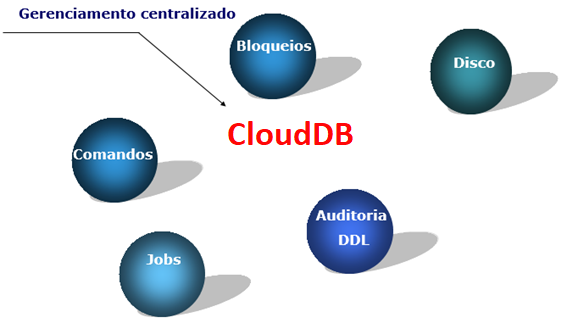


****

****

**Implementando Snapshot ou Read Committed Snapshot Isolation em SQL Server**

Desenho com traços pretos em fundo branco

Descrição gerada automaticamente com confiança baixa

Como alterar os níveis de isolamento sem perder suas marbles.

Um cliente me disse a coisa mais legal outro dia. Ele disse: “Nós conversamos antes sobre por que gostaríamos de começar a usar o bloqueio otimista em nosso código. Como chegamos lá?"

Se você não é um nerd do SQL Server, esse comentário provavelmente nem faz sentido. Mas para alguns de nós, quando você tem um ambiente a ponto de poder lidar com o bloqueio otimista E você tem desenvolvedores que estão prontos para fazer algumas alterações no código para usá-lo a seu favor, é empolgante!

Eu sou um grande fã do bloqueio otimista no SQL Server e tenho visto melhorias dramáticas no desempenho quando ele é bem implementado. Para o banco de dados certo e a equipe certa, um pouco de trabalho árduo facilita o dimensionamento do aplicativo a longo prazo.

**Optimistic Locking 101: Uma revisão rápida**

Se você é novo nos Níveis de Isolamento no SQL Server, esta postagem pode ser difícil de seguir. Considere fazer uma pausa rápida para navegar em nossa grande lista de referências sobre Níveis de Isolamento no SQL Server na Web.

Aqui está uma atualização muito simplificada para aqueles que conhecem seus níveis de isolamento, mas precisam escovar as teias de aranha:

O SQL Server usa o bloqueio “pessimista” nos bancos de dados do usuário, a menos que você diga para fazer o contrário. Seu nível de isolamento padrão no SQL Server é READ COMMITTED. Embora isso pareça reconfortante – quem não quer um pouco de compromisso de seu banco de dados? – é uma forma de bloqueio pessimista. Sob o nível de isolamento READ COMMITTED, os leitores podem bloquear os escritores e os escritores podem bloquear os leitores. Às vezes, o nível de isolamento padrão não é forte o suficiente para fornecer o nível certo de consistência, então as pessoas precisam usar níveis de isolamento mais altos (“mais pessimistas”, se preferir) para evitar fenômenos como leituras não repetíveis e registros fantasmas.

Digamos que temos uma tabela Orders e adicionaremos uma linha a ela:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | CREATE TABLE dbo.Orders  (Id INT IDENTITY(1,1) PRIMARY KEY CLUSTERED,  CreationDate DATETIME2 NOT NULL,  ItemOrdered NVARCHAR(100) NOT NULL);    INSERT INTO dbo.Orders(CreationDate, ItemOrdered)  VALUES (GETDATE(), N'Jaguar XKR-S'); |

Quando nosso banco de dados não está muito carregado, é fácil consultar os pedidos abertos - basta SELECT \* FROM dbo.Orders, e você verá instantaneamente o único pedido que tivemos.

No entanto, à medida que a carga aumenta em um banco de dados, a tensão entre garantir o isolamento da transação/resultados corretos e também o suporte à simultaneidade aumenta. Você começa a batalhar com bloqueios e impasses.

Enquanto alguém está tentando carregar uma nova linha na tabela Pedidos:

|  |  |
| --- | --- |
| 1  2  3  4 | BEGIN TRAN  INSERT INTO dbo.Orders  (CreationDate, ItemOrdered)  VALUES (GETDATE(), N'Ferrari 328'); |

Se alguém tentar verificar nossos pedidos ao mesmo tempo, será bloqueado:

[Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente](https://brentozarultd.wpenginepowered.com/wp-content/uploads/2013/01/blocked_orders_query.png)

Na captura de tela acima, a janela esquerda está inserindo uma linha e a janela direita está bloqueada. Esse select ficará lá para sempre até que a consulta esquerda confirme sua transação ou a reverta.

Para contornar isso, as pessoas costumam usar NOLOCK – que funciona no sentido de que a consulta à direita agora pode terminar instantaneamente:

[Interface gráfica do usuário, Aplicativo, Teams

Descrição gerada automaticamente](https://brentozarultd.wpenginepowered.com/wp-content/uploads/2013/01/nolock_query.png)

Mas isso tem resultados desastrosos porque NOLOCK mostrará linhas que nunca foram confirmadas, pulará linhas, lerá linhas duas vezes e sua consulta poderá falhar completamente.

**Insira o bloqueio otimista.**

O SQL Server 2005 introduziu dois novos níveis de isolamento para ajudá-lo em sua missão em direção a uma simultaneidade cada vez maior: isolamento SNAPSHOT e READ COMMITTED SNAPSHOT (esta segunda opção geralmente é chamada de “RCSI”). Esses dois níveis de isolamento usam controle de versão para permitir que algumas ou todas as suas consultas sejam aceleradas por transações “em andamento” e leiam uma versão dos dados confirmada anteriormente.

Eu só tenho que fazer duas coisas: ativar a simultaneidade otimista (mais sobre isso em um segundo) e adicionar uma única linha às minhas consultas somente leitura que desejam contornar problemas de bloqueio e ainda ver resultados precisos:

|  |  |
| --- | --- |
| 1  2  3 | SET TRANSACTION ISOLATION LEVEL SNAPSHOT;    SELECT \* FROM dbo.Orders; |

O resultado é, bem, resultados! Posso obter resultados instantaneamente em vez de ser bloqueado:

[Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente](https://brentozarultd.wpenginepowered.com/wp-content/uploads/2013/01/optimistic_concurrency.png)

A consulta de seleção vê apenas os dados verdadeiros - como se a inserção à esquerda ainda não tivesse sido confirmada, o que é verdade!

Readers Don’t Block Writers, Writers Don’t Block Readers

Desenho de personagem de desenho animado

Descrição gerada automaticamente com confiança baixa

Pessimistic marbles

O benefício mais simples e puro dos níveis de isolamento otimista no SQL Server é apenas este: você pode permitir que consultas mais pesadas e estilo de relatório sejam executadas sem bloquear suas gravações críticas no OLTP. De repente, você tem outra opção entre ajustar todas as consultas caras, usar leituras sujas ou ter que executar consultas em um servidor secundário.

Às vezes, você pode decidir: “Esta consulta não é a melhor, mas não há problema em executá-la se não bloquear ninguém e apenas ler dados confirmados anteriormente”.

E sim, você pode fazer mais do que isso. Você pode usar o nível de isolamento SNAPSHOT para consultas de modificação de dados, mas precisa começar a detectar e lidar com conflitos de atualização. (Observação: isso não se aplica a READ COMMITTED SNAPSHOT.) Não há nada de errado em fazer isso, mas na maioria dos ambientes não é a mudança de “maior retorno para seu investimento” a ser implementada.

**Como você implementa o instantâneo ou o isolamento de instantâneos confirmados de leitura… com segurança?**

Mudar para o bloqueio otimista parece ótimo, mas não é uma mudança rápida. Apresse-se muito cedo e você poderá sofrer grandes problemas de desempenho, perda de disponibilidade e resultados de consulta incorretos.

Essa lista de problemas potenciais assusta a maioria das pessoas. Essa lista faz um gerente de projeto pensar: “Tenho 99 problemas e o bloqueio otimista não é um”.

Mas realmente não é tão difícil testar e implementar o bloqueio otimista. Leva apenas um pouco de tempo e um bom plano. Aqui estão três etapas que sempre incluo ao planejar uma mudança no nível de isolamento.

**1. Meça o efeito que a habilitação do bloqueio otimista tem no desempenho do SQL Server**

Quando você tem uma base de código existente e está avaliando o bloqueio otimista, a princípio parece que você tem um pequeno problema de galinha e ovo.

1. Você não tem certeza se seu servidor de produção pode lidar com a carga

2. Os recursos de teste são necessários para garantir que a alteração não crie dados incorretos

Ninguém quer investir em testes se o ambiente não aguentar a carga. E as pessoas não sabem como testar o impacto do recurso com segurança se não for comprovado que não cria dados ruins. Pode parecer um impasse de gerenciamento de projetos.

Ícone

Descrição gerada automaticamente com confiança média

Optimistic marbles

Não tenha medo – existe um método para testar a carga primeiro! A chave está em entender a diferença entre os dois níveis de isolamento. Se você habilitar o isolamento SNAPSHOT em um banco de dados, o SQL Server começará a implementar tecnologias de controle de versão para que as consultas possam definir seu nível de isolamento como instantâneo. (As consultas precisam fazer isso explicitamente - como em executar um comando 'SET TRANSACTION ISOLATION LEVEL SNAPSHOT'.)

Aqui está a parte que é fácil de perder. Assim que você habilita o isolamento SNAPSHOT em um banco de dados, o SQL Server aguarda a conclusão das transações em execução e, em seguida, começa imediatamente a usar o controle de versão para modificações de dados. Você começa a usar 14 bytes extras por linha nas tabelas do próprio banco de dados. Além disso, as versões são criadas no armazenamento de versão tempdb para manter o valor anterior dos dados para atualizações, exclusões e algumas inserções. Isso acontece mesmo se nenhuma consulta for executada usando o isolamento SNAPSHOT. Seriamente.

Isso pode parecer um pouco como um roubo. Sim, você precisa pagar pelo desempenho, mesmo que não esteja usando o SNAPSHOT! Mas é por um bom motivo – seu SQL Server ainda não tem o recurso de ser psíquico. Ele não pode prever se a qualquer momento você iniciará uma consulta usando SNAPSHOT, e precisa garantir que o banco de dados esteja pronto se você fizer isso a qualquer momento.

Mas isso também pode ser um recurso útil. Você pode testar o impacto no desempenho do controle de versão de um banco de dados apenas definindo ALLOW\_SNAPSHOT\_ISOLATION como ON. Quando você emitir o comando para alterar isso, o SQL Server aguardará a conclusão de todas as transações ativas antes de retornar um status para você (os Manuais Online explicam por que e como lidar com isso aqui.) Você pode monitorar o desempenho pelo tempo que tiver planejado e depois inverta o processo.

Contanto que você não tenha alterado nenhum código e esteja usando a opção ALLOW\_SNAPSHOT\_ISOLATION, nenhuma consulta realmente USE o nível de isolamento alterado e você pode simplesmente monitorar o impacto no desempenho do bloqueio otimista.

Uma pegadinha: mencionei uma configuração específica de propósito. A opção READ\_COMMITTED\_SNAPSHOT tem o mesmo nome, mas é muito diferente! Essa configuração altera seu nível de isolamento padrão para usar o bloqueio otimista sem alterações de código em suas consultas. Assim que você ativá-lo, algumas consultas podem começar a obter resultados inesperados ou gravar dados incorretos. (Um exemplo disso está abaixo em #3.) O processo de ativação de READ\_COMMITTED\_SNAPSHOT também funciona de forma diferente. Por um único instante, a conexão que emite o comando para habilitar o RCSI deve ser o único comando ativo no banco de dados. (Eu entro em mais detalhes sobre isso na seção “pegadinhas” no final deste guia.)

Por esses dois motivos, ALLOW\_SNAPSHOT\_ISOLATION é muito mais adequado para descobrir como habilitar o bloqueio otimista afeta sua carga de trabalho. Ainda siga o bom senso: teste primeiro em horários de baixo volume e trabalhe até horários mais movimentados. Saiba que desabilitar e voltar ao “normal” pode nem sempre acontecer tão rápido quanto você deseja se você tiver transações de longa duração ou uma carga de trabalho muito pesada.

**2. Planeje a prevenção de desastres com o isolamento de instantâneos**

O segundo passo é identificar o que pode sair do controle se algo incomum acontecer – como uma sessão não autorizada que deixa uma transação aberta. Este não é um cenário distante. Isso aconteceu devido a erro humano e bugs nas estruturas de aplicativos. Realmente não é tão raro e pode acontecer com você.

Seu plano precisa identificar os maiores riscos da mudança para o seu negócio e o que pode ser um desafio para a equipe que responde a incidentes. Em seguida, configure as coisas para que, se coisas estranhas acontecerem, as pessoas sejam notificadas e saibam como fazer a triagem do problema.

Aqui está um exemplo de duas coisas que você pode identificar como suas maiores preocupações se você implementar o bloqueio otimista no SQL Server:

• Tempdb pode preencher

• Cadeias de versões longas podem ser criadas, fazendo com que o desempenho da consulta fique super lento

A forma como você lida com isso varia de acordo com seu sistema de monitoramento, ferramentas disponíveis e documentação. Mesmo se você não tiver ferramentas sofisticadas, poderá usar os Alertas do contador de desempenho do SQL Server para saber se o espaço usado no Tempdb está acima de uma determinada linha de água. Você também pode alertar sobre contadores de desempenho para o tempo de transação mais longo ou usar DMVs para monitorar o armazenamento de versão em detalhes. (Observe que nem todos esses DMVs são leves para usar quando o armazenamento de versão fica grande.)

Sua missão é simples: identificar a maneira mais simples de monitorar as condições com as quais está preocupado. Teste se funciona em um sistema de não produção. Mais importante ainda, anote a documentação do alerta que ajudará alguém a decodificar o problema se você não estiver por perto para lidar com tudo e realizar ações básicas.

**3. Escolha com cuidado entre o instantâneo e o isolamento de instantâneo confirmado de leitura (RCSI)**

OK, agora é aí que entram os desenvolvedores. Você pode usar o bloqueio otimista no SQL Server usando uma dessas opções de banco de dados – ou ambas. Qual é a certa para você?

A maior consideração com uma base de código existente é quanto teste está disponível para o código. Eu acho que o nível de isolamento READ COMMITTED SNAPSHOT é ótimo, mas vai mudar seu nível de isolamento padrão para cada consulta imediatamente. Isso pode levar a resultados incorretos, dependendo de como seu código é escrito. O pior exemplo é se o seu código:

1. Lê os dados da tabela em variáveis ​​ou tabelas temporárias

2. Usa as variáveis ​​ou os dados da tabela temporária para atualizar os dados da tabela, e o código apenas assume que nenhuma alteração foi feita entre as etapas 1 e 2

Isso é chamado de condição de corrida, e explicar isso está além do escopo deste post. Nem todos os padrões de codificação são propensos a esse problema. No entanto, com uma base de código existente, validar todo o código e garantir que você não tenha nenhum código legado propenso a esse problema nem sempre é uma coisa simples. Se você tem a capacidade de fazer alterações em seu código e já conhece uma classe de consultas que gostaria de usar com bloqueio otimista, talvez prefira introduzir o isolamento SNAPSHOT em seu ambiente primeiro, em vez de READ\_COMMITTED\_SNAPSHOT.

Para implementar o isolamento SNAPSHOT em algumas instruções, você precisa primeiro habilitá-lo usando a opção de banco de dados ALLOW\_SNAPSHOT\_ISOLATION. Você pode então implementar o isolamento SNAPSHOT para algumas consultas – você faz isso definindo o nível de isolamento com 'SET TRANSACTION ISOLATION LEVEL SNAPSHOT'. É mais fácil implementar isso no início para consultas de leitura que você testou e sabe que não são propensas a problemas de dados semelhantes ao que mostramos acima. Isso significa que você pode precisar usar strings de conexão separadas para essas consultas ou certifique-se de redefinir e verificar o nível de isolamento em que está operando após a conclusão das instruções.

Em outras palavras, o nível de isolamento é uma configuração de nível de sessão. Você precisa garantir que não use o isolamento SNAPSHOT por acidente para mais consultas do que pretende.

**Pegadinhas com a implementação de instantâneos e níveis de isolamento de instantâneos confirmados de leitura**

Há alguns pequenos fatos que descobri ao longo dos anos que não esperava. Eles fazem todo o sentido, eu não tinha pensado neles até trabalhar com bloqueio otimista no SQL Server.

• As dicas em seu código ainda se aplicam. Digamos que você tenha um problema com o bloqueio. Ao longo dos anos, dicas NOLOCK são adicionadas em muitos lugares para ajudar a melhorar isso. Você finalmente recebe a confirmação de sua equipe de desenvolvimento de que READ\_COMMITTED\_SNAPSHOT é seguro para seus aplicativos e sua alteração foi aprovada, então você a ativa. Você está gastando todos esses recursos de desempenho em versionamento, mas adivinhe? Essas dicas NOLOCK ainda estão fazendo com que as consultas façam leituras sujas em vez de usar o controle de versão de dados! As dicas do NOLOCK precisam ir.

• Os gravadores ainda bloqueiam os gravadores no isolamento de instantâneos confirmados de leitura (RCSI) e, dependendo de como você gravou suas transações, isso pode alterar os resultados em seu aplicativo. Leia mais aqui.

• Conflitos de atualização não são o mesmo que deadlocks. Os conflitos de atualização só são possíveis quando você usa o isolamento SNAPSHOT para consultas de modificação de dados – você não precisa se preocupar com isso com READ\_COMMITTED\_SNAPSHOT. No entanto, muitas vezes é mais prático para as pessoas implementarem o SNAPSHOT por causa dos problemas de teste que descrevi acima. Mesmo se você estiver implementando SNAPSHOT apenas para transações de leitura, familiarize-se com os códigos de erro e mensagens para conflitos de atualização e certifique-se de que seu código trata o erro 3960 (“Transação de isolamento de instantâneo abortada devido a conflito de atualização…”).

• Ativar READ\_COMMITTED\_SNAPSHOT em um sistema ocupado é mais difícil do que parece. Como mencionei antes, ativar ou desativar READ\_COMMITTED\_SNAPSHOT é um pouco incomum. Você não precisa tecnicamente colocar o banco de dados no modo de usuário único, mas para concluir o comando, você precisa executar o único comando ativo no momento. A maneira mais simples de fazer isso é usar a cláusula ‘WITH ROLLBACK IMMEDIATE’ do comando ALTER DATABASE. No entanto, não descobri que isso seja executado de maneira previsível ou fácil em sistemas de transações muito altas. Eu recomendo planejar uma mudança para ativar ou desativar a configuração READ\_COMMITTED\_SNAPSHOT em um banco de dados em um tempo de volume muito baixo, se você precisar manter as coisas previsíveis.

• Reverter e desabilitar SNAPSHOT requer mais alterações de código. Para interromper o controle de versão de linha, você precisa desabilitar o SNAPSHOT — e assim que fizer isso, as consultas que definem o nível de isolamento como SNAPSHOT e tentam executar falharão com o erro 3292: “A transação de isolamento de instantâneo falhou ao acessar o banco de dados 'dbname' porque o isolamento de instantâneo não é permitido neste banco de dados”

Mudanças no nível de isolamento não são realmente Rocket Science

Os níveis de isolamento no SQL Server são complicados. Mas se você é novo nisso, não pense que é apenas o bloqueio otimista que o torna complicado. Acontece que o nível de isolamento de leitura confirmada também não é tão direto e previsível quanto você imagina!

À medida que trabalho cada vez mais com o SQL Server, continuo aprendendo que os níveis de isolamento são complexos, quer você esteja usando bloqueio pessimista ou otimista. O essencial é usar um plano inteligente de teste e implementação para encontrar com segurança as configurações que funcionam para sua equipe e seus aplicativos.