Bases de Dados

TRANSAÇÕES E CONCORRÊNCIA

Transações - Contexto

- A utilização de bases de dados por mais que um utilizador (ou programa) pode pôr em risco a consistência da informação contida na BD.
  - **Exemplo**: O utilizador U1 lê os dados da conta de um cliente e credita 5€ mas entre o momento de leitura do valor e o momento de escrita do novo valor na BD, o utilizador U2 lê os dados da conta do mesmo cliente e debita 2€. Neste caso, quando U1 escreve o valor que leu adicionado de 5€ na BD, o débito feito por U2 perde-se.
- Interrupções imprevistas na execução de um programa podem também originar inconsistências na BD.
  - **Exemplo**: Numa transferência bancária, a quebra de energia elétrica pode fazer com que o montante a transferir tenha sido debitado na conta de origem mas não tenha sido creditado na conta destino.

Transações – Definição

- **transação** = uma unidade lógica de trabalho que envolve uma ou diversas operações sobre a base de dados.
- Sequência de instruções SQL.
- Definidas de forma:
  - **implícita**, a partir de uma instrução SQL (por defeito), ou
  - **explícita**, por parte do programador através de instruções próprias para o efeito (a ver mais à frente), envolvendo várias instruções SQL.

Transações – Propriedades

- As transações devem ser **ACID**:
  - **Atómicas**: cada uma das instruções da transação só toma efeito se todas tomarem efeito
  - **Consistentes**: a execução de uma transação isoladamente não coloca a BD inconsistente (esta propriedade é da responsabilidade do programador)
  - **Isoláveis**: a execução de uma transação em concorrência produz o mesmo efeito da execução isolada
  - **Duráveis**: quando uma transação é dada como bem sucedida, os efeitos devem perdurar na BD mesmo que haja falhas de sistema.

Transações – Escalonamento (1)

- Estabelece o plano de execução das transações:
  - Cada transação é descrita pela sequência de ações
  - Cada ação é de um destes quatro tipos:
    - **Read**: leitura da BD
    - **Write**: escrita na BD
    - **Commit**: ação que finaliza uma transação bem sucedida guardando de forma persistente as alterações feitas
    - **Abort**: ação que finaliza uma transação mal sucedida descartando as alterações feitas pela transação e repondo a BD num estado de consistência
  - A lista de ações é obtida pelo SGBD a partir das instruções SQL. Não é da responsabilidade do programador defini-la.

Transações – Escalonamento (2)

- Os escalonamentos que contêm um commit ou um abort por cada transação diferente que contenha, designam-se por **escalonamentos completos**
- Os escalonamentos que não têm ações alternadas de diferentes transações, i.e., cada transação só começa quando a transação anterior termina, designam-se por **escalonamentos em série**

Motivação para escalonamentos alternados

- Quando os escalonamentos são em série, algumas transações, especialmente as mais pequenas, podem ter tempos de resposta inesperadamente grandes por serem precedidas por transações grandes
- Permitindo, de forma controlada, a execução alternada de ações de diferentes transações, é possível aumentar o nível de satisfação global dos utilizadores das BDs
- Isso é potenciado pela possibilidade de executar atividades de CPU e de I/O (i.e., de leitura/escrita na BD), em paralelo

Escalonamento serializável

- Um escalonamento diz-se serializável quando o resultado da sua execução é idêntico ao resultado de uma das possíveis execuções em série das transações que o compõem.
  - A ordem pela qual as transações são executadas (em série) pode afetar o resultado final
  - A execução serializável deve garantir a equivalência para com uma das possíveis execuções em série, não sendo possível determinar qual delas

Conflito: Perda de atualizações

- **Atualizar um registo "por cima" de outro**
- Exemplo:
  - T1: levantar 10
  - T2: depositar 100

Tempo	T1	T2	saldo
t_1		begin_transaction	100
t_2	begin_transaction	read($\mathbf{bal_x}$)	100
t_3	read($\mathbf{bal_x}$)	$\mathbf{bal_x} = \mathbf{bal_x} + 100$	100
t_4	$\mathbf{bal_x} = \mathbf{bal_x} - 10$	write($\mathbf{bal_x}$)	200
t_5	write($\mathbf{bal_x}$)	commit	90
t_6	commit		90

Conflito: Atualização temporária

- **Uma transação lê resultados intermédios de outra transação**
- Exemplo:
  - T3: levantar 10
  - T4: depositar 100
    - aborta

Tempo	T3	T4	saldo
t_1		begin_transaction	100
t_2		read($\mathbf{bal_x}$)	100
t_3		$\mathbf{bal_x} = \mathbf{bal_x} + 100$	100
t_4	begin_transaction	write($\mathbf{bal_x}$)	200
t_5	read($\mathbf{bal_x}$)	⋮	200
t_6	$\mathbf{bal_x} = \mathbf{bal_x} - 10$	rollback	100
t_7	write($\mathbf{bal_x}$)		190
t_8	commit		190

Conflito: Análise inconsistente

- **Uma transação lê vários valores da BD mas, enquanto isso, outra transação atualiza alguns valores**
- Exemplo:
  - T5: transferir 10 de sX para sZ
  - T6: somar o saldo de todas as contas

Tempo	T5	T6	sX	sY	sZ	soma
t_1		begin_transaction	100	50	25	
t_2	begin_transaction	sum = 0	100	50	25	0
t_3	read($\mathbf{bal_x}$)	read($\mathbf{bal_x}$)	100	50	25	0
t_4	$\mathbf{bal_x} = \mathbf{bal_x} - 10$	sum = sum + $\mathbf{bal_x}$	100	50	25	100
t_5	write($\mathbf{bal_x}$)	read($\mathbf{bal_y}$)	90	50	25	100
t_6	read($\mathbf{bal_z}$)	sum = sum + $\mathbf{bal_y}$	90	50	25	150
t_7	$\mathbf{bal_z} = \mathbf{bal_z} + 10$		90	50	25	150
t_8	write($\mathbf{bal_z}$)		90	50	35	150
t_9	commit	read($\mathbf{bal_z}$)	90	50	35	150
t_{10}		sum = sum + $\mathbf{bal_z}$	90	50	35	185
t_{11}		commit	90	50	35	185

Escalonamento recuperável

- Um escalonamento diz-se recuperável se cada transação que o compõe é cometida só depois de todas as transações cujas alterações foram lidas por essa transação terem sido cometidas
- Num escalonamento recuperável é possível cancelar a transação sem ter de o fazer em cascata.

Possíveis problemas ao cancelar uma transação

- Quando uma transação é cancelada, deve-se colocar a BD num estado consistente, desfazendo as ações até então executadas
- Para o evitar são necessários mecanismos de recuperação complexos

Controlo de concorrência com bloqueios

Protocolo de bloqueios:

- conjunto de regras que cada transação deve seguir
- regras asseguradas pelo SGBD
- de forma a que o escalonamento definido seja serializável

Protocolo em duas fases estrito

- Fases:
  1. Se uma transação T quer ler ou modificar um objeto, tem de previamente requerer respetivamente um bloqueio partilhado ou exclusivo sobre o objeto;
     - Os bloqueios partilhados só permitem ler
     - Os bloqueios exclusivos permitem ler e escrever
     - Uma transação que requer um bloqueio fica em espera até que o SGBD possa executá-lo
  2. Todos os bloqueios requeridos por uma transação só são libertados quando a transação termina.

- O cumprimento do protocolo em duas fases estrito garante escalonamentos recuperáveis

Protocolo em duas fases

- Fases:
  1. Se uma transação T quer ler ou modificar um objeto, tem de previamente requerer respetivamente um bloqueio partilhado ou exclusivo sobre o objeto;
     - Os bloqueios partilhados só permitem ler
     - Os bloqueios exclusivos permitem ler e escrever
     - Uma transação que requer um bloqueio fica em espera até que o SGBD possa executá-lo
  2. Uma transação não pode requerer bloqueios adicionais se já tiver feito algum desbloqueio

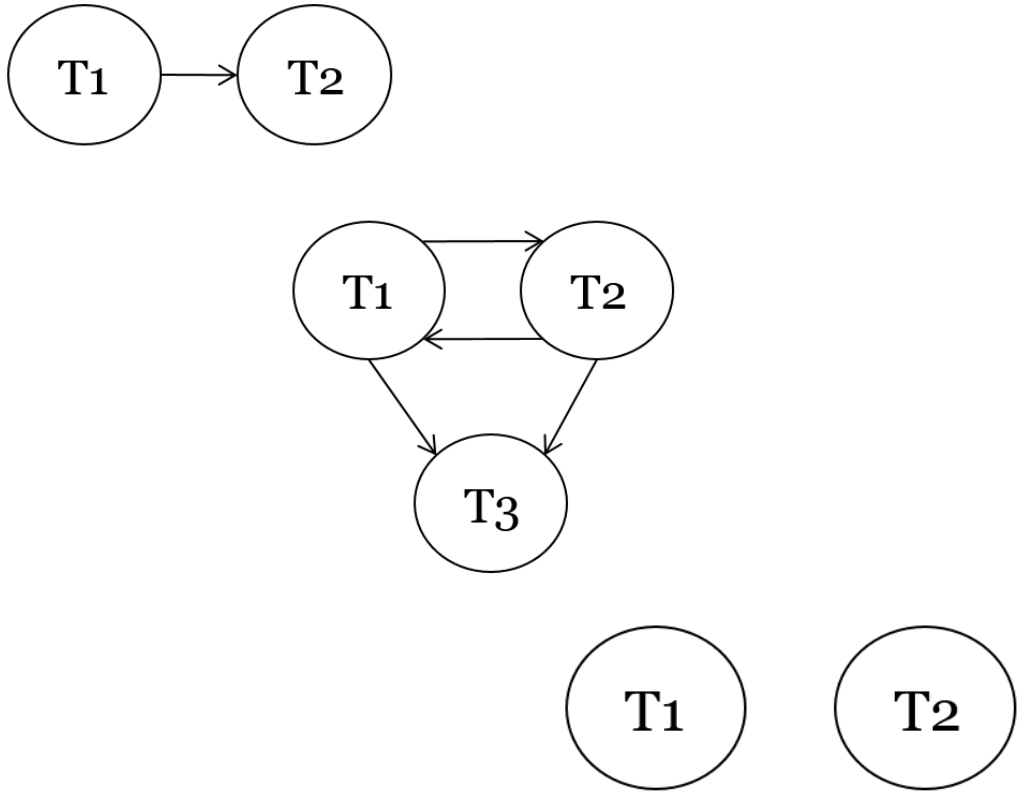
- A diferença para a versão estrita é que a segunda fase, apesar de só conter desbloqueios, é feita de forma progressiva, em vez de desbloquear tudo só no fim
- O cumprimento do protocolo em duas fases não garante escalonamentos recuperáveis

Equivalência em conflitos

- Dois escalonamentos têm equivalência de conflitos se compreendem as ações das mesmas transações e ordenam cada par de ações conflituosas de duas transações cometidas da mesma forma
  - O resultado de um escalonamento só depende da ordem das ações conflituosas
- Um escalonamento diz-se serializável por conflitos se tem equivalência de conflitos com algum escalonamento em série
- Qualquer escalonamento serializável por conflitos é serializável, sob o pressuposto que não se insere nem se apaga registos da BD (mas pode-se alterar valores)
  - O contrário pode não se verificar

Grafo de precedências

- O grafo de precedências para o escalonamento S contém
  - Um nó por cada transação cometida em S
  - Um arco de Ti para Tj se uma ação de Ti precede e entra em conflito com uma ação de Tj
- Um escalonamento S é serializável por conflitos se e só se o seu grafo de precedência for acíclico
- Qualquer escalonamento que assegure o bloqueio em duas fases ou em duas fases estrito tem um grafo de precedências acíclico



Gestor de bloqueios

- Componente do SGBD que guarda informação dos bloqueios
- O gestor de bloqueios mantem uma tabela de bloqueios
- Cada registo dessa tabela contém:
  - número de transações que têm um bloqueio sobre um dado objeto,
  - a natureza desse bloqueio (partilhado ou exclusivo)
  - apontador para a lista de pedidos de bloqueio
- O número de transações que têm um bloqueio sobre um dado objeto pode ser maior do que um se o bloqueio for partilhado

Pedidos de bloqueio ou desbloqueio

- Os bloqueios podem ser feitos sobre objetos de diferentes tipos (dependendo do SGBD): páginas, registos, …
  - As páginas são as unidades elementares para acesso a disco. A identificação da página onde se encontra cada registo é feita pelo gestor de ficheiros do SGBD
- Quando uma transação faz um pedido de bloqueio sobre um objeto ao gestor de bloqueios:
  - Se é um bloqueio partilhado, se a fila de pedidos sobre esse objeto está vazia e se o objeto não se encontra bloqueado por um bloqueio exclusivo, o gestor de bloqueios executa o bloqueio e atualiza a tabela de bloqueios
  - Se é um bloqueio exclusivo e não há nenhum bloqueio sobre esse objeto (o que implica não existir nenhum pedido na fila), o bloqueio é executado e a tabela de bloqueios é atualizada
  - Caso contrário, o pedido não é concedido, sendo adicionado à fila de pedidos de bloqueio desse objeto, e a transação fica suspensa
- Quando um bloqueio sobre um objeto é libertado, o próximo pedido na fila de bloqueios desse objeto é avaliado

Possíveis problemas dos protocolos de bloqueios

- Bloqueio ativo (livelock):
  - T1 e T2 pedem bloqueio de A;
  - T1 obtém;
  - T3 pede bloqueio de A;
  - quando T1 desbloqueia, T3 é servido e T2 fica à espera…
  - Hipótese de resolução: servir sempre o pedido mais antigo. Fácil.
- Encravamento (deadlock):
  - Seja:
    - T1: WLOCK(A); WLOCK(B); UNLOCK(A); UNLOCK(B);
    - T2: WLOCK(B); WLOCK(A); UNLOCK(B); UNLOCK(A).
  - Caso o escalonamento seja feito, por exemplo, pela ordem:
    - WLOCKT1(A); WLOCKT2 (B); WLOCKT1 (B); WLOCKT2 (A); …
  - Encrava ...

Transações SQL-92

- As transações são iniciadas de forma automática quando o utilizador executa uma instrução que modifica a base de dados ou o catálogo
- As transações terminam:
  - Explicitamente com uma das instruções:
    - COMMIT: faz o cometimento da transação
    - ROLLBACK: cancela a transação
  - Implicitamente: quando é executada uma instrução LDD-SQL

Transações SQL-92

- Características das transações:
  - Dimensão do diagnóstico: especifica o número de erros que podem ser registados na área de diagnóstico
  - Modo de acesso:
    - READ ONLY: só permite leituras
    - READ WRITE: permite leituras e escritas
  - Nível de isolamento: controla o quanto uma transação fica exposta a outras transações concorrentes, ou seja, permite dosear entre escalonamentos mais alternados e o risco de não garantir a seriabilidade desse escalonamento. Existem quatro níveis de isolamento:
    - Serializável (serializable)
    - Leitura repetível (repeatable read)
    - Leitura cometida (read commited)
    - Leitura não cometida (read uncommitted)

Transações SQL-92

- O nível de isolamento serializável assegura que a transação T:
  - Só lê alterações feitas por transações já cometidas;
  - Nenhum valor lido ou escrito por T é alterado por outra transação até T terminar;
  - O conjunto de valores correspondente a um dado critério de pesquisa e que tenha sido lido por T, não pode ser alterado por nenhuma transação até T completar
- Na implementação baseada em bloqueios, uma transação serializável faz bloqueios antes de ler ou escrever nos objetos, incluindo bloqueios em conjuntos de objetos de forma a evitar o problema fantasma

Transações SQL-92

- O nível de isolamento leitura repetível assegura que a transação T:
  - Só lê alterações feitas por transações já cometidas;
  - Nenhum valor lido ou escrito por T é alterado por outra transação até T terminar.

- A única diferença entre os níveis de isolamento serializável e leitura repetível é que este último não faz bloqueio aos índices, ou seja pode bloquear objetos mas não garante o bloqueio a todos os objetos associados (dinamicamente) a um critério de pesquisa

Transações SQL-92

- O nível de isolamento leitura cometida assegura que a transação T:
  - Só lê alterações feitas por transações já cometidas;
  - Nenhum valor escrito por T é alterado por outra transação até T terminar sendo, no entanto, possível que um valor lido por T seja modificado por outra transação antes de T terminar.

- A única diferença entre os níveis de isolamento leitura repetível e leitura cometida é que, neste último, os bloqueios de leitura são libertados logo após a leitura. Os bloqueios de escrita, em ambas as situações, só são libertados no fim.

Transações SQL-92

- O nível de isolamento leitura não cometida assegura que a transação T:
  - Pode ler alterações feitas a um objeto por uma transação em curso

- Na leitura não cometida não são efetuados bloqueios partilhados antes da leitura dos objetos. Devido ao elevado nível de exposição a alterações não cometidas, a SQL só permite associar este nível de isolamento a transações com modo de acesso READ ONLY. Sendo só de leitura e não fazendo bloqueios de leitura, significa que não faz qualquer tipo de bloqueio

Transações SQL-92

- Resumo

Nível	Leitura suja	Leitura irrepetível	Fantasma
Serializável	Não	Não	Não
Leitura repetível	Não	Não	Eventualmente
Leitura cometida	Não	Eventualmente	Eventualmente
Leitura não cometida	Eventualmente	Eventualmente	Eventualmente

Transações SQL-92

- O nível de isolamento serializável é o mais seguro, sendo o mais adequado para a maioria das situações. No entanto, se outro nível de isolamento for adequado para uma dada transação, a sua utilização poderá melhorar o desempenho do sistema
  - **Exemplo**: utilizando o nível de isolamento leitura cometida ou leitura não cometida, é possível calcular a média de idades dos alunos de forma mais eficiente e sem distorcer, de forma significativa, o valor verdadeiro, pois a perda ou ganho de um ou dois valores não alterará significativamente a média

Transações SQL-92

- Instrução para definir o modo de acesso e o nível de isolamento de uma transação: SET TRANSACTION
  - Usando o exemplo anterior:
    - SET TRANSACTION ISOLATION LEVEL READ UNCOMMITED READ ONLY
- Quando uma transação se inicia, os valores por defeito para o nível de isolamento e o modo de acesso são, respetivamente, SERIALIZABLE e READ WRITE

```
SET [GLOBAL | SESSION] TRANSACTION
    transaction_characteristic [,
    transaction_characteristic] ...

transaction_characteristic:
    ISOLATION LEVEL level
  | READ WRITE
  | READ ONLY

level:
     REPEATABLE READ
   | READ COMMITTED
   | READ UNCOMMITTED
   | SERIALIZABLE
```

Controlo de concorrência sem bloqueios

- A utilização de protocolos de bloqueio para controlo de concorrência é uma abordagem pessimista:
  - cancelam transações ou bloqueiam objetos de forma preventiva para resolver conflitos.
- Em sistemas com níveis baixos de concorrência, o custo deste tipo de abordagens pode ser significativo.
- Existem protocolos alternativos que optam por, mais do que prevenir os conflitos, tentar resolvê-los.
  - Parte-se do princípio que os conflitos são eventos raros e, por isso, o custo de resolução, sendo elevado por cada conflito, tem um custo global baixo.

Controlo de concorrência baseada em marcas temporais

- Princípios do controlo de concorrência por marcas temporais:

1. É atribuída a cada transação Ti uma marca temporal TS(Ti) quando Ti é iniciada
2. Assegurar durante a execução que, no caso de haver conflito entre a ação Ai de Ti e a ação Aj de Tj, Ai ocorre antes de Aj se TS(Ti) < TS(Tj)
3. Se uma acção viola esta ordenação, a transação é cancelada e reiniciada

- Para implementar esta forma de controlo de concorrência, define-se, para cada objeto (O), a marca temporal da transação que fez a última leitura e a última escrita, respetivamente TL(O) e TE(O).

Recuperação

- O gestor de recuperações de um SGBD é responsável por assegurar:
  - Atomicidade: desfazendo as ações das transações ainda não cometidas quando uma transação é cancelada
  - Durabilidade: garantindo que todas as ações das transações cometidas perduram a falhas do sistema
- Guarda num log todas as alterações feitas à BD

Recuperação

Causas para a necessidade de recuperação

- Falha lógica de sistema (falha de energia, core dump, …)
- Falha física do sistema de armazenamento (disco, …)
- Interação de outros algoritmos com o controlo de concorrência

Recuperação - log

- Contém o historial das ações executadas pelo SGBD.
- É importante minimizar a possibilidade de perder informação.
  - Exemplo: escrevendo duas cópias em discos diferentes / locais diferentes.
- A componente mais recente do log é mantida na memória principal e guardada periodicamente em local seguro.
- A cada entrada do log é atribuída um número sequencial crescente (i.e., números maiores correspondem a entradas mais recentes), designado *Last Sequence Number (LSN)*.
- Para facilitar a recuperação, cada página da BD tem o número sequencial da última ação de alteração feita sobre essa página, designado por *pageLSN*.

Recuperação - log

Acrescenta-se uma entrada ao jornal nas seguintes situações:

- Página alterada: acrescenta-se um registo do **tipo alteração**. O page*LSN* é atualizado com o valor do *LSN* do novo registo.
- Transação cometida: acrescenta-se um registo do **tipo cometimento**. De seguida, guarda-se em disco a componente do jornal mantida em memória, incluindo a entrada referente ao cometimento. O cometimento fica, assim, terminado, mas há ainda outros passos a realizar (ex.: remover a transação da tabela de transações).
- Transação cancelada: acrescenta-se um registo do **tipo cancelamento**. De seguida, a ação desfazer é iniciada.
- Transação finalizada: acrescenta-se um registo do **tipo fim**. Este registo só é criado após terem sido realizadas todas as ações efectuadas após o cancelamento ou o cometimento de uma transação (já referidas anteriormente).
- Alteração desfeita: acrescenta-se um registo do **tipo compensação** quando a ação descrita por um registo do tipo alteração é desfeita. Tal pode ocorrer após cancelamento de uma transação ou após recuperação devido a falha.

Recuperação - log

Atributos comuns a todos os registos (seja qual for o tipo):

- *prevLSN*: *LSN* do registo anterior referente à mesma transação;
- *transID*: *ID* da transação que gerou o registo;
- tipo: tipo do registo (alteração, cometimento, cancelamento, fim ou compensação).

Alguns dos atributos adicionais dos registos do tipo 'alteração':

- *pageID*: *ID* da página alterada.
- *before_image*: valor antes da alteração. Permite desfazer uma alteração.
- *after_image*: valor após a alteração. Permite refazer uma alteração.

Atributo adicional dos registos de 'compensação':

- *before_image*:
- *undoNextLSN*: contém o *LSN* do próximo registo do jornal a desfazer. Este atributo terá o valor de *prevLSN* do registo da alteração a ser desfeita.

Recuperação – log - Ponto de verificação

Até onde percorrer o jornal numa recuperação?

- Operação de verificação (checkpoint)
  - proibir o início de transações e esperar que as ativas estejam cometidas ou abortem
  - copiar os blocos alterados na memória central para memória estável
  - registar no log o ponto de verificação e escrever o log
- Só é necessário analisar o log até ao último ponto de verificação, para recuperar de uma falha
- a parte anterior do log só interessa para arquivo de segurança

Recuperação – Algoritmo de recuperação ARIES

ARIES (**A**ggregate **R**ecoverable **I**tem **E**valuation **S**ystem)

Quando o gestor de recuperações é invocado após falha do sistema, a reinicialização é feita em três fases:

- **Analisar**: identifica páginas sujas do *buffer* (i.e., que não tenham sido escritas em disco) e transações ativas no momento da falha do sistema
- **Refazer**: repete todas as ações desde um determinado ponto do jornal, designado por ponto de verificação (*checkpoint*), repondo a base de dados no estado em que se encontrava antes da falha do sistema.
- **Desfazer**: desfaz as ações das transações que não foram cometidas

Recuperação – Algoritmo de recuperação ARIES

Princípios básicos:

- Escrita prévia no log: uma alteração, antes de ser realizada, é registada no log; o registo feito no log deve ser guardado em local seguro (disco) antes de a alteração à base de dados ser escrita em disco.
- Repetir todas as ações durante o refazer: depois de uma falha do sistema, todas as ações são feitas de novo mesmo as ações daquelas transações que, não tendo sido cometidas no momento da falha, tenham de ser canceladas.
- Registar no log as ações da fase 'desfazer': as alterações feitas à base de dados quando se desfazem as ações das transações que não foram cometidas, devem ser registadas no jornal para garantir a recuperabilidade no caso de haver nova falha de sistema.