ダイナミックトランザクション

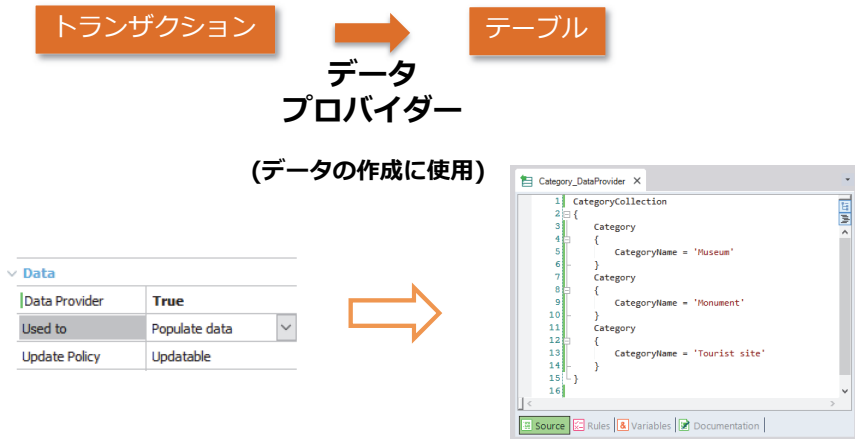
「ビュー」としてのトランザクション

GeneXus™

これまで、すべての Transaction オブジェクトで、そのデータを格納して後で取得するためのテーブルがレベルごとに作成されることを確認しました。

GeneXus

データを初期化するためのデータプロバイダーが設定されたトランザクション



既に、データプロバイダーをトランザクションと関連付けて、そのテーブルにデータを作成できることを確認しました。

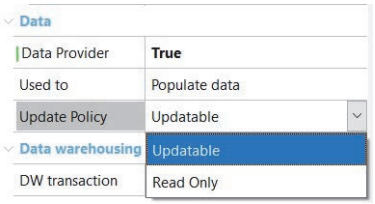
GeneXus

データを初期化するためのデータプロバイダーが設定されたトランザクション

データ
プロバイダー

(データの作成に使用)

トランザクションの用途:
- データの挿入、更新、削除
- データのナビゲート (取得)



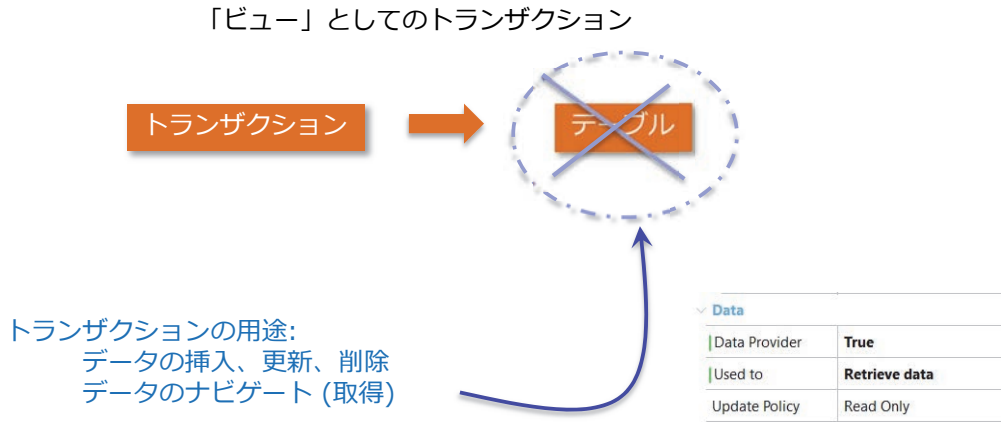
このシナリオでは、データプロバイダーは初期化のためにのみ使用されます。トランザクションは通常の動作になります。つまり、テーブルにアクセスしてデータを取得し、通常どおり、レコードの**挿入**、**更新**、**削除**を行います。

更新などの動作を可能にするため、[**Update Policy**] プロパティの値は Updatable になっています。

このトランザクションの動作を変更して、データが更新されないようにすることもできます。つまり、データが入っているテーブルが初期化されると、テーブルの変更はできなくなり、新しいデータも追加できなくなります。
この場合、[Update Policy] を既定値の Updatable から Read Only に変更する必要があります。

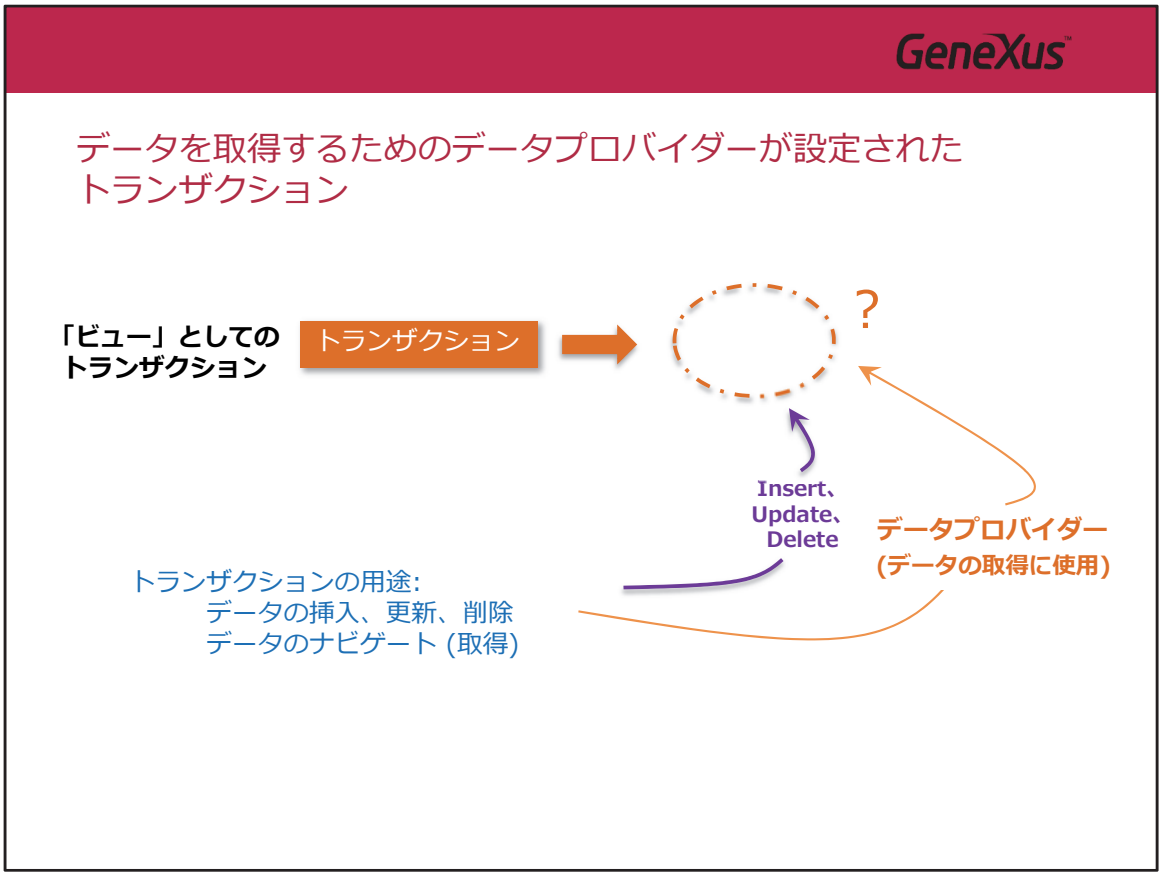
GeneXus

データを取得するためのデータプロバイダーが設定されたトランザクション



トランザクションは、関連するテーブルにデータを保存しなくても、通常どおりに使用することができます。つまり、トランザクションで、アプリケーションのデータベースにテーブルが作成されない場合があるということです。

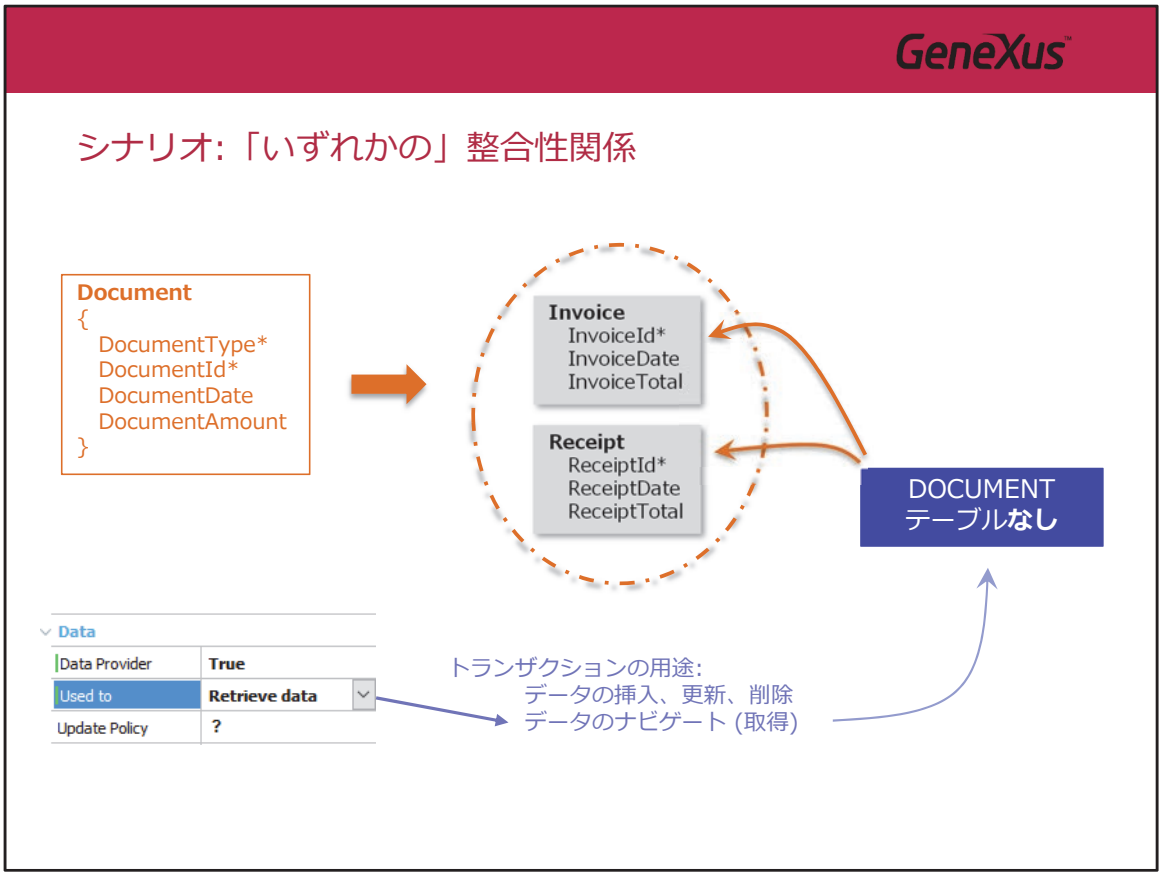
その場合は、そのトランザクションと関連付けられたデータプロバイダーを、**データの取得**に使用するように指定します。



しかし、テーブルが作成されない場合、ユーザーがデータを参照するたびに、情報をどこから取得するかを指定する必要が生じます。また、ユーザーが画面上で入力したデータを、対応するテーブルで挿入、更新、または削除する際に、どの処理を行うかを指定する必要もあります。

Insert、Update、または Delete を行うためには、これらの名前を持つ 3 つのイベントを明示的にプログラムする必要があります。

トランザクションのデータを取得するためには、それに関連付けられたデータプロバイダーのプログラムを作成する必要があります。
データを作成する場合と同様に、[Update Policy] プロパティを使用して、トランザクションを情報の取得にのみ使用するか、更新にも使用するかを指定できます。



例を見てみましょう。

標準的なトランザクションが 2 つあるとします。

- **Invoice**: チケットやツアーの購入に対して旅行代理店が顧客に発行した請求書を表します。これらの請求書には連番が振られます。
- **Receipt**: 支払いに対して旅行代理店が顧客に発行した領収書を表します。領収書にも連番が振られます。

旅行代理店の会計システムでは、これらの請求書や領収書を一般的なドキュメントとして扱い、さまざまな会計処理でこれらのドキュメントを扱えるようにする必要があります。

最初に、Document トランザクションを作成し、DocumentType と DocumentId で構成されるキーを設定します。このような複合キーが必要なのはなぜでしょうか。たとえば、請求書 1 と領収書 1 のどちらを処理しているかを判断するためです。

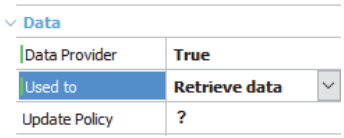
このトランザクションは、Invoice テーブルと Receipt テーブルに格納されている情報を一元化する「ビュー」のようなものです。つまり、データを格納するためのテーブルは作成せず、Invoice と Receipt に対応するテーブルからデータを取得します。

GeneXus

シナリオ:「いずれかの」整合性関係

Document
{
DocumentType*
DocumentId*
DocumentDate
DocumentAmount
}

```
DocumentCollection
{
    Document from Invoice
    {
        DocumentId = InvoiceId
        DocumentType = "Invoice"
        DocumentDate = InvoiceDate
        DocumentAmount = InvoiceTotal
    }
    Document from Receipt
    {
        DocumentId = ReceiptId
        DocumentType = "Receipt"
        DocumentDate = ReceiptDate
        DocumentAmount = ReceiptTotal
    }
}
```



この処理を行うため、[Data Provider] プロパティを True に設定し、このデータプロバイダーを使用して情報を受け取るように指定します。

これにより、GeneXus は、このデータプロバイダーがデータの取得先を指定するのに使用されるものであり、トランザクションに関連付けられたテーブルを作成する必要はないことを自動的に理解します。ここでは、対応するトランザクションとして関連付けられた INVOICE テーブルと RECEIPT テーブルからデータを取得します。

このトランザクションに関連付けられたデータプロバイダーのソースを見てみましょう。

請求書であるドキュメントをすべて取得する Document グループと、領収書であるドキュメントをすべて取得する別の Document グループがあります。

以降は、データをナビゲーションするためにこのトランザクションが実行されるたび、このデータプロバイダーが実行され、開発者とユーザーの両方に透過的な方法で、対応する情報が画面にロードされます。このトランザクションにテーブルがないことを意識する人はいません。

GeneXus

シナリオ:「いずれかの」整合性関係

ベーストランザクションとしてのダイナミックトランザクション: ドキュメントのリスト

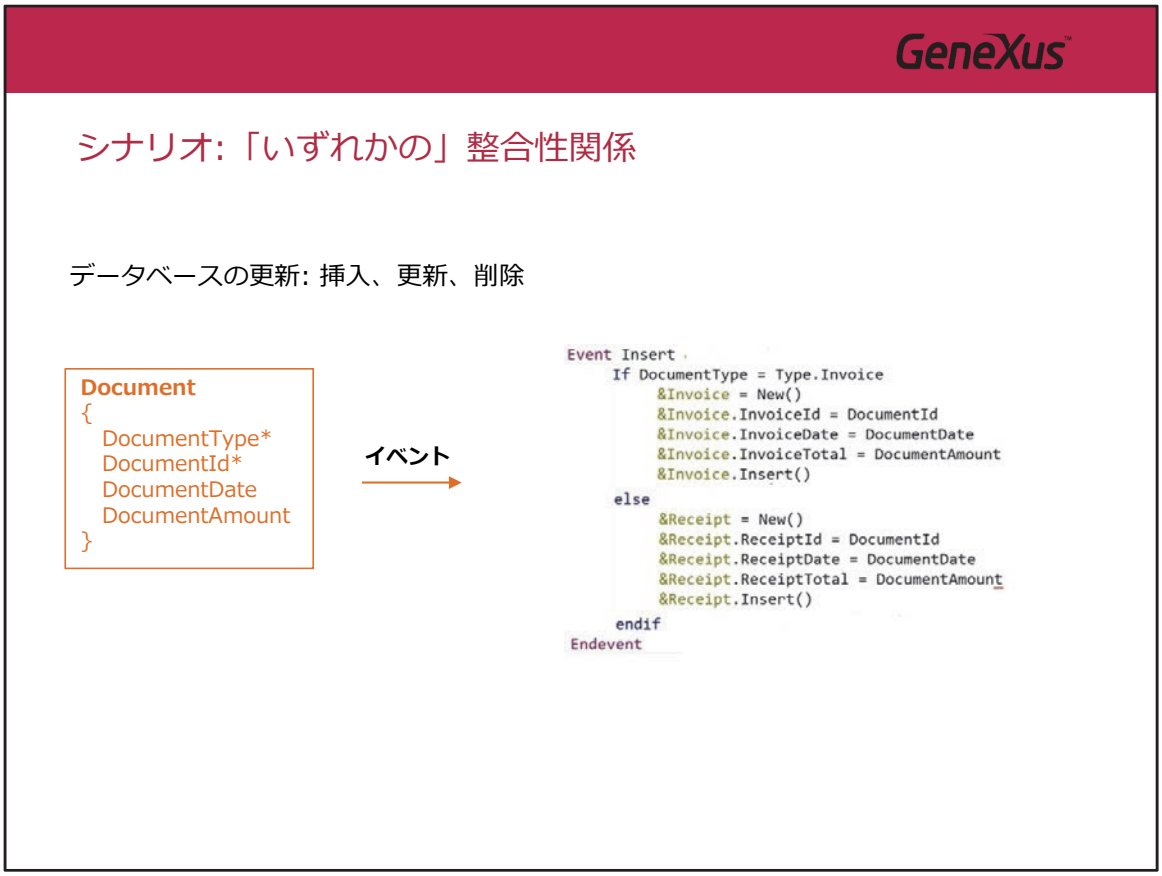
```
Document
{
   DocumentType*
   DocumentId*
   DocumentDate
   DocumentAmount
}
```

```
For each Document order (DocumentDate)
      print Documents
Endfor
```

この後は、他のトランザクションと同じようにダイナミックトランザクションを使用します。たとえば、すべてのドキュメントを日付の降順で並べ替えて出力するには、ここに示すような For each コマンドを使用してプロシージャーを作成します。

興味深いのは、この For each コマンドでは Document がベーストランザクションとして宣言されていることです。printblock で指定されている項目属性が Document に属するため、GeneXus では、この For each コマンドのベーステーブルが Document であると判断されます。

しかし、Document はデータベース内でテーブルとして存在するのではなく、データプロバイダーを介したビューになります。



トランザクションは、データの取得だけでなくデータの更新にも使用されます。このトランザクションにはテーブルが関連付けられていないので、どのようにしたらよいのでしょうか。

[Update Policy] プロパティを見てみましょう。このプロパティを [Updatable] に設定した場合、Insert、Update、Delete のイベントがトランザクションで使用可能になり、ユーザーが画面上で入力したデータをどのように挿入、更新、削除するかをプログラミングできます。この情報から、それぞれの場合にどのような処理を行うかを知っているのは開発者だけです。

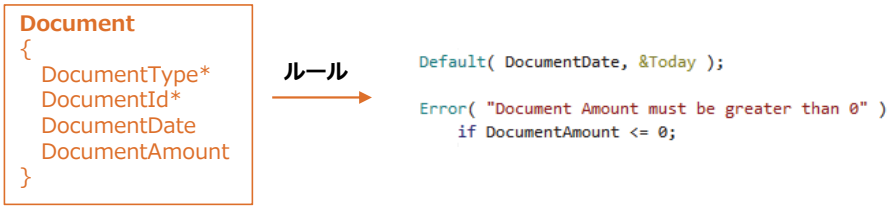
これらのアクションは、状況に応じて有効になります。

トランザクションのフォームを見てください。新しいドキュメントを挿入するために、ユーザーがこの画面の各フィールドに入力し、[実行] ボタンをクリックすると、DocumentType 項目属性に入力された値に応じて Invoice テーブルまたは Receipt テーブルに新しいレコードが挿入される必要があります。

このトランザクションの [Events] エレメントを見てみましょう。Invoice および Receipt の各ビジネス コンポーネント データ タイプに基づき、Invoice 変数と Receipt 変数を使用して、Insert イベントをプログラミングしました。

ビジネスコンポーネントの Insert メソッドを使用するのではなく Save メソッドを使用することもできます。ここでは、Commit コマンドを記述する必要はありません。この Document トランザクションの [Commit on Exit] プロパティは、既定で Yes に設定されているため、Commit は暗黙的に実行されます。

ルールとイベントのトリガー



- 評価ツリーとトリガーのタイミングは同一である
- 標準的なトランザクションと同じように指定され、トリガーされる

ダイナミックトランザクションのレベルでルールを指定した場合について考えてみましょう。

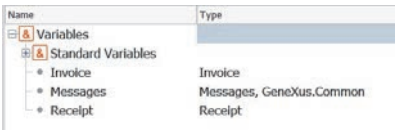
ルールはいつトリガーされるでしょうか。評価ツリーはどうなるでしょうか。

評価ツリーも、ルールのトリガーのタイミングも、通常のトランザクションの場合と同じです。

ビジネスコンポーネントでトリガーされるメッセージ

```
Document
{
   DocumentType*
   DocumentId*
   DocumentDate
   DocumentAmount
}
```

イベント

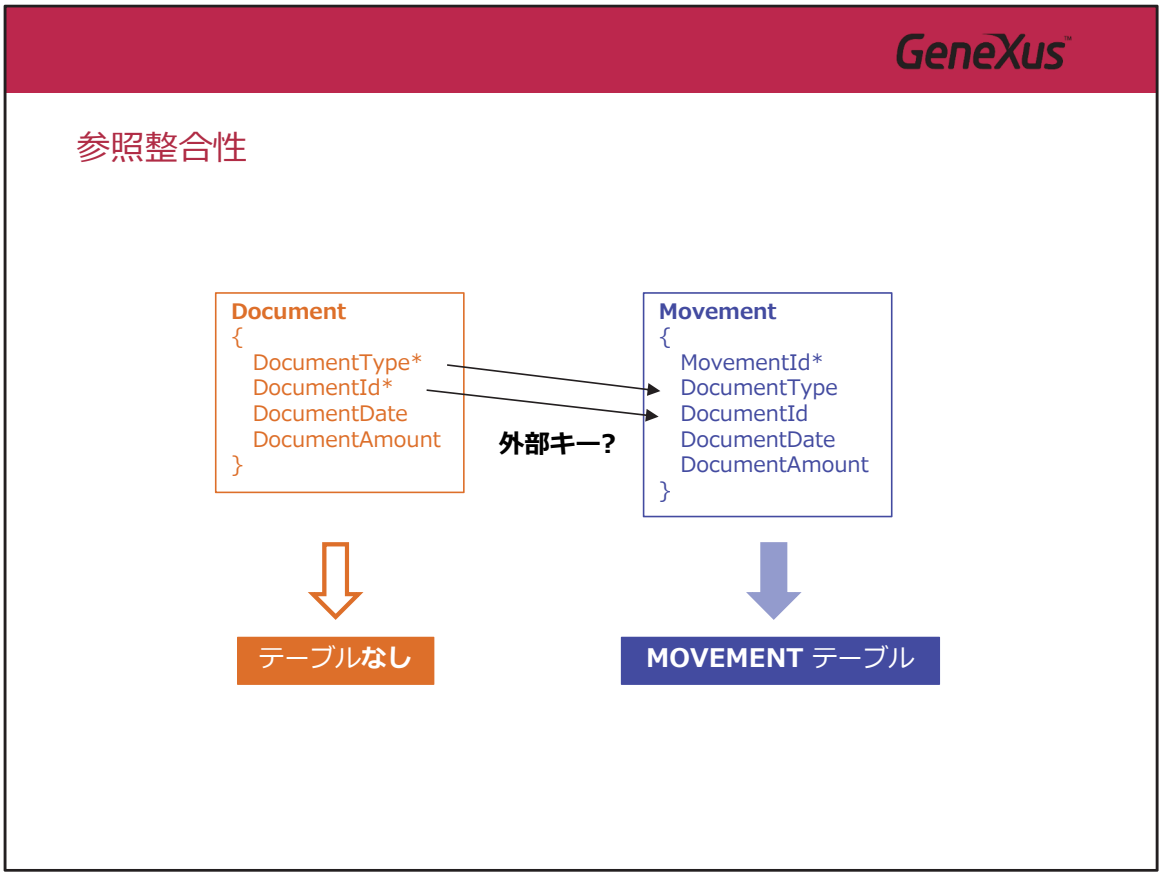



```
Event Insert(&Messages)
     If DocumentType = Type.Invoice
          &Invoice = New()
          &Invoice.InvoiceId = DocumentId
          &Invoice.InvoiceDate = DocumentDate
          &Invoice.InvoiceTotal = DocumentAmount
          &Invoice.Insert()
          &Messages = &Invoice.GetMessages()
     else
          &Receipt = New()
          &Receipt.ReceiptId = DocumentId
          &Receipt.ReceiptDate = DocumentDate
          &Receipt.ReceiptTotal = DocumentAmount
          &Receipt.Insert()
          &Messages = &Receipt.GetMessages()
     endif
Endevent
```

ビジネスコンポーネントである Invoice および Receipt の実行時にトリガーされる成功または失敗のメッセージも取得することができます。

これを行うには、トランザクションレベルの Insert、Update、および Delete イベントで、Messages データタイプ (コレクション) に基づいた Messages 変数をパラメーターとして設定します。

これらのメッセージは、透過的な方法で、ダイナミックトランザクションのフォームに表示されます。



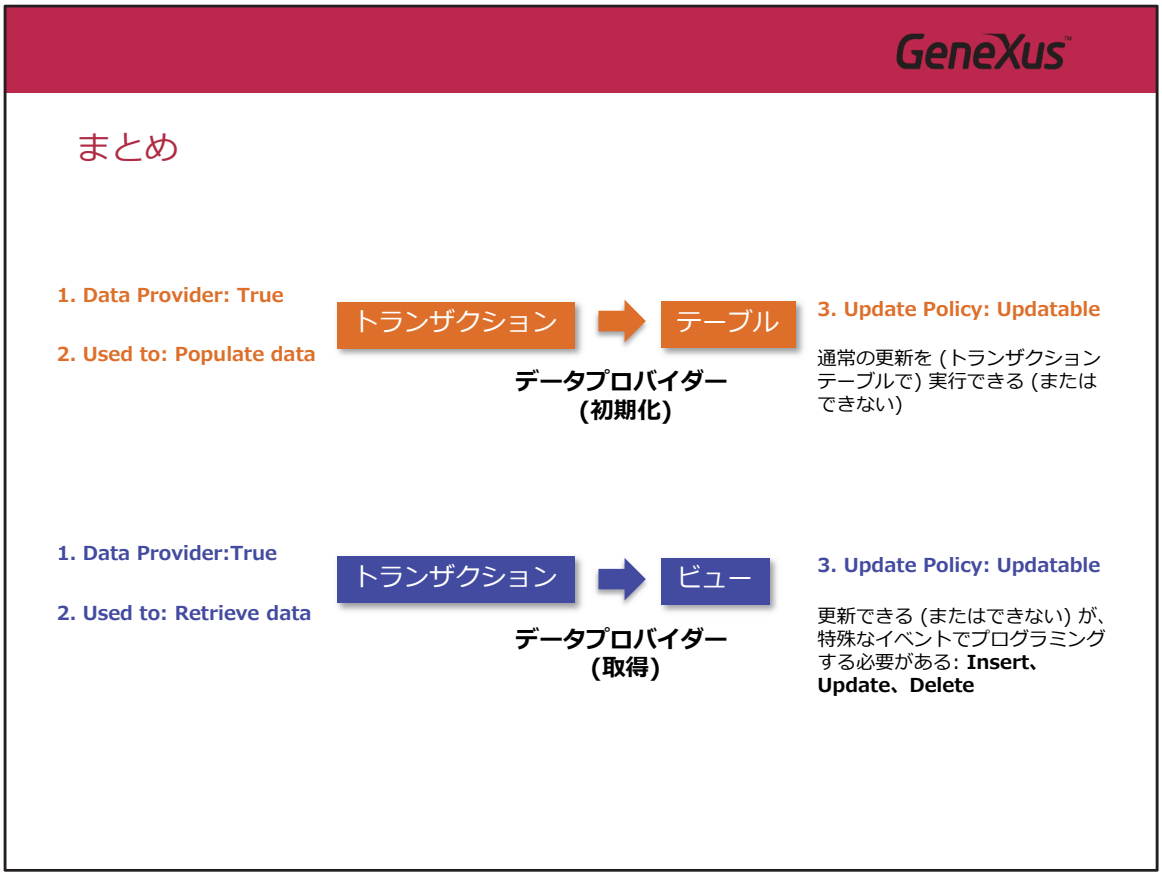
前に述べたように、この旅行代理店の会計システムでは、すべてのドキュメントを会計処理で扱えるようにする必要があります。

Document トランザクションに関連付けられた DOCUMENT テーブルは作成されないため、標準の Movement トランザクションに関連付けられた MOVEMENT テーブルでは、DocumentType と DocumentId で構成される項目属性のペアで外部キーを構成することができません。

この場合、参照整合性チェックはどうなるでしょうか。GeneXus で実行できるでしょうか。

参照整合性を確保する必要があるため、GeneXus で SQL トリガーが生成されます。したがって、DocumentType と DocumentId のペアは、Movement の「疑似」外部キーになると言えます。

つまり、Document で、関連する Movement を持つ請求書や領収書を削除することはできません。また、Document として存在していない Movement を追加することもできません。



これまでに学習した内容を確認しましょう。

トランザクションレベルで [Data Provider] プロパティの値が True の場合、GeneXus はそれを何に使用するかを確認します。

テーブルへのデータの作成に使用すると指定した場合、トランザクションによって、対応する関連テーブルが生成されます。
[Update Policy] プロパティでは、通常の方法でのレコードの更新を許可するかどうかを指定します。

トランザクションと関連付けられた [Data Provider] プロパティを True に設定し、データの受け取りに使用すると指定すると、GeneXus は、そのトランザクションには関連するテーブルがなく、そのデータプロバイダーからビューを作成するものと判断します。

次に、[Update Policy] プロパティで指定した値に基づいて、対応するイベントをプログラミングし、対応するテーブルのレコードの挿入、変更、または削除の操作を実行できるようにする必要があります。

GeneXus

ダイナミックトランザクションに関する追加情報

- その他のユースケース:

  選択
  データのグループ化
  一時的なリレーション

- ダイナミックトランザクションのその他の使用例

  http://wiki.genexus.jp/hwikibypageid.aspx?28062

この章では、ダイナミックトランザクションの 1 つの使用例を見てきましたが、このほかにも、選択、データのグループ化、一時的なリレーションなど、さまざまな使用例があります。これらについては、別のコースで取り上げます。

スライド上のリンクから詳細情報にアクセスできます。