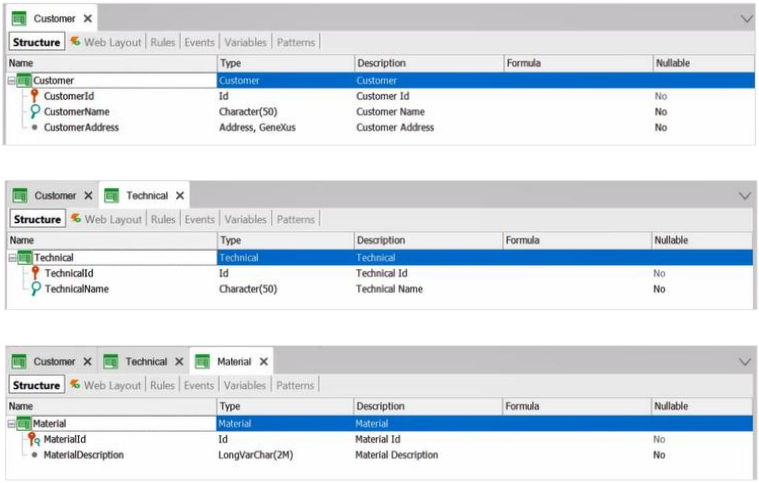
Transações paralelas

GeneXus™

GeneXus

Parallel Transactions

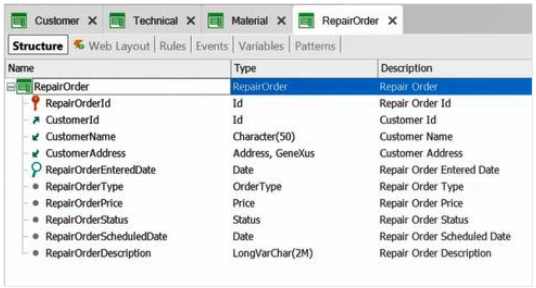


Suponhamos que estamos projetando uma pequena aplicação, para uma empresa que presta serviços técnicos em domicílio para seus clientes. A partir da qual, entre outras coisas, poderemos gerar as ordens de requisição de reparo realizadas pelos clientes.

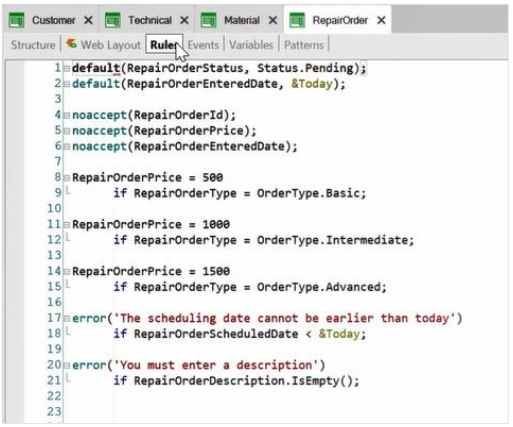
Vejamos apenas as transações que nos interessam para nosso exemplo. Temos uma transação para registrar os clientes. Outra transação para registrar os técnicos da empresa. E uma transação onde será possível registrar os materiais, os quais os técnicos utilizam para resolver os diferentes problemas dos clientes.

RepairOrder Trn

Structure



Rules



Então, temos esta transação (RepairOrder), onde o operador que recebe a reclamação do cliente, vai inserir todos os dados necessários para gerar a ordem de reparo.
Nela, temos como chave primária o atributo RepairOrderId, o qual tem atribuído um tipo de dados autonumerado. Cada ordem terá um cliente associado, portanto, temos inferidos nesta transação atributos da tabela Customer.
E, por último, temos estes outros atributos próprios da transação. Para registrar a data de inserção, o tipo de ordem (cujo tipo de dados será um enumerado que contém estes valores), o preço, um estado (cujo tipo de dados também será um enumerado, que aceitará os seguintes valores), a data de agendamento e uma descrição.

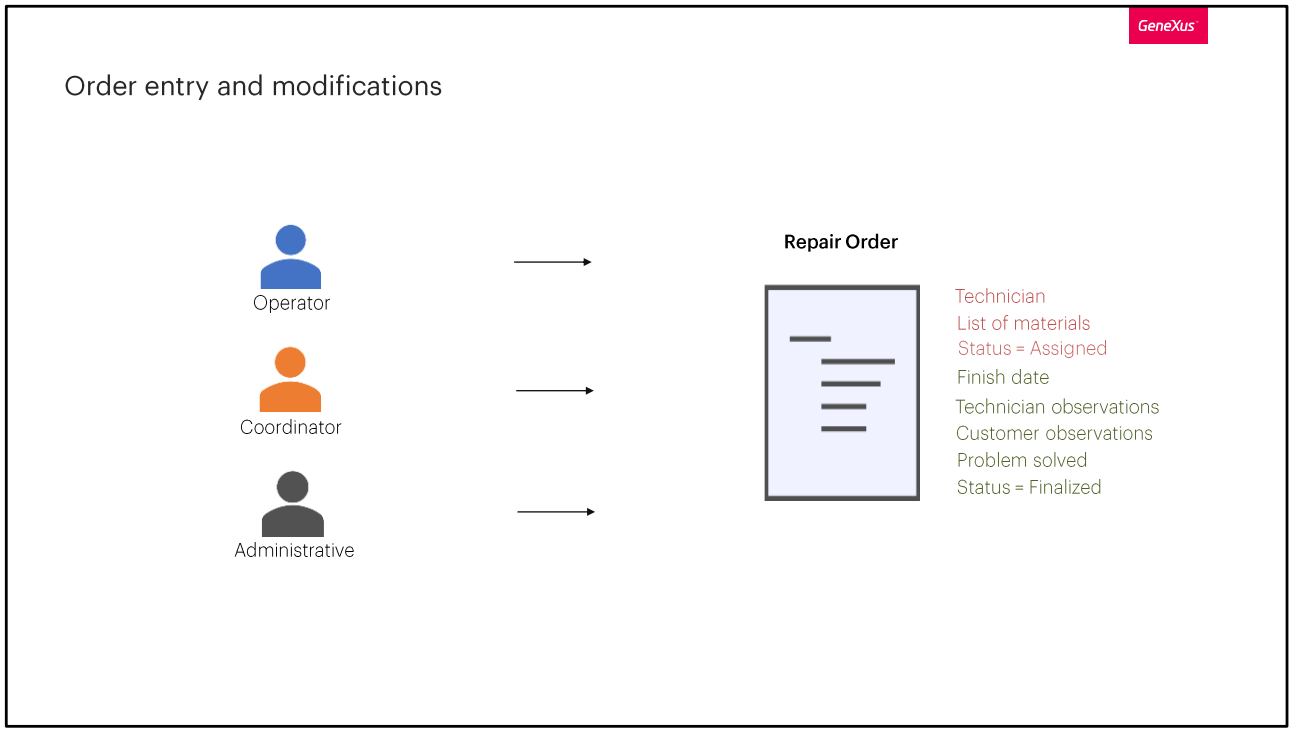
Vejamos agora as regras que inserimos.

Temos um par de regras “default” e várias regras “noaccept”.
Nas regras default atribuímos um valor ao atributo que armazenará o status da ordem, neste caso “P”, de pendente. E neste outro caso, atribuiremos a data do dia ao atributo que armazenará justamente a data de inserção da ordem de serviço.

Vemos também que este atributo (RepairOrderPrice) da transação, será

calculado a partir do valor inserido neste outro atributo (RepairOrderType), que registrará o tipo de ordem.
E por último, declaramos um par de regras “error”.

Vejamos esta transação em execução, a partir da qual inserimos uma ordem de reparo.

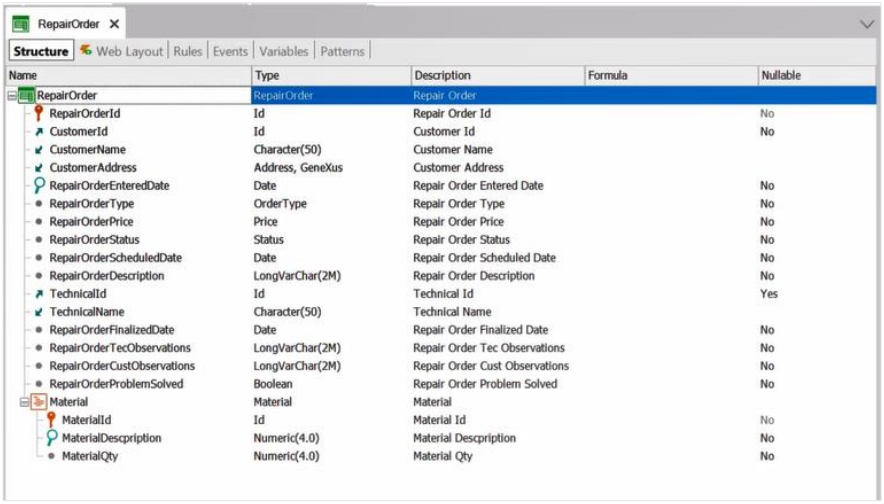


Agora, suponhamos que em nossa realidade, depois de inserida uma ordem, um coordenador deverá atribuir um técnico e os materiais que são fornecidos a ele, para poder ter um controle. E mudar o status dela para que fique como atribuída.

E por sua vez, necessitamos que quando seja finalizada a tarefa pelo técnico, um administrativo possa atribuir à ordem uma data de término, observações do técnico e do cliente, registrar se a falha foi corrigida ou não e alterar o status da ordem para finalizada.

Como podemos fazer isto?

RepairOrder Trn with all attributes



Uma opção seria adicionar todos estes atributos à transação e ir preenchendo aqueles que necessitamos em cada momento.

Vejamos em execução.

Como vemos, esta solução não apresenta uma interface amigável para o usuário, pois sempre haverá campos que não interessam nem devam ser preenchidos.

Por exemplo, quando o operador recebe a reclamação do cliente e insere pela primeira vez a ordem, antes que a mesma seja atribuída a um técnico, veremos vários campos que não serão preenchidos, todos estes. Neste caso, são relativamente poucos, mas se a quantidade de atributos aumenta, isto impactará cada vez mais o usuário e sua experiência com o sistema.
O mesmo acontece quando o coordenador entra na transação, seleciona a ordem que deseja modificar, então esta passa para o modo Update, atribui um técnico à mesma e carrega os materiais.

Já ao pesquisar a ordem e carregá-la na tela, são exibidos atributos que talvez não nos interessem visualizar. Então, para carregar os materiais, deverá deixar campos sem preencher, que ficarão para uma inserção

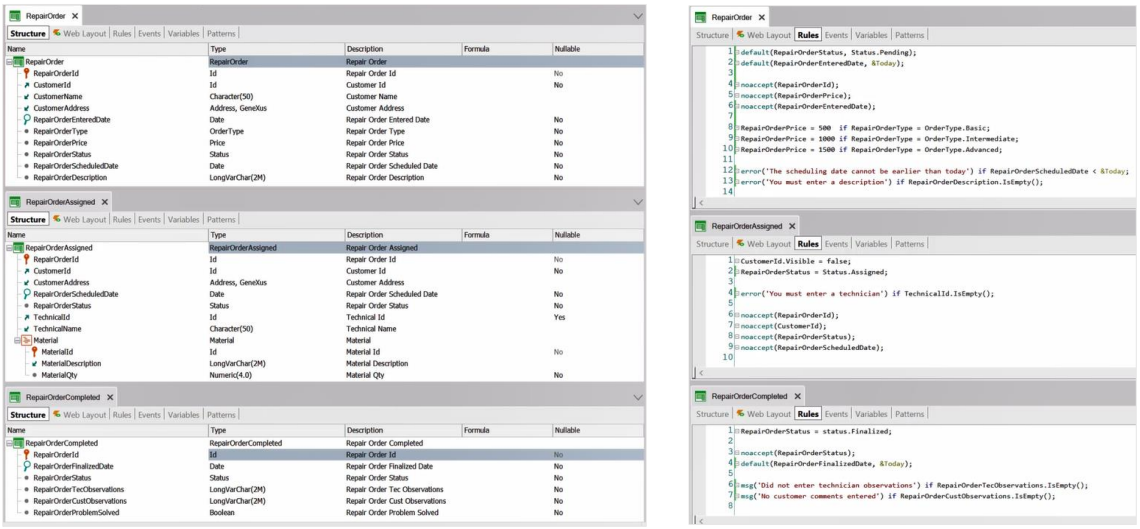
posterior.
Isto evidentemente não é nada cômodo para a inserção. E como acabamos de dizer, por serem poucos atributos neste exemplo, não impacta tanto, mas conforme os atributos aumentam, já que podemos necessitar registrar mais dados, a inserção começa a ficar complexa.

Ou mesmo quando o administrativo deseja finalizá-la com as observações correspondentes. Deverá selecionar a ordem sobre a qual deseja trabalhar, passando a transação para o modo Update, e verá todos os atributos desta transação, quando na realidade só precisa ver alguns poucos dados.

Outra desvantagem é a complexidade que pode haver para programar o comportamento desta transação. Pois, como vimos, ela passará por diferentes estados, e modificações com inserções posteriores, podendo precisar a cada momento de diferentes controles que queiramos fazer, inclusive dos mesmos atributos. Mas, dado este desenho, tudo será programado no mesmo objeto. Não podendo personalizar regras ou eventos próprios de cada um dos momentos que acabamos de ver.

Poderíamos solucionar estas desvantagens de uma maneira simples, mediante o uso do que chamamos de transações paralelas.

Parallel transactions

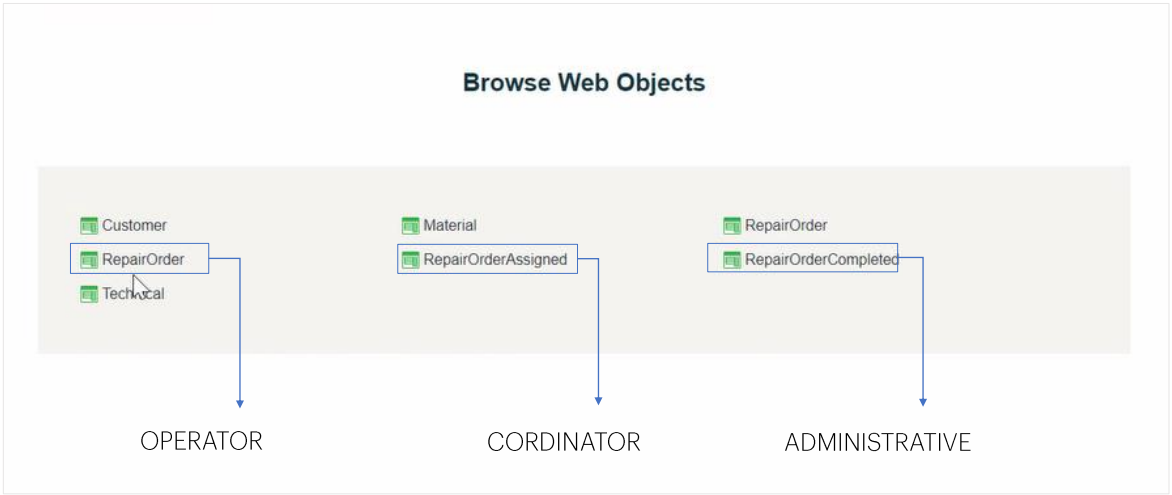


Desta forma teríamos uma transação para a inserção da ordem em primeira instância pelo operador. Outra transação, onde o coordenador atribuirá essa ordem ao técnico e carregará os materiais que serão entregues ao mesmo. E por último uma transação, onde o administrativo poderá carregar as observações do técnico e do cliente, se houverem, e se o problema foi resolvido ou não. Ficando cada uma, com os atributos que realmente nos interessam para cada instância.

Para isto, as três transações deverão ter o mesmo atributo como chave primária. Isto é o que as torna transações paralelas.
Uma coisa importante a ser mencionada é que a paralelização das transações é por nível. Neste exemplo, são paralelos os níveis principais destas transações.

Se observamos a seção regras das transações, vemos que cada uma tem as suas próprias, totalmente independentes umas das outras. Esta é uma das grandes vantagens que se tem ao trabalhar com transações paralelas. O mesmo acontece com os eventos que podemos programar.

Vejamos a aplicação em execução com estas modificações.

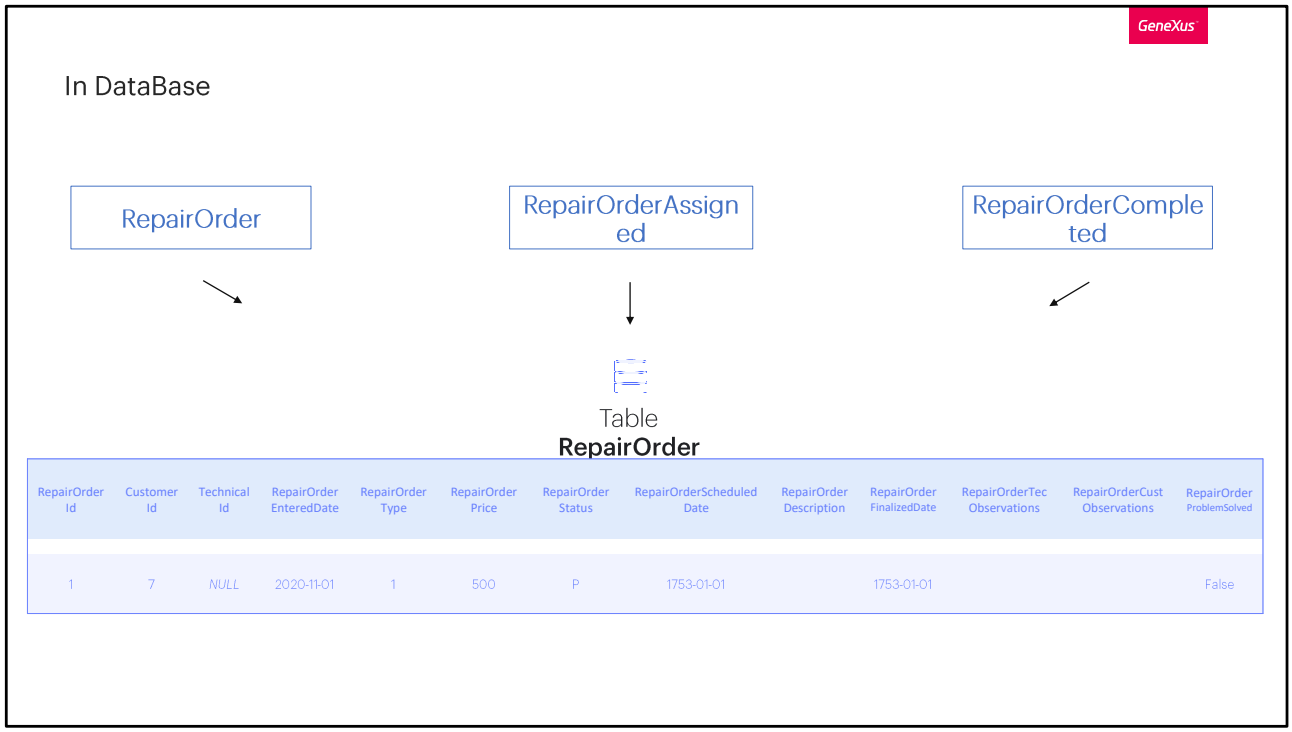


A partir desta transação (RepairOrder), um operador recebe a reclamação do cliente e preenche todos os campos. Vemos que é atribuído automaticamente o valor “P” de pendente, por tê-lo definido na regra default.

Em algum momento posterior, o coordenador entra nesta outra transação (RepairOrderAssigned), pesquisa a ordem e completa os dados necessários. Alterando automaticamente o valor que registra o status da ordem para “A” de atribuído.

Por último, após a realização da tarefa e recebida a parte técnica, o administrativo a controlará a partir da terceira transação criada (RepairOrderCompleted), preenchendo os campos correspondentes, e será alterado automaticamente o status, atribuindo o valor “F”, de finalizada.

Vemos que, desta forma, fica muito mais simples, claro e intuitivo para os usuários. E ajuda a que a aplicação seja escalável, para enfrentar futuras mudanças que possam existir no procedimento de inserção das ordens.



No nível da base de dados, a partir dos níveis principais destas três transações será gerada uma única tabela física, já que, como sabemos, GeneXus desenha a base de dados seguindo critérios de normalização.

A tabela gerada conterá os atributos que resultam da união das três transações.
Quando inserimos um registro a partir da primeira transação, o que acontece então na tabela de nossa base de dados, com os atributos aos quais ainda não atribuímos nenhum valor? Com que valor ficarão?

Serão atribuídos valores vazios por padrão: “0” em campos numéricos, string vazia nos campos do tipo character, uma data por padrão que representa o vazio para os do tipo Date, e false para os atributos booleanos.

As transações paralelas nos oferecem uma maneira limpa de manter na estrutura de cada transação estritamente as informações com as quais se deseja trabalhar dentro desse programa, independentemente das informações da mesma tabela que possam ser tratadas em outra das transações

Há uma grande diversidade de realidades que podem nos levar à

necessidade de utilizar este tipo de transações, que não entraremos em detalhes neste vídeo. Convidamos você a consultar em nossa WIKI para mais detalhes sobre este tema.

GeneXus™

training.genexus.com
wiki.genexus.com