サブルーチン

GeneXus™

GeneXus

サブルーチン

Code ブロック

次の場所で使用可能:

- Web パネル
- プロシージャー
- パネル
- トランザクション

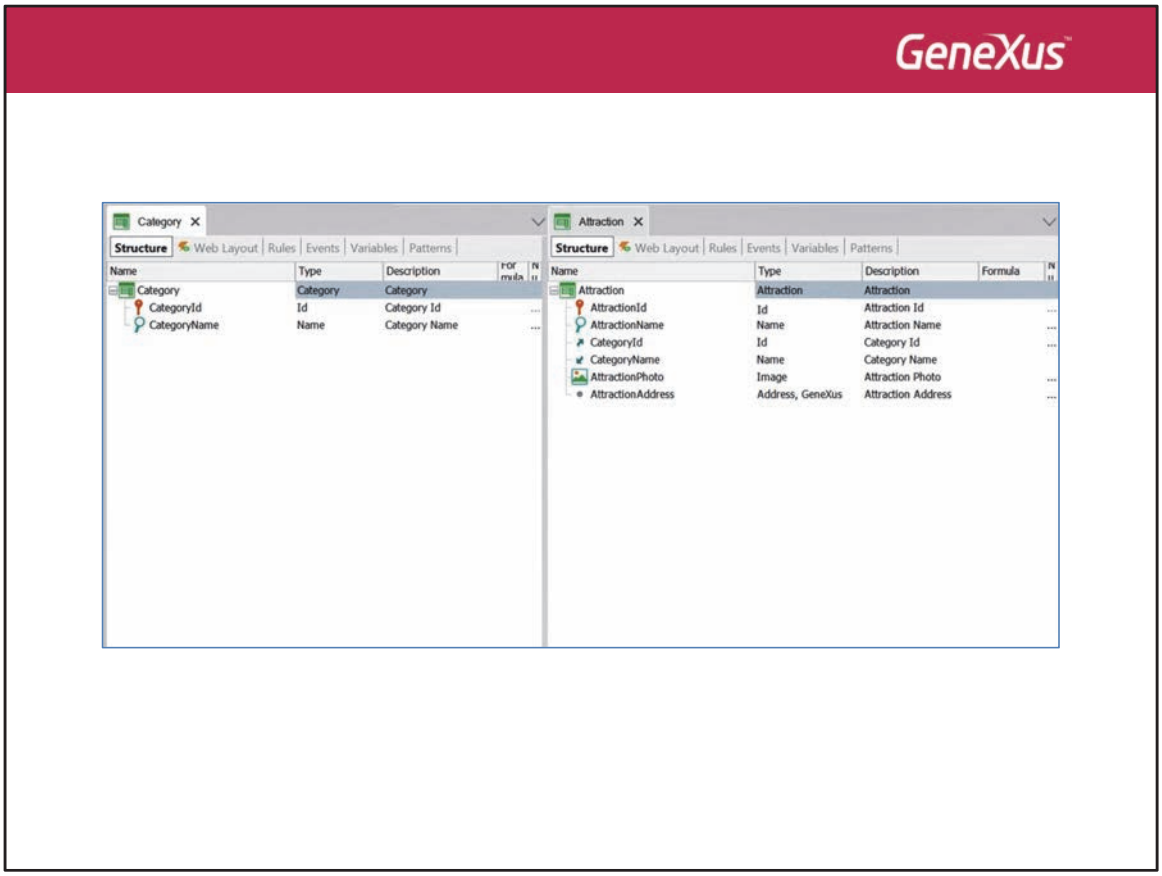
```
&var = value
att1 = &var + 2
For each trn
    where att2 = value
    att2 = att1
Endfor
```

この章では、GeneXus におけるサブルーチンについて説明します。仕組みと実装方法を説明し、どのような用途があるかを確認します。

サブルーチンは基本的に Code ブロックであり、同じオブジェクト内にあるかぎり、何度でも呼び出すことができます。
そのため、オブジェクトのさまざまな場所から、または同じ場所から、同じブロックを複数回実行できます。サブルーチンでは、コードを 1 回だけ記述し、それに名前を割り当てることで、その名前で呼び出すことができます。

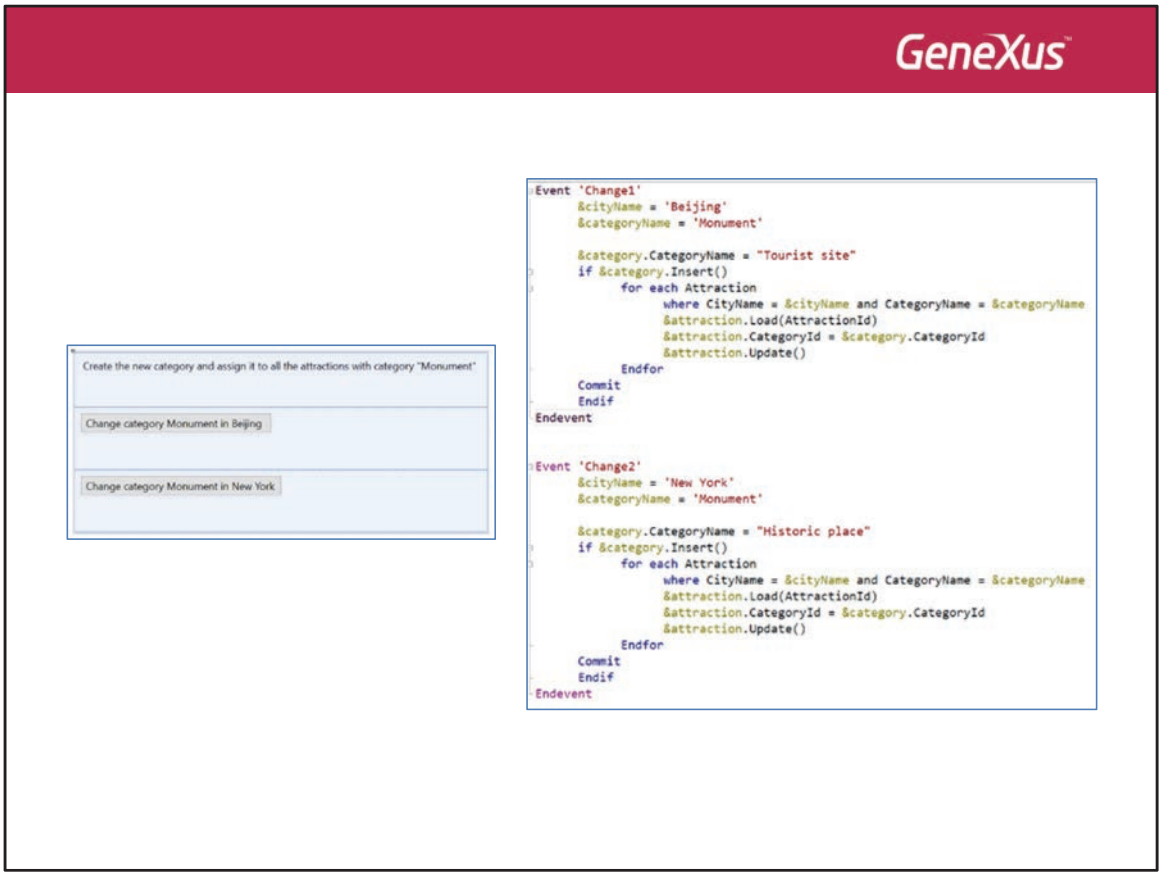
サブルーチンは、データプロバイダーを除き、プログラミングを受け入れるすべてのオブジェクトで使用できます。具体的には次のものがあります: Web パネル、プロシージャー、パネル、トランザクション。

簡単な例を使って、この仕組みを見てみましょう。



旅行代理店向けに開発しているアプリケーションを使います。トランザクションとして、さまざまなカテゴリを登録するための Category トランザクションと、Attraction トランザクションがあります。後者のトランザクションには、それ自体の項目属性に加えて、Category トランザクションの CategoryId および CategoryName 項目属性もあります。

このアプリケーションの Web パネルの 1 つで、その実装を確認しましょう。

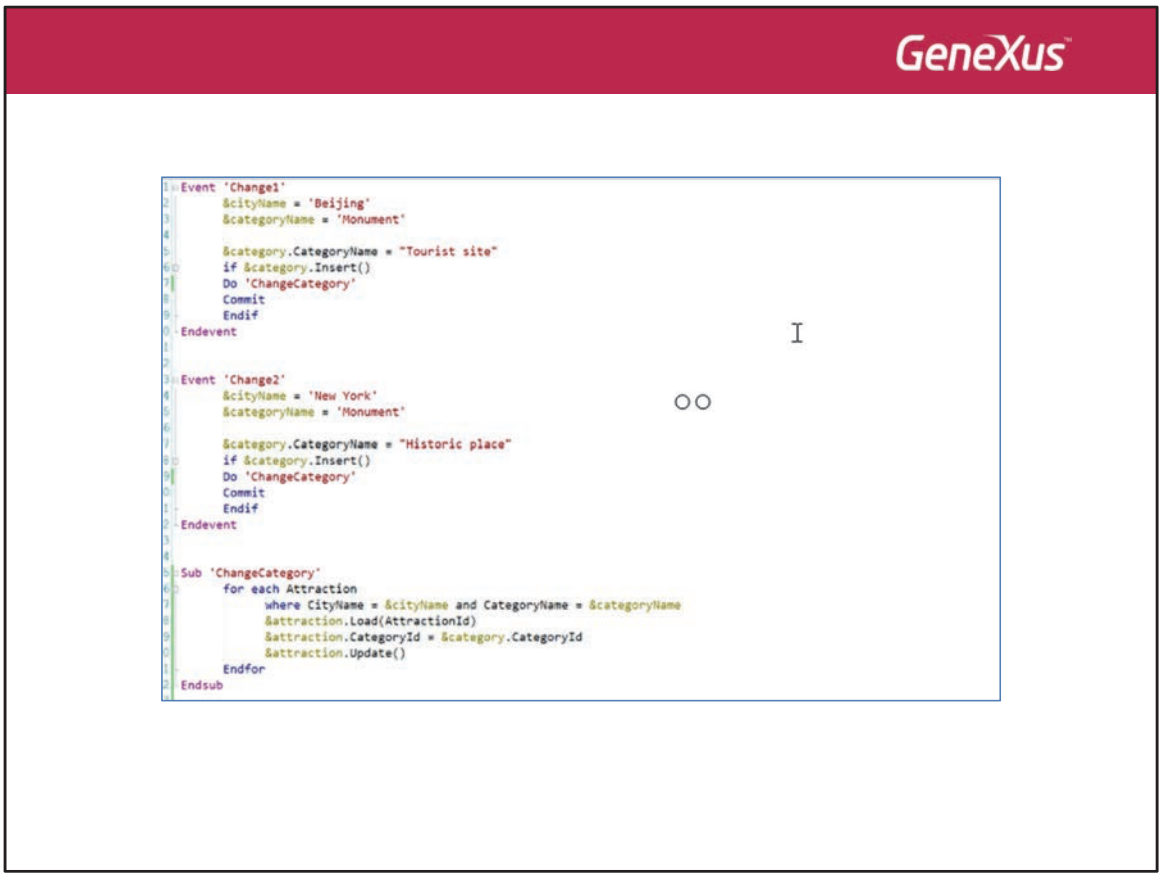


レイアウトには 2 つのボタンがあります。1 つは「Change category Monument in Beijing」で、もう 1 つは「Change category Monument in New York」です。
最初のボタンのイベント名は Change1 で、2 番目のボタンのイベント名は Change2 です。

Event セクションを見ると、これら 2 つのイベントにコードが割り当てられています。最初のものを見てみましょう。2 つの変数 cityname と categoryname が宣言され、それぞれに「Beijing」と「Monument」と言うテキストが割り当てられています。
さらに、ビジネス コンポーネント カテゴリタイプのカテゴリ変数があり、カテゴリの名前が「Tourist site」であると指定されています。その後で、Category テーブルへの新しいカテゴリの挿入が試行されます。
レコードが正しく挿入されると、観光名所のテーブルが参照され、CityName の値が CityName 変数と同じであり、CategoryName の値が CategoryName 変数と同じであるレコードのみがフィルタされます。
それらのレコードに対して観光名所のカテゴリが更新され、入力したカテゴリ (この場合は「Tourist Site」) に変更されます。

2 番目のボタンも最初のボタンと同じ処理を行いますが、異なるデータを使用します。挿入されるカテゴリ名は「Historic Place」で、カテゴリが「Monument」であるニューヨークの観光名所がフィルタされます。

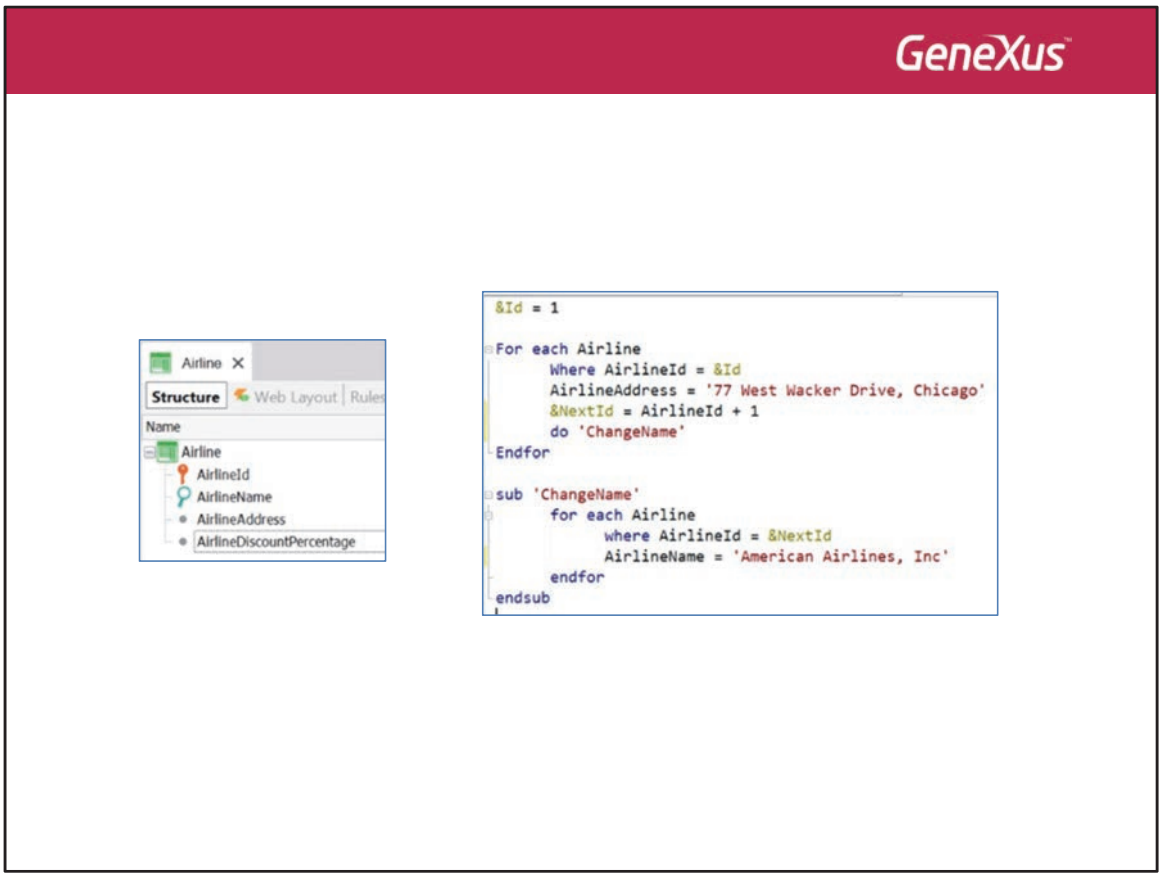
最初のイベントと 2 番目のイベントを見ると、まったく同じ Code ブロックであることが分かります。
この場合、コードを 1 か所で宣言し、イベントから必要に応じて呼び出すことができます。これをやってみましょう。



Sub コマンドではサブルーチンを定義できます。そのサブルーチンには名前を割り当てる必要があります。終了する場所は Endsub で指定します。
この中には、後で呼び出すコードを入力する必要があります。これをコピーして貼り付けます。
イベント内のブロックを削除し、代わりにサブルーチンを呼び出します。これを行うには、Do コマンドを使用して、その後に名前を指定します。
この実装は、サブルーチンを宣言する前とまったく同じ処理が行われます。

このように、コードをモジュール化すると、より明確で読みやすいコードにすることができます。もう 1 つのメリットは、このブロック内で何かを変更する必要が生じた場合、1 回変更するだけで、コードを呼び出すあらゆる場所に適用できることです。また、同じオブジェクト内 (このケースでは Web パネル InsertCategoriesAndAttractions) にあるかぎり、サブルーチンを使用してこのコードを何度でも再利用できます。

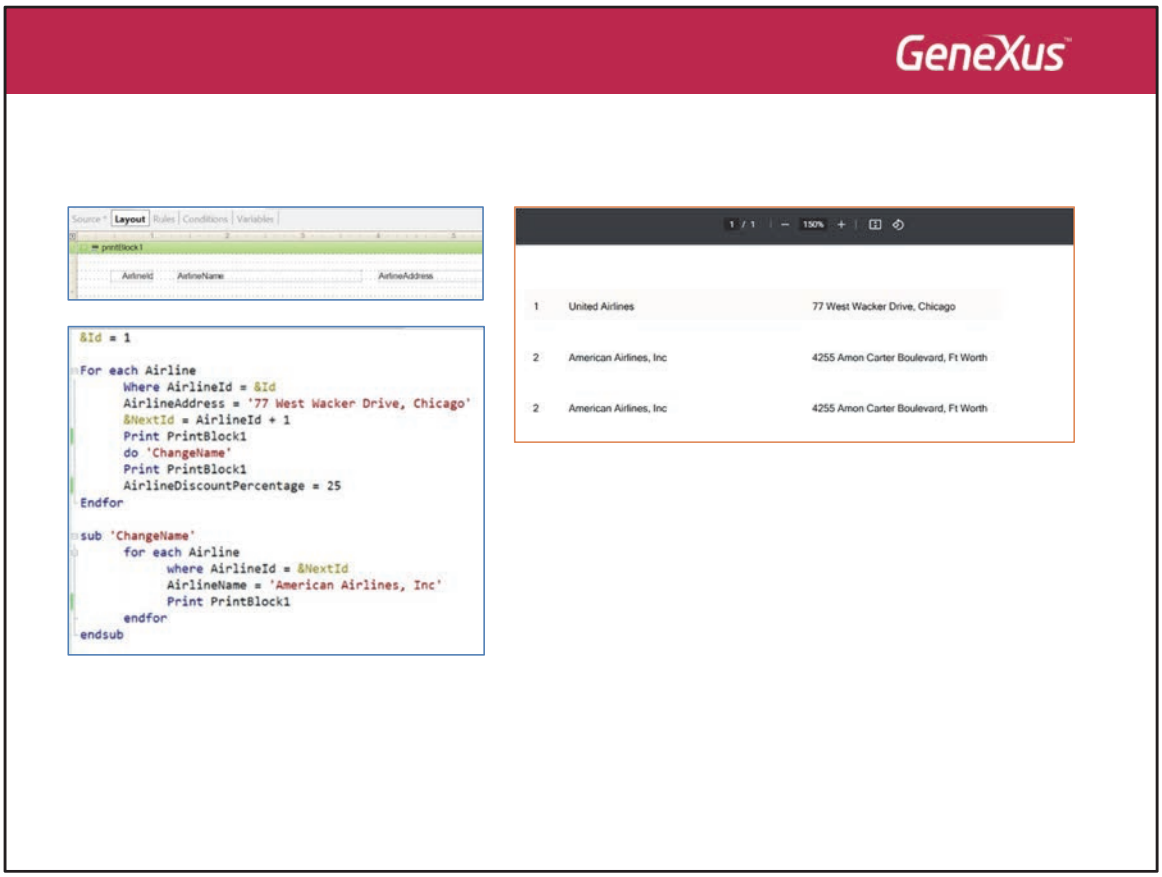
次に、別の例を使用して、もう少し詳しく説明します。



このアプリケーションには、さまざまな航空会社を複数の項目属性で記録する Airline トランザクションがあります。

このプロシージャーオブジェクトのソースには、ここに示すコードがあります。
この For each は、Airline テーブルを参照し、航空会社を ID 1 でフィルタします。
ID 1 はこの変数に割り当てられた値です。
このレコードのアドレスを AirlineAddress 項目属性を使用して更新する必要があります。
次に、AirlineId に 1 を足した値を割り当てる nextId という変数があります。このケースでは値が 2 になります。次に、ChangeName サブルーチンを呼び出します。
このサブルーチンも Airline テーブルに対して For each を実行し、AirlineId を 2 のレコードでフィルタします。
該当するレコードが見つかると、その名前が更新されます。

サブルーチンの実行が終了し、メインの For each に戻ると、どのレコードの位置になるでしょうか。ここで Airline トランザクションの項目属性、たとえば AirlineDiscountPercentage があった場合、それに値を割り当てると、どのレコードに適用されるでしょうか。サブルーチンを呼び出す前に位置していた ID 1 のレコードでしょうか。それとも、ID 2 のレコードでしょうか。



レイアウトで 3 つの項目属性を宣言し、航空会社の ID、名前、およびアドレスを表示して確認してみましょう。

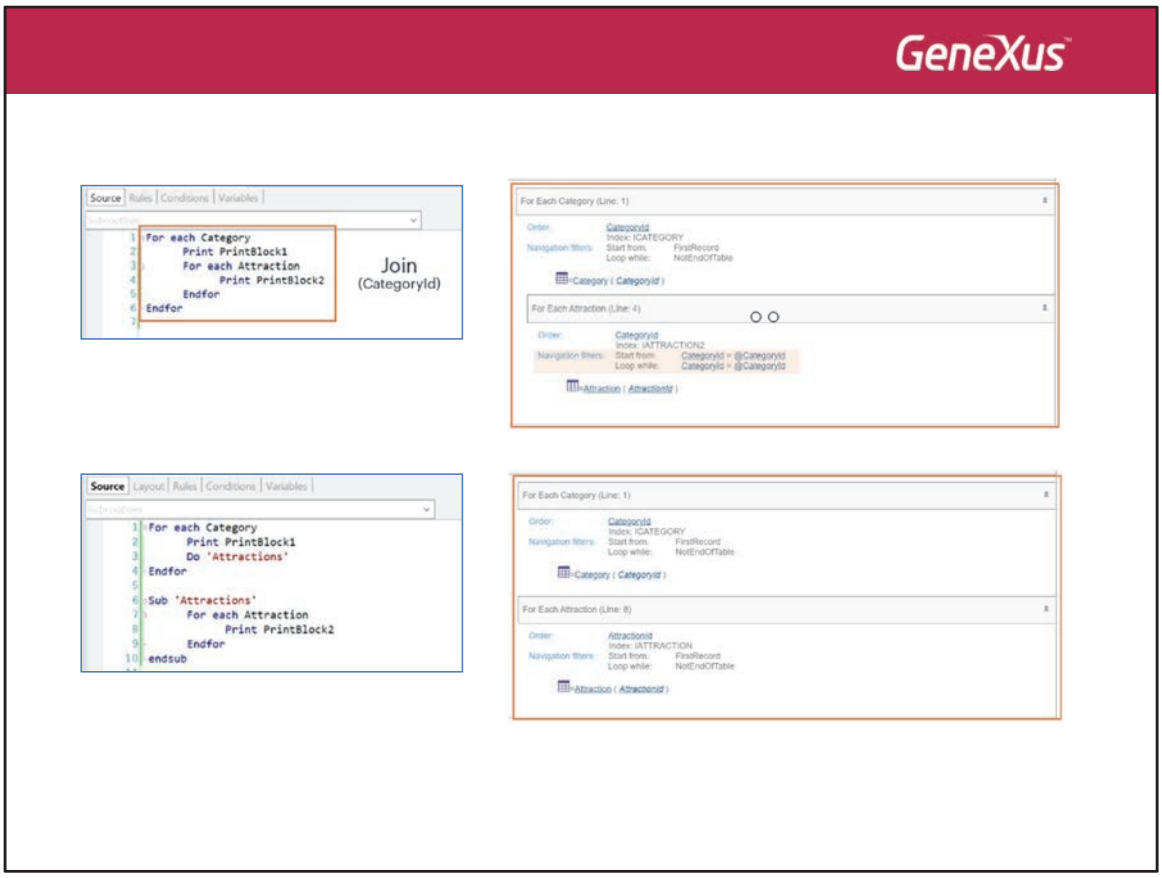
ソースでは、指定されたタイミングでどの航空会社に位置しているかを確認するために、3 つの Print Printblock を追加しています。1 つはサブルーチンを呼び出す前、1 つはサブルーチンの実行中、1 つはサブルーチンを離れた直後です。

試してみましょう。
サブルーチンを呼び出す前の最初の Print Printblock では、ID 1 の航空会社に位置され、アドレスは既に更新されています。
実行される 2 番目の Print Printblock は、サブルーチン内のものに対応します。
ID 2 のレコードが画面に出力され、新しい更新された名前が付けられています。
3 番目の Print Printblock は、サブルーチンを終了すると実行されます。そのタイミングでは、サブルーチン内での ID2 のレコードに位置したままです。サブルーチンを呼び出す前に位置していた、ID 1 のレコードではありません。
そのため、このタイミングで AirlineDiscountPercentage 項目属性を 1 つの値で更新すると、ID 2 のレコードに対して実行されます。
宣言された項目属性がオブジェクトに対してグローバルであるため、このように処理されます。そのため、オブジェクトの特定のセクションで項目属性が値を取得した後で、同じ項目属性に値を割り当てるサブルーチンが呼び出された場合、呼び出されたサブルーチンから戻り、その項目属性の値を照会すると、サブルーチンで割り当てられた値になります。これを AirlineId 項目属性で確認しました。

サブルーチンではパラメーターの送信はサポートされません。そのため、データのやり取りには、オブジェクトに対してグローバルな変数を使用します。

この動作が望ましくない場合は、サブルーチンを使用する代わりに、たとえばプロシージャーを呼び出すことができます。このケースでは、フィルタに使用する変数をパラメーターを使用して送信します。

次に、3 つ目の例を見てみましょう。



このケースには、Category テーブルをナビゲートする For each と Attraction テーブルをナビゲートするネストされた For each を実行するプロシージャーがあります。すべての観光名所にカテゴリが割り当てられるため、CategoryId によって結合が実行されます。CategoryId は、両方のテーブルを結合できる共通の項目属性です。ナビゲーション表示を見てみましょう。

プロシージャーのソースを見ると、2 つの Print を実装していることが分かります。1 つはカテゴリの名前を実装し、2 つ目は観光名所の名前を実装します。

これを実行すると、カテゴリの名前が出力され、その内部に、それと関連付けられたカテゴリの観光名所が出力されます。
この動作を望まない場合、つまり結合は実行されず、カテゴリが出力され、その後、どのカテゴリに属するかに関係なくすべての観光名所が出力される動作を望む場合は、どのように実装したらよいでしょうか。
1 つのオプションは、このコードをサブルーチン内に入れることです。
ナビゲーション表示を見てみましょう。
Category テーブル全体が参照されてから、Attraction テーブル全体が、最初のレコードから最後のレコードまで、どのようなフィルタも適用せずに確認されることが分かります。

このように、GeneXus は自動推論を実行せず、CategoryId によるフィルタも実行しません。2 つの別のナビゲーションを行います。

ここまで、サブルーチンの例と使用方法を見てきました。

GeneXus

サブルーチン

- Code ブロック
- 次の場所で使用可能:
  - Web パネル
  - プロシージャー
  - パネル
  - トランザクション
- Sub コマンドで定義し、Do コマンドで呼び出す
- 項目属性がオブジェクトに対してグローバルである
- For each はネストされない

簡単にまとめます。

サブルーチンとは、

- コードをモジュール化できる Code ブロックです。同じオブジェクト内であれば何度でも呼び出すことができます。
- Web パネル、プロシージャー、パネル、トランザクションなどで使用できます。
- Sub コマンドで定義し、Do コマンドで呼び出します。
- パラメーターの送信はサポートされておらず、データのやり取りには変数を使用します。
- 項目属性に値が指定されている場合、サブルーチンを呼び出すときに値が変わります。サブルーチンから戻り、値を照会すると、サブルーチンで割り当てられた値が示されます。これは、項目属性がそのオブジェクトに対してグローバルであるためです。
- For each 内から呼び出しが行われ、サブルーチンにも For each コマンドがある場合、For each はネストされません。つまり、推論もフィルタも行われません。

詳細については GeneXus の Wiki を参照してください。