Administração de banco de dados

Aula 07 - Alta concorrência em banco de dados

Gustavo Maia

Gustavo.Maia@FaculdadeImpacta.com.br

Sumário

- ACID – Transações e Locks
- Gerenciamento de Transações e Bloqueios
- Níveis de Isolamento de Transação
- Alta Concorrência - Overview
- Demonstração

Alta Concorrência - Overview

- Sistemas de banco de dados devem permitir que um grande número de conexões simultâneas faça uso de seus dados para leitura ou escrita de informações.

- Um dos objetivos dos sistemas de gerenciamento de banco de dados é fornecer a cada usuário a ilusão, sempre que possível, que é o único usuário no sistema.

- Sistemas de banco de dados lutam com a necessidade de equilibrar a consistência e simultaneidade. Muitas vezes há um trade-off ("perde-e-ganha") direto entre estes dois objetivos. O desafio é reduzir o impacto da concorrência, mantendo a consistência suficiente.

Alta Concorrência - Overview

- É da responsabilidade de um sistema de banco de dados fornecer mecanismos e funcionalidades de bloqueio que preservam isolamento da transação e garantam a integridade física de cada transação.

- Níveis de isolamento de transação são essenciais para minimizar o impacto de um usuário em outro.

- Este tópico explica como usar transações e os mecanismos de bloqueio para atender aos requisitos de desempenho e integridade de dados de seus aplicativos.

Alta Concorrência - Overview

- A funcionalidade central de sistemas de gerenciamento de banco de dados relacionais é fornecer a capacidade de agrupar um conjunto de mudanças que precisam ser feitas e garantir que irão ocorrer integralmente ou não. As transações são como estes requerimentos serão atendidos dentro do banco de dados.

ACID – Transações

- Uma transação é uma sequência de operações executadas como uma única unidade lógica de trabalho.

```
INSERT INTO Veiculo VALUES (1, 'Strada')
```

```
UPDATE Veiculo SET Descricao = 'Strada Adventure'
WHERE id = 1

INSERT INTO Veiculo VALUES (2, 'Montana')
INSERT INTO Veiculo VALUES (3, 'Saveiro')

DELETE Veiculo WHERE id = 1
```

ACID – Transações

- Transações devem apresentar quatro propriedades que são conhecidas como ACID, responsáveis por assegurar que várias modificações de dados são processadas como uma unidade. Por exemplo, uma transação bancária pode creditar uma conta e debitar outra. Ambas as etapas devem ser concluídas em conjunto ou não concluídas (desfazendo qualquer modificação realizada). SGBDs devem suportar o processamento de transações para gerenciar múltiplas transações.

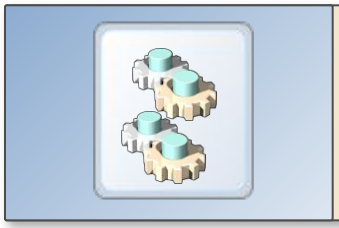
Atomicidade

Consistência

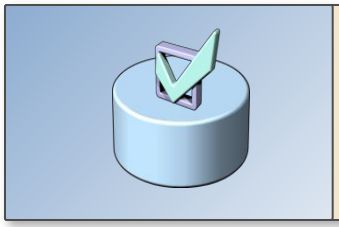
I solamento

Durabilidade

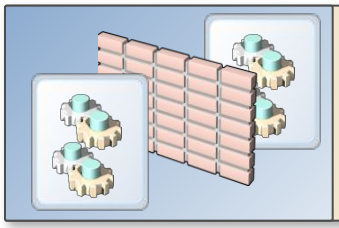
ACID – Transações



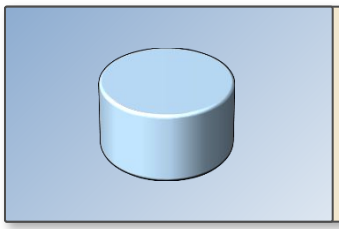
Uma transação é uma unidade de trabalho **Atômica**



Uma transação deixa os dados num estado **Consistente**



Uma transação é **Isolada** de outras transações correntes



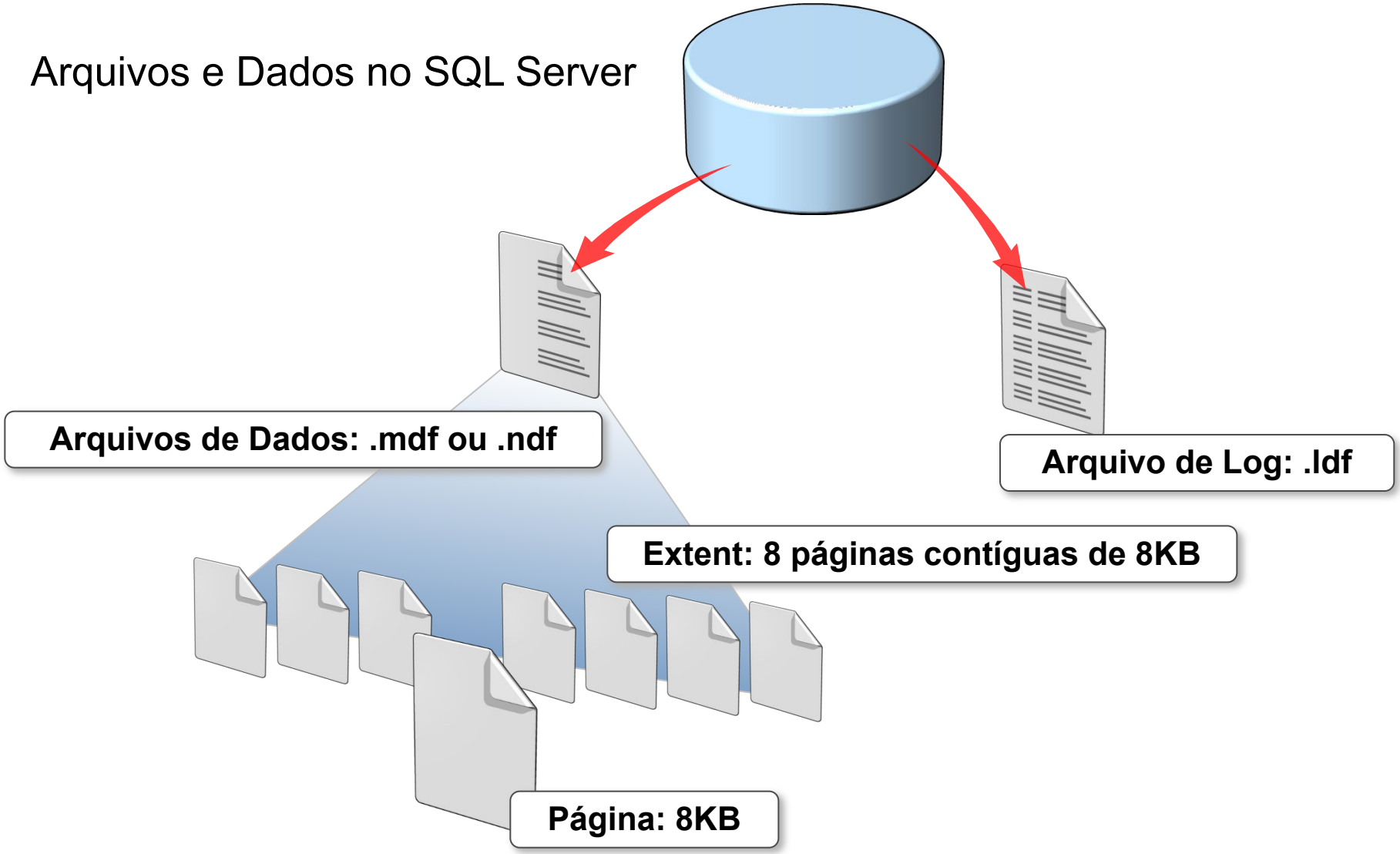
Uma transação é **Durável**

ACID – Transações

- Transações usam LOCK (travas) para impedir que outros usuários alterem ou leiam os dados em uma transação que não foi concluída.
- Locks são a reserva de recursos para serem utilizados em uma transação.
- O BLOCK (bloqueio) é necessário no processamento de transações online (OLTP) para sistemas multi-usuário.
- Blocks são causados quando mais uma uma transação deseja utilizar os recursos já reservados (LOCK) por outro processo.

Transações e Sistema de Gravação

- Arquivos e Dados no SQL Server



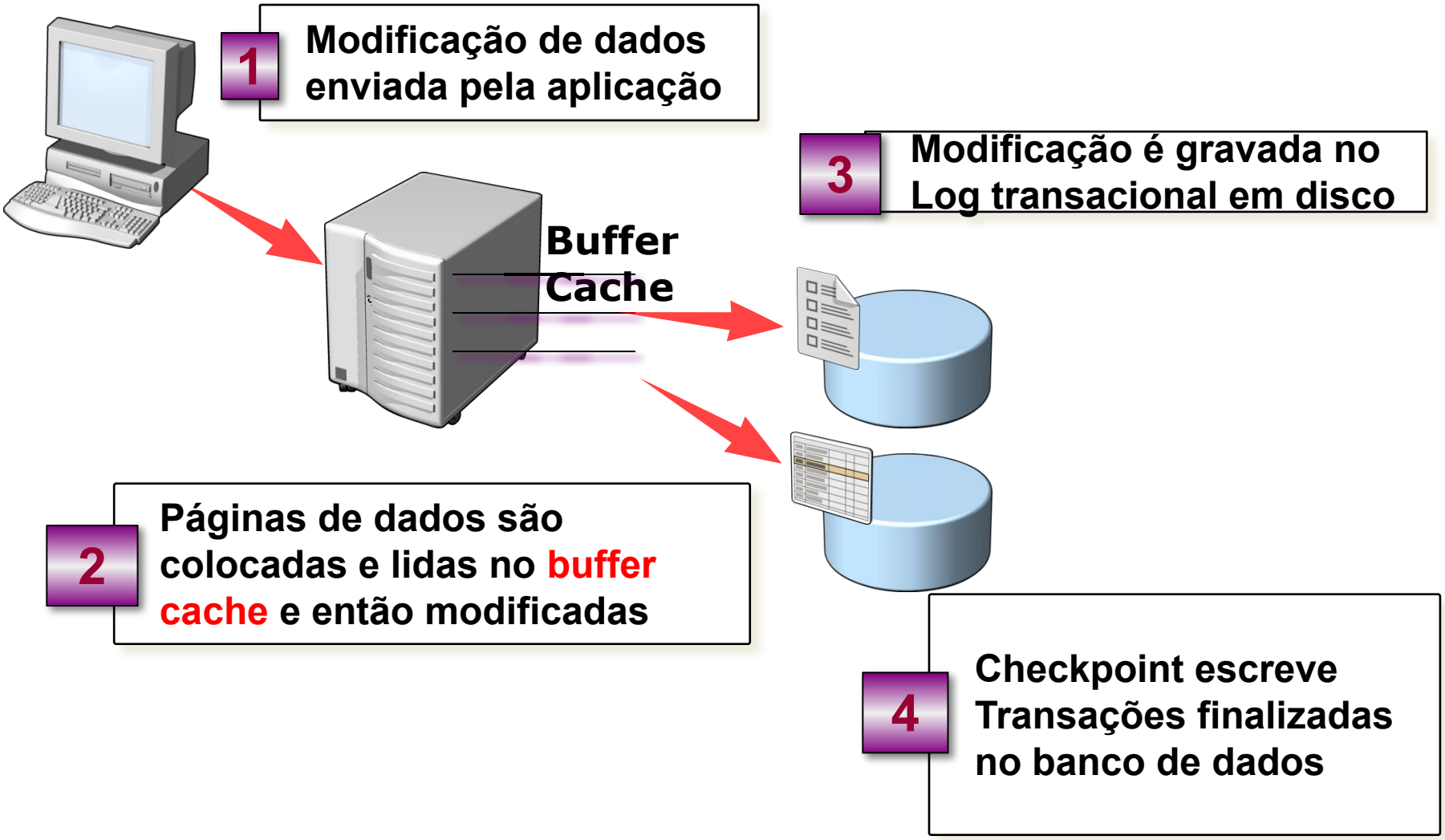
Transações e Sistema de Gravação

- Todas as transações são registradas em um log de transações para manter a consistência do banco de dados e para ajudar na recuperação da transação.

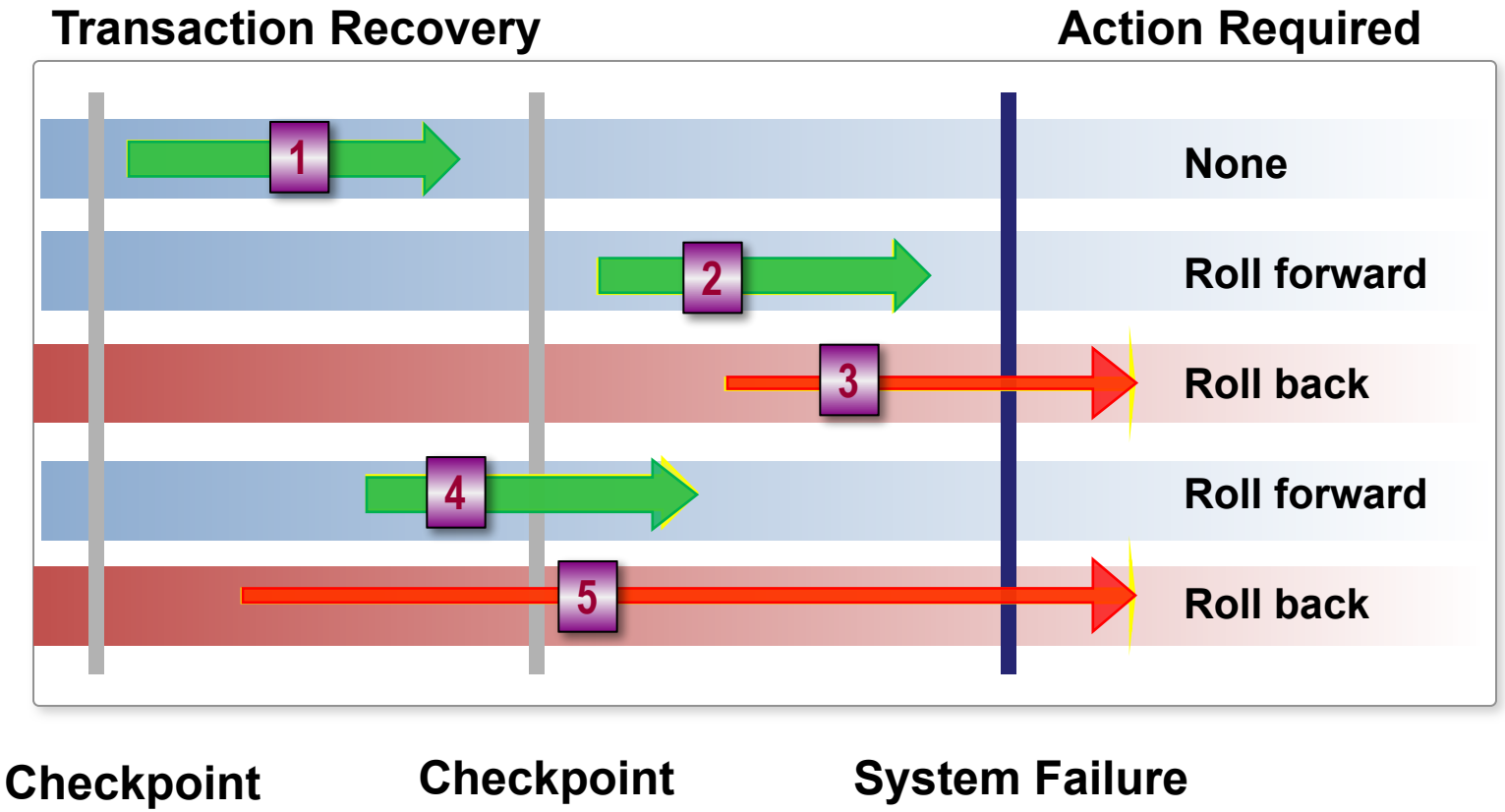
- Quando forem feitas alterações de dados, estas alterações são gravadas primeiramente no **log de transações** no disco e nas páginas de dados armazenadas em memória (em algum momento posterior, as páginas serão gravadas em disco no banco de dados (data files) – checkpoint).

- Se qualquer parte da transação falhar, todas as mudanças feitas são revertidas para deixar o banco de dados em seu estado original. Este sistema garante que as atualizações são completas e recuperáveis.

Transações e Sistema de Gravação



Recuperação de Transações



Transações

- Existem 3 tipos de transações
  - Explícitas - declaradas intencionalmente pelo usuário (manual)
  - Implícitas - abertas automaticamente pelo cliente / sessão.
  - Automáticas - abertas e gerenciadas pelo sistema.

Transação Explícita

É uma transação que é defina explicitamente através de um início e um final. Isso é feito com as declarações abaixo

- BEGIN TRANSACTION
- COMMIT TRANSACTION
- ROLLBACK TRANSACTION

```
BEGIN TRANSACTION FundsTransfer;
      EXEC Banking.DebitAccount '100', 'account1';
      EXEC Banking.CreditAccount '100', 'account2';
COMMIT TRANSACTION;
```

- SAVE TRANSACTION
- Podemos usar a opção WITH MARK para marcar um ponto específico no log transacional

Transação Explícita – Testes

```
SELECT * FROM   cliente
/*
  id  nome        cpf      telefone    professor
  ---  ------------------ -----------  -------------- -------------
  1  Almir dos Santos  78654552421  (11)91234-5678  0
*/
--Teste ROLLBACK
BEGIN TRANSACTION
     INSERT INTO cliente(nome,cpf,telefone,professor)
     SELECT 'Klaus Petherson','12345678901','11976531251',1
ROLLBACK TRANSACTION

SELECT * FROM   cliente
/*
  id  nome        cpf      telefone    professor
  ---  ------------------ -----------  -------------- -------------
  1  Almir dos Santos  78654552421  (11)91234-5678  0
  O que mais ele retornou ?
*/
```

Transação Explícita – Testes

```
SELECT * FROM   cliente
/*
  id  nome        cpf      telefone    professor
  ---  ------------------ -----------  -------------- -------------
  1  Almir dos Santos  78654552421  (11)91234-5678  0
*/
--Teste COMMIT
BEGIN TRANSACTION
     INSERT INTO cliente(nome,cpf,telefone,professor)
     SELECT 'Klaus Petherson','12345678901','11976531251',1
COMMIT TRANSACTION

SELECT * FROM   cliente
/*
  id  nome        cpf      telefone    professor
  ---  ------------------ -----------  -------------- -------------
  1  Almir dos Santos  78654552421  (11)91234-5678  0
  O que mais ele retornou ?
*/
```

Transação Explícita - Nested

- Observe que, embora a sintaxe T-SQL suporte aninhamento de transações, aninhamentos verdadeiros não ocorrem. A reversão de uma transação interna também reverte todas as transações externas e um rollback na transação mais externa irá desfazer qualquer transação confirmada interior.

```
begin transaction A
  insert Veiculo values ('palio'), ('corsa')

  begin transaction B
      insert Veiculo values ('Polo'), ('S10')
  rollback transaction B

commit transaction A
```

Modo de Confirmação – Auto Commit

- Auto Commit – Modo de transação Default
- Toda declaração TSQL é confirmada (**committed**) ou revertida (**rolled back**) quando for concluída. **Confirmada** se for bem sucedida e **revertida** em caso de erro.
- Motor de banco de dados opera em **autocommit** até que uma transação explícita seja iniciada.
- Erros de tempo de execução (**Run time errors**) podem permitir que parte do lote seja confirmado.
- XACT_ABORT configurado para ON converte declarações terminadas com erros de Run Time em erros de compilação, isso faz com que um simples erro em tempo de execução não seja confirmado dentro de um lote, cancelando assim toda a operação.

```
SET XACT_ABORT ON;
```

Transação Implícita

- Definindo o mode de transação implícita

```
SET IMPLICIT_TRANSACTIONS ON;
```

- Uma transação implícita começa quando uma das seguintes declarações é executada e a declaração não faz parte de uma transação existente

ALTER TABLE	INSERT
CREATE	OPEN
DELETE	REVOKE
DROP	SELECT
FETCH	TRUNCATE TABLE
GRANT	UPDATE

- Transação deve ser explicitamente concluída com COMMIT or ROLLBACK TRANSACTION

Transação Implícita

- Por padrão, o modo de transação implícita está desligado (default), o banco de dados funciona em modo de confirmação automática (auto commit).

- Transações aninhadas (onde uma transação é iniciada dentro de outra transação) não são permitidas no modo de transação implícita. Se a conexão já está em uma transação, essas declarações não iniciam uma nova transação.

- Na maioria dos casos, é melhor trabalhar no modo de confirmação automática e definir operações explicitamente usando a instrução BEGIN TRANSACTION. No entanto, para aplicações que foram originalmente desenvolvidas em diferentes sistemas, o modo de transação implícita desligado pode ser útil.

Considerações para uso de Transações

Há uma série de considerações gerais que precisam ser mantidas em mente quando se trabalha com transações.

- **Mantenha Transações mais curtas possível**
  - Transações mais longas aumentam a probabilidade de que os usuários não serão capazes de acessar os dados bloqueados.
  - Não exijam INPUT de usuários durante uma transação.
  - Não abra a transação enquanto navega através de dados, se possível. Transações não devem começar até que todas as análises de dados preliminares sejam concluídas.
  - INSERT, UPDATE e DELETE devem ser as primeiras declarações em uma transação e elas devem ser escritas para afetar o menor número de linhas.
  - Acesse a menor quantidade possível de dados durante uma transação. Isto diminui o número de linhas bloqueadas e reduz a contenção de dados.
  - Certifique-se de que a indexação está apropriada, pois isso reduz o número de páginas que precisam ser acessadas e bloqueadas.

Considerações para uso de Transações

- **Tente acessar recursos na mesma ordem**
  - Acessando recursos na mesma ordem dentro de transações tende a seriar naturalmente o seu acesso ao banco de dados e pode ajudar a evitar deadlocks (impasses). No entanto, isso nem sempre é possível.

Métodos de Controle de Concorrência

- Controle de concorrência é um sistema que administra muitas conexões que tentam acessar os mesmos dados ao mesmo tempo, de forma que uma solicitação não prejudique a outra.

- SQL Server suporta uma ampla gama de mecanismos de controle de concorrência otimista e pessimista. Programadores podem especificar o tipo de controle de concorrência, alterando o nível de isolamento de transação em uma conexão. Ele trabalha com dos dois tipos básicos de controle de concorrência: **pessimista e otimista.**

Métodos de Controle de Concorrência

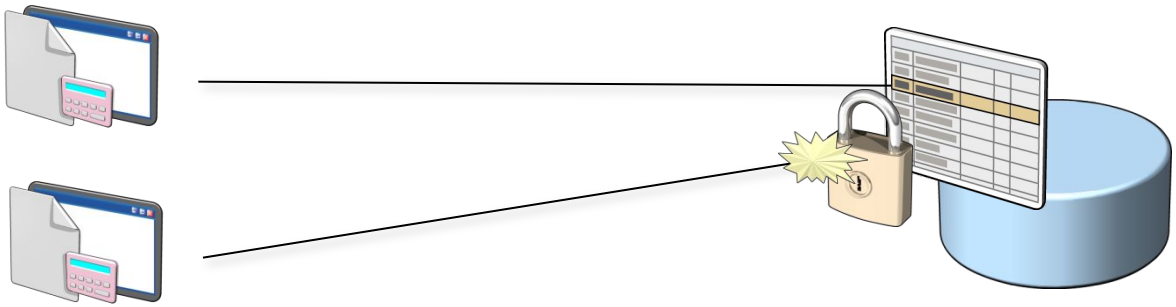
- **Pessimistic**
  - Controle de concorrência pessimista de dados lança locks (fechaduras) quando os dados são lidos na preparação para uma atualização**.** Outros usuários não podem executar ações que alterem os dados subjacentes até que o usuário que solicitou o lock, finalizar as atualizações nos dados. O controle de concorrência padrão do SQL Server é o pessimista.

- **Optimistic**
  - Controle de concorrência otimista não mantém locks sobre os dados quando os dados são inicialmente lidos. Em vez disso, quando uma atualização é realizada, o SQL Server verifica para determinar se os dados subjacentes foram alterados, desde quando foram inicialmente lidos. Se foram alterados, o usuário recebe um erro e a transação será revertida (rollback), tendo que a operação começar novamente. Este tipo funciona bem onde existe um baixo nível de contenção de dados.

Locks

- Antes de uma transação adquirir uma dependência sobre o estado atual de um conjunto de dados, como ler ou modificar os dados, ela deve se proteger contra os efeitos de outra transação modificar os mesmos dados. A transação faz isso, solicitando um lock (cadeado, trava, fechadura) na parte de dados que irá trabalhar.

- O bloqueio é um mecanismo usado pelo motor de banco de dados para sincronizar o acesso por vários usuários ao mesmo conjunto de dados ao mesmo tempo. É um mecanismo comum e necessário no modelo de bancos de dados transacionais.



Locks

- Existem vários tipos de locks, mais dois deles são os principais:
  - **Read Locks -** Permite que outras transações leiam os mesmos dados, mas não permite que gravem dados .
  - **Write Locks -** Impedem tanto a leitura ou escrita dos mesmos dados em outras transações.
- No SQL Server, esses bloqueios são implementados através de diferentes modos, como **compartilhado** ou **exclusivo** (shared or exclusive). O modo de bloqueio define o nível da dependência que a transação tem sobre os dados.

Locks

- Quando uma transação modifica uma parte dos dados, **o bloqueio protege a modificação até o final da transação**. O tempo que uma transação vai manter os bloqueios adquiridos para proteger as operações de leitura, **depende da definição do nível de isolamento da transação**. Todos os bloqueios mantidos por uma transação são liberados quando a **transação for concluída** (commit ou rollback).

- **Locks previnem conflitos de modificação de dados**

  - Locks garantem que as transações serão serializadas
  - Locks são lançados automaticamente pelo banco de dados
  - Locks habilitam o uso de dados de forma concorrente

ISOLAMENTO - Testes

```
/*Testes de isolamento*/
--Sessão 1
SELECT * FROM   cliente
/*
  id  nome         cpf       telefone    professor
  ---  ------------------ -----------  -------------- -------------
  1  Almir dos Santos  78654552421  (11)91234-5678  0
*/
BEGIN TRANSACTION
UPDATE cliente
     SET    nome = 'Joacir dos Santos'
     WHERE  nome = 'Almir dos Santos'

SELECT * FROM   cliente
/*
  id  nome         cpf       telefone    professor
  ---  ------------------ -----------  -------------- -------------
  1  ????? dos Santos  78654552421  (11)91234-5678  0
  Almir ou Joacir dos Santos ?
*/
```

ISOLAMENTO - Testes

```
/*Testes de isolamento*/
--Sessão 2
SELECT * FROM   cliente WITH(nolock)

/*
  id  nome        cpf      telefone    professor
  ---  ------------------ -----------  -------------- -------------
  1  ????? dos Santos  78654552421  (11)91234-5678  0
  Almir ou Joacir dos Santos ?
*/
SELECT * FROM   cliente
/*
  O que aconteceu ?
*/

/*
  O que é o WITH(NOLOCK) ?
*/
```

Blocking X Locking

- SQL Server (e muitos outros bancos de dados) faz uso intenso de bloqueios ao assegurar o isolamento entre os usuários e consistência das transações. É importante ter uma compreensão de como funciona o bloqueio e entender como locking difere de blocking.

- Os dois termos locking and blocking não são a mesma coisa e são frequentemente confundidos um com o outro.
  - **locking** é a ação de tomar e manter locks que é utilizado para implementar o controle de concorrência.
  - **blocking** é o que acontece com um processo, enquanto ele precisa esperar por um recurso que outro processo bloqueou (has locked).

- O bloqueio (blocking) é uma ocorrência normal para sistemas que usam travamento (locking). Apenas números excessivos de blocking é que se tornam um problema.

Recursos Bloqueáveis

RID	Um identificador de linha utilizado para bloquear uma única linha dentro de um heap
KEY	Um bloqueio de linha dentro de um índice usado para proteger intervalos de chaves em transações serializáveis
PAGE	Uma página (8KB), tais como dados ou páginas de índice
EXTENT	Grupo de 8 páginas contíguas, tais como dados ou páginas de índices.
HoBT	Uma heap or B-tree. Um lock para prevenir páginas de tabela heap ou índice B-tree (index) de uma tabela heap
TABLE	Uma tabela inteira, incluindo todos os dados e índices
FILE	Um arquivo de um banco de dados
APPLICATION	Um recurso especificado pelo aplicativo
METADATA	Metadata locks.
ALLOCATION_UNIT	Uma unidade de alocação
DATABASE	Todo o banco de dados

Para um desempenho ideal, o número de bloqueios que o SQL Server mantém deve ser equilibrado com a quantidade de dados que cada bloqueio mantém. Para minimizar o custo de bloqueio, SQL Server bloqueia automaticamente os recursos em um nível que é apropriado para a tarefa.

Tipos de Locks

Lock mode	Description
Shared (S)	Usado para operações de leitura
Update (U)	Usado em recursos que podem ser atualizados
Exclusive (X)	Usado para operações de modificação de dados como INSERT, UPDATE ou DELETE.
Intent	Usado para estabelecer uma hierarquia nas solicitações de lock
Schema	Usada quando uma operação depende do schema da tabela que está sendo executada
Bulk Update (BU)	Usado nacópia massiva de dados para uma tabela, quando o hint TABLOCK é especificado
Key-range	Protege um conjunto de linhas lidas por uma consulta ao usar o nível de isolamento serializável

Gerenciamento de Bloqueios

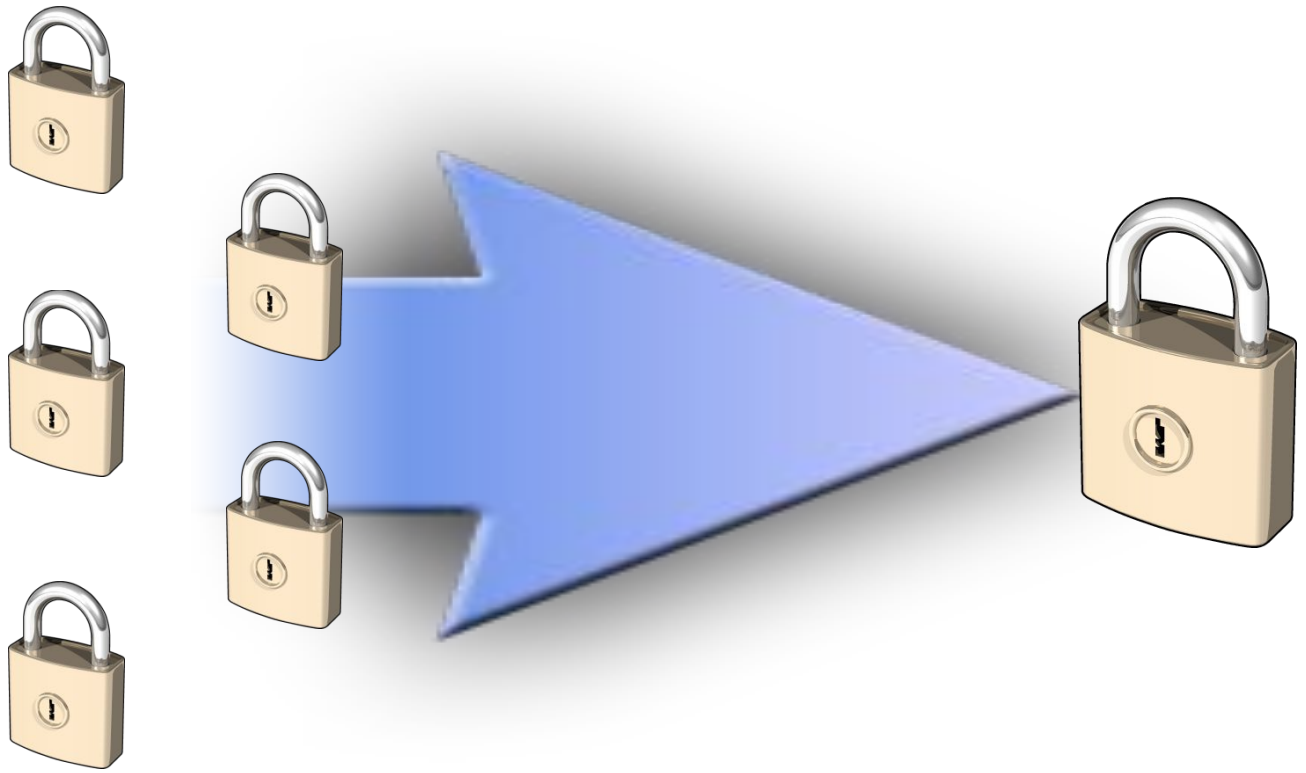
- Comportamento de bloqueio no SQL Server opera sem qualquer necessidade de uma gestão ou intervenção aplicação. No entanto, pode ser desejável em algumas situações exercer um controle sobre o comportamento de bloqueio.

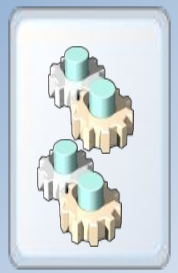
Gerenciamento de Bloqueios

- Aplicações se necessário esperam algum tempo por bloqueios mantidos por outras aplicações até serem liberados.
- SET LOCK_TIMEOUT especifica um número em milissegundos para espera
- -1 (default) aguarda eternamente
- Quando tempo limite expirar, o erro é retornado e a declaração revertida (rolled back)
- Muitas vezes não é usado, mas podem limitar o tempo de espera em consultas.
- READPAST Hint – Também disponível mas raramente utilizada.

```
SET LOCK_TIMEOUT 5000;
```

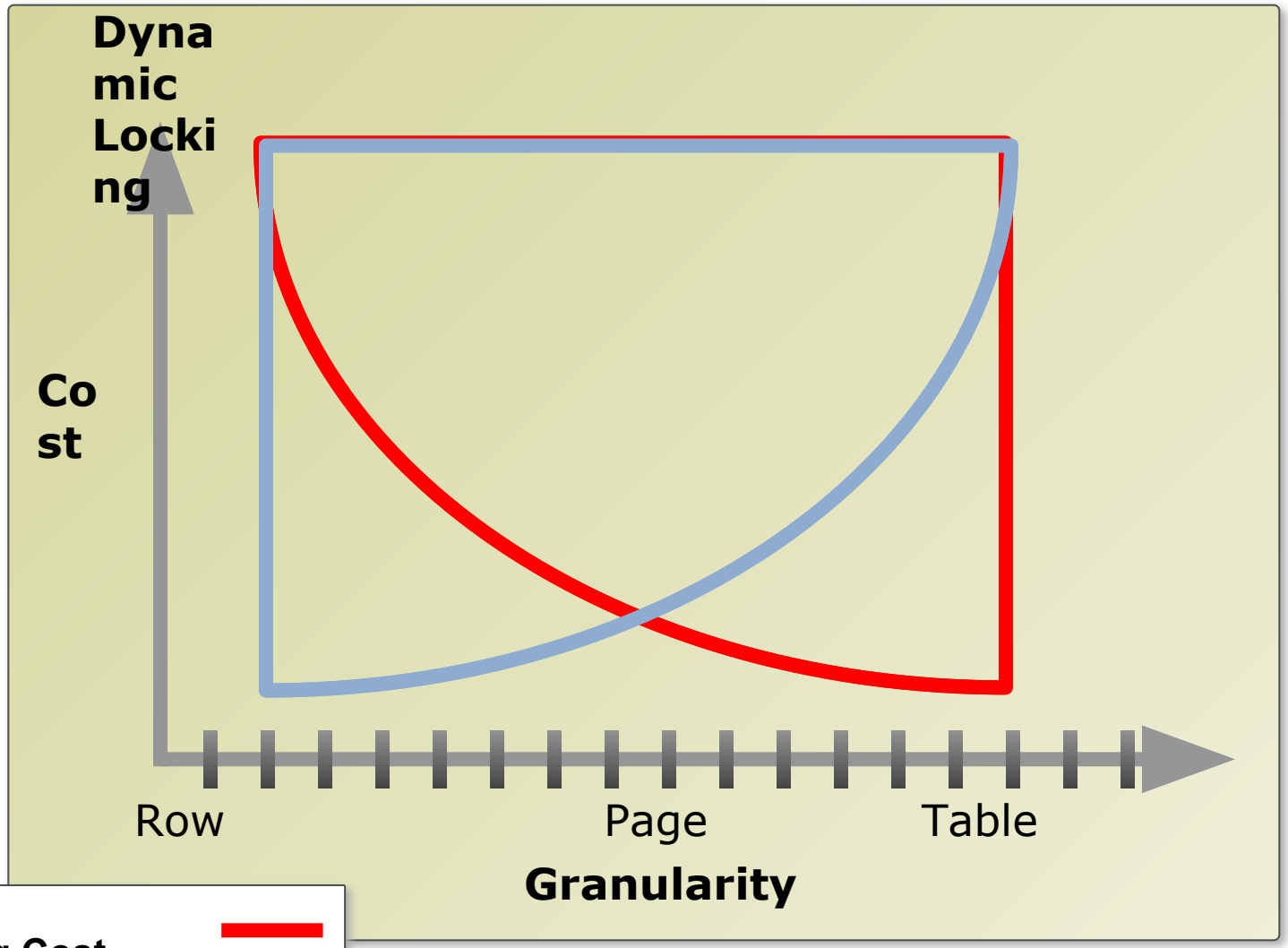
Escalação de Bloqueios





Escalação de bloqueio é o processo de conversão de muitos bloqueios de grão fino em bloqueios de grãos grossos, reduzindo a sobrecarga do sistema, aumentando a probabilidade de contenção da concorrência.

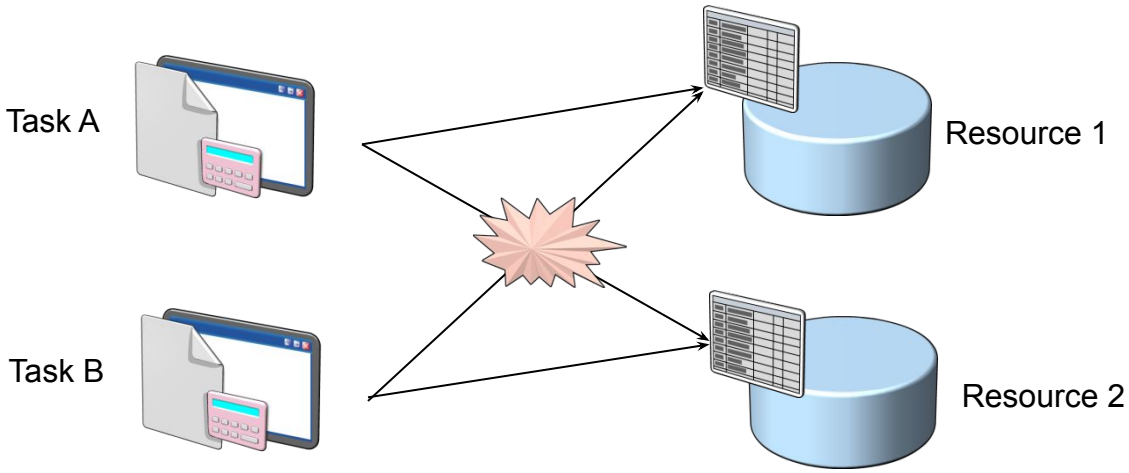
Escalação de Bloqueios



Escalação de Bloqueios

- **Deadlocks** (Impasses) ocorrem quando demandas por recursos não são capazes de ser resolvidas por espera de bloqueios que não são liberados, não importa quanto tempo os processos envolvidos esperem.

  - Task A holds a shared lock on Resource 1.
  - Task B holds a shared lock on Resource 2.
  - Task A requests an exclusive lock on Resource 2, but it cannot be granted until Task B releases the shared lock.
  - Task B requests an exclusive lock on Resource 1, but it cannot be granted until Task A releases the shared lock.
  - Neither task can continue and a deadlock state exists. SQL Server automatically detects this situation and raises an error 1205.



Níveis de Isolamento

- O conceito final e importante que precisa ser entendido quando se trabalha com a simultaneidade no SQL Server é o de níveis de isolamento de transação.

- Foi mencionado anteriormente que o papel de qualquer mecanismo de banco de dados é dar a cada usuário a melhor ilusão possível que eles são os únicos usuários do sistema.

- Isto não é totalmente possível, mas o uso apropriado de níveis de isolamento de transação podem ajudar nesse sentido.

Níveis de Isolamento

- Um nível de isolamento protege uma operação contra os efeitos de outras transações concorrentes. Use o nível de isolamento da transação para definir o nível de isolamento para todas as operações durante a **sessão**. Quando você define o nível de isolamento, você especifica o comportamento de **bloqueio padrão** para todas as declarações em sua **sessão**.

- A definição de níveis de isolamento de transação permitem aos programadores a aceitar o **aumento do risco de problemas de integridade** em troca de **maior acesso simultâneo aos dados**. Quanto **maior for o nível de isolamento**, **menor o risco de problemas de integridade de dados**. Este é o custo de **locks** sendo realizados por mais tempo e os próprios bloqueios serem mais restritivos com relação a transações concorrentes.

Níveis de Isolamento

Isolation Level	Dirty Read	Nonrepeatable Read	Phantoms
Read uncommitted	Yes	Yes	Yes
Read committed (default)	No	Yes	Yes
Repeatable read	No	No	Yes
Serializable	No	No	No
Snapshot	No	No	No

- Você pode substituir um nível de isolamento em nível de sessão usando a especificação de bloqueio. Você pode definir o nível de isolamento de transação para uma sessão usando a instrução SET.

SET TRANSACTION ISOLATION LEVEL {READ COMMITTED | READ UNCOMMITTED | REPEATABLE READ | SERIALIZABLE | SNAPSHOT}

Problemas de Concorrência

- Os usuários que modificam dados podem afetar outros usuários que estão lendo ou modificando os mesmos dados ao mesmo tempo. Se um sistema de armazenamento de dados não tem controle desta concorrência, os usuários podem ter efeitos secundários como:

  - **Atualizações perdidas (Lost Updates)** - ocorrem quando duas ou mais operações, selecionam a mesma linha e em seguida, atualizam a linha com base no valor selecionado originalmente. Cada transação não tem conhecimento das outras transações. A última atualização substitui atualizações feitas pelas outras transações, o que resulta em perda de dados.

  - **Leitura Suja (Dirty Read - Uncommitted Dependency)** - Ocorre quando uma segunda transação seleciona uma linha que está sendo atualizada por outra transação. A segunda operação está lendo dados que ainda não foram confirmados e podem ser alterados pela transação que está atualizando a linha.

Problemas de Concorrência

- **Análise Inconsistente (Non-Repeatable Read)** - Quando uma segunda transação acessa a mesma linha várias vezes e lê dados diferentes a cada vez. Análise inconsistente é semelhante à dependência descompromissada em que outra transação está mudando os dados que uma segunda transação está lendo. No entanto, na análise inconsistente, os dados lidos pela segunda transação foi confirmada pela transação que fez a alteração.

- **Leitura Fantasma (Phantom reads) - O**corre quando uma inserção ou exclusão é executada contra uma linha que pertence a uma série de linhas que está sendo lidas por uma transação. A primeira transação primeiro lê um intervalo de linhas que não existe mais na segunda leitura ou leituras seguintes mostram resultados de uma exclusão por uma transação diferente. Da mesma forma, a segunda transação ou sucessivas leituras da transação mostram uma linha que não existia na leitura original como o resultado de uma inserção por uma transação diferente.

- **Falta de leitura ou leitura dupla (causada por atualizações de linhas) -** Se outro usuário alterar a coluna de chave de índice da linha durante a sua leitura, a linha pode aparecer novamente se a mudança fundamental mudou a linha para uma posição à frente da sua digitalização. Da mesma forma, a linha pode não aparecer se a mudança fundamental mudou a linha para uma posição no índice que você já tinha lido.

Obrigado !

Gustavo Maia
Gustavo.Maia@FaculdadeImpacta.com.br