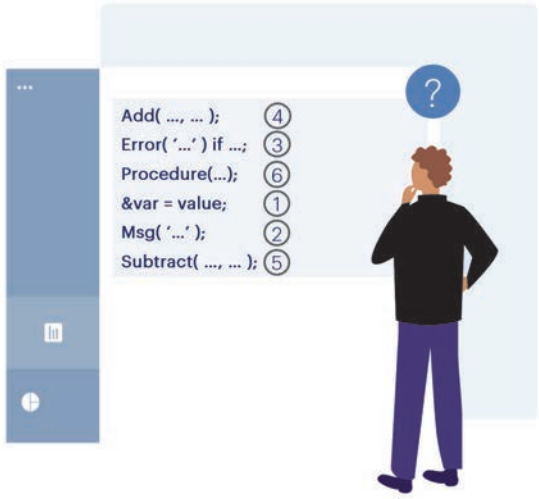
ルールおよび式のトリガーの評価ツリー

GeneXus™

GeneXus

トランザクション内のルール

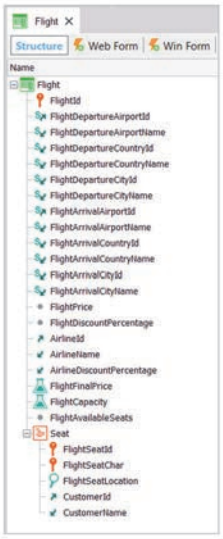


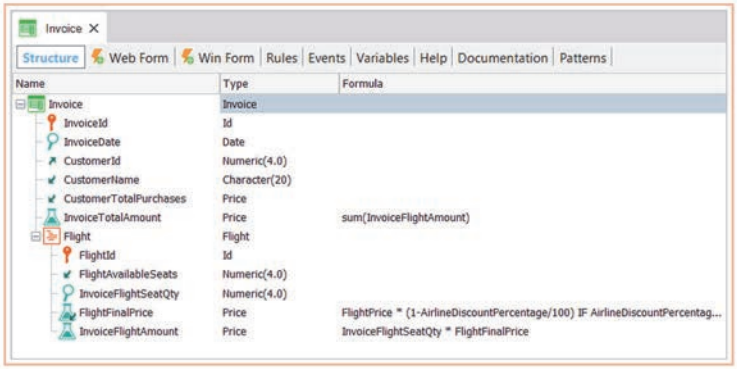
トランザクション内のルールは任意の順序で宣言でき、各ルールがトリガーされるタイミングは GeneXus が判断します。これは、開発者にとっては分かりにくい場合があります。自分で制御できないと感じるためです。

しかし、これは実際にはメリットになります。開発者はロジックの宣言のみに集中でき、各ルールがトリガーされる状況とタイミングは GeneXus が自動で推論するためです。

GeneXus

例

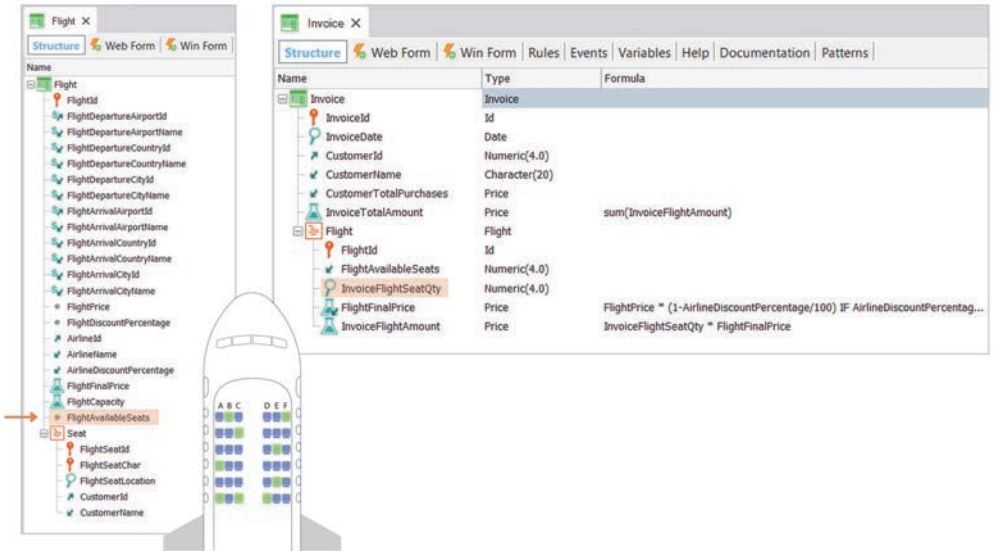




このトピックについて、Invoice トランザクションを使用して説明します。このトランザクションには第 2 レベル (Flight) があり、これは請求書に含まれるフライトを表します。

このトランザクションと、トランザクション内のルールのトリガーに焦点を当てます。

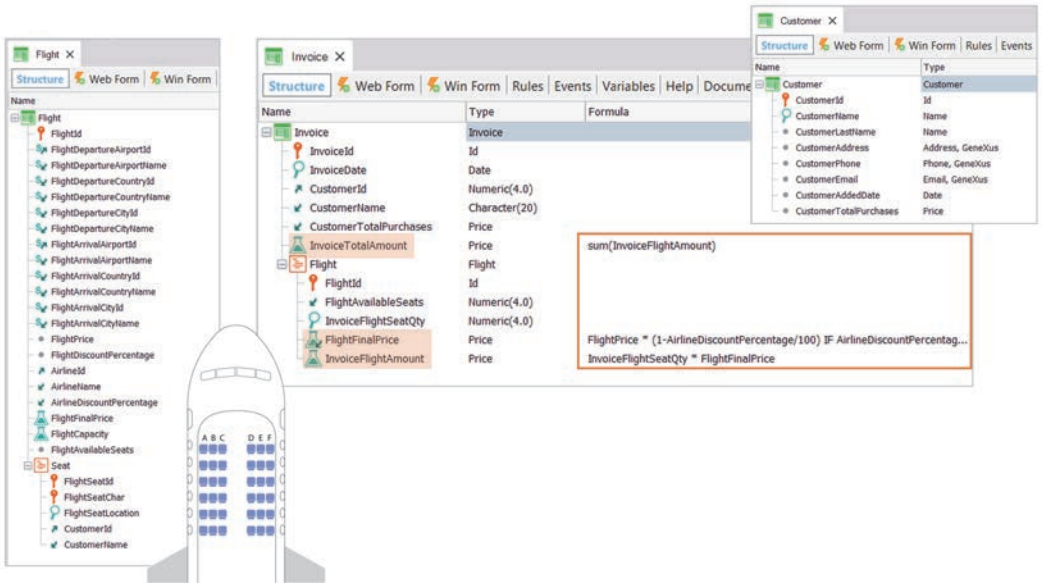
例



Flight トランザクションには **FlightAvailableSeats** 項目属性が追加されています。これは各フライトの利用可能な座席数を記録するために使用されます。顧客がフライトの座席を購入し、請求書が発行されるたびに、利用可能な座席数が減っていきます。

このレベルにこの項目属性が追加されています。これは推論された項目属性になります。

例

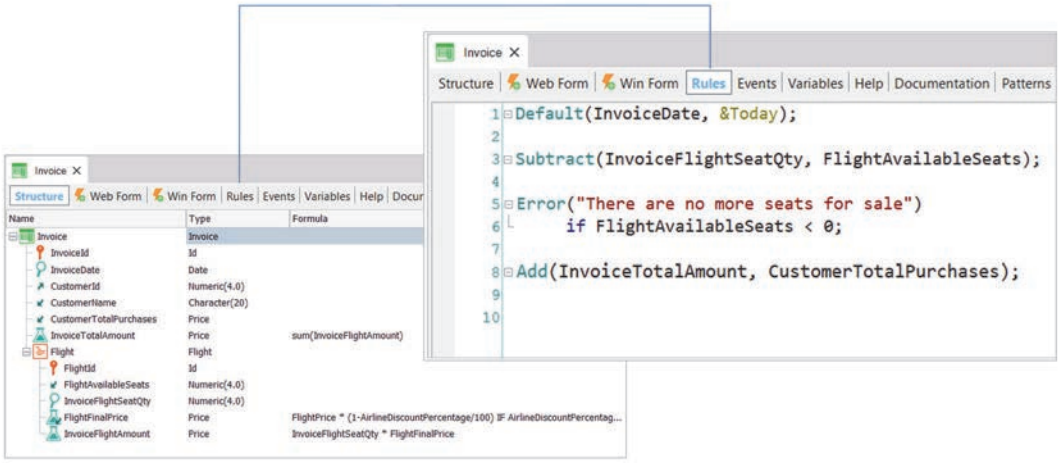


Customer トランザクションには **CustomerTotalPurchases** 項目属性も追加されています。この項目属性には、フライトチケット購入時の顧客の合計支出額を記録します。これも推論された項目属性としてトランザクションに追加されています。

Invoice 構造の InvoiceTotalAmount、FlightFinalPrice、InvoiceFlightAmount の各項目属性は、式として定義されています。

GeneXus

Invoice のルール



Invoice トランザクションには、動作を指定する次の各ルールが定義されています:

Default ルールでは、請求書の日付を表す項目属性を当日の日付で初期化します。**Subtract** ルールでは、利用可能な座席数から、請求書上の購入された座席数を引きます。これにより Flight テーブルの項目属性が減算されます。ここでは FlightAvailableSeats が推論されます。**Error** ルールでは、利用可能な座席がフライトにない場合にエラーメッセージを表示します。**Add** ルールでは、顧客の合計購入金額に、請求書の合計金額を加えます。この合計購入金額の項目属性も、Customer の拡張テーブルに含まれます。

評価ツリー

```
(R) Default( InvoiceDate, &Today );
(R) Add( InvoiceTotalAmount, CustomerTotalPurchases );
(F) InvoiceTotalAmount =  Sum( InvoiceFlightAmount )
(F) InvoiceFlightAmount = FlightFinalPrice * InvoiceFlightSeatQty
(F) FlightFinalPrice = FlightPrice * (1 - AirlineDiscountPercentage...)
(R) Subtract( InvoiceFlightSeatQty, FlightAvailableSeats );
(R) Error( "There are no more seats for sale" ) if FlightAvailableSeats < 0;
```

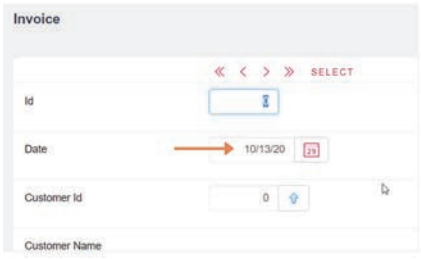
つまり、ここに示すルールおよび式がすべて Invoice トランザクションに定義されています。

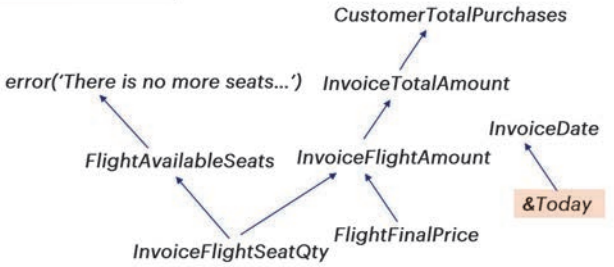
では、トリガーされる順序、トリガーされるタイミング、トリガーされないタイミングを、GeneXus はどのように認識するのでしょうか。

当然、まずは画面上の項目属性の順序に対応した自然な順序があります (上から下、左から右)。

評価ツリー

```
(R) Default( InvoiceDate, &Today );
(R) Add( InvoiceTotalAmount, CustomerTotalPurchases );
(F) InvoiceTotalAmount =  Sum( InvoiceFlightAmount )
(F) InvoiceFlightAmount = FlightFinalPrice * InvoiceFlightSeatQty
(F) FlightFinalPrice = FlightPrice * (1 - AirlineDiscountPercentage...)
(R) Subtract( InvoiceFlightSeatQty, FlightAvailableSeats );
(R) Error( "There are no more seats for sale" ) if FlightAvailableSeats < 0;
```



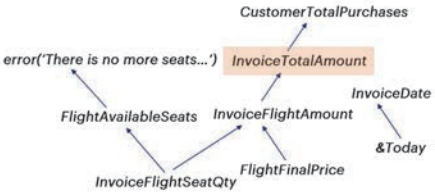


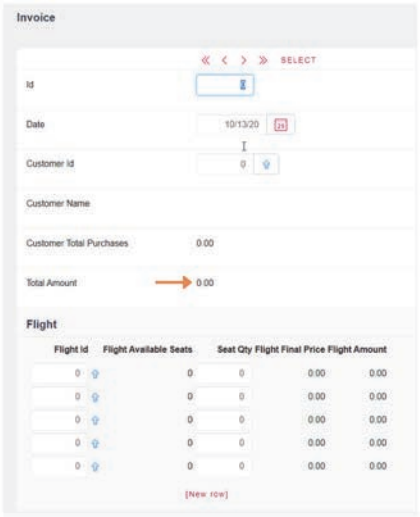
ルールまたは式は、それらに必要な情報がアクセスされるとすぐにトリガーされます。たとえば、Default ルールでは、&Today 変数の値と、Insert モードかどうかの情報のみが必要です。Insert モードで画面を開くと、対象項目に移動する前に既にフィールドに値が表示されているのはそのためです (InvoiceId のフィールドを選択している時点で日時が表示されています)。

GeneXus

評価ツリー

```
(R) Default( InvoiceDate, &Today );
(R) Add( InvoiceTotalAmount, CustomerTotalPurchases );
(F) InvoiceTotalAmount =  Sum( InvoiceFlightAmount )
(F) InvoiceFlightAmount = FlightFinalPrice * InvoiceFlightSeatQty
(F) FlightFinalPrice = FlightPrice * (1 - AirlineDiscountPercentage...)
(R) Subtract( InvoiceFlightSeatQty, FlightAvailableSeats );
(R) Error( "There are no more seats for sale" ) if FlightAvailableSeats < 0;
```

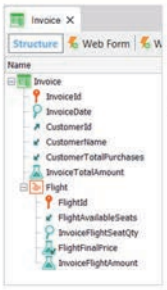


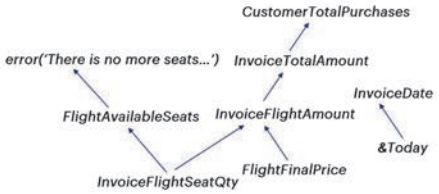


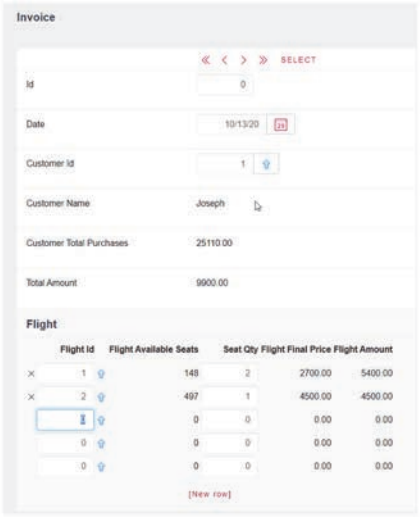
次の第 1 レベルの式では何が起こるかを考えてみましょう: InvoiceTotalAmount。これは第 2 レベルの項目属性の合計です。Sum 式に必要なのは InvoiceFlightAmount 項目属性のみであるため、最初の行が入力されるよりも前にヘッダー用にトリガーされ、値は 0 になります。

評価ツリー

```
(R) Default( InvoiceDate, &Today );
(R) Add( InvoiceTotalAmount, CustomerTotalPurchases );
(F) InvoiceTotalAmount =  Sum( InvoiceFlightAmount )
(F) InvoiceFlightAmount = FlightFinalPrice * InvoiceFlightSeatQty
(F) FlightFinalPrice = FlightPrice * (1 - AirlineDiscountPercentage...)
(R) Subtract( InvoiceFlightSeatQty, FlightAvailableSeats );
(R) Error( "There are no more seats for sale" ) if FlightAvailableSeats < 0;
```



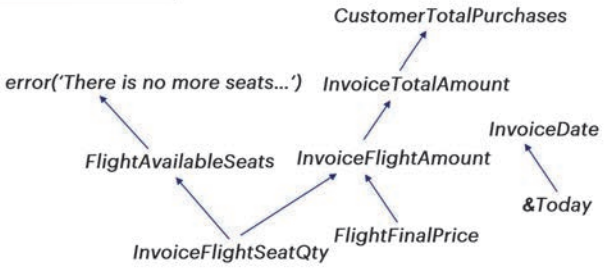




その後、各行に入力すると、行ごとに再度トリガーされます。ここで、Update モードでトランザクションにアクセスし、たとえば InvoiceFlightAmount の値にまったく影響しない行内の項目を変更したらどうなるでしょうか。(この例では、変更しても InvoiceFlightAmount の値に影響しない項目属性はありません。編集できる項目属性が行 ID および InvoiceFlightSeatQty だけであるためです。行 ID は主キーの一部であるため変更できません。InvoiceFlightSeatQty を変更すると InvoiceFlightAmount の値に影響します。そこで、項目属性を 1 つ想定します。たとえば、乗客が糖尿病であるかどうかを示す項目属性があるとします。この行が当初は「Yes」で、それを「No」に変更するとします。)その場合は明らかにヘッダーの式は再計算されません。

評価ツリー

```
(R) Default( InvoiceDate, &Today );
(R) Add( InvoiceTotalAmount, CustomerTotalPurchases );
(F) InvoiceTotalAmount =  Sum( InvoiceFlightAmount )
(F) InvoiceFlightAmount = FlightFinalPrice * InvoiceFlightSeatQty
(F) FlightFinalPrice = FlightPrice * (1 - AirlineDiscountPercentage...)
(R) Subtract( InvoiceFlightSeatQty, FlightAvailableSeats );
(R) Error( "There are no more seats for sale" ) if FlightAvailableSeats < 0;
```

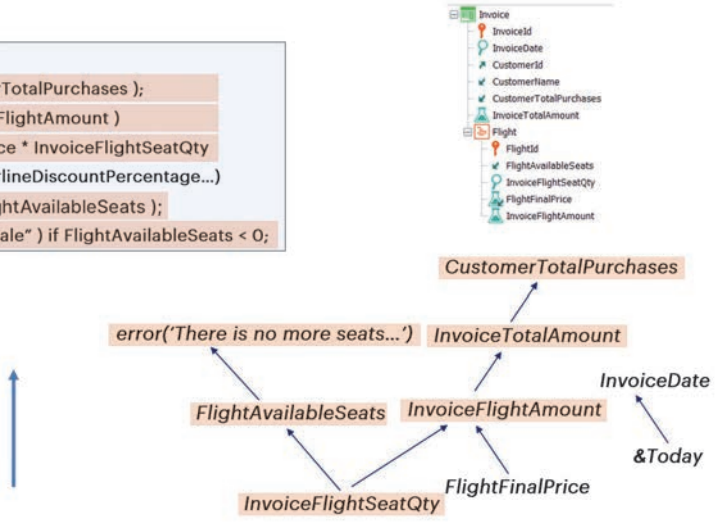


GeneXus にとってもこれは明らかです。つまり、GeneXus は、画面上のコントロール、ルール、式から想定される項目同士の既存の依存関係を内部的に抽出して、依存関係のツリー (評価ツリー) を構築します。画面上の項目属性に変更が加えられた場合に、この評価ツリーによって、再度トリガーする必要があるルールおよび式が判断されます。この例では、評価ツリーは図に示されているようになります:

GeneXus

評価ツリー

```
(R) Default( InvoiceDate, &Today );
(R) Add( InvoiceTotalAmount, CustomerTotalPurchases );
(F) InvoiceTotalAmount =  Sum( InvoiceFlightAmount )
(F) InvoiceFlightAmount = FlightFinalPrice * InvoiceFlightSeatQty
(F) FlightFinalPrice = FlightPrice * (1 - AirlineDiscountPercentage...)
(R) Subtract( InvoiceFlightSeatQty, FlightAvailableSeats );
(R) Error( "There are no more seats for sale" ) if FlightAvailableSeats < 0;
```

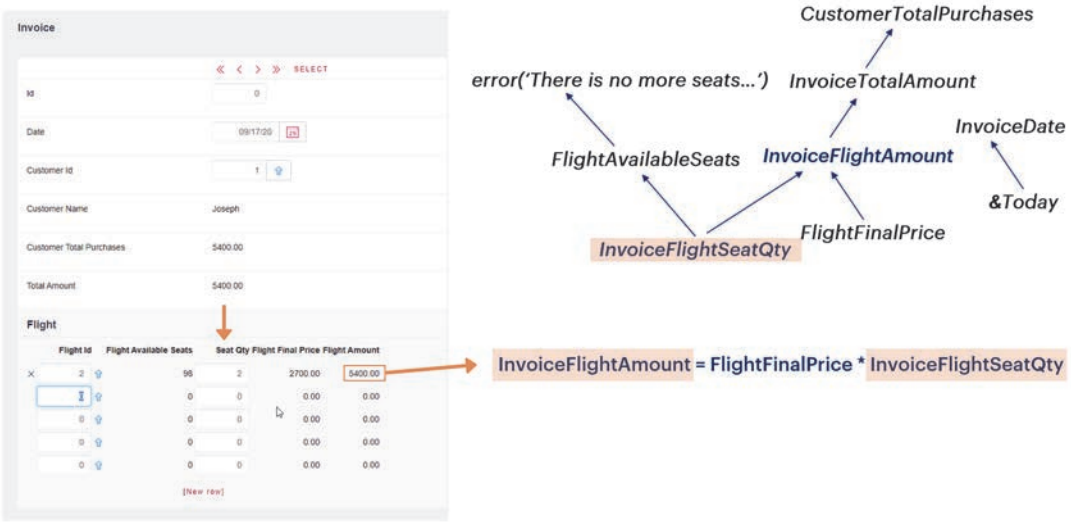


たとえば、InvoiceFlightSeatQty 項目属性に関して何が起こるでしょうか。FlightAvailableSeats 項目属性の Subtract を使用する更新処理は InvoiceFlightSeatQty 項目属性に依存しており、その FlightAvailableSeats 項目属性には Error ルールのトリガーが依存しています。そのため、Error ルールの条件を記述する際は、この依存関係があるために Subtract が必ず実行される点を理解している必要があります。「0 未満」の条件が指定されているのはそのためです。既に学習したように、負の数が入力された場合、このエラーが発生するまでに行われたツリー内の処理がすべて元に戻されます。

また、InvoiceFlightAmount 式による更新も InvoiceFlightSeatQty に依存しており、ヘッダーと同様に、InvoiceTotalAmount が InvoiceFlightAmount に依存し、顧客の合計購入金額 (CustomerTotalPurchases) の更新が InvoiceTotalAmount に依存しています。

ツリーが下から上に向かって実行されることが想像できます。つまり、項目属性の値が更新されるたびに、その項目属性に依存するすべてのルールおよび式 (およびツリーでそれより上部に配置されているもの) が実行されます。

評価ツリー

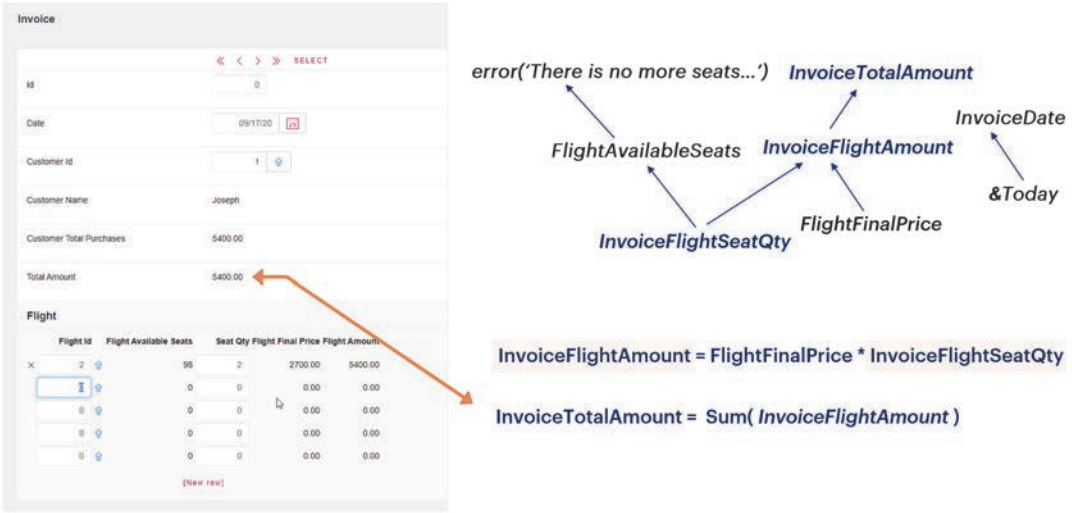


引き続き前の例を確認します。

請求書の行で座席数 (InvoiceFlightSeatQty) が更新されると、この項目属性はフライトの費用 (InvoiceFlightAmount) を計算する式の一部であるため、その式が再度トリガーされます。

GeneXus

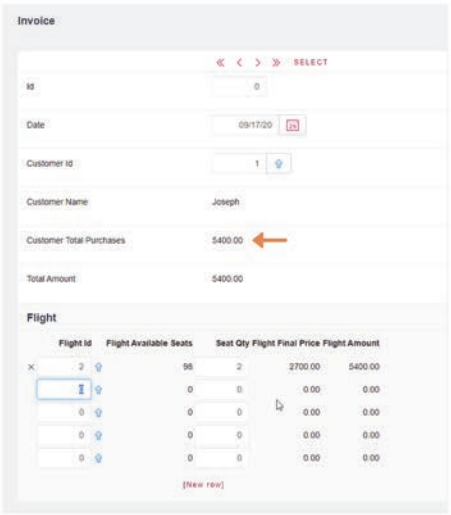
評価ツリー

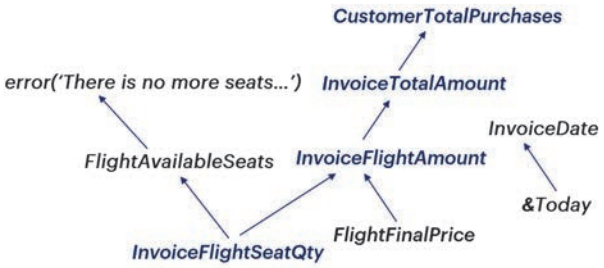


再度トリガーされると、請求書の合計金額 (InvoiceTotalAmount) に対応する式も再度トリガーされる必要があります。

GeneXus

評価ツリー





InvoiceFlightAmount = FlightFinalPrice * InvoiceFlightSeatQty

InvoiceTotalAmount = Sum(*InvoiceFlightAmount*)

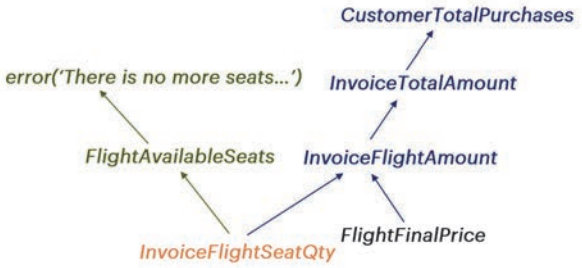
Add(*InvoiceTotalAmount*, CustomerTotalPurchases);

最後に、合計金額が変更されると、Add(InvoiceTotalAmount, CustomerTotalPurchases) ルールもトリガーされる必要があります。これは、顧客の合計購入金額を更新する必要があるためです。

GeneXus

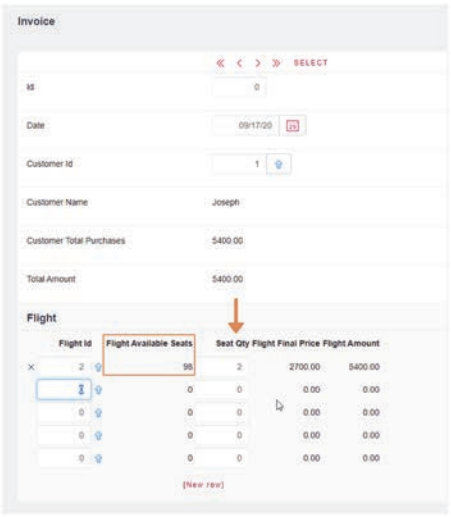
評価ツリー

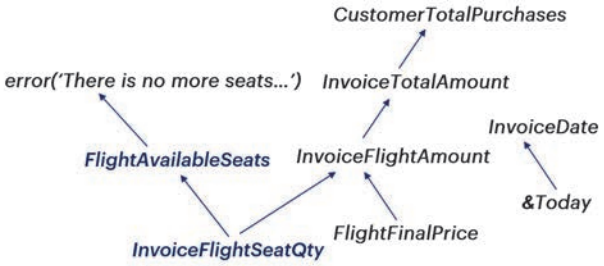
```
(R) Default( InvoiceDate, &Today );
(R) Add( InvoiceTotalAmount, CustomerTotalPurchases );
(F) InvoiceTotalAmount =  Sum( InvoiceFlightAmount )
(F) InvoiceFlightAmount = FlightFinalPrice * InvoiceFlightSeatQty
(F) FlightFinalPrice = FlightPrice * (1 - AirlineDiscountPercentage...)
(R) Subtract( InvoiceFlightSeatQty, FlightAvailableSeats );
(R) Error( "There are no more seats for sale" ) if FlightAvailableSeats < 0;
```



InvoiceFlightSeatQty 項目属性によって、ツリーの右側の枝に含まれるすべての式およびルールがトリガーされることに加え、左側の枝に含まれる式およびルールもトリガーされます。

評価ツリー






```
Subtract( InvoiceSeatQty, FlightAvailableSeats );

Error( "There are no more seats for sale" )
if FlightAvailableSeats < 0;
```

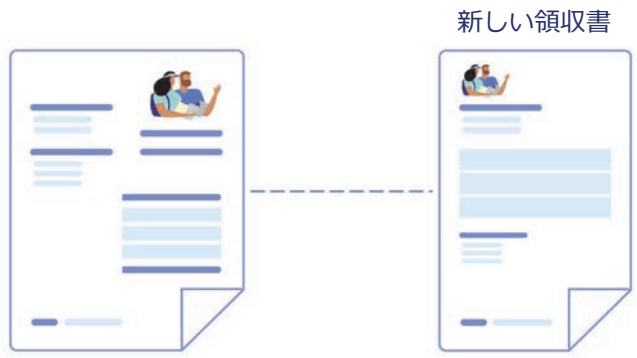
既に確認したように、InvoiceFlightSeatQty 項目属性の値が変更されると、フライトの利用可能な座席数 (FlightAvailableSeats) を更新する Subtract(InvoiceFlightSeatQty, FlightAvailableSeats) ルールも再度トリガーされます。

その結果、このルールが変更されると、Error ルール (利用可能な座席がもうないことを表す) がトリガーされる必要があるかどうかを確認するために、FlightAvailableSeats 項目属性の値が評価されます。

Error ルールをトリガーする条件が満たされた場合、InvoiceFlightSeatQty 項目属性の変更後にツリー内で行われた処理はすべて自動的に元に戻され、データベースのデータは Error ルール実行前の状態に戻ります。

GeneXus

例



ここでは、ルールが必ずしも想定どおりのタイミングでトリガーされない例を確認しましょう。購入済みのフライトの座席について顧客に請求書を発行した直後に、支払いに対する領収書を生成するとします。顧客に対する過去の請求書がすべて支払い済みである場合は新しい領収書を生成します。

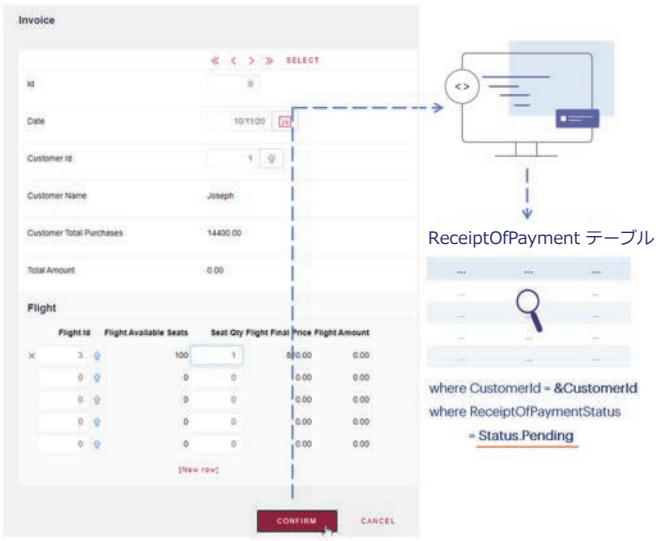
GeneXus

例

そうでない場合は、過去の未払いの領収書に、今回の請求書の金額を加算します。ReceiptOfPayment トランザクションが作成されています。このトランザクションは、領収書の識別子、日付、状態 (「保留中」または「完了済み」の値を持つ列挙型ドメイン)、顧客、第 2 レベルの項目属性 (領収書の発行対象となる請求書を記録する) で構成されます。

GeneXus

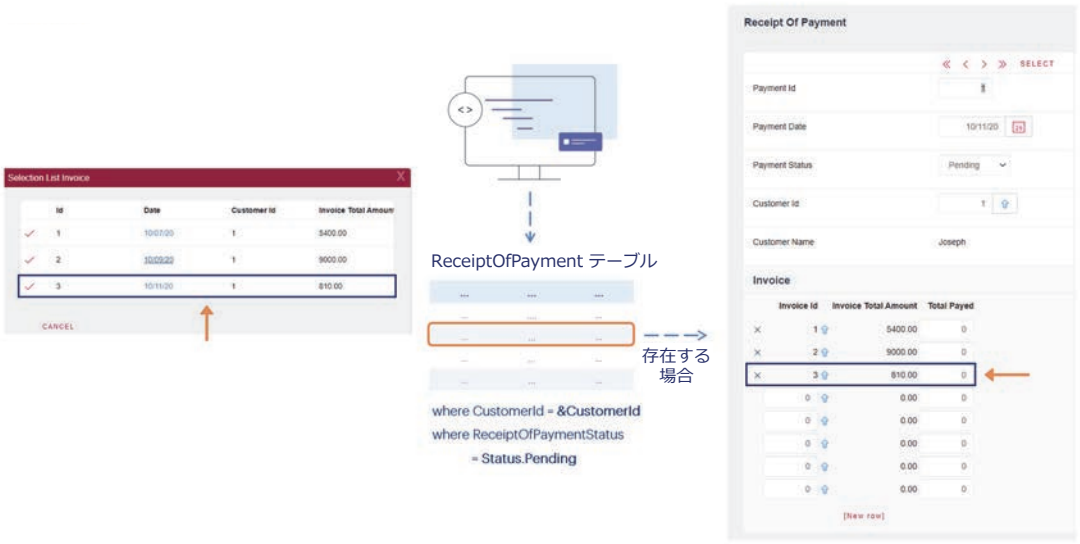
例



新しい請求書の情報が入力されると、その顧客の「保留中」ステータスの領収書が既に存在するかどうかをプログラムが検索します。

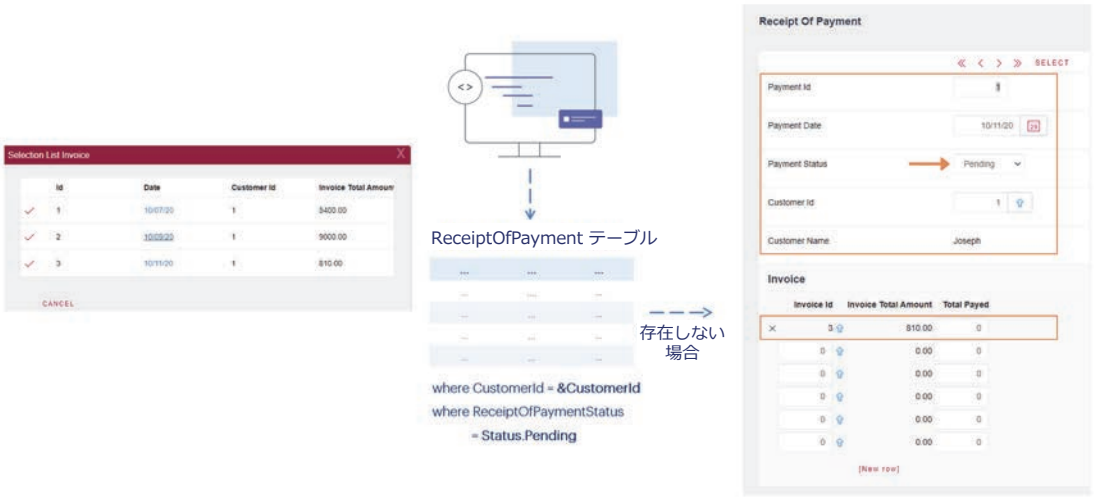
GeneXus

例



存在する場合は、この請求書を含む新しい行が追加されます。

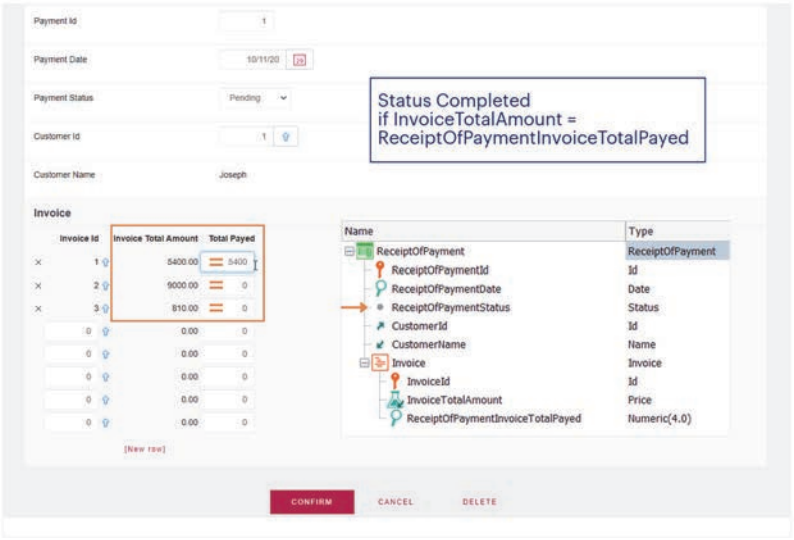
例



存在しない場合は、ヘッダーおよび行が作成されます。ヘッダーは「保留中」ステータスになります。

GeneXus

例

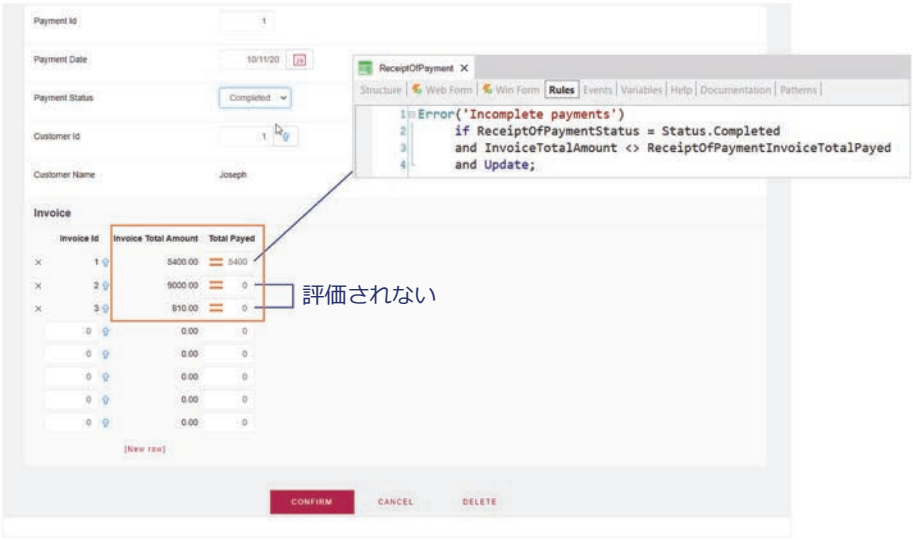


その後、顧客が支払いのために来店したら、従業員はトランザクションを開いて、顧客の支払い対象の請求書の ReceiptOfPaymentInvoiceTotalPaid 項目属性を変更します。

すべての行について InvoiceTotalAmount と ReceiptOfPaymentInvoiceTotalPaid の値が一致した場合にのみ、ReceiptOfPaymentStatus を「完了済み」に変更できます。この実装は、ReceiptOfPaymentStatus の値をユーザーが変更を行うと想定します。そのため、未払いの請求書または支払い内容が不適切な請求書がある場合は「完了済み」に変更できない仕組みになっている必要があります。

GeneXus

例



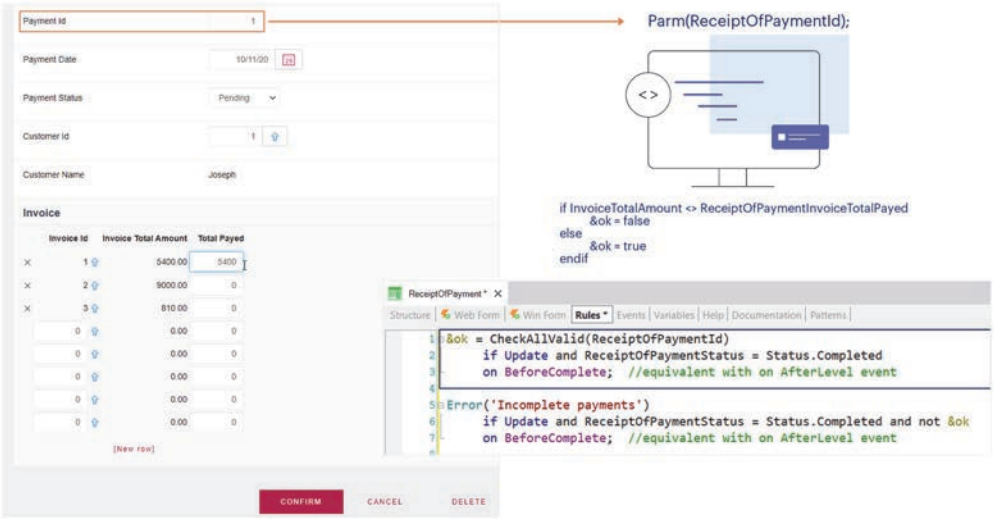
次のような Error ルールを使用することを考えます。領収書の情報を更新してステータスが「完了済み」になったときに、行の支払い予定の金額を表す項目属性の値と、支払い済みの金額を表す項目属性の値が一致しない場合にトリガーされるルールです。

このルールはいつトリガーされるのでしょうか。

トランザクションにアクセスしてステータスを「完了済み」に変更し、行に想定とは異なる値を入力するとトリガーされます。ここで、支払い済みの金額が 0 であるほかの行にアクセスしなかった場合はどうなるでしょう。ルールはトリガーされるでしょうか。

答えは「いいえ」です。Update モードでトランザクションを実行する場合に、ヘッダーを変更せずに 1 つの行だけを変更したい場合があります。その場合は何がトリガーされるでしょうか。ヘッダーは常に更新され、その後、変更された行のみが更新されます。そのため、評価ツリーに従ってヘッダーと変更された行に対し、すべてがトリガーされます。

例



このケースをどのように解決すればよいでしょうか。

解決策の 1 つは、すべての行が変更されて、それらの変更がデータベースに反映された後で、領収書 ID を渡してプロシージャーを呼び出すことです。ただし、これは元に戻せるようにコミット前に行います。このプロシージャーでは、すべての行を参照し、該当する項目属性について値が一致していることを確認します。

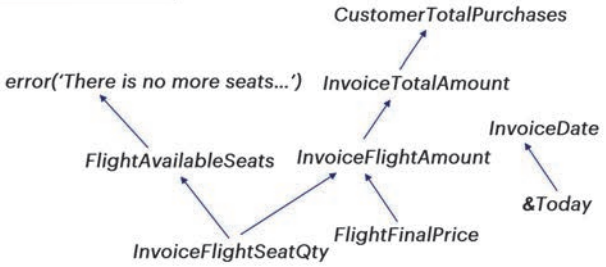
評価対象の &ok 変数が既に条件に設定されているため、Error ルールでは BeforeComplete イベントを条件に設定する必要がないように思うかもしれませんが、実際には必要です。正確に同じイベントを Error の条件に指定しないと、両ルール間の依存関係が崩れます。

トリガーとなる各イベントには固有の評価ツリーがあります。つまり、同じイベントに多数のルールを条件設定すると、既に確認したように、**イベント発生時**に依存関係に従って並べ替えられます。

GeneXus

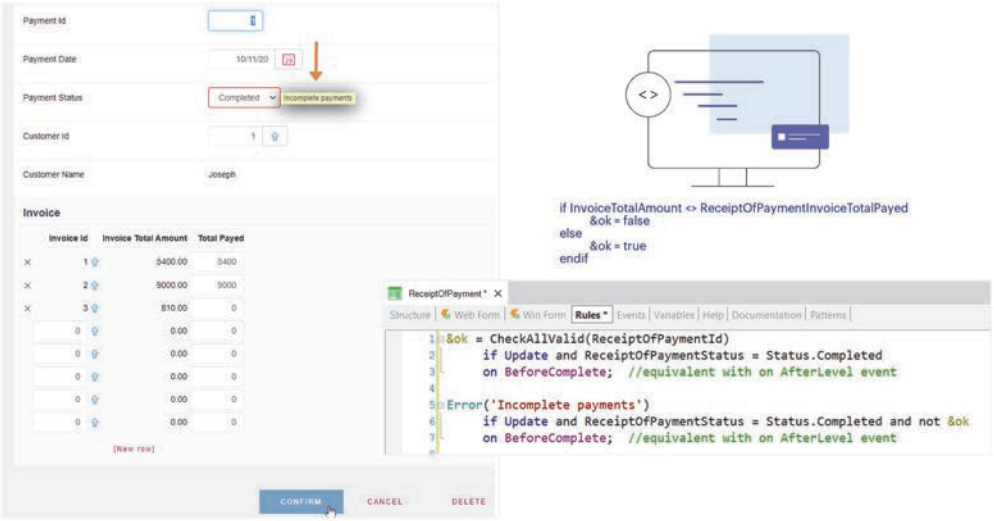
評価ツリー

```
(R) Default( InvoiceDate, &Today );
(R) Add( InvoiceTotalAmount, CustomerTotalPurchases );
(F) InvoiceTotalAmount =  Sum( InvoiceFlightAmount )
(F) InvoiceFlightAmount = FlightFinalPrice * InvoiceFlightSeatQty
(F) FlightFinalPrice = FlightPrice * (1 - AirlineDiscountPercentage...)
(R) Subtract( InvoiceFlightSeatQty, FlightAvailableSeats );
(R) Error( "There are no more seats for sale" ) if FlightAvailableSeats < 0;
```

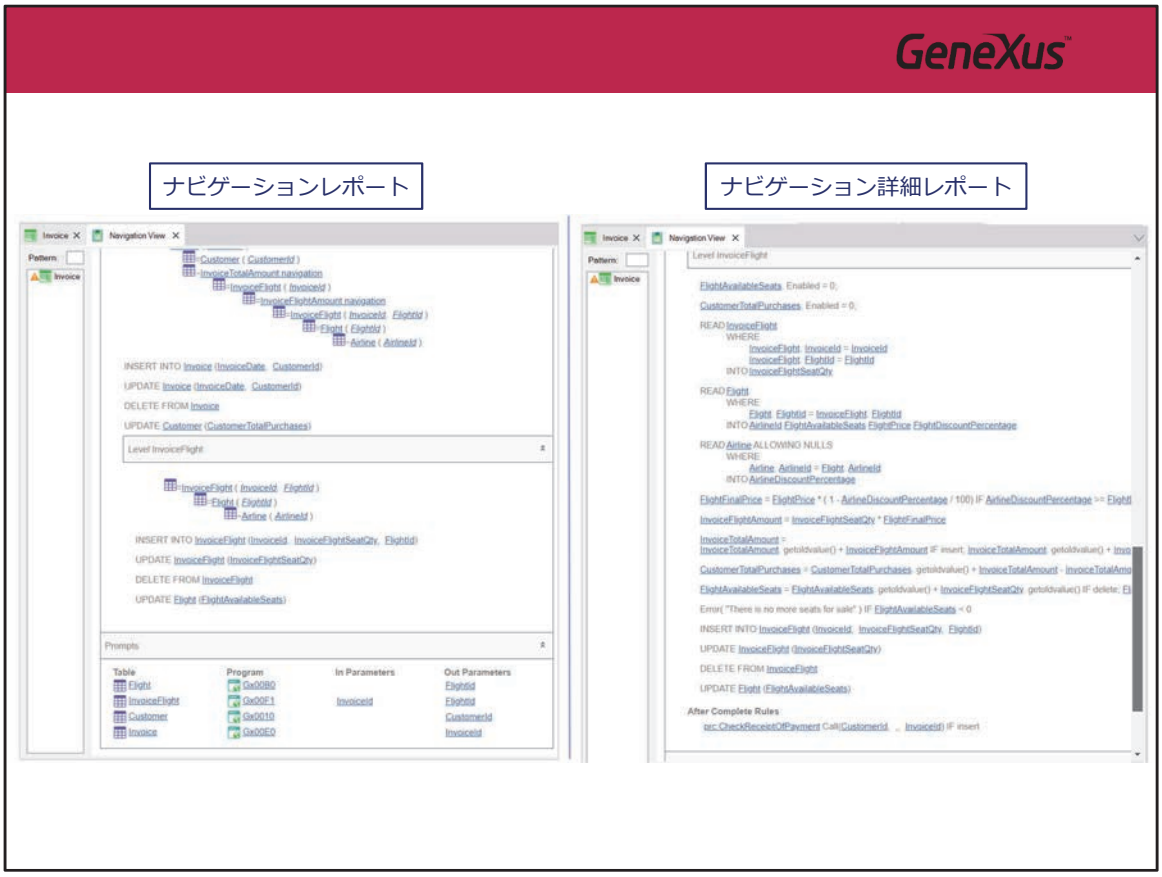


まとめると、トランザクションで宣言されるルールおよび式は通常は相互に関連しており、GeneXus はそれらの間の依存関係と、評価の順序を判断します。

例



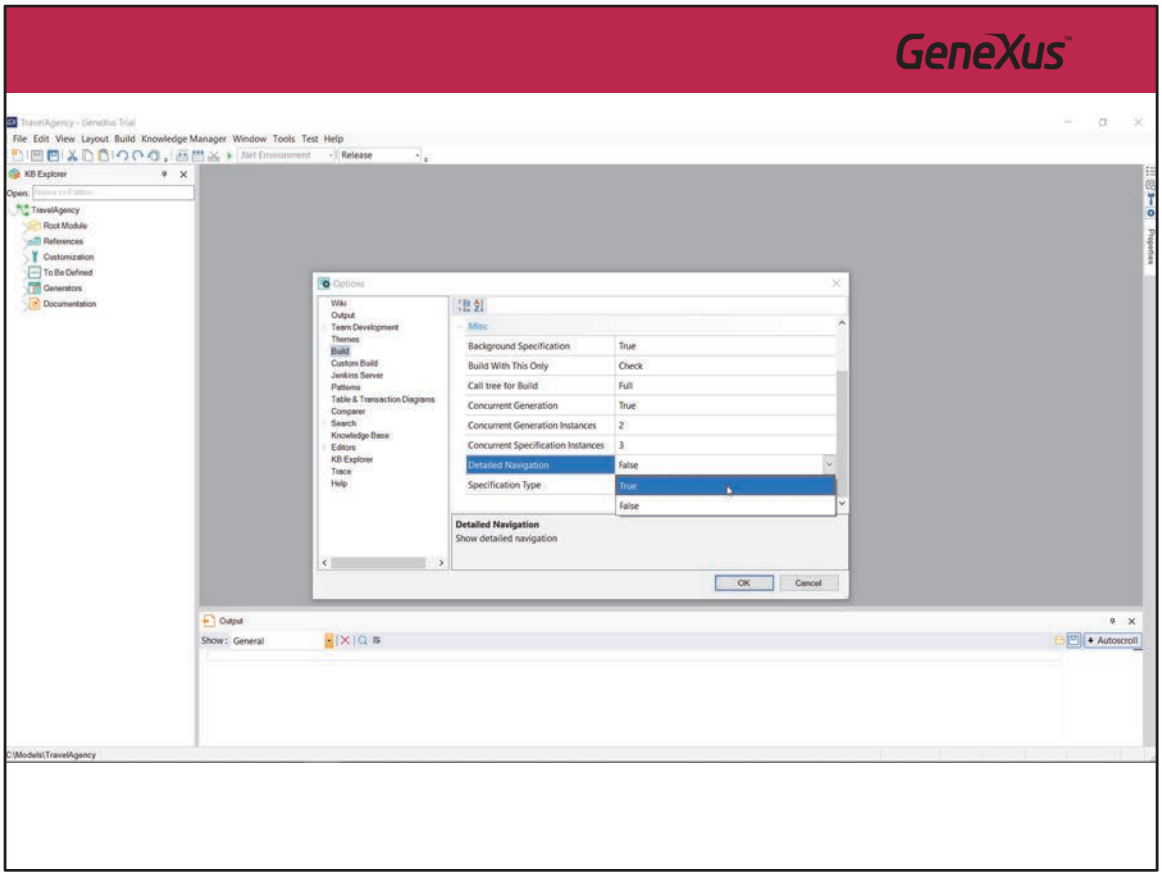
場合によっては、評価ツリーの判断する実行順序が想定どおりにならないことがあります。先ほど確認したケースが分かりやすい例で、行のレコードとその後のエラーを確認するプロシージャーのトリガーのタイミングを遅らせる必要がありました。



GeneXus がトリガーする評価の順序の詳細については、ナビゲーション表示をご覧ください。

ここに示す 2 つの画面の違いを確認しましょう。たとえば、ナビゲーション詳細には、ルールとそれらがトリガーされるタイミングが表示されています。これはもう一方の画面には表示されていません。

式またはルールがトリガーされる正確なタイミングを知りたい場合はナビゲーション詳細が役に立つことがあります。しかし、分析に時間がかかることが多いため、多くの場合は必要ありません。



ナビゲーション詳細を有効にするには、メニューオプションで [ツール] > [オプション] を選択し、[ビルド] カテゴリで [ナビゲーション詳細] プロパティを True にします。