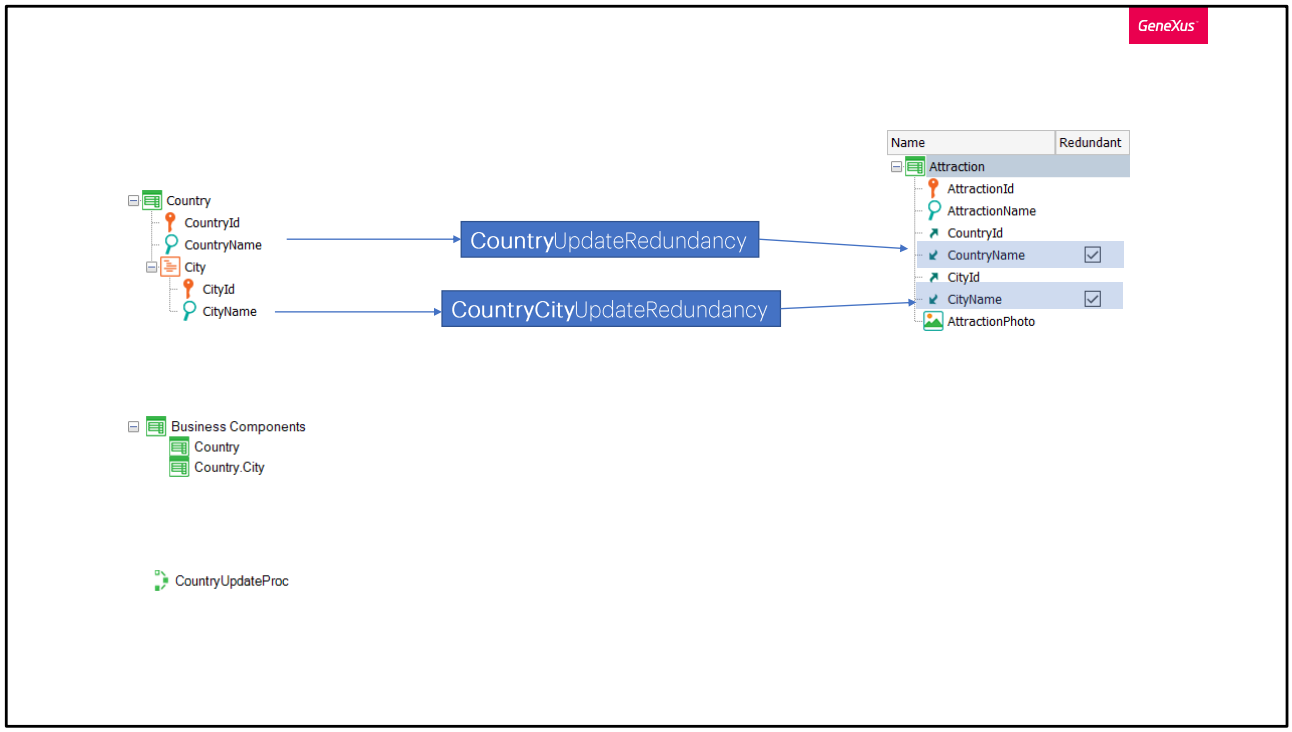
Vejamos agora a tabela no SQLServer. Foram adicionados os atributos redundantes e foi atribuído o valor que correspondia.
Vamos agora mudar o nome do país France para seu nome em espanhol, Francia. O faremos através da transação.
Vamos ver os dados da tabela... mudou o nome na tabela da transação... e ver o que aconteceu com o atributo redundante... exatamente como esperávamos, o atualizou.
Da mesma forma, se vamos modificar um nome de cidade, por exemplo o de Niza, escrevendo-o agora em espanhol... vemos que na tabela das cidades está logicamente alterado, e agora na de Atrações... também.

Isso foi por fazer a modificação através da transação Country.



Vejamos o que acontece se fizermos isso através de um procedimento... Temos um web panel para que o usuário insira um id de país, e ao pressionar o botão invocamos um procedimento que recebe esse id e vai para a tabela Country, para modificar para o país com esse id o valor do atributo CountryName para este “Something”. Vamos executar para testar.

Escolhemos o país de id 1, que era a França. Vemos que na tabela Country efetivamente foi modificado seu valor pelo atribuído pelo procedimento. Mas agora, vamos ver se o procedimento manteve atualizada a redundância em Attraction. Não, ele não fez isso.

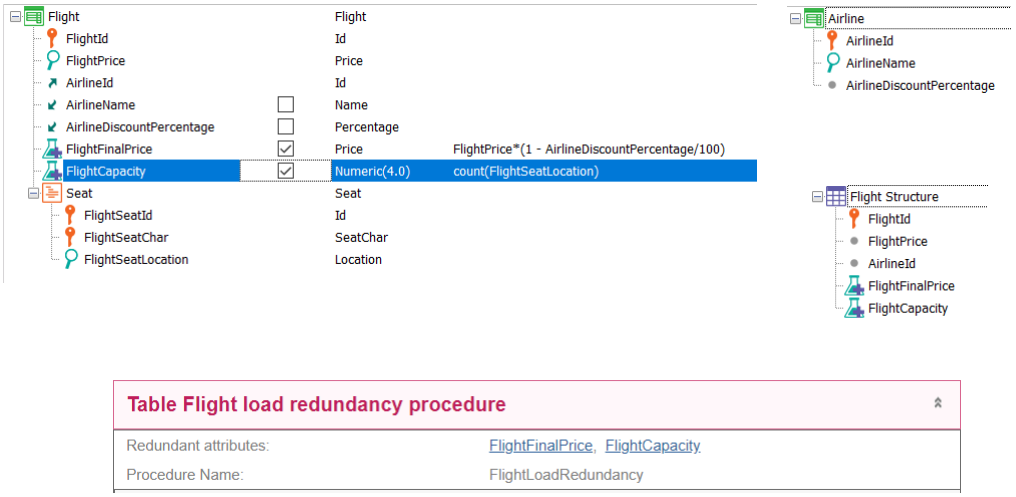


GeneXus invoca esses procedimentos que vimos cujos nomes terminam em UpdateRedundancy somente quando as modificações são realizadas através da transação (ou do business component, logicamente). Não os invoca quando as modificações são realizadas de outra forma.

Portanto, deve-se tomar cuidado e, no caso de modificar atributos redundantes em outras tabelas por outro meio diferente da transação ou do business component, o desenvolvedor deverá se encarregar de manter os atributos redundantes atualizados. GeneXus não o fará.

GeneXus

Formulas redundancy

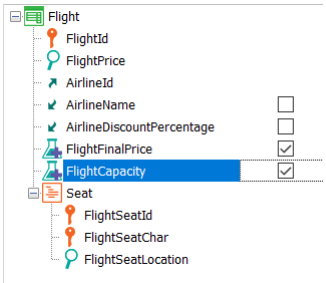


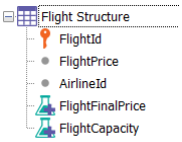
O outro caso de redundâncias que já mencionamos foi o de fórmulas. Neste exemplo vemos que além de nos oferecer redundar os atributos inferidos, também nos oferece redundar as duas fórmulas definidas para a transação: a horizontal que aplica um desconto no preço do voo de acordo com o percentual de desconto definido pela companhia aérea, e a aggregate que conta a quantidade de assentos do voo.

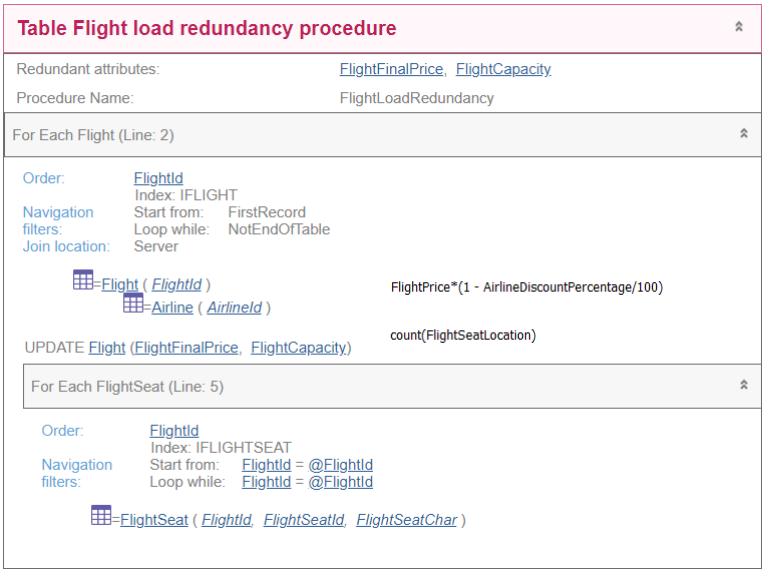
Ao definir estas fórmulas como redundantes serão adicionados os atributos à tabela. Mas além disso, como é evidente, na reorganização GeneXus deverá criar também um procedimento para carregá-los com os valores correspondentes.

GeneXus

Formulas redundancy



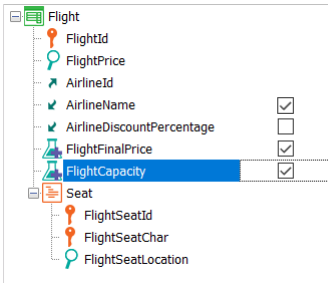


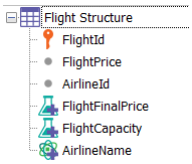


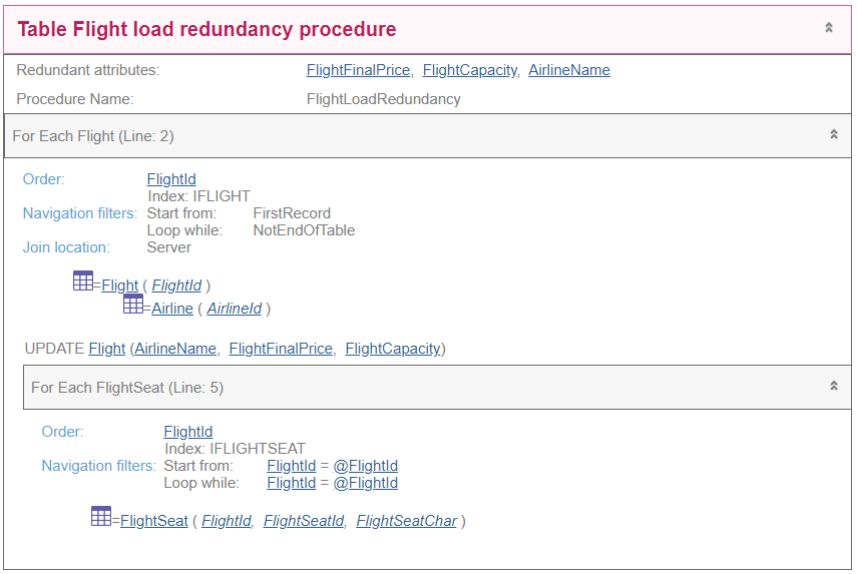
Ou seja, um procedimento no qual para cada registro da tabela Flight disparará o cálculo de cada fórmula... para FlightFinalPrice deverá ir buscar para Airline o valor de AirlineDiscountPercentage e armazenará seu valor no atributo redundante; e para FlightCapacity deverá contar os registros associados na tabela FlightSeat, e armazenar o resultado no atributo redundante.

GeneXus

Formulas redundancy



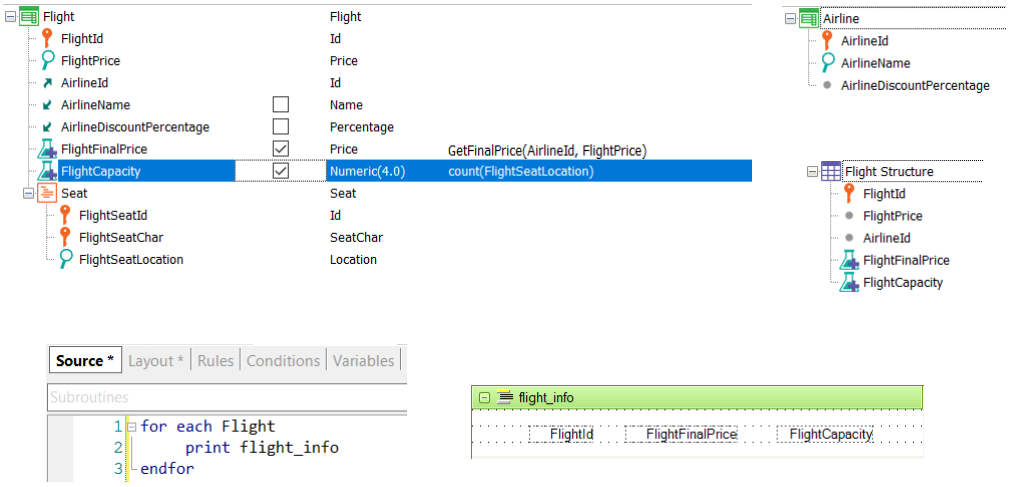




Se também definíssemos, por exemplo, AirlineName como redundante, ou seja, uma redundância referencial, neste mesmo procedimento é onde seria carregado seu valor na tabela. Por isso o nome do procedimento sempre é composto pela concatenação do nome da tabela onde estão as redundâncias, neste caso Flight, e LoadRedundancy (carga de redundâncias). Ou seja, neste procedimento serão carregadas todas as redundâncias da tabela: as referenciais e as de fórmulas.
E será o procedimento chamado automaticamente na reorganização, após serem adicionados os novos atributos à tabela da base de dados.

GeneXus

Formulas redundancy



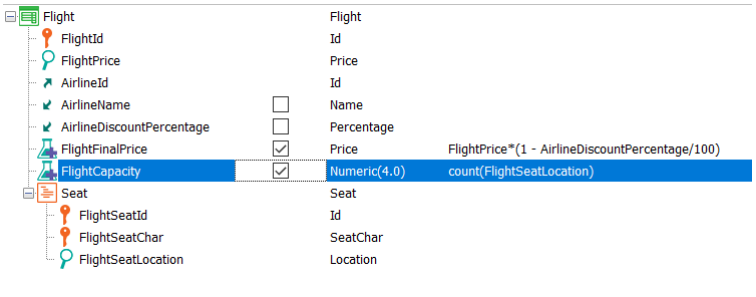
Vamos voltar a nos concentrar apenas nas fórmulas. Claro que, uma vez redundadas, quando for necessária a informação de preço do voo ou capacidade, não será disparada novamente a fórmula, mas será trazido o valor armazenado, e essa é justamente a graça.

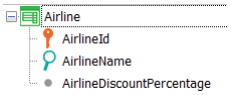
Se executássemos milhões de vezes esta lista, e mesmo que isso não aconteça na realidade, houvesse milhares de assentos para cada voo, então executar cada vez o cálculo da fórmula FlightCapacity poderia se tornar um problema de desempenho. Tê-la redundante nos economiza o gasto do cálculo para cada consulta. Temos apenas o gasto do cálculo para carregar o atributo redundante a primeira vez e para mantê-lo atualizado posteriormente.

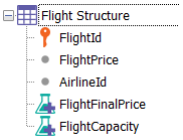
Para a fórmula horizontal do exemplo, não parece muito clara a utilidade de defini-la redundante, pelo menos em termos de desempenho, a menos que seja necessário, por exemplo, um panel ou web panel onde o usuário queira filtrar os voos exibidos em um grid por preço de voo.
O caso se tornaria mais interessante se o cálculo horizontal envolvesse muitas tabelas da estendida. Ou, ainda mais, se fosse uma fórmula horizontal daquelas que se resolvem invocando um procedimento que realiza um cálculo complexo.

GeneXus

Formulas redundancy








```
For each Flight
     FlightPrice *= 1.1
endfor
```

```
For each Flight.Seat
     where FlighId = 4
     Delete
endfor
```

```
new
     FlightId = 12503
     FlightPrice = 500
     AirlineId = find(AirlineId, AirlineName = "Air Europe")
     new
          FlighSeatId = 1
          FlightChar = 'B'
          FlightSeatLocation = Location.Window
     endnew
endnew
```

Se a vantagem de redundar atributos fórmula é clara, qual é sua desvantagem?

Que devem ser mantidos sempre atualizados, e essa atualização tem um custo. Se através da transação Flight (ou de seu business component) é modificado o valor de FlightPrice, como sempre, a fórmula horizontal que o envolve é disparada novamente. O mesmo se for adicionada uma linha ou se for removida uma, a fórmula FlightCapacity é disparada novamente. Em ambos os casos é armazenado seu valor nos atributos redundantes e o desenvolvedor não precisa se preocupar em fazer isso.

Porém, como no caso das redundâncias referenciais, se a modificação for feita por procedimento, como nestes exemplos, GeneXus não fará nada. Aqui o desenvolvedor terá que se preocupar em atualizar os atributos redundantes adicionando código.

GeneXus

Formulas redundancy

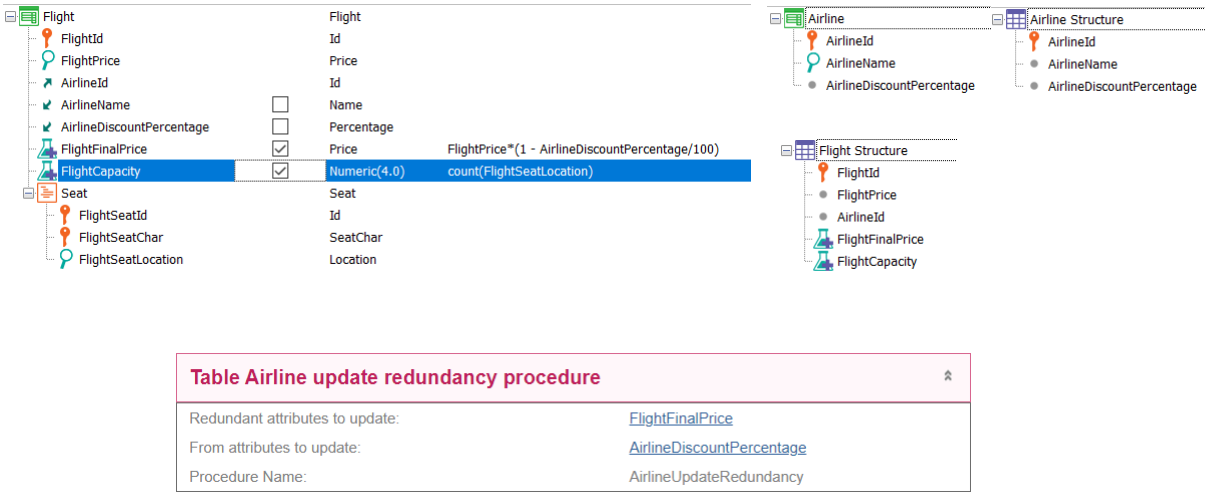


Table Airline update redundancy procedure

Redundant attributes to update:	FlightFinalPrice
From attributes to update:	AirlineDiscountPercentage
Procedure Name:	AirlineUpdateRedundancy

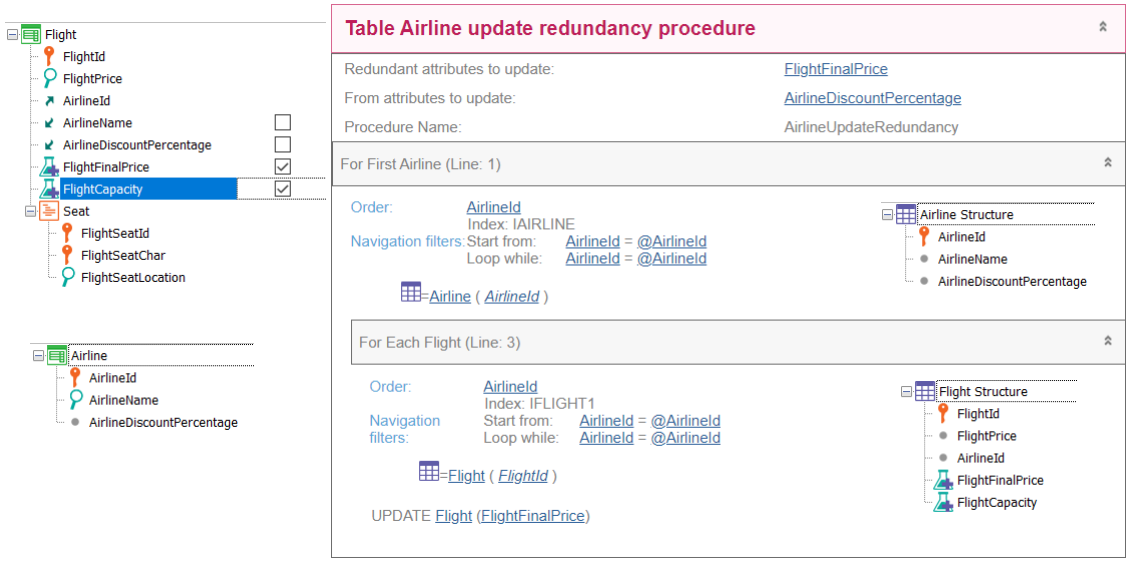
E o que acontece se o usuário entra na transação Airline e modifica para uma companhia aérea o percentual de desconto? (ou se fizer isso por meio do business component).

Em Flight, a fórmula redundante FlightFinalPrice depende desse valor, portanto, deveria ser atualizado o valor redundante armazenado para todos os registros de Flight que pertencem a essa companhia aérea.

GeneXus criará na reorganização o procedimento cujo nome é a concatenação do nome da tabela, neste caso Airline, e UpdateRedundancy, assim como fez com as redundâncias referenciais.

GeneXus

Formulas redundancy

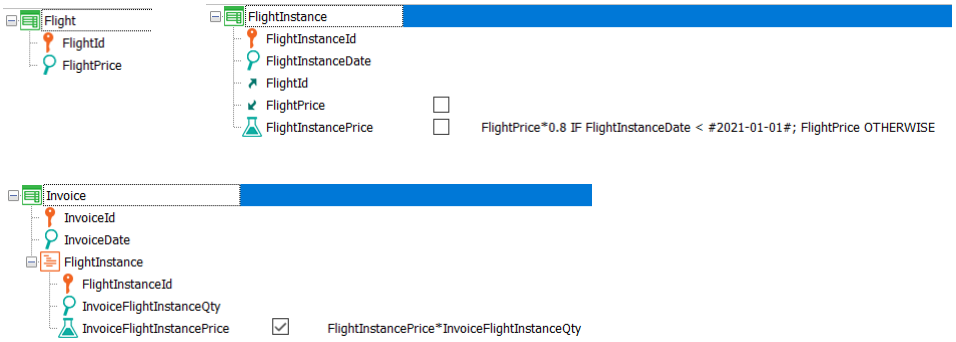


Este procedimento será invocado a partir da transação Airline passando-lhe o Id e o procedimento acessará o registro correspondente, e então percorrerá a tabela Flight filtrando por essa companhia aérea e disparará novamente a fórmula FlightFinalPrice, armazenando seu resultado na tabela.

Tudo isto significa um custo de desempenho, por isso é necessário avaliar muito bem quando convém redundar uma fórmula e quando não.

GeneXus

Formulas redundancy: limitations



Manter fórmulas redundantes por enquanto tem suas limitações.

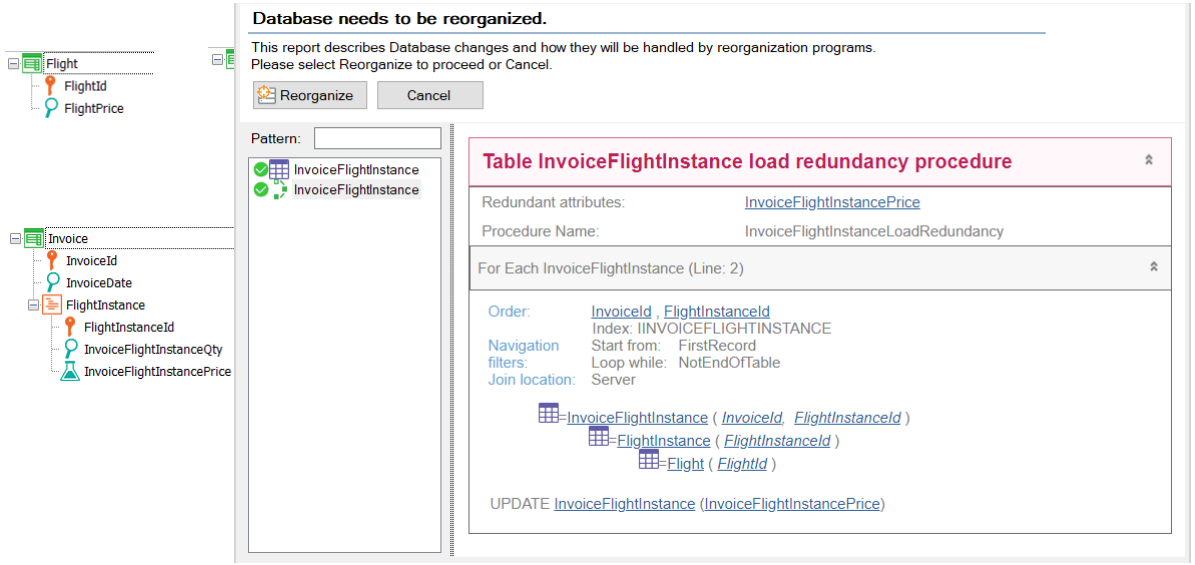
Por exemplo, se temos estas três transações relacionadas, onde a transação Flight registra as informações genéricas de um voo, como seu preço, mas é a transação FlightInstance a que corresponde ao voo real, em uma data determinada. Nesta, o preço do voo real é obtido com uma fórmula que leva em consideração o preço de lista do voo de acordo com a data.

E então temos uma transação para registrar o faturamento de voos. As linhas registram os voos para os quais estão sendo compradas passagens e quantas passagens para cada um. Em seguida, esta fórmula calcula o preço de cada linha, utilizando para isso o atributo fórmula inferido de FlightInstance, que por sua vez é uma fórmula, como dissemos.

Suponhamos que queremos tornar redundante esta fórmula na tabela InvoiceFlightInstance. Se apenas redundássemos ela...

GeneXus

Formulas redundancy: limitations

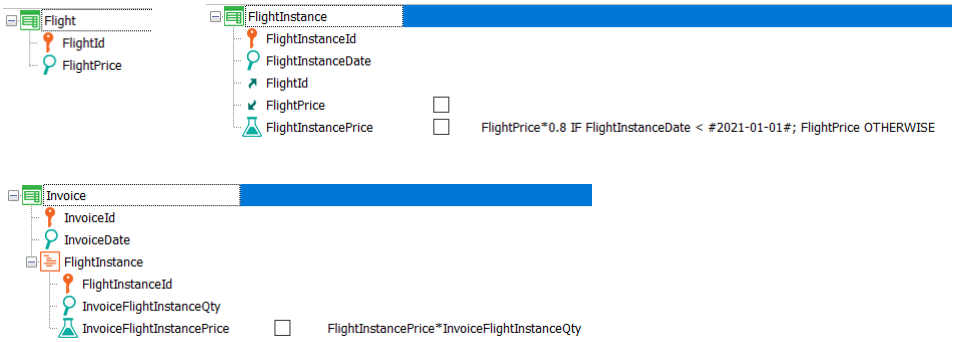


...o procedimento de carga da redundância será resolvido corretamente, indo para a tabela InvoiceFlightInstance e disparando a fórmula horizontal para cada registro e armazenando-a.

Mas o que não acontecerá é que sejam criados procedimentos de atualização das redundâncias.

GeneXus

Formulas redundancy: limitations



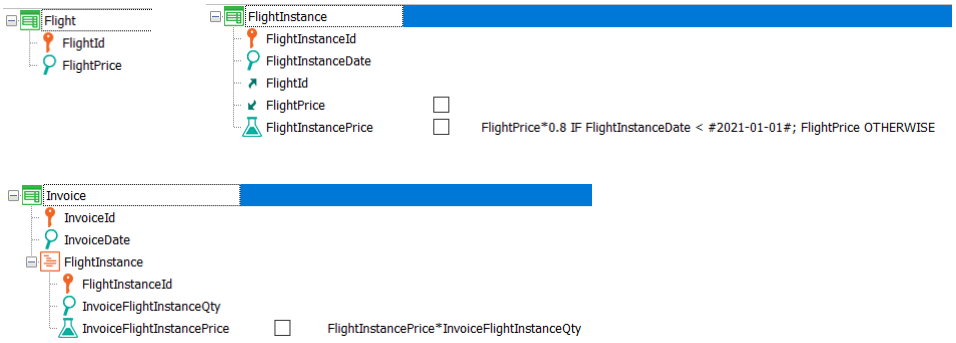
O que queremos dizer?

Isto significará que se o usuário executa a transação FlightInstance e altera, por exemplo, o valor de FlightInstanceDate (suponhamos que de uma data menor que esta, para uma maior), como a fórmula horizontal será disparada novamente e alterará seu valor, esperaríamos que a transação invoque um procedimento, FlightInstanceUpdateRedundancy, que vá para a tabela FlightInstance e atualize as redundâncias correspondentes. Mas não fará isso.

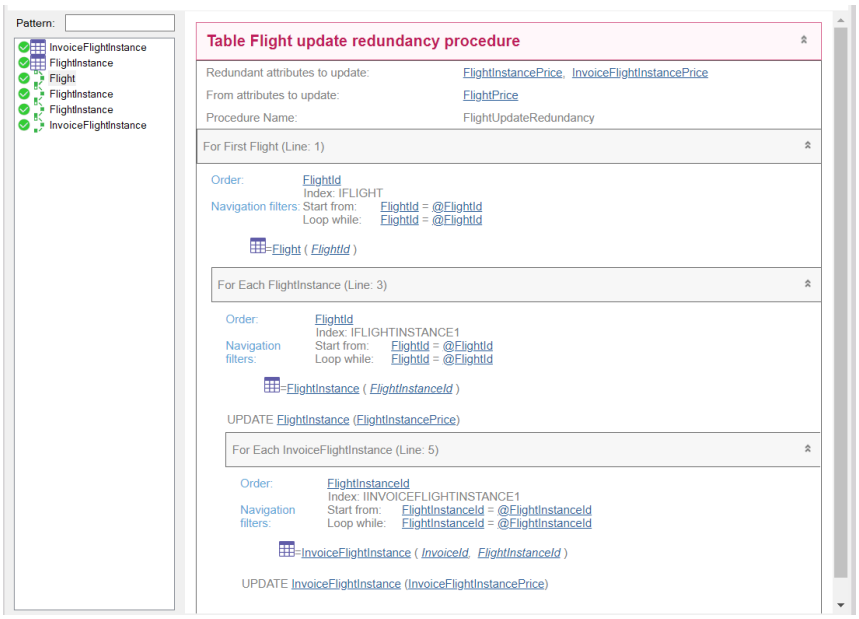
Da mesma forma, esperaríamos que se o usuário modifica o valor do atributo FlightPrice através da transação Flight, também exista um procedimento FlightUpdateRedundancy que vá para a tabela InvoiceFlightInstance para recalcular e armazenar novamente as redundâncias do atributo fórmula que devam ser modificados pela modificação de FlightInstancePrice, quando corresponda. Também não fará isso.

GeneXus

Formulas redundancy: limitations

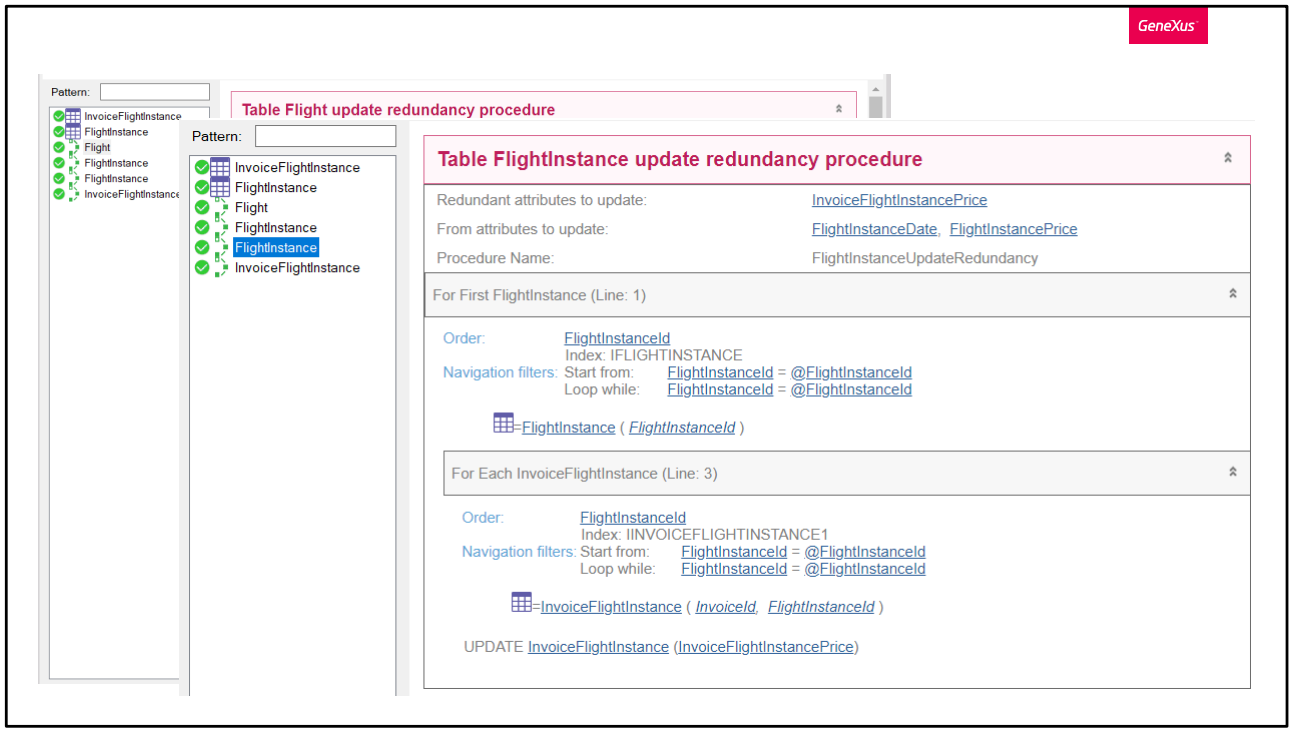


Mas e se também redundamos a fórmula horizontal envolvida?



Aqui veremos ao reorganizar que além de informar as alterações nas duas tabelas para adicionar as duas fórmulas como redundantes, e os dois procedimentos de carga das redundâncias, serão criados ambos os procedimentos de UpdateRedundancy.

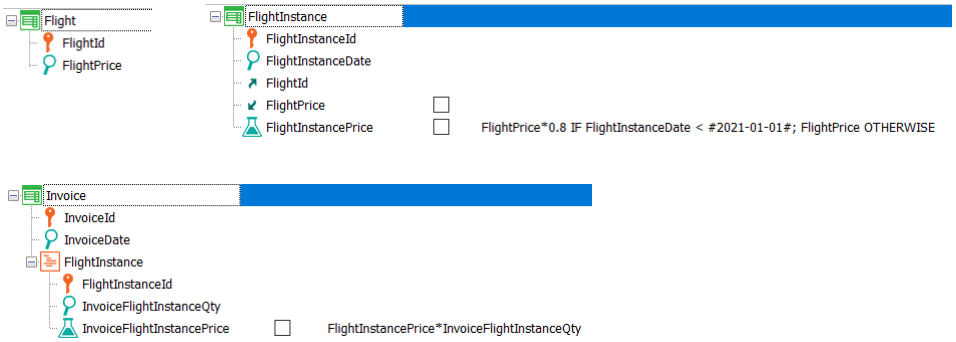
O primeiro que é disparado a partir da transação Flight quando é alterado o preço, que irá atualizar a primeira fórmula redundante em FlightInstance e então, a partir dela, a segunda, em InvoiceFlightInstance.



E o segundo procedimento, que será executado quando a partir da transação FlightInstance é modificada a data do voo, que irá atualizar o atributo redundante em InvoiceFlightInstance.

GeneXus

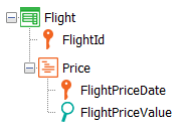
Formulas redundancy: limitations

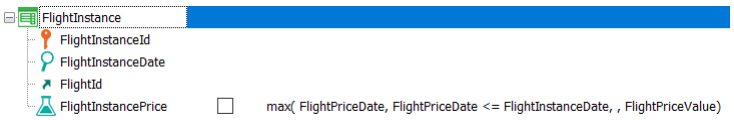


Portanto, se temos uma fórmula que em seu cálculo envolve outra, para que a manutenção da sua redundância seja mantida automaticamente, precisamos definir a fórmula interna como redundante também.

GeneXus

Formulas redundancy: limitations





FlighId	FlightPriceDate	FlighPriceValue
1	01/01/2022	800
1	02/02/2022	1000
1	04/04/2022	950

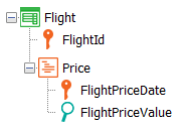
FlightId: 1
FlightInstanceDate: 03/03/2022
FlightInstancePrice:

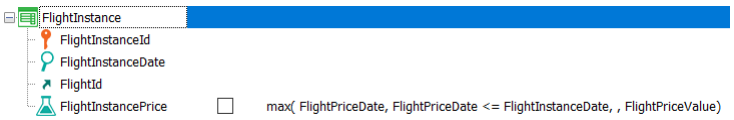
Mas há mais limitações. Por exemplo, as fórmulas aggregate/select na maioria das vezes não poderão ser mantidas através desses procedimentos de atualização.

Por exemplo, se mantivermos em Flight uma lista de preços do voo por data, em FlightInstance teremos que calcular o preço de um voo específico de acordo com o preço que lhe corresponda à data do voo real. Ou seja, calculamos o preço de acordo com uma fórmula max. Assim, tendo estes dados, se está sendo inserida uma instância do voo 1, com esta data, a fórmula max ficará com este registro...

GeneXus

Formulas redundancy: limitations





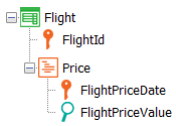
FlighId	FlightPriceDate	FlighPriceValue
1	01/01/2022	800
1	02/02/2022	1000
1	04/04/2022	950

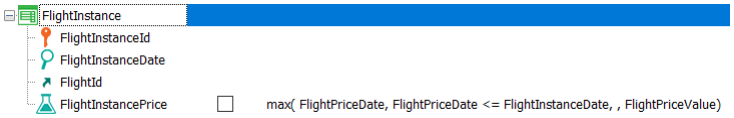
FlightId: 1
FlightInstanceDate: 03/03/2022 LIST
FlightInstancePrice:

... então retornará o valor 1000 para o preço do voo. Se estivesse imprimindo em uma lista, mostraria 1000.

GeneXus

Formulas redundancy: limitations





FlightId	FlightPriceDate	FlighPriceValue
1	01/01/2022	800
1	02/02/2022	1500
1	04/04/2022	950

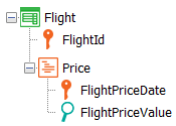
FlightId: 1
FlightInstanceDate: 03/03/2022 LIST
FlightInstancePrice: 1500

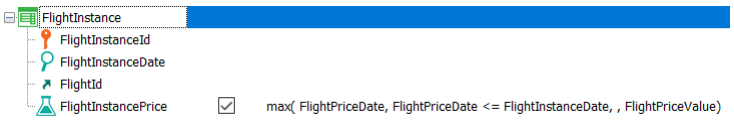
FlightPriceUpdateRedundancy?

Se então o usuário entra na transação Flight e para a linha correspondente a este registro altera 1000 para 1500, como a fórmula é virtual, ao executar novamente a lista, será disparado o max e será listado 1500.

GeneXus

Formulas redundancy: limitations





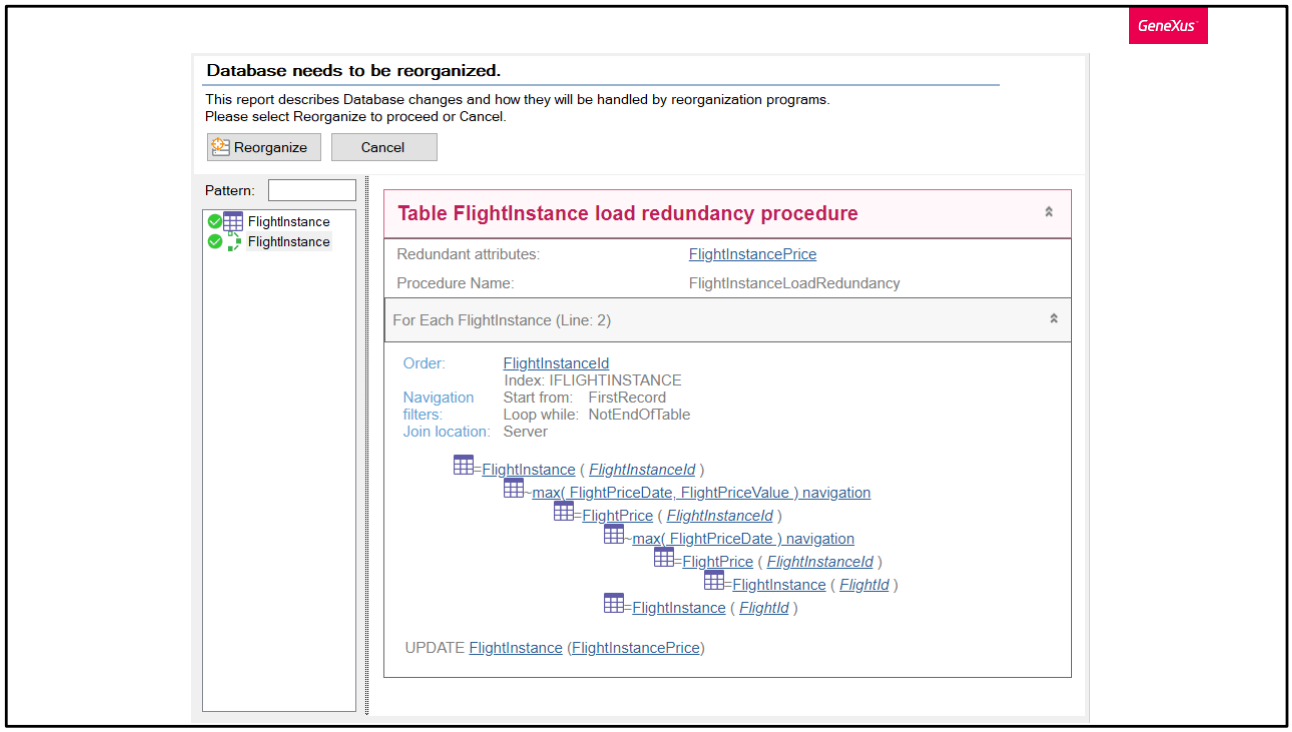
FlightId	FlightPriceDate	FlighPriceValue
1	01/01/2022	800
1	02/02/2022	1500
1	04/04/2022	950

FlightId: 1
FlightInstanceDate: 03/03/2022 LIST
FlightInstancePrice: 1500

FlightPriceUpdateRedundancy?

Mas e se tornarmos a fórmula max redundante? Esperaríamos que fosse criado um procedimento de atualização da redundância para a tabela FlightPrice, de forma que quando fosse alterado o valor de FlightPriceValue através da transação Flight, fosse acessada a tabela FlightInstance procurando quais registros seriam afetados pela alteração para modificar o valor de FlightInstancePrice. Mas quando se tenta pensar em quais seriam esses registros afetados, vemos como é difícil saber.

GeneXus, portanto, não criará neste caso esse programa de manutenção da redundância. Vemos isso claramente quando definimos o atributo como redundante e vemos o relatório de reorg.

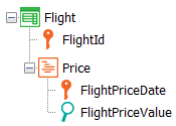


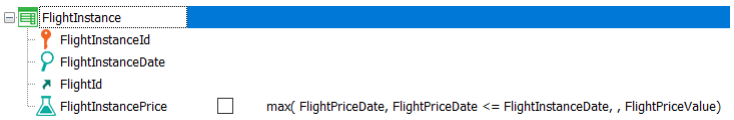
Apenas criará o procedimento de carga da redundância, mas não o de atualização.

Portanto, a recomendação é sempre inspecionar este relatório de análise de impacto para saber quais fórmulas redundantes serão mantidas e quais não serão.

GeneXus

Formulas redundancy: limitations





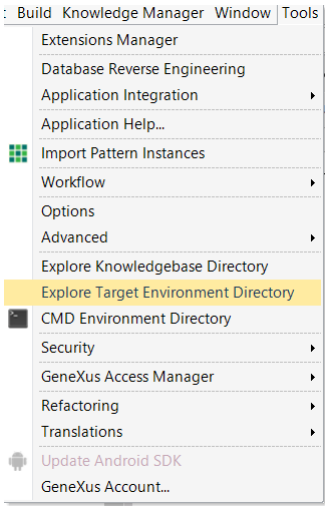
FlightId	FlightPriceDate	FlighPriceValue
1	01/01/2022	800
1	02/02/2022	1500
1	04/04/2022	950

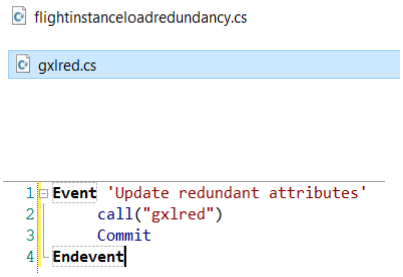
FlightId: 1
FlightInstanceDate: 03/03/2022 LIST
FlightInstancePrice: 1500

Como observação lateral: claro, se o usuário entra na transação FlightInstance e modifica o valor de FlightInstanceDate, lá será mantido atualizado o atributo redundante, pois dentro da transação a fórmula será disparada normalmente e será armazenado seu valor. O problema surge quando um atributo envolvido no cálculo da fórmula é modificado a partir de outra transação, não a da fórmula. Neste caso, a partir da transação Flight.

GeneXus

Rebuild redundancy





Se precisássemos ter esse atributo redundante de qualquer maneira, sabendo que não será mantida atualizada automaticamente a redundância a partir de Flight, o que sempre podemos fazer é invocar o procedimento de carga das redundâncias criado por GeneXus.

Se formos procurar os arquivos no diretório do environment, encontraremos para cada tabela com atributos redundantes o procedimento de carga das redundâncias da tabela. Aquele que termina com LoadRedundancy, nome de tabela load redundancy. Em nosso último caso, o flightinstanceloadredundancy, que podemos invocar como uma chamada a um procedimento externo.

Também, embora não apareça listado no relatório de análise de impacto, GeneXus cria um procedimento denominado gxlred, por GeneXus Load Redundancy, que o que faz é invocar cada um dos procedimentos LoadRedundancy de cada tabela com redundâncias.
Então se em algum momento quisermos ter certeza de que todas as redundâncias definidas na KB sejam atualizadas, podemos, por exemplo, invocar este programa a partir de um evento.

GeneXus

Other limitations in defining redundancies

To define as redundant a formula that is already a formula, we must first define that other form

Redundant aggregate formulas that add more than one level of nested redundant formulas will not be pr

Subtypes cannot be defined redun

To change the definition of a formula that is redundant, you must first remove th

Existem algumas outras limitações para definir atributos redundantes, que precisamos considerar.

Aqui deixamos algumas listadas, que podem ser consultadas com mais detalhes na wiki.

GeneXus™

training.genexus.com
wiki.genexus.com