Sumário

- O que são transações.
- Propriedades.
- Controlo da concorrência.
- Recuperação.

O que são transações

- Sequências de operações que são executadas de modo indivisível ou atómico.

  Mesmo na presença da execução concorrente de outras transações.

  Mesmo na presença de falhas de alguns componentes (neste caso, a transacção pode não se executar).

Transações e concorrência

- Considere a seguinte sequência de acções:

```
void deposita (numero_conta, valor) {
  x = le_conta (numero_conta);
  x = x + valor;
  actualiza_conta (numero_conta, x)
}
```

- A execução concorrente desta sequência pode gerar valores incongruentes. *Ex: Seja '0' valor inicial na conta, a execução concorrente de* `deposita (10)` *com* `deposita (20)` *pode resultar num saldo de '10', '20' ou '30'.*

Transações e tolerância a faltas

- Considere a seguinte sequência de acções::

```
void transfere (conta_origem, conta_destino, valor) {
  x = le_conta (conta_origem);
  x = x - valor;
  actualiza_conta (conta_origem, x)
  y = le_conta (conta_destino);
  y = y + valor;
  actualiza_conta (conta_destino, y)
}
```

- A falha da transacção pode deixar o estado incoerente.

Propriedades das transacções

- Designadas por propriedades ACID:

  *Do Inglês, "**A**tomicity, **C**onsistency, **I**solation, and **D**urability".*

- Atomicidade.
- Coerência.
- Isolamento.
- Durabilidade.

Atomicidade

- Atomicidade refere-se ao facto de uma transacção ser tratada como uma unidade operacional.
- Todas as operações são executadas com sucesso ou nenhuma operação deve ser executada.

  Se uma transacção é interrompida por uma falta, todas as operações executadas até então devem ser anuladas.

- Uma transacção cujas operações são executadas com sucesso diz-se confirmada (*committed*).
- Uma transacção cujas operações são anuladas, diz-se cancelada (*aborted*).

Coerência e isolamento

- Coerência refere-se à correcção dos resultados da transacção, mesmo quando executada isoladamente.
- Isolamento referem-se à correcção dos resultados da transacção quando esta é executada em paralelo com outras transacções.

  Define que uma transacção não observa os resultados de outras transacções concorrentes, a não ser que estas venham a ser confirmadas e serializadas no seu passado.

Durabilidade

- A durabilidade garante que os resultados de uma transacção confirmada são permanentes e não podem ser apagados.

  Excepto na eventualidade de acontecimentos catastróficos.

- Os sistemas podem ser concebidos de modo a reduzir ao mínimo o risco de catástrofes.

  O recurso à replicação é uma das técnicas disponíveis para atingir este objectivo.

Controlo da concorrência

- É necessário definir um modelo de coerência (tal como para a memória distribuída).
- O modelo de coerência mais usado nos sistemas transaccionais é a serializabilidade (*serializability*).

  A execução concorrente de várias transacções é serializável *se produz os mesmos resultados que uma execução em série dessas transacções.*

- Tal como a coerência sequencial, no caso de operações não agrupadas, a serializabilidade é um modelo de coerência intuitivo e simples de compreender.

Controlo da concorrência (cont.)

- Mecanismos : grafos de serialização, trincos (*locks*), ou estampilhas temporais (*timestamps*).
- Políticas.

  Uma política pessimista *assume que a maioria das transacções vão concorrer sobre os mesmos dados e tenta sincronizá-las o mais cedo possível durante a sua execução.*

  Uma política optimista *assume que os conflitos são raros e só estabelece a sincronização no final da execução das transacções.*

Trincos

- Mecanismo comum para controlo da concorrência.
- Geralmente dois tipos de operações:

  Reserva para escrita.

  Reserva para leitura.

- Duas operações são incompatíveis se pelo menos uma delas é uma operação de escrita.

  Nesse caso é necessário seriar as operações.

Conjuntos de escrita e leitura

- Geralmente, uma transacção acede a várias variáveis.

  A cada variável estará associado um trinco.

- Designa-se o conjunto de escrita o conjunto de variáveis acedidas em modo escrita pela transacção.

  Define-se conjunto de leitura do mesmo modo.

- Um transacção exclusivamente de leitura (*read only*) não escreve em nenhuma variável.

Trinco em duas fases

- Trinco em duas fases (*two-phase locking*):

  Numa primeira fase, a transacção reserva os dados a que acede.

  Numa segunda fase, uma transacção liberta os dados a que já não necessita de aceder.

- Geralmente os dados acedidos ficam reservados até que a transacção confirme o aborte (*strict two-phase locking*).

Desvantagem de libertar mais cedo os dados

- Se uma transacção liberta dados que alterou antes de terminar, é possível que outras transacções leiam esses dados.
- Se mais tarde a transacção necessita de ser cancelada, é necessário cancelar todas as transacções que "viram" os resultados que vão ser anulados.

  Cancelamentos em cascata.

Outra desvantagem dos trincos

- Interblocagem:

  Duas transacções podem aceder aos mesmos dados por ordem diferente e ficarem ambas bloqueadas.

- Evitar, prevenir o detectar.

  Evitar: usar tempos máximo de espera e abortar.

  Prevenir: estabelecer regras no acesso aos trincos.

  Detectar: descobrir quais as transacções em interblocagem.

Outros mecanismos: estampilhas

- É atribuida uma estampilha a cada transacção, quando esta é lançada.
- Cada variável mantém a estampilha da última transacção que lhe acedeu (para escrita e para leitura).
- As transacções são seriadas por ordem das estampilhas.

Estampilhas: regras

- Leitura: se a estampilha da transacção é anterior à à ultima escrita na variável, a transacção é cancelada.
- Escrita: se a estampilha da transacção é anterior à última operação (escrita ou leitura) na variável, a transacção é cancelada.
- Pode abortar muitas transacções (e adiamento indefinido) mas não causa interblocagem.

Limitações da serializabilidade.

- Transacções de longa duração:

  Quando as transacções possuem elevados tempos de execução, a garantia da propriedade de isolamento impede o progresso eficiente do sistema.

- Cooperação:

  Por vezes é útil tornar resultados intermédios visíveis a outras transacções. Por exemplo, duas transacções podem necessitar de trocar informação. Neste caso, é a própria propriedade de isolamento que não se adequa aos requisitos da aplicação.

Recuperação

- Por vezes a transacção necessita de ser cancelada, e o estado do sistema reposto.
- Causas para cancelar uma transacção:

  Controlo da concorrência (ex: interblocagem).

  O utilizador decide cancelar a transacção.

  O sistema ou alguns componentes falham.

- Técnicas básicas:

  Actualização imediata e actualização diferida.

Actualização imediata

- Quando se escreve um valor: *memoriza-se o valor anterior num histórico de reposição (*undo log*) e actualiza-se a variável.*
- Quando se lê um valor: *lê-se simplesmente o valor actual.*
- Quando se confirma a transacção: *apaga-se o histórico de reposição.*
- Quando se aborta a transacção: *usa-se o histórico de reposição para repor o valor de todas as variáveis alteradas.*

Actualização diferida

- Quando se escreve um valor: *memoriza-se o valor novo num histórico de actualização (*redo log*).*
- Quando se lê um valor: *lê-se o valor actual (que pode estar no histórico de actualização.*
- Quando se confirma a transacção: *actualizam-se os dados de acordo com o histórico de actualização.*
- Quando se aborta a transacção: *apaga-se o histórico de actualização.*

Comparação

- A actualização imediata optimiza as leituras e a confirmação das transacções.
- A actualização diferida optimiza o cancelamento das transacções.
- Como geralmente se deseja que a maioria das transções seja confirmada, e como geralmente os acessos para leitura são maioritários, prefere-se a actualização imediata.

Recuperação de falhas

- Geralmente, o sistema transaccional têm pelo menos dois níveis de memória: *memória volátil (RAM) e memória persistente (disco).*
- Para garantir a atomicidade das transacções em caso de falha do sistema, são necessários alguns requisitos adicionais.

Requisitos para recuperação

- Não se pode actualizar a memória persitente sem antes guardar o histórico de reposição (*write-ahead log rule*).
- É necessário saber quais as transacções que já estavam confirmadas, e quais as que foram interrompidas pela falha.
- Não se pode confirmar uma transacção sem que se memorize este facto em memória persistente (*redo rule*).
- Deve ser possível executar o histórico de reposição múltiplas vezes.

Transacções distribuídas

- Se uma transacção executar acções em diversos nós, estes devem chegar a acordo acerca da confirmação da transacção.
- Necessário executar um protocolo de confirmação átomica (*atomic commitment*).

Resumo

- Motivação.
- Atomicidade, coerência, isolamento e durabilidade.
- Utilização de trincos (strict two-phase locking)
- Recuperação.