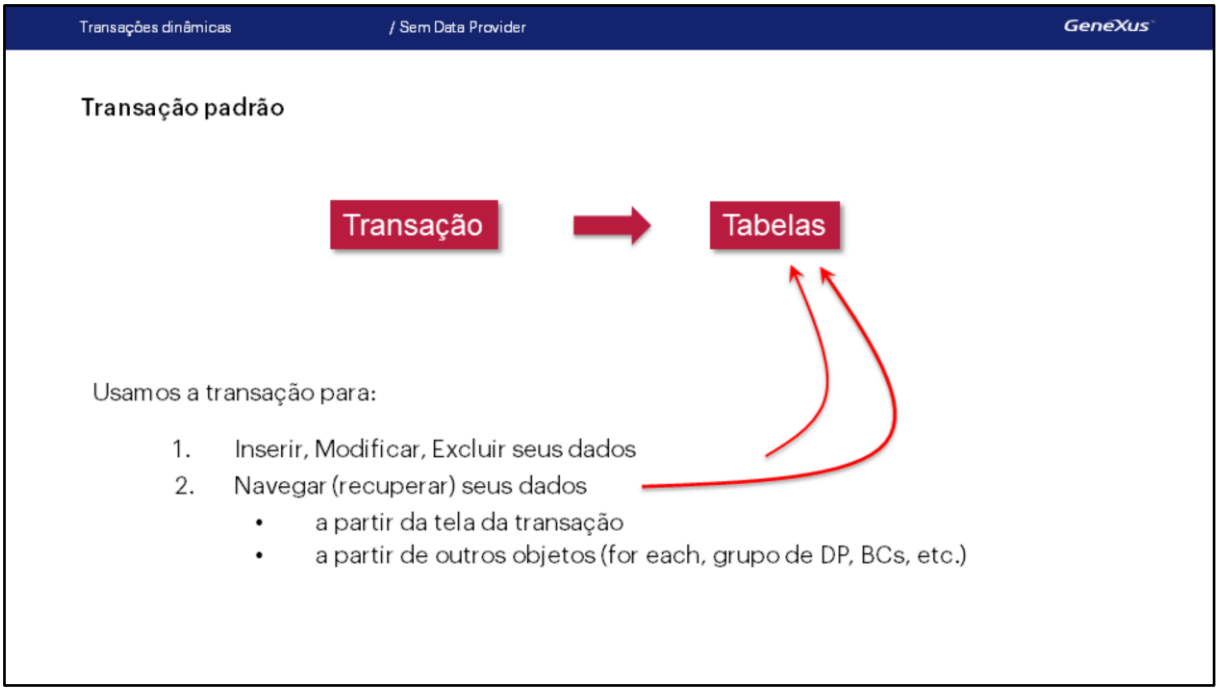
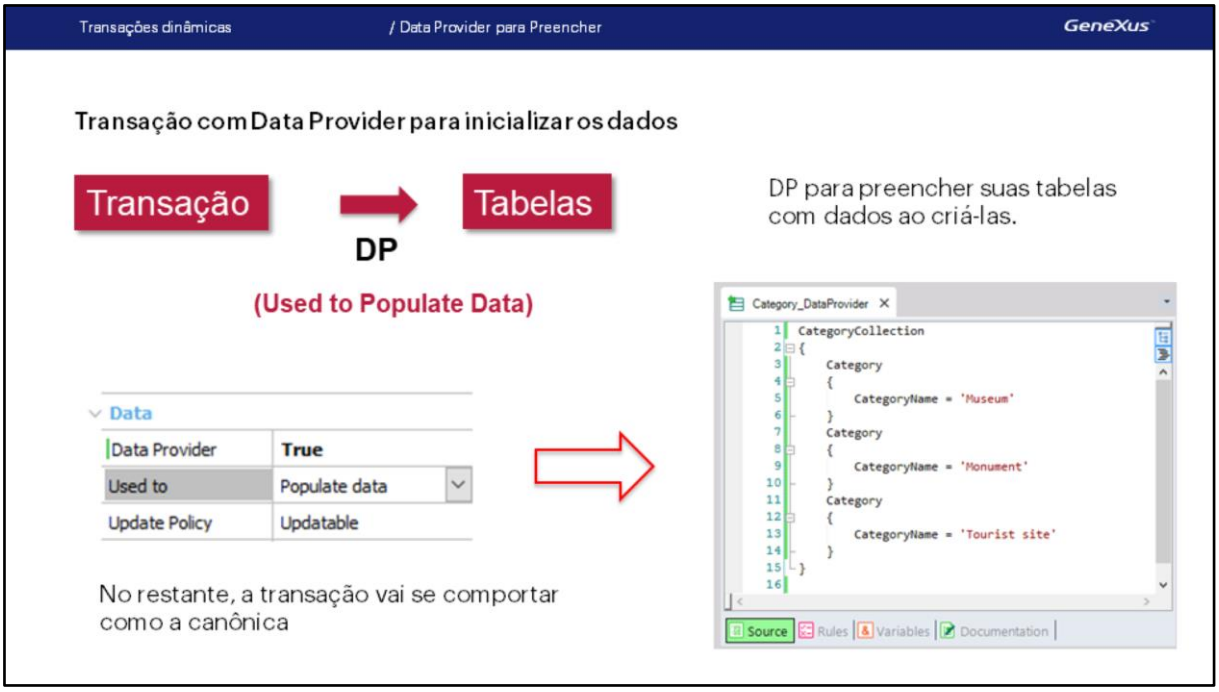
Transações dinâmicas
Recuperação de dados sob demanda
Transações como “Views”

GeneXus 16



Até agora, vimos que para cada objeto transação, uma tabela para cada nível é criada para armazenar seus dados e recuperá-los posteriormente.

A transação, em sua forma canônica, é usada para executar operações de **inserção**, **atualização** e **exclusão** nestas tabelas através de sua tela, bem como navegar (**recuperar**) os dados nestas tabelas.

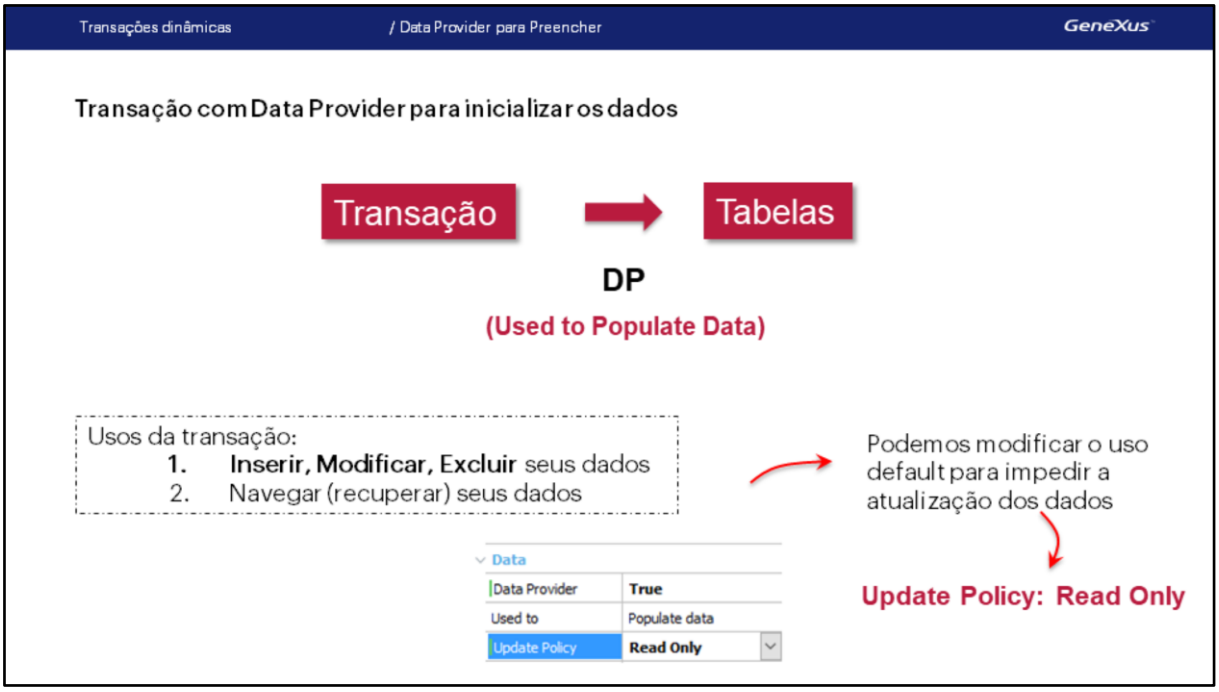


Já tínhamos visto como **associar um Data Provider com a transação** para **popular** sua(s) tabela(s) com dados.

Lembre-se que o Data Provider "Used to" popular as tabelas com dados ("Populate data") será executado na reorganização, quando são criadas as tabelas associadas com a transação.

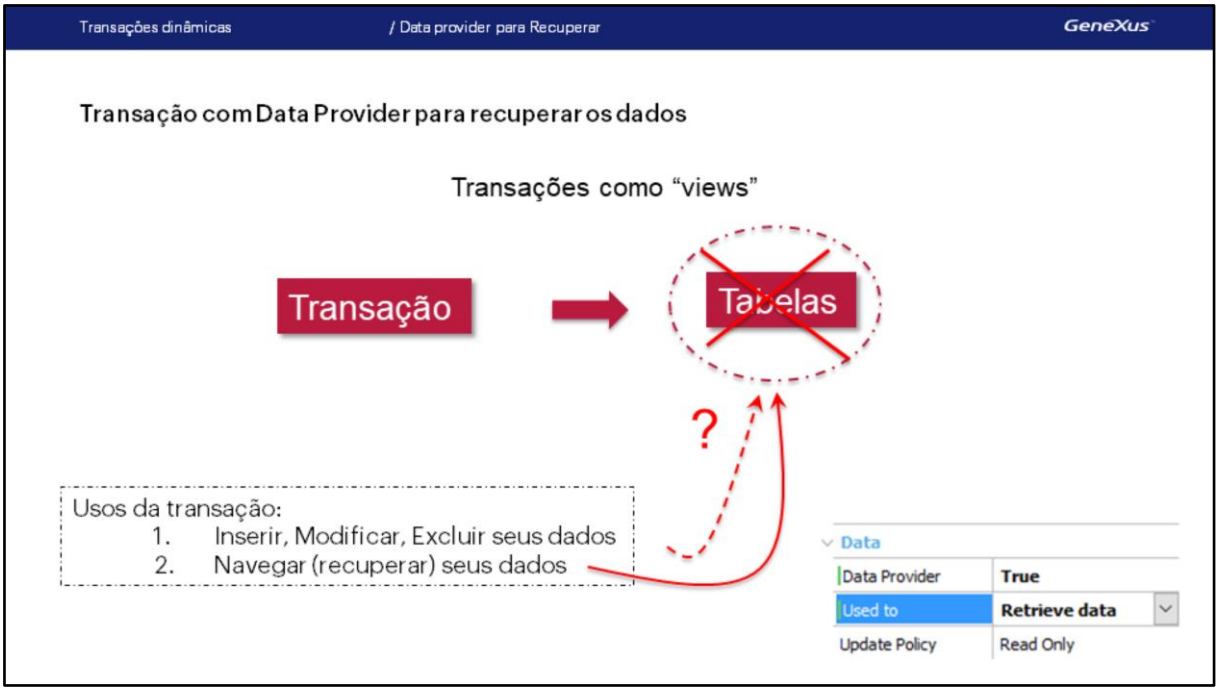
Nota: Se esse Data Provider é alterado a qualquer momento mais tarde, ele será executado novamente no próximo F5. Portanto, os dados que já estão incluídos nas tabelas devem ser manuseados com cuidado.

O Data Provider é usado apenas para fins de inicialização. Em seguida, a transação se comportará como um canônico; isso quer dizer, ela irá acessar suas tabelas como de costume para recuperar os dados e permitirá **inserir**, **atualizar** e **excluir** registros de forma habitual. Preste atenção à propriedade **Update Policy** e seu valor **Updatable** .



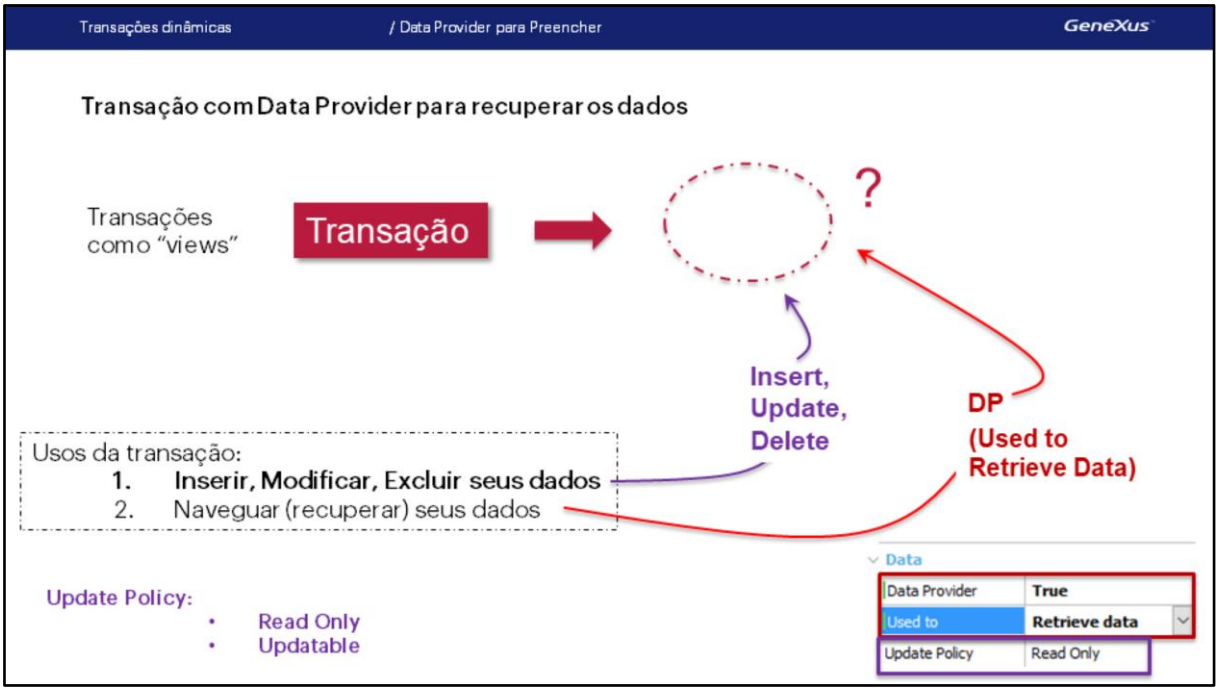
Podemos também mudar o uso padrão da transação para evitar que seus dados sejam atualizados. Isso quer dizer, uma vez que as tabelas foram inicializadas com dados, não será possível alterar os dados existentes ou criar novos dados.

Para este fim, depois da indicação que a transação será associada a um Data Provider usado para "Popular dados", a update policy que está como “Updatable” por padrão será alterada para “Read Only”.



Aqui veremos que é possível manter os usos de transação — inserir, atualizar, excluir e navegar (recuperar) seus dados — sem armazenar dados nas tabelas canônicas. Em suma, teremos as transações que não criam tabelas no banco de dados da aplicação.

Se não são criadas tabelas, teremos que indicar onde as informações serão recuperadas cada vez que seus dados são navegados. Além disso, temos de indicar o que fazer quando o usuário insere dados na tela e quer "Inserir" nas "tabelas" (ou "atualizar" ou então "deletar").

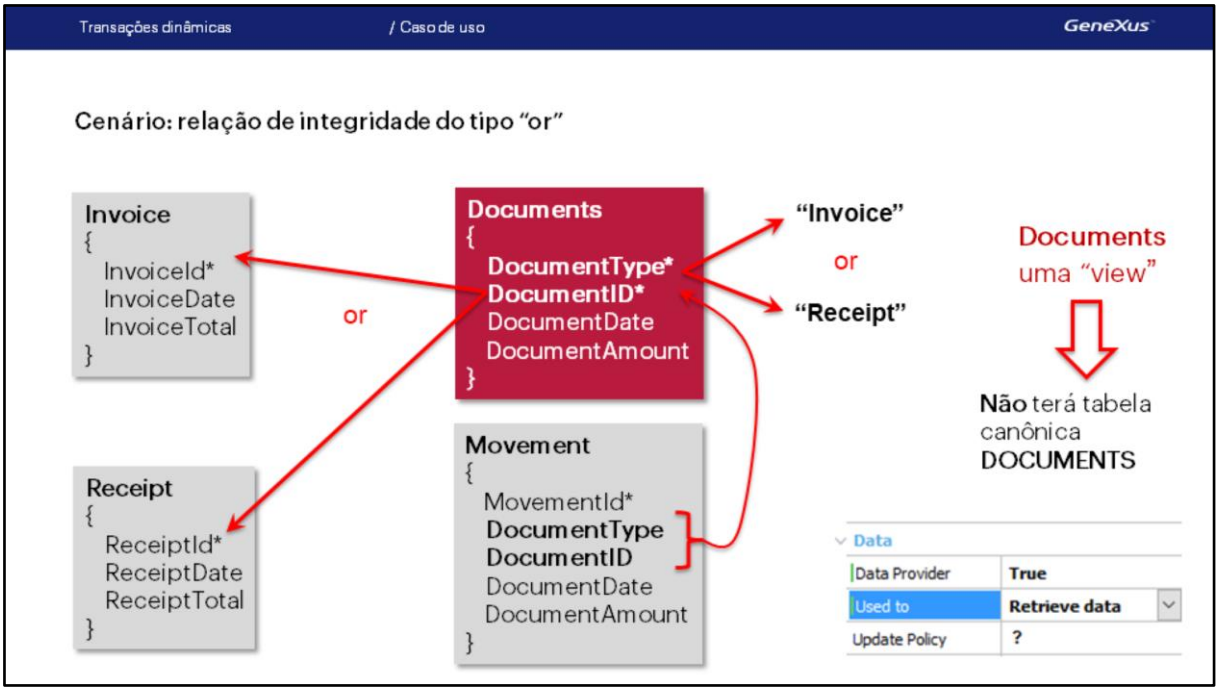


Para inserir, atualizar ou excluir dados, teremos que programar explicitamente três eventos com esses nomes.

Para Navegar (recuperar) dados de uma transação, teremos que programar um Data Provider associado com a transação.

Quando usamos um Data Provider apenas para popular as tabelas com dados, poderíamos evitar atualizações, definindo uma propriedade para indicar que o processo seria read-only; aqui também podemos querer usar a transação somente para recuperar seus dados, não para atualizá-los. A propriedade usada será a mesma: é a chamada Update Policy e aceita os dois valores que mostramos.

Em seguida vamos ver um exemplo para explicar o que fazer e como fazê-lo.

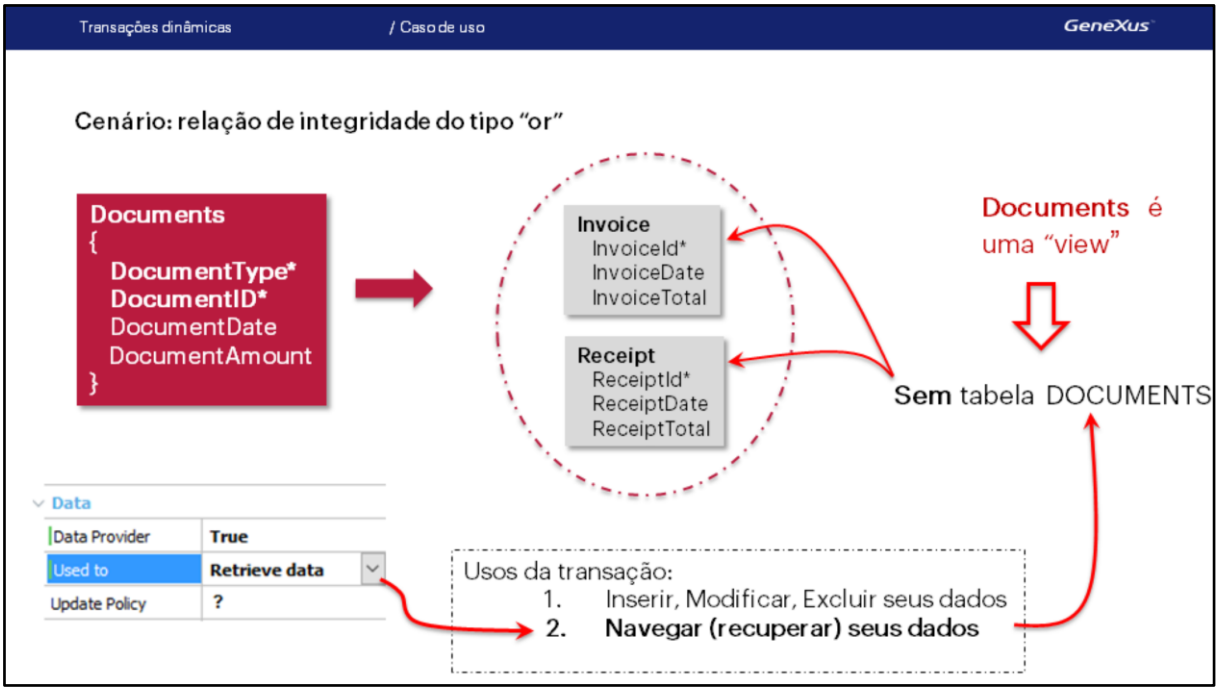


Vamos supor que temos duas transações padrão:
Invoice, para representar as faturas emitidas pela agência de viagens aos seus clientes para compra de ingressos, viagens e assim por diante. Estas faturas são identificadas com um número sequencial.
Receipt que é usado para representar os recibos emitidos pela agência de viagens aos seus clientes para suas compras. Os recibos também são identificados com números sequenciais.

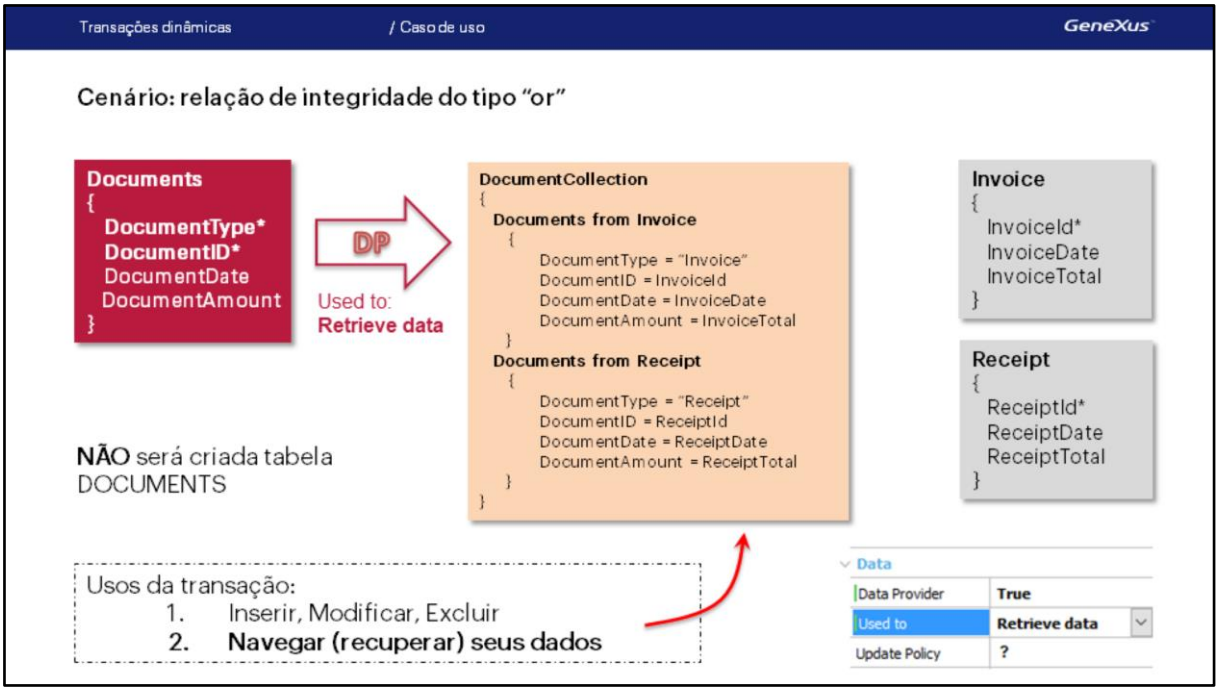
O sistema de contabilidade da agência de viagens terá que lidar com as faturas e recibos em "movements". Em movimentos, recibos são um tipo de documento, como faturas. Os movimentos são identificados com um único ID autonumerado . Observe que o movimento 1 pode pertencer à fatura 1, e movimento 2 pode pertencer ao recibo 1. A transação de movimento unifica os dados das faturas e recibos.

É por isso que a transação Document é criada com um identificador formado por DocumentType e DocumentID. Esta transação será como uma "visão" que combina as informações incluídas nas tabelas Invoice e Receipt. Isso quer dizer, ele não cria uma tabela para conter os dados; em vez disso, ele os pegará de tabelas correspondentes a Invoice e Receipt, que têm nomes idênticos. Então, a transação Movement será uma transação padrão que irá gerar uma tabela MOVEMENT com DocumentType, DocumentID como pseudo chaves estrangeiras.

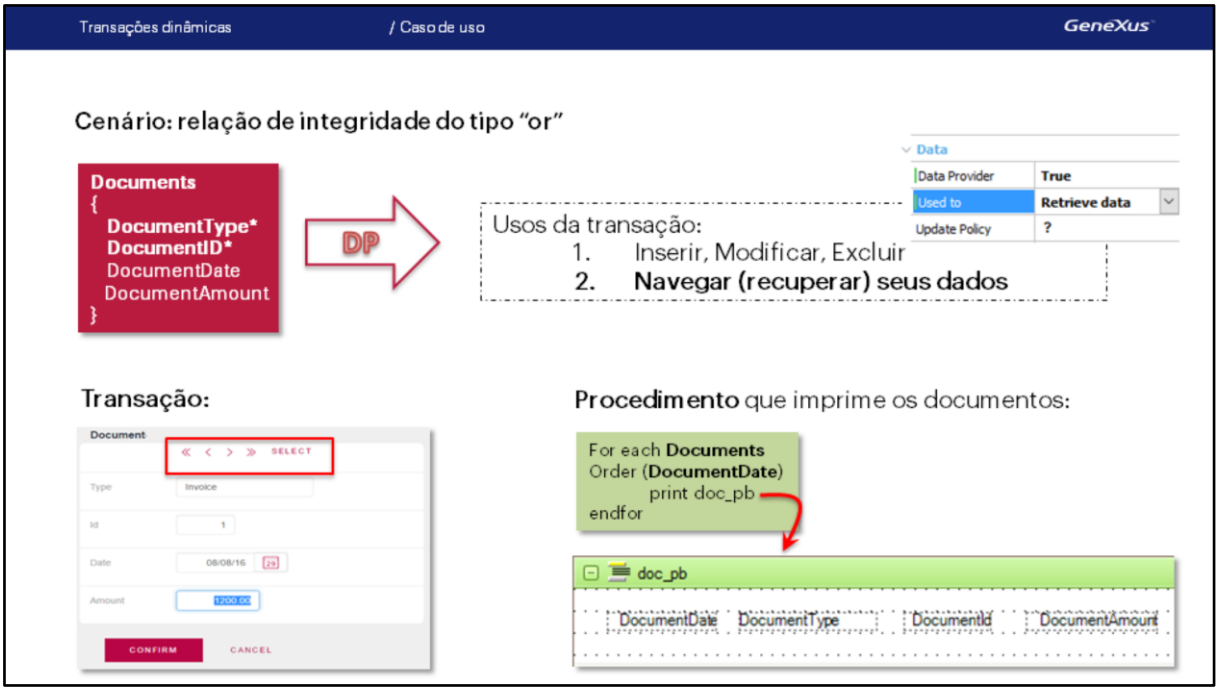
Por que é uma "pseudo" chave estrangeira? Porque não haverá uma tabela física DOCUMENT com chave primária DocumentType, DocumentID para a qual fazer referência. No entanto, continuará a existir no nível lógico e, como nós mencionaremos mais tarde, serão feitos os controles de integridade referencial.



Quando indicamos que a transação terá um Data Provider associado para os dados, a propriedade "Used to" está habilitada (a "Update Policy" está sempre habilitada). Se indicamos que este Data Provider será usado para **recuperar os dados** (Propriedade Used to: Retrieve Data) GeneXus automaticamente entende que a tabela associada à transação não deve ser criada porque esse Data Provider será usado para indicar de onde obter os dados. Neste caso, será das tabelas INVOICE e RECEIPT, que estão associadas com as transações que possuem o mesmo nome.

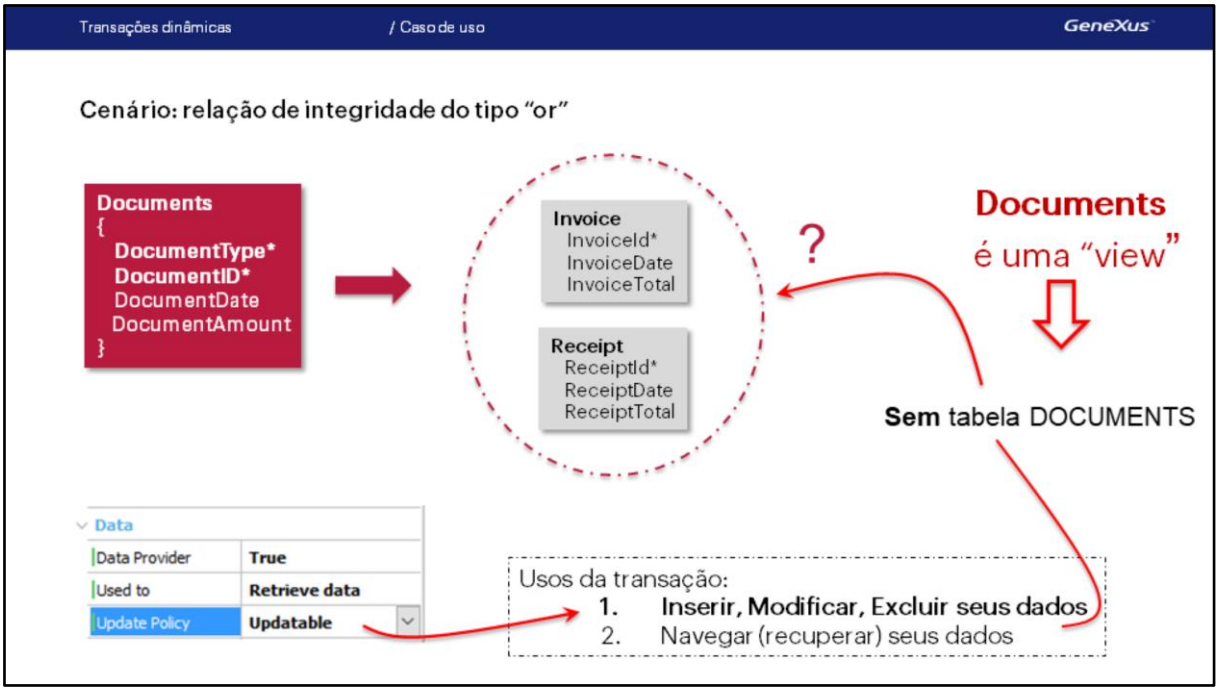


O Data Provider Source estará declarado desta forma. Temos um grupo Document para recuperar todos os documentos que são faturas e outro grupo para recuperar todos os documentos que são recibos.

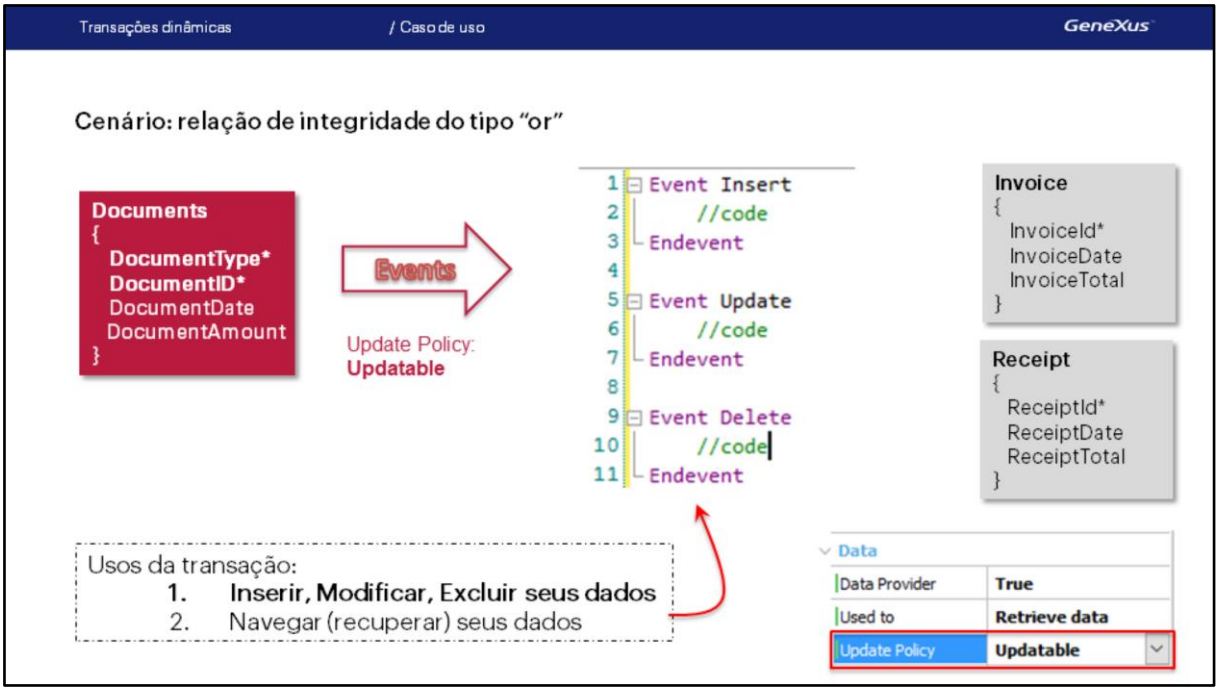


Daqui em diante, toda vez que a transação é executada para navegar seus dados, este Data Provider será executado para carregar as informações correspondentes na tela, de forma transparente para ambos, desenvolvedor e usuário, que nunca notarão que é uma transação sem uma tabela.

Então, a transação dinâmica é usada como qualquer outra transação. Por exemplo, para imprimir todos os documentos ordenados por data em ordem decrescente, será criado um procedimento com o comando For Each que é exibido. No printblock, os atributos DocumentDate, DocumentType, DocumentID, DocumentAmount são adicionados como de costume. GeneXus gerencia para obter estes dados do Data Provider que contém a sua lógica.

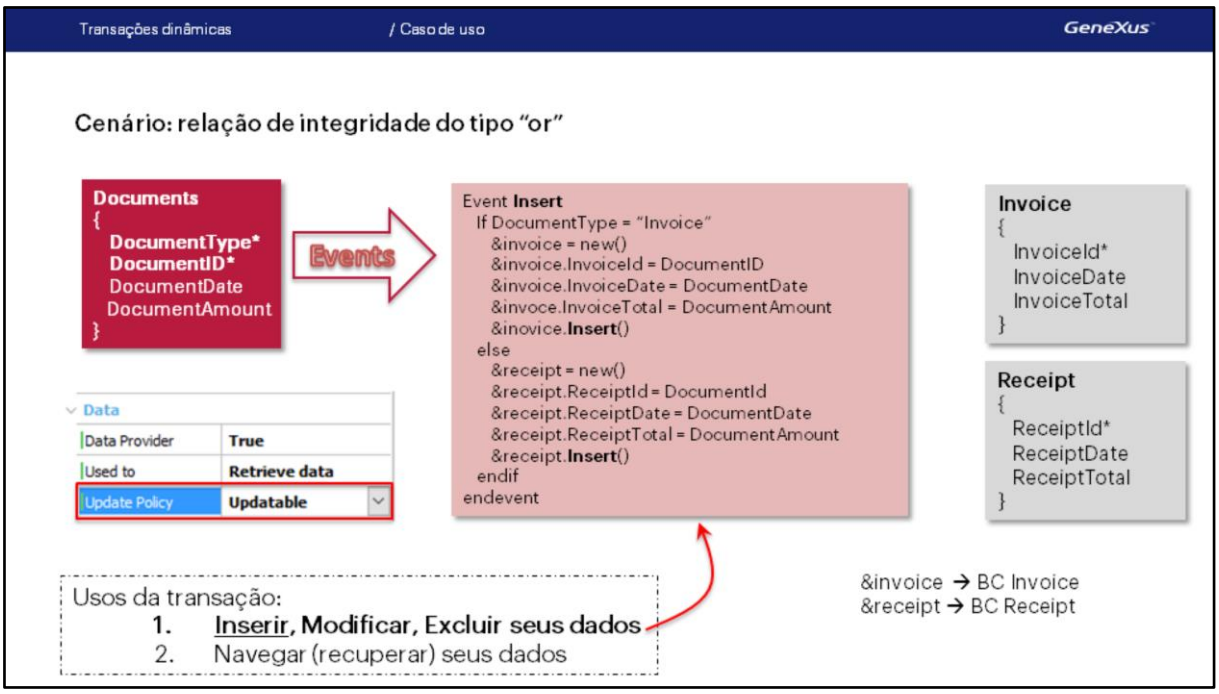


Transações não são usadas apenas para recuperar dados, mas também para atualizá-los. Como podemos fazer isso se não temos uma tabela associada com a transação?



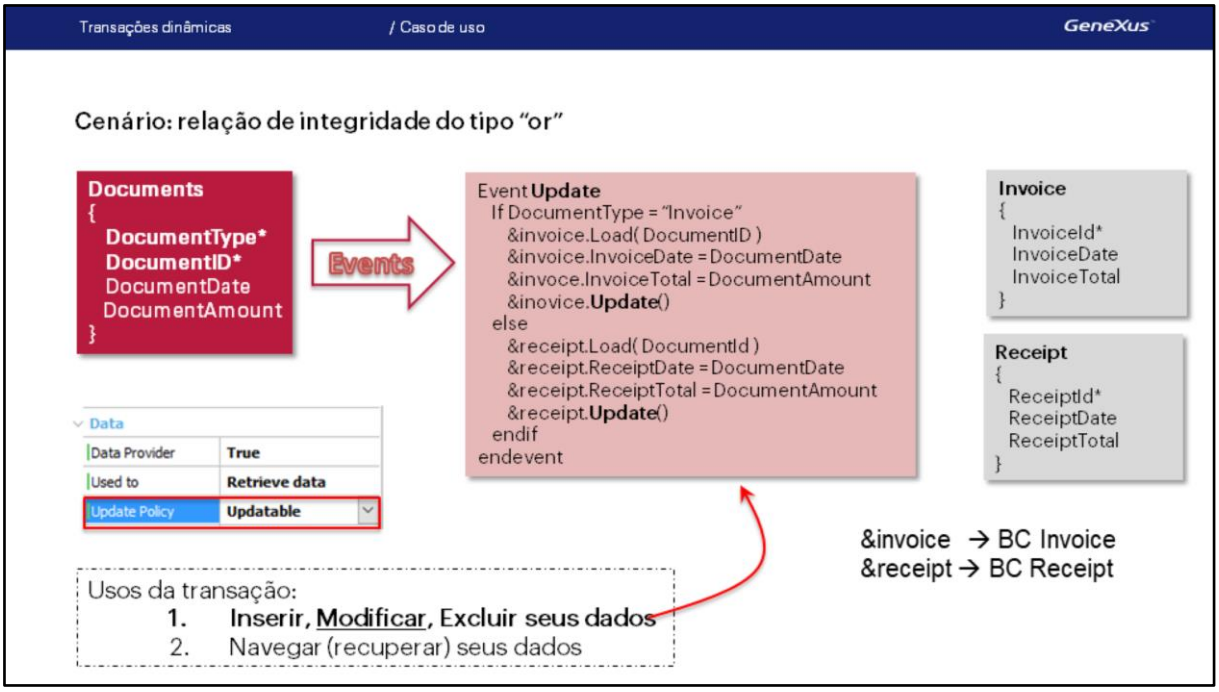
Se a **Update Policy** for definida como "Updatable", os eventos Inserção, Atualização e Exclusão de eventos serão oferecidos para programar a inserção, atualização e exclusão dos dados inseridos pelo usuário na tela. Somente o desenvolvedor irá saber o que fazer em cada caso com esta informação.

Essas ações podem ou não ser permitidas de acordo com a realidade. No nosso caso, parece que eles não deveriam ser permitidos. No entanto, vamos supor sua permissão.

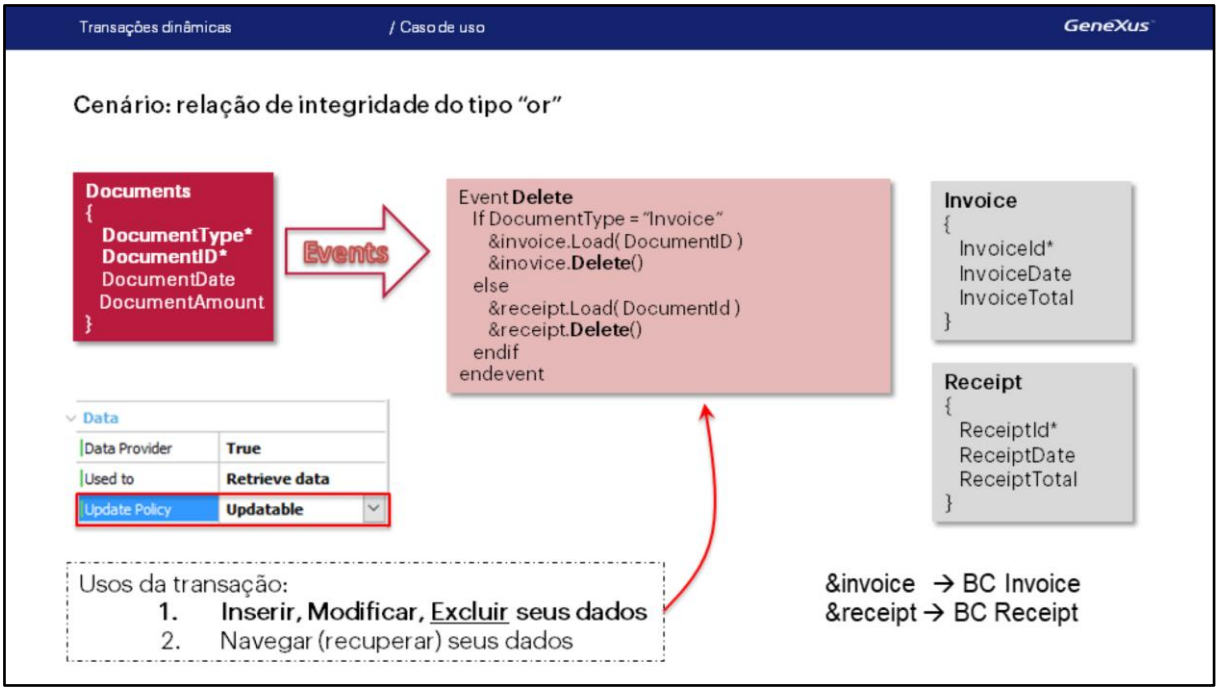


Quando o usuário terminou de preencher os campos da tela para inserir um novo movimento e pressionou Confirm, teremos que inserir um novo registro na tabela Invoice ou Receipt , dependendo do valor fornecido pelo usuário para o campo DocumentType. Para isso, usamos as variáveis &Invoice e &Receipt do tipo de dados Business Component Invoice e Receipt, respectivamente (que devem ter sido obtidos das transações).

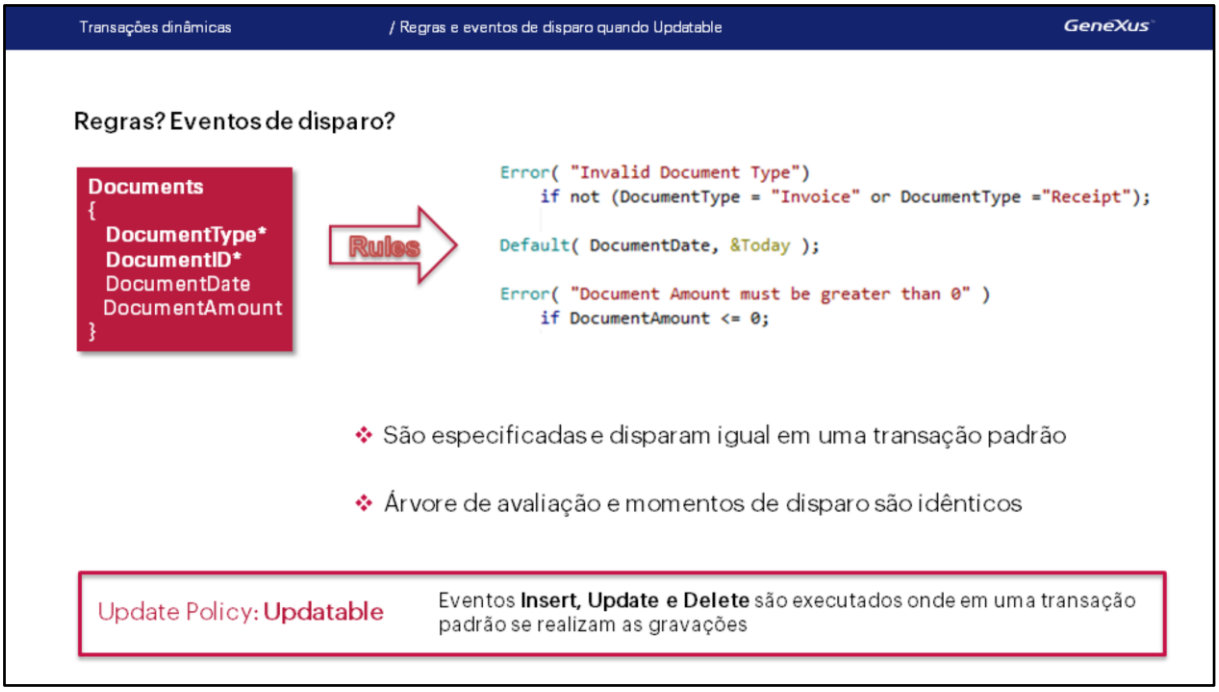
Em vez do método Insert do Business Component, poderíamos ter usado o método Save. Observe que não precisamos escrever o comando Commit, porque estamos na transação Document que ainda tem o Commit na propriedade Exit definida como Yes por padrão. Isso quer dizer, ele executará implicitamente o Commit .



Aqui vemos como codificaríamos a atualização. Como não sabemos quais campos foram alterados pelo usuário, atribuímos todos os valores.

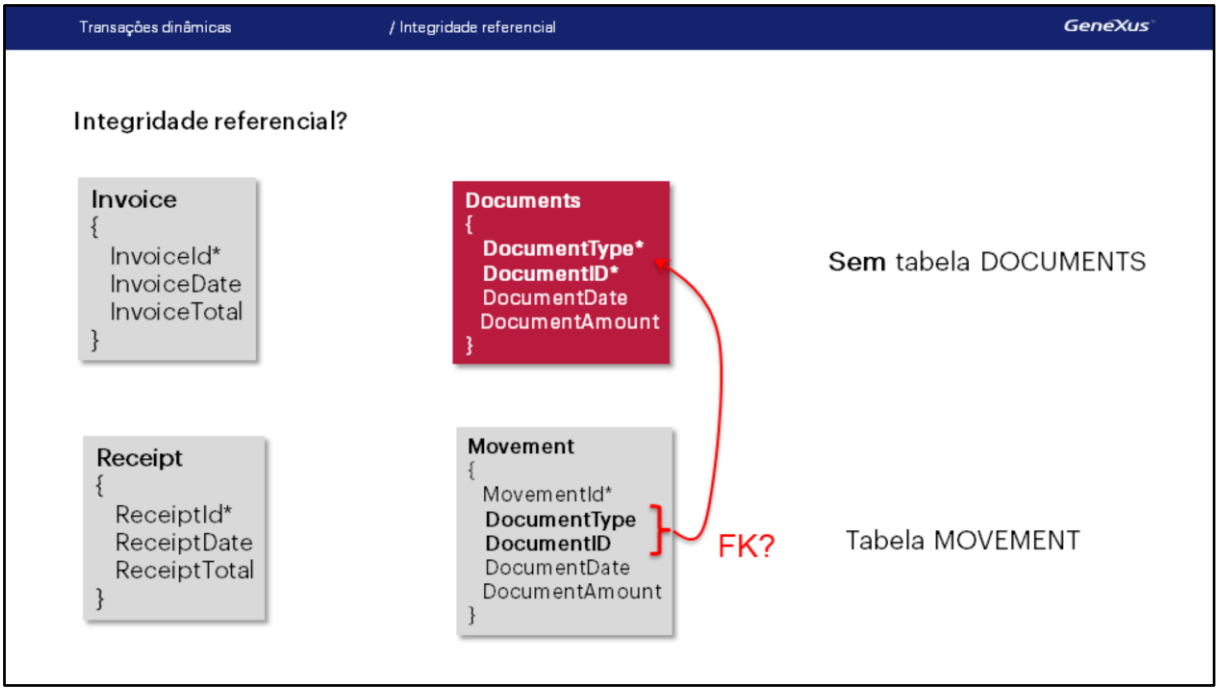


Por último, a operação de Exclusão.



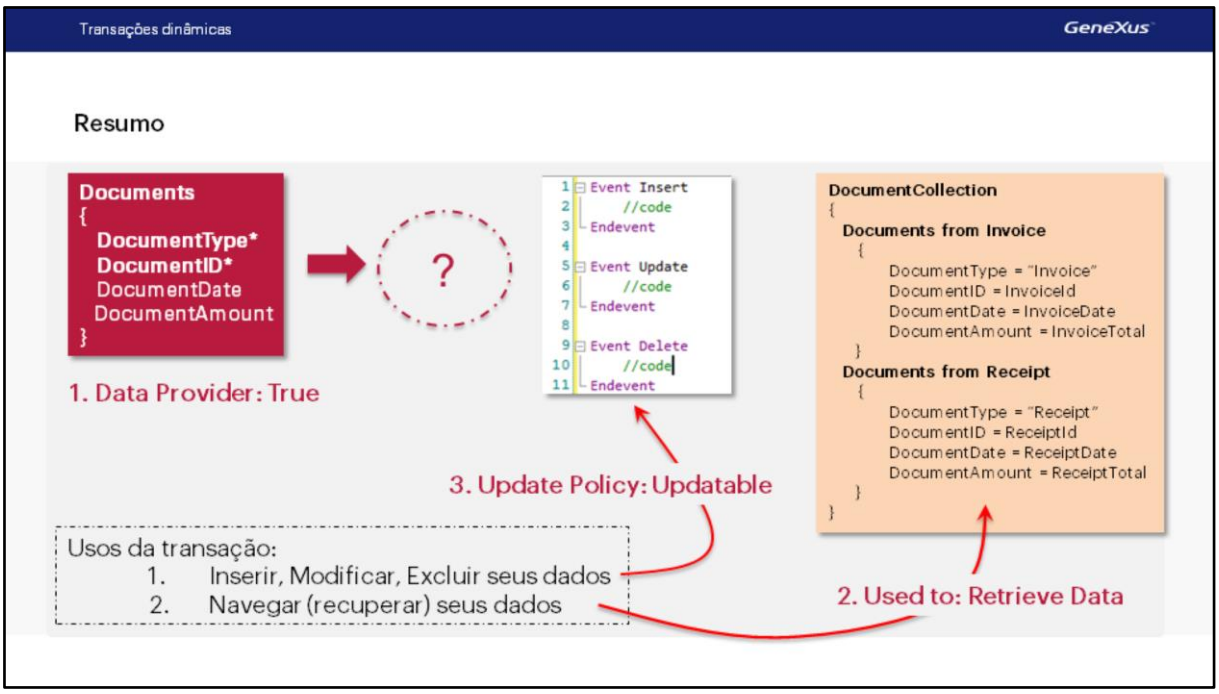
Não nos perguntamos o que acontece se temos regras declaradas no nível da transação dinâmica. Quando eles são acionados? O que acontece com a árvore de avaliação?

O que acontece com as mensagens de sucesso ou falha de operações de Insert, Update, Delete? Elas são similares a “Data was successfully added”.



Como a tabela DOCUMENT associada com a transação Document não será criada, podemos presumir que na tabela MOVEMENT associada com a transação padrão Movement, os atributos DocumentType e DocumentId não serão capazes de compor a chave estrangeira como deveriam. Então, isso significa que o GeneXus não será capaz de verificar a integridade referencial?

Uma vez que deve assegurar a integridade referencial, GeneXus gera gatilhos SQL para fazê-lo. Portanto, poderia ser dito que {DocumentType, DocumentID} compõem uma "pseudo" chave estrangeira em Movement. Em suma, em Document não será possível eliminar faturas ou recibos que tenham um movimento associado. Além disso, não será possível adicionar um Movement que não exista como um documento.



Aqui está um resumo das propriedades e seus efeitos

Resumo

1. Data Provider: True

2. Used to: Populate Data

DP (para inicializar)

3. Update Policy: Updatable

Permite (ou não) as atualizações usuais (sobre as tabelas da transação)

1. Data Provider: True

2. Used to: Retrieve Data

DP (para recuperar)

3. Update Policy: Updatable

Permite (ou não) as atualizações, mas devem ser programadas em eventos especiais: **Insert, Update, Delete**

Aqui você vai encontrar mais exemplos de uso de transações dinâmicas e uma explicação detalhada sobre o tópico. http://wiki.genexus.com/commwiki/servlet/wiki?28062,Dynamic%20Transactions.

Para os provedores de dados externos que forem lidar com repositórios de dados com alguma álgebra relacional (não têm de ser no sentido de bancos de dados SQL), é solucionado de forma diferente importando o serviço (por exemplo: Odata, CouchDB, outros que não são SQL). Ao fazê-lo, GeneXus gera automaticamente a transação e um **Data view**, que é um objeto criado por GeneXus para fornecer a interface de comunicação entre a transação e a "tabela" externa. Este será mais um caso em que a transação não cria uma tabela no banco de dados proprietário. É usado quando a Engenharia Reversa é aplicada (com a ferramenta Database Reverse Engineering).

Neste caso, como com transações dinâmicas, o desenvolvedor usará BCs e comandos For Each como de costume, que serão internamente traduzidos em invocações para o serviço externo.

GeneXus™

The power of doing.

Videos	training.genexus.com
Documentation	wiki.genexus.com
Certifications	training.genexus.com/certifications