EAD

EDUCAÇÃO A DISTÂNCIA

UNIDADE III

Linguagem de Programação de Banco de Dados

Profa. Dra. Vanessa Lessa

Gerenciamento de transação

- O gerenciamento de transações é um assunto que merece atenção especial. Os primeiros sistemas de informação utilizavam arquivos sequenciais indexados ou arquivos de acesso direto para guardar e atualizar seus dados, esses mecanismos de armazenamento e consulta de dados eram limitados e não possuíam recurso de gerenciamento como encontramos atualmente nos sistemas de bancos de dados.
- Com o início dos sistemas multiusuário surgiu o problema da concorrência pelos dados e era imprescindível administrar todos os acessos, consultas e atualizações que ocorriam simultaneamente. Todos esses acessos poderiam gerar vários problemas nos dados, como por exemplo: garantir o isolamento de cada transação, isto é, não deixar que uma transação interfira na outra podendo ocasionar atualizações incorretas resultando na inconsistência dos dados; além disto, garantir que cada transação fosse atômica, isto é, todas as transações deveriam ser concluídas integralmente ou não serem efetivadas, pois poderiam causar inconsistência nos dados.

Transações

- Uma transação é uma unidade lógica de operações de banco de dados. Uma transação pode consistir em uma ou mais operações de banco de dados que podem ser lidas ou atualizadas, podendo ser inserção (INSERT), atualização (UPDATE) ou exclusão (DELETE). As operações de banco de dados que compõem uma transação podem estar embutidas nos programas ou podem ser executadas interativamente usando SQL em um programa de banco de dados. Em outras palavras, as transações podem ser enviadas para um SGBD por programas aplicativos ou programas de banco de dados e devem ser processadas de forma apropriada.
    - Uma maneira de caracterizar uma transação é especificar o início e fim da transação, os limites da transação com a utilização das instruções <BEGIN TRANSACTION> e <END TRANSACTION>. Dessa forma, todas as operações de banco de dados entre essas instruções são reconhecidas como parte da transação.

Transações

- Toda transação deve respeitar as quatro propriedades que garantem o correto funcionamento do banco de dados e impedem que os dados sejam perdidos ou corrompidos no processamento, essas propriedades são conhecidas pela sigla ACID:
- **Atomicidade –** Cada transação deve ser atômica, o que significa que não pode ser dividida ou fragmentada. A ideia é que todas as operações do banco de dados que o compõem sejam executadas como se fossem uma única operação. Se qualquer operação de banco de dados falhar, toda a transação deverá ser desfeita. Depois que todas as operações do banco de dados que compõem a transação forem concluídas com êxito, a transação poderá ser confirmada.
- **Consistência –** Cada transação deve deixar o banco de dados em um estado consistente após a execução. Isso significa que a transação deve atender a todas as regras e restrições definidas no banco de dados, que incluem regras de integridade referencial, regras de domínio para valores de coluna permitidos, definição de chave primária, índices exclusivos e colunas obrigatórias.

Transações

- **Isolamento –** Cada transação deve ser isolada de outras transações no banco de dados. Os resultados parciais de cada transação não devem estar disponíveis para outras transações. A ideia é que nenhuma transação possa atrapalhar a execução de outra transação no mesmo banco de dados.

- **Durabilidade –** Cada transação tem resultados permanentes no banco de dados e só pode ser desfeita na próxima transação.

Estado da transação

- Uma transação nem sempre completa sua execução com sucesso e quando isso acontece a transação é considerada abortada. Se tivermos de assegurar a propriedade de atomicidade, uma transação abortada não deverá mudar o estado do banco de dados. Dessa forma, quaisquer mudanças que a transação abortada fez no banco de dados deverá ser desfeita. No momento que todas as mudanças causadas por uma transação abortada tiverem sido desfeitas, diremos que a transação foi revertida (*rolled back*).

- Uma transação cuja execução foi concluída com sucesso é considerada confirmada (*committed*). A transação que fez as atualizações altera o banco de dados para um novo estado persistente que deve ser preservado mesmo se o sistema falhar.

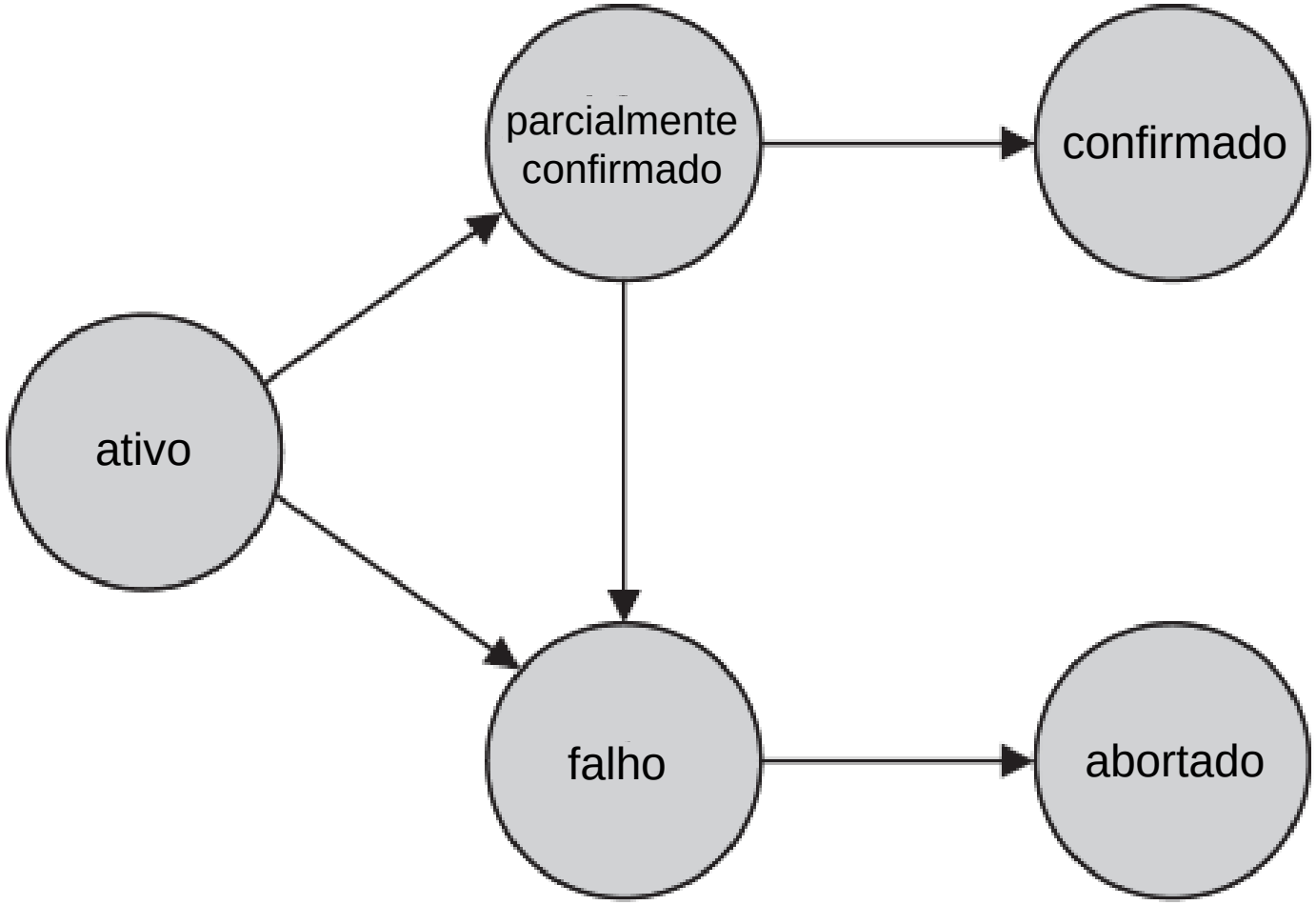
Estado da transação

Devemos ser mais específicos sobre o que significa para uma transação ser bem-sucedida. Então, vamos criar um modelo simples de classificação de transação em que analisaremos alguns estados:

- **Ativo**: o estado inicial, a transação permanece nesse estado enquanto está executando.
- **Parcialmente confirmado**: depois que a instrução final foi executada.
- **Falho**: depois da descoberta de que a execução não pode mais prosseguir.
- **Abortado**: depois que a transação foi revertida e o banco de dados foi restaurado ao seu estado anterior ao início da transação.
- **Confirmado**: após o término bem-sucedido.

Estado da transação

- Baseando-se nos possíveis estados de uma transação apresentamos o diagrama de estado correspondente a uma transação.



Fonte: Silberschatz (2020, p. 455).

Interatividade

Analise as afirmações:

I. Uma transação é uma unidade lógica de operações de banco de dados.

II. Cada transação deve ser atômica, o que significa que não pode ser dividida ou fragmentada.

III. Cada transação deve deixar o banco de dados em um estado consistente após a execução.

IV. Os resultados parciais de cada transação devem estar disponíveis para outras transações.

V. Uma transação sempre completa sua execução com sucesso.

Estão corretas:

a) I, II e III.

b) III e IV.

c) I, II e V.

d) III, IV e V.

e) Todas as afirmações.

Resposta

Analise as afirmações:

I. Uma transação é uma unidade lógica de operações de banco de dados.

II. Cada transação deve ser atômica, o que significa que não pode ser dividida ou fragmentada.

III. Cada transação deve deixar o banco de dados em um estado consistente após a execução.

IV. Os resultados parciais de cada transação devem estar disponíveis para outras transações.

V. Uma transação sempre completa sua execução com sucesso.

Estão corretas:

a) I, II e III.

b) III e IV.

c) I, II e V.

d) III, IV e V.

e) Todas as afirmações.

Execuções simultâneas

- Os sistemas de processamento de transações geralmente permitem que várias transações sejam processadas ao mesmo tempo. Permitir que várias transações atualizem dados simultaneamente causa alguns problemas de consistência de dados. Assegurar a consistência apesar de executar as transações simultaneamente requer trabalho adicional; é muito mais simples exigir que as transações sejam executadas sequencialmente, iniciando cada uma somente após a conclusão da anterior.

Execuções simultâneas

- O sistema de banco de dados deve controlar a interação entre transações concorrentes para que não destruam a consistência do banco de dados. Ele faz isso por meio de vários mecanismos chamados sistemas de controle concorrência.
- Considere o cadastro de uma conta bancária, e um conjunto de transações que acessam e atualizam essas contas. Considere que T1 e T2 sejam duas transações que movimentam os saldos dessas contas. A transação T1 transfere 500 da conta C1 para a conta C2.

A transação T2 transfere 20% do saldo da conta C1 para a conta C2. Ela é definida como:

T1:
read(C1);
C1 := C1 – 500;
write(C1);
read(C2);
C2 := C2 + 500;
write(C2).

T2:
read(C1);
temp := C1 * 0.2;
C1 := C1 – temp;
write(C1);
read(C2);
C2 := C2 + temp;
write(C2).

Execuções simultâneas

Suponha que os valores atuais das contas C1 e C2 sejam 1000 e 2000, respectivamente. Suponha, também, que as duas transações sejam executadas uma de cada vez na ordem T1 e em seguida T2. a sequência de etapas de instrução deve estar em ordem cronológica de cima para baixo, com as instruções de T1 aparecendo na coluna esquerda e as instruções de T2 aparecendo na coluna direita. Os valores finais das contas C1 e C2, após a execução são 400 e 2600, respectivamente. A quantia nas contas C1 e C2 – ou seja, a soma C1 + C2 – é preservada depois da execução das duas transações. O Schedule 1 serial que apresenta a sequência de execução T1 e T2:

T1:	**T2:**
read(C1);	
C1 := C1 – 500;	
write(C1);	
read(C2);	
C2 := C2 + 500;	
write(C2).	
commit	
	read(C1);
	temp := C1 * 0.2;
	C1 := C1 – temp;
	write(C1);
	read(C2);
	C2 := C2 + temp;
	write(C2).
	commit

Execuções simultâneas

- De modo semelhante, se as transações forem executadas uma de cada vez na ordem T2 seguida por T1. Conforme esperamos, a soma C1 + C2 é preservada, e os valores finais das contas C1 e C2 agora são 300 e 2700, respectivamente. Schedule 2: T2 seguido de T1.

T1:	**T2:**
	read(C1);
	temp := C1 * 0.2;
	C1 := C1 – temp;
	write(C1);
	read(C2);
	C2 := C2 + temp;
	write(C2).
	commit
read(C1);	
C1 := C1 – 500;	
write(C1);	
read(C2);	
C2 := C2 + 500;	
write(C2).	
commit	

Execuções simultâneas

- Vamos supor que as duas transações sejam executadas simultaneamente. Depois que essa execução acontece, chegamos ao mesmo estado daquele em que as transações são executadas em série na ordem T1 seguida por T2. A soma C1 + C2 é realmente preservada. Schedule 3: um Schedule simultâneo equivalente ao Schedule 1.

T1:	**T2:**
read(C1);	
C1 := C1 – 500;	
write(C1);	
	read(C1);
	temp := C1 * 0.2;
	C1 := C1 – temp;
	write(C1);
read(C2);	
C2 := C2 + 500;	
write(C2).	
commit	
	read(C2);
	C2 := C2 + temp;
	write(C2).
	commit

Execuções simultâneas

- Nem todas as execuções simultâneas resultam em um estado correto. Depois da execução desse schedule, chegamos a um estado em que os valores finais das contas C1 e C2 são 500 e 2200, respectivamente. Esse estado final é um estado inconsistente, pois perdemos 300 no processo da execução simultânea. De fato, a soma C1 + C2 não é preservada pela execução das duas transações. Schedule 4 – um Schedule simultâneo resultando em um estado inconsistente.

T1:	**T2:**
read(C1);	
C1 := C1 – 500;	
	read(C1);
	temp := C1 * 0.2;
	C1 := C1 – temp;
	write(C1);
	read(C2);
write(C1);	
read(C2);	
C2 := C2 + 500;	
write(C2).	
commit	
	C2 := C2 + temp;
	write(C2).
	commit

Facilidade de recuperação

- Considere o schedule parcial 5, em que T4 é uma transação que realiza apenas uma instrução: read(C1). Denominamos isso de schedule parcial, pois não incluímos uma operação commit ou abort para T3. Observe que T4 é confirmada imediatamente após a execução da instrução read(C1). Assim, T4 é confirmada enquanto T3 ainda está no estado ativo. Agora, suponha que T3 falhe antes de ser confirmada. Como T4 leu o valor do item de dados C1 escrito por T3, dizemos que T4 é dependente de T3. Por causa disso, temos de abortar T4 para garantir a atomicidade. Porém, T4 já foi confirmada e não pode ser abortada. Assim, temos uma situação em que é impossível recuperar-se corretamente da falha de T3.

T3:	**T4:**
read(C1);	
write(C1);	
	read(C1);
	commit
read(C2);	

Facilidade de recuperação

- O schedule 5 é um exemplo de schedule não recuperável. Um schedule recuperável é aquele em que, para cada par de transações Ti e Tj tal que Tj leia um item de dados previamente escrito por Ti, a operação commit de Ti apareça antes da operação commit de Tj. Para que o exemplo do schedule 5 seja recuperável, T4 teria de adiar a confirmação até que T3 fosse confirmada.

T3:	**T4:**
read(C1);	
write(C1);	
	read(C1);
	commit
read(C2);	

Implementação do isolamento

- Existem várias formas de controle de concorrência que podemos utilizar para assegurar que, mesmo quando diversas transações são executadas ao mesmo tempo, somente os schedules aceitáveis sejam gerados, independentemente de como o sistema operacional realize o compartilhamento de tempo dos recursos entre as transações.

- Considere este padrão como um exemplo trivial de um esquema de controle de concorrência: uma transação bloqueia todo o banco de dados antes da execução e libera o bloqueio após a execução. Mesmo que uma transação retenha o bloqueio, nenhuma outra transação tem permissão para adquirir o bloqueio e, portanto, todos devem esperar que o bloqueio seja liberado. Como resultado do sistema de bloqueio, apenas uma transação pode ser executada por vez. Portanto, apenas schedules seriais são criados. Eles são organizados trivialmente em conjuntos e é fácil verificar que também são schedules não em cascata.

Implementação do isolamento

- Esse esquema de controle de simultaneidade leva a um desempenho ruim porque força as transações a esperar até que as transações anteriores sejam concluídas antes de poderem começar. Em outras palavras, oferece menor concorrência.

- Os métodos de controle de simultaneidade visam garantir um alto nível de concorrência e, garantindo que todos os schedules criados não sejam serializáveis por conflito ou visão, e não em cascata.

Interatividade

Analise as afirmações:

I. Os sistemas de processamento de transações geralmente permitem que várias transações sejam processadas ao mesmo tempo.
II. O sistema de banco de dados deve controlar a interação entre transações concorrentes para que não destruam a consistência do banco de dados.
III. Uma transação pode bloquear o BD antes da execução e liberar o bloqueio após a execução.
IV. Os métodos de controle de simultaneidade visam garantir um alto nível de concorrência, garantindo que todos os schedules criados não sejam serializáveis por conflito ou visão.
V. Todas as execuções simultâneas resultam em um estado correto.

Estão corretas:

a) I, II e III.
b) I, II e IV.
c) I, II e V.
d) I, II, III e IV.
e) Todas as afirmações.

Resposta

Analise as afirmações:

I. Os sistemas de processamento de transações geralmente permitem que várias transações sejam processadas ao mesmo tempo.
II. O sistema de banco de dados deve controlar a interação entre transações concorrentes para que não destruam a consistência do banco de dados.
III. Uma transação pode bloquear o BD antes da execução e liberar o bloqueio após a execução.
IV. Os métodos de controle de simultaneidade visam garantir um alto nível de concorrência, garantindo que todos os schedules criados não sejam serializáveis por conflito ou visão.
V. Todas as execuções simultâneas resultam em um estado correto.

Estão corretas:

a) I, II e III.
b) I, II e IV.
c) I, II e V.
d) I, II, III e IV.
e) Todas as afirmações.

Controle de Concorrência

- Uma das principais características de uma transação é o isolamento. No entanto, se várias transações forem executadas simultaneamente no banco de dados, o atributo de isolamento não poderá mais ser mantido. Para garantir isso, o sistema deve monitorar as interações entre transações concorrentes; esse controle é obtido por meio de diversos mecanismos chamados sistemas de controle simultâneos.

- Existe uma variedade de esquemas de controle de concorrência. Nenhum é especificamente o melhor; cada um tem suas vantagens. Na prática, os dois esquemas mais frequentemente usados são o bloqueio de duas fases e o isolamento baseado em instantâneo (também simplesmente chamado de isolamento de instantâneo, ou snapshot isolation).

Protocolos baseados em bloqueio

- Uma maneira de garantir o isolamento é exigir que os itens de dados sejam usados de maneira mutuamente exclusiva, ou seja, se uma transação usa um objeto de dados, nenhuma outra transação pode modificar esse objeto de dados. A maneira mais comum de implementar esse requisito é permitir que uma transação acesse um objeto de dados somente se estiver atualmente bloqueado nesse objeto.

Bloqueios (locks)

1. **Compartilhado.** Se uma transação Ti tiver obtido um bloqueio no modo compartilhado (indicado por S do inglês, shared) sobre o item Q, então Ti pode ler, mas não pode escrever Q.
2. **Exclusivo.** Se uma transação Ti tiver obtido um bloqueio no modo exclusivo (indicado por X do inglês, exclusive) sobre o item Q, então Ti pode ler e escrever Q.

Protocolos baseados em bloqueio

- Em geral, dado um conjunto de modos de bloqueio, podemos definir uma função de compatibilidade para eles da seguinte forma. Deixe C1 e C2 representar qualquer forma de inibição. Suponha que o evento Ti solicite um bloqueio de modo C1 para o destino Q, em que o evento Tj (Ti ≠ Tj) atualmente detém um bloqueio de modo C2. Se o evento Ti pode aceitar imediatamente o bloqueio Q, apesar da existência do bloqueio no estado C2, então o estado C1 corresponde ao estado C2. Esta função pode ser convenientemente representada como uma matriz.

	S	X
S	true	false
X	false	false

Fonte: Silberschatz (2020, p. 471).

Protocolos baseados em bloqueio

- Considere novamente o sistema bancário simplificado em que C1 e C2 sejam duas contas que são acessadas pelas transações T5 e T6. A transação T5 transfere 50 da conta C2 para a conta C1. A transação T6 apresenta a quantia total nas contas C1 e C2 – ou seja, a soma C1 + C2.

T5:
lock-X(C2);
read(C2);
C2 := C2 – 50;
write(C2);
unlock(C2);
lock-X(C1);
read(C1);
C1 := C1 + 50;
write(C1);
unlock(C1);

T6:
lock-S(C1);
read(C1);
unlock(C1);
lock-S(C2);
read(C2);
unlock(C2);
display(C1+C2)

Protocolos baseados em bloqueio

- Suponha que os valores das contas C1 e C2 sejam 100 e 200, respectivamente. Se essas duas transações forem executadas em série ou na ordem T5, T6 ou na ordem T6, T5, então a transação T6 mostrará o valor 300. No entanto, se essas transações forem executadas simultaneamente, o schedule 6 é possível.

T5:
lock-X(C2);
read(C2);
C2 := C2 – 50;
write(C2);
unlock(C2);
lock-X(C1);
read(C1);
C1 := C1 + 50;
write(C1);
unlock(C1);

T6:
lock-S(C1);
read(C1);
unlock(C1);
lock-S(C2);
read(C2);
unlock(C2);
display(C1+C2)

Protocolos baseados em bloqueio

- Nesse caso, a transação T6 mostra 250, que é incorreto. O motivo para esse engano é que a transação T5 desbloqueou o item de dados C2 muito cedo e, como resultado, T6 viu um estado inconsistente.

T5:	T6:	Gerenciador de controle de concorrência
lock-X(C2);		
		grant-X(C2,T5)
read(C2);		
C2 := C2 – 50;		
write(C2);		
unlock(C2);		
	lock-S(C1);	
		grant-S(C1,T6)
	read(C1);	
	unlock(C1);	
	lock-S(C2);	
		grant-X(C2,T6)
	read(C2);	
	unlock(C2);	
	display(C1+C2)	
lock-X(C1);		
		grant-X(C1,T5)
read(C1);		
C1 := C1 + 50;		
write(C1);		
unlock(C1);		

Protocolos baseados em bloqueio

- Vamos supor que o desbloqueio seja adiado para o final da transação. A transação T7 corresponde a T5 com desbloqueio adiado. A transação T8 corresponde a T6 com o desbloqueio adiado.

T7:
lock-X(C2);
read(C2);
C2 := C2 – 50;
write(C2);
lock-X(C1);
read(C1);
C1 := C1 + 50;
write(C1);
unlock(C2);
unlock(C1);

T8:
lock-S(C1);
read(C1);
lock-S(C2);
read(C2);
display(C1+C2)
unlock(C1);
unlock(C2);

Protocolos baseados em bloqueio

- Infelizmente, o bloqueio pode levar a uma situação indesejável. Considere o schedule parcial para T7 e T8. Como T7 está mantendo um bloqueio no modo exclusivo sobre C2, e T8 está solicitando um bloqueio no modo compartilhado sobre C2, T8 está esperando que T7 desbloqueie C2. De modo semelhante, como T8 está mantendo um bloqueio no modo compartilhado sobre C1, e T7 está solicitando um bloqueio no modo exclusivo sobre C1, T7 está esperando que T8 desbloqueie C1. Essa situação é chamada de impasse (deadlock). Quando ocorre impasse, o sistema precisa reverter uma das duas transações. Quando uma transação tiver sido revertida, os itens de dados que estavam bloqueados por essa transação são desbloqueados. Esses itens de dados, então, ficam disponíveis para a outra transação, que pode continuar com sua execução.

T7:	**T8:**
lock-X(C2);	
read(C2);	
C2 := C2 – 50;	
write(C2);	
	lock-S(C1);
	read(C1);
	lock-S(C2);
lock-X(C1);	

Protocolos baseados em bloqueio

Concessão de bloqueios (granting of locks)

- Suponha que uma transação T6 tenha um bloqueio no modo compartilhado sobre um item de dados, e outra transação T5 solicite um bloqueio no modo exclusivo sobre o item de dados. T5 tem de esperar que T6 libere o bloqueio no modo compartilhado. Nesse meio-tempo, uma transação T7 pode solicitar um bloqueio no modo compartilhado sobre o mesmo item de dados. A solicitação de bloqueio é compatível com o bloqueio concedido a T6, de modo que T7 pode receber o bloqueio no modo compartilhado. Nesse ponto, T6 pode liberar o bloqueio, mas ainda T5 precisa esperar que T7 termine. Novamente, porém, pode haver uma nova transação T8 que solicite um bloqueio no modo compartilhado sobre o mesmo item de dados, recebendo o bloqueio antes que T7 o libere. De fato, é possível que haja uma sequência de transações em que cada uma solicite um bloqueio no modo compartilhado sobre o item de dados, e cada transação libere o bloqueio pouco depois que ele foi concedido, mas T5 nunca receba o bloqueio no modo exclusivo sobre o item de dados. A transação T5 pode nunca fazer progresso e é considerada estagnada.

Protocolos baseados em bloqueio

Podemos evitar a estagnação de transações concedendo bloqueios da seguinte maneira: no momento em uma transação Ti solicita um bloqueio sobre um item de dados Q em um modo específico M, o gerenciador de controle de concorrência concede o bloqueio desde que:

- Não haja outra transação mantendo um bloqueio sobre Q em um modo que entre em conflito com M.

- Não exista outra transação que esteja esperando por um bloqueio sobre Q e que fez sua solicitação de bloqueio antes de Ti.

Protocolos baseados em bloqueio

Protocolo de bloqueio em duas fases

Um protocolo que garante a serializabilidade é o protocolo de bloqueio em duas fases. Esse protocolo requer que cada transação envie solicitações de bloqueio e desbloqueio em duas fases:

1. **Fase de crescimento.** Uma transação pode obter bloqueios, mas não pode liberar nenhum bloqueio.

2. **Fase de encolhimento.** Uma transação pode liberar bloqueios, mas não pode obter novos bloqueios.

Protocolos baseados em bloqueio

- De acordo com o bloqueio em duas fases, pode ocorrer rollback em cascata. Cada transação observa o protocolo de bloqueio em duas fases, mas a falha de T10 após a etapa read(A) de T12 leva a um rollback em cascata de T11 e T12.

T10:	**T11:**	**T12:**
lock-X(C1);		
read(C1);		
lock-S(C2);		
read(C2);		
write(C1);		
unlock(C1);		
	lock-X(C1);	
	read(C1);	
	write(C1);	
	unlock(C1);	
		lock-S(C1);
		read(C1);

Interatividade

Analise as afirmações:

I. Se várias transações forem executadas simultaneamente no banco de dados, o atributo de isolamento não poderá mais ser mantido.

II. Uma maneira de garantir o isolamento é exigir que os itens de dados sejam usados de maneira mutuamente exclusiva.

III. Na fase de crescimento uma transação pode liberar bloqueios, mas não pode obter novos bloqueios.

IV. Na fase de encolhimento uma transação pode obter bloqueios, mas não pode liberar nenhum bloqueio.

V. De acordo com o bloqueio em duas fases, pode ocorrer rollback em cascata.

Estão corretas:

a) I, II e III.
b) I, II e V.
c) II, III e V.
d) III, IV e V.
e) Todas as afirmações.

Resposta

Analise as afirmações:

I. Se várias transações forem executadas simultaneamente no banco de dados, o atributo de isolamento não poderá mais ser mantido.

II. Uma maneira de garantir o isolamento é exigir que os itens de dados sejam usados de maneira mutuamente exclusiva.

III. Na fase de crescimento uma transação pode liberar bloqueios, mas não pode obter novos bloqueios.

IV. Na fase de encolhimento uma transação pode obter bloqueios, mas não pode liberar nenhum bloqueio.

V. De acordo com o bloqueio em duas fases, pode ocorrer rollback em cascata.

Estão corretas:

a) I, II e III.

b) I, II e V.

c) II, III e V.

d) III, IV e V.

e) Todas as afirmações.

Protocolos baseados em timestamp

- Os protocolos baseados em timestamp determinam a ordem entre cada par de transações em conflito em tempo de execução começando a partir do primeiro bloqueio envolvendo modos incompatíveis que os dois membros do par solicitam. Outro método de determinar a ordem de serializabilidade consiste em selecionar uma ordenação das transações com antecedência. O método mais comum para fazer isso é usar um esquema de ordenação por timestamp.

Protocolos baseados em timestamp

Timestamps

A cada transação Ti no sistema, associamos um timestamp fixo exclusivo, indicado por TS(Ti). Esse timestamp (registro de data e hora) é atribuído pelo sistema de banco de dados antes que a transação Ti inicie sua execução. Se uma transação Ti tiver recebido o timestamp TS(Ti), e uma nova transação Tj entrar no sistema, então TS(Ti) < TS(Tj). Existem dois métodos para elaborar esse esquema:

1. Use o valor do clock do sistema como timestamp, ou seja, o timestamp de uma transação é igual ao valor do clock quando a transação entrar no sistema.

2. Use um contador lógico que é incrementado após um novo timestamp ter sido atribuído, ou seja, o timestamp de uma transação é igual ao valor do contador quando a transação entrar no sistema.

Protocolos baseados em timestamp

Para implementar esse esquema, associamos a cada item de dados Q dois valores de timestamp:

1. W-timestamp(Q) que indica o maior timestamp de qualquer transação que executou write(Q) com sucesso.

2. R-timestamp(Q) que indica o maior timestamp de qualquer transação que executou read(Q) com sucesso.

Protocolos baseados em timestamp

Protocolo de ordenação por timestamp

- Consideramos as transações T13 e T14. A transação T13 apresenta o conteúdo das contas C1 e C2:

T13:
read(C2);
read(C1);
display(C1 + C2)

T14:
read(C2);
C2 := C2 – 50;
write(C2);
read(C1);
C1 := C1 + 50;
write(C1);
display(C1 + C2)

Protocolos baseados em timestamp

- Na apresentação de schedules sob o protocolo de timestamp, vamos considerar que uma transação recebeu um timestamp imediatamente antes de sua primeira instrução. Assim, no schedule, TS(T15) < TS(T16), e o schedule é possível sob o protocolo de timestamp.

T15:	**T16:**
read(C2);	
	read(C2);
	C2 := C2 – 50;
	write(C2);
read(C1);	
	read(C1);
display(C1 + C2)	
	C1 := C1 + 50;
	write(C1);
	display(C1 + C2)

Protocolos baseados em validação

- Nos casos em que a maioria das transações são transações somente leitura, a taxa de conflito de transações pode ser baixa. Assim, muitas dessas transações, se executadas sem serem verificadas pelo sistema de controle concorrente, deixariam o sistema em um estado consistente. O sistema de controle de simultaneidade coloca uma carga adicional na execução do código e possível atraso na transação. Pode ser melhor usar um sistema alternativo que crie menos sobrecarga. Uma das dificuldades para reduzir os overheads é que não sabemos de antemão quais eventos estarão envolvidos no conflito. Para obter essas informações, precisamos de um diagrama para controlar o sistema.

Protocolos baseados em validação

O protocolo de validação exige que cada transação de Ti seja executada em dois ou três momentos diferentes durante sua vida útil, dependendo se é uma transação somente leitura ou uma transação de atualização. As etapas estão na seguinte ordem:

1. Fase de leitura. Durante essa fase, o sistema executa a transação Ti. Ele lê os valores dos diversos itens de dados e os guarda em variáveis locais da Ti. Ele realiza todas as transações write em variáveis locais temporárias, sem atualizações do banco de dados real.
2. Fase de validação. O teste de validação é aplicado para a transação Ti. Ele determina se Ti tem permissão para seguir para a fase de escrita sem causar uma violação de serializabilidade. Se a transação falhar no teste de validação, o sistema aborta a transação.
3. Fase de escrita. Se a transação Ti tiver sucesso na validação, as variáveis temporárias locais que mantêm os resultados de quaisquer operações write realizadas por Ti são armazenadas no banco de dados. As transações somente leitura omitem essa fase.

Protocolos baseados em validação

Cada transação deve passar por três etapas na ordem mostrada. No entanto, fases de transações simultâneas podem ser omitidas. Para realizar um teste de validação, precisamos saber quando ocorreram as diferentes etapas das transações. Portanto, associaremos três timestamps diferentes à transação Ti.

1. **StartTS(Ti)**, o momento em que Ti iniciou sua execução.

2. **ValidationTS(Ti)**, o momento em que Ti terminou sua fase de leitura e iniciou sua fase de validação.

3. **FinishTS(Ti)**, o momento em que Ti terminou sua fase de escrita.

Protocolos baseados em validação

- Considere novamente as transações T15 e T16. Suponha que TS(T15) < TS(T16). Então, a fase de validação é bem-sucedida no schedule. Podemos observar que as escritas para as variáveis em si são realizadas somente após a fase de validação de T16. Assim, T15 lê os valores antigos de C2 e C1, e, dessa forma, esse schedule é serializável.

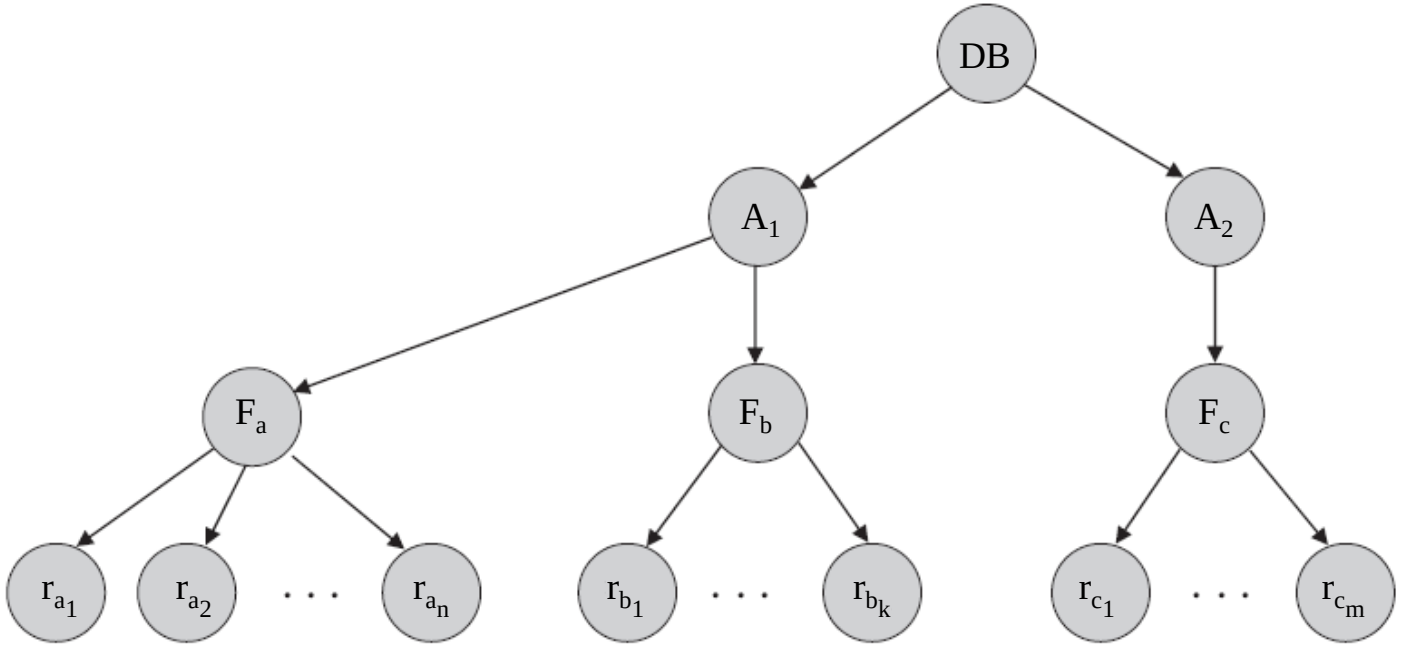
T15:	**T16:**
read(C2);	
	read(C2);
	C2 := C2 – 50;
	read(C1);
	C1 := C1 + 50;
read(C1);	
<validar>	
display(C1 + C2)	
	<validar>
	write(C2);
	write(C1);

Granularidade múltipla

- Há situações em que seria útil agrupar várias unidades de dados e processá-las como uma única unidade de planejamento. Por exemplo, se a transação Ti precisar de acesso a todo o relacionamento e o protocolo de bloqueio for usado para bloquear uma tupla, Ti deve bloquear todas as tuplas no relacionamento. Obviamente, obter muitos desses bloqueios leva tempo; pior ainda, a tabela de bloqueio pode ficar muito grande e não caber mais na memória. Seria melhor se Ti pudesse fazer uma única solicitação de chave para bloquear todo o relacionamento. Por outro lado, se a transação Tj precisar usar apenas algumas tuplas, ela não precisará bloquear toda a conexão porque a concorrência é perdida.

Granularidade múltipla

- Considere a árvore, que consiste em quatro níveis de nó. O nível superior representa todo o banco de dados. Abaixo estão os nós do tipo região; o banco de dados consiste nessas regiões. Cada região, por sua vez, tem dois nós de tipo de arquivo como filhos. Cada área contém exatamente os arquivos que são seus filhos. Nenhum arquivo reside em mais de uma região. Por fim, cada arquivo possui nós de armazenamento. Como antes, um arquivo consiste exatamente nos registros que são filhos dele, e nenhum registro pode estar em mais de um arquivo.



Fonte: Silberschatz (2020, p. 481).

Interatividade

Analise as afirmações:

I. Os protocolos baseados em timestamp determinam a ordem entre cada par de transações em conflito em tempo de execução.
II. O protocolo de validação exige que cada transação de Ti seja executada em dois ou três momentos diferentes durante sua vida útil.
III. Nos casos em que a maioria das transações são transações somente leitura, a taxa de conflito de transações pode ser baixa.
IV. Há situações em que seria útil agrupar várias unidades de dados como uma única unidade.
V. Na fase de escrita, se a transação Ti tiver sucesso na validação, as variáveis temporárias locais que mantêm os resultados de quaisquer operações write realizadas por Ti são armazenadas no BD.

Estão corretas:

a) I, II e III.
b) I, II e IV.
c) I, II e V.
d) III, IV e V.
e) Todas as afirmações.

Resposta

Analise as afirmações:

I. Os protocolos baseados em timestamp determinam a ordem entre cada par de transações em conflito em tempo de execução.
II. O protocolo de validação exige que cada transação de Ti seja executada em dois ou três momentos diferentes durante sua vida útil.
III. Nos casos em que a maioria das transações são transações somente leitura, a taxa de conflito de transações pode ser baixa.
IV. Há situações em que seria útil agrupar várias unidades de dados como uma única unidade.
V. Na fase de escrita, se a transação Ti tiver sucesso na validação, as variáveis temporárias locais que mantêm os resultados de quaisquer operações write realizadas por Ti são armazenadas no BD.

Estão corretas:

a) I, II e III.
b) I, II e IV.
c) I, II e V.
d) III, IV e V.
e) Todas as afirmações.

Referências

- SILBERSCHATZ, A; SUDARSHAN, H.F. *Sistema de banco de dados*. Rio de Janeiro: Elsevier, 2020.

ATÉ A PRÓXIMA!