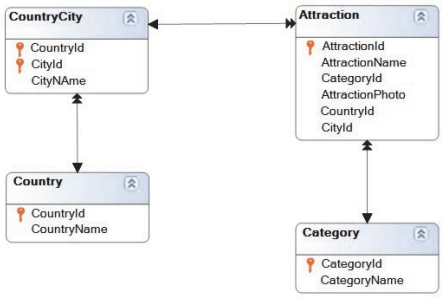
For Each コマンドの詳細

GeneXus™

復習: ベーストランザクション



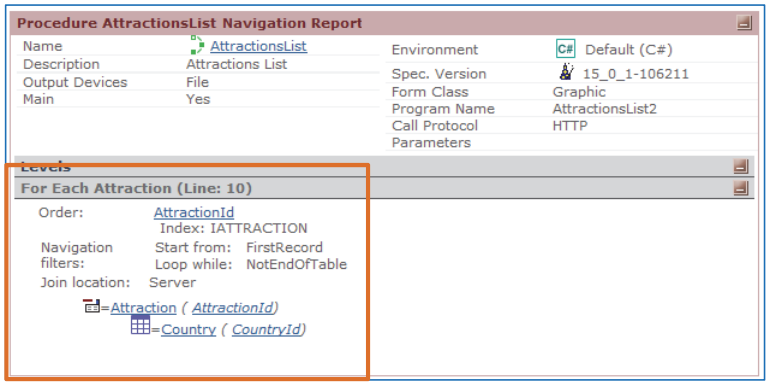

```
print Title

for each Attraction
     print Attractions
endfor
```

GeneXus は、For Each コマンドのベーステーブルを判断する際に、For Each コマンドの横で宣言されているトランザクションの名前を考慮します。このトランザクションの名前はベーストランザクションの名前に対応しています。つまり、このトランザクションに関連付けられている物理テーブルを参照します。

さらに、For Each コマンド内で宣言されている項目属性は、printblock、Where 節、Order 節など、どこにあるかにかかわらず、For Each コマンドのベーステーブルの拡張テーブルに属している必要があります。

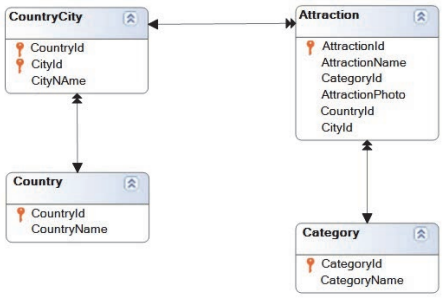
この例では、For Each コマンドのベーステーブルは Attraction になっています。つまり、必要なデータを取得するために Attraction テーブルを参照し、その拡張テーブルにアクセスします。

復習: ベーストランザクション



ナビゲーション表示を見ると、ベーステーブルが Attraction であることがはっきりと分かります。このテーブルが主キー AttractionId で並べ替えられ、テーブル全体が参照されます。また、Country テーブルにもアクセスし、CountryName の値を取得します。この値は観光名所の国に対応しています。

復習: ベーストランザクション



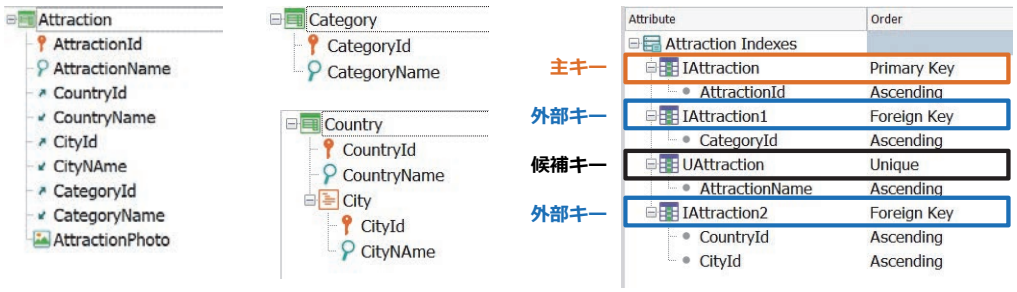
For Each コマンドに
ベーストランザクションを
指定するのは必須ではない

```
print Title

for each        ?
     print Attractions
endfor
```

For Each コマンドには、必ずベーストランザクションを指定しなければならないのでしょうか。

答えは「いいえ」です。GeneXus では、コマンドに含まれている項目属性から For Each コマンドのベーステーブルを計算することができます。ベーステーブルを見つける方法は、このコースでは取り上げません。

インデックスおよびインデックスとデータベースクエリの関係



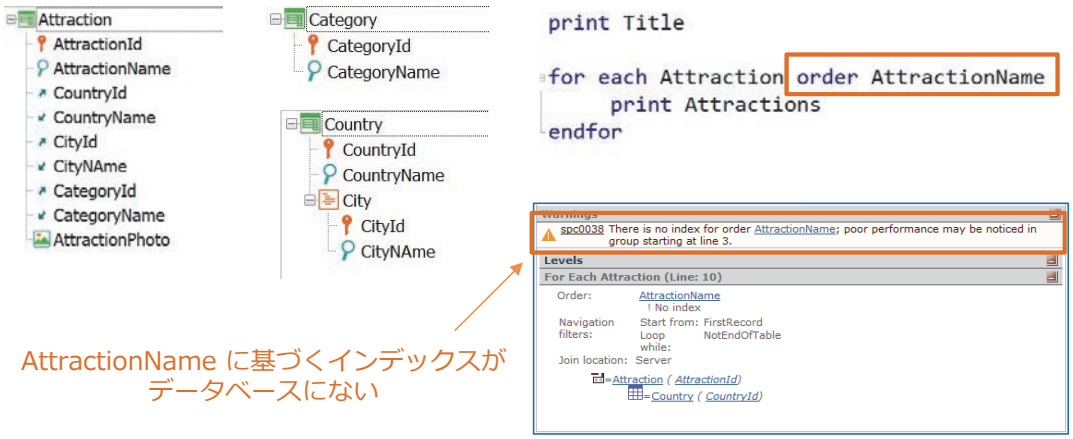
ここでは、インデックスおよびインデックスとデータベースクエリの関係について説明します。

既に学習したとおり、**インデックス**を使用すると、データに効率的にアクセスできます。

また、GeneXus は各テーブルで、主項目属性 (単体のキーまたは候補キー) および外部キーにインデックスを作成します。これは、テーブル間でのデータの整合性をより効率的に管理するために行われます。

インデックスを定義することもできます。その際、値の重複を許可するかどうかを指定できます。値の重複を許容しないインデックス、つまり一意のインデックスを定義する場合は、GeneXus に対して、値の一意性を自動で制御する必要があることと、インデックスを定義する項目属性または項目属性のセットが候補キーになることを指示します。

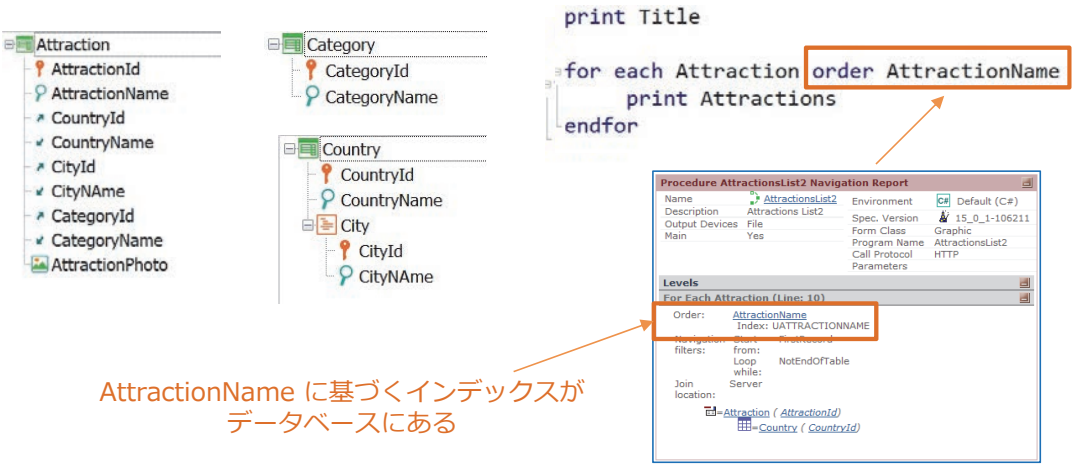
インデックスおよびインデックスとデータベースクエリの関係



観光名所の名前で並べ替える場合など、Order 節を追加すると、ナビゲーション表示に「情報を並べ替える必要がある項目属性に基づくインデックスがデータベースにないため、このクエリはパフォーマンスが低くなる可能性がある」という内容の警告が表示されます。

GeneXus にデータを並べ替える項目属性を指定すると、GeneXus は効率的に並べ替えようとして、その項目属性のインデックスを探します。それが見つからない場合は、そのことを示す通知が表示されます。

インデックスおよびインデックスとデータベースクエリの関係



Attraction テーブルのレコードを AttractionName 項目属性で並べ替えて取得する必要がある場合、これらのレコードは既定で主キーの項目属性の値で並べ替えられているため、もう一度並べ替えを行う必要があります。

クエリを定義する際に、テーブル内で並べ替える項目属性に対して物理的なインデックスが作成されている場合、GeneXus はそれを使用します。ここでは、クエリは従属項目属性 AttractionName に基づいて並べ替える必要があります。既に見たとおり、GeneXus は、インデックスが作成されていないことをナビゲーション表示で警告します。

インデックスが存在している場合はクエリが最適化されます。ただし、インデックスを作成すると、メンテナンスしなければならないというデメリットがあります。つまり、ユーザーが Attraction テーブルに対して項目属性を追加、変更、削除を行うと、インデックスの再考が必要になります。
これらの処理を行ったときは、F5 キーを押すことでデータベースが再編成され、新しいインデックスが作成されます。その後、ナビゲーション表示に、作成されたインデックスを GeneXus が使用する旨のメッセージが表示されます。

インデックスは、作成と同じようにいつでも削除でき、F5 キーを押して再編成することで、以前の状態に戻ります。

インデックスおよびインデックスとデータベースクエリの関係: 例


```
Parm (in:&NameFrom, in:&NameTo);

print Title

for each Attraction order AttractionName
    Where AttractionName >= &NameFrom
    Where AttractionName <= &NameTo

    print Attractions
endfor
```

Where AttractionName >= &NameFrom and AttractionName <= &NameTo

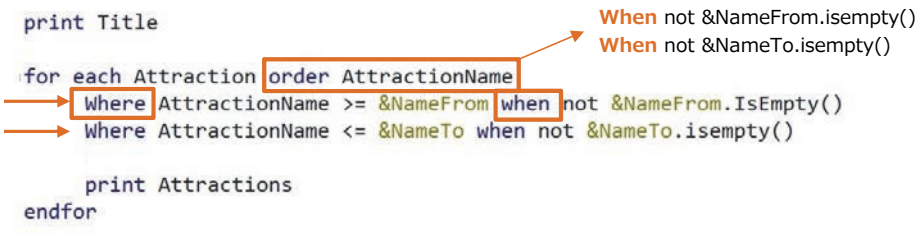
例を見てみましょう。

パラメーターで渡された 2 つの値の間に名前がある観光名所を、アルファベット順のリストで取得するとします。たとえば「B」と「N」の間とします。
この例で Where 節を指定しているのはそのためです。

Where 節を複数使用すると、条件を「and」論理演算子で組み合わせた単一の Where 節と同じ意味になります。つまり、すべての条件を同時に満たすレコードのみが取得対象になります。
AttractionName でフィルタリングする際に、その項目属性に基づくインデックスがある場合は、常に **AttractionName で並べ替えて**クエリを最適化する必要があります。

Order 節を指定しないと、GeneXus は主キーで並べ替えます。その場合、観光名所が指定の範囲内に含まれているかどうかを判断するために、テーブル全体を参照する必要があります。

When 節



&NameFrom 変数および &NameTo 変数が空の場合、この For Each コマンドでは、どのような結果が得られるでしょうか。

名前が空の観光名所が 1 つあった場合、それが条件を満たす唯一のデータとなるため、そのデータのみが返されます。名前が空の観光名所がなかった場合、観光名所は表示されません。

では、特定の状況でのみ並べ替えとフィルタが適用されるようにすることは可能でしょうか。たとえば、&NameFrom 変数が空でない**場合のみ**に最初の Where 節を適用することはできるでしょうか。また、&NameTo 変数が空でない**場合のみ**に 2 つ目の Where 節を適用することはできるでしょうか。
答えは「はい」です。Where 節および **when** で条件を指定することで実現できます。各 Where 節は、When の条件が満たされた場合にのみ適用されます。
こうすると、実行時に両方の変数が空のままであると、Where 節はどちらも適用されません。そのため、そのテーブルのすべての観光名所が一覧表示されます。
&NameFrom 変数が空で &NameTo 変数が空でない場合は、1 つ目の Where 節は適用されませんが、2 つ目の Where 節は適用されます。そのため、名前が &NameTo と同じかそれより順序が前のすべての観光名所が表示されます。

同様に、並べ替えを行うかどうかの条件を設定できます。実際、条件付きの並べ替えを複数指定して、最初に条件が満たされたものを選択することができます。

When none 節

```
print Title

for each Attraction order AttractionName
     Where AttractionName >= &NameFrom when not &NameFrom.IsEmpty()
     Where AttractionName <= &NameTo when not &NameTo.isempty()

     print Attractions
When none
     Print NoAttractions
endfor
```

次に、When none 節について説明します。

ベーステーブルのどのレコードも指定された条件を満たさない場合はどうなるでしょうか。
ここでは、「関連するレコードがない」という警告メッセージが出力として表示されるようにします。

これを実現するために **When none** 節を使用します。

When none と endfor の間に記述されているすべてのコマンドが順番に実行されます。これは、**For Each コマンドのベーステーブル内に指定した条件を満たすレコードが見つからなかった場合にのみ実行されます。**

この例ではメッセージを出力するようにしていますが、別の For Each コマンドなど、一連のコマンドを入力することもできます。
When none 節の後に実行される処理は検索が成功しなかったことを示すため、ここに For Each コマンドを入力しても When none 節はネストされません。その場合、For Each は単体のコマンドのように機能します。

まとめ

For Each <ベーストランザクション>

order <項目属性>$_1$, <項目属性>$_2$, … , <項目属性>$_n$ [when <条件>]

order <項目属性>$_1$, <項目属性>$_2$, … , <項目属性>$_n$ [when <条件>]

where <条件> [when <条件>]

where <条件> [when <条件>]

<メインコード>

When none

.....

endfor

ここまでの内容をまとめます。

既に確認したように、For Each コマンドの**ベーステーブル**は、指定されたベーストランザクションに基づいて決定されます。指定されているその他の項目属性 (For Each コマンドの本文 (メインコード)、Order 節、Where 節) は、そのベーステーブルの拡張テーブルに属している必要があります。

When none ブロックで指定されている項目属性は考慮されません。

これまでに確認したものはすべてグレーで表示されています。ここでは、**When** 節および **When none** 節が追加されています。

この後、データベースにアクセスするために、この基本的なコマンドに追加の節を指定できることを確認します。

ケーススタディ

Customer
{
CustomerId*
CustomerName
}

Trip
{
TripId*
TripDescription
TouristGuideId
TouristGuideName
}

TouristGuide
{
TouristGuideId*
TouristGuideName
Phone
{
TouristGuidePhoneId*
TouristGuidePhoneNumber
}
}

Reservation
{
ReservationId*
ReservationDate
CustomerId
CustomerName
Trip
{
TripId*
TripDescription
}
}

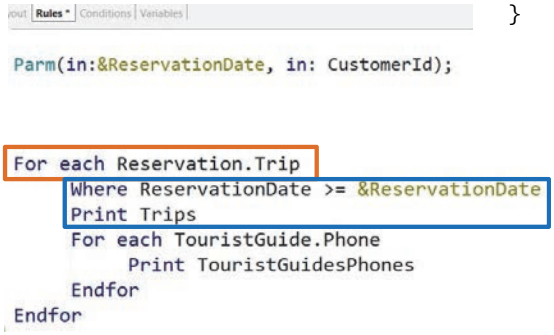
最後に、ケーススタディを確認しましょう。

次の要素を含むトランザクション設計を考えます。

Customer トランザクション、Trip トランザクション (担当のツアーガイドがいる旅行に対応)、Tourist Guide トランザクション (電話番号が設定されている)、Reservation トランザクション (顧客ごとに、予約した旅行のセットを記録する)。

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旅行と、担当する各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。

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
```

Trip
{
TripId*
TripDescription
TouristGuideId

TouristGuideName
}

TouristGuide
{
TouristGuideId*
TouristGuideName
Phone
{
TouristGuidePhoneId*
TouristGuidePhoneNumber
}
}

Reservation
{
ReservationId*
ReservationDate
CustomerId
CustomerName
Trip
{
TripId*
TripDescription
}
}

こちらのソースを提案します。

ここに示している情報が、求めているものかどうかを確認します。ネストされた 2 つの For Each コマンドがあります。1 つ目の For Each コマンドでは、ベーステーブルが Reservation トランザクションの Trip レベルに対応していることが明示的に指定されています。つまり、ReservationTrip です。

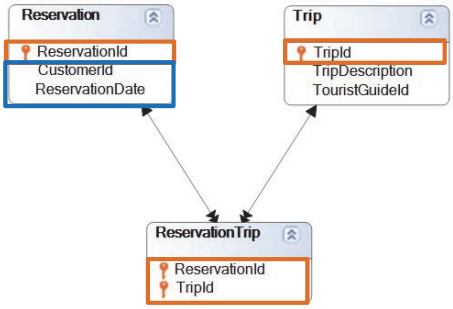
外側にあるその For Each コマンドでは、ReservationTrip の拡張テーブルに属さない項目属性が使用されていないことを確認します。使用されている場合は、ナビゲーション表示に「項目属性にアクセスできない」という警告が表示されます。

確認が必要な項目属性は、Where 節および Trips という printblock 内にある項目属性です。この場合、Trip に含まれている TripDescription と、Reservation に含まれている ReservationDate です。

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
```



テーブルダイアグラムを確認します。

ReservationTrip から、Trip テーブルの単一のレコードと、Reservation テーブルの単一のレコードにアクセスすることがはっきりと分かります。そのため、GeneXus は For Each コマンドを繰り返すたびに、それら 2 つのテーブルにアクセスする必要があります。

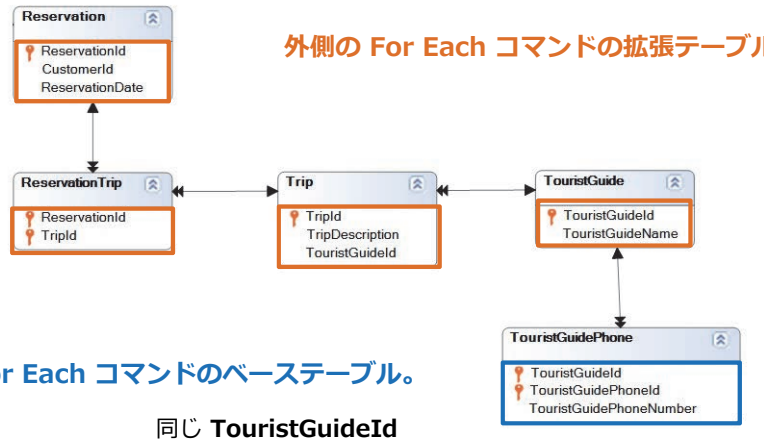
では、For Each コマンドではベーステーブルのどのレコードを参照するのでしょうか。それは、Reservation テーブルにアクセスして ReservationDate の値を評価する際、パラメーターで受け取った &ReservationDate 変数の値と等しい、またはそれより大きいデータです。
また、CustomerId の値もパラメーターで直接受け取った項目属性の値と一致する必要があります。「項目属性で受け取る」というのは、この項目属性がインスタンス化されるということです。つまり、その項目属性がどこかで使用されると、オブジェクトが呼び出された際に受け取った値と一緒に使用されます。

確認する For Each コマンドは、その項目属性を含む Reservation テーブルに既にアクセスしています。そのため、その CustomerId の値に自動フィルタが適用されます。

次の点を明確にしておくことが重要です。Parm ルールで受け取った項目属性が For Each コマンドの拡張テーブルに属していても、自動的にフィルタとして適用されるわけでは**ありません**。これが行われるためには、For Each コマンドで、その特定の拡張テーブルにアクセスして処理を行う必要があります。

ケーススタディ

間接的な 1 対 N の関係



ネストされた For Each コマンドではどうなるでしょうか。ベーステーブルは、TouristGuide トランザクションの Phone レベルに明確に関連付けられているものです。ここで気になるのは、使用する情報の暗示的なフィルタは確立されるかどうかということです。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 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の関係を探します。

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 がある場合は、GeneXus はその関係から、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。

引き続き For Each コマンドについて詳しく学習します。