Lock

Administração de Banco de Dados

Tópicos

- Lock
- Tipos de Lock
- Recursos que podem ser bloqueados
- Dica de Lock sobre uma Tabela
- Dica de Lock em uma Sessão
- Compatibilidade entre Locks

Lock

- Os **Locks** servem para impedir que os dados de suas tabelas sejam atualizadas por mais de um usuário ao mesmo tempo, o que poderia provocar inconsistência no seu database.
- Bancos de Dados Relacionais como o SQL Server implementam Locks para que um determinado dado esteja travado enquanto alguem faz uma modificação naquele dado, nenhum usuário poderá modificar os dados que estão sendo modificados por outro usuário

Tipos de Lock

- Lock Compartilhado (SHARED LOCK – X)
- Lock Exclusivo (EXCLUSIVE LOCK – X)
- Lock de Update (UPDATE LOCK – U)
- Lock de Intenção (INTENT LOCK – I)
- Lock de Alteração em Volume (BULK UPDATE – BU)
- Lock de Schema (SCHEMA – SCH)

SHARED LOCK - X

- É o tipo de Lock usado durante o processamento de comandos que não modificam dados, como o SELECT.
- O SQL Server permite que um dado possa ser lido por mais de uma cenxão ao mesmo tempo. Enquanto esses dados são recuperados, o SQL Server coloca um lock do tipo compartilhado o que impede que outros atualizem estes dados enquanto estão sendo lidos

SHARED LOCK - SH

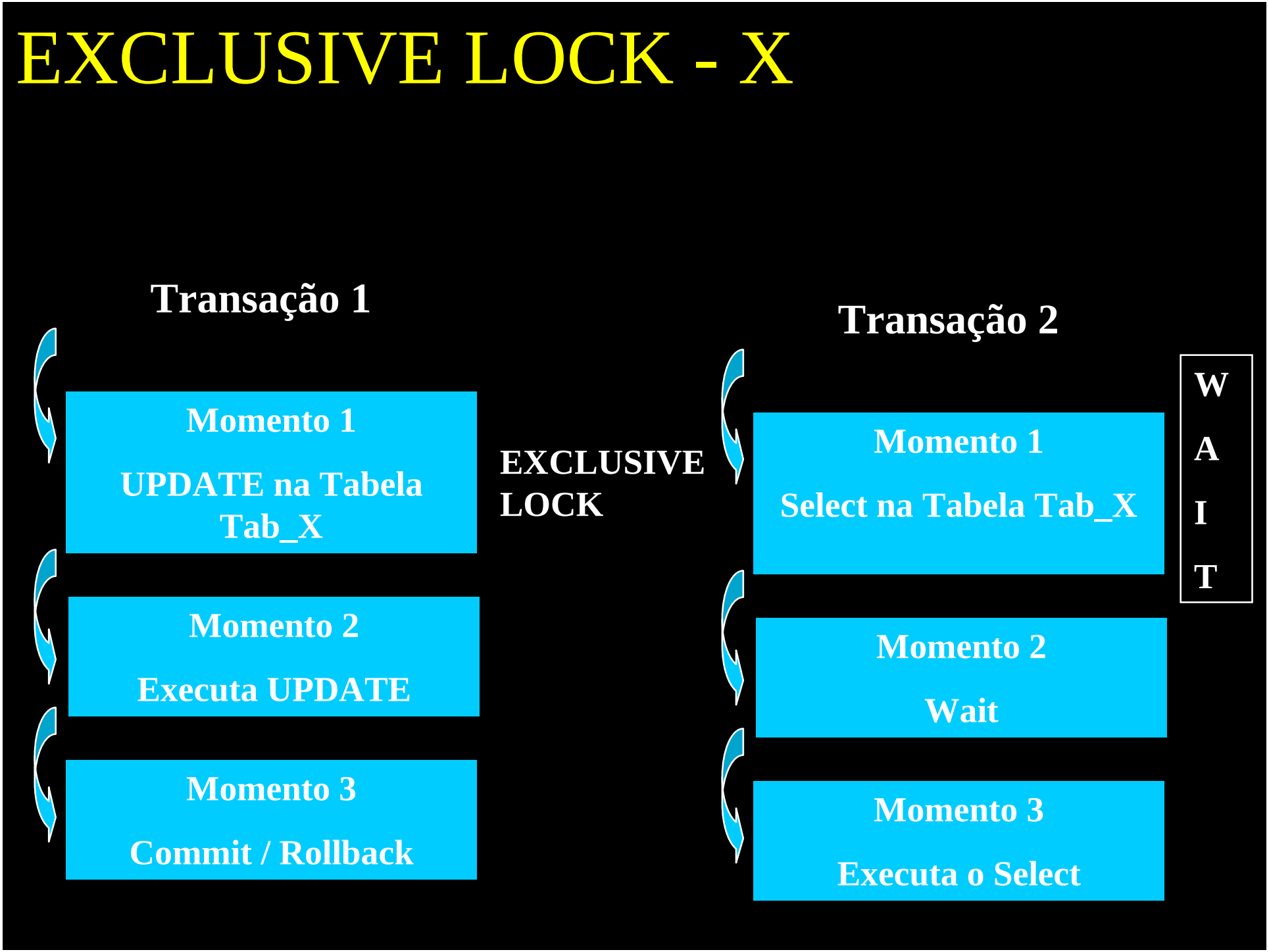
TRANSAÇÃO 1
Momento 1
Select na tabela Tab_x
O SQL Server coloca
SHARED
NOS REGISTROS
LIDOS

TRANSAÇÃO 2
Momento 1
Select na tabela Tab_x
O SQL Server coloca
SHARED
NOS REGISTROS
LIDOS

EXCLUSIVE LOCK - X

- Bloqueia qualquer acesso inclusive a leitura de um determinado dado que se encontra bloqueado, este é utilizado durante operações de inserção/deleção/atualização de dados;
- São grandes causadores de bloqueio do sistema;
- Operações que coloquem um Lock exclusivo devem demorar o mínimo possível;

EXCLUSIVE LOCK - X



UPDATE LOCK - X

- Prevenção de DeadLocks;
- Um Update consiste em uma transação que lê os dados (portanto coloca nesses dados um Shared Lock(S)) e em seguida modifica-os (colocando sobre ele o Exclusive Lock (X)). Você pode notar que ocorre uma conversão de lock de Shared em Exclusive.
- O SQl Server resolve problema de DeadLock, desfazendo uma das transações para que a outra prossiga, envia uma mensagem para a transação desfeita, informando do deadlock, que ela foi desfeita e avisa o usuário para reexecutar seu processamento.

UPDATE LOCK - U

- Apenas Uma tabela pode lançar um UPDATE Lock sobre determinada coluna;
- UPDATE Lock é seguido de um EXCLUSIVE Lock (Caso as modificações precisem ser feitas) os dados são atualizados, e o Lock é desfeito, assim, se duas transações tentam fazer ambas uma atualização a primeira que lançar o UPDATE Lock em seguida lança um Exclusive Lock, a segunda espera a operação de Exclusive Lock e só depois pode entrar em execução

INTENT LOCK - I

- Indica a intenção de futuramente o SQL Server colocar Locks em determinado conjunto de dados para uma determinada transação.
- Os Locks exclusivos são enfileirados, O lock de intenção impede que outras transações coloquem exclusive lock nesses dados.
- Existem três tipos
  - ✓Shared Intent Lock (IS)
  - ✓Exclusive Intent Lock (ix)
  - ✓Shared com Intent Exclusive Lock (SIX)

INTENT LOCK - I

Shared Intent Lock(IS)

O Lock IS indica que a transação irá ler alguns(mas não todos) dados na tabela Colocando Shared Locks

Intent Exclusive Lock(IX)

O Lock IX indica que a transação deve modificar alguns (Mas não todos) os dados na tabela colocando Exclusive Locks.

Shared With Intent Exclusive Lock(SIX)

O Lock SIX indica que a transação deve ler todas as fontes, e modificar algumas (Mas não todas). Isto é executado colocando shared locks nas fontes lidas e exclusive locks nas colunas modificadas.

BULK UPDATE - BU

- Lock de alteração de volume;
- Usado durante uma cópia de grande volume de dados de uma tabela para outra, ou quando o usuário utilizar a opção TABLOCK;
- Melhora a performance enquanto uma cópia do tipo bulk é feita;
- De fato ele diminue a concorrencia a tabela de destino, desabilitando o acesso tanto de leitura quanto de escrita a tabela.;

SCHEMA LOCK - SCH

Existem dois tipos de lock de esquema, que são colocados quando se fazem alterações nos esquemas das tabelas:

1. **SCH_M** – Locks Schema Modification são colocados no esquema de uma tabela quando sua estrutura estiver sendo alterada, quando comando de DDL (Definição de dados) são executados.
2. **SCH_S** – Lock Schema Stability indica que a query que está acessando a tabela está sendo compilada. Esse lock não impede exclusive lock (X), mais impede que outras sessões alterem o esquema da tabela bloqueada.

Recursos que podem ser bloqueados

Nome	Nível do Lock	Quem coloca o Lock nos dados
RID	Row Identifier. Usado para bloquear uma única linha de dado dentro de uma tabela.	a. O SQL Server bloqueia automaticamente as linhas individuais. b. O usuário pode colocar lock na linha com a opção ROWLOCK.
KEY	É um lock de linha dentro de um índice. Usado para proteger ranges de chaves em transações em série.	O SQL Server bloqueia as linhas dos índices automaticamente.
PAGE	Bloquear páginas inteiras (8kb).	a. O SQL Server. b. O usuário (PAGLOCK)

Recursos que podem ser bloqueados

Nome do nível do Lock	Nível do Lock	Quem coloca o Lock nos dados
ENTENT	É um nome que se dá para um conjunto contíguo de oito páginas (de 8KB cada uma).	O SQL Server bloqueia entents automaticamente.
TABLE	Uma tabela incluindo todos os dados e índices.	a. O SQL Server b. O usuário pode com as opções TABLOCK e TABLOCKX.
ESQUEMA	Todos os objetos de um usuário.	Colocada automaticamente pelo SQL Server quando a estrutura de uma tabela estiver sendo modificada.
DB	Database.	SQL Server quando um database estiver sendo restaurado.

Otimização com Locks

Opção de Lock	Granularidade	Duração do Lock
ROWLOCK	Lock de linha	Por default, o SQL Server mantém este lock até o final do comando.
PAGLOCK	Lock de página	Por default, o SQL Server mantém este lock até o final do comando.
TABLOCK	Lock de tabela	Por default, o SQL Server mantém este lock até o final do comando.
TABLOCKX	Lock exclusivo de tabela.	Por default, o SQL Server mantém este lock até o final da transação.

Otimização com Locks

Opção de Lock	Granularidade	Tempo de duração
HOLDLOCK	Lock de Update	Mantém este lock até o final da transação. É equivalente a SERIALIZABLE.
UPDLOCK	Lock de Update	Mantém este lock até o final do comando.
NOLOCK	Não bloqueia os dados. Apenas consegue ler dados que estão bloqueados de forma exclusiva – em processo de alteração.	Só pode ser aplicado no comando SELECT. É chamado lock de leitura suja, pois mostra alteração ainda não confirmada que, portanto, pode ser desfeita. Mantém o lock até o final do comando.
READPAST		Salta linhas bloqueadas. Esta opção faz com que uma transação salte sobre linhas bloqueadas por outras transação.

Otimização com locks

Opção do Lock	Tipo de Lock	Tempo de duração
READ COMMITED	Exibe apenas dados de tabelas sem lock.	Existe enquanto outro lock for configurado na sessão. É o lock default.
READ UNCOMMITED	Equivale ao NOLOCK.(leitura suja).	Existe enquanto outro lock for configurado na sessão
REPEATABLEREAD	Permite que a 2ª transação leia e insira dados, mas impede que ela altere dados lidos pela 1ª transação.	Dura até o final da transação.
SERIALIZABLE	Permite que a 2ª transação apenas leia dados existentes na faixa de dados atingidos pelo lock. (HOLDLOCK)	Dura até o final da transação.

UPDLOCK

- Se você quiser, pode colocar, com o comando SELECT, o lock de update em uma tabela, para garantir que outras transações possam ler os dados, mas só a sua transação possa alterá-los. Observe o exemplo a seguir:

UPDLOCK

Transação 1	Transação 2
Momento 1	Momento 2
BEGIN TRANSACTION SELECT * FROM Funcionario WITH (UPDLOCK) WHERE Cod_Func = 1 **Hint**	BEGIN TRANSACTION SELECT * FROM Funcionario --OK WHERE Cod_Func = 1 UPDATE Funcionario --Wait SET Sal_Func = Sal_Func * 1.1 WHERE Cod_Func = 1
Momento 3	Momento 4
UPDATE Funcionario SET Sal_Func = Sal_Func * 1.1 WHERE Cod_Func = 1	SELECT * --Wait FROM Funcionario WHERE Cod_Func = 1

NOLOCK

Permite que o SELECT leia dados que estejam em processo de alteração pela primeira transação. Veja o exemplo seguinte:

Transação 1	Transação 2
Momento 1	Momento 2
BEGIN TRANSACTION UPDATE Funcionario SET Sal_Func = Sal_Func * 1.1 WHERE Cod_Func = 1	BEGIN TRANSACTION SELECT * FROM Funcionario --OK WITH (NOLOCK) WHERE Cod_Func = 1 UPDATE Funcionario --Wait SET Sal_Func = Sal_Func * 1.1 WHERE Cod_Func = 1

READPAST

Le os dados da tabela “pulando” os dados bloqueados, melhor que deixar a transação bloqueada enquanto aguarda a primeira terminar seu processamento Veja o exemplo a seguir:

Transação 1	Transação 2
Momento 1	Momento 2
BEGIN TRANSACTION UPDATE Funcionario SET Sal_Func = Sal_Func * 1.1 WHERE Cod_Func = 1	BEGIN TRANSACTION SELECT * FROM Funcionario --OK WITH (READPAST) WHERE Cod_Func = 1 UPDATE Funcionario --Wait SET Sal_Func = Sal_Func * 1.1_{23} WHERE Cod_Func = 1

Lock de sessão
(Nível de Isolamento de Transações)

- Dependendo da situação, você pode optar pelo lock de sessão para as suas transações.
  - READ COMMITED → DEFAULT
  - READ UNCOMMITED → NOLOCK(table)
  - SERIALIZABLE → HOLDLOCK(table)
  - REPEATABLE READ

Sem lock

Transação 1	Transação 2
Momento 1	Momento 2
BEGIN TRANSACTION SELECT * FROM Funcionario	Este comando UPDATE altera os dados da tabela funcionario, fazendo com que o segundo SELECT da primeira transação não devolva os mesmos dados da primeira. UPDATE Funcionario SET Sal_Func = Sal_Func * 1.1 WHERE Cod_Func = 1
SELECT * FROM Funcionario	

Lock REPEATABLE READ

Transação 1	Transação 2
Momento 1	Momento 2
Para garantir que outras transações não interferirão nesta primeira você poderia aplicar o lock de sessão. SET TRANSACTION ISOLATION LEVEL REPEATABLE READ BEGIN TRANSACTION SELECT * FROM Funcionario	Este comando UPDATE fica aguardando até que a primeira transação termine. UPDATE Funcionario --Wait SET Sal_Func = Sal_Func * 1.1 WHERE Cod_Func = 1
SELECT * FROM Funcionario	

Sem lock

Transação 1	Transação 2
Momento 1	Momento 2
BEGIN TRANSACTION UPDATE Funcionario SET Sal_Func = Sal_Func * 1.1	Apesar de o Update da primeira transação bloquear a tabela toda, ainda é possível inserir dados nessa tabela: INSERT Funcionario VALUES (....,5000)
Este select apresenta todos os dados afetados pelo update e uma linha a mais inserida pela segunda transação – essa linha a mais é chamada linha "fantasma". SELECT * FROM Funcionario	

Lock SERIALIZABLE

Transação 1	Transação 2
Momento 1	Momento 2
Para evitar os “FANTASMAS” na sua aplicação, configure este tipo de lock de sessão: SET TRANSACTION ISOLATION LEVEL SERIALIZABLE BEGIN TRANSACTION UPDATE Funcionario SET Sal_Func = Sal_Func * 1.1	Este Insert ficará aguardando até que a primeira transação termine. INSERT Funcionario VALUES (...,5000) /*WAIT */
SELECT * FROM Funcionario /* SELECT sem fantasma */	

DEAD LOCK

- Um Dead Lock é provocado pelo próprio SQL Server quando ocorrer uma situação de inanição. Esta situação acontece quando duas ou mais transações aguardam infinitamente por um recurso. Observe:

DEAD LOCK

Transação 1	Transação 2
Momento 1	Momento 1
BEGIN TRANSACTION SELECT * FROM Funcionario WITH (HOLDLOCK)	BEGIN TRANSACTION SELECT * FROM Funcionario WITH (HOLDLOCK)
MOMENTO 2	MOMENTO 2
Este Update ficaria aguardando a segunda transação terminar. UPDATE Funcionario SET Sal_Func = 1000	Este Update ficaria aguardando a segunda transação terminar. UPDATE Funcionario SET Sal_Func = 5000

Escolher ou não o Lock a ser utilizado?

- Sempre que possível, deixe que o SQL Server implemente o lock de acordo com as escolhas do otimizador, porque você pode provocar problemas de contenção, ou seja, você pode bloquear uma tabela por muito tempo e outras transações poderão ficar muito tempo aguardando.