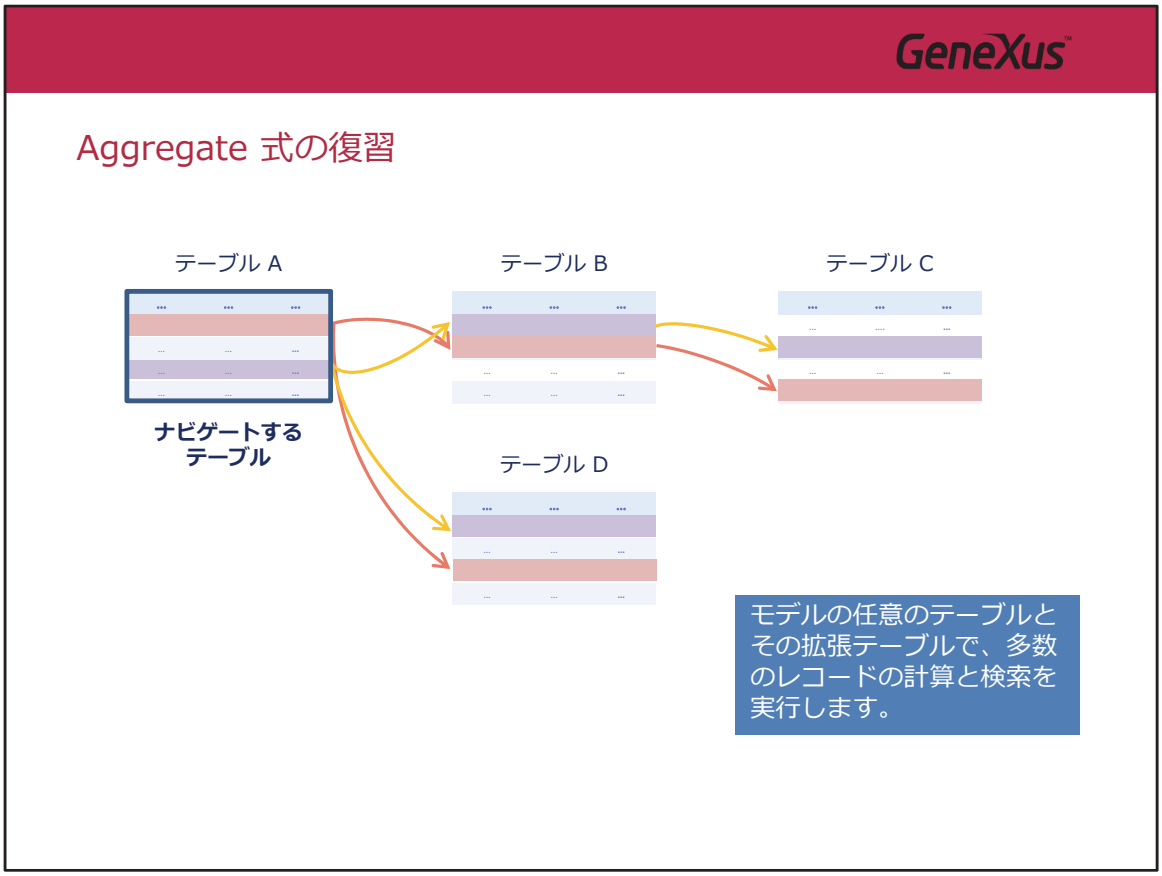
Aggregate 式

GeneXus™

ここでは、Aggregate 式の詳細とその使用例について取り上げます。

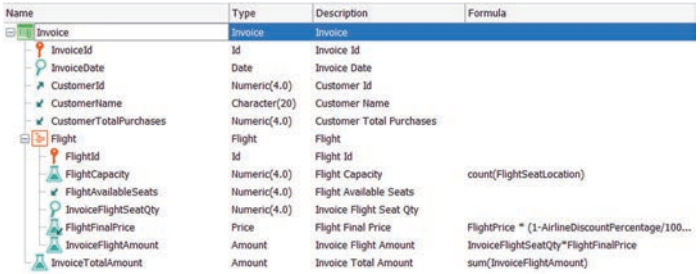


まず、概念を復習しましょう。

Aggregate 式を使用すると、テーブル内の多数のレコードと、参照されるテーブルの拡張テーブルの関連する値の計算や検索を実行できます。

Aggregate 式の復習

Aggregate 式:
- Count
- Sum
- Average
- Max
- Min
- Find



Name	Type	Description	Formula
Invoice	Invoice	Invoice	
InvoiceId	Id	Invoice Id	
InvoiceDate	Date	Invoice Date	
CustomerId	Numeric(4.0)	Customer Id	
CustomerName	Character(20)	Customer Name	
CustomerTotalPurchases	Numeric(4.0)	Customer Total Purchases	
Flight	Flight	Flight	
FlightId	Id	Flight Id	
FlightCapacity	Numeric(4.0)	Flight Capacity	count(FlightSeatLocation)
FlightAvailableSeats	Numeric(4.0)	Flight Available Seats	
InvoiceFlightSeatQty	Numeric(4.0)	Invoice Flight Seat Qty	
FlightFinalPrice	Price	Flight Final Price	FlightPrice * (1-AirlineDiscountPercentage/100...
InvoiceFlightAmount	Amount	Invoice Flight Amount	InvoiceFlightSeatQty*FlightFinalPrice
InvoiceTotalAmount	Amount	Invoice Total Amount	sum(InvoiceFlightAmount)

構文:

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

オプション　オプション　オプション　オプション

Aggregate 式にはコンテキストは必要ありませんが、コンテキストがある場合は結果をフィルタできます。

Aggregate 式では、レコードの件数を数える (Count)、合計する (Sum)、または平均を求める (Average) ことができます。また、検索を実行して一連のレコードやテーブルの多数のレコードの項目属性の最大値 (Max) や最小値 (Min) を見つけたり、特定の条件を満たす最初のレコードの項目属性の値を見つける (Find) ことができます。

この構文では、**AggregateExpression** は、AggregateCondition を満たすレコードを対象に、検索、最大値、最小値、合計、または平均を求めます。項目属性 (式項目属性を含む)、定数、変数 (ユーザーが作成した変数はインライン式でのみ使用可能) を含めることができます。Count の場合のみ、エクスプレッションではなく項目属性になります。Sum と Average の場合、AggregateExpression の結果は数値でなければなりません。

AggregateCondition は、レコードを Aggregate で考慮するために検証する必要がある条件です。項目属性、定数、変数 (ユーザーが作成した変数はインライン式でのみ使用可能) を含めることができます。これはオプションであるため、含めなくてもかまいません。

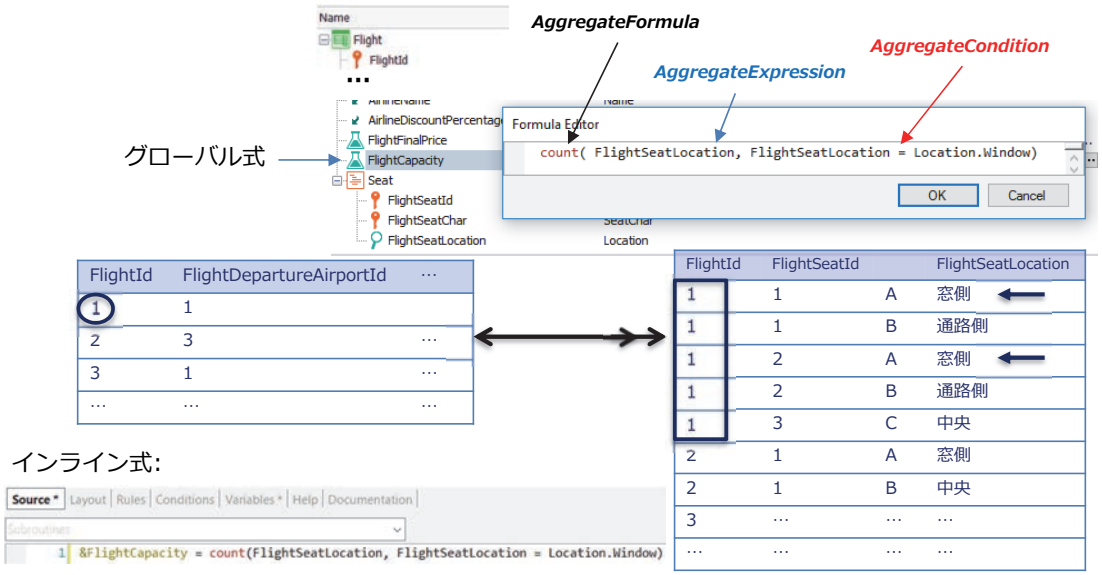
Aggregate を実行するレコードが見つからない場合は、既定値が返されます。見つからない理由として、AggregateCondition を満たすレコードがない、暗示的な AggregateCondition を満たさない、またはテーブルが空であることが考えられます。既定値は定数で、これもオプションです。

ReturnedValue は項目属性で、その値は、AggregateCondition を満たすレコードが見つかったときに式によって返されます。ReturnedValue は値の場合もありますが (既定値が返される場合など)、通常は項目属性です。この 4 つ目のパラメーターはオプションで、一部の式にだけ含まれます。

TriggeringCondition は、その前にある AggregateExpression で式を計算するかどうかを判断する条件です。
これはオプションで、関連付けられたテーブルとその拡張テーブルに属する項目属性のみ使用できます。TriggeringCondition はグローバル Aggregate 式でのみ使用できます。インライン式の場合、これは式の定義には含まれず、式のトリガー条件がコードの条件節 (if、else、do case など) で指定されるためです。

Horizontal 式には評価対象となるコンテキストが必要ですが、Aggregate 式の場合は必ずしも必要ではありません。ただし、コンテキストテーブルがある場合は、式で明示的に指定されたレコードすべてが考慮されるのではなく、コンテキストの暗示的な条件にも一致するもののみが考慮されます。

例: Count



たとえば、Flight トランザクションの FlightCapacity 項目属性は Count 式で、FlightSeat テーブルを参照してフライトの座席数をカウントします。

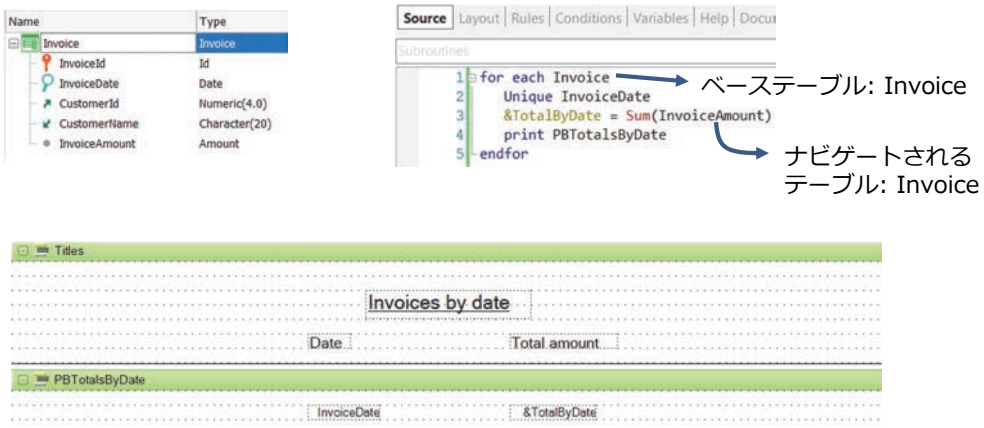
項目属性に関連付けられているテーブルは Flight テーブルで、このテーブルは、Count がナビゲートする FlightSeat テーブルと 1 対 N の関係です。そのため、関連するレコード、つまり、Flight でインスタンス化されるフライトに対応する座席のみがカウントの対象になります。関係がない場合、FlightSeat テーブルのすべてのレコードがカウントされます。

さらに、窓側の座席のみをカウントするように指定したため、関連する一連のレコードの中で、フィルタ条件を満たすもののみがカウントされます。

これは Count 式であるため、AggregateExpression はレコードがカウントされるテーブル (この場合は FlightSeatLocation) を決定する項目属性にすぎません。AggregateCondition は座席が満たす必要があるフィルタ (窓側) です。また、これは Max 式でも Min 式でもないため、既定値も戻り値もなく、トリガー条件も定義されていません。

プロシージャーの [Source] では、インライン式と同じ定義になっています。この場合、いわば式が「緩い」ため、暗示的な条件はありません。つまり、グローバル式の場合やこの For Each 内に配置した場合とは異なり、特定のフライトではなく、すべてのフライトの窓側の座席がカウントされます。

例: 関連付けられたテーブル = ナビゲートされるテーブル



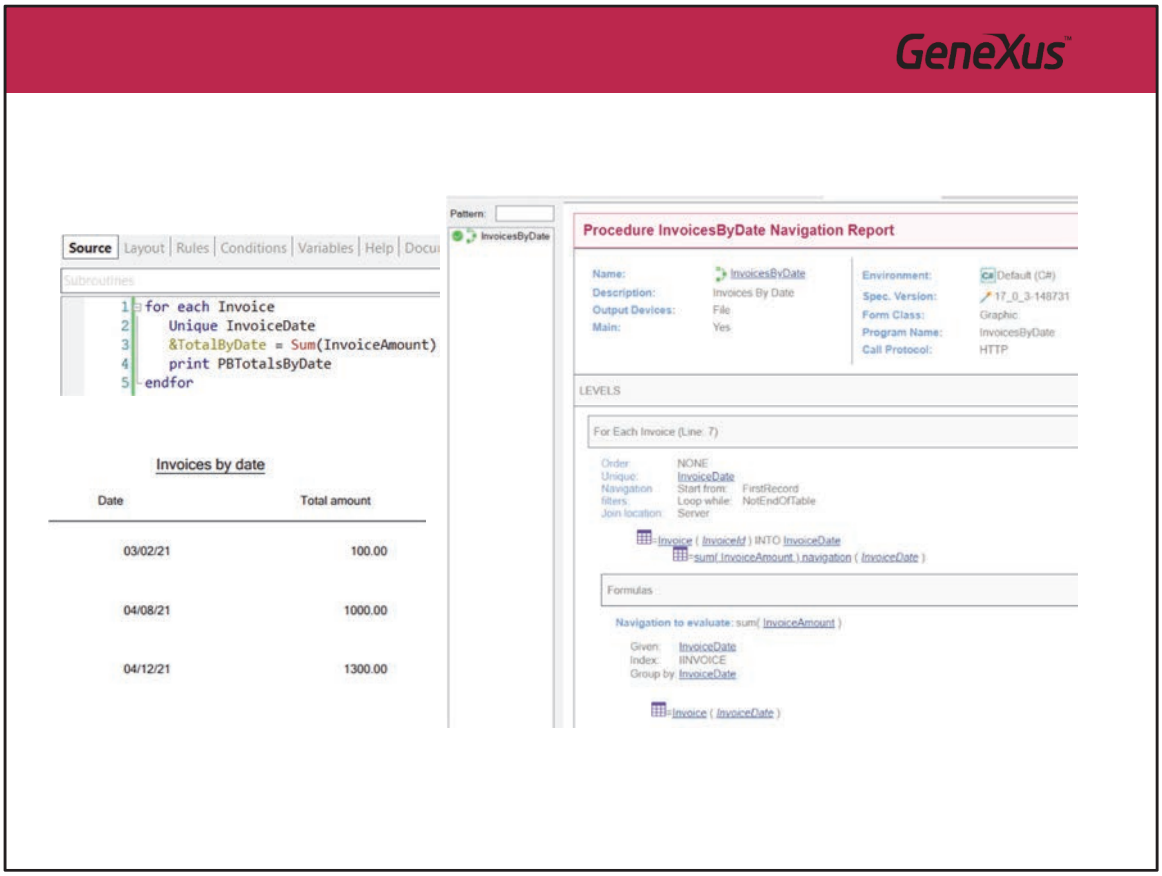
式が特定のコンテキスト内にあり、式によりナビゲートされるテーブルがそのコンテキストテーブルと一致する特別なケースを見てみましょう。

この例では、各請求書の日付について、その日付のすべての請求書の合計を出力します。

Sum 式がナビゲートするテーブルは Invoice で、これは For Each のベーステーブル (こちらも Invoice) と一致します。

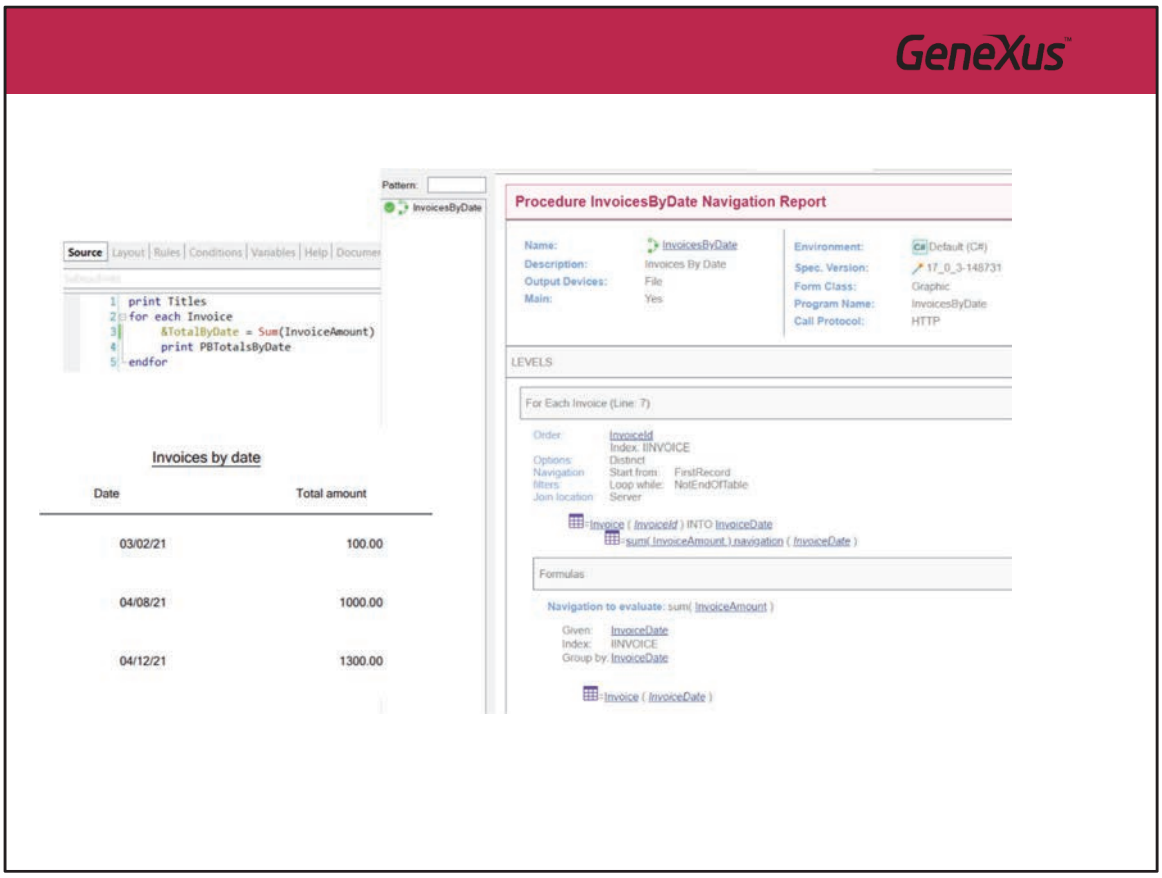
この場合、InvoiceDate による Unique 節があるため、GeneXus は For Each と Sum 式の両方で、請求書の日付に基づいて情報をグループ化します。つまり、Sum 式には InvoiceDate 項目属性による暗示的な等価条件が含まれ、それが取得値で考慮されます。

まるで InvoiceDate によるコントロールブレーク処理のようです。ナビゲーション表示を確認してみましょう。



For Each のベーステーブルは Invoice で、Unique は InvoiceDate に基づきます。

下に、InvoiceDate によってグループ化され (Group by)、InvoiceDate が指定されている式が確認できます。したがって、目的どおりに、その日付のレコードのみが加算されます。



Unique 節を削除したらどうなるでしょうか。
Unique 節がない場合に、特定の日付の請求書の請求書合計のみを加算するように GeneXus に指示するにはどうすればいいでしょうか。
つまり、式を記述する代わりに、For Each を使用する計算に分解します。

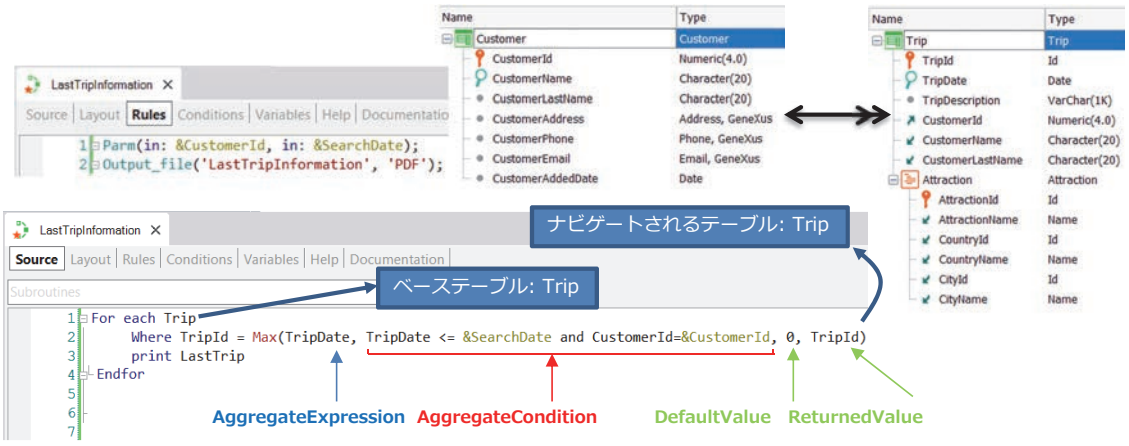
出力はどうなるでしょうか。コントロールブレークはありますが、InvoiceDate に基づくブレーク基準は指定しません。外部の For Each の Order 節を指定していないため、主キーが選択されます。また、1 つのレコード (InvoiceId) のみがある内部の For Each 内に常にとどまります。

では、GeneXus のナビゲーション表示を見てみましょう。

Unique 節は表示されていませんが、Distinct 節が自動的に追加されているため同じ結果になります。GeneXus は、請求書の日付別にグループ化する必要があることを検知し、それをコード生成に反映させて、期待する内容を表示することができます。

GeneXus は開発者が実行したいことを自動的に検知することができますが、コードのプログラムで作成側の意図が明確になるように、Unique 節を含めることをお勧めします。

例 2: 関連付けられたテーブル = ナビゲートされるテーブル



同じようにテーブルをナビゲートし、同じテーブルに対して Aggregate を実行するケースを見てみましょう。この場合、使用する Aggregate は Max 式です。

たくさん旅行をした顧客がいて、特定の日付の直前の旅行のデータを取得するとします。つまり、その日付より前の、最後の旅行です。

情報は旅行データのリストとして出力するとします。プロシージャーは、情報を求めている顧客の ID と検索日をパラメーターとして受け取ります。

プロシージャーの [Source] には、Trip テーブルを参照する For Each があり、Max 式で返される TripId に基づき Where でフィルタされます。次に、その旅行に対応するデータが出力されます。

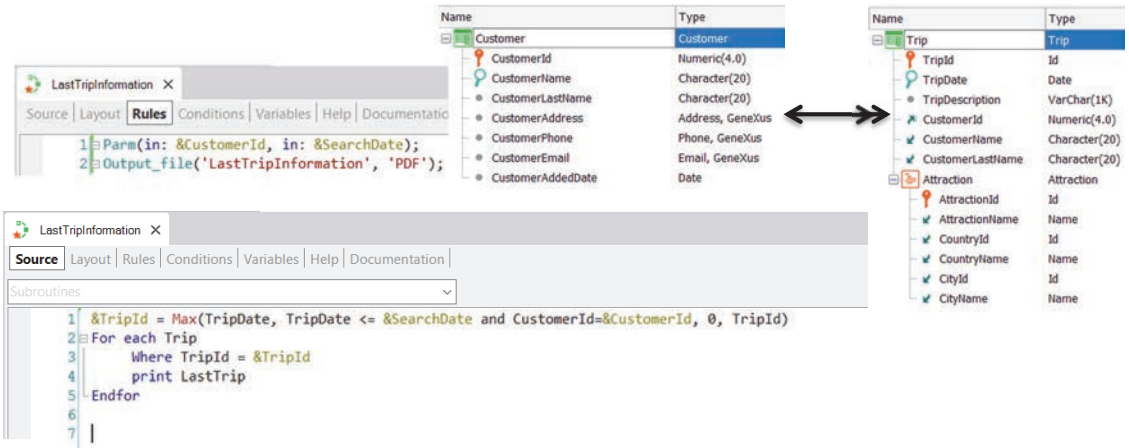
Max 式で最大値を求める項目属性は TripDate です。特定の日付の直前の旅行を取得したいため、検索日より小さいか、検索日と等しい有効な最大の日付になります。AggregateCondition で、旅行が特定の日付以下 (特定の日を含む) の日付を持ち、さらに情報を求めている顧客に属している必要があることを指定します。これらの条件を満たす旅行が見つからない場合は既定値の 0 が返され、見つかった場合はその旅行の ID が返されます。

プログラムの内容を詳しく見てみましょう。Trip テーブルに対して For Each が繰り返され、Max 式のコンテキストが設定されます。
Trip テーブルに対して Max 式も繰り返されますが、両方のテーブルが関連しているため、コンテキストによって自動フィルタが設定されます。この場合は同じテーブルであるため、For Each に配置された TripId によって式がフィルタされます。
したがって、常に、For Each が配置された場所の TripId が返され、何も記述していない場合と同じになります。これは期待した結果ではありません。

つまり、Aggregate 式がコンテキストのテーブルと同じテーブルをナビゲートする場合は注意が必要です。ほとんどの場合、特に指定しない限り、主キーによるフィルタが適用され、ナビゲートする必要があるテーブルがナビゲートされず、配置した場所にある同じレコードが常に取得されます。

GeneXus

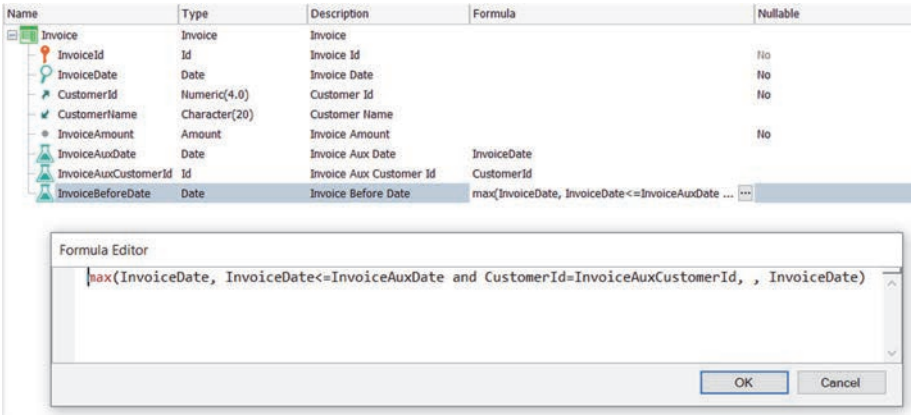
例 2: 関連付けられたテーブル = ナビゲートされるテーブル



これを解決するには、まず、式を実行して必要な条件を満たす旅行の ID を検索し、次にその ID 値で For Each をフィルタします。

GeneXus

例 3: 関連付けられたテーブル = ナビゲートされるテーブル



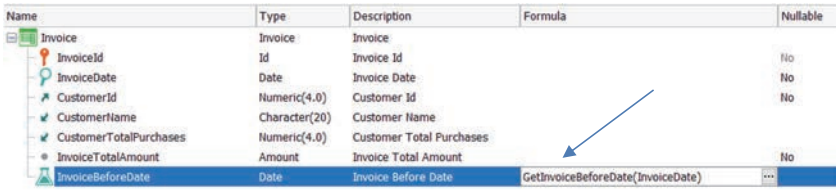
前の例と似たケースで、グローバル式を使用する場合を見てみましょう。ある顧客の特定の請求書について、同じ顧客の前の請求書の日付を確認したいとします。そのためには、先ほど定義したものと同様に Max 式を使用しますが、今回は項目属性をグローバル式として定義します。

この場合も、式に関連付けられた同じテーブルをナビゲートする式を定義しようとしているため、式がコンテキストによってフィルタされ、1 つのレコードしかナビゲートできません。

グローバル式であるこのケースと、インライン式である先ほどのケースの違いは、先ほどのケースでは、コンテキストテーブルの外部で式をトリガーして、暗示的なフィルタのブレーク処理を実行できましたが、グローバル式の場合はできないことです。したがって、同じテーブルをナビゲートするグローバルの Aggregate 式を定義することはできません。

GeneXus

例 3: 関連付けられたテーブル = ナビゲートされるテーブル



Aggregate 式にはコンテキストは必要ありませんが、
コンテキストがある場合は結果をフィルタできます。

このためには、Horizontal 式として定義して、プロシージャー内で計算を実行する必要があります。

要するに、これは前の説明を裏付けています。つまり、Aggregate 式の場合は (Horizontal 式とは異なり) コンテキストを評価する必要はありません。ただし、コンテキストが存在する場合は、式で明示的に設定されたすべてのレコードが考慮されるとは限らず、コンテキストから発生する暗示的な条件にも一致するもののみが考慮されます。

重要なのは、最後の例で示したように、式に求められる目的がコンテキストによって無効にならないようにすることです。