Definindo a arquitetura para aplicativos de Service Broker

Roger Wolter - Microsoft Corporation

Publicado em: 15 de janeiro de 2007

Aplica-se a:

- Microsoft SQL Server 2005 Service Broker

Resumo:

Nos últimos anos, muitos artigos, livros, Web casts e apresentações discutiram o modo de trabalho do Microsoft SQL Server 2005 Service Broker (SSB) e como construir aplicativos tendo-o como base. Este artigo sobe um degrau do "como" para discutir o "por quê" usar o Service Broker e quais decisões têm de ser tomadas para projetar e construir, de modo bem-sucedido, um aplicativo com o Service Broker.

Nesta página

Arquitetura Service Broker

Quando se fala em arquitetura de software, é sempre válido pensar quais tarefas são realizadas pela arquitetura de edifícios, pois existem muitos paralelos entre a arquitetura de edifícios e a de aplicativos.

A arte e a ciência do projeto

Ao projetar um edifício, o terreno que receberá a construção é, em geral, tomado como certo. Você poderá iniciar com o projeto do edifício e, depois, encontrar um local para construí-lo, mas isso é raro acontecer. O terreno impões limites ao projeto. O tamanho define as dimensões máximas da fundação. As condições do solo (ou as leis de zoneamento) podem limitar a altura máxima. A inclinação pode determinar o tipo de edificação a ser construído. Os edifícios existentes à volta do terreno poderão exercer influências no projeto e nos materiais usados. No caso de um aplicativo de SSB, o terreno seria o Microsoft Windows e o Microsoft SQL Server. Como o SSB é um recurso de banco de dados que pode ser acessado de várias formas, de muitas plataformas diferentes (basicamente, qualquer produto que possa ser conectado a um SQL Server), é impossível construir um Service Broker sem o SQL Server e este só pode ser executado em um Windows.

Ao desenvolver um projeto, as maiores restrições referem-se às exigências do cliente. Será um edifício residencial ou para escritórios? Precisará de 2 ou de 200 banheiros? Trata-se de um banco ou de uma usina de aço? Nenhum arquiteto começaria a construir, sem conhecer estas exigências primordiais. Algumas decisões podem ficar para mais tarde: cor da pintura, os móveis, mas, todas as decisões necessárias para cada fase da construção devem ser tomadas e aprovadas antes do início da construção. Se o cliente decide que, na verdade, deseja construir uma casa com 3 dormitórios e duas entradas, depois de concluídos 50 andares, de acordo com o pedido original de um edifício de escritórios, isso custará muito caro. Têm-se uma idéia formada da flexibilidade do software e que alterações importantes nos requisitos podem ser aceitas a qualquer tempo. Ainda que o custo de alterar os requisitos de software não seja tão alto quanto mudar o projeto de um edifício construído pela metade, existem custos e as alterações são dolorosas; portanto, obter os requisitos corretos é

de vital importância.

A próxima etapa do projeto de construção trata da escolha das ferramentas e dos materiais a serem usados. Em alguns casos, existe muita liberdade nessas escolhas mas, se você estiver construindo uma torre de escritórios de 30 andares, estruturá-la com sarrafos de 2"x 4" não é uma opção. As exigências do cliente também podem limitar suas opções. Se ele ama alvenaria, você não poderá usar argila, ainda que este seja o melhor material para o trabalho. A limitação básica é, quase sempre, a disponibilidade financeira do cliente. Algumas decisões serão tomadas pelo cliente mas, outras, ficarão a cargo dos critérios do arquiteto. O cliente poderá discutir muito sobre o telhado, madeira ou cerâmica, mas deixará a escolha do material da tubulação, cobre ou PVC, para o arquiteto. Em outras oportunidades, quando as exigências do cliente são insensatas ou impossíveis, é responsabilidade do arquiteto convencer o cliente do contrário.

Por exemplo, independentemente do quanto o cliente possa gostar de tubos de chumbo, você não pode permitir isso. Os mesmos tipos de escolhas e compensações aplicam-se ao projeto de um aplicativo de Service Broker. Primeiramente, decidir se o Service Broker deve ou não ser usado. Embora seja a melhor coisa depois do pão em fatias, o Service Broker não é a resposta para tudo. O Service Broker oferece um grande número de recursos e opções. É preciso combinar as exigências do cliente, as capacidades do Service Broker, os limites de projeto e o próprio discernimento para determinar quais recursos devem ser usados e como usá-los.

Depois de determinados os limites, as exigências e os materiais, o arquiteto pode projetar um edifício que atenda as exigências, atenha-se aos limites (inclusive prazo e recursos disponíveis) e satisfaça o senso estético e o orgulho profissional do arquiteto e do cliente. Um arquiteto profissional não projetaria um prédio feio, mesmo se esse for exatamente o desejo do cliente, porque afetaria negativamente a reputação e a competência conhecidas do arquiteto. Da mesma forma, um arquiteto de software não deveria nunca projetar um aplicativo que não funcionasse, apenas porque o cliente assim quis. Se aquilo que o cliente deseja não vai funcionar, é preciso que ele seja informado, e se isso significar que ele pode procurar outra pessoa para projetar o aplicativo, no mínimo o seu nome não ficará associado a um fracasso.

Depois de ganhar um projeto, as responsabilidades do arquiteto estão apenas começando. O arquiteto deve acompanhar a construção atentamente, estar pronto para inserir alterações de projeto na medida em que as circunstâncias se modifiquem e ser responsável por assegurar que a construção satisfaça os critérios do projeto. Os operários não podem mudar a posição das paredes de moto próprio, ainda que isso resolvesse um problema efetivo. De modo similar, um arquiteto de software não pode simplesmente jogar o projeto na parede dos desenvolvedores. O arquiteto de software precisa conduzir o projeto pelas fases de implementação, teste e implantação.

Inicio da pagina

Quando construir com o Service Broker?

Service Broker é uma plataforma para construção de aplicativos de banco de dados distribuídos, confiáveis, de baixo acoplamento. Faz parte de todas as edições do SQL Server 2005 e, dessa forma, pode ser usado em qualquer de seus aplicativos. Se desejar mais informações sobre o Service Broker, consulte este artigo na Biblioteca MSDN[1]. Se quiser um volume maior de informações, recomendo o documento The Rational Guide to SQL Server 2005 Service Broker, disponível aqui[2].

Embora goste de pensar que todos os programas sejam aplicativos de Service Broker em potencial, na verdade, essa idéia só é válida para uma grande parte. Arquitetos decidem usar um determinado material construtivo fazendo a correspondência entre características e requisitos. A seguir, algumas orientações sobre como fazer isso com o Service Broker.

Filas

Um dos recursos fundamentais do Service Broker é a fila como um objeto de banco de dados nativo. A maior parte dos aplicativos de banco de dados de grande porte, com os quais já trabalhei, usa uma ou mais tabelas como filas. Os aplicativos colocam em uma tabela os dados que não querem manipular, no momento; depois, o aplicativo original ou outro fará a leitura da fila e a manipulação do que precisa ser feito. Um bom exemplo são os aplicativos de negociação de ações. As negociações precisam acontecer com tempo de resposta de subsegundo ou perde-se dinheiro e violam-se as

regras da SEC, mas todo o trabalho para concluir a operação de transferência das ações, converter moeda, faturar os clientes, pagar as comissões, etc., pode ser feito mais tarde. Esse trabalho administrativo pode ser processado colocando as informações necessárias em fila. O aplicativo de negociação fica, assim, livre para manipular o próximo negócio e a liquidação acontecerá quando o sistema tiver alguns ciclos de sobra. É crítico assegurar que, depois de confirmada a transação, as informações de liquidação não se percam pois a negociação não estará concluída até que a liquidação seja feita. É por isso que a fila precisa estar em um banco de dados. Se as informações de liquidação forem colocadas em uma fila ou em um arquivo de memória, uma queda de sistema poderia resultar em negociação perdida. A fila também deve ser transacional, porque a liquidação deve ser concluída ou devolvida e reiniciada. Provavelmente, um sistema de liquidação verdadeiro usaria muitas filas para que cada parte da atividade de liquidação pudesse continuar em seu ritmo e em paralelo.

Assim, filas no banco de dados são uma boa coisa. Mas, por que não usar apenas tabelas como filas, em lugar de inventar um novo objeto de banco de dados? É que é difícil usar tabelas como filas. Simultaneidade, escalonamento de lock, deadlocks, poison messages, etc., são problemas difíceis de resolver. A equipe do Service Broker levou anos para propor uma fila confiável, de alto desempenho, para que bastasse a você chamar CREATE QUEUE para desfrutar de todo esse desenvolvimento em seu aplicativo. A lógica de por uma mensagem em fila, tirá-la da fila e excluí-la ao final do trabalho foi incorporada aos novos comandos TSQL (SEND e RECEIVE) e, assim sendo, você não precisará nem mesmo escrever essa lógica.

As filas do SSB podem ser usadas para quase todas as atividades assíncronas necessárias ao aplicativo de banco de dados. Um aplicativo de entrada de pedidos pode precisar fazer a expedição e o faturamento de modo assíncrono, para aprimorar os tempos de resposta do pedido. Um acionador que precise processar um volume significativo pode usar o SSB para fazer o processamento de modo assíncrono, para não afetar as atualizações da tabela original. Um procedimento armazenado pode precisar chamar vários outros procedimentos armazenados em paralelo. A lista não pára.

Aplica-se o modelo de fila assíncrona a um grande número de aplicativos. Praticamente quase todo aplicativo escalável, de grande porte, usa filas em algum momento. O Windows faz quase todo o seu trabalho de Entrada e Saída (ES) por meio de filas. O IIS recebe as mensagens HTTP em filas. Todos os comandos TSQL do SQL Server são executados a partir de uma fila.

Agora, a pergunta óbvia é a seguinte: Se as filas do Service Broker são tão boas, por que não são usadas por esses aplicativos? A resposta mais sucinta é que as filas do Service Broker são persistentes. Colocar mensagens persistentes em uma fila do SSB - ou removê-las e processá-las - envolve uma gravação no log de transação do SQL Server. Isso é uma boa coisa quando se liquida ou fatura negociações de bolsa ou se você quiser estar certo de que a negociação ou o pedido não se perderá se faltar energia. Mas, se faltar energia em Windows ou IIS, as conexões de entrada caem e o que estava em andamento desaparece. Neste ponto, as mensagens em fila são inúteis pois os aplicativos que esperam uma resposta já se foram e, assim, as mensagens persistentes em fila representam uma perda de recursos e uma desnecessária redução de atividade. É muito bom pensar sobre um modo de consulta confiável no qual suas consultas persistem de modo que a resposta volte, ainda que, no meio tempo, ocorra uma interrupção do cliente ou do servidor (e vários clientes usam o Service Broker para tal). Mas, os milhares de aplicativos existentes não são construídos para aproveitar as mensagens persistentes.

Assim, o melhor desempenho ou o "sweet spot" do uso das filas do SSB ocorre com os aplicativos de banco de dados que devem fazer tudo de modo confiável e assíncrono. Se a ação tiver de acontecer de modo síncrono, uma chamada de função normal, COM ou RPC será a tecnologia correta. Se a ação tiver de ser iniciada de modo assíncrono, mas se o seu desaparecimento não causar problemas caso o aplicativo seja desativado, qualquer tipo de fila em memória terá melhor desempenho. Além disso, o Service Broker representa um recurso do SQL Server e, provavelmente, não será lógico usá-lo a não ser que haja um banco de dados SQL Server por perto. Existem aplicativos para os quais filas persistentes e confiáveis são tão importantes que a adição de um banco de dados do SQL Server justifica-se apenas para o estabelecimento das filas. Todavia, em geral, o SSB é uma opção melhor para aplicativos de banco de dados. Vale também ressaltar que, como o acesso ao SSB é feito pelos comandos TSQL, qualquer plataforma, idioma e aplicativo que possa conectar-se com um banco de dados do SQL Server por meio do SQL Client, OLEDB, ODBC, etc., poderá enviar/receber mensagens de uma fila do Service Broker. Esse fato facilita integrar aplicativos em muitas plataformas de modo confiável, transacional e assíncrono.

Diálogos

Um dos recursos mais exclusivos do Service Broker é o diálogo. Define-se diálogo como um fluxo de mensagens confiável, ordenado, persistente e bidirecional. Na maioria dos sistemas de mensagens/enfileiramento, a primitiva da troca de mensagens é a mensagem; cada uma delas é independente e não relativa a outras mensagens no nível de troca de mensagens. Se o aplicativo quiser estabelecer relacionamentos entre mensagens, vinculando uma requisição a uma resposta, por exemplo, o aplicativo será responsável pelo rastreamento.

No SSB, o diálogo é a primitiva de troca de mensagens. As mensagens enviadas em um diálogo são processadas na ordem em que foram enviadas, mesmo se tiverem sido enviadas em transações diferentes, de diferentes aplicativos. Os diálogos são bidirecionais e, dessa forma, os relacionamentos requisição-resposta são automaticamente rastreados. Os diálogos são persistentes e, deste modo, o diálogo permanece ativo mesmo quando as duas extremidades do diálogo desaparecem, o banco de dados é desativado, transferido para outro servidor ou o que for. Isso significa que os diálogos podem ser usados para implementar transações de negócio conversacional de longa execução, que duram meses ou anos. Por exemplo, o processamento de um pedido envolve, de modo geral, uma troca de mensagens de longa duração entre o comprador e o fornecedor, na medida em que os preços são negociados, as datas de entrega estabelecidas, os posicionamentos informados, as entregas confirmadas e o pagamento realizado. Toda essa negociação pode ser um único diálogo do Service Broker que pode durar meses. A Figura 1 ilustra uma conversação em diálogo de longa duração.

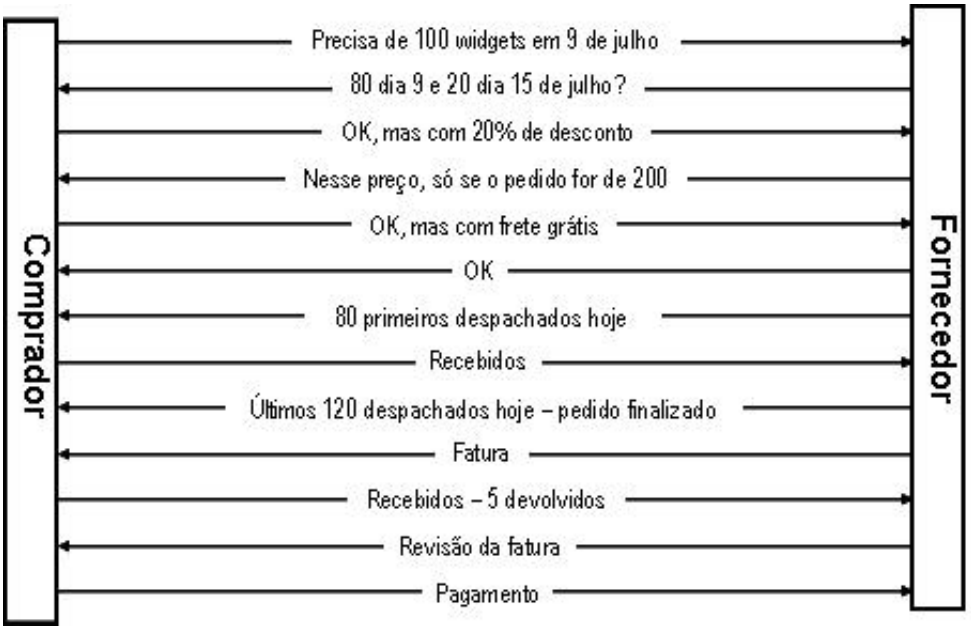

Figura 1. Conversação em diálogo de um pedido

Grupos de conversação

Diálogos existem em grupos de conversação. Define-se grupo de conversação como a unidade de bloqueio de uma fila do Service Broker. Sempre que um comando RECEIVE ou SEND for executado, o grupo de conversação que contém o diálogo usado para o RECEIVE ou SEND ficará bloqueado. Se as mensagens correlatas forem recebidas por diferentes segmentos do aplicativo, o estado dos aplicativos pode ficar corrompido devido a alterações simultâneas ou alterações processadas fora de ordem e este é um dos problemas mais difíceis dos aplicativos assíncronos de enfileiramento. Por exemplo, uma linha da ordem pode ser processada antes do seu cabeçalho, provocando a rejeição dessa linha da ordem. Em muitos casos, para resolver esse problema, basta fazer com que o aplicativo tenha uma única linha de execução mas isso, obviamente, limita a escalabilidade e o desempenho. Com o Service Broker, o aplicativo coloca todos os diálogos relativos a uma determinada transação do negócio em um único grupo de conversação e, assim, apenas um segmento estará processando essa transação num determinado momento. Por exemplo, um aplicativo de entrada de pedidos colocaria todos os diálogos associados a um determinado pedido no mesmo grupo de conversação para que quando centenas de segmentos estiverem processando centenas de mensagens de pedidos, simultaneamente, as mensagens de qualquer pedido específico só sejam processadas em um segmento de cada vez. Este procedimento permitirá escrever um

aplicativo com uma única linha de execução e fazer com que o Service Broker administre a execução de centenas de segmentos, simultaneamente.

Provavelmente, uma fila de multileitor é o sistema de balanceamento de carga mais eficiente ora disponível. Os leitores de filas, quer estejam no banco de dados ou em servidores remotos, abrem uma conexão com o banco de dados e iniciam o recebimento e o processamento das mensagens. Depois de processada cada mensagem, o aplicativo leitor de filas recebe outra. Dessa forma, cada leitor de filas recebe o volume de trabalho que pode processar. Se um dos leitores reduzir sua velocidade por qualquer motivo, acionará os comandos TSQL RECEIVE com menor freqüência e os outros leitores ficarão livres para assumir a posição deste mais lento. Se um dos leitores parar ou quebrar, a transação RECEIVE da mensagem que estava em processamento será devolvida e a mensagem aparecerá novamente na fila, para ser manipulada por outro leitor. Se a fila começar a ficar grande porque os leitores não estão dando conta do volume, você poderá começar um novo leitor que começará a processar as mensagens. Não é preciso reconfigurar: basta iniciar e interromper os leitores, conforme necessário. O bloqueio do grupo de conversação faz tudo isto possível. O aplicativo que faz o envio não conhece, nem se preocupa, com a quantidade de leitores de filas ou com o local em que estão sendo executados.

Ativação

O comando RECEIVE tira as mensagens da fila para processamento e essa é uma das questões fundamentais dos aplicativos assíncronos de enfileiramento. Isto significa que o aplicativo de recepção precisa estar em execução no momento em que uma mensagem chega na fila. Para tal fim, existem várias abordagens como a recepção de um serviço Windows que está sempre em execução ou de um procedimento armazenado na inicialização, cujo início é simultâneo ao do banco de dados. Estas são boas soluções, desde que as mensagens cheguem em fluxo constante, mas em muitos casos, o aplicativo de recepção usa recursos à toa, quando não há mensagens na fila, e impõe atrasos nas horas de pico.

O Service Broker oferece uma alternativa melhor, denominada ativação. Para usar a ativação, basta associar uma fila a um procedimento armazenado que sabe como manipular as mensagens dessa fila. Quando uma mensagem chega na fila, a lógica do SSB que a manipula confirma as verificações para saber se há uma cópia do procedimento armazenado em execução. Se houver, a confirmação continua; caso contrário, a lógica de ativação iniciará uma. Isto é melhor do que os acionadores que alguns sistemas de mensagens oferecem, pois uma nova cópia será iniciada apenas quando necessária. A ativação presume que o procedimento armazenado continuará a ler as mensagens até que a fila esteja vazia; já os acionadores iniciarão um novo leitor para cada mensagem. Se uma fila recebe 1.000 mensagens por segundo, a ativação iniciará 1 leitor; os acionadores iniciariam 1.000. A ativação também observa o crescimento da fila caso as mensagens estejam chegando mais rápido do que a velocidade de processamento do procedimento armazenado. Se a fila estiver crescendo, a ativação iniciará novas cópias do procedimento armazenado até que a fila pare de crescer. Quando estiver vazia, os procedimentos armazenados serão encerrados porque não haverá mais trabalho a fazer. Dessa forma, a ativação garante a existência de recursos suficientes, dedicados ao processamento das mensagens da fila, mas não mais do que o necessário.

A ativação tem outro benefício colateral bastante útil. Permite executar um procedimento armazenado pelo envio de mensagem para uma fila. O procedimento armazenado é executado no segundo plano de uma execução, transação e contexto de segurança, diferente do procedimento armazenado que enviou a mensagem. É isto o que ativa triggers e procedimentos armazenados que iniciam vários outros procedimentos armazenados em paralelo. Como o procedimento ativado é executado em outro contexto de segurança, ele pode ter mais ou menos privilégios do que o chamador. E porque é executado em outra transação, os deadlocks ou falhas não afetam a transação original.

Por exemplo, trabalhei com um cliente que inseriu um registro de auditoria em uma tabela de logs ao final de cada transação. Em muitos casos, esta inserção causaria um deadlock ou timeout e toda a transação teria de ser desfeita resultando em frustração para o usuário. Eles trocaram a lógica de auditoria do comando SEND para que a mensagem seja enviada para uma fila do SSB e, agora, qualquer problema com a tabela de auditoria não ocasiona falha na transação original. Outro cliente escreveu um procedimento armazenado simples, que recebe a mensagem de uma fila, chama EXEC no conteúdo da mensagem e devolve o resultado para o originador, no mesmo diálogo. Agora, ele pode executar comandos TSQL no segundo plano de qualquer sistema do centro de dados, com o

simples envio de uma mensagem. A segurança do SSB faz com que isto seja mais seguro do que permitir o logon do administrador no servidor e entrega confiável do SSB significa que comandos e respostas nunca se perdem.

Troca confiável de mensagens

Já vimos o valor do Service Broker para projetos de aplicativos assíncronos de enfileiramento. Os diálogos, o bloqueio do grupo de conversação e a ativação fazem do Service Broker uma plataforma exclusiva para a construção de serviços de banco de dados de baixo acoplamento. Depois que você conhecer todos os aplicativos, robustos, que podem ser escritos com as filas do SSB, não demorará muito para ter idéias que precisarão colocar mensagens em uma fila, em outro banco de dados. Se o outro banco de dados for executado em um servidor diferente, as garantias de confiabilidade do SSB exigirão que a mensagem seja enviada de modo confiável, ou seja, o banco de dados remoto confirma o recebimento da mensagem e o Service Broker local continuará enviando-a até receber uma confirmação. Desta forma, o aplicativo pode acionar do comando SEND para enviar uma mensagem a uma fila remota e ter as mesmas garantias de confiabilidade, como se estivesse enviando-a para uma fila local. Na verdade, o aplicativo não sabe, nem se preocupa, com o processamento, local ou remoto, da mensagem que envia. Escrever aplicativos assíncronos, baseado em filas e distribuídos com o Service Broker não é diferente do trabalho de escrever aplicativos locais. Isto significa que você poderá começar com um aplicativo local e torná-lo distribuído, na medida em que houver modificações no volume de processamento ou nas necessidades do negócio.

Infelizmente, incluir troca de mensagens confiável no Service Broker tem ocasionado muita confusão. Assim que as pessoas vêem a expressão 'troca de mensagens confiável', pensam em MSMQ ou MQ Series. Apesar de o SSB ter muitos dos mesmos recursos, ele é basicamente uma plataforma para a construção dos aplicativos de banco de dados distribuídos. Por exemplo, é extremamente fácil para um procedimento armazenado iniciar um procedimento armazenado de modo confiável e assíncrono em um banco de dados remoto com o Service Broker; entretanto, fazer o mesmo com o MSMQ seria bastante difícil. (No meu blog[3] e no Jornal de Arquitetura, No.8[4] teço outras considerações sobre essas questões.)

Como o Service Broker estabelece comunicações confiáveis entre filas de bancos de dados, a confiabilidade e a tolerância a falhas construídas no SQL Server aplicam-se automaticamente às mensagens do Service Broker. Todas as medidas adotadas por sua empresa para garantir a disponibilidade do banco de dados - clusters, SANs, logs de transações, backups, espelhamento de banco de dados, o que for - também funcionam para manter disponíveis as mensagens de SSB. Por exemplo, se estiver usando espelhamento de banco de dados para alta disponibilidade, se houver falha do primário para o secundário, haverá também falha de todas as mensagens para este último e as filas permanecerão transacionalmente consistentes com o resto dos dados. Além disso, o Service Broker é compatível com espelhamento e, desse modo, se houver falha do banco de dados primário para o secundário, os outros Service Brokers com os quais estiver se comunicando farão a detecção imediata dessa mudança e passarão a se comunicar com o banco de dados secundário.

Inicio da pagina

Projetando o aplicativo de Service Broker

Depois de concluir a análise que decidirá se o Service Broker é a solução adequada para atender as exigências do seu cliente, podemos presumir que você estará construindo um aplicativo de banco de dados assíncrono, confiável. Se (a) não estiver preocupado com a confiabilidade, (b) suas atividades precisam ser síncronas e (c) não houver dados para armazenar, projetar uma solução Service Broker que atenda suas exigências será bem difícil. Você pode usar o Service Broker para implementar atividades síncronas ou adicionar um banco de dados ao seu aplicativo apenas para ter condições de usar o Service Broker, mas, em geral, isso só se justificaria se fosse premente ter a confiabilidade do SSB ou se outras partes do aplicativo já estivessem usando o SSB.

Esta seção discute uma série de decisões a serem analisadas ao projetar um aplicativo de Service Broker. Nem todas as decisões são necessárias para um determinado aplicativo, mas vale a pena considerá-las, todas, para verificar se não falta nenhum elemento importante. Essas decisões são apresentadas em uma determinada ordem; na verdade, entretanto, projetar é um processo bastante cíclico e, freqüentemente, será preciso rever decisões anteriores, à medida que se alcançam fases posteriores do projeto. É meu hábito colocar as etapas na ordem em que as executo, mas você

poderá encontrar uma outra ordem, que funcione melhor para você.

Identificação dos serviços SSB

O Service Broker habilita a comunicação assíncrona entre os serviços; assim, a primeira coisa a se decidir é o que esses serviços devem ser. Em muitos casos, os serviços ficam óbvios a partir da definição do problema. Se estiver usando o Service Broker para registrar os eventos de CREATE TABLE, por exemplo, os serviços serão o comando CREATE TABLE e o seu código de registro. O código do evento faz parte do SQL Server e, assim, você terá apenas um serviço para projetar.

A maior parte dos diálogos do Service Broker envolvem três elementos de código:

- Um **iniciador** que inicia um diálogo e envia uma mensagem. Em geral, o código iniciador não é o serviço especificado no parâmetro FROM SERVICE do comando BEGIN DIALOG.
- Um **serviço de destino** que recebe essa mensagem, executa algum trabalho e envia uma resposta.
- Um **serviço de resposta** que manipula a mensagem de resposta. Este é o serviço especificado no parâmetro FROM SERVICE do comando BEGIN DIALOG.

Pode parecer estranho, à primeira vista, que o iniciador não receba a resposta, mas esta é a natureza de um aplicativo assíncrono. Seria síncrono se o iniciador tivesse esperado pela resposta (na verdade, é assim que se implementam requisições síncronas em um sistema de mensagens assíncrono). Em um sistema assíncrono, o iniciador dá partida na atividade assíncrona e continua para fazer outra coisa. Quando a resposta voltar, ele poderá estar processando outra requisição ou até nem existir mais. Se a resposta for manipulada por outro serviço, este será ativado quando chegar a mensagem de resposta. O mesmo serviço pode manipular respostas de vários iniciadores. Como exemplo, temos um aplicativo de entrada de pedidos que inicia um diálogo com um serviço de expedição para despachar um item solicitado. Assim que a mensagem de embarque é enviada, o programa de entrada de pedidos poderá tratar de outros pedidos. Quando o serviço de despacho responde com uma confirmação de embarque (talvez dias ou semanas mais tarde), um serviço no banco de dados de entrada de pedidos recebe a mensagem, atualiza o status do pedido para **Despachado**, e envia um e-mail ao cliente.

Ainda que o ponto de destino não emita uma mensagem de resposta, haverá um serviço mínimo, no lado do iniciador, para manipular as mensagens de controle do Service Broker como mensagens de erro e de EndDialog. Devido à natureza assíncrona do Service Broker, um comando SEND bem-sucedido significa apenas que a mensagem foi colocada em fila (fila de destino ou sys.transmission_queue). Um comando SEND para um serviço remoto que não existe será bem-sucedido porque o Service Broker não pode dizer a diferença entre um serviço que não existe e outro que não esteja em execução, no momento. É por isso que o comando FROM SERVICE é um parâmetro exigido do comando BEGIN DIALOG. A Figura 2 descreve uma interação simples do serviço Service Broker na qual um serviço de produção envia uma mensagem **AddItem** para um serviço de estoque, para incluir um novo item ao estoque.

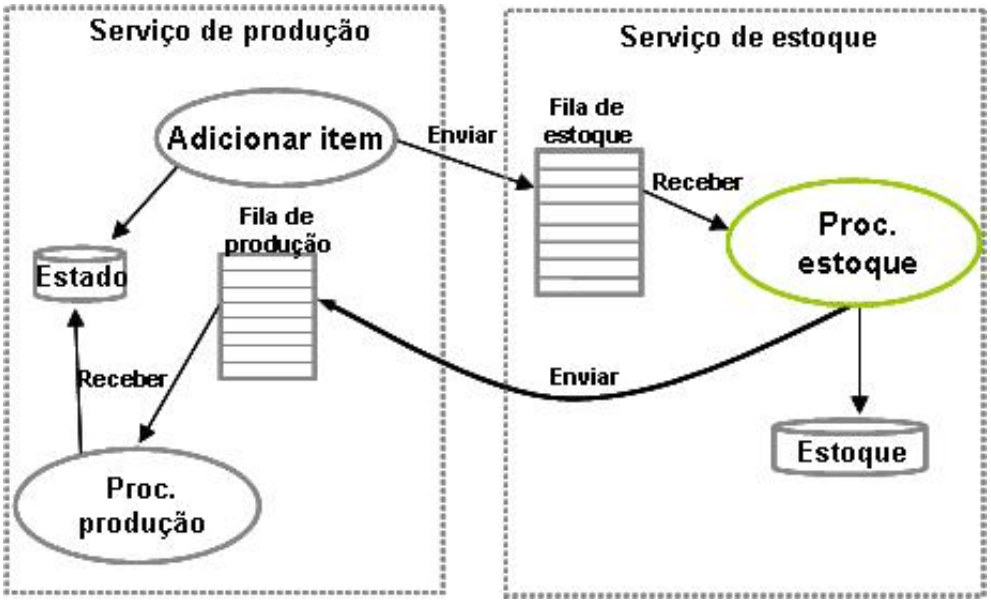

Figura 2. Interação de serviço típica

Em um aplicativo mais completo, existem muitas outras funções envolvidas, algumas síncronas, outras assíncronas. A escolha das funções que executarão os serviços do Service Broker é importante. As primeiras candidatas dos serviços SSB são funções que não precisam ser concluídas antes do final da lógica principal. Alguns exemplos são a expedição e o faturamento de pedidos, as liquidações de negociações em bolsa e as reservas de locação de veículos para um itinerário de viagem. Em todos esses casos, a transação original pode ter sido concluída muito antes da volta da resposta e o status da resposta será devolvido ao usuário por meio de comunicações como e-mail ou por meio de um status que poderá ser consultado pelo usuário, mais tarde.

Se houver uma função que precise ser concluída antes do controle voltar para o usuário, ainda faria sentido usar um serviço do Service Broker se dois ou mais serviços puderem ser executados em paralelo. O aplicativo de call-center é um exemplo clássico de cenário com o qual já trabalhei. Os clentes do chamado foram identificados pelo ID do chamador e todos os registros do cliente, de todos os sistemas internos, foram recuperados para poderem ser exibidos ao representante do serviço quando a chamada for atendida. O problema era que esse procedimento envolvia consultas remotas a sete sistemas e, algumas vezes, demorava tanto, que o cliente desistia antes de a chamada ser atendida. Fizemos com que isso funcionasse iniciando as sete consultas em paralelo e os resultados eram devolvidos quando todas estivessem de volta. Esse procedimento reduziu o tempo de resposta de 5 segundos para 1 segundo. Observe-se que esta é uma abordagem "anti-escalabilidade". Em lugar de uma thread, foram usadas oito threads do banco de dados e, desse modo, nosso tempo de resposta aprimorado saiu pelo preço de uma escalabilidade menor; todavia, o efeito não foi estrondoso mas tivemos de fazê-lo para atender as exigências de resposta. Outro caso de uso assíncrono poderia ser um site operando com um aplicativo de financiamentos pelo qual pergunta-se ao usuário qual o valor do empréstimo desejado; em seguida, vários cálculos da tabela de amortização são realizados em segundo plano e, enquanto isso, outras perguntas são feitas ao cliente. Quando o cliente estiver pronto para analisar as opções de empréstimo, os resultados estarão prontos e, assim, o cliente pensará que os cálculos são feitos instantaneamente. Sem muito esforço, por certo você criará dúzias de cenários similares. É por isso que as atividades assíncronas são tão usadas em aplicativos de alto desempenho.

O ponto final da escolha dos serviços é que, em geral, é um erro definir serviços SSB muito refinados. As mensagens do SSB são escritas no banco de dados e, assim, se você projetar um serviço que faz dezenas de chamadas para outros serviços executarem, você poderá descobrir que a sobrecarga do banco de dados é superior ao tempo de processamento atual. O serviço do Service Broker deverá realizar um item de trabalho razoavelmente significativo para justificar a sobrecarga na manipulação de mensagens. Existem exceções a isso se confiabilidade, execução remota ou isolamento de contexto de segurança fizerem com que um serviço do Service Broker seja adequado, ainda que muito pequeno. Na verdade, isso não é muito diferente do DCOM em que poucas chamadas ao DCOM com muitos dados é muito mais eficiente do que muitas chamadas de DCOM com poucos dados, para realizar o mesmo trabalho.

Definir diálogos

Depois de ter definido os serviços, será preciso definir os diálogos a serem usados na comunicação. Basicamente, esta fase consiste em decidir quais mensagens são necessárias para a comunicação e quais os serviços a serem criados para enviá-las. Os diálogos são mais complexos do que as semânticas normais de pedido/resposta com as quais você está acostumado para os serviços Web ou DCOM, pois os diálogos podem ser conversações prolixas que envolvem muitas mensagens, nas duas direções, e podem durar meses ou anos. O diálogo deve modelar toda a transação do negócio. Por exemplo, concluir um pedido pode envolver as seguintes etapas: envio do pedido, confirmação do seu recebimento, negociação de preço e de datas de entrega, informações de status, notificação de embarque, confirmação do recebimento e faturamento. Esta transação pode continuar durante meses e envolver a troca de centenas de mensagens. Com o Service Broker, essa conversação toda poderia ser modelada em um único diálogo. O diálogo fará a correlação e ordenará as mensagens no decorrer do tempo, de tal modo, que todas as mensagens relativas a esse pedido terão o mesmo tipo de manipulação de diálogo e ID do grupo de conversação. Se você armazenar o ID do grupo de conversação na tabela de cabeçalhos do pedido, o aplicativo identificará facilmente a qual PO refere-se a mensagem. Como os diálogos são persistentes, o ID será sempre o mesmo, ainda que o diálogo dure anos.

Os diálogos são razoavelmente baratos, mas não são grátis. Iniciar ou encerrar um diálogo pode envolver gravação em banco de dados e a uma mensagem desse BD para que as duas extremidades saibam que o diálogo está se encerrando. Por essa razão, quando serviços são envolvidos em uma transação do negócio, os diálogos usados para a comunicação com outros serviços devem ficar disponíveis até que o serviço os encerre. Alguns aplicativos de SSB, de desempenho muito elevado, reusam os diálogos para várias transações do negócio. Isso pode melhorar o desempenho de modo significativo, mas se não for feito do modo correto, poderá trazer problemas de bloqueio. (No meu blog[5] há uma discussão das questões envolvidas.) A reciclagem dos diálogos pode aprimorar o desempenho, mas o preço desse desempenho será maior complexidade. Se o seu aplicativo for simples ou se o desempenho máximo for um requisito-chave, você precisará procurar fazer reciclagem de diálogos; mas, em geral, é melhor limitar a vida útil do diálogo a uma única transação do negócio, a não ser que descubra estar precisando de um aumento de desempenho.

Nota Usei muitas vezes o termo transação do negócio sem defini-lo claramente. Para os objetivos deste artigo, transação do negócio representa uma atividade completa no nível do negócio. No exemplo do pedido, a transação do negócio estava processando o pedido, que durou muitos dias, e envolveu dezenas de transações de banco de dados. Outro exemplo é a reserva de uma viagem em um site especializado. A transação do negócio para reserva da viagem envolve os sistemas de reservas do hotel e do aluguel do veículo, os sistemas da companhia aérea, do faturamento, de um banco ou cartão de crédito e, possivelmente, vários outros sistemas. Há muitas transações de banco de dados em muitos bancos de dados envolvidas na reserva de uma viagem.

Os diálogos impõem a ordenação das mensagens no diálogo. Essa ordenação é imposta para todas as transações a partir de vários e diferentes serviços. Sobrevive às reinicializações e falhas do banco de dados. Este recurso é muito poderoso e permite que a lógica do aplicativo confie na ordem de envio da mensagem. Por exemplo, um aplicativo do SSB não precisa cuidar de uma linha do pedido que chegue antes do respectivo cabeçalho, se estiverem no mesmo diálogo. Os diálogos também podem resolver os tipos de dados misturados na mensagem. Nos aplicativos de serviços Web, os dados binários incorporados em um documento XML representam um dos problemas de mais difícil solução. Por exemplo, uma mensagem de funcionário pode ser um documento XML com foto, impressão digital ou diploma enviado como dados binários incorporados. Com o Service Broker, você poderá enviá-los em mensagens separadas. No momento em que o aplicativo recebe os documentos XML, poderá recebê-los no mesmo diálogo e certificar-se de que as mensagens chegariam na ordem correta e no mesmo segmento, pois a ordenação e o bloqueio do SSB se encarregariam disso.

A ordenação é um recurso poderoso mas, o seu projeto deve considerar que as mensagens do diálogo serão sempre processadas em ordem. Uma das perguntas mais comuns que me fazem refere-se a desenvolvedores que abrem um diálogo e, em seguida, começam a mandar muitas mensagens nesse diálogo. Definem a ativação para iniciar muitos leitores de filas mas descobrem que apenas um deles está processando as mensagens. Para garantir a ordem das mensagens do diálogo, o Service Broker deve usar bloqueios de grupo de conversação para assegurar que apenas uma transação de banco de dados, de cada vez, possa receber as mensagens de um determinado diálogo. Se quiser

usar recursos multisegmentados do SSB, você precisará ter, no mínimo, número igual de diálogos ativos e de segmentos. (Na verdade, deveria ter dito: número igual de grupos de conversação ativos e de segmentos, pois o que está bloqueado é o grupo de conversação. Se em três grupos de conversação houver 10 diálogos ativos, apenas três segmentos, de cada vez, poderão receber mensagens.)

As decisões tomadas sobre quais mensagens serão enviadas em cada serviço serão usadas para criar o CONTRACT do diálogo. Ao iniciar um diálogo, você especifica qual contrato o Service Broker usará para determinar quais mensagens podem ser enviadas no diálogo.

Definir grupos de conversação

Em muitos casos, diálogos são entidades bem independentes, mas alguns aplicativos usarão vários diálogos para concluir uma transação do negócio. Já falamos sobre um sistema de entrada de pedidos no qual o serviço de entrada de pedidos se comunica com os serviços de expedição, estoque, limite de crédito, CRM e de faturamento para concluir um pedido. Em geral, o serviço de entrada de pedidos iniciaria diálogos com cada um desses serviços em paralelo, para otimizar a eficiência do processamento. Como os serviços podem mandar mensagens de retorno a qualquer tempo, em qualquer pedido, os diálogos de todos esses serviços devem ser colocados no mesmo grupo de conversação. Quando uma mensagem é recebida em qualquer desses diálogos, o bloqueio do grupo de conversação, compartilhado entre todos os diálogos do grupo, assegurará que nenhum outro segmento possa processar mensagens de nenhum dos diálogos desse grupo. Este procedimento evitará problemas como o processamento simultâneo em diferentes segmentos de uma mensagem de crédito OK e outra de status de estoque, provocando atualizações conflitantes no estado do pedido.

Quase sempre me fazem perguntas sobre aspectos de desempenho dos bloqueios do grupo de conversação. Intuitivamente, você acha que o bloqueio desses diálogos causará obstrução e lentidão do sistema. Na verdade, esses bloqueios podem aprimorar o desempenho. Se dois ou mais segmentos estiverem trabalhando no mesmo objeto de pedido, simultaneamente, precisarão serializar o acesso às linhas do banco de dados para evitar os conflitos de atualizações, ou seja, dois ou três segmentos serão bloqueados enquanto esperam o primeiro terminar. O bloqueio do grupo de conversação impedirá o acesso às mensagens de um determinado pedido, enquanto um dos segmentos do aplicativo estiver processando esse pedido. Isto significa que apenas um segmento estará envolvido, em lugar de haver vários segmentos bloqueando-se entre si. Os segmentos liberados por esse procedimento podem, então, ser usados para processar mensagens de outros pedidos, o que aprimora o desempenho geral. Assim, deduz-se que bloqueios de grupo de conversação não apenas simplificam a lógica do aplicativo, mas também aprimoram sua eficiência.

Definir tipos de mensagem

O processo que define o diálogo determina quais mensagens são necessárias para implementá-lo; desse modo, nesta etapa, vamos decidir qual será o conteúdo da mensagem. No que se refere ao Service Broker, uma mensagem é uma partição de dados binários de 2GB. O Service Broker fará todo o trabalho de desmontagem/montagem necessário para transporte da mensagem ao seu destino. Os contratos usados para definir o conteúdo de um diálogo contêm definições de tipo de mensagens. A definição mínima de um tipo de mensagem é apenas o nome do tipo, que o serviço pode usar para determinar a natureza da mensagem recebida. Se a mensagem for um documento XML, o Service Broker poderá, opcionalmente, verificar o conteúdo da mensagem para confirmar se é um XML bem-formado ou válido em um Esquema XML.

Para cada mensagem de um diálogo, será preciso definir o que conterá o corpo da mensagem. XML é o mais comum, porque faz com que o serviço seja mais flexível e de baixo acoplamento, mas há sobrecarga na análise do XML. Se o conteúdo da mensagem for binário, basta enviá-la como mensagem binária, por exemplo, imagens, música, programas, etc. Já vi muitos clientes usarem um objeto .NET serializado como corpo da mensagem. Obviamente, isto faz com que os serviços iniciador e de destino sejam acoplados, pois ambos devem ter a mesma versão do objeto; entretanto, é mais eficiente e fácil codificar, e isso é o que se faz, comumente. Se o corpo da mensagem for XML, deve-se definir um esquema específico, como parte do processo de projeto. Pode ser que você queira que o Service Broker valide o conteúdo em relação ao esquema, mas ter um faz você ter esta opção, e esquemas são uma forma unívoca de informar aos desenvolvedores qual é o formato da mensagem.

Usar um esquema para validar mensagens de entrada pode ser muito caro, pois cada mensagem será carregada em um analisador (parser) no momento da recepção. Se, depois, o serviço carregar a mensagem em seu próprio analisador para processá-la, cada mensagem será analisada duas vezes. Em geral, recomendo que o esquema de validação esteja ativo para desenvolvimento e testes unitários mas, depois, desativado para testes de integração e produção. A exceção a isso seria um serviço que recebesse mensagens de várias fontes não confiáveis e, assim, a sobrecarga representada pela análise extra se justificaria, pois as mensagens incorretas seriam rejeitadas antecipadamente.

Serviços de projeto

Na última etapa do projeto do aplicativo devemos definir os serviços que processam as mensagens do Service Broker. Embora este seja um trabalho de grande porte, não me estenderei muito, porque a maior parte do esforço está na lógica do negócio que atualmente processa o conteúdo da mensagem. Destacarei apenas alguns aspectos que devem ser considerados no projeto dos serviços.

Local do serviço

O serviço deveria ser executado em um procedimento armazenado ou como um aplicativo externo? Se for um procedimento armazenado, deveria ser CLR ou TSQL? Em muitos aplicativos de Service Broker o serviço realiza, basicamente, tarefas de banco de dados. Se o serviço realizar, basicamente, atualizações de banco de dados, e não um grande volume de tarefas que utilizem muito o processador, o serviço deveria ser um procedimento armazenado. Se realizar muitas entradas e saídas de banco de dados, mas também um volume significativo de processamento, deveria ser um procedimento armazenado CLR.

Os serviços que não realizam grande volume de tarefas de banco de dados - ou que executam trabalhos que exigem muito processamento ou muitas entradas e saídas de disco ou rede - devem ser executados, em geral, como aplicativos externos que conectam com o banco de dados para obter as mensagens. Tudo que um aplicativo tem a fazer para processar mensagens SSB é abrir uma conexão com o banco de dados. Isto quer dizer que os serviços que realizam grande volume de processamento ou E/S de rede podem ser executados em outra caixa e conectar-se com o servidor do banco de dados para obter mensagens e realizar outras tarefas de TSQL. Parte significativa da lógica do negócio pode ser executada dessa forma. Outro uso comum é a interface com serviços Web. Embora alguns destes possam ser feitos no SQL Server, as sobrecargas da rede e do processamento XML dos serviços Web representam um motivo a mais para fazer esse processamento em um servidor comum, em lugar de usar o servidor de banco de dados que é mais caro. Se o servidor externo falhar, todas as transações que foram abertas serão desfeitas (rollback). As mensagens voltam para a fila e, assim, se houver mais de um servidor nesse processamento, tudo continuará sem solução de continuidade. Se a fila começar a aumentar, você poderá conectar mais dos servidores comuns na rede para manipular a carga. No meu blog[6] há uma discussão mais ampla sobre o local em que deve ser executada a lógica do serviço.

Loop de processamento de mensagens

Quase todos os serviços do Service Broker são construídos à volta do mesmo loop de processamento de mensagens. Este loop é usado em vários exemplos e, por isso, não vou abordá-lo prolongadamente nesta oportunidade. Basicamente, o loop tem um comando RECEIVE que recebe uma ou mais mensagens, processa-as e, pelo comando SEND, encaminha quaisquer mensagens de saída para, depois, recomeçar o processo. A maioria dos exemplos têm um comando RECEIVE TOP (1) no topo do loop. Isto funciona para um código exemplo simples, mas não é necessariamente o mais eficiente a se fazer. Sem a cláusula TOP (1), o comando RECEIVE devolverá todas as mensagens para a fila de um único grupo de conversação.

Executar um comando RECEIVE e receber de volta muitas mensagens é mais eficiente do que recebê-las uma de cada vez; portanto, vale a pena considerar este fato em seu projeto. O motivo pelo qual quase nenhum dos exemplos exibe isso é porque são diálogos simples de pedido/resposta nos quais apenas uma mensagem é enviada em um determinado diálogo, de modo que, deixar de fora a cláusula TOP (1) não modificará muito o comportamento do aplicativo. Se o seu aplicativo envia muitas mensagens de uma só vez no mesmo diálogo (por exemplo, um aplicativo de logon), receber muitas mensagens por instrução RECEIVE aprimorará em muito a eficiência.

A outra coisa ruim que a maioria dos exemplos faz é confirmar a transação (commit) ao final de cada loop. Mais uma vez, isto simplificará a sua lógica; e, se o desempenho for adequado, este será o melhor design. Mas, escrever a confirmação no log de transações é, quase sempre, o fator limitante definitivo do desempenho e, assim, se realmente precisar do melhor desempenho possível, você fará a confirmação apenas depois de algumas idas e vindas no loop de processamento. Esse procedimento aprimorará o desempenho, mas poderá aumentar a latência, pois nenhuma resposta será enviada até que a transação seja confirmada. Em muitas operações assíncronas, a latência não é muito importante, portanto, esta é uma boa troca. A manipulação das transações devolvidas (rollback) é capciosa neste caso, pois você terá de voltar e processar as mensagens, uma de cada vez, para contornar a incorreta.

A Figura 3 mostra um loop típico de processamento de mensagens.

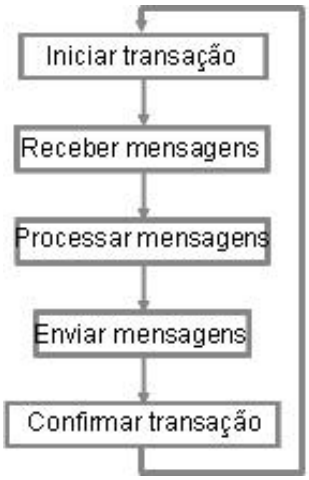

Figura 3. Fluxo da lógica do serviço

Manipulação de estados

Leia qualquer livro sobre Arquitetura Orientada a Serviços (SOA) e você verá que os serviços não deveriam ter estado. Embora essa condição aprimore o escalonamento horizontal e o desempenho, na realidade, muitas transações de negócio, na vida real, envolvem muitas mensagens em um período significativo, de modo que o estado precisa ser mantido em algum lugar. Se o estado não for mantido de modo persistente, poderá se perder, e isso ocasionará a falha de toda a transação do negócio. Com o Service Broker, os diálogos e os grupos de conversação são persistentes e, assim, mantêm o estado como característica essencial. O seu aplicativo pode usar este fato com outro: as mensagens estão no banco de dados, de qualquer forma, para manter o estado de modo escalável e de alto desempenho. Isto faz com que as transações de longa duração do negócio sejam mais fáceis de implementar.

Embora haja muitas formas de projetar a manipulação de estados, os aplicativos de Service Broker têm uma vantagem especial. Todas as mensagens de qualquer dos diálogos de um grupo de conversação têm o mesmo ID do grupo de conversação. Como um comando RECEIVE retorna mensagens apenas de um único grupo de conversação, todas as mensagens devolvidas terão o mesmo ID do grupo de conversação e - se os diálogos estiverem corretamente projetados - estarão associadas à mesma transação do negócio. A vantagem desse procedimento é que se você armazenar o estado do aplicativo em tabelas com o ID do grupo de conversação como a chave, poderá obter a chave do estado de qualquer mensagem recebida, ou seja, você poderá escrever facilmente um lote TSQL que devolva as duas mensagens para a fila e as informações de estado necessárias para processá-las, para que a manipulação de estados seja fácil e rápida. Além disso, lembre-se que apenas um segmento pode processar mensagens de um determinado grupo de conversação de cada vez e, assim, se o ID do grupo de conversação for a chave de estado, apenas um segmento acessará o estado do aplicativo em um determinado momento e, assim, o conflito de atualizações não será um problema.

Poison Messages

São mensagens que nunca podem ser processadas corretamente. Um exemplo simples é um cabeçalho de pedido com um número que já existe no banco de dados. Ao tentar inserir o cabeçalho

no banco de dados, haverá falha na inserção com uma violação de restrição exclusiva (unique constraint), a transação será devolvida e a mensagem voltará para a fila. Independentemente do número de tentativas, a inserção falhará e o serviço entrará em loop, processando a mesma mensagem inúmeras vezes. Isso causará obstrução na entrada do pedido e poderá afetar gravemente o desempenho do banco de dados. Para evitar que uma poison message arruíne o seu servidor, o Service Broker desativará a fila se houver cinco devoluções seguidas. Este procedimento permite que o restante do servidor continue a operar, mas o aplicativo de entrada de pedidos estará morto, porque a fila estará desativada.

A forma de evitar isso é dar roolback apenas na transação que acionou o comando RECEIVE, caso haja alguma esperança de que uma nova tentativa possa processar a transação com êxito na próxima vez. Se a transação falhar devido ao limite de tempo de um lock, ser selecionada como vítima de deadlock, falta de memória ou motivo semelhante, será sensato devolver a transação e tentar novamente. Todavia, se o erro for permanente, você precisará manipulá-lo no aplicativo. O que mais acontece é encerrar o diálogo com um erro e registrar o erro em uma tabela de erros. O método escolhido para manipular as poison messages depende das exigências do seu aplicativo, mas é importante incluir essa manipulação no seu projeto.

Prioridade da mensagem

Sem rodeios, informo que o Service Broker não tem uma forma interna para garantir prioridade da mensagem. Este fato traz muita angústia entre desenvolvedores, acostumados a usar prioridade da mensagem para assegurar o processamento antecipado de determinadas mensagens. Acredito, pela minha experiência pessoal, que a maioria das pessoas, de fato, não precisa de prioridade da mensagem em seus aplicativos. Precisam, apenas, de condições para assegurar que mensagens de alta prioridade não fiquem na fila atrás de várias mensagens de baixa prioridade. Estabelecer filas de alta e baixa prioridades é forma mais fácil de fazer isso no Service Broker. A ativação pode ser usada para atribuir leitores de fila suficientes para que a fila de alta prioridade manipule a carga e atribuir um único leitor para a fila de baixa prioridade. Assim, as mensagens de baixa prioridade serão processadas em paralelo com as de alta prioridade, mas estas nunca ficarão bloqueadas pelas mensagens de baixa prioridade. Se precisar ter mais controle sobre prioridade do que o quanto esta abordagem lhe oferece, existem outras formas, discutidas no meu blog, nestes dois artigos localizados aqui[7] e aqui[8].

Transações de compensação

Os serviços do Service Broker processam as mensagens recebidas em uma transação diferente daquela - e, possivelmente, muito mais tarde - que enviou a mensagem. Uma transação do negócio pode incluir dezenas de transações de banco de dados. Por isso, não é possível devolver uma transação do negócio, em caso de erro. Ainda que fosse possível, desfazer os efeitos de uma transação do negócio envolve, em geral, muito mais do que simplesmente reverter o banco de dados a um estado anterior. Para cancelar um pedido, por exemplo, será preciso transferir o item da expedição para o estoque (ou até mesmo tirá-lo do caminhão), cancelar o débito do cartão de crédito, enviar uma nota de cancelamento, etc. O projeto do serviço talvez tenha de incluir configurações de transações de compensação para desfazer os efeitos de uma atividade com erros ou que seja cancelada. No meu blog[9] existe uma discussão mais completa sobre transações de compensação, assim como neste artigo da MSDN Library[10].

Inicio da pagina

Aspectos de infra-estrutura de um aplicativo de Service Broker

Esta seção final cobre alguns dos aspectos de infra-estrutura e implantação de uma solução Service Broker. Um dos pontos do projeto do Service Broker é que o aplicativo deve conhecer o menos possível sobre como deve ser implantado. Por exemplo, o nível de segurança pode ser determinado e alterado por uma DBA, sem necessidade de alterar o código do aplicativo. Por isso, quase sempre é possível falar sobre questões de infra-estrutura de SSB de modo isolado das questões de projeto do aplicativo.

Como se trata de uma discussão arquitetural, não abordarei os comandos para configuração da infra-estrutura, mas destacarei questões de infra-estrutura que devem ser consideradas na

implantação de um aplicativo do Service Broker. Consulte os livros sobre SQL Server on-line e um dos livros sobre Service Broker para obter informações detalhadas para realizar a configuração e a implantação.

Questões de segurança

Um aplicativo que envia mensagens através de uma conexão de rede deve ter uma infra-estrutura de segurança. O Service Broker possui várias camadas de segurança e as opções de cada uma delas permitem ajustar a segurança do SSB de acordo com as necessidades da rede e dos dados.

Uma das opções é a Segurança do diálogo. Quando se estabelece um diálogo seguro, chaves (pública e privada) assimétricas são usadas para autenticar a conexão do diálogo, as permissões SQL Server são confirmadas, uma chave de sessão é definida para o diálogo e todas as mensagens são assinadas e criptografadas. Este procedimento assegura que as mensagens serão entregues inalteradas e não lidas. A segurança do diálogo é estabelecida entre as duas extremidades do diálogo e, assim, se as mensagens forem encaminhadas por intermediários, não serão descriptografadas. Este procedimento é tanto mais seguro quanto mais eficiente do que a criptografia SSL via rede. A segurança do diálogo deveria ser usada se a mensagem tiver de ser transmitida por redes desprotegidas.

Se você achar que o tráfego de mensagens de um determinado diálogo é tão sigiloso que precisaria estar criptografado, em qualquer rede, configure o parâmetro ENCRYPTION do comando BEGIN DIALOG em On, como padrão. Se não for necessário criptografar os dados a serem transmitidos em algumas redes, você deve configurar o parâmetro ENCRYPTION em Off. A configuração em Off não significa que os dados não serão criptografados. Significa que se a DBA definir a segurança do diálogo, ele será criptografado; caso contrário, não. Se o parâmetro ENCRYPTION estiver configurado em On (padrão), nenhuma mensagem será enviada para fora da instância local a não ser que a segurança do diálogo esteja configurada.

O Service Broker também tem segurança no nível de conexão TCP/IP. Todas as conexões do Service Broker exigem autenticação, autorização e assinaturas digitais para confirmar se a conexão está autorizada e se as mensagens não serão alteradas na rede. A conexão TCP/IP pode ser criptografada, opcionalmente. Se as mensagens já estiverem criptografadas pela segurança do diálogo, não passarão novamente por esse processo, pois as seguranças do diálogo e do transporte são complementares.

O plano de implantação do aplicativo deve definir quais diálogos e conexões devem ser criptografados. Se você usar a segurança do diálogo, deverá incluir um plano para manipular a validade dos certificados.

Questões de disponibilidade

O Service Broker é muito usado em sistemas de alta disponibilidade e, por isso, a disponibilidade deve ser considerada no projeto da infra-estrutura de SSB. Como faz parte do mecanismo do banco de dados, o Service Broker estará altamente disponível se você fizer com que o banco de dados assim também esteja. Muito já se escreveu sobre a disponibilidade do SQL Server, por isso não vou abordar este tema. O único e exclusivo recurso de disponibilidade oferecido pelo Service Broker é a total integração com o espelhamento de banco de dados. Se o Service Broker abrir uma conexão com um banco de dados espelhado, também abrirá uma conexão com o banco de dados secundário no espelho, de modo que se houver falha do primário para o secundário, o Service Broker fará essa detecção, automaticamente, e passará a enviar as mensagens para o novo banco primário. Como as mensagens do SSB são transacionais e ficam armazenadas no banco de dados, quaisquer mensagens em andamento e quaisquer mensagens não processadas continuarão ativas na fila depois da falha, para que tudo continue sem perda de dados nem trabalhos não concluídos.

Questões de roteamento

Os diálogos do Service Broker ficam abertos entre serviços: um serviço FROM e outro TO especificados no comando BEGIN DIALOG. Um serviço do Service Broker é, basicamente, o nome de um endpoint (ponto de extremidade) do diálogo. O ponto de extremidade do diálogo é uma fila do SSB em um banco de dados SQL Server. Esta comunicação indireta (indirection) do serviço para a fila significa que você poderá escrever o seu aplicativo para comunicar-se entre serviços lógicos e tomar a decisão sobre onde os serviços devem estar realmente localizados na hora da implantação.

Na verdade, os serviços podem ser transferidos para outro local enquanto os diálogos estiverem ativos, sem que nenhuma mensagem se perca.

O mapeamento entre o nome lógico do serviço e o endereço de transporte para onde as mensagens são enviadas é feito com um roteamento do Service Broker. Roteamento é apenas uma linha da tabela sys.routes do banco de dados. Quando o Service Broker precisa enviar uma mensagem, procura um roteamento para o serviço de destino e informa o endereço para a camada de transporte, que o envia ao banco de dados em que reside o serviço de destino.

Os serviços iniciador e de destino de um diálogo precisam de um rotear para endpoints opostos. Um dos cenários do SSB mais comuns envolve muitos iniciadores que enviam mensagens para um único destino - por exemplo, terminais de ponto de venda que enviam transações para a matriz ou estações de trabalho de um corretor de ações que enviam as negociações para um sistema administrativo. A atualização de centenas de roteamentos no servidor de destino com todos os iniciadores pode ser confusa. Para evitar esse problema, o SSB fornece o comando de roteamento TRANSPORT. O nome do serviço iniciador contém o seu endereço de rede e esse nome é usado como o serviço FROM no comando BEGIN DIALOG.

Os serviços iniciador e de destino de um diálogo precisam de um rotear para endpoints opostos. Um dos cenários do SSB mais comuns envolve muitos iniciadores que enviam mensagens para um único destino - por exemplo, terminais de ponto de venda que enviam transações para a matriz ou estações de trabalho de um corretor de ações que enviam as negociações para um sistema administrativo. A atualização de centenas de roteamentos no servidor de destino com todos os iniciadores pode ser confusa. Para evitar esse problema, o SSB fornece o comando de roteamento TRANSPORT. O nome do serviço iniciador contém o seu endereço de rede e esse nome é usado como o serviço FROM no comando BEGIN DIALOG. Se o destino quiser responder uma mensagem ao iniciador, usará o nome do serviço como o endereço da rede. Desta forma, o iniciador fornece seu próprio endereço de retorno para que você não tenha de atualizar os endereços de retorno no sistema de destino.

O Service Broker também é compatível com encaminhamento. As mensagens que chegam a uma instância do SQL Server são encaminhadas de acordo com os roteamentos no banco de dados MSDB. Se o roteamento de um serviço especificar "local" como o endereço de rede, a mensagem será encaminhada para um serviço dessa instância. Se o endereço de rede do roteamento MSDB do serviço especificar um endereço TCP/IP, a mensagem será encaminhada ao endereço especificado. Mensagens encaminhadas não são gravadas no banco de dados: são retidas na memória até serem encaminhadas e, por isso, o encaminhamento SSB é bastante eficiente. Um uso comum do encaminhamento do SSB é a configuração de um concentrador que aceita mensagens de um grande número de conexões e depois as encaminha por uma única conexão ao destino. Este procedimento transfere a maior parte da manipulação da conexão TCP/IP para o encaminhador, que reduz a sobrecarga da máquina de destino. Se você usar o SQL Express na máquina que encaminha as mensagens, isto poderá ser uma forma bastante econômica de reduzir a sobrecarga no destino.

Gerenciamento e monitoração

Nenhum aplicativo importante será completo sem provisões para monitoração e gerenciamento do aplicativo em produção. Este aspecto é especialmente importante nos aplicativos do Service Broker porque devido à natureza do SSB, confiável e assíncrona, é quase sempre difícil dizer se o aplicativo está ou não sendo executado de modo correto. Por exemplo, se um dos serviços pára de processar as mensagens, o Service Broker coloca as mensagens no fim da fila desse serviço, até que o funcionamento volte a ser normal. Se o pessoal de operação não perceber que o serviço não está funcionando, as mensagens podem ficar enfileiradas por horas.

Como o Service Broker faz parte do SQL Server, as ferramentas e técnicas usadas para monitorar o SQL Server fazem o mesmo para o Service Broker. Existem vários contadores Perfmon que medem desempenhos e velocidades de processamento. Há muitos eventos de rastreamento que podem ser usados para rastrear a entrega das mensagens para resolver problemas. O pacote SQL Server MOM inclui várias estatísticas do Service Broker. Todas as filas têm uma visão do SQL para que você possa saber, facilmente, quantas mensagens há na fila e de onde vieram.

O projeto do aplicativo de Service Broker deve incluir um plano para monitorar a saúde e o desempenho do aplicativo. O serviço também deve incluir lógica para diagnóstico e relatórios de

problemas. Por exemplo, incluir uma mensagem "echo" em cada contrato que apenas retorna uma mensagem ao serviço que a enviou é uma ferramenta útil para determinar se o serviço está ativo e processando as mensagens.

Inicio da pagina

Conclusões

Os recursos exclusivos do SQL Server 2005 Service Broker oferecem toda uma nova classe de aplicativos assíncronos e confiáveis. O Service Broker pode trazer novos níveis de eficiência e de tolerância a falhas aos aplicativos de banco de dados.

O potencial das operações assíncronas de enfileiramento pode ser usado para projetar aplicativos de banco de dados de alto desempenho, mas os modelos de projeto assíncronos podem ser difíceis de dominar para um arquiteto treinado nos aplicativos RPC síncronos, convencionais. Este artigo destacou algumas das decisões de projeto que qualquer pessoa que esteja projetando um aplicativo de Service Broker deve considerar. O trabalho de projeto de um aplicativo de Service Broker pode ser bem diferente daquele de um aplicativo convencional, mas os resultados valem o esforço.

Inicio da pagina

Tabela de Ligações

[1]*http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsql90/html/sqlsvcbroker.asp*

[2]*http://www.amazon.com/rational-server-service-broker-guides/dp/1932577270/sr=1-1/qid=1158009357/ref=pd_bbs_1/103-0049023-7244631?ie=utf8&s=books*

[3]*http://blogs.msdn.com/rogerwolterblog/archive/2006/02/28/540803.aspx*

[4]*http://www.architecturejournal.net/2006/issue8/f1_reliability/*

[5]*http://blogs.msdn.com/rogerwolterblog/archive/2006/05/20/602938.aspx*

[6]*http://blogs.msdn.com/rogerwolterblog/archive/2006/04/04/568351.aspx*

[7]*http://blogs.msdn.com/rogerwolterblog/archive/2006/03/11/549730.aspx*

[8]*http://blogs.msdn.com/rogerwolterblog/archive/2006/03/17/554134.aspx*

[9]*http://blogs.msdn.com/rogerwolterblog/archive/2006/05/24/606184.aspx*

[10]*http://msdn.microsoft.com/architecture/solutions_architecture/data/default.aspx?pull=/library/en-us/dnbda/html/concurev4m.asp*

Conteúdo da Comunidade