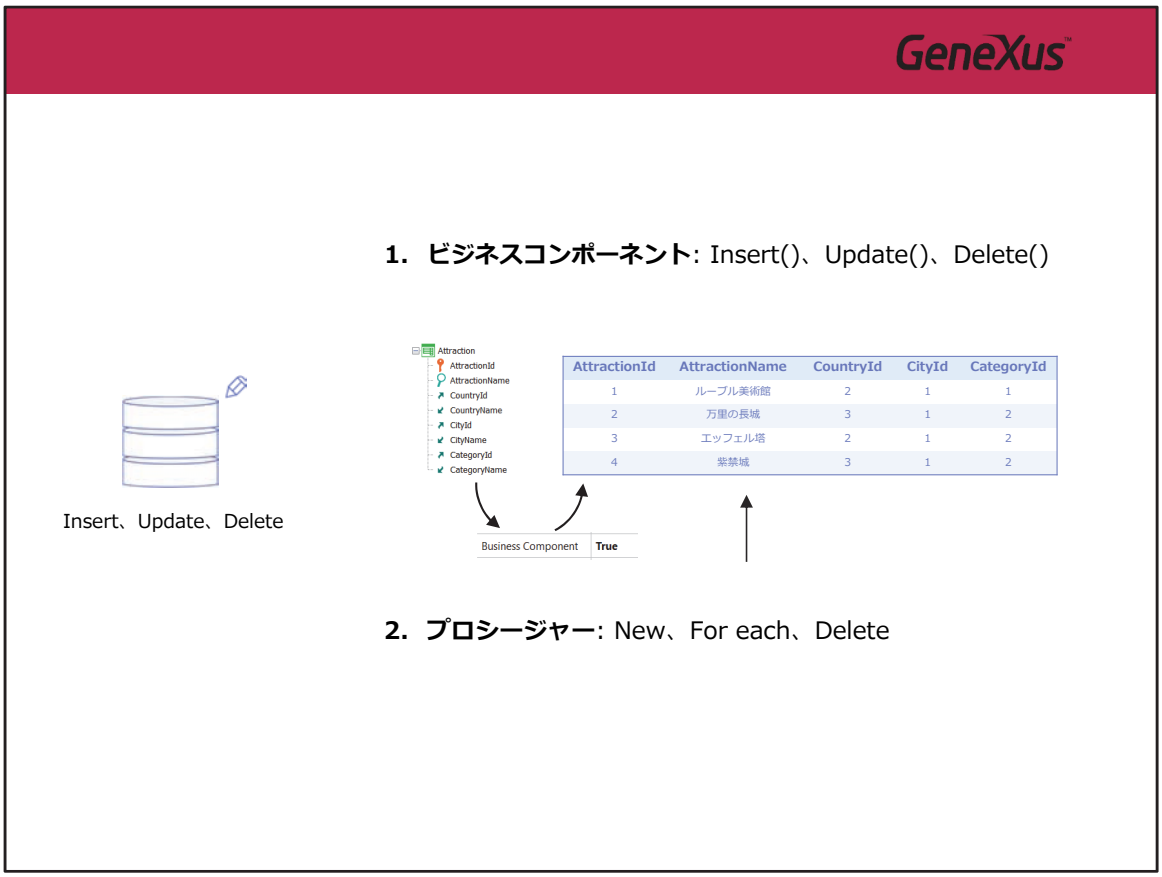
プロシージャー固有のコマンドによるデータベースの更新

データの挿入方法

GeneXus™



コードを使用してデータベース情報を更新する場合、次の 2 つの方法があります。

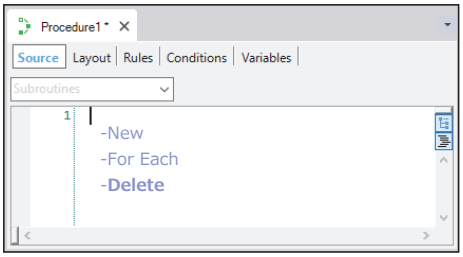
トランザクションのビジネスコンポーネントを使用して Insert、Update または Delete メソッド (Save および InsertOrUpdate) で実行するか、プロシージャー内のみで New、For each および Delete コマンドで実行します。

最初のケースについては、別の章で詳しく説明しています。ここでは 2 番目の方法について取り上げます。

GeneXus

プロシージャーのみ

Insert、Update、Delete

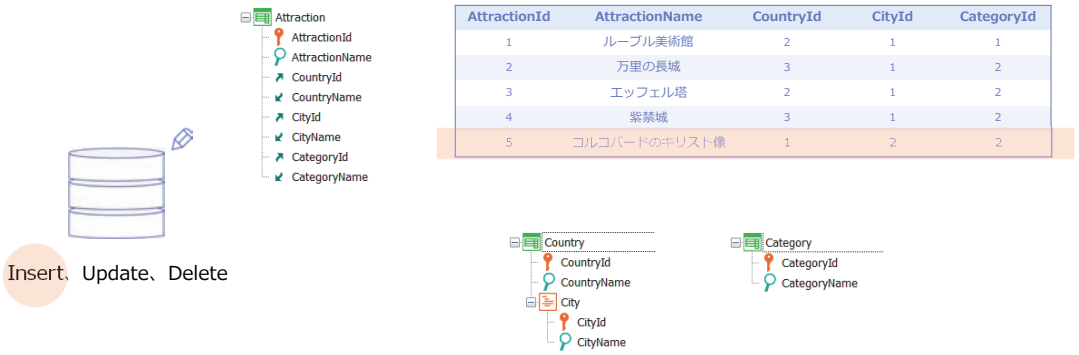


重要なのは、この 2 番目のタイプの更新を実行できるのはプロシージャーのソースのみだということです。
ここで学ぶコマンドは、パネルや Web パネルイベントなどほかの場所では無効で、有効なのはプロシージャー内だけです。

挿入

では、テーブルにレコードを挿入するコマンドから始めましょう。

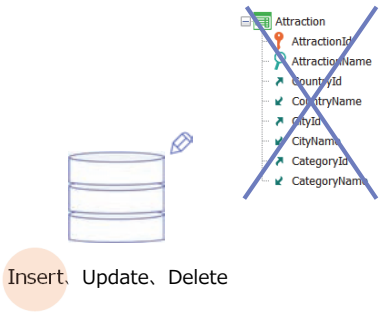
New コマンド



これは表示のように **1 つ**のテーブルに **1 つ**のレコードを挿入します。

Attraction トランザクションがあり、観光名所が記録されるとします (さらに Country トランザクションがあり、各国の情報およびその都市が記録されます。また、Category では観光名所が分類されるカテゴリが記録されます)。ここでは、プロシージャーにより、Attraction に関連付けられたテーブルにコルコバードのキリスト像 (Christ the Redeemer) を挿入します。

New コマンド



AttractionId	AttractionName	CountryId	CityId	CategoryId
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

New

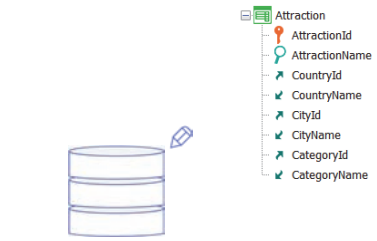
```
AttractionId     = 5
AttractionName  = "Christ the Redeemer"
CountryId = find( CountryId, CountryName = "Brazil" )
CityId  = find( CityId, CityName = "Rio de Janeiro" )
CategoryId  = find( CategoryId, CategoryName = "Monument" )
```

endnew

そのために、プロシージャーのソースで New コマンドを実行し、挿入したいレコードについて、**テーブル**の各項目属性に値を割り当てます。ここでは、テーブルで直接作業するため、テーブルを作成したトランザクションとは完全に切り離されています。つまり、トランザクションのルールやイベントなどは関係ありません。New コマンドはトランザクションを無視します。考慮するのは、New コマンドがレコードを挿入しようとするデータベーステーブルの構成のみです。

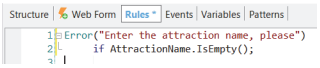
推論された項目属性やトランザクションの式はまったく関係しません。New コマンドに関しては存在しないのと同じです。関係するのはテーブルの項目属性のみです。そのため、ここではすべてに値を割り当てています。

New コマンド



AttractionId	AttractionName	CountryId	CityId	CategoryId
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		1	2	2

Insert、Update、Delete



New

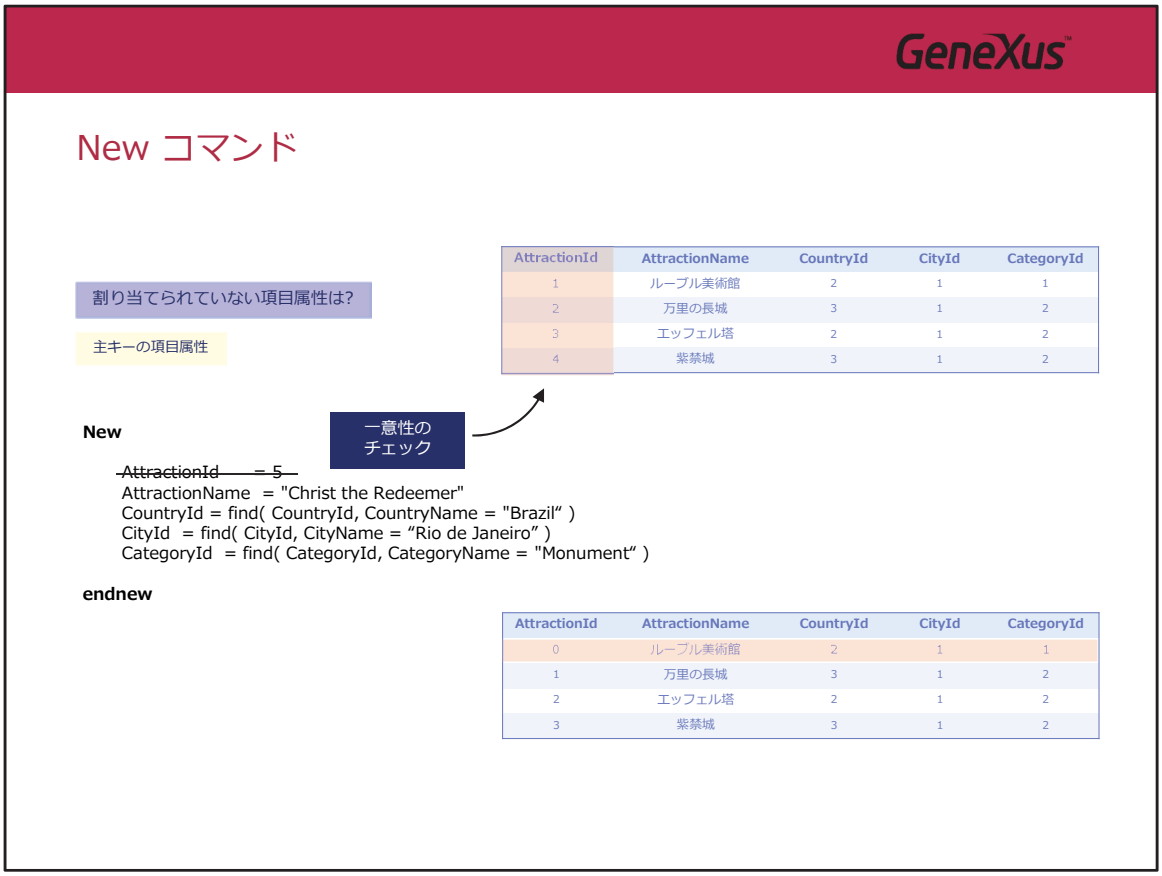
AttractionId = 5
~~AttractionName = "Christ the Redeemer"~~
CountryId = find(CountryId, CountryName = "Brazil")
CityId = find(CityId, CityName = "Rio de Janeiro")
CategoryId = find(CategoryId, CategoryName = "Monument")

endnew

割り当てられていない項目属性は?

従属項目属性

すべての項目属性に値を割り当てることは必須でしょうか。もちろん違います。項目属性に値を割り当てない場合は空だとみなされます。
したがって、AttractionName に値を割り当てない場合、観光名所 5 は名前なしで挿入されます。トランザクションにこれを回避するエラールールがあっても同じです。先ほど述べたように、この場合のトランザクションはデータベース内のテーブルを存在させるためのみに機能します。これは、ビジネスコンポーネントを介した挿入との第一の大きな違いです。



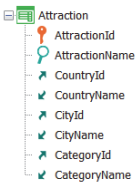
では、主キーが割り当てられていない場合はどうなるでしょうか。

空のキーレコードの挿入が試行されます。明示的に 0 を割り当てたかのようになります。
主キーが自動採番**ではない**場合、この ID 0 のレコードがテーブルに存在していなければ、挿入されます。
既に存在する場合はどうでしょうか。何も実行されません。New コマンドを実行した場合と実行しなかった場合の違いは、私たちにはわかりません。

つまり、New コマンドがレコードの一意性を管理します。重複したキーレコードが挿入されることはありません。

New コマンド

Insert、Update、Delete



主キーの項目属性

AttractionId	AttractionName	CountryId	CityId	CategoryId
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	マティス美術館	2	2	1

New

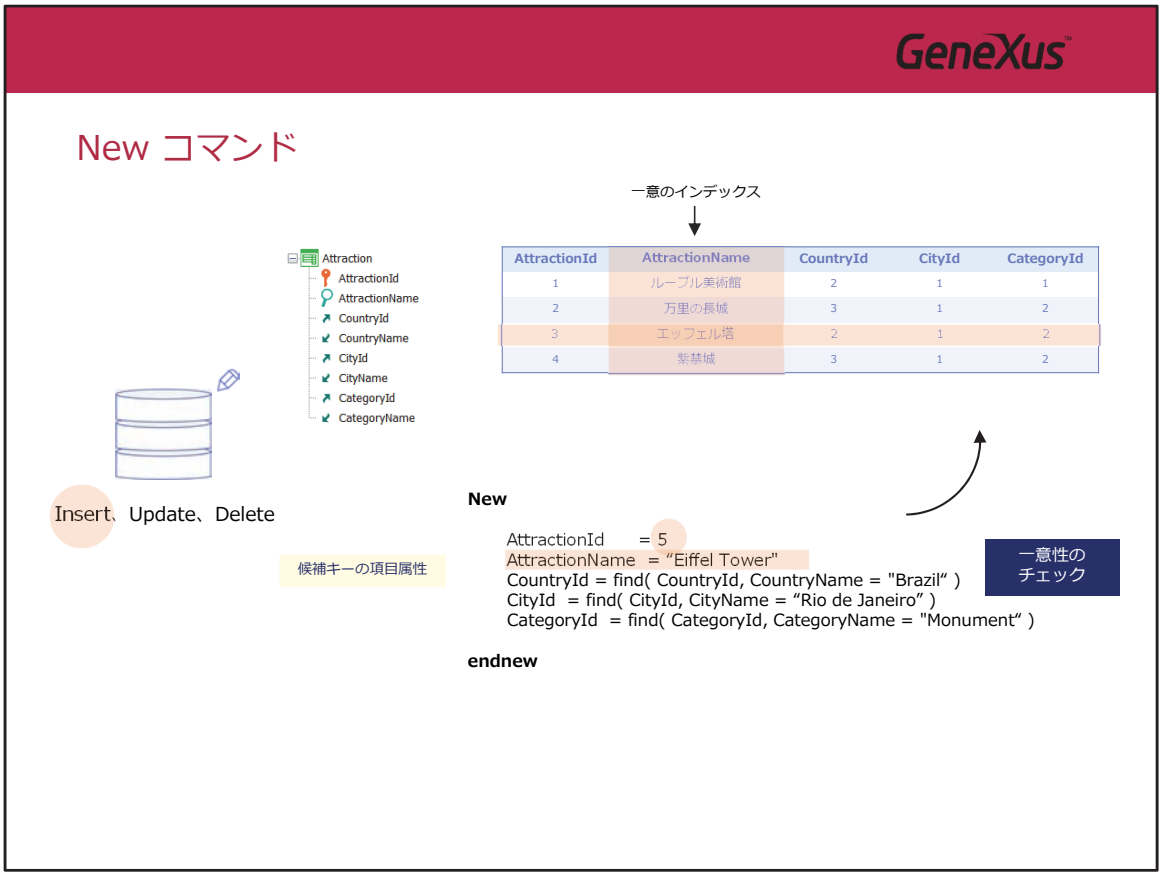
AttractionId = 5
AttractionName = "Christ the Redeemer"
CountryId = find(CountryId, CountryName = "Brazil")
CityId = find(CityId, CityName = "Rio de Janeiro")
CategoryId = find(CategoryId, CategoryName = "Monument")

一意性のチェック

endnew

New コマンドを実行しようとするときに、ID 5 の観光名所が既にテーブルに存在する場合も同様に動作します。この場合は一意性の制御により、何も実行されません。

これは主キーの場合も候補キーの場合も同様です。



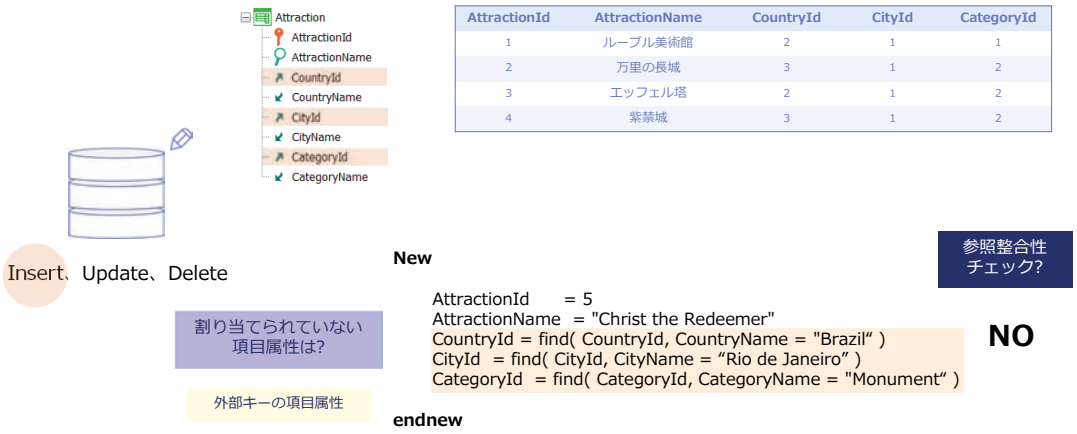
たとえば、AttractionName が候補キーだとします。つまり、この項目属性には一意のインデックスが定義されています。

ID 3 のレコードが AttractionName の値として、ここでレコード 5 として挿入するものと同じ値を持っているとします。
この場合、New コマンドを実行しても何も実行されません。その理由は、挿入したいレコードと同じ AttractionName のレコードが既に存在することを検知するからです。

New コマンド

では、主キーの値を指定しない場合はどうなるかという話題に戻りましょう。自動採番の場合は、重複した主キーが原因で失敗することはありません。常に、次の番号を持つレコードが挿入されます。どの番号が割り当てられたか知るにはどうすればいいでしょうか。項目属性の値は、New の実行後、直ちにメモリーに格納されます。これにより、たとえば、次に項目属性を使用するときに値が失われないように、変数に割り当てることができます。

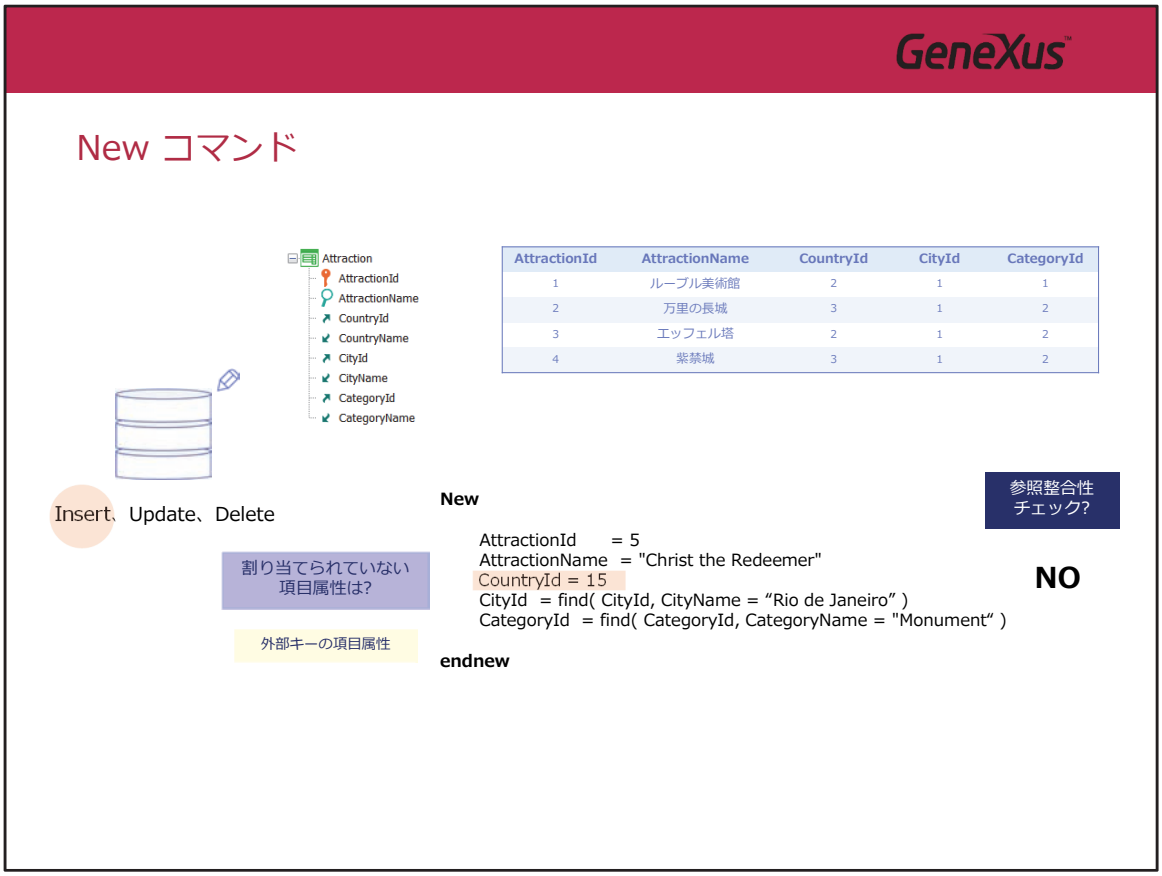
New コマンド



項目属性が外部キーの場合はどうでしょうか。CountryId、CityId、CategoryId などです。

New コマンドは参照整合性チェックを実行するのでしょうか。

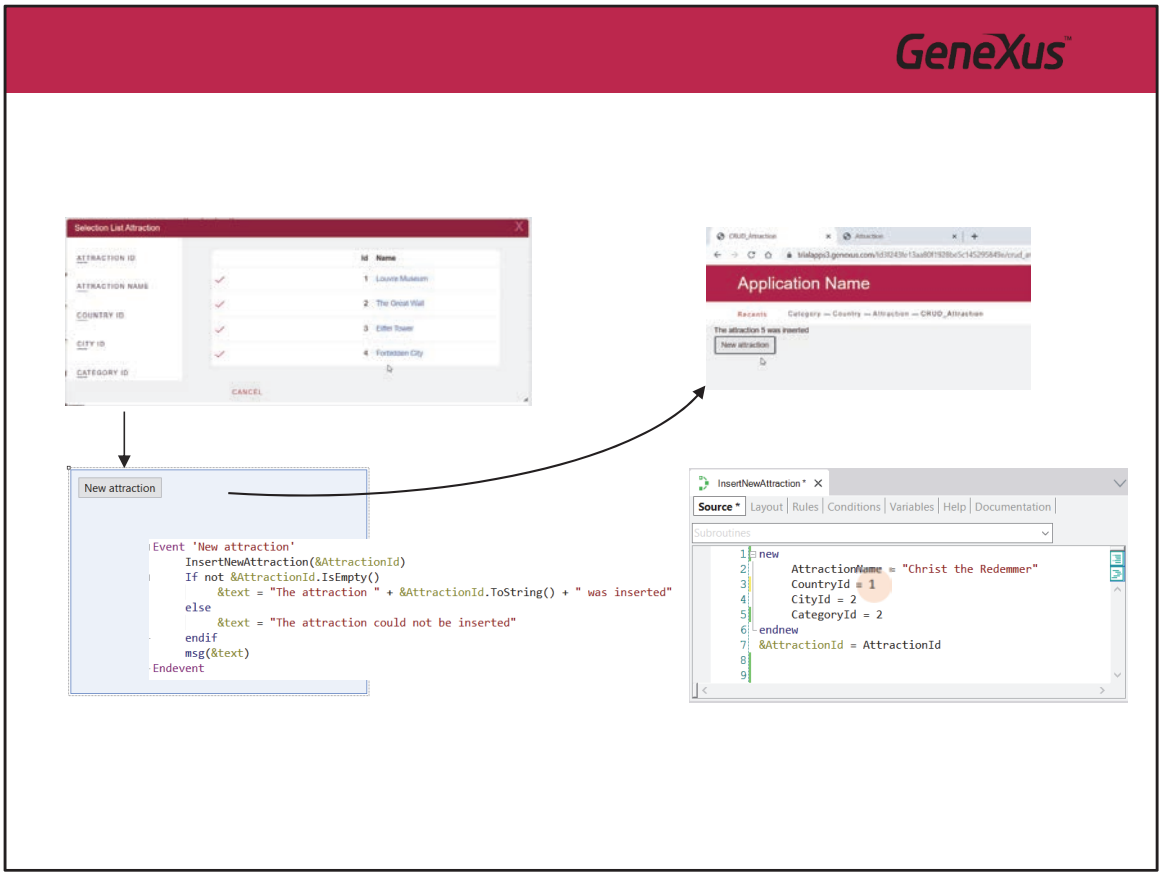
答えは「いいえ」です。なぜなら、New コマンドは、できるだけパフォーマンスが高い更新手段を得るために作られたものだからです。このようなチェックを実行すると常に操作が遅くなります。1 つのレコードを扱う場合は問題になりませんが、数百万のレコードを挿入する場合にどうなるか考えてみてください。



たとえば、新しいレコードに ID 15 の国を割り当てたい場合、New コマンドは、国が格納されたテーブルにこの値を持つ国があるかどうかをチェックしません。コマンドは問題なくレコードを挿入します。そのため、データベースの一貫性が失われたままになります。

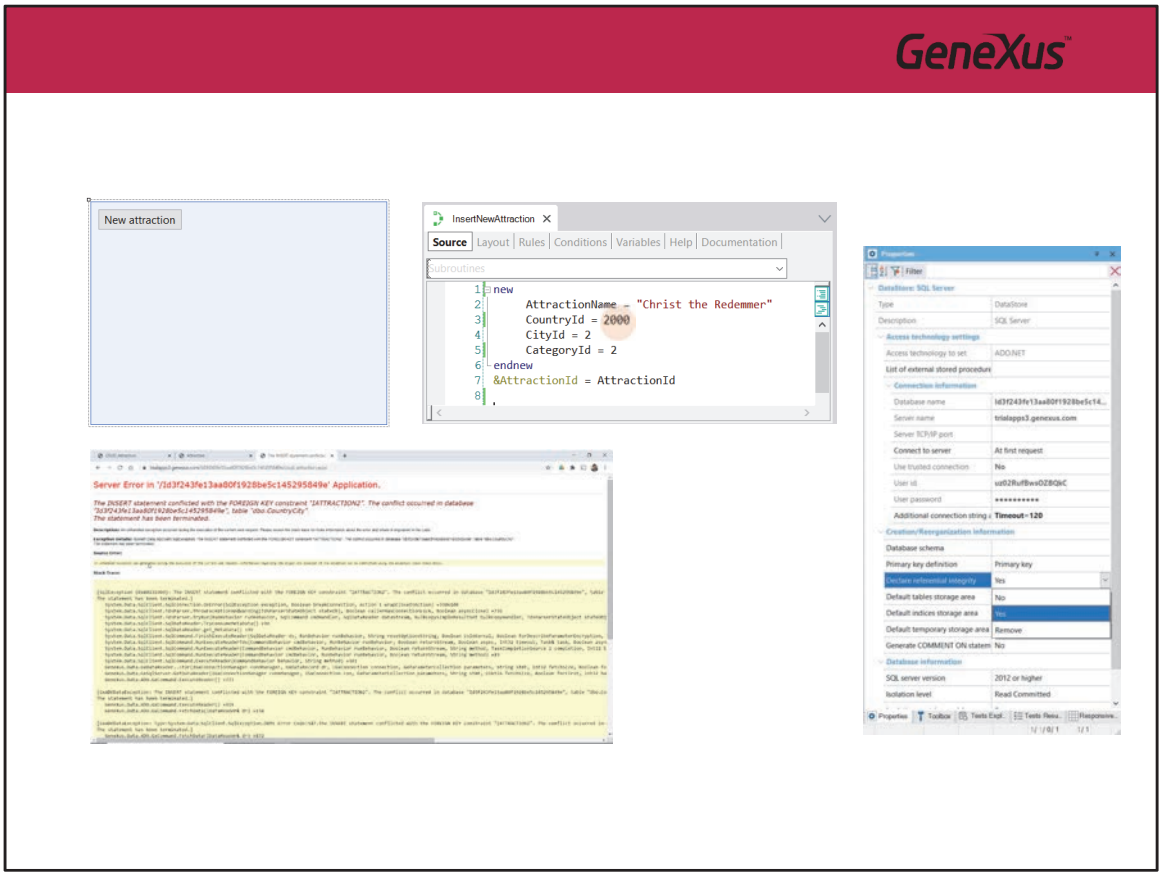
では、GeneXus で試してみましょう。

デモ



データを持つ 3 つのトランザクションがあります。特に、Attraction は 4 つのレコードを持っています。
AttractionId は自動採番です。

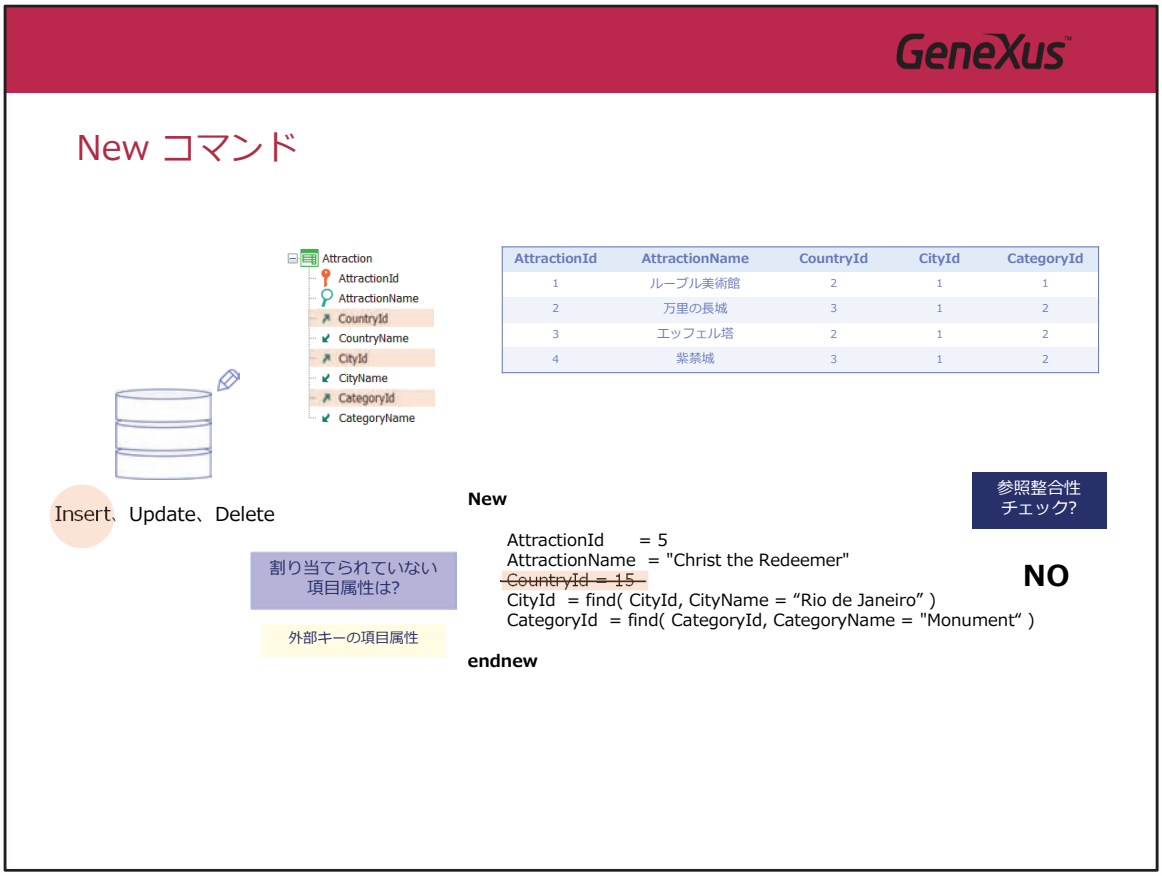
私たちが作成したこの Web パネルで、ユーザーが [New attraction] ボタンをクリックすると、プロシージャーが呼び出され、Attraction テーブルにコルコバードのキリスト像 (Christ the Redeemer) の新しいレコードを挿入しようとします。
プロシージャーは、挿入されたレコードにデータベースが割り当てた AttractionId を返します。これにより、パネルでユーザーに対して表示されるメッセージを作成します。
試してみましょう。
Attraction 5 was inserted. と表示されます。確認しましょう。実際に、観光名所 5 がテーブルに挿入されています。



新しいレコードに存在しない国を割り当てるとどうなるでしょうか。たとえば、2000 を割り当てます。これを実行します。
プログラムがクラッシュしました。New コマンドは参照整合性をチェックしないため、問題なくレコードを挿入できるはずでした。何が起こったのでしょうか。

プログラムは参照整合性をチェックしませんが、**データベース**はチェックします。New コマンドは挿入を試行しましたが、データベースがそれを受け入れず、例外が発生しました。

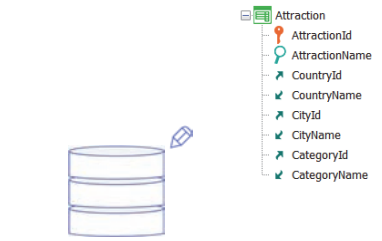
既定では、データベースは参照整合性をチェックします。この制御はプロパティを使用してオフにすることができます。
いずれにせよ、New コマンドが参照整合性チェックを行わないことははっきりわかりました。したがって、このコマンドは慎重に使用する必要があります。このようなプログラムのクラッシュは、エンドユーザーにとっては許容できないものです。



外部キーを割り当てないままにしたらどうでしょうか。

この場合も、New コマンドはまったくチェックを実行せず、レコードの挿入を試みます。データベースがチェックを実行する場合、先ほどと同様に例外が発生し、例外を捕捉して処理しない限りプログラムが停止します。
項目属性が Null を受け入れる場合、データベースは外部キーとしてこの Null を受け入れるため、失敗することはないと考えるかもしれません。しかし、データベースがそれをどのように空の値ではなく Null だと解釈するかについて、慎重に考える必要があります。これについては別の章で取り上げます。

New コマンド



AttractionId	AttractionName	CountryId	CityId	CategoryId
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

Insert、Update、Delete

コミット？

New

AttractionId = 5
AttractionName = "Christ the Redeemer"
CountryId = find(CountryId, CountryName = “Brazil”)
CityId = find(CityId, CityName = “Rio de Janeiro”)
CategoryId = find(CategoryId, CategoryName = "Monument“)

endnew

....
....
Commit

問題がなければ、New コマンドはテーブルにレコードを挿入します。Commit を実行すると何が起こるでしょうか。

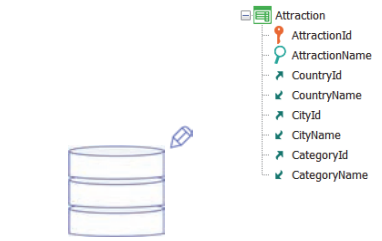
先ほど実行したプロシージャーのナビゲーションを見ると、別のプログラムによりプログラムが呼び出された可能性があること、および [Commit on Exit] プロパティが Yes に設定されているという警告が示されています。ここに示されています。このプロパティは、データベースを操作するほかのオブジェクト (Transaction オブジェクト) にもあります。ご覧のように、Transaction integrity グループの下にあります (トランザクションの整合性を定義して実現するこの重要なトピックについては、別の章で学習します)。

重要な点は、このプロパティを Yes に設定すると、(オブジェクトがデータベースに対して何らかの操作を行う限り)、オブジェクトのソースコードに Commit が自動的に追加されるということです。

New コマンド

トランザクションの場合は、ヘッダーとその行の操作の最後に追加されます。

New コマンド



Insert、Update、Delete

コミット？

AttractionId	AttractionName	CountryId	CityId	CategoryId
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

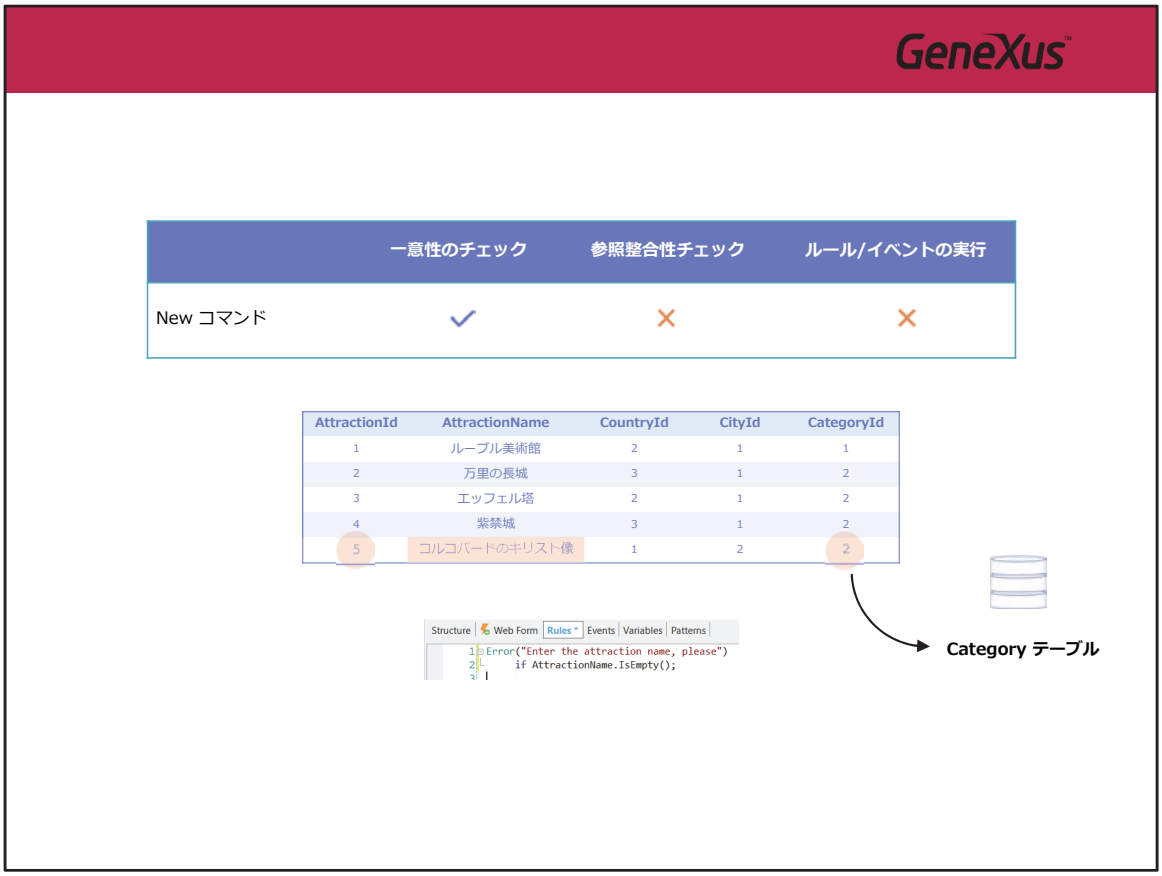
New

AttractionId = 5
AttractionName = "Christ the Redeemer"
CountryId = find(CountryId, CountryName = “Brazil”)
CityId = find(CityId, CityName = “Rio de Janeiro”)
CategoryId = find(CategoryId, CategoryName = "Monument“)

endnew

....
....
Commit

Procedure オブジェクトの場合は、ソースの最後に追加されます。そのため、指定する必要はありませんでした。プロパティが No になっている場合は、ビジネスコンポーネントの場合と同様に Commit コマンドを明示的に記述する必要があります。

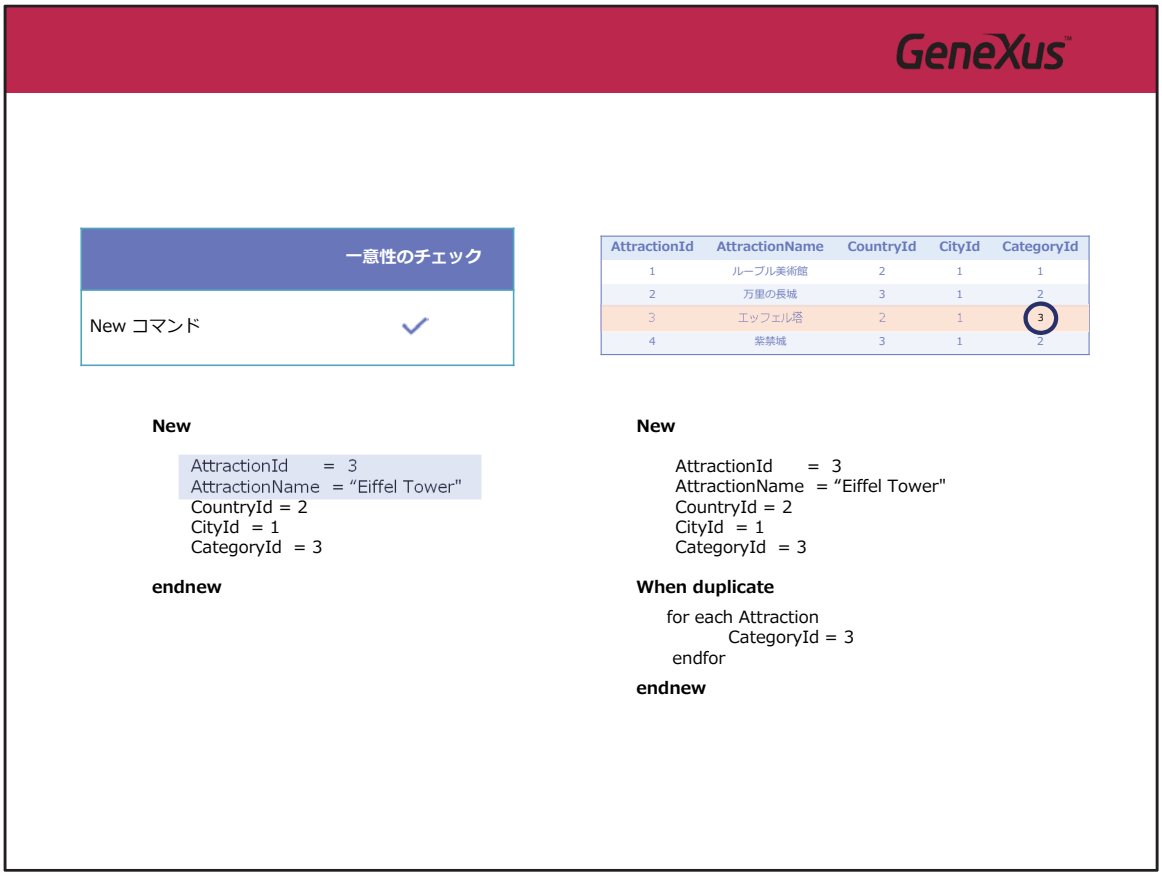


これまで学んだことをまとめます。

プロシージャーで New コマンドを使用してテーブルにレコードを挿入しようとすると、コマンドは主キーおよび候補キーによる一意性の制御を実行して、キーが重複することになるテーブルにレコードが追加されることを防ぎます。重複するキーが見つかった場合は何も実行されません。

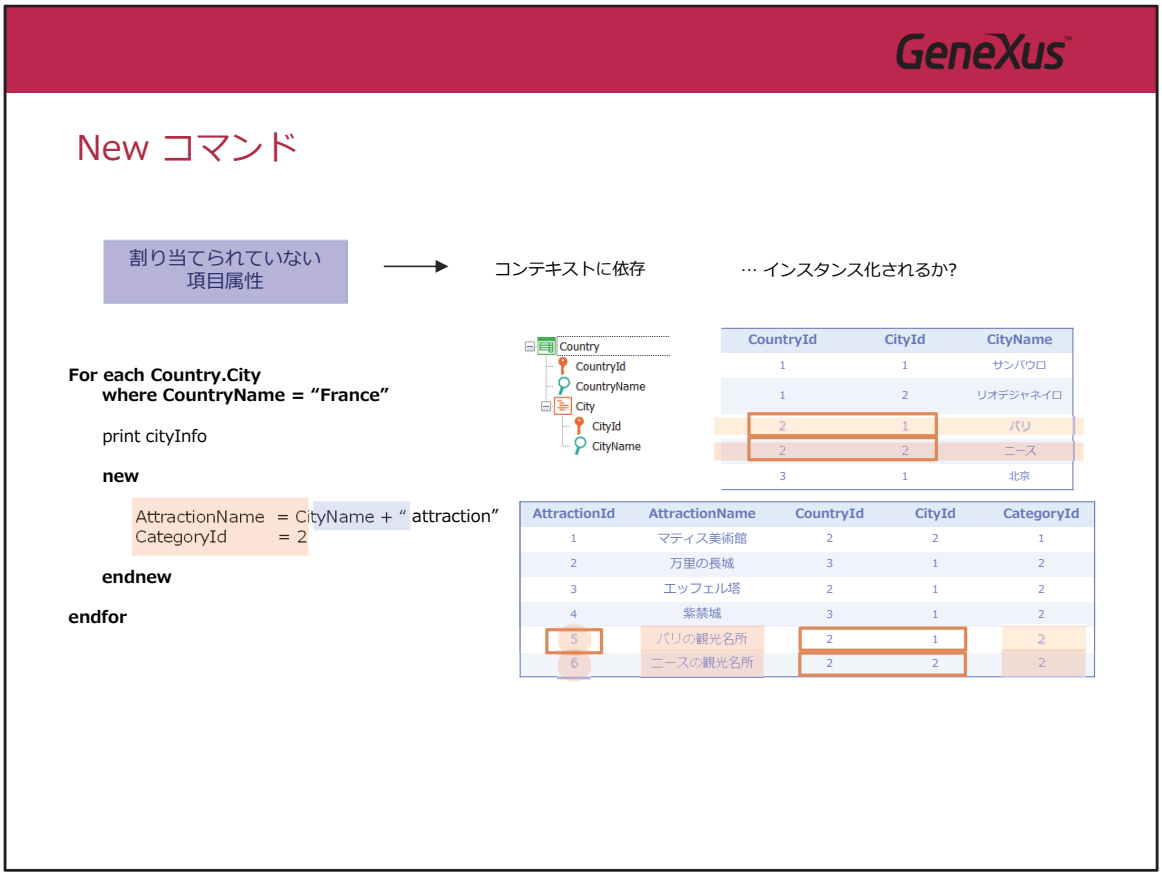
挿入されるレコードの項目属性のいずれかが外部キーであった場合、New コマンドは参照整合性チェックを実行しません。つまり、レコードを挿入するために、使用したい値を持つレコードが参照先のテーブルにあるかどうかを検索しません。
しかし、データベースで参照整合性が宣言されている場合はチェックが実行され、整合性の違反が見つかったときはプログラムがキャンセルされます。参照整合性が宣言されていない場合は、整合性の違反の有無にかかわらず、レコードの挿入が許可されます。

テーブルと関連付けられたトランザクションのルールとイベントについては、この場合、まったく関係がありません。前述のとおり、New コマンドの場合は関係があるのはテーブルのみで、テーブルの作成元は関係ありません。



ただし、レコードでキーが重複するときに、何らかのアクションを実行したい場合もあるでしょう。たとえば、挿入ではなく更新を実行するなどです。この例では、CategoryId の値を変更したいと考えています。

そのために **when duplicate** 節を使用します。この節に含まれるコードは、主キーまたは候補キーの重複が見つかった場合のみ実行されます。この場合、必要なのがレコードの更新ならば、ここに示すように、For each 節内で実行する必要があります。変更対象の項目属性に値が直接割り当てられると思うかもしれませんが、そうではなく、For each を記述して実行する必要があります。既に暗示的に指定されているため、AttractionId でフィルタする必要はありません。重複が見つかったレコードの項目属性を更新する必要があることを、New コマンドはこのようにして認識します。この場合、CategoryId 項目属性のみが更新されます。ここでは、拡張テーブルの項目属性も更新できます。



値が割り当てられて**いない** New コマンドテーブルの項目属性がどうなるかという話題に戻りましょう。空のままになると説明しましたが、実際はコンテキストに依存します。New コマンドが見つかったコンテキストでインスタンス化されない場合は空になります (つまり、値を持たない場合)。

たとえば、For each コマンドでフランスの都市を参照し、それぞれを出力するとします。さらに、この New コマンドを直ちに実行します。

まず、GeneXus は New コマンドのベーステーブルをどのように特定するのでしょうか。値が割り当てられている項目属性のみに注目することで特定します。データベースで、これらが含まれるテーブルを見つける必要があります。もちろん Attraction になります。

この CityName は関係するでしょうか。関係ありません。これが存在するとしたら、New コマンドを記述したコンテキストのためです。これがインスタンス化されることはわかっています。この CityName は For each のものとなります。要するに、目的は、各都市に都市名とカテゴリ 2 を持つ新しい観光名所を挿入することです。

では、挿入するレコードには、観光名所、国、都市のどのような ID が割り当てられるのでしょうか。この場合、空の ID は AttractionId のみです。これは For each の拡張テーブルのものではないため、コンテキストでインスタンス化されません。また、インスタンス化の別の方法として、項目属性のパラメーターで受け取ることもできますが、この場合はそうされていません。
AttractionId が自動採番の場合は、レコードが挿入されるときに、直近で使われた番号の次の番号がデータベースにより与えられます。
ただし、CountryId および CityId はコンテキストでインスタンス化されます。したがって、このコンテキストの値が割り当てられます。この場合は、For each 内で位置付けられている国と都市です。

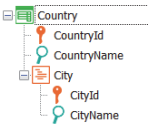
そして、For each 内の次の都市も同じように処理されます。

New コマンド

割り当てられていない項目属性 → コンテキストに依存 … インスタンス化されるか?

```
For each Country.City
    where CountryName = "France"

    print cityInfo

    new

        AttractionName = CityName + " attraction"
        CategoryId = 2

    endnew

endfor
```

警告
ベーステーブル: CATEGORY!!!!



CountryId	CityId	CityName
1	1	サンパウロ
1	2	リオデジャネイロ
2	1	パリ
2	2	ニース
3	1	北京

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	マティス美術館	2	2	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		2	1	2
6		2	2	2

同じ 2 つのレコードを Attraction に挿入したいが、AttractionName は空にしたい場合はどうなるでしょうか。

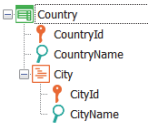
AttractionName に値を割り当てないだけで、実装できそうに思えます。しかし、ベーステーブル (New の対象) を決定するために GeneXus が使用する項目属性を見ると、CategoryId しかありません。したがって、ベーステーブルとして Attraction ではなく Category が選択されます。

Attraction テーブルを選択させるにはどうすればいいでしょうか。1 つの方法として、AttractionName に明示的に空の値を割り当てる方法が考えられます。このように名前を指定することにより、この項目属性がベーステーブルの決定に関与し、ベーステーブルを Category ではなく Attraction にすることができます。

New コマンド

割り当てられていない項目属性 → コンテキストに依存 … インスタンス化されるか?

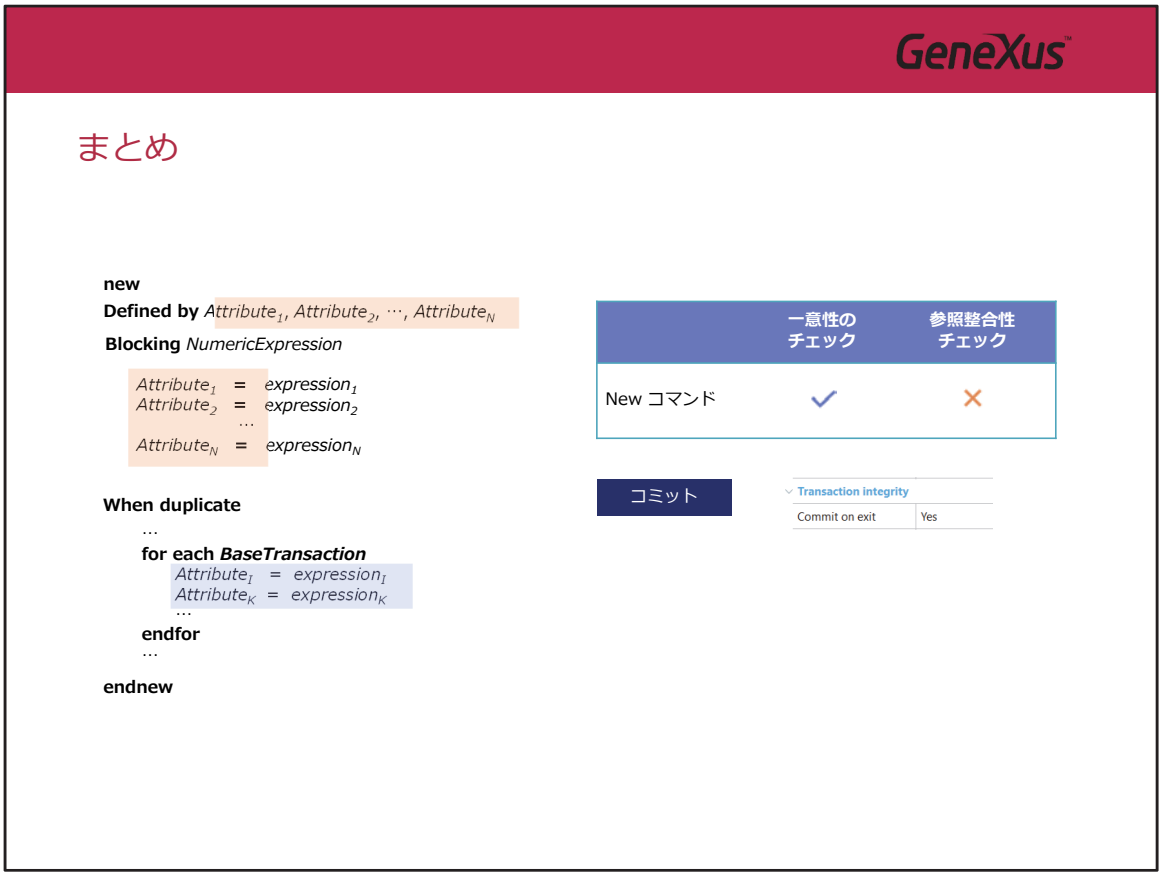
```
For each Country.City
    where CountryName = “France”

    print cityInfo

    new
        Defined by AttractionName
        CategoryId      = 2
    endnew
endfor
```



CountryId	CityId	CityName
1	1	サンパウロ
1	2	リオデジャネイロ
2	1	パリ
2	2	ニース
3	1	北京

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	マティス美術館	2	2	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		2	1	2
6		2	2	2

もう 1 つの方法は Defined by 節を使用して、ベーステーブルの決定で考慮される項目属性を追加することです。



この章では、New コマンドを使用してテーブルにレコードを挿入する方法について確認しました。テーブルは、割り当てられた項目属性により決定されます。Defined By 節を追加すると、そこに記述された項目属性も決定に関与します。ここでは、拡張テーブルという概念は意味がありません。

一方、先ほど、New コマンドが実行するプログラム制御は一意性の制御のみだと確認しました。挿入しようとするキーを持つレコードが見つかった場合、New コマンドは何も実行しません。ただし、When Duplicate 節をプログラムした場合は異なります。

既に述べたように、特に、For each を含めて、重複が見つかったレコードの項目属性を更新できます。では、すべての項目属性を更新できるでしょうか。たとえば、主キーを更新できるでしょうか。答えは「できない」です。ただし、拡張テーブルの項目属性は更新できます。

最後になりますが、データベースでレコードをコミットするには、確実に Commit コマンドを実行する必要があります。プロシージャーでは、既定で暗示的な Commit が最後に配置されます。ただし、ソース内で都合の良い場所に明示的に Commit を記述できます。

ここには示されていませんが、オプションで Blocking 節を指定することもできます。これにより、繰り返し構造内に New コマンドがある限り、レコードごとではなくブロック単位でデータベースに挿入できます。これは、もちろん、挿入のバッチが非常に大きい場合に効率化のために使用できます。

更新

次の資料では、プロシージャーを使用する更新方法について見てみましょう。