Bases de Dados

Gestão de Transações - Introdução

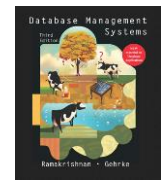
FCUL, Departamento de Informática

Ano Letivo 2015/2016

Ana Paula Afonso

Sumário e Referências

- Sumário
  - Transações
    - Motivação e Conceito
  - Propriedades ACID
  - Execução Concorrente de Transações
    - Tipos de anomalias
    - Controlo de concorrência baseado em *Locks*
  - Gestor de Recuperação
    - O *Log*
    - Recuperação de uma falha
- Referências
  - R. Ramakrishnan (**capítulo 16**)



Transações - Motivação

Produto

pid	pnome	preço	Stock
1	P1	20	25
2	P2	12	10
3	P3	15	10
4	P4	20	10

Cenário

- 9:00:00, uma aplicação cliente (C1) efetua uma pesquisa sobre a quantidade de produtos "P1" em stock.

```
SELECT stock FROM Produto
WHERE pnome='P1'
```

- 9:00:01, outra aplicação (C2) efetua uma pesquisa semelhante.
- O SGBD responde a ambos "25"
- C1 reserva 20 unidades, pelo que efetua uma dedução da respetiva quantidade à BD
- C2 efetua uma operação semelhante

 3

Transação - Definição

- Uma **transação** num SGBD é uma abstração de um procedimento
  - Uma sequência de operações de leitura e/ou escrita SQL …
  - ….. executada de forma atómica pelo SGBD

Início da transação
```
Instrução 1
Instrução 2
...
Instrução N
```
Fim da transação -- OK (COMMIT) ou Erro (Rollback)

- O mecanismo de transações é essencial
  - sempre que a BD servir várias clientes simultaneamente
  - recuperação de falhas

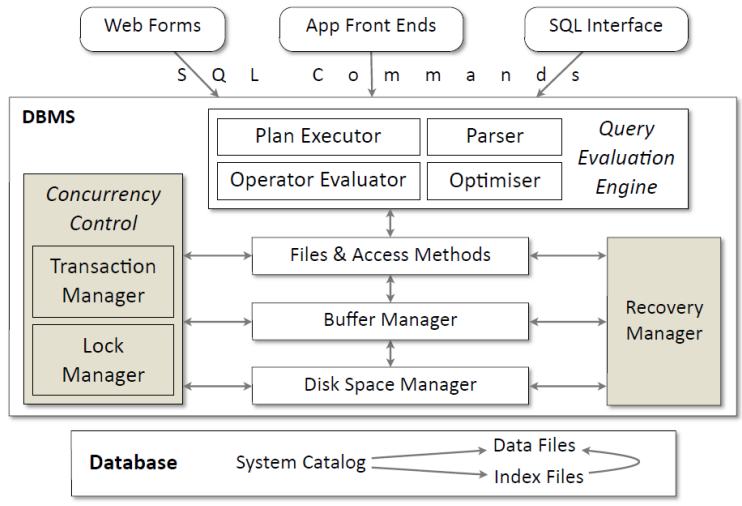
 4

Concorrência e Consistência

- Requisitos de um sistema transacional
  - Gestão de múltiplas **transações em simultâneo**
  - Gestão das regras de **integridade**
- Concorrência
  - Múltiplos utilizadores e respetivos pedidos em simultâneo
- Consistência (Integridade)
  - A BD está num estado consistente quando cumpre …
  - … as regras de integridade
- **Um SGBD é um sistema transacional**

 5

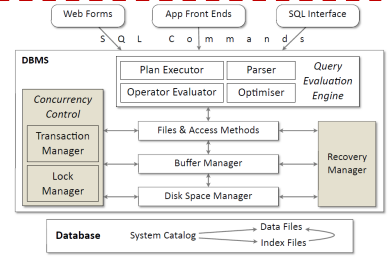
Componentes de um SGBD



Fonte: António Ferreira, Guião SIBD, 2015

 6

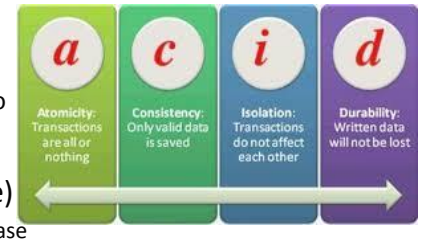
Componentes de um Sistema Transacional



- **Gestor de transações**
  - Cria, gere e termina transações
  - Efetua pedidos de *locks* para a acesso a dados
- **Gestor de *locks***
  - Controla o acesso concorrente a dados
  - Fornece *locks* para leitura ou escrita, assim que disponíveis
  - *Locks* podem ser exclusivos a uma transação ou partilhados
- **Gestor de recuperação**
  - Regista as ações realizadas por uma transação num ficheiro de *log*
  - Em caso de *crash*, faltas de software e hardware …
  - …. repõe a BD num estado coerente

 7

Propriedades Fundamentais Transações



- **Atomicidade**
  - Todas as operações executam ou todas são anuladas
- **Consistência** (coerência ou integridade)
  - A transação deve preservar a consistência da base de dados
- **Isolamento** (serialização)
  - Proteção contra riscos de execuções concorrentes
- **Durabilidade** (persistência)
  - Depois de concluída com sucesso os efeitos de uma transação são persistentes
  - … mesmo em caso de falta (*crash*) do sistema

 8

Propriedades **A**CID

- **Atomicidade**
  - Todas as operações de uma transação executam na totalidade ou então são anuladas
  - Este comportamento "tudo-ou-nada" serve para garantir a consistência em operações que só fazem sentido como um todo
  - Exemplo (clássico da necessidade desta propriedade)

    Processo de transferência de fundos entre duas contas bancárias

- Responsabilidade do sistema transacional (**Gestor de recuperação**)
  - Regista alterações efetuadas por transações não terminadas (não fizeram *commit*) num **ficheiro de *log***
  - Guarda estados antes e após cada alteração
  - Em caso de *crash*, **DESFAZ** (*UNDO*) operações/ações das transações parcialmente executadas

Propriedades ACI**D**

- **Durabilidade**
  - Depois de concluída com sucesso (COMMIT) os efeitos de uma transação são persistentes
  - Estado final persiste mesmo em caso de falha do sistema

- Responsabilidade do sistema transacional (**Gestor de recuperação**)
  - Regista todas as alterações efetuadas por transações terminadas (*commited*) …
  - … num **ficheiro *log***
  - Ficheiro *log* atualizado imediatamente antes do COMMIT
  - Garante que todas as operações/ações que fizeram *commit* sobrevivem à falta
  - Em caso de *crash,* **REFAZ** (*REDO*) as alterações que ainda não estão registadas na BD

Recuperação de uma falha

- Após *crash* do sistema, BD tem de ficar num estado coerente
  - Transações *uncommitted* têm de ser anuladas (**atomicidade**)
  - Transações *committed* têm de persistir (**durabilidade**)
- Dados guardados em cada registo no *log*
  - Identificador da transação
  - Tipo registo: `UPDATE, COMMIT, ABORT`
    Com o valor antigo e novo no `UPDATE`
- ARIES *(Algorithms for Recovery and Isolation Exploiting Semantics)*
  - **Análise** Identificar as transações que não tinham sido completadas
  - **Redo** Escrever no disco o que ficou por escrever
  - **Undo** Fazer *rollback* das transações não completadas

 11

Propriedades A**C**ID

- **Consistência**
  - Uma transação deve iniciar a sua execução tendo o sistema um estado conhecido …
  - …. e ao terminar deixá-lo num estado igualmente consistente
  - A transação preserva a consistência da BD
  - Ou seja o estado inicial e final têm de cumprir as regras de integridade
- Responsabilidade do utilizador
  - A coerência é a que tiver sido definida pelo utilizador
  - Supõe-se que a transação é um programa correto
    Ex. se a transação de transferência de dinheiro for mal especificada
    o SBGD não corrige estes erros
  - Supõe-se que todas as regras de integridade estão corretamente especificadas
- Responsabilidade do sistema transacional
  - Aplicar as RI definidas pelo utilizador

 12

Propriedades ACID

- **Isolamento**
  - As transações não devem depender de outras ou influenciar a execução de outras transações
  - Cada transação deve ter a percepção de que está a executar isoladamente no sistema
  - Garante que a **execução concorrente** de transações é equivalente a **execução em série**
  - Exemplo
    - T1 executada em paralelo com T2
    - O estado final tem de ser o mesmo T1 após T2 ou T2 após T1
    - O Sistema Transacional não garante uma ordem específica
- Riscos de execuções concorrentes (anomalias)
  - Leitura de dados obsoletos
  - Escritas em simultâneo dos mesmos dados
- Responsabilidade do sistema transacional
  - Concretiza um mecanismo de **controle da concorrência**

Execução Concorrente de Transações - Anomalias

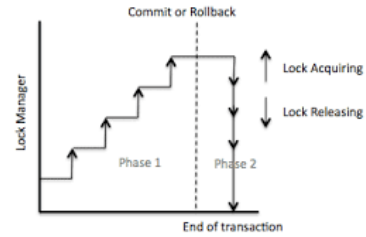
- Unrepeatable read (read-write) T1:READ(O), T2:WRITE(O), T1:READ(O)
  - Transação T1 lê duas vezes o mesmo objeto, em momentos diferentes
  - T1 não altera esse objeto, mas mesmo assim pode ler valores diferentes
- Dirty read (write-read) T1:READ(O), T1:WRITE(O), T2:READ(O), T1:WRITE(O)
  - T2 faz a leitura de um objeto que havia sido modificado por T1
  - T1 pode modificar novamente o mesmo objeto (ou fazer ROLLBACK)
  - T2 leu um valor que deixou de ser válido
- Lost update (write-write): T1:READ(O), T2:READ(O), T1:WRITE(O), T2:WRITE(O)
  - T1 e T2 fazem a leitura do mesmo objeto e escrevem o respetivo valor
  - Apenas o último valor escrito no objeto fica registado

Execução Concorrente de Transações - Resolução

- *Lock-Based Concurrency Control*
- Gestor de transações requer automaticamente *locks* sobre objetos
  - *Locks* são proteções de objetos de operações de leitura e/ou escrita
  - Os SGBD permitem a criação de diferentes tipos de bloqueios (*locks*) em páginas ou **tabelas de uma base de dados**
- Tipos de ***Locks***
  - SHARED LOCK (*s-lock*)
    Normalmente associados a operações de leitura de informação
    Várias transações podem obter *s-locks* sobre o mesmo objeto
  - EXCLUSIVE LOCK (*x-lock*)
    Normalmente associado a operações de escrita de informação
    Apenas uma transação pode obter um *x-lock* sobre um objeto
    Não permite que outra transação adquira um *s-lock* (visualizar informação)

Execução Concorrente de Transações - Resolução

- **Protocolo *Strict Two-phase Locking*** (Strict 2PL)
  Cada transação deve obter:
  1) ***shared lock*** por cada objeto antes de **ler**
  e ***exclusive lock*** por cada objeto antes de **escrever**
  2) Todos os ***locks*** são **libertos** no final da transação



- Cláusula FOR UPDATE no comando SELECT
  - Permite obter logo um *X-lock* aquando da leitura de um objeto
  - Ex. X-LOCK(O), READ(O), WRITE(O), COMMIT
  - Execução concorrente já não origina anomalia *lost update*
- *Deadlocks* podem causar **esperas mútuas** para aceder a objetos
  - Ex. T1:X-LOCK(O1), T2:X-LOCK(O2), T1:X-LOCK(O2), T2:X-LOCK(O1)
  - Deteção automática pelo gestor de *locks* por mecanismo de *timeout*

Propriedades ACID e Componentes

- Gestor de recuperação
  - Em caso de falta do sistema, garante **atomicidade** e **durabilidade**
  - Ficheiro de *log* usado com dois propósitos
    Desfazer operações de transações *uncommitted*
    Refazer operações de transações *committed*
- Gestores de transações e de *locks*
  - Durante a execução das transações, garantem **coerência** e **isolamento**
  - *Locks* usados para equivalência da execução de transações em série
  - Têm de resolver eventuais conflitos no acesso concorrente a recursos

 17