ネストされた For Each の詳細

ケースおよびナビゲーション

GeneXus™

ネストされた For Each コマンドがある場合、GeneXus は最初にそれぞれのベーステーブルを確認します。

開発者がベーストランザクションを指定していれば、この確認が即座に行われます。指定していないときには、それぞれの For Each コマンドに含まれている項目属性に基づいて、GeneXus がベーステーブルを判断する必要があります。こちらは複雑なので、この章では説明しません。

次に、GeneXus は、複数のクエリを解決するために必要なナビゲーションを定義します。次の 3 つのうちいずれかが適用されます。

- 結合
- デカルト積
- コントロールブレーク

GeneXus

ネストされた For Each

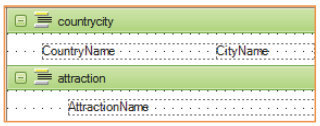
```
For each Attraction order AttractionName
      Where CountryId = 1
      &attractionName = AttractionName
      &attractionDescription = AttractionDescription

      For each Categories
            &categoryName = CategoryName
      Endfor

      When none
      ....
EndFor
```

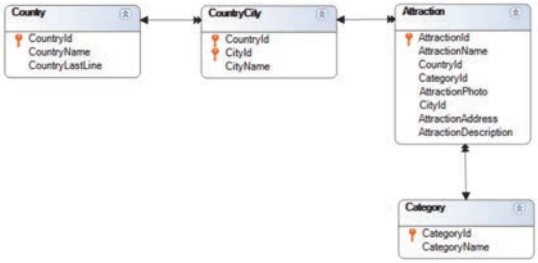
GeneXus では、ネストされた For Each コマンドが検出されると、それぞれのコマンドのベーステーブルが確認されます。この確認は、最も外側のコマンドから内側のコマンドの順に行われます。その後、ナビゲーションの方法が決定されます。

For Each コマンドでは、指定されたベーストランザクションと、その For Each コマンドに属する項目属性のみが使用されます (Order、Where など)。また、ネストされた For Each コマンドに含まれるものを除いて、その本文に含まれているものも使用されます。つまり、ネストされた For Each コマンドを取り除いたシンプルな For Each コマンドと同じようにベーステーブルが判断されます。When none 節の項目属性は考慮されません。すべての項目属性は、検出されたベーステーブル (ベーストランザクションと関連付けられたベーステーブルと一致する) の拡張テーブルに属している必要があります。この条件を満たさない項目属性は到達不可能であるため、「インスタンス化可能」にはなりません。

ネストされた For Each

```
For each Country.City
      print countrycity
      For each Attraction
            Print attraction
      Endfor
EndFor
```





この例では、次の順序で処理が行われます。

1) 外側の For Each コマンドのベーステーブルが確認されます。指定されたベーストランザクション (Country.City) が確認され、printblock に含まれる項目属性 (CountryName と CityName) がその拡張テーブルに属しているかどうかがチェックされます。属していない場合はナビゲーション表示に警告が表示され、一部の項目属性がその For Each コマンドの拡張テーブルから到達できないため、インスタンス化できないことが示されます。このケースでは、CountryName と CityName は、その For Each コマンドのベーステーブルである CountryCity の拡張テーブルに属しています。

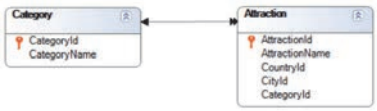
2) ネストされた For Each コマンドのベーステーブルが確認されます。Attraction ベーストランザクションと、printblock に含まれる AttractionName 項目属性が確認されます。ベーストランザクションが記述されていなかった場合は、外側の For Each コマンドの項目属性に関連するものが確認されて、内側の For Each コマンドのベーステーブルが判断されますが、このケースには該当しません。このように、スタンドアロンの For Each コマンドと同じようにベーステーブルが判断されます。そのため、ベーステーブルは Attraction になります。

異なるベーステーブル

外側の For Each ⟷ ネストされた For Each

結合

```
For Each Category
    ...
    For each Attraction
        ...
    Endfor
Endfor
```



Category テーブル

CategoryId	CategoryName
1	美術館
2	遺跡
3	観光名所

Attraction テーブル

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	ルーブル美術館	2	1	1
2	万里の長城	3	1	3
3	エッフェル塔	2	1	2
4	紫禁城	3	1	3
5	コルコバードのキリスト像	1	2	2

ここまでは、ベーステーブルの確定に関する For Each コマンドの例を見てきました。もう少し詳しく見てみましょう。

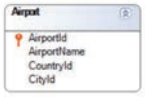
ベーステーブルが異なる場合、それらに直接的または間接的な 1 対 N の関係がある場合とない場合があります。関係がある場合、メインの For Each のそれぞれのレコードについて、ネストされた For Each は関連付けられた N 個のレコードのみを対象とします。別のテーブルのデータを利用してデータを切り取るナビゲーションは、結合と呼ばれます。

異なるベーステーブル

外側の For Each ⟶ ネストされた For Each

デカルト積

```
For each Airport
      .....
      For each Attraction
            ....
      Endfor
EndFor
```



Airport テーブル

AirportId	AirportName	CountryId
1	グアルーリョス	2
2	シャルル・ド・ゴール	1
3	テーゲル	3

Attraction テーブル

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	ルーブル美術館	2	1	1
2	万里の長城	3	1	3
3	エッフェル塔	2	1	2
4	紫禁城	3	1	3
5	コルコバードのキリスト像	1	2	2

2 番目の、関係がないケースでは、メインの For Each で確認される各レコードについて、ネストされた For Each は別のテーブルのレコードに対して全件を対象とします。これは、テーブル間の関係が検出されなかったためです。このナビゲーションはデカルト積と呼ばれます。

ベーステーブルが一致

コントロールブレーク

```
For each Attraction order CategoryId
      .....
      For each Attraction
            ....
      Endfor
Endfor
```

Attraction テーブル

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	ルーブル美術館	2	1	1
6	ポンピドゥーセンター	2	1	1
7	ケ・ブランリ	2	1	1
3	エッフェル塔	2	1	2
5	コルコバードのキリスト像	1	2	2
2	万里の長城	3	1	3
4	紫禁城	3	1	3

ベーステーブルが同じである場合は、コントロールブレークと呼ばれるナビゲーションが実行されます。このナビゲーションは、テーブルのデータをグループ化する必要があるときに行われ、そのグループの共通データを確認する特定の問い合わせを実行し、各メンバーを参照してから、次のグループに移動し、別の問い合わせを実行します。このプロセスを繰り返します。この場合、グループを構成する項目属性を Order 節で指定する必要があります。

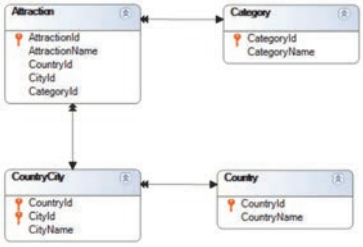
これは、外側の For Each コマンドとネストされた For Each コマンドに同じベーストランザクションが指定されると発生します。
For Each コマンドのベーストランザクションが指定されていない場合、GeneXus は、それらの For Each コマンド内で検出された項目属性からベーステーブルを判断します。ただし、検出されたベーステーブルが同じ場合は、コントロールブレークを実装する必要があると推論されます。これについては、このコースでは取り上げません。

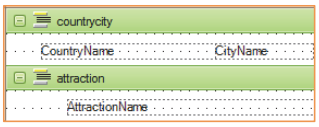
別のケースを見てみましょう。

異なるベーステーブル

外側の For Each ⟷ ネストされた For Each

コントロールブレーク

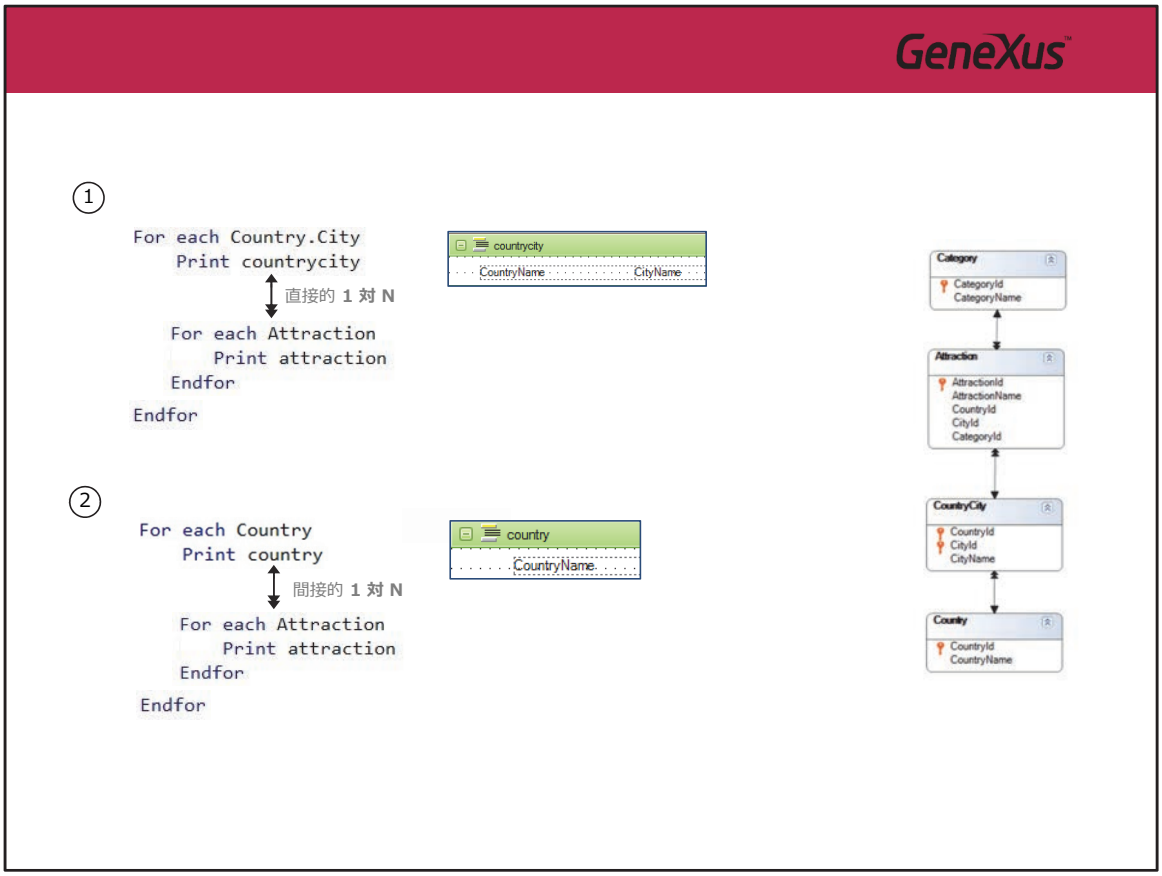





```
For each Attraction
    print attraction
    for each Country City
        print countrycity
    endfor
endfor
```

```
For each Attraction
    print attraction
    print countrycity
endfor
```

この場合は 2 番目の For Each コマンドは不要です。これは、外側の For Each コマンドで特定のタイミングに選択されたすべての観光名所に対して、関連する CountryCity レコードが 1 つしかないためです。そのため、2 番目の For Each コマンドを記述せず、どちらも Attraction 拡張テーブルに属す CountryName と CityName を含む CountryCity の printblock を出力するように指示することと同じになります。

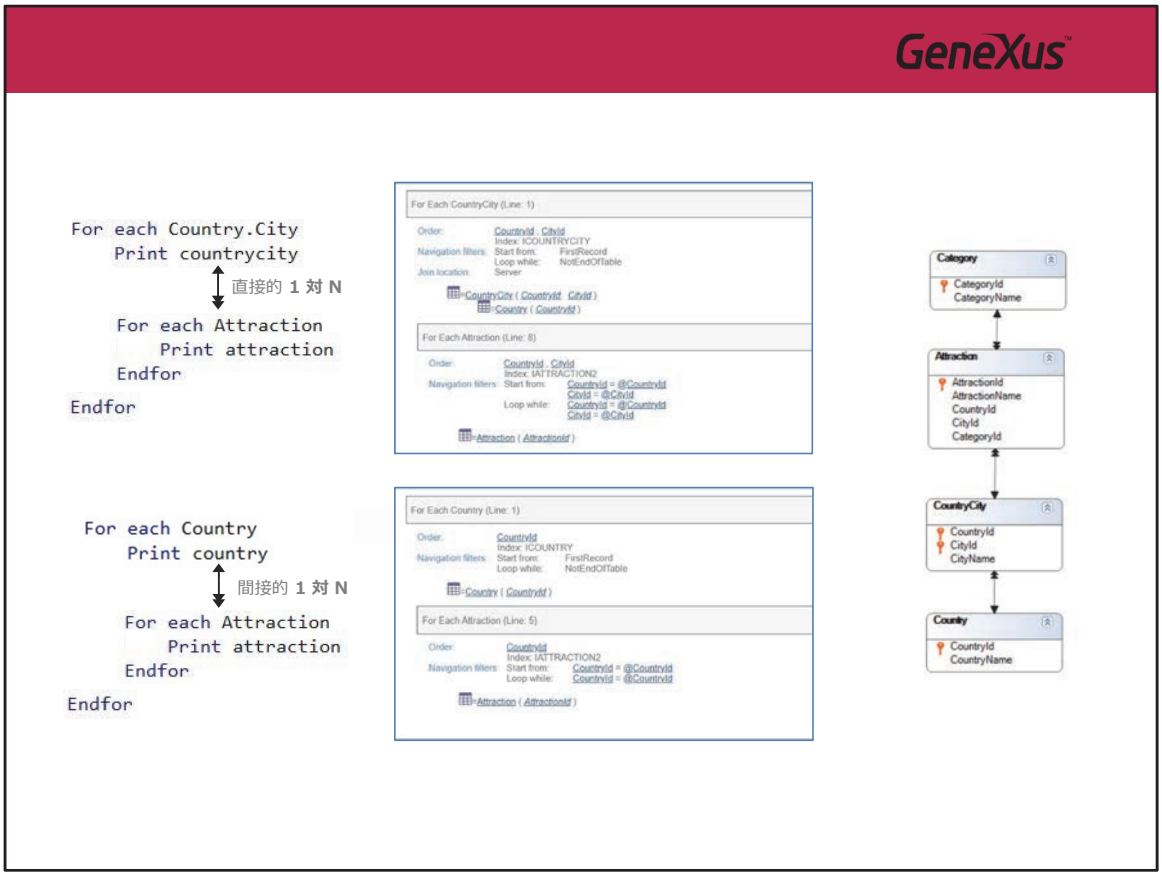


For Each コマンド間の 1 対 N の関係を示す 2 つのケースについて考えてみましょう (結合)。

1 つ目は直接的な関係です。外側の For Each とネストされた For Each のベーステーブルはそれぞれ CountryCity と Attraction であり、1 対 N の関係で関連付けられています。

国と都市の名前が出力され、それぞれのペアについて観光名所の名前も出力されます。

2 つ目は間接的な関係です。外側の For Each のベーステーブルは Country であり、ネストされた For Each のベーステーブルは Attraction です。詳しく見てみると、それぞれの国に N 個の都市があり、その各都市に N 個の観光名所があります。For Each コマンドのベーステーブルには直接的な 1 対 N の関係はありませんが、CountryCity テーブルを通じた間接的な関係があります。つまり、最初の For Each のベーステーブル Country が、ネストされた For Each のベーステーブルの拡張テーブル Attraction に含まれています。これにより結合が行われます。



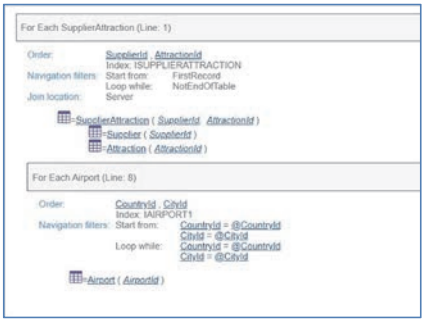
次に、ナビゲーション表示を見てみましょう。ネストされた For Each では、どちらの場合も Attraction をナビゲートし、テーブル全体は参照しません。関係が直接的な 1 つ目のケースでは、関係を確立する複合外部キー (CountryId と CityId) によってフィルタされることがアットマーク (@) で示されています。2 つ目のケースでは、複合外部キーの一部である CountryId によってのみフィルタされます。

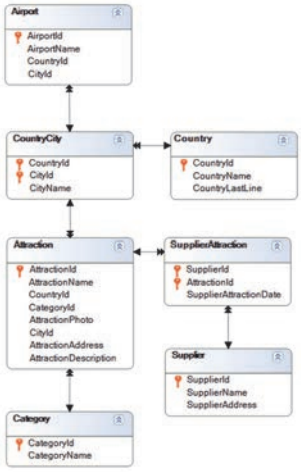
どちらのケースでも、Attraction の主キー (AttractionId) によってではなく、関連項目属性または項目属性のセットによってナビゲーションの順序が決まります。このような実装のため、GeneXus は外部キーによってインデックスを自動的に作成しています。このようにして、データベースへのアクセスが最適化されます。

そのため、GeneXus では、結合を行うと判断した後にナビゲーションの最適化が試行されます。

```
For each Supplier.Attraction
    print info
            ↕ 間接的 1 対 N
    For each Airport
        print airports
    endfor
endfor
```

結合



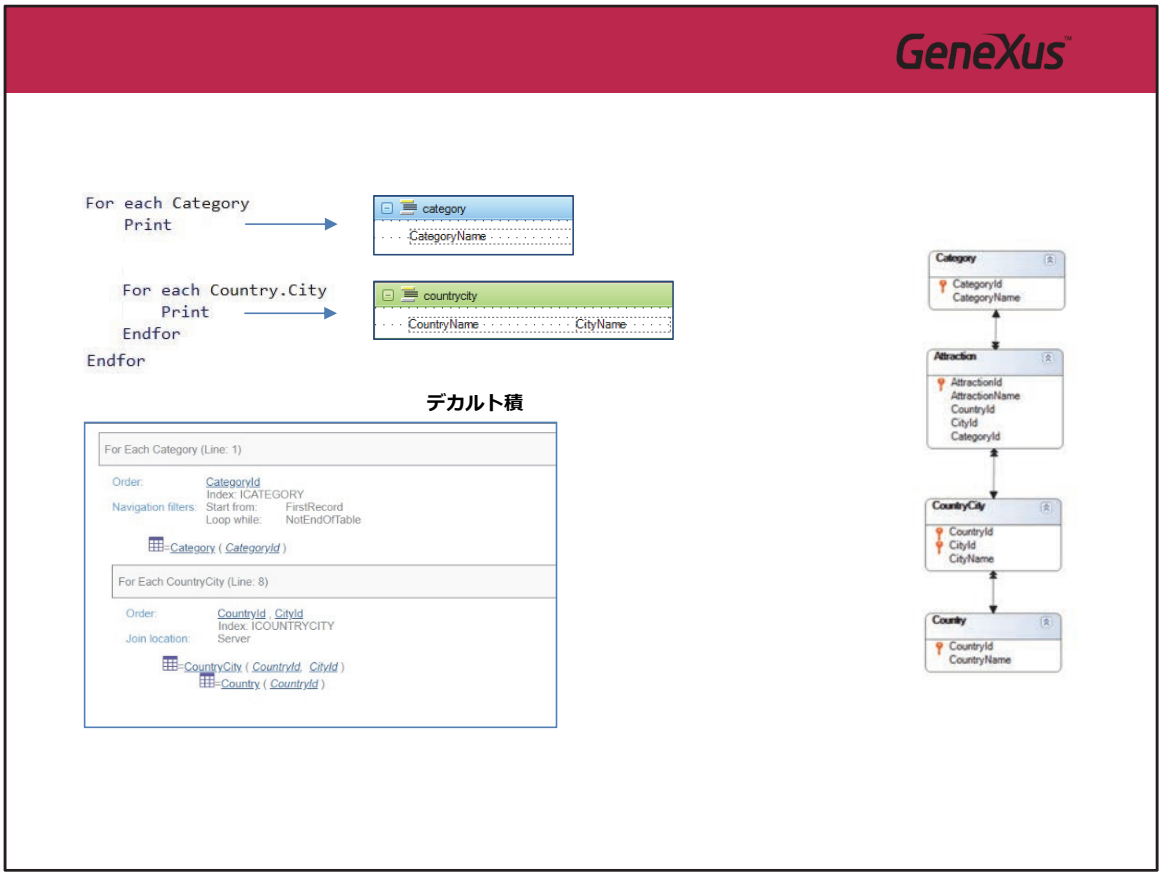


3 つ目の例を見てみましょう。このケースでは、観光名所を提供するサプライヤーが追加されているため、さらに 2 つのテーブルが表示されています。Supplier と、その下位テーブルである SupplierAttraction です。
また、国と都市に属する Airport もあります。
For Each を見ると、外側の For Each のベーステーブルが SupplierAttraction で、ネストされた For Each のベーステーブルが Airport であることが分かります。
この例では、printblock の info でサプライヤーの名前、観光名所の名前、その観光名所が示される日付が指定され、2 つ目の printblock で空港の名前が指定されるとします。

間接的な 1 対 N の関係があることに注目してください。つまり、すべての SupplierAttraction レコードに対して、Attraction レコードは 1 つだけです。CountryCity から 1 つだけ取得する場合、それが Airport の N 個のレコードに関連付けられます (その国/都市で N 個の空港が見つかったとしても、通常、実際には 1 つしかありません。このモデルでは複数存在する可能性があります)。

ここで GeneXus は、メインの For Each コマンドの拡張テーブルと、ネストされた For Each コマンドのベーステーブルとの間で共通する項目属性を検出します。どれでしょうか。{CountryId, CityId} のペアです。これにより、結合が行われます。

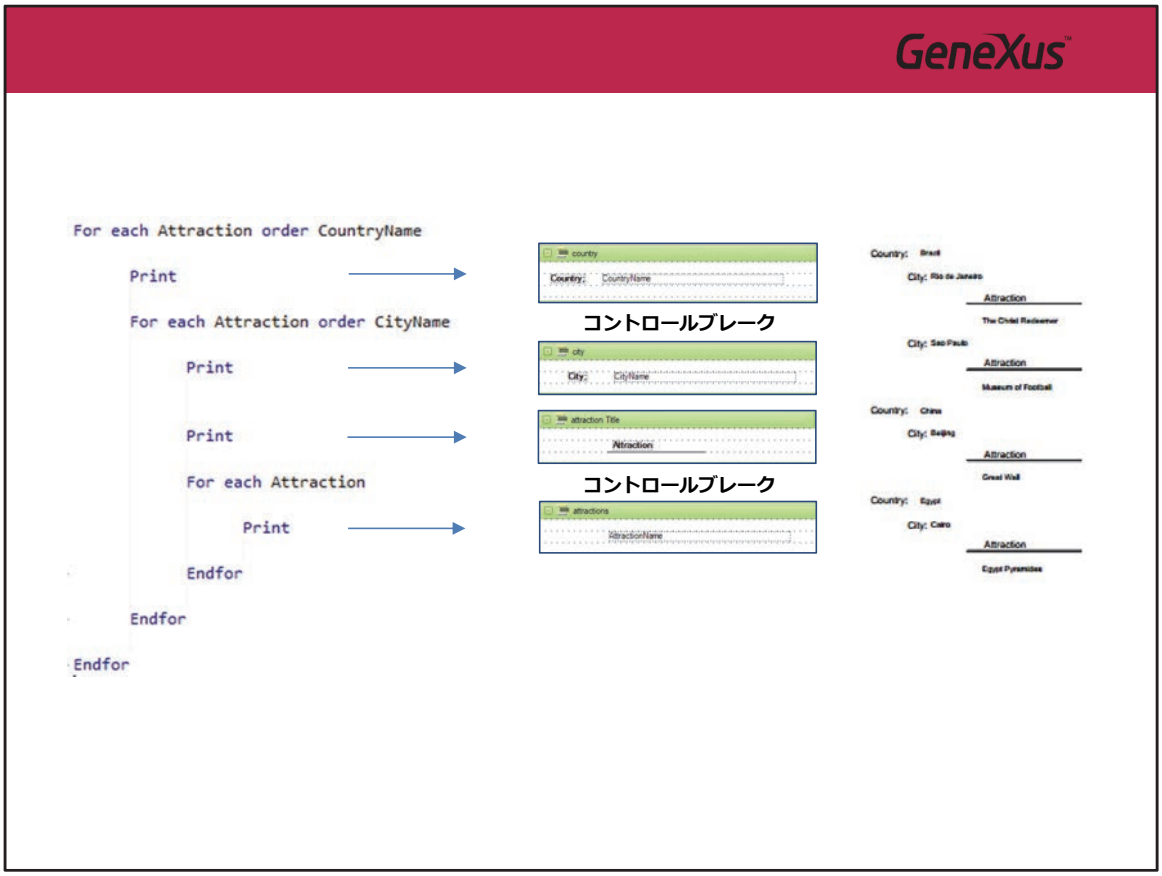
最適化のために、Airport テーブルの外部キーによってインデックスが選択されます。つまり、これはベーステーブル間の関係が直接的な 1 対 N ではなく、中間テーブルを介した間接的な関係であるケースです。



次に、関係のない複数の異なるベーステーブルを使用する、ネストされた For Each コマンドのケースを見てみましょう。

最初の For Each は Category をナビゲートしますが、ネストされた For Each は CountryCity をナビゲートします。このケースでは、見かけに反して 1 対 N の関係はありません。テーブルダイアグラムを見ると、1 つの Category に対して N 個の観光名所があり、それぞれの観光名所に対して CountryCity が 1 つだけあります。それぞれのカテゴリごとに、各観光名所の CountryCity がリストされると考えるかもしれません。ネストされた For Each のベーステーブルが Attraction であれば、そうなります。しかし、実際は CountryCity です。そのため、GeneXus は、観光名所によって関連付けられたレコードを探しているのではないことを理解しています。そのようなレコードを探す場合は、2 番目の For Each のベーストランザクションに Attraction を指定するだけで済みます。このケースは誤解を招く可能性があるため、ナビゲーション表示を注意深く確認して、混乱しないようにすることが必要です。表示を見ると、結合が行われていないことが明確に分かります。

このケースでは、GeneXus はテーブル間に直接的または間接的な 1 対 N の関係を見つけることができません。そのため、ネストされた For Each レコードには暗黙的なフィルタが適用されません。つまり、テーブル間でのデカルト積が行われます。外側の For Each の各ベーステーブルレコード (Category) に対して、ネストされた For Each のベーステーブル (CountryCity) のすべてのレコードが考慮されます。

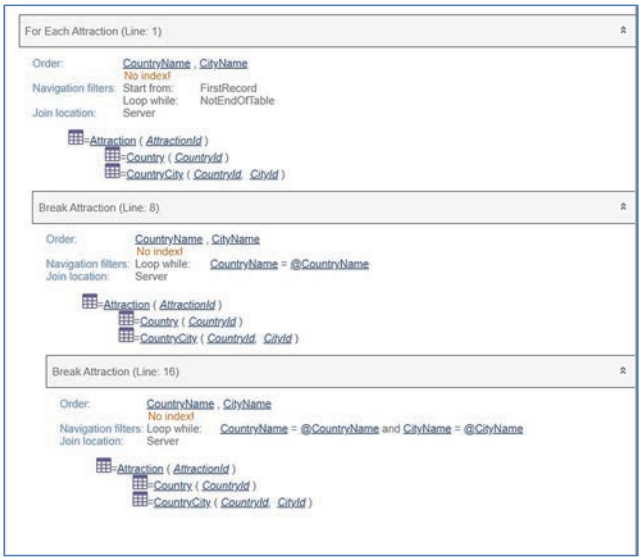


このケースでは、すべての国をリストし、それぞれの国について都市をリストし、それぞれの都市について観光名所をリストします。唯一の制限事項は、観光名所のレコードがある国と都市についてのみ、この処理を行うことです。

つまり、コントロールブレークを 2 つ実装して、最初に国でグループ化し、その中で都市でグループ化します。後者のグループ内では、すべての観光名所の名前を表示します。これを行うには、次の処理を行います。

Order 節を使用してグループ化の基準を定義します。コントロールブレークでは、Order 節が非常に重要であることを忘れないでください。Order 節は、データのリストに使用する項目属性 (単数または複数) を指定するだけでなく、グループ化の方法も設定します。

最も内側の For Each に順序を示すこともできますが、その順序は従来の方法でしか使用できません。つまり、順序付けのためにのみ使用されます。これについてはすぐにお見せします。



コントロールブレーク

コントロールブレーク

コントロールブレークが 2 つあるので、For Each コマンドが 3 つあることが分かります。
ナビゲーション表示を見ると、内側の各 For Each には Break が記述されており、同じベーステーブル Attraction が指定されています。これがコントロールブレークです。

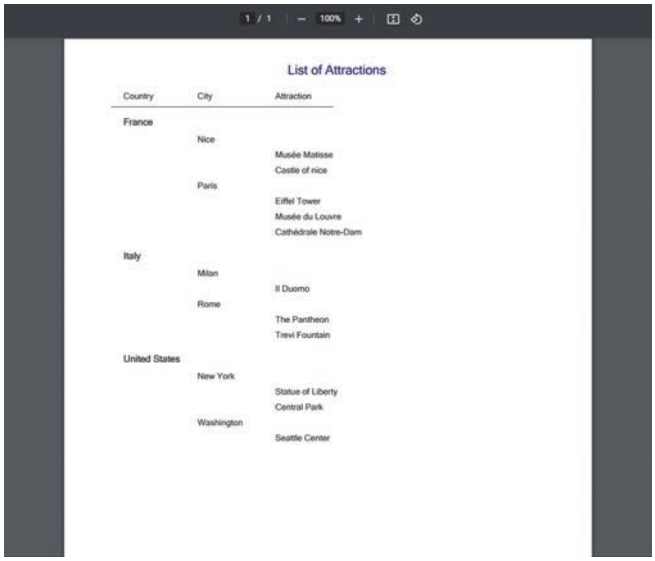
また、このベーステーブルは 1 回だけ参照されます。そのためには、For Each コマンドの Order に含まれる項目属性を連結して順序付けする必要があります。それが、CountryName と CityName が選ばれている理由です。

2 番目の For Each では、国によってブレーク処理が実行され、最初の For Each に配置されている国に対して処理が反復されます。3 番目の For Each では、国と都市によってブレーク処理が実行され、2 番目の For Each に配置されている都市に対して処理が反復されます。

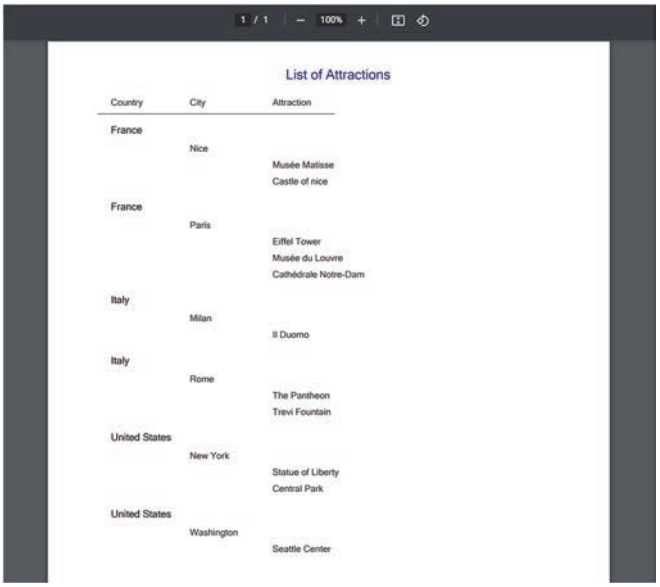
ここでは、これがデータであった場合にどのように表示されるかを見てみましょう。まず国が表示され、その国の名前に対応する 1 つ目の都市が表示され、さらにその国と都市の観光名所が表示されます。次に、同じ国の 2 つ目の都市と、その観光名所が表示されます。その国に該当する都市がそれ以上ない場合は、次の国や都市がアルファベット順に表示されます。この処理が繰り返し行われ、国、都市、観光名所が表示されます。

最初の For Each を CountryName で、また、2 番目の For Each を CityName で順序付けする代わりに、CountryName と CityName のペアで両方の For Each または最初の For Each のみを並べ替えたとしたら、前のリストがどのように実行されるか考えてみましょう。

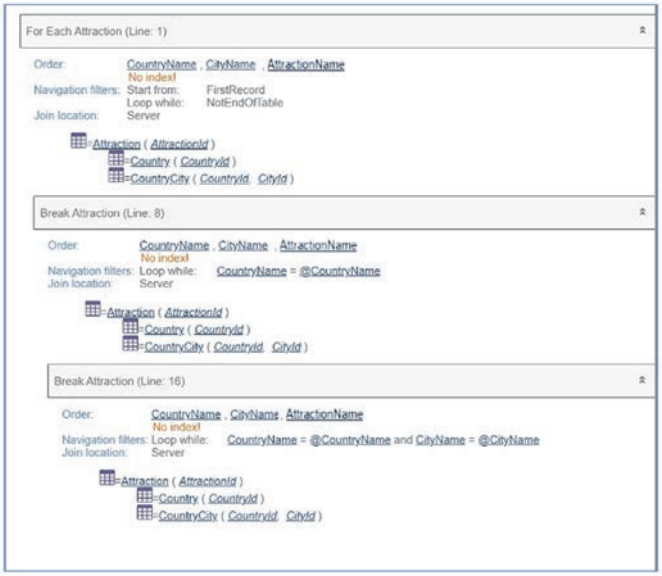
この場合、2 番目の For Each では、ナビゲーション表示が先ほどのものとは異なることに注意してください。Loop while は、"CountryName = @CountryName and CityName = @CityName" を読み取ります。



これは、実行時に、同じ情報が別の方法で表示されることを意味します。CityName を 2 番目の For Each に配置した場合、国名、その国の 1 つ目の都市、その都市の観光名所の順に表示されます。次に、同じ国の 2 つ目の都市、その都市の観光名所が表示されます。国が変わるまでこの形で表示されます。次の国も、同じ形で表示されます。
このように、CityName を最初の For Each に配置するか、2 番目の For Each に配置するかだけで、情報が明らかに異なる方法で表示されます。



CityName を最初の For Each に配置した場合、このように表示されます。



ここで、最初の例 (まず国で、次に都市でブレーク処理が行われ、観光名所がリストされる) で、最後の、最も内側の For Each に order by AttractionName を追加するとします。つまり、国と都市に該当する観光名所を、観光名所名で順序付けます。

ナビゲーション表示が変わり、AttractionName が 3 つの For Each コマンドの順序に追加され、CountryName、CityName、および AttractionName が残されます。

この変更を適用して実行すると、説明したように、国名でブレーク処理が行われます。France が最初に表示され、Italy が後になるのはそのためです。France 内では CityName でブレーク処理が行われるため、アルファベット順では Paris の前になる Nice が最初に表示されます。これらの各サブグループ内で、観光名所がアルファベット順に表示されます。

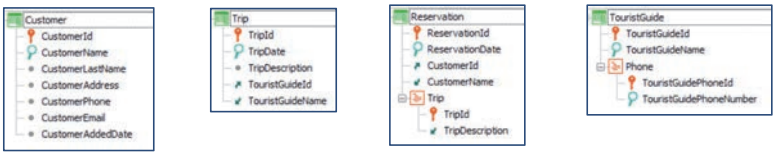
これまで見たように、ブレーク処理の基準はナビゲーションフィルタにあります。次のように指示していることがわかります。CountryName、CityName、AttractionName の順に実行し、最初の For Each の本文を実行して、CountryName のみが出力されるようにします。次に、2 番目の For Each を実行します。ここでは、CountryName が変わらない限り、CityName のみを含む printblock を出力しますが、その後ですぐに内側の For Each を実行します。ここでは、CountryName と CityName のペアが変わらない限り、AttractionName を出力します。

つまり、フィルタは、最も内側の For Each を AttractionName で順序付けしなかったときとまったく同じです。そのため、入力した順番は並べ替えのみに使用され、ブレーク処理を行うことはできません。

GeneXus

ケーススタディ

トランザクション設計



ソース

```
Parm(in:&ReservationDate, in: CustomerId);

For each Reservation.Trip
     Where ReservationDate >= &ReservationDate
     Print Trips
     For each TouristGuide.Phone
          Print TouristGuidesPhones
     Endfor
Endfor
```

前の章で、ケーススタディを分析し、Parm ルールでフィルタを項目属性として受け取ったときに GeneXus がそれをどのように適用するかを確認しました。

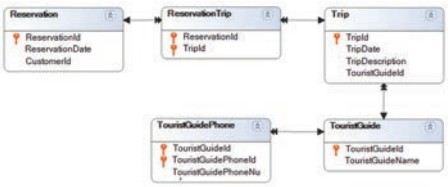
同じケースに戻って、フィルタが適用されないようにする方法を確認します。

まず、表示されているトランザクション設計から始めます。特定の顧客について、指定した日付以降の、その顧客が予約したすべてのツアーと、各ツアーを担当するツアーガイドの連絡先電話番号が表示されたリストを入手する必要があります。

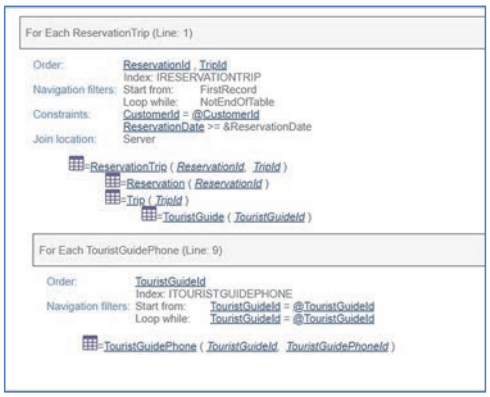
これを実現するために、ここに示すソースを提案します。

GeneXus

ケーススタディ




```
Parm(in:&ReservationDate, in: CustomerId);

For each Reservation.Trip
      Where ReservationDate >= &ReservationDate
      Print Trips
      For each TouristGuide.Phone
            Print TouristGuidesPhones
      Endfor
Endfor
```



外側の For Each コマンドのベーステーブルは ReservationTrip、内側の For Each コマンドのベーステーブルは TouristGuidePhone とした場合、使用される情報に対する暗黙的なフィルタは確立されるでしょうか。答えは「はい」です。各ツアーガイドの電話番号が表示されます。

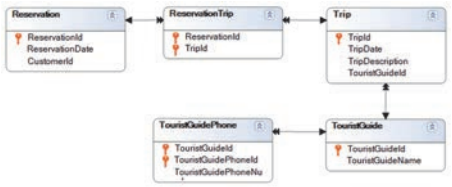
なぜでしょうか。GeneXus では、外側の For Each コマンドの拡張テーブルと、ネストされた For Each コマンドのベーステーブルとの関係が確認されます。

この方法で 1 対 N の関係を探すこともできます。ただし、このケースは間接的な関係です。各 ReservationTrip に TouristGuideId があり、ナビゲートするテーブルにも TouristGuideId がある場合、GeneXus は情報間の関係から、それらが同じものであることを理解します。
結合が行われるのはそのためです。

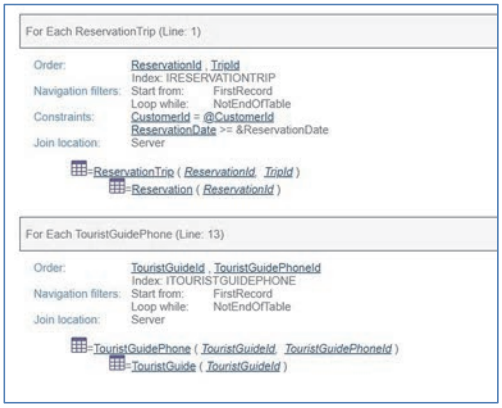
ナビゲーション表示を見ると明らかで、@ マークは常にコンテキストデータを示しています (@CustomerId は、Parm ルールの CustomerId 項目属性で受け取る値を参照します)。この @TouristGuideId は、最初の For Each コマンドがアクセスする Trip テーブル内で、該当する名前の項目属性の値を参照します。

GeneXus

ケーススタディ




```
For each Reservation.Trip
     Where ReservationDate >= &ReservationDate
     Print Trips
     Do "ListTouristGuidePhones"
Endfor

Sub "ListTouristGuidePhones"
     For each TouristGuide.Phone
          Print TouristGuidesPhones
     Endfor
EndSub
```



プログラマーが希望していない場合、このような自動フィルタを回避する方法はあるでしょうか。

はい、あります。サブルーチンを使用して、この 2 番目の For Each コマンドのコードをカプセル化することで回避できます。
これを行うと、ナビゲーション表示にそれが表示されます。

サブルーチンはコードブロックであり、Sub コマンドを使用してオブジェクト内で定義されます。同じオブジェクト内であれば、Do コマンドを使用して何度でも呼び出すことができます。

次の資料では、これらを実装する方法について詳しく説明します。