

プロシージャ-用の  
特殊なコマンドを使用した  
データベースの更新

*GeneXus*<sup>TM</sup>

## データベースの更新

### トランザクションを使用

### ビジネスコンポーネントを使用

Name	Type	Description	For...	Nullable
Attraction	Attraction	Attraction		
AttractionId	Id	Attraction Id	No	
AttractionName	Name	Attraction Name	No	
CountryId	Id	Country Id	No	
CountryName	Name	Country Name		
CityId	Id	City Id	Yes	
CityName	Name	City Name		
CategoryId	Id	Category Id	Yes	
CategoryName	Name	Category Name		

```

Event 'New Attraction'
&Attraction.AttractionName = "Forbidden City"
&Attraction.CountryId = find( CountryId, CountryName = "China")
&Attraction.CityId = find( CityId, CityName = "Beijing")
&Attraction.CategoryId = find( CategoryId, CategoryName = "Monument")
&Attraction.Save()
Commit
Endevent
  
```

これまで、データベースのデータを更新するには、次の 2 とおりの方法でトランザクションを使用しました：

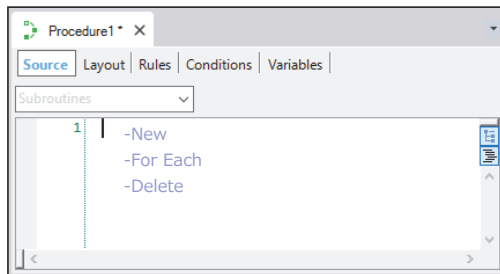
- 画面を実行し、対話形式でデータを入力する方法
- 画面を使わず、ビジネスコンポーネントとして実行し、変数を使用する方法

たとえば、新しい観光名所を追加する場合、開発者は、ボタンを配置した Web パネルにスライド右下に示されているイベントを記述して追加することもできます。

## データベースを更新する別の方法

Procedure オブジェクト

データベース



ここでは、データベースに対して挿入、変更、削除を行う別の方法について学びます。

この方法は、プロシージャータイプのオブジェクトにのみ使用できることに注意してください。あらゆるオブジェクトに使用できるビジネスコンポーネントの方法とは異なります。

## プロシージャーでのみ有効

### 挿入

#### New <テーブル項目属性>

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



#### EndNew

Transaction integrity

Commit on exit Yes

このコマンドを使用することで、1 つの物理テーブルの項目属性に値を割り当て、1 つのレコードを挿入できる。

プロシージャーには、テーブルにレコードを挿入するための New というコマンドがあります。

このコマンドを使用することで、1 つの物理テーブルの項目属性に値を割り当てることができます。

ここでトランザクションではなくテーブルを挙げているのは、トランザクション構造内のすべての項目属性が物理テーブルに含まれるわけではないからです。

たとえば、観光名所を挿入する場合、Attraction トランザクションの構造で宣言されていても ATTRACTION テーブルには含まれない項目属性が多数あります。そのような項目属性は、ATTRACTION テーブルの拡張テーブルに含まれています。これらは、フォームに表示したり、ルールで使用したりするために構造に含めます。

## Attraction テーブル内の項目属性と New コマンド

Attraction トランザクション

Name	Type	Description	Formula	Null...
Attraction	Attraction	Attraction		
AttractionId	Id	Attraction Id		No
AttractionName	Name	Attraction Name		No
CountryId	Id	Country Id		No
CountryName	Name	Country Name		
CategoryId	Id	Category Id		Yes
CategoryName	Name	Category Name		
AttractionPhoto	Image	Attraction Photo		No
CityId	Id	City Id		Yes
CityName	Name	City Name		

Attraction テーブル

Name	Type	Description	Formula
Attraction Structure	Attraction	Attraction	
AttractionId	Id	Attraction Id	
AttractionName	Name	Attraction Name	
CountryId	Id	Country Id	
CategoryId	Id	Category Id	
AttractionPhoto	Image	Attraction Photo	
CityId	Id	City Id	

New コマンドは、これらの項目属性に値を割り当てることができる

開発環境で [表示] > [テーブル] のあと、Attraction テーブルを選択することで、ATTRACTION テーブルに属する項目属性のみを見ることができます。

これらの項目属性は、New コマンドを使用して値を割り当てることができます。New コマンドでは、1 つのレコードのみがテーブルに挿入されます。

## プロシージャの内容

### 挿入

```
new
  AttractionName = "Forbidden City"
  CountryId = find( CountryId, CountryName = "China" )
  CityId = find( CityId, CityName = "Beijing" )
  CategoryId = find( CategoryId, CategoryName = "Monument" )
endnew
```

ベーステーブル

**ATTRACTION**

**AttractionId:** AttractionId 項目属性は自動採番のため、値を割り当てない

**AttractionPhoto:** 写真がなくてもレコードは挿入される

等号の左側の項目属性を分析することで、GeneXus はレコードの挿入対象の物理テーブルを判断します。

すべて同じ物理テーブルに属する場合、レコードはそのテーブルに挿入されます。それ以外の場合、挿入するテーブルを選択できないことが通知されます。

GeneXus が特定したテーブルが New コマンドのベーステーブルとなります。この例では ATTRACTION です。

ここでは、AttractionPhoto 項目属性に画像を割り当てていません。この場合、挿入は行われますが、AttractionPhoto 項目属性は割り当てられず、観光名所は写真なしで作成されます。

また、外部キーである CountryId、CityId、CategoryId の各項目属性に値を割り当てないままにすることも可能です。あるいは、Find 式を使用して対応する ID を検索せずに、存在しない ID を入力することもできます。New コマンドでは参照整合性がチェックされないからです。この理由より、このコマンドを使用する際には細心の注意が必要です。

## New コマンドを使用した挿入とビジネスコンポーネントを使用した挿入

プロシージャー内で New 節を使用

```
new
  AttractionName = "Forbidden City"
  CountryId = find(CountryId, CountryName = "China")
  CityId = find(CityId, CityName = "Beijing")
  CategoryId = find(CategoryId, CategoryName="Monument")
endnew
```

Transaction integrity  
Commit on exit Yes

ビジネスコンポーネントを使用

```
Event 'New Attraction'
  &Attraction.AttractionName = "Forbidden City"
  &Attraction.CountryId = find( CountryId, CountryName = "China")
  &Attraction.CityId = find( CityId, CityName = "Beijing")
  &Attraction.CategoryId = find( CategoryId, CategoryName = "Monument")
  &Attraction.Save()
  Commit
Endevent
```

挿入を行うとき、GeneXus は、すでに存在する主キーではない挿入ということだけをチェックします。また、ある項目属性に対して一意のインデックスが定義されている場合は、その項目属性に重複する値が入力されていないこともチェックされます。重複があった場合、New コマンドは何も行わず、ユーザーへの警告もありません。この状況を特定し、適切な対応をとる方法については後述します。一意性の制御が唯一の制御であり、参照整合性の制御はありません。このため、実行時に存在しない CountryId を割り当てた場合、参照整合性のエラーが発生します。これがビジネスコンポーネントを使用して挿入を行う場合とは異なります。ビジネスコンポーネントの場合は、一意性と参照整合性の制御が行われます。また、トランザクションで定義されているルールがトリガーされます。

ビジネスコンポーネントを使用して挿入を行う場合、ほかのオブジェクトから行う場合と同様に、明示的に Commit を実行する必要があります。つまり、最後の Commit 操作時のデータ変更を維持するようデータベースに指示します。

プロシージャーには [Commit on exit] プロパティがあり、既定で Yes に設定されています。これはどのような影響があるのでしょうか。プロシージャーがデータベースにアクセスする場合、コードの末尾に Commit コマンドが追加されるため、開発者が指定する必要がありません。実際には、オブジェクトに記述されるのではなく、生成されるプログラムに追加されます。このため、コミット操作をすぐに実行する場合を除き、New コマンドの直後に明示的に Commit を追加する必要はありません。

New、For Each を使用して更新や削除（詳細は後述）を行うとき、GeneXus では、プロシージャー内でデータベースにアクセスすることが認識されます。このような場合は、暗示的に Commit が追加されます。それ以外の場合、データベースへのアクセスが必要であることが理解されないため、Commit は追加されません。このため、ビジネスコンポーネントを使用したデータベースの操作では暗示的なコミットは追加されません。

たとえば、次のように入力するとします：

```
&BC.element1 = ...  
&BC.element2 = ...  
&BC.Save()
```

この場合、Commit コマンドは追加されません。挿入を行うことが認識されないからです (Insert メソッドを直接使用した場合も同様です)。結果的に、ビジネスコンポーネントはほかの SDT と同じように処理されます。この場合は、Commit コマンドを明示的に記述する必要があります。

プロシーチャーのコード内に、更新または削除を行う New、For Each もある場合は、データベースへの接続が既に確立されているため、Commit を明示的に記述する必要はありません。プロシーチャーに上記のコードのみがある場合は、Commit コマンドを明示的に追加する必要があります。



## プロシージャでのみ有効

挿入

```
New  
  CategoryName = "Tourist site"  
EndNew
```

ベーステーブル  
**CATEGORY**

**CategoryId (?)**

New 節で新しいレコードを作成するには、テーブルレコードに属する項目属性に値を割り当てます。

プロシージャ内の New 節で、上記のように CategoryName 項目属性にのみ値を割り当てた場合、ベーステーブルが CATEGORY であることは明らかなです。このテーブルに CategoryName 項目属性が物理的に格納されているからです。

既に説明したとおり、CategoryId 項目属性は自動採番のため、値は割り当てません。

## プロシージャでのみ有効

### 更新

```
For each Attraction
  Where CityName = 'Beijing'
  Where CategoryName = 'Monument'
  CategoryId = find(CategoryId, CategoryName = 'Tourist site')
Endfor
```

ベーステーブル

**ATTRACTION**

ATTRACTION の拡張テーブルの項目属性は、For Each コマンドで更新できる  
(主キーを除く)

次に、データベース内に既存する値を更新する方法を見ていきます。

テーブルレコードの項目属性に格納されている値を別の値に置換するには、For Each コマンドを使用してそのレコードにアクセスし、割り当てによって新しい値を付与します。

この例では、For Each コマンドのベーステーブルは ATTRACTION です。ベーストランザクションが Attraction だからです。その上で、北京で「遺跡」カテゴリを持つ観光名所をすべて参照しています。これらの観光名所のカテゴリを「観光地」に変更します。

この例では、Where 節の条件を満たす多数のレコードを更新しています。ここでは単一の項目属性の値を更新していますが、複数の項目属性を更新することもできます。また、拡張テーブルの項目属性を更新することも可能です。たとえば、観光名所のカテゴリ名、観光名所の国名、観光名所の都市名などです。これについては次のページで確認します。

New コマンドでは、テーブルに物理的に存在する項目属性にのみ値を割り当てることができます。新しいレコードを 1 つ挿入することだけが目的だからです。これに対して、For Each コマンドでは、常に既存のレコードが対象であるため、そこから拡張テーブルを通じて関連付けられているすべてのレコードを編集できます。

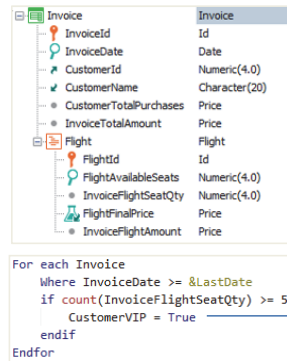
候補キー (1 つの項目属性に対して定義されている一意のインデックス) があり、For Each コマンドでその値をほかのレコードにすでに存在している別の値に置換しようとした場合、更新は行われません。この場合、前述の New コマンドと同様に、警告は表示されません。このような場合を特定し、適切な対応をとる方法については後述します。

For Each コマンド内で、参照対象となるテーブルの主キーの変更を許可しない制限事項があります。

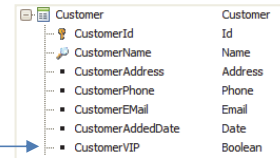
## プロシージャでのみ有効

### 更新

#### Invoice トランザクション



#### Customer トランザクション



旅行代理店が、特定の日付以降、同一の請求書でフライト数が 5 便以上の顧客を特定し、VIP としてマークしたいと考えたとします。そのために、Customer トランザクションに CustomerVIP 項目属性を追加します。この項目属性は対応するテーブルに格納されます。

対象となる顧客を特定し、マークするには、Invoice テーブル全体を走査し、それぞれのフライト数を数える必要があります。数が 5 便を超える場合は CustomerVIP 項目属性の値を True に更新します。

ここでは、For Each コマンドのベーステーブルは INVOICE です。そのため、Count 式は、このテーブルに属するフライトのみが数えられます。フライト数が 5 便以上の場合は、参照しているテーブルの拡張テーブルである CUSTOMER テーブルの項目属性の値が更新されます。

## 更新における For Each コマンドとビジネスコンポーネントの比較

### プロシージャーで For Each コマンドを使用

```
For each Attraction
  where CityName = "Beijing" and CategoryName = "Monument"
  CategoryId = find( CategoryId, CategoryName = "Tourist Site")
endfor
```

Transaction integrity
Commit on exit Yes

✓ PK と CK の一意性の制御

### ビジネスコンポーネントを 使用

```
For each Attraction
  where CityName = "Beijing" and CategoryName = "Monument"
  &Attraction.Load(AttractionId)
  &Attraction.CategoryId = find( CategoryId, CategoryName = "Tourist site")
  &Attraction.Save()
Endfor
Commit
```

✓ PK と CK の一意性の制御

✓ RI の制御

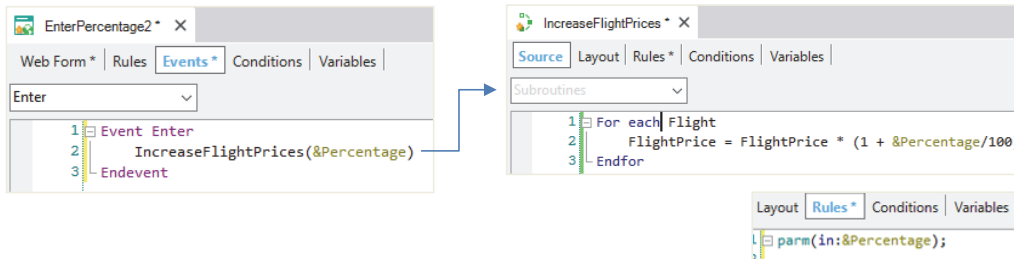
✓ ルール

前述のように、プロシージャー内で For Each コマンドを使用した場合、GeneXus によって制御されるのは、主キーと候補キーの一意性のみです。

これに対して、ビジネスコンポーネントを使用した場合は、参照整合性もチェックされます。このため、存在しない値を CategoryId に割り当てた場合、この更新は許可されません。さらに、ビジネスルールも実行されます。

注意: New コマンドのところで説明した [Commit on exit] プロパティも思い出してください。

## 価格を上げるプロシージャ



ユーザーにパーセント値の入力を求め、その値を適用してすべてのフライトの価格を計算し直すとします。

そのために、ユーザーにデータ入力を求める Web パネルを作成します。そこから FLIGHT テーブルを更新するプロシージャを呼び出し、パラメーターで受け取ったパーセント値に従って価格の値を変更します。

更新を行うには、For Each コマンドで FLIGHT テーブルを参照し、FlightPrice 項目属性の値を、ここに示すエクスプレッションの結果に置き換えます。

## 方法の比較

### ビジネスコンポーネントを使用

```

1 Event Enter
2   For each Flight
3     $BCFlight.Load(FlightId)
4     $BCFlight.FlightPrice = $BCFlight.FlightPrice * (1+ &Percentage/100)
5     $BCFlight.Save()
6     If $BCFlight.Success()
7       Commit
8     Else
9       Rollback
10    Endif
11  Endfor
12 Endevent

```

### プロシージャーを使用

EnterPercentage2 \* X

Web Form \* Rules Events \* Conditions Variables |

Enter

```

1 Event Enter
2   IncreaseFlightPrices(&Percentage)
3 Endevent

```



IncreaseFlightPrices \* X

Source Layout Rules \* Conditions Variables |

Subroutines

```

1 For each Flight
2   FlightPrice = FlightPrice * (1 + &Percentage/100)
3 Endfor

```

データベースを更新する 2 つの方法を比較します。

最初の方法では、EnterPercentage Web パネルの Enter イベントで For Each コマンドを入力し、Flight トランザクションをビジネスコンポーネントとして使用してデータベースを更新しています。

2 つ目の方法では、EnterPercentage2 Web パネルの Enter イベントにプロシージャーの呼び出しのみが含まれています。プロシージャーでパーセント値を受け取り、For Each コマンドでフライトごとに新しい価格を計算して割り当てます。

これらの方法の違い、それぞれのメリットとデメリットは何でしょうか。

## カテゴリの挿入 + 観光名所の変更の例

```

1 New
2   CategoryName = 'Tourist Site'
3 Endnew
4 For each Attraction
5   Where CityName = 'Beijing' and CategoryName = 'Monument'
6   CategoryId = find(CategoryId, CategoryName = 'Tourist Site')
7 Endfor

```

Procedure オブジェクトでのみ有効

```

1 Event 'Do'
2   &category.CategoryName = "Tourist site"
3   &category.Save()
4   For each Attraction
5     Where CityName = "Beijing" and CategoryName = "Monument"
6     &Attraction.Load(AttractionId)
7     &Attraction.CategoryId = &category.CategoryId
8     &Attraction.Save()
9   Endfor
10  Commit
11 Endevent

```

任意のオブジェクトで有効

これらは、新しいカテゴリを挿入し、特定の観光名所の値をこのカテゴリに置き換える方法を示しています。

Web パネルの [Events] エLEMENTのコードは、どの GeneXus オブジェクトでも実行されます。一方、プロシーチャーのコードは Procedure オブジェクト内でのみ有効であることを覚えておいてください。

プロシーチャーを確認します。

CategoryId は自動採番であるため、New コマンドが失敗することはありません。Find 式を使用して、直前に New コマンドで挿入した「観光地」というカテゴリの ID を探します。カテゴリ名の入力にミスがあった場合、Find 式でレコードは見つからず、空の値、数値では 0 が返されます。この場合、北京にある、カテゴリが遺跡の観光名所のカテゴリ ID を値 0 に変更することになります。CategoryId 項目属性で null 値が許可されていない場合、データベース内で不整合が生じます。For Each コマンドを使用して更新を行う場合、参照整合性チェックが行われないことは既に説明したとおりです。これに対して、ビジネスコンポーネントの場合は参照整合性がチェックされます。この例では、CategoryId で null 値が許可されているため、プロシーチャーを通じてこの方法を使用した場合に整合性の問題は生じません。

値の更新にはビジネスコンポーネントを使用するのが安全です。この場合、New コマンドおよび For Each コマンドと同じ制御に加え、参照整合性のチェック、関連付けられたトランザクションで指定されたルールが実行されるからです。プロシーチャーのプログラミングを入念に行っても、対応する現実がいつ変化するかわかりません。たとえば、プロシーチャーのプログラミング中には思いつかなかったエラールールをトランザクションに追加する必要がある可能性があります。ビジネスコンポーネントを使用することで、こうしたルールをすべてのチェックに含めることができます。

## プロシージャでのみ有効

削除:

```
For each Category
    where CategoryName = "Tourist Site"
    Delete
Endfor
```

---

```
For each Attraction
    Delete
Endfor
```

```
For each Category
    Delete
Endfor
```

プロシージャを使用してデータベースのレコードを挿入、更新する方法を見てきました。次に、レコードを削除する方法を見ていきます。

レコードを削除するには、For Each コマンド内で Delete コマンドを使用します。

Delete コマンドは、指定したテーブルからレコードを削除します。そのレコードにアクセスするために For Each コマンドを使用します。

最初の例では、Category テーブルに「観光地」のフィルタを適用しています。このため、Delete コマンドで削除しているレコードは 1 つだけです。

テーブルを空にしたい場合は、すべての観光名所とすべてのカテゴリを削除することもできます。

先にカテゴリを削除し、その後、観光名所を削除した場合、データベースでは既定で参照整合性がチェックされるため、最初のカテゴリを削除しようとしたときに、関連する観光名所があった場合、削除は失敗し、プログラムも失敗します。これについては後述します。



## 特殊なコマンドの使用 (New、For Each、Delete)

データ検証: 一意性のみ、参照整合性はない

プロシージャーでのみ有効

## ビジネスコンポーネントの使用

データ検証: 一意性と参照整合性の両方

トランザクションルールがトリガーされる

任意のオブジェクトで有効

ビジネスコンポーネントを使用した場合、データベースで更新されるデータの整合性がチェックされ、ビジネスコンポーネントとして実行されているトランザクションで指定されているルールが実行されます。

メッセージを生成する msg や error などのルールもトリガーされます。対応するメッセージはコレクションに保存され、逐次的に参照し出力できます。

プロシージャーでは、これらの処理はいずれも実行されません。

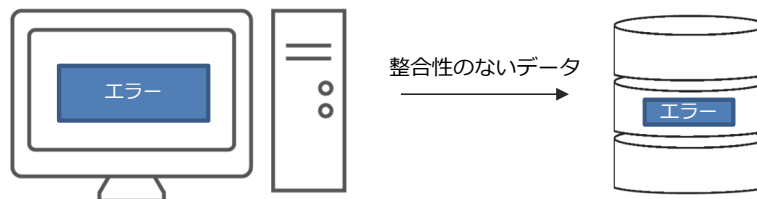
繰り返しになりますが、For Each コマンドは Web パネルで使用することもできますが、この場合、項目属性に値を割り当てることでデータベースを直接変更することはできません。変更は Procedure オブジェクトからのみ行うことができます。また、Web パネル内で New コマンドを記述することはできず、For Each コマンドに Delete コマンドを含めることもできません。これはプロシージャー内でのみ行うことができます。

これに対して、ビジネスコンポーネントを使用したデータベースの変更は任意のオブジェクトで可能です。

プロシージャーでは、割り当てるデータの整合性がチェックされないことを考慮に入れる必要があります。チェックされるのは、主キーと候補キーの一意性のみです。

## 特殊なコマンドの使用 (New、For Each、Delete)

関連性の高い有効なデータを割り当てる責任は開発者にある



プロシージャーでこれらのコマンドを使用して直接更新を行うときは、格納されているほかのデータと整合性のあるデータを割り当てまたは挿入する責任があります。

前の New コマンドの例では、CategoryName には任意の値を割り当てることができます。このデータはほかのテーブルと関係がないからです。

これに対して、観光名所のカテゴリを更新した例では、外部キーである項目属性 CategoryId に新しい値を割り当てました。割り当てた値が CATEGORY テーブルにカテゴリ ID として存在しない場合、プロシージャーで検証されないため、整合性のないデータが入力される可能性があります。

データベースでは、相互に関連するデータの整合性がチェックされるため、ユーザーがアプリケーションを実行して整合性のない値を割り当てようとすると、データベースでその操作が拒否され、整合性のないデータは保存されません。

しかし、プログラムは実行が停止されるため、ユーザーの操作性の面で問題となります。

したがって、プロシージャーを使用してデータベースを更新する場合は、関連性の高い有効なデータを割り当てる責任を開発者が負うことになります。

## 特殊なコマンドの使用 (New、For Each、Delete)

データ検証で一意性のみチェックする場合: 主キーまたは候補キーの重複が検出された場合に実行する処理のプログラミングはどこで行うか

```
new
  AttractionName = "Forbidden City"
  CountryId = find( CountryId, CountryName = "China")
  CityId = find( CityId, CityName = "Beijing")
  CategoryId = find( CategoryId, CategoryName = "Tourist Site")
  when duplicate
endnew
```

AttractionName は候補キーで、重複が認められないとします。New コマンドを実行し、AttractionName に値「紫禁城」を割り当て、新しいレコード (ID は自動採番) を挿入するとします。このとき、既に同じ値のレコードが存在していた場合には、その対処を New コマンドの When duplicate 節に記述できます。

## 特殊なコマンドの使用 (New、For Each、Delete)

### 主キーまたは候補キーの重複

```
for each CreditCard
  where CreditCardCode = &creditCardCode
  CustomerId = &newCustomerId
  when duplicate
endfor
```

CreditCard		CreditCard
CreditCardCode	Code	
CustomerId	Id	
CustomerName	Name	
CreditCardLimit	Limit	

Customer		Customer
CustomerId	Id	
CustomerName	Name	
CustomerAddress	Address	
CustomerPhone	Phone	
CustomerEMail	Email	

同様に、Customer と CreditCard の 2 つのトランザクションがあり、1 対 1 の関係にあるとします。つまり、CustomerId が、一意のインデックスを通じて候補キーに設定されています。プロシージャーでは、パラメーターを通じて新しい顧客の ID と既存のクレジットカードの ID の 2 つの変数を受け取り、For Each コマンドを使用してカードの顧客を新しい顧客に変更するとします。同じ顧客の別のカードがデータベースに既存する場合、For Each コマンドで更新は行われません。この場合に何らかの処理を行う場合は、When duplicate 節を使用できます。

この節の詳細については、Wiki  
(<http://wiki.genexus.jp/hwikibypageid.aspx?24843>) を参照してください。

When duplicate 節に含まれる項目属性は、For Each コマンドのベーステーブルを判断する際には考慮されません。

## For Each の構文

```

For each      BaseTransaction

    skip <エクスプレッション>1 count <エクスプレッション>2
    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>]
    using <データセクター>(<パラメーター>1' <パラメーター>2' ... , <パラメーター>n)
    unique <項目属性>1' <項目属性>2' ... , <項目属性>n
    where <条件> [when <条件>]
    where <条件> [when <条件>]
    where <項目属性> IN <データセクター>(<パラメーター>1' <パラメーター>2' ... , <パラメーター>n)
    blocking N
    メインコード
    When duplicate
    ...
    When none
    ...
endfor

```

Blocking 節を使用すると、N レコード単位でデータベースを更新できます。この場合、データベースへのアクセス回数が減るため、パフォーマンスが向上します。ここでは詳細な説明は省きます。詳細については、こちらの記事を参照してください：  
<http://wiki.genexus.jp/hwikibypageid.aspx?4837>

When duplicate 節は、For Each コマンドと New コマンドのどちらにも使用できます。詳細については、次のリンクをクリックしてください：  
<http://wiki.genexus.jp/hwikibypageid.aspx?24843>

New コマンドの詳細な構文については、次のリンクをクリックしてください：  
<http://wiki.genexus.jp/hwikibypageid.aspx?6714>

繰り返しになりますが、When duplicate 節に含まれる項目属性は、For Each コマンドのベーステーブルを判断する際には考慮されません。



動画	<a href="https://www.genexus.com/community-and-support-jp/training?ja">https://www.genexus.com/community-and-support-jp/training?ja</a>
ドキュメント	<a href="http://wiki.genexus.jp/">http://wiki.genexus.jp/</a>
認定資格	<a href="https://training.genexus.com/certifications">training.genexus.com/certifications</a>