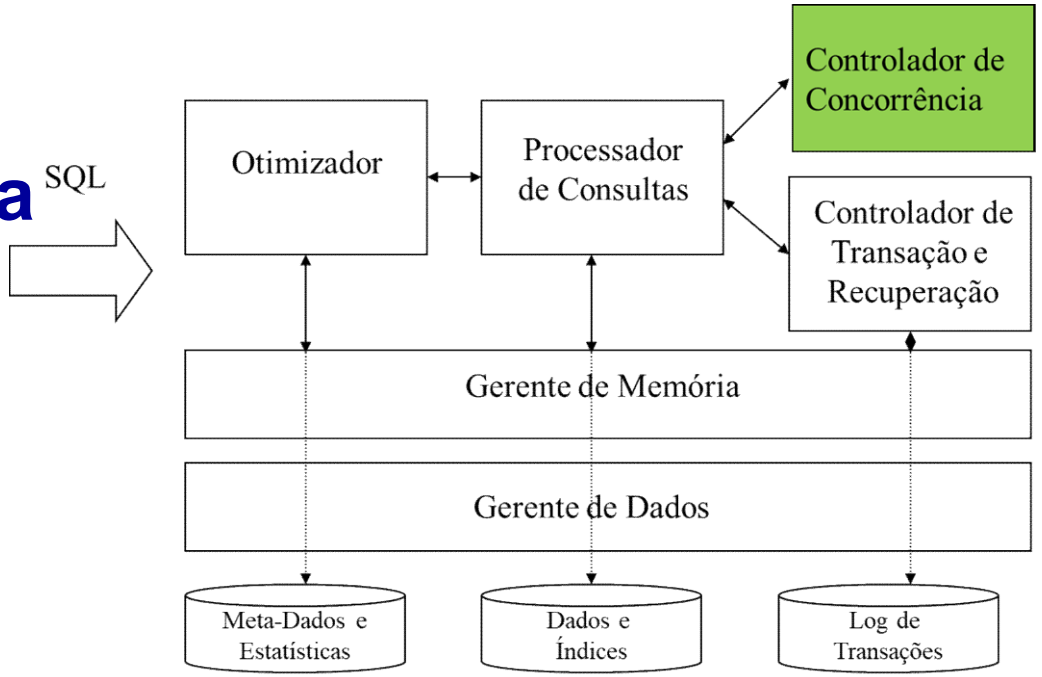
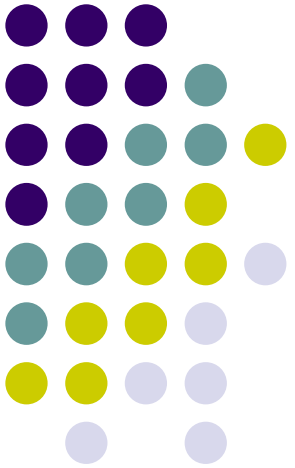
Bancos de Dados II

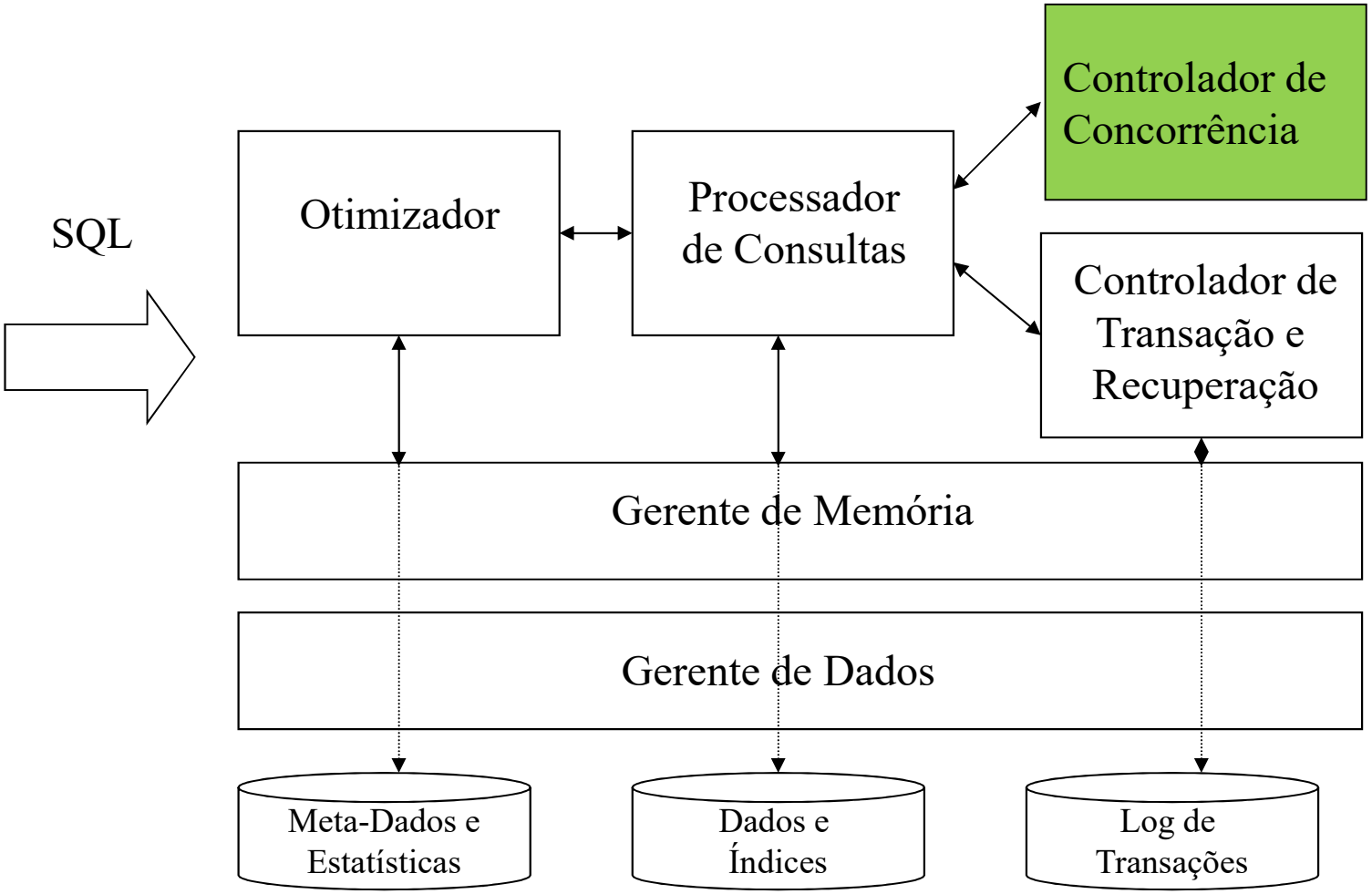
Controle de Concorrência



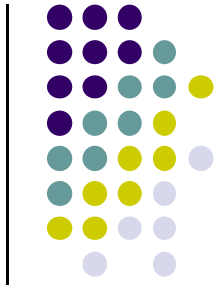


Arquitetura Funcional

Componentes de um SGBD



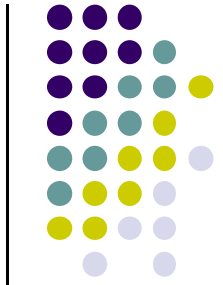
Bloqueios



Um bloqueio (lock) é uma variável associada a um item de dados que descreve as condições do item em relação às possíveis operações que podem ser aplicadas a ele.

Geralmente há um bloqueio para cada item de dados no banco e eles são usados para sincronizar o acesso por transações concorrentes a estes itens.

Bloqueio binários

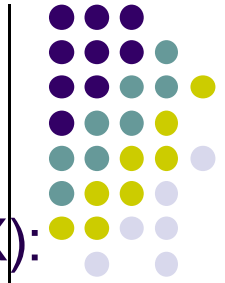


Um bloqueio binário pode ter dois estados ou valores: bloqueio e desbloqueio (1 e 0).

Um bloqueio é associado a cada item X do banco de dados. Se o valor do bloqueio for 1, o item não pode ser acessado e se for 0 o item pode ser acessado. A referência à variável de bloqueio é indicada como: LOCK(X).

Duas operações lock_item e unlock_item são usadas com bloqueio binário e são indivisíveis (como regiões críticas em sistemas operacionais).

Bloqueio binários



Implementação das operações lock_item(X) e unlock_item(X):

```
lock_item(X):
B:      if lock(X)==0    /*item desbloqueado*/
                then lock(X)=1/*bloqueia o item*/
        else begin
                wait(até que lock(X)=0)
                transação vai para fila de bloqueadas (em espera)
                goto B
                end;

unlock_item(X):
        lock(X)=0 /*item desbloqueado*/
        se existir transação na fila
                reinicia a primeira transação da fila (de espera)
```

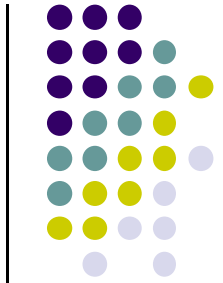
Bloqueio binários

Se for utilizado o esquema de bloqueio binário, toda transação deve obedecer às seguintes regras:

1. Uma transação T deve garantir a operação lock_item(X) antes que qualquer operação ler_item(X) ou escrever_item(X) seja executada em T;
2. Uma transação T deve garantir a operação unlock_item(X) depois que todas as operações ler_item(X) ou escrever_item(X) sejam completadas em T;
3. Uma transação T não resultará em uma operação lock_item(X) se ela já tiver o bloqueio no item X;
4. Uma transação não resultará em uma operação unlock_item(X), a menos que ela já tenha bloqueado o item X.

Esse esquema é muito restritivopara itens no banco de dados porque, no máximo, uma transação pode bloquear um item.

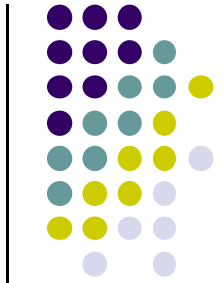
Bloqueios leitura/escrita ou compartilhados/exclusivos



Deveria ser permitido que diversas transações acessem o mesmo item, se todas o acessarem apenas com propósito de leitura. Se uma transação for alterar um item X ela deve ter acesso exclusivo a X.

Neste caso existem 3 estados possíveis para bloqueio a um item X:
read-lock ou share_lock, que permite compartilhamento do item entre transações, para leitura;
write_lock ou exclusive_lock, que realiza um bloqueio exclusivo da transação ao item, e
unlock, que indica que o item está desbloqueado.

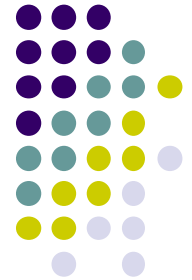
Bloqueios leitura/escrita ou compartilhados/exclusivos



Matriz de compatibilidade:

	shared	exclusive
shared	S	N
exclusive	N	N

Bloqueios leitura/escrita ou compartilhados/exclusivos



Gerenciador de bloqueios

- Gerencia o bloqueio de itens
- Mantém uma tabela de controle de bloqueio

  Exemplo: Controle(Transação, Item, Modo, Próximo_item)

Bloqueios leitura/escrita ou compartilhados/exclusivos

Operação read_lock(X)

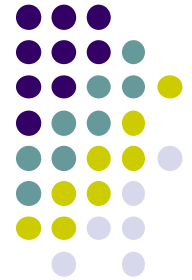
B: if LOCK(X) = “unlocked”
then { LOCK(X) = “read-locked”
no_of_reads(X) = 1 }
else if LOCK(X)=“read-locked”
then no_of_reads(X)++
else { wait (até que LOCK(X) = “unlocked” e
o gerente de lock acorde a transação)
goto B }

Bloqueios leitura/escrita ou compartilhados/exclusivos

Operação write_lock(X)

B: if LOCK (X) = “unlocked” then
LOCK (X) = “write-locked”;
else { wait (até que LOCK(X) = “unlocked” e o gerente de lock acorde a transação)
goto B }

Bloqueios leitura/escrita ou compartilhados/exclusivos



Operação unlock(X)

```
if LOCK(X) = “write-locked”
    then { LOCK(X) = “unlocked”
           wakeup up one of the transactions, if any }
    else if LOCK(X) = “read-locked”
           then { no_of_reads(X) = no_of_reads(X) - 1 ;
                  if no_of_reads(X) = 0
                     then { LOCK (X) = “unlocked”;
                            wakeup up one of the
                                transactions, if any } }
```

Bloqueios leitura/escrita ou compartilhados/exclusivos

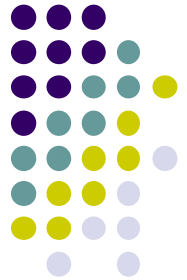
Para cada transação:

read_lock(X) ocorre...
- antes de ler(X)
- se ainda não tiver read_lock ou write_lock

write_lock(X) ocorre...
- antes de ler(X) com intenção de escrever(X)
- antes de escrever(X)
- se ainda não tiver write_lock

unlock(X) ocorre...
- depois de todas as operações ler(X)/escrever(X)
- apenas se tiver o lock de X

Bloqueios leitura/escrita ou compartilhados/exclusivos



Promoção e rebaixamento de lock:

Lock Upgrade

read_lock(x) → write_lock(x)

condição: não há outro read_lock em X

Lock Downgrade

write_lock(x) → read_lock(x)

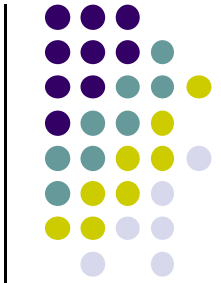
Garantindo a serialização

Usar bloqueios binários ou de leitura/escrita nas transações não garante a serialização de planos de execução sobre si mesmos.
Sejam as transações:

T1:	T2
read_lock(Y)	read_lock(X)
read_ítem(Y)	read_ítem(X)
unlock(Y)	unlock(X)
write_lock(X)	write_lock(Y)
read_ítem(X)	read_ítem(Y)
X=X+Y	Y=X+Y
write_ítem(X)	write_ítem(Y)
unlock(X)	unlock(Y)

Valores iniciais: X=20,Y=30
Resultado do plano T1 seguido de T2: X=50,Y=80
Resultado do plano T2 seguido de T1: X=70, Y=50

Garantindo a serialização



Bloqueio em duas fases (two phase lock)

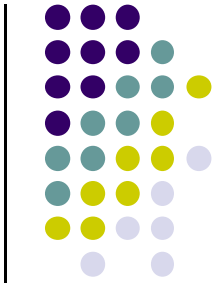
Uma transação segue o protocolo de bloqueio em duas fases se ***todas*** as operações (read_lock e write_lock) precedem a ***primeira*** operação de desbloqueio na transação. Há uma primeira fase de crescimento e outra de encolhimento (segunda fase) onde bloqueios podem ser liberados mas não podem se adquiridos.

Protocolo de bloqueio de duas fases:

- Fase de crescimento:
  Transação pode obter bloqueios,
  mas não pode liberar

- Fase de encolhimento:
  Transação pode liberar bloqueios,
  mas não pode obter

Garantindo a serialização



Bloqueio em duas fases (two phase lock)

Sejam as transações:

T1	T2
read_lock(Y)	read_lock(X)
read_item(Y)	read_item(X)
wite_lock(X)	write_lock(Y)
unlock(Y)	unlock(X)
read_item(X)	read_item(Y)
X=X+Y	Y=X+Y
write_item(X)	write_item(Y)
unlock(X)	unlock(Y)

Estas transações seguem o protocolo 2PL mas podem produzir deadlock.

O protocolo 2PL - variações

2PL estático ou conservador (livre de deadlock)

- Bloqueia todos os itens a serem lidos/gravados antes de iniciar a transação
- Exige pré-declaração de locks (leituras/gravações) no início da transação (quando a transação começa ela está na fase de encolhimento)

2PL Estrito (não é livre de deadlock)

- Não libera write_locks até o commit ou abort
- Nenhuma outra transação pode ler ou escrever um item que seja escrito por T, a menos que T tenha efetivado

2PL Rigoroso

- Não libera read_locks/write_locks até o commit ou abort (a transação está em fase de expansão até o seu final)

Lidando com deadlock e starvation

O uso de bloqueios pode gerar deadlock ou starvation.

Uso de protocolos de prevenção de deadlocks usados em 2PL conservador:

1) A transação tenta bloquear todos os itens, se algum não puder ser bloqueado, todos serão liberados. A transação espera e tenta o bloqueio dos itens mais tarde.
2) Ordenação de todos os itens do BD e fazer os bloqueios obedecendo esta ordem.
3) Uso de timestamps (marcas de tempo nas transações). A transação mais velha tem o menor timestamp.

Deadlock/Starvation: Protocolos de transações com timestamp

O uso de bloqueios pode gerar deadlock ou starvation.

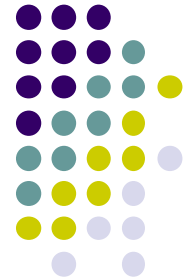
Uso de protocolos de prevenção de deadlocks usados em **2PL conservador**:

1) A transação tenta bloquear todos os itens, se algum não puder ser bloqueado, todos serão liberados. A transação espera e tenta o bloqueio dos itens mais tarde.
2) Ordenar todos os itens do BD e fazer os bloqueios obedecendo esta ordem.

Os protocolos descritos reduzem a concorrência e são difíceis de serem implementados, sujeitando as transações a falhas.

Solução: Uso de timestamps (marcas de tempo nas transações). A transação mais velha tem o menor timestamp.

Deadlock/Starvation: Protocolos de transações com timestamp



Prevenção de deadlock/Transações com timestamps(TS)

Considere que Ti quer lock(X) e Tj tem lock(X)

Regra **"esperar-morrer"**

- Se TS(Ti) < TS(Tj), (Ti é mais velha que Tj)
  Ti pode esperar
  Caso contrário, (Ti é mais nova que Tj)
  aborta Ti (morre) e Ti reinicia mais tarde com o mesmo TS

Regra **"ferir-esperar"**

- Se TS(Ti) < TS(Tj), (Ti é mais velha que Tj)
  aborta Tj (Ti fere Tj) e reinicia mais tarde com o mesmo TS
  Caso contrário, (Ti é mais nova que Tj)
  Ti espera

Deadlock/Starvation: Protocolos de transações com timestamp

Detecção de deadlock/Transações com timestamps(TS)

- Verifica o número de transações executadas concorrentemente ou o tempo para uma transação bloquear os seus itens. Ação:
  - Aborta uma das transação em deadlock
  - Algoritmo de seleção da vítima – evitar selecionar transações executadas há muito tempo

- Timeout: aborta transação com tempo de execução maior do que o tempo estimado para timeout, independentemente se houve deadlock ou não.

Deadlock/Starvation: Protocolos de transações com timestamp

Starvation/Transações com timestamps(TS)

Transação não pode prosseguir por um período indefinido

- Soluções
  - Primeiro a chegar, primeiro a ser atendido

    As transações estão habilitadas a bloquear um item na sequência em que elas solicitaram o bloqueio
  - Prioridade aumenta com a espera

    Aumentar a prioridade de uma transação quanto maior for o seu tempo de espera.

Perguntas?