データベースの更新

ビジネスコンポーネント: メソッドの違い

GeneXus™

GeneXus

データベースに対して実行されるメソッド

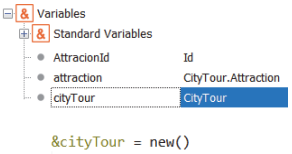
- Load()、Update()、Delete()、Insert()
- Save()、Load の代わりに割り当て、InsertOrUpdate()

&cityTour …
…
&cityTour.Save()

&cityTour …
…
&cityTour.Insert()

&cityTour …
…
&cityTour.Update()

TrnMode.Insert



TrnMode.Update

```
&cityTour.Load(2)
&cityTour.Insert()
&cityTour.Update()
```

TrnMode.Delete

```
&cityTour.Delete()
```

先に進む前に、単一レベルと 2 レベルのビジネスコンポーネントを比較した章の最後の部分を思い出してみましょう。

「メソッドは繰り返し使用されています。Load メソッドで既存の情報をビジネスコンポーネントにロードし、Update() または Delete() メソッドを組み合わせてデータの更新または削除を実行し、Insert() メソッドで挿入を実行します。

ほかの方法もあります。これらの違いおよびユースケースを明確に理解する必要があります。

たとえば、Save メソッドを使用すると、該当する操作を指定する必要がありません。これはトランザクションの [実行] ボタンと似ています。変数がデータの挿入または更新のいずれを試行するかは、変数のモードに依存します。Insert モードの場合は挿入、Update モードの場合は更新が試行されます。

たとえば、オブジェクトで定義されたすべてのビジネスコンポーネント変数は Insert モードになります。new() を使用して新しいメモリー領域を割り当てた場合も Insert モードになります。

Load() を使用して変数にロードした場合は、直ちに Update モードになります。何らかの操作を実行して成功したとき、挿入または変更の場合は Update モードになります。Delete() メソッドを適用した場合は Delete モードになります。

Save が見つかったときにどのような操作が試行されたか知るためには、この変数のコンテキストを知る必要があります。ただし、Insert メソッドおよび Update メソッドの場合はこれは必要ありません。モードに関係なく、データベースの情報の挿入または更新が実行されます」

GeneXus

データベースに対して実行されるメソッド

- Load()、Update()、Delete()、Insert()
- Save()、Load の代わりに割り当て、InsertOrUpdate()

```
&cityTour.Load(2)
&cityTour.CityTourName = “Paris, oh la la”
…
&cityTour.Save()
```

&cityTour

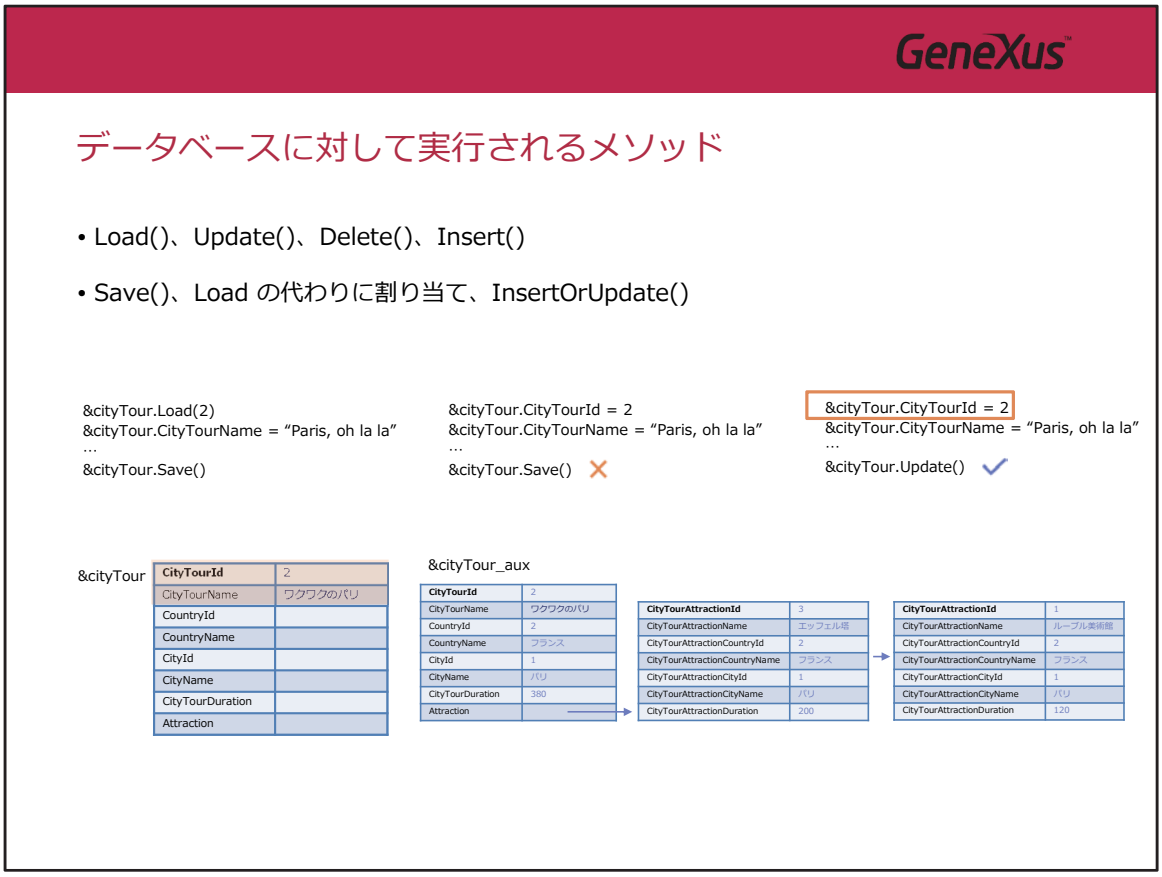
CityTourId	2
CityTourName	ワクワクのパリ
CountryId	2
CountryName	フランス
CityId	1
CityName	パリ
CityTourDuration	380
Attraction	

CityTourAttractionId	3
CityTourAttractionName	エッフェル塔
CityTourAttractionCountryId	2
CityTourAttractionCountryName	フランス
CityTourAttractionCityId	1
CityTourAttractionCityName	パリ
CityTourAttractionDuration	200

CityTourAttractionId	1
CityTourAttractionName	ルーブル美術館
CityTourAttractionCountryId	2
CityTourAttractionCountryName	フランス
CityTourAttractionCityId	1
CityTourAttractionCityName	パリ
CityTourAttractionDuration	120

この例を見るとよくわかります。

ご覧のように、Load 操作の後で市内観光 2 がある場合は、変数がデータベースから値をロードして Update モードになります。したがって、Save により更新が試行されます。違いのあるエレメントのみが更新されます。この場合は CityTourName です。

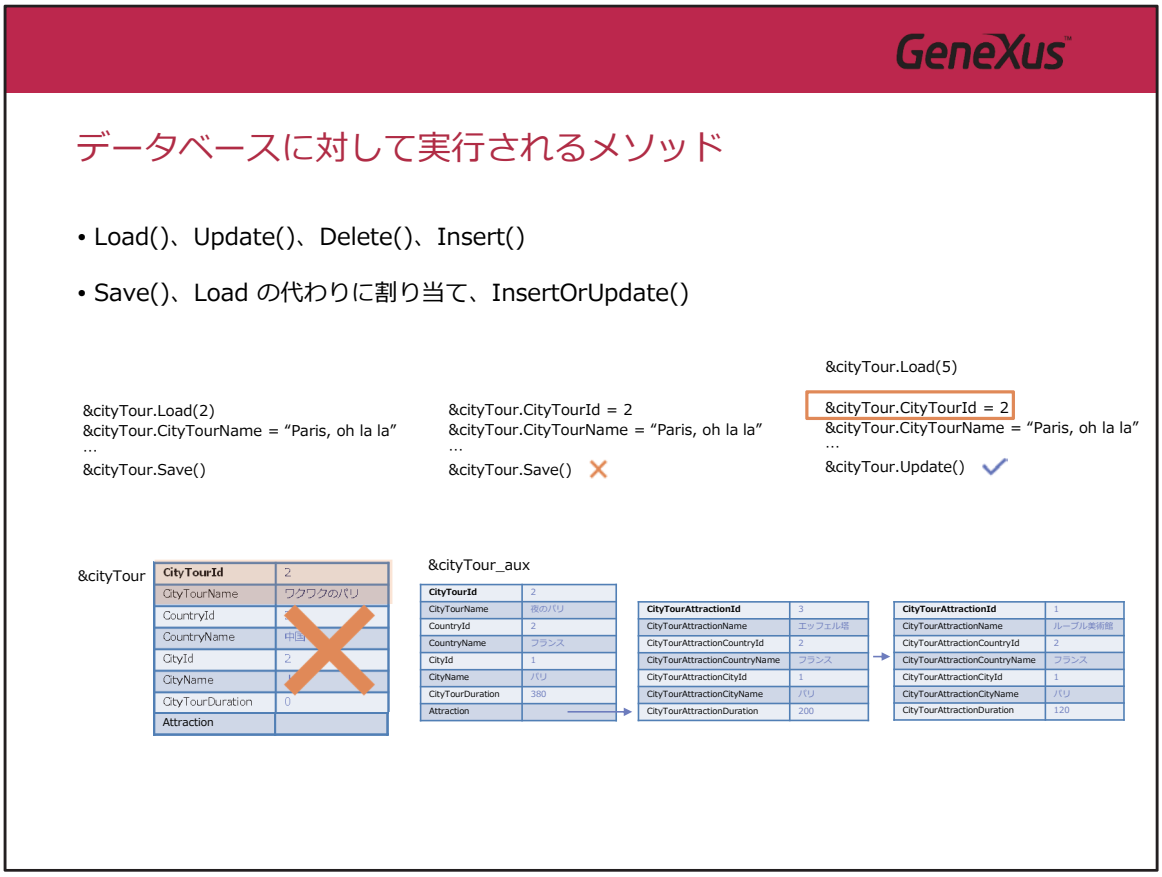


Load メソッドを使用する代わりに、主キーに値を直接割り当てるとどうなるでしょうか。
これは変数のモードによって異なります。たとえば、変数がオブジェクトで定義されているだけで、これが変数で実行する最初の操作である場合などは、Update モードではありません。そのため、Save 操作は市内観光 2 の挿入を試みますが、重複した主キーがあるために失敗します。

ただし、Save メソッドの代わりに Update メソッドを使用すれば、問題は発生しません。&cityTour 変数は空ではないエレメントを 2 つ持つだけなので、一見すると問題があるように思われるかもしれません。さらに、CityTour 2 が持つとわかっている 2 つの行はここには示されていません。したがって、Update 操作を実行すると、データベースの混乱を招くと考えるでしょう。
しかし、Update は高度な処理です。変数が Insert モードであるため、すべてのデータがロードされていないということを認識します (なぜなら、Load 操作が実行されていたなら、変数は Insert モードではなく Update モードになるためです)。
したがって、データベースでは、ビジネスコンポーネントで実際に割り当てられたデータのみが変更され、その他は変更されません。

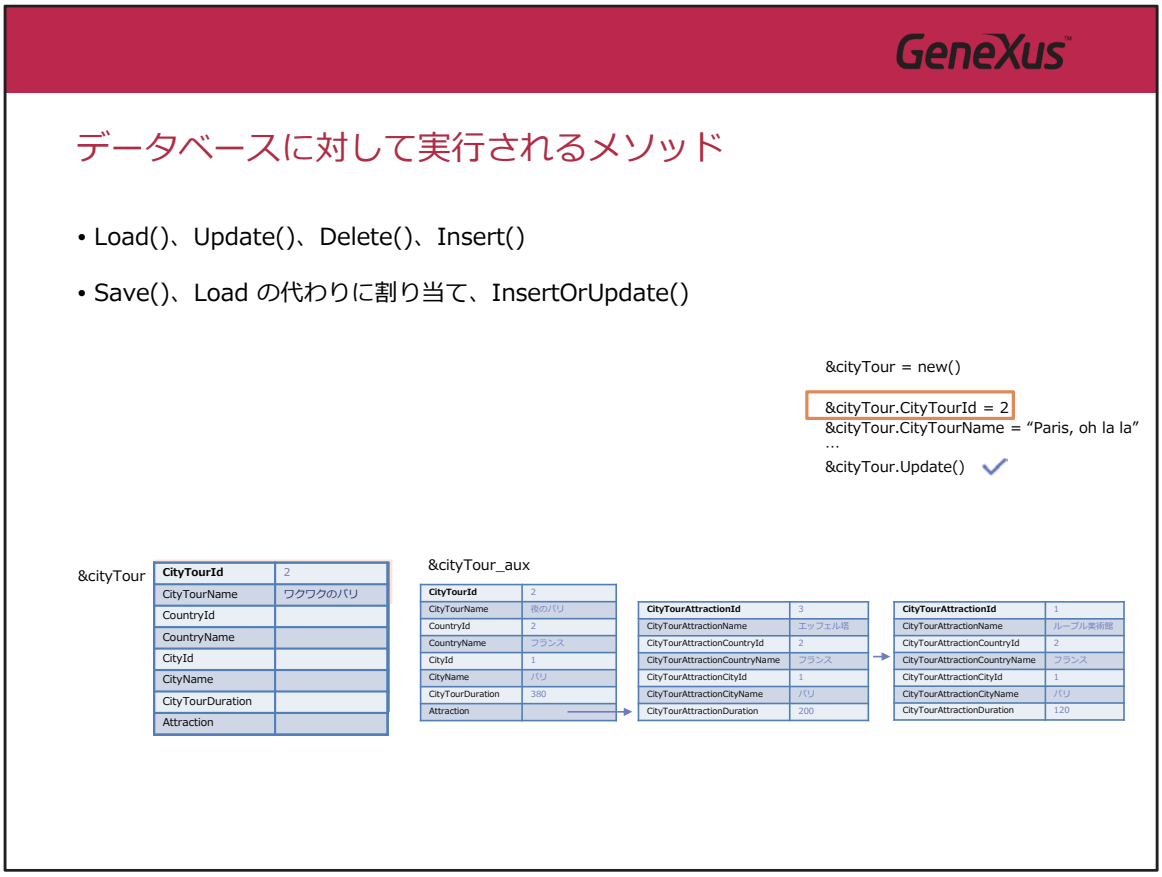
内部では、ビジネスコンポーネントの補助変数にデータがロードされます。もちろん、この変数は Update モードのままになります。Insert モードの変数で変更されているデータ (ここでは CityTourName のみ) が変更され、Save() 操作が実行されます。
補助変数はロードされたため Update モードであり、更新が実行されます。

Load 操作を実行せずにビジネスコンポーネント変数の主キーを直接割り当てるこの方法は、一般的には適切とはいえません。



たとえば、変数が前に使用されていて (たとえばヘッダーのみを持つ市内観光 5 がロードされたため) Update モードになっている場合、慎重に対応しないと、前の使用時の値がそのまま残され、不適切な値となる可能性があります。

データプロバイダーでビジネスコンポーネントをロードした場合は、InsertOrUpdate メソッドのところで説明するように、割り当てが唯一の方法です。



割り当てを使用する場合、まず、変数の新しいメモリー領域を要求することを強くお勧めします。これにより、変数がクリーンな状態になり、かつ Insert モードになります。

データプロバイダーでビジネスコンポーネントをロードした場合は、InsertOrUpdate メソッドのところで説明するように、割り当てが唯一の方法です。

GeneXus

データベースに対して実行されるメソッド

- Load()、Update()、Delete()、Insert()
- Save()、Load の代わりに割り当て、InsertOrUpdate()

Insert()
Update()
Delete()
InsertOrUpdate()

ビジネスコンポーネントの
コレクションにも使用可能

```
Parm( in: &cityTour )

If &cityTour.InsertOrUpdate()
   Commit
else
   Rollback
endif
```

```
If not &cityTour.Insert()
   for &message in &cityTour.GetMessages()
         if &message.Id = "DuplicatePrimaryKey"
           &cityTour.Update()
         endif
  endfor
endif
If &cityTour.Success
   Commit
else
   Rollback
endif
```

InsertOrUpdate メソッドは、ビジネスコンポーネントで処理している情報がデータベースにあるかどうかわからない場合に使用します。

たとえば、プロシージャーを使用し、ビジネスコンポーネント変数でデータを受け取る場合などです。この場合、コードでどのような情報がロードされるのかわかりません。しかし、たとえパラメーターであっても、このオブジェクトに対して新規であるため、変数が Insert モードになることはわかっています。読み出し元オブジェクトのモードは受け取りません。受け取るのはエレメントの値のみです。

したがって、Save() を実行しようとすると、ビジネスコンポーネントの主キーがデータベースにない場合のみ成功します。そうでない場合、キーの重複のため失敗します (変数は必然的に Insert モードになるため、Save の操作では挿入を試みます)。したがって、プロシージャーを呼び出したオブジェクトが更新のために Save メソッドを実行すると、問題が発生します。

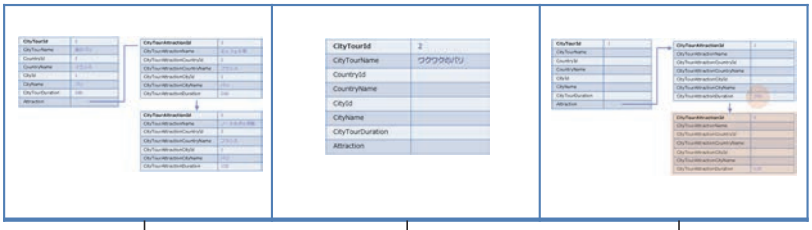
InsertOrUpdate メソッドを使用すれば、この両方の可能性をプログラムする手間を省くことができます。つまり、挿入を試行し、(ほかの理由ではなく) キーの重複のため失敗した場合、更新を試行します。これが InsertOrUpdate() の動作です。

また、既に述べたように、Insert、Update、Delete および InsertOrUpdate はビジネス コンポーネント コレクション タイプの変数に対して使用できます。この場合も、InsertOrUpdate メソッドが大いに役立ちます。

GeneXus

データベースに対して実行されるメソッド

&cityTours



Insert　　Update　　Update

&cityTours = GetCityTours(…) ⟶ データプロバイダーのソース?

```
If &cityTours.InsertOrUpdate()
    Commit
else
    Rollback
endif
```

ここにビジネス コンポーネント コレクション タイプの変数があります。データプロバイダーが返す内容に基づいて、またはコードを介して、同じプログラムでこの変数をロードできます。この場合、データプロバイダーは CityTour ビジネス コンポーネント タイプのアイテムのコレクションを返します。ここでの目的は、コレクションの各ビジネスコンポーネントで実行された処理をデータベースに適用することです。

コレクションの最初のアイテムは挿入する情報に相当します (この場合はヘッダーとその 2 行)。2 番目のアイテムは変更する情報で (たとえば、ヘッダーの CityTourName のみ)、最後のアイテムもやはり変更する情報です (この例では 1 行を変更し、別の 1 行を追加します)。

この場合、操作は各ビジネスコンポーネント変数のモードには依存できません。すべての場合にモードが Insert になるからです。なぜなら、Load メソッドを利用していないためです。
このため、コレクションを参照して Save することはできません (Save は挿入する最初のエレメントの場合は機能しますが、その他の場合は、挿入しようとすると重複したキーが見つかるため失敗します)。

したがって、操作は変数のモードではなく、データベースに主キーが存在するかしないかに依存します。つまり、この場合も適切なメソッドは InsertOrUpdate です。コレクションに適用すると、各アイテムに適用されます。最初のアイテムの場合は挿入を試み、成功します。第 2 のアイテムの場合も挿入を試みますが、重複したキーが見つかるため、更新を試みて成功します。最後のアイテムの場合も同様です。

操作の結果で処理を分岐できます。コレクション全体で成功した場合はコミットを実行できます。そうでない場合、一部のレコードが変更できていますが、すべてを戻すために、ロールバックできます。

これまでの説明に基づいて考えてください。このようにデータをロードするには、データプロバイダーをどのように宣言すればいいでしょうか。

データプロバイダーのソース

Output

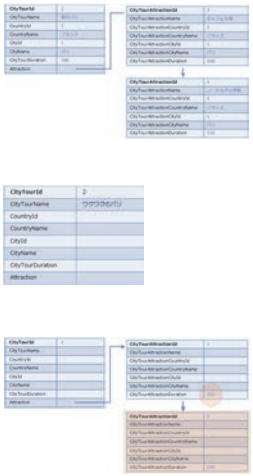
Infer Structure	No
Output	**CityTour**
Collection	**True**
Collection Name	GetCityTours

GetCityTours *

Source * | Rules | Variables

```
CityTour
{
    CityTourId = 2
    CityTourName = Paris at night
    CountryId = find(CountryId, CountryName = "France")
    CityId = find(CityId, CityName = "Paris")
    Attraction
    {
        CityTourAttractionId = 3
        CityTourAttractionDuration = 260
    }
    Attraction
    {
        CityTourAttractionId = 5
        CityTourAttractionDuration = 120
    }
}
CityTour
{
    CityTourId = 2
    CityTourName = "Paris, oh la la"
}
CityTour
{
    CityTourId = 1

    Attraction
    {
        CityTourAttractionId = 1
        CityTourAttractionDuration = 150
    }
    Attraction
    {
        CityTourAttractionId = 5
        CityTourAttractionDuration = 120
    }
}
```



ここに概要を示します。データプロバイダーは CityTour ビジネス コンポーネント データ タイプのコレクションを返します。**コレクション**です。

CityTourId は自動採番ではなく、2 は存在しないとみなされます。返されるコレクションの 3 つのアイテムについて、3 つの CityTour グループがあります。最初のものは、すべてのヘッダーデータと 2 つの行をロードします。

第 2 のものは、最初のものに対する Insert 操作が既に実行されたとみなし、同じ CityTour に対して名前の変更を試みます。

第 3 のものは、ID 1 の行の値を変更し、ID 5 の行を追加します (ここでは、Attraction レベルの格納された項目属性がこれら 2 つのみであるため、違いには気づきません)。しかし、テーブルにさらに多くの項目属性がある場合は、すべて (または値を指定したいすべて) がここに表示されます。この場合は、変更したいもののみが表示されています。