Transacções

Propriedades Transaccionais

- **A**tomicidade: Ou todas ou nenhumas modificações
- **C**onsistência: A transacção deixa os dados válidos
- **I**solamento: As transacções aparentam correr sequencialmente
- **D**urabilidade: Alterações feitas por transacções confirmadas persistem

Atomicidade

- Considere uma transferência entre contas bancárias:

```
begin transaction
    update contas set saldo=saldo-100 where nib=2
    update contas set saldo=saldo+100 where nib=6
commit transaction
```

- Atomicidade garante que os 100 EUR não são perdidos nem duplicados nas duas contas

Consistência

- Exemplo de uma transferência inconsistente:

```
begin transaction
    update contas set saldo=saldo-100 where nib=2
    update contas set saldo=saldo+200 where nib=6
commit transaction
```

- Responsabilidade do programador (e da ausência de bugs...)

Isolamento

- Considere dois levantamentos a acontecer simultaneamente:

  begin transaction

  begin transaction

  select saldo from contas where nib=2 (sim, é 110!)

  select saldo from contas where nib=2 (sim, é 110!)

  update contas set saldo=saldo-100 where nib=2

  update contas set saldo=saldo-100 where nib=2

  commit transaction (saldo é 10)

  commit transaction (?)

- Isolamento garante que uma das transacções aborta ou então...

Isolamento

- ...atrasa algumas operações:

begin transaction

begin transaction

select saldo from contas where nib=2 (sim, é 110!)

update contas set saldo=saldo-100 where nib=2

select adiado para mais tarde...

commit transaction (saldo é 10)

select saldo from contas where nib=2 (não, é 10!)

commit transaction (saldo é 10)

Durabilidade

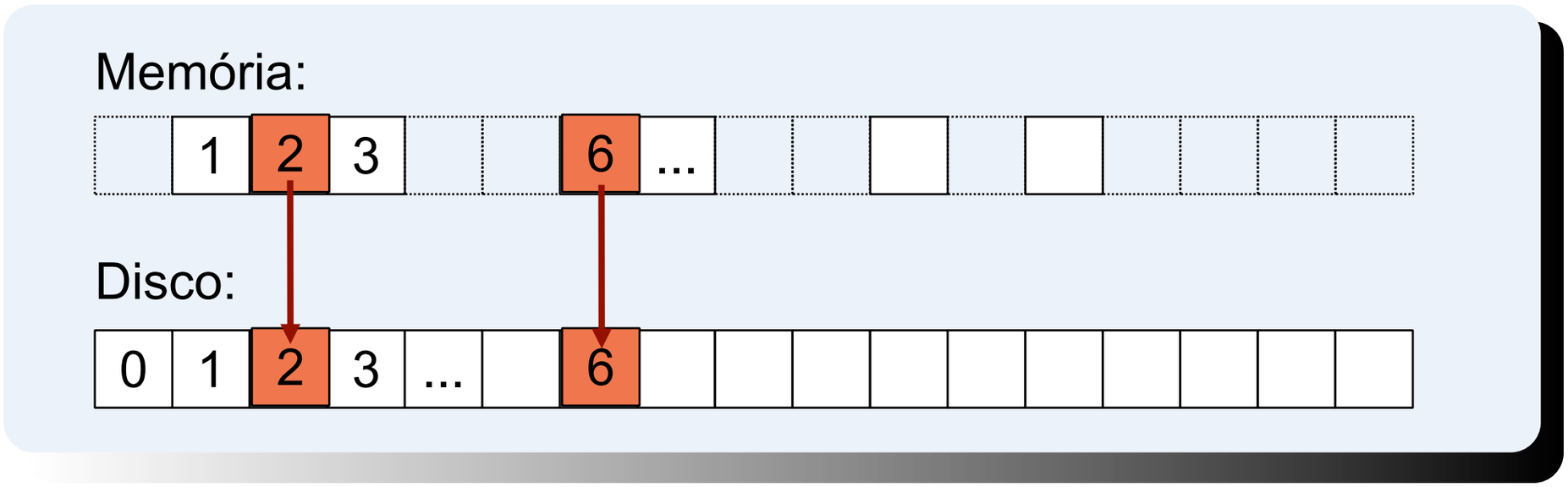
- Considere um levantamento:

```
begin transaction
        update contas set saldo=saldo-100 where nib=2
commit transaction
```

- Durabilidade garante que depois de confirmada a transacção, o saldo persiste mesmo em caso de falha

Atomicidade vs. Durabilidade

- Como garantir ambas as propriedades em caso de falha?
  - Uma transacção pode implicar modificar a base de dados em vários sitios
  - Em caso de falha, pode ter sido feito um qualquer sub-conjunto de modificações



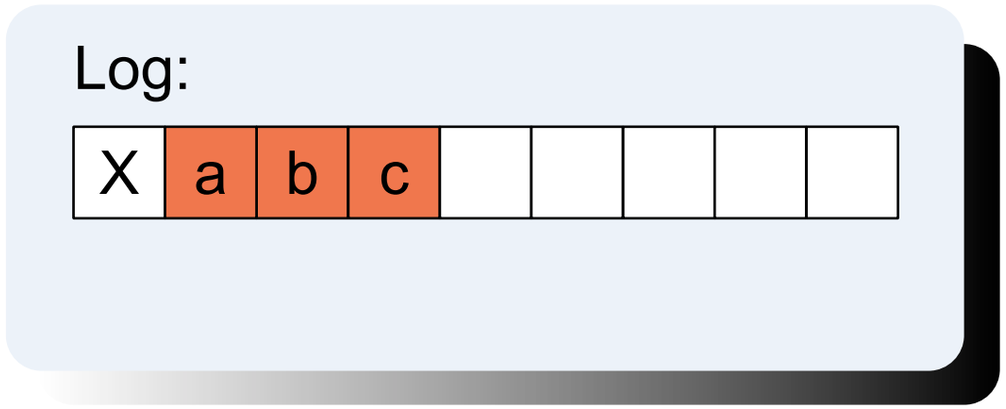
Atomicidade vs. Durabilidade

- Durabilidade sem atomicidade é fácil:
  - Actualizar a base de dados antes de confirmar
- Atomicidade sem durabilidade é fácil:
  - Em caso de falha, apaga-se a base de dados (ou repõe-se um backup anterior...)

Atomicidade

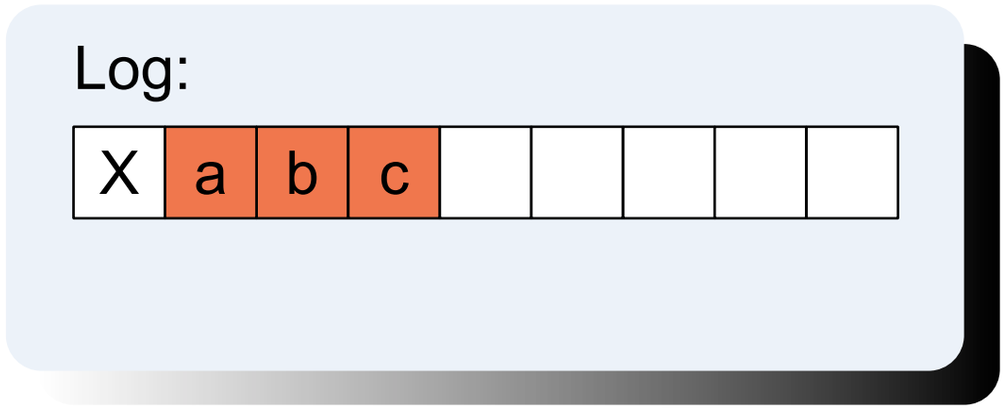
- Uma sequência de modificações atómicas não é uma modificação atómica
- Uma técnica geral para o conseguir exige que:
  - Exista uma operação de modificação de dados garantidamente atómica
  - Seja possível esperar pela conclusão das operações de modificação parciais
  - As modificações não sejam perdidas com um reínicio da máquina

Ficheiro de Log



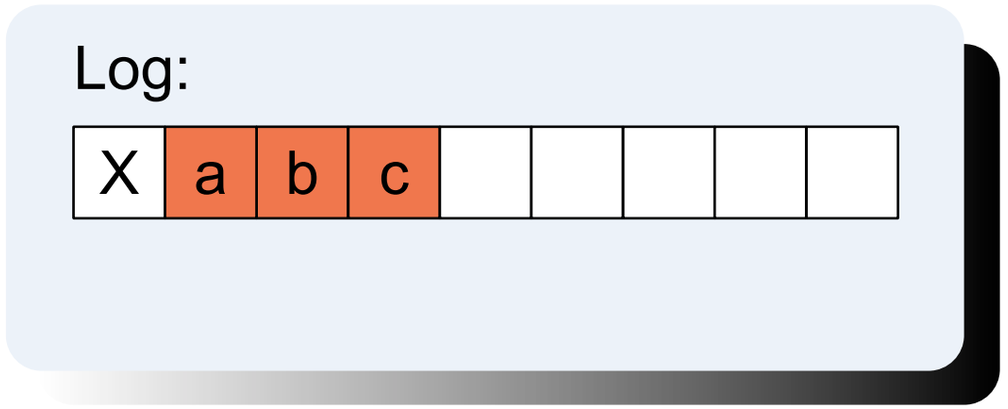
Ficheiro de Log

- Um ficheiro de log contém três secções:



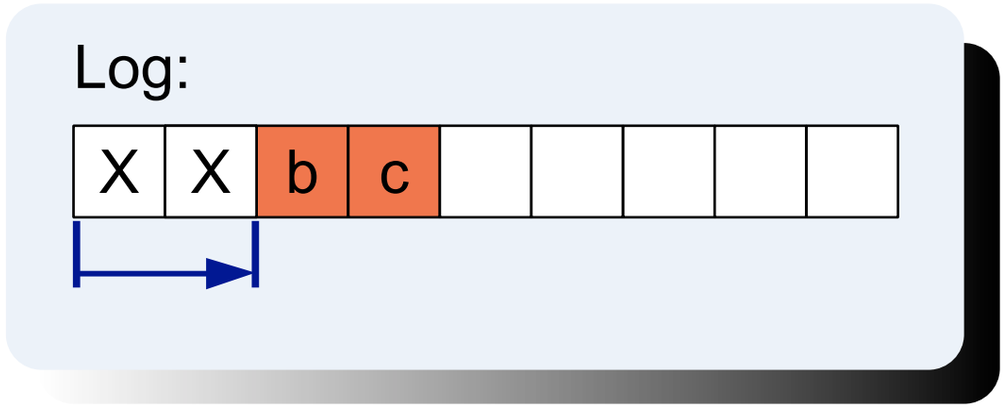
Ficheiro de Log

- Um ficheiro de log contém três secções:
  - Registos apagados



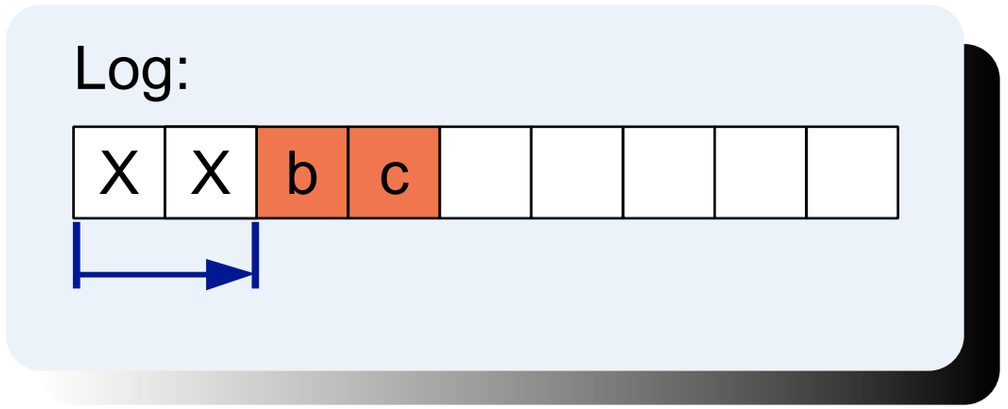
Ficheiro de Log

- Um ficheiro de log contém três secções:
  - Registos apagados



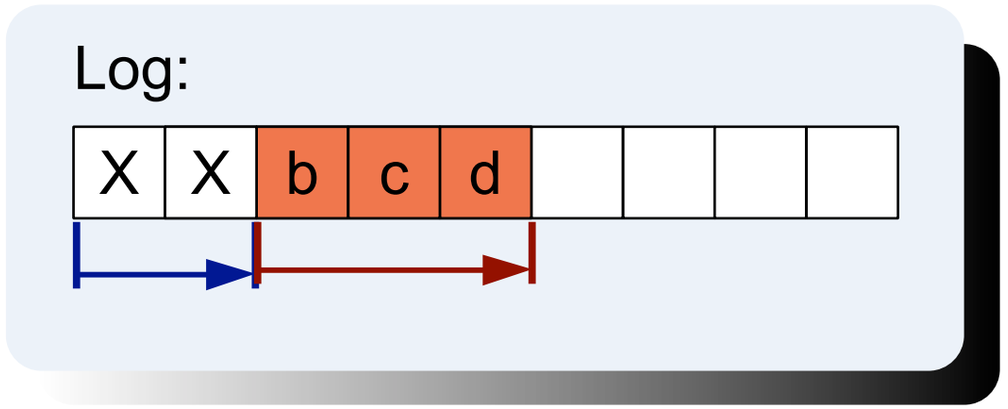
Ficheiro de Log

- Um ficheiro de log contém três secções:
  - Registos apagados
  - Registos activos



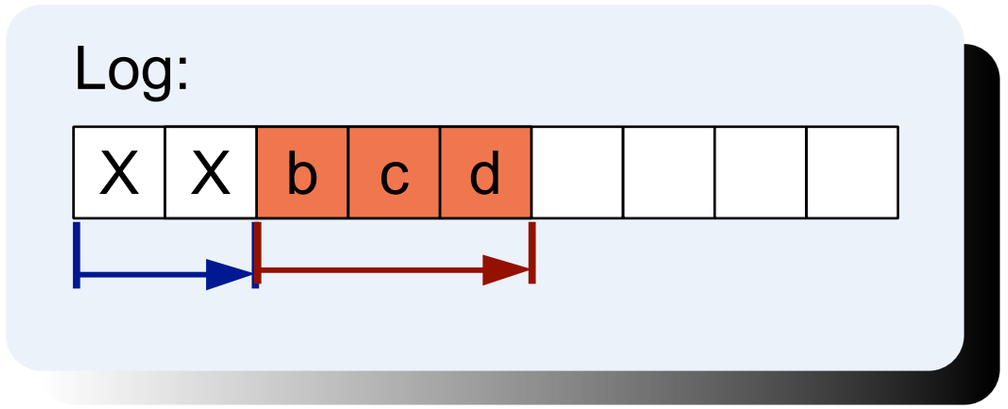
Ficheiro de Log

- Um ficheiro de log contém três secções:
  - Registos apagados
  - Registos activos



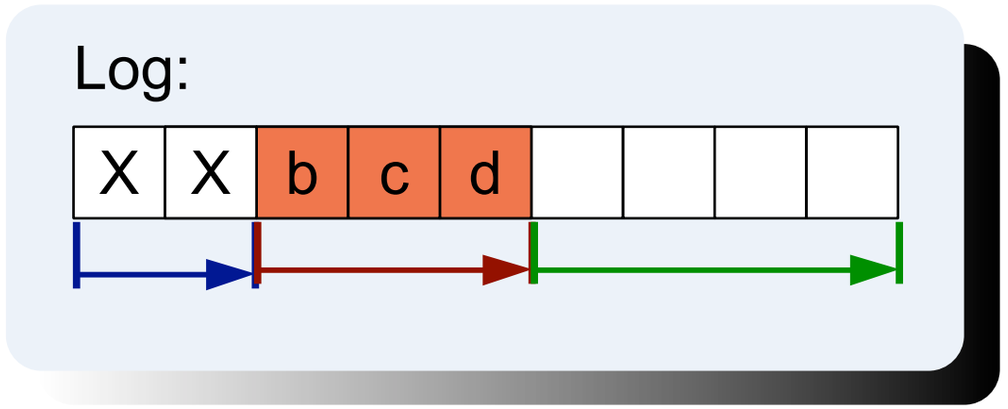
Ficheiro de Log

- Um ficheiro de log contém três secções:
  - Registos apagados
  - Registos activos
  - Registos livres



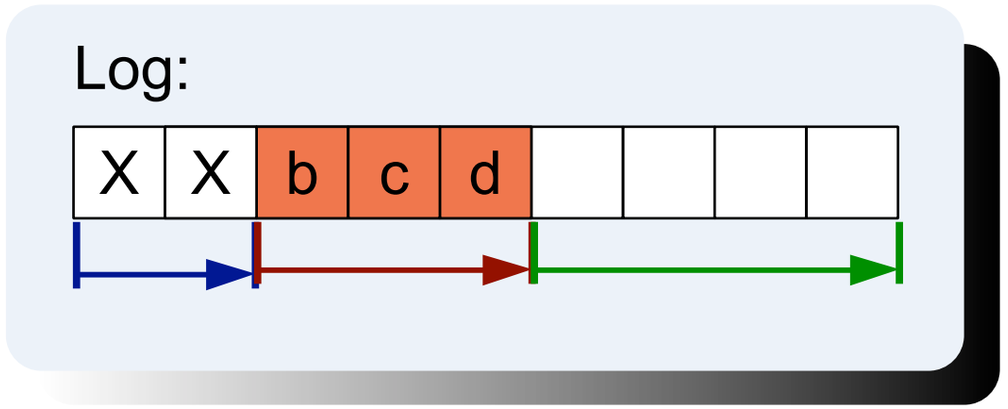
Ficheiro de Log

- Um ficheiro de log contém três secções:
  - Registos apagados
  - Registos activos
  - Registos livres



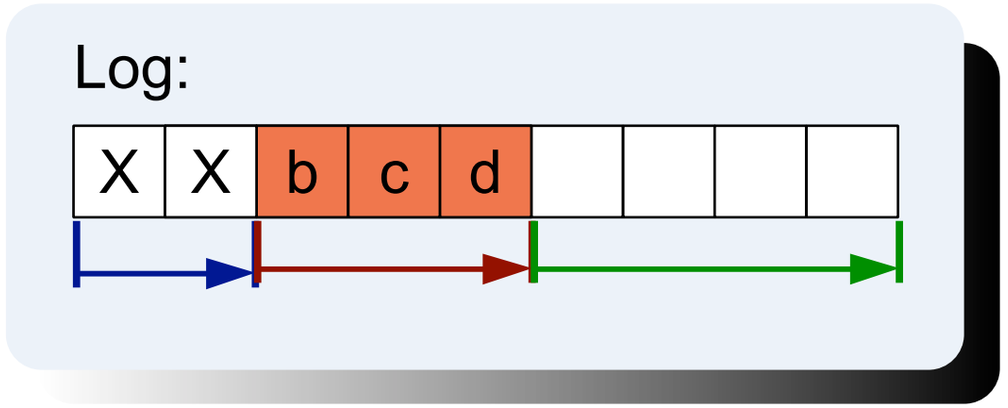
Ficheiro de Log

- Um ficheiro de log contém três secções:
  - Registos apagados
  - Registos activos
  - Registos livres
- A modificação de cada um dos registos (de livre para activo ou de activo para apagado) é atómica



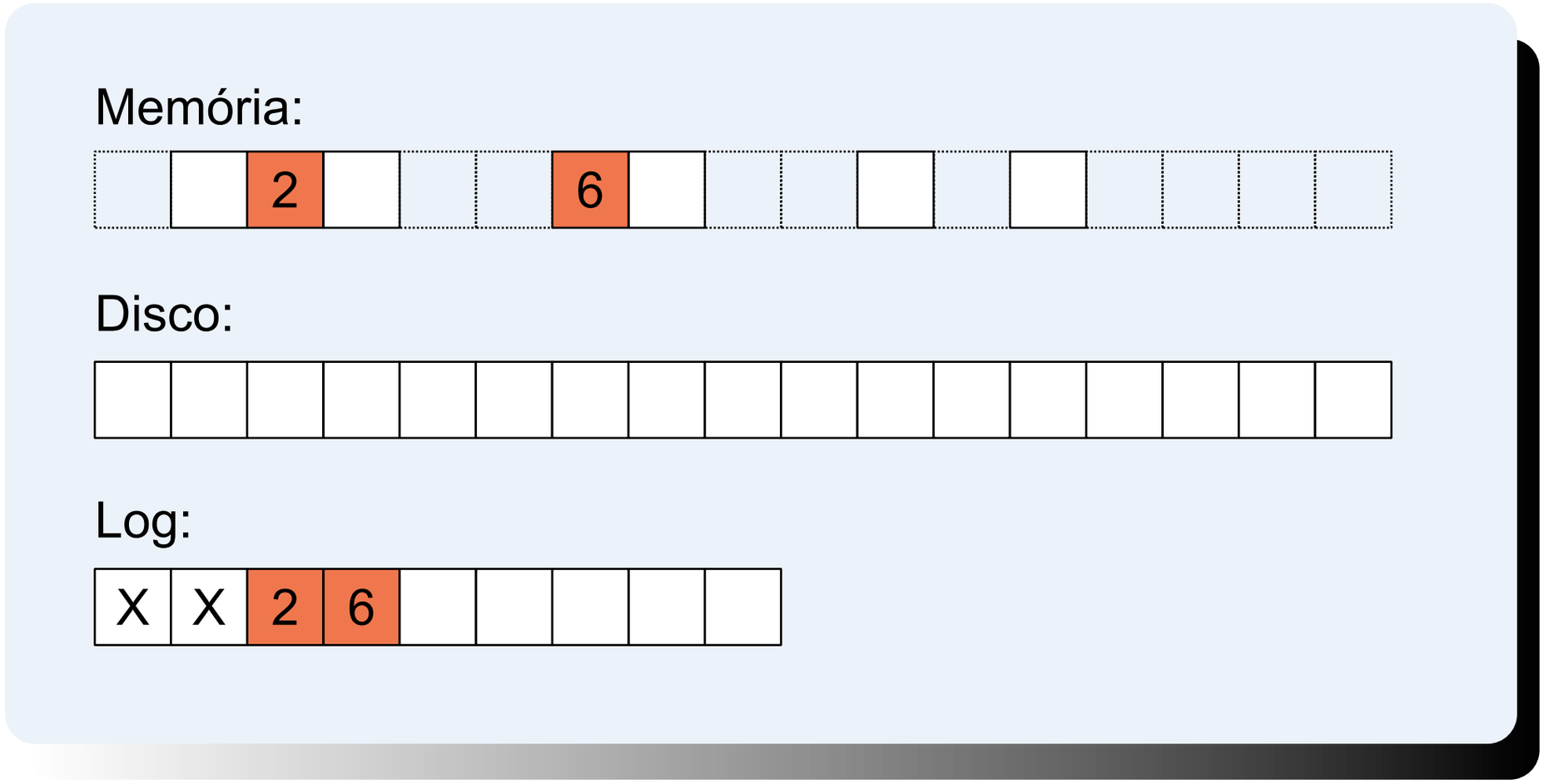
Ficheiro de Log

- Um ficheiro de log contém três secções:
  - Registos apagados
  - Registos activos
  - Registos livres
- A modificação de cada um dos registos (de livre para activo ou de activo para apagado) é atómica
- É possível reutilizar o ficheiro de log depois de cheio



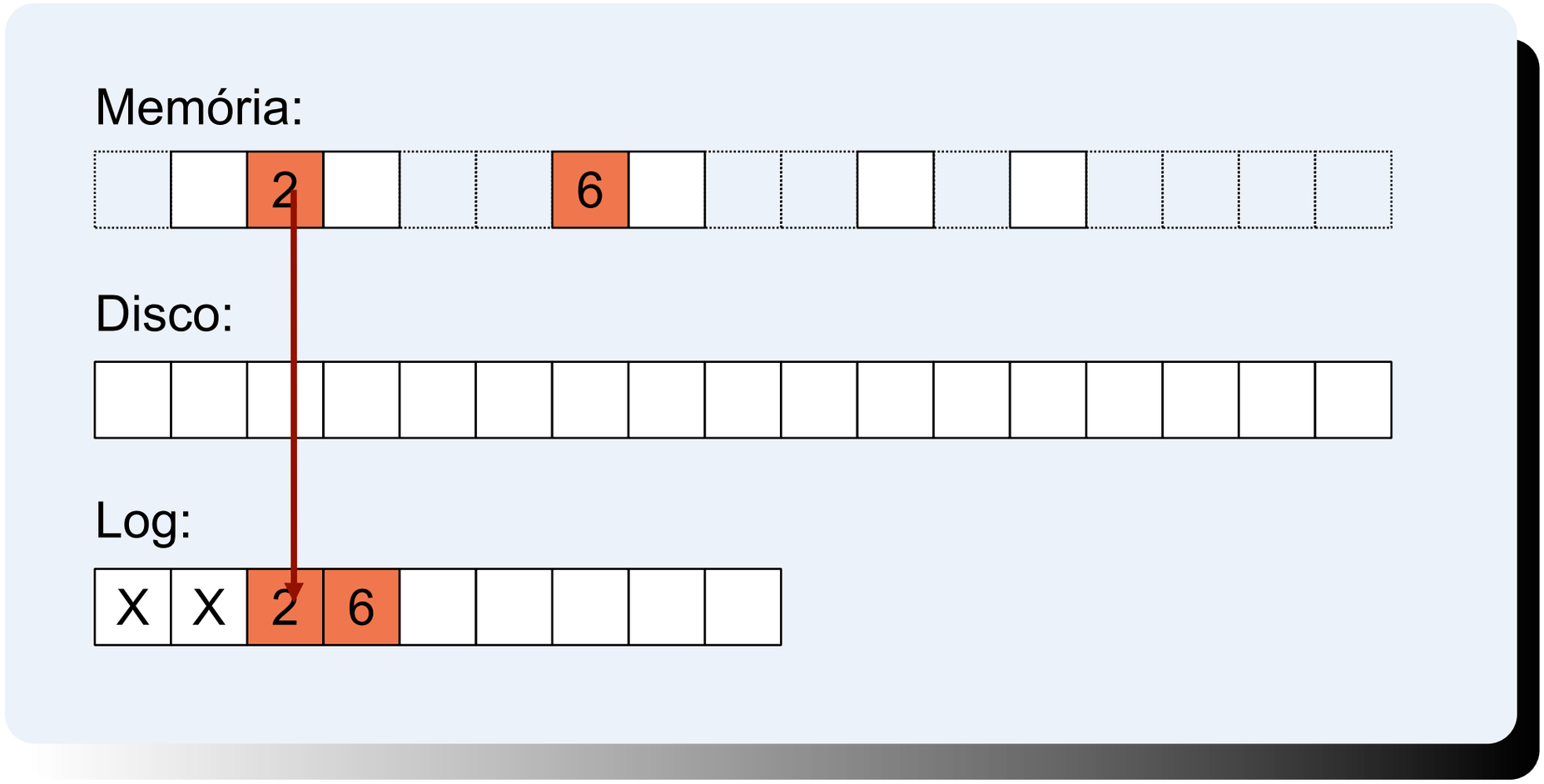
Método "Redo"

- Items modificados são copiados para o log:



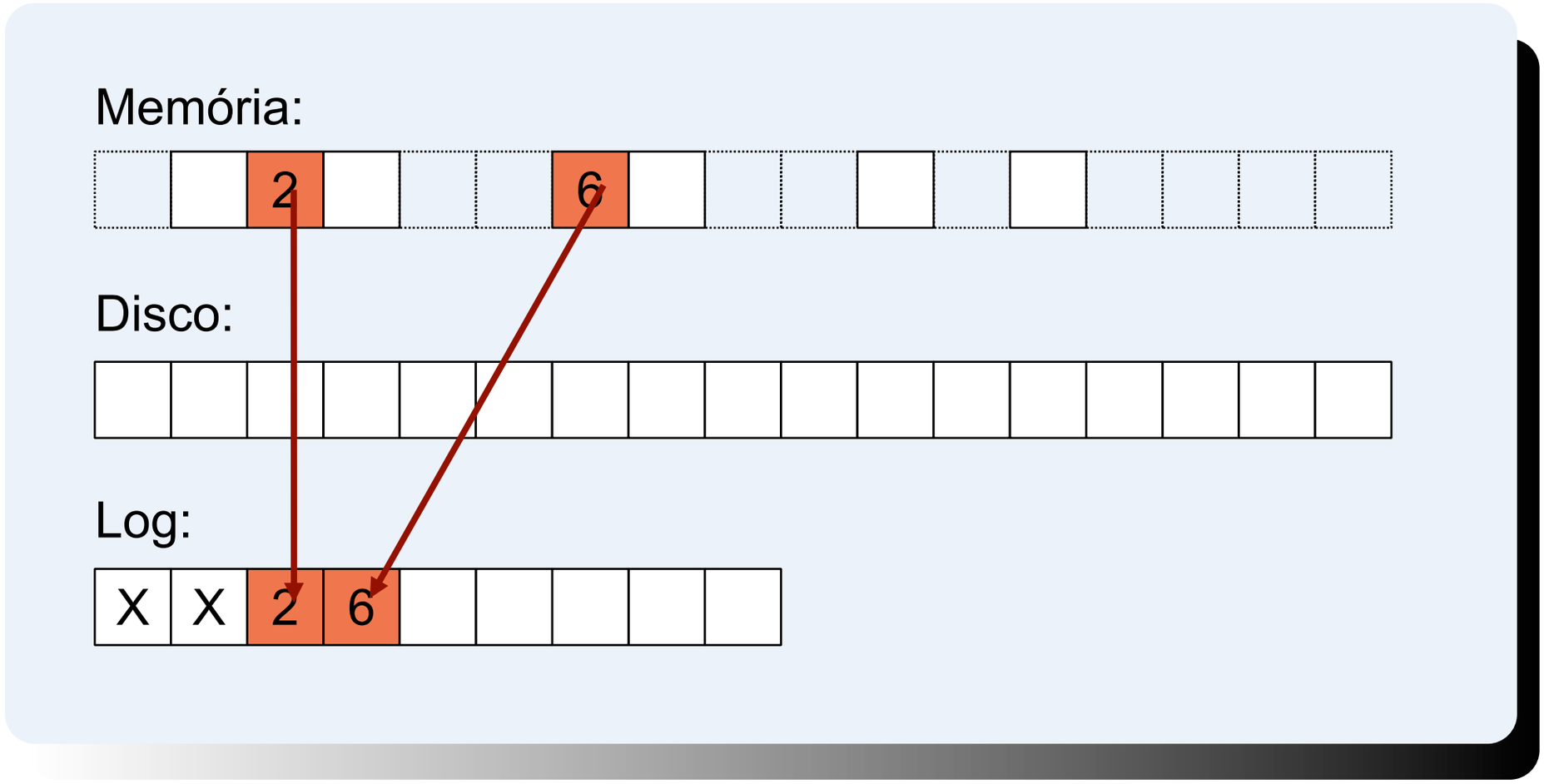
Método "Redo"

- Items modificados são copiados para o log:



Método "Redo"

- Items modificados são copiados para o log:

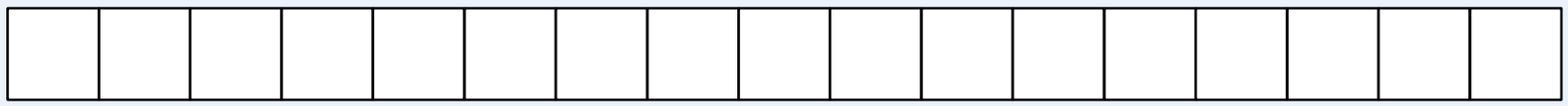


Método "Redo"

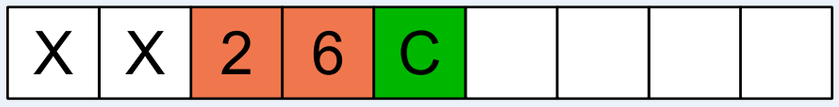
- O marcador de confirmação é acrescentado:

Memória:

Disco:

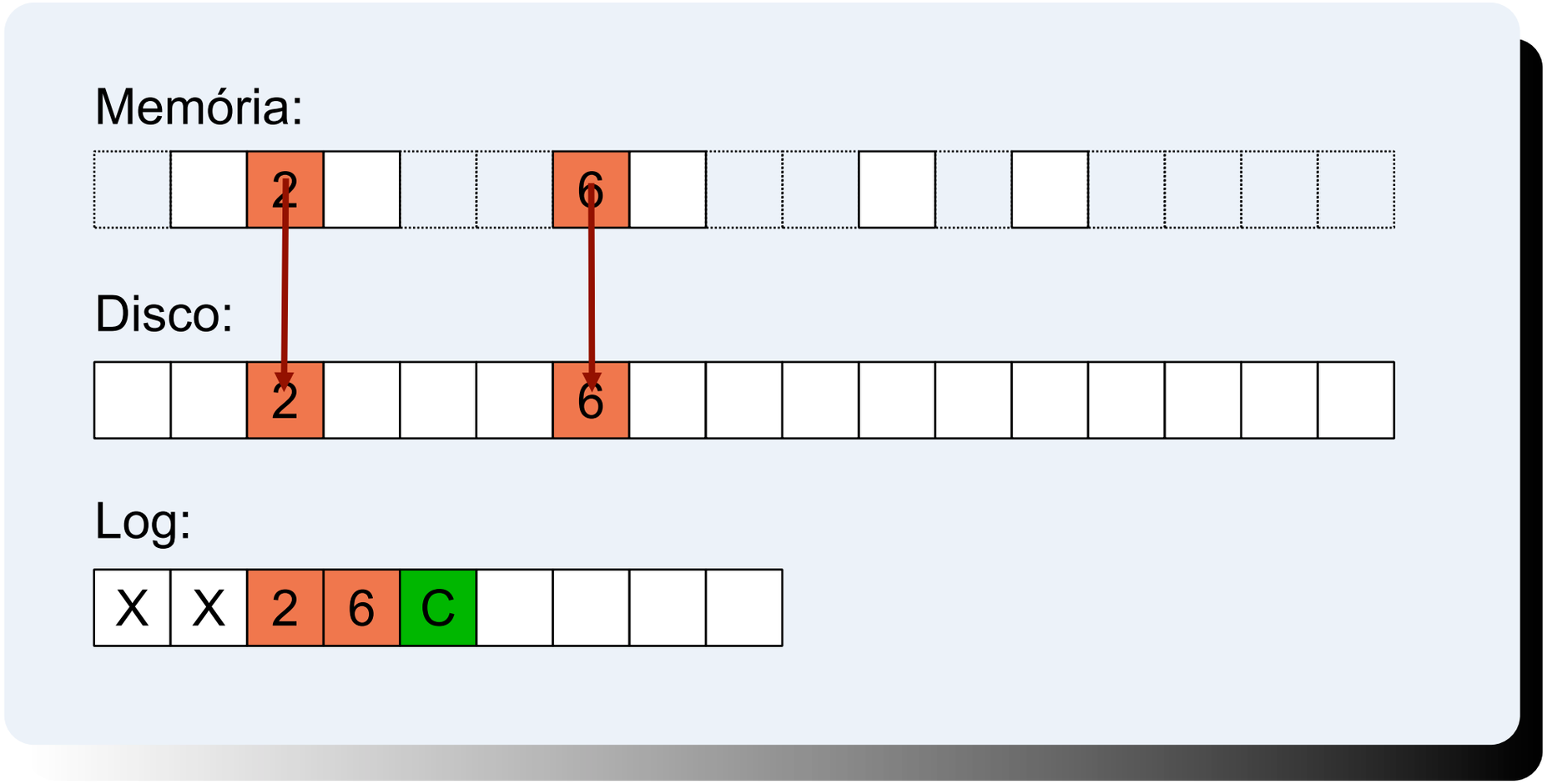


Log:



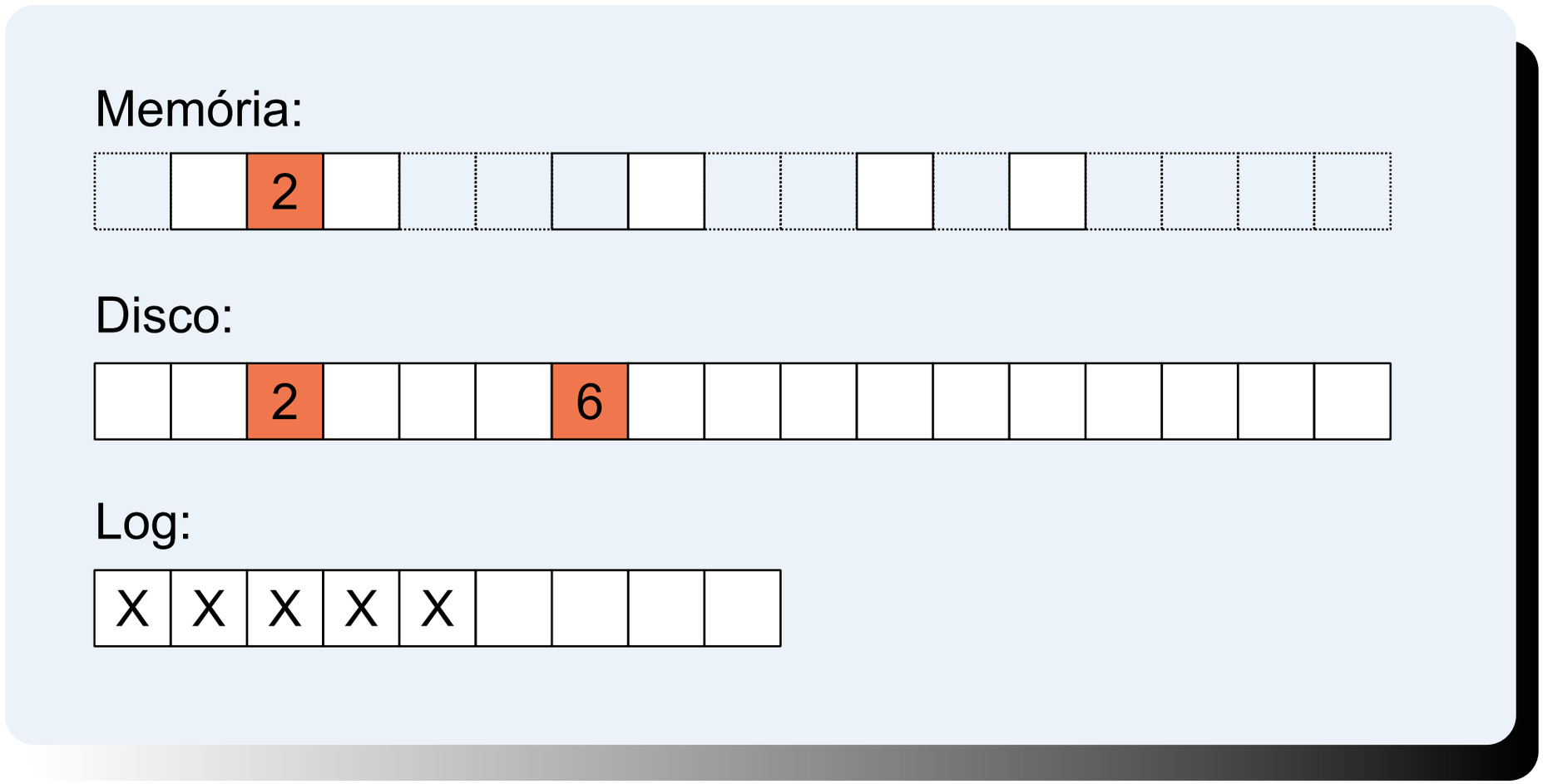
Método "Redo"

- Quando oportuno, os items modificados são escritos na base de dados em disco:



Método "Redo"

- Finalmente, o log é apagado e os items podem ser removidos da memória:

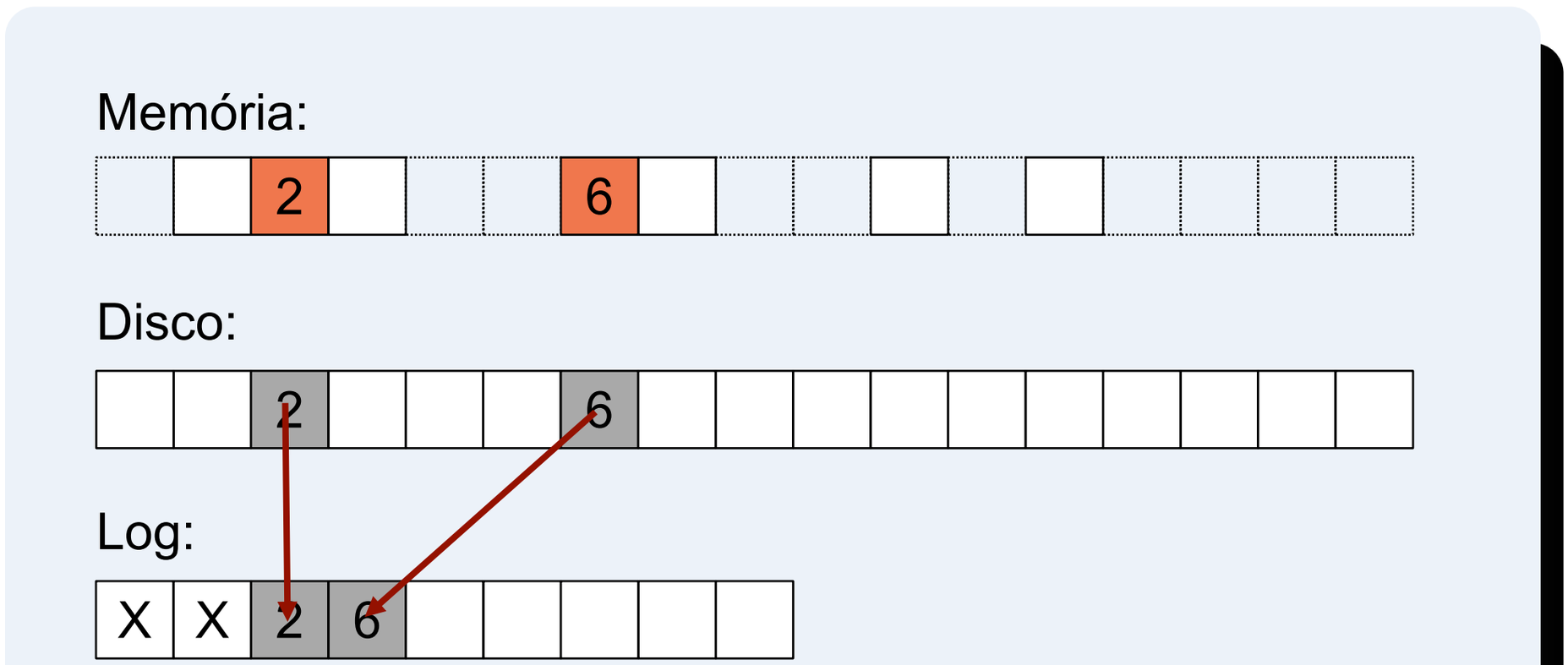


Método "Redo"

- Processo de recuperação:
  - Os items no log com marcador de confirmação são copiados para a BD
- Vantagens:
  - A transacção pode ser confirmada sem necessidade de actualizar a BD (no force)
- Desvantagens:
  - A BD só pode ser modificada depois da transacção ser confirmada pelo utilizador
  - Assume que a memória é suficientemente grande para conter todos os items modificados (no steal)

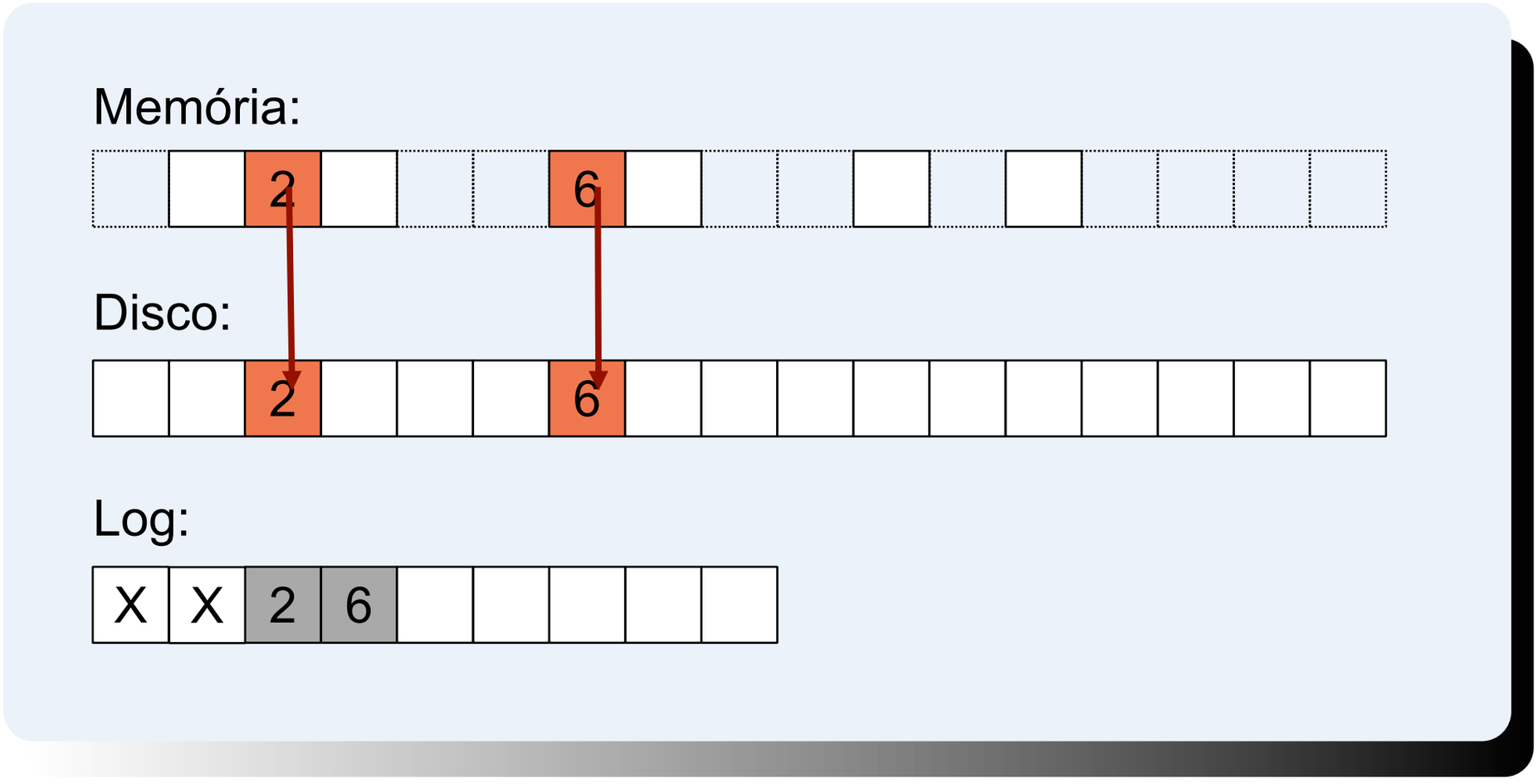
Método "Undo"

- Os valores originais dos items modificados são copiados para o log:



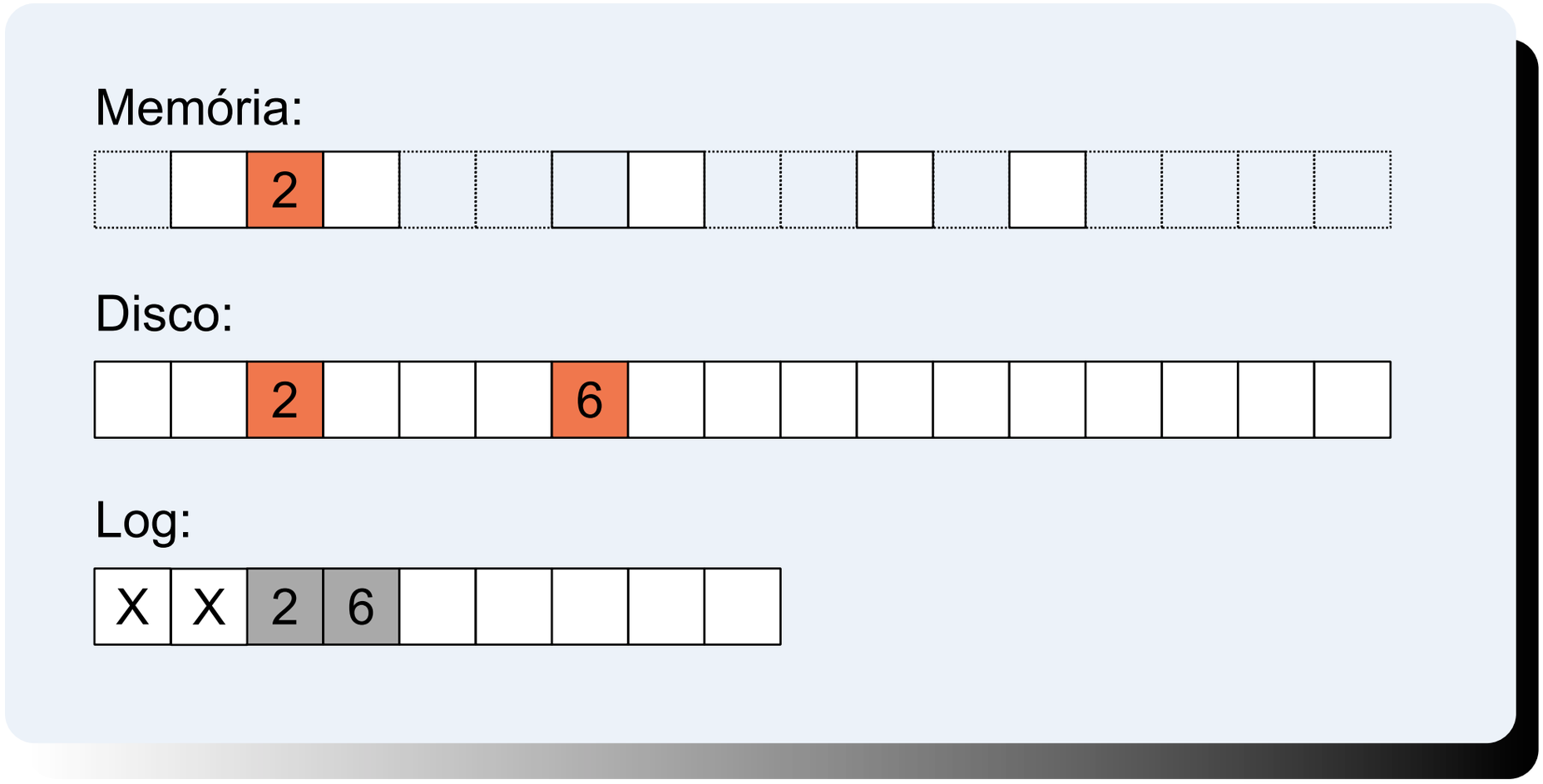
Método "Undo"

- Os items podem então ser modificados directamente na BD:



Método "Undo"

- Os items modificados podem agora ser retirados da memória:



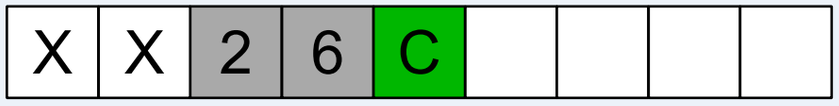
Método "Undo"

- O marcador de confirmação é acrescentado:

Memória:

Disco:

Log:



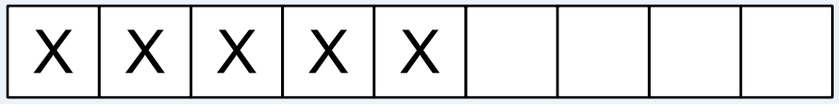
Método "Undo"

- Finalmente, o log é apagado:

Memória:

Disco:

Log:

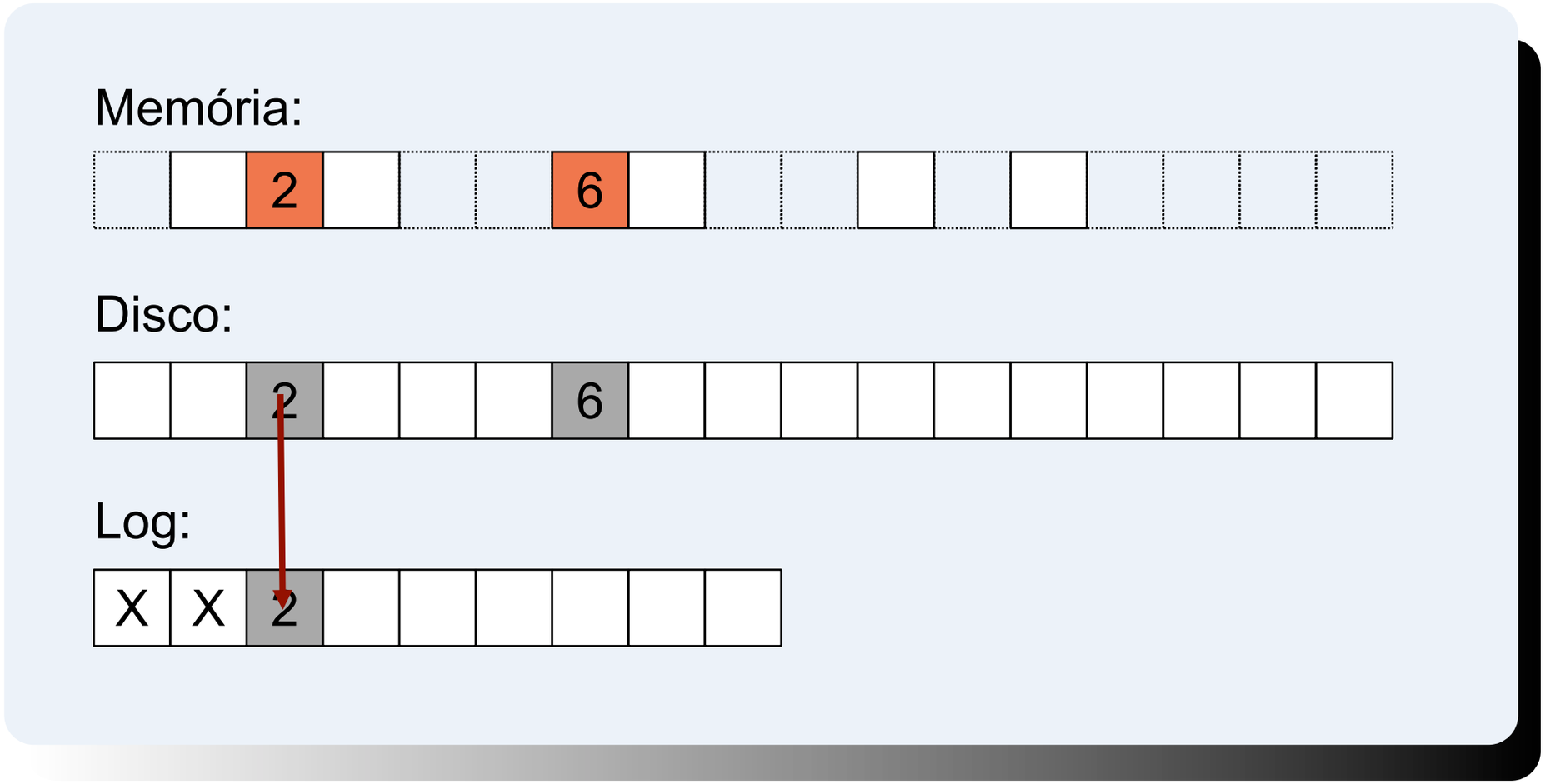


Método "Undo"

- Processo de recuperação:
  - Copiar de volta os valores originais dos items sem marcador de confirmação
- Vantagens:
  - As modificações podem ser escritas para a BD antes do utilizador confirmar (steal)
  - Assume apenas que o log é suficientemente grande para conter os items modificados
- Desvantagens:
  - Todas as modificações têm ser escritas para a BD antes de confirmar (force)

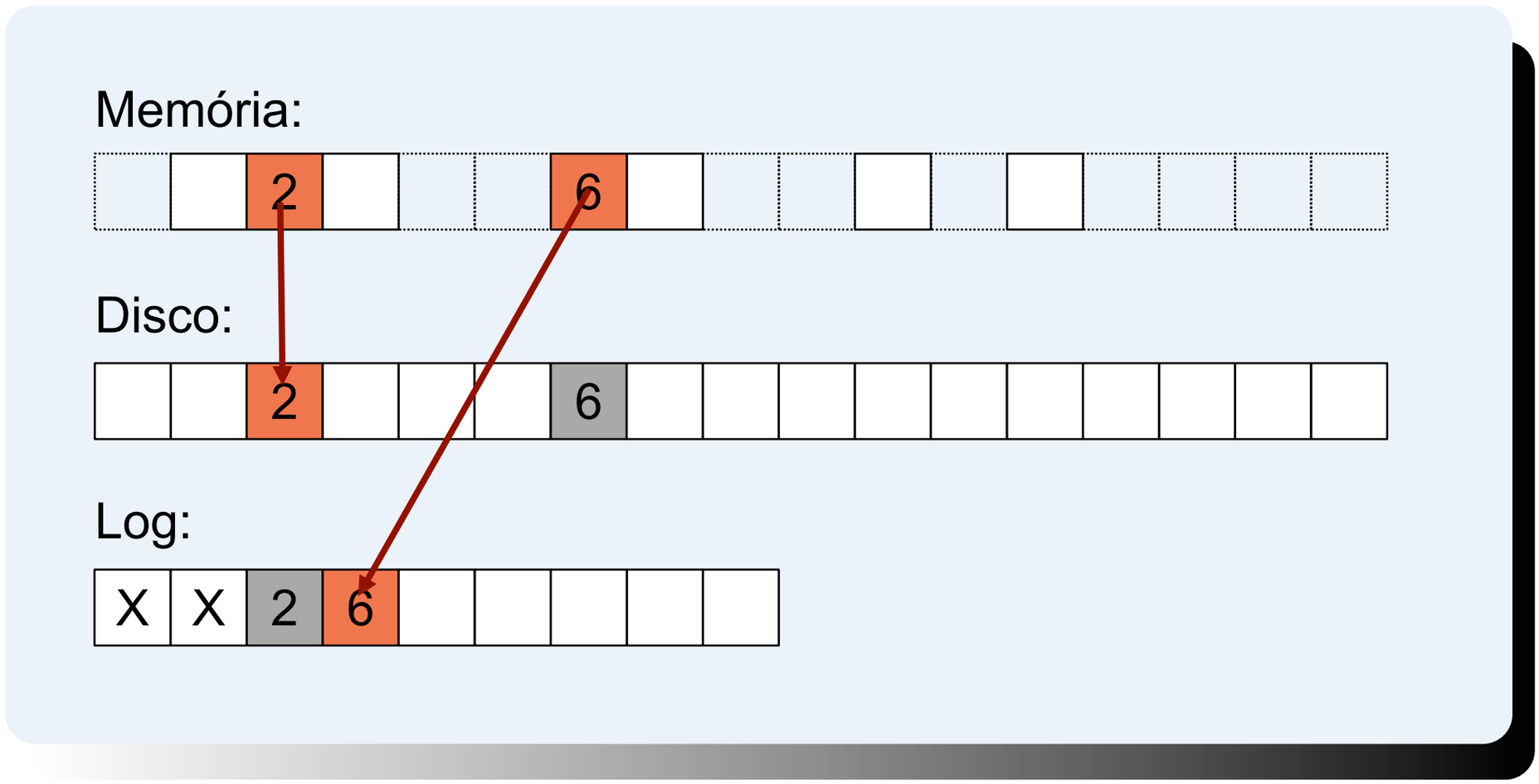
Método "Undo-Redo"

- Os valores originais de alguns dos items modificados são copiados para o log:



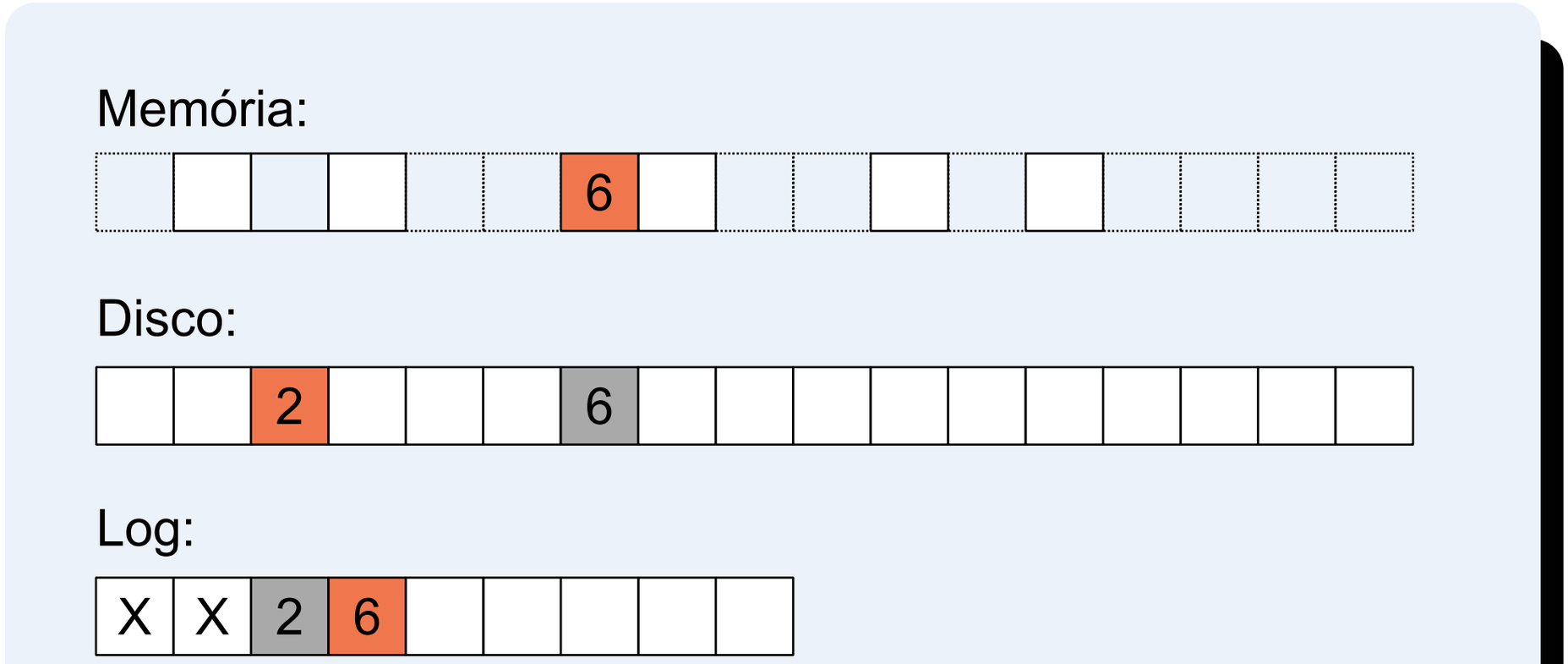
Método "Undo-Redo"

- Items modificados são copiados para o log ou para a BD (se existe cópia prévia no log):



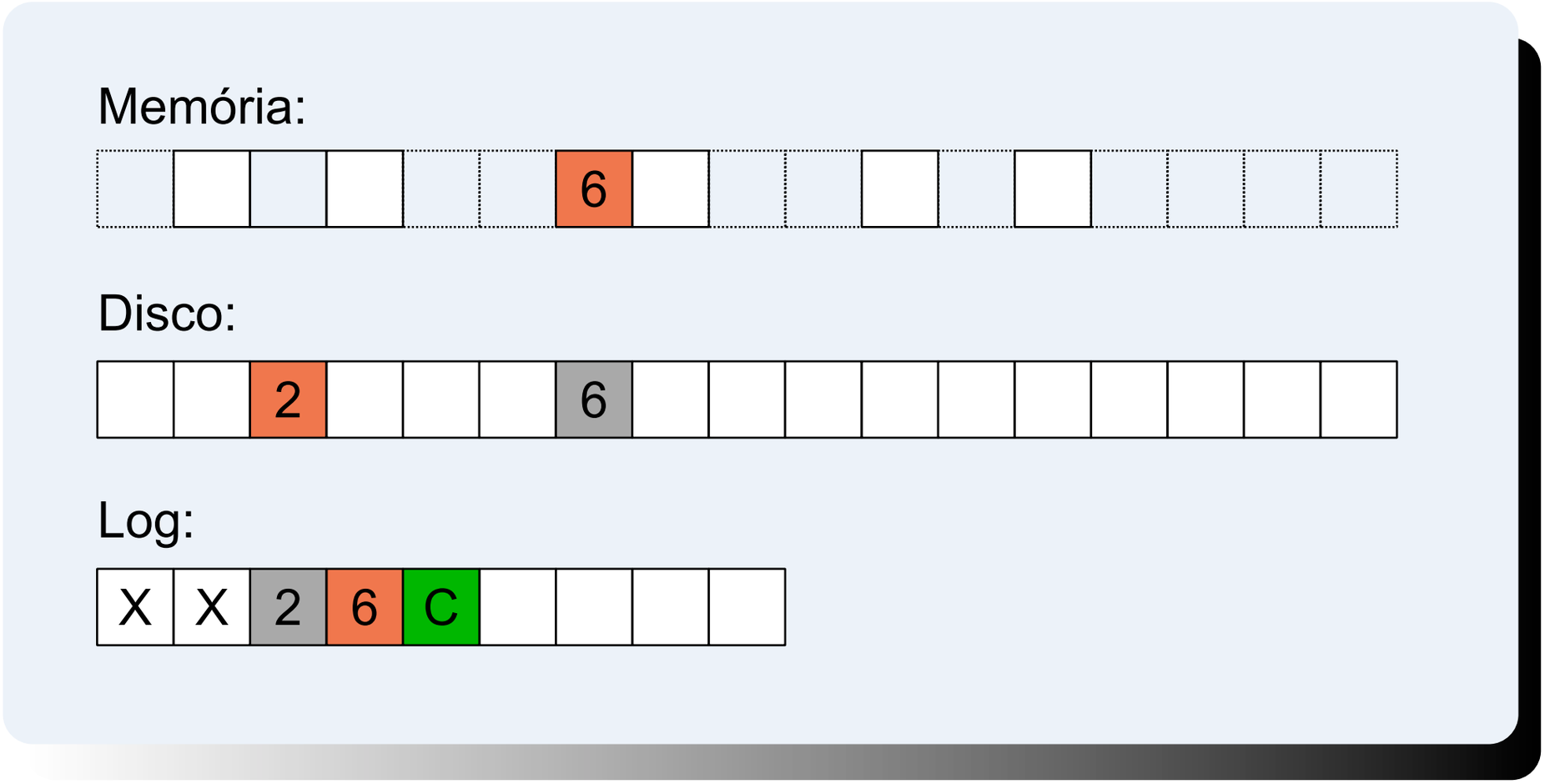
Método "Undo-Redo"

- Os items já copiados para a BD podem ser removidos da memória:



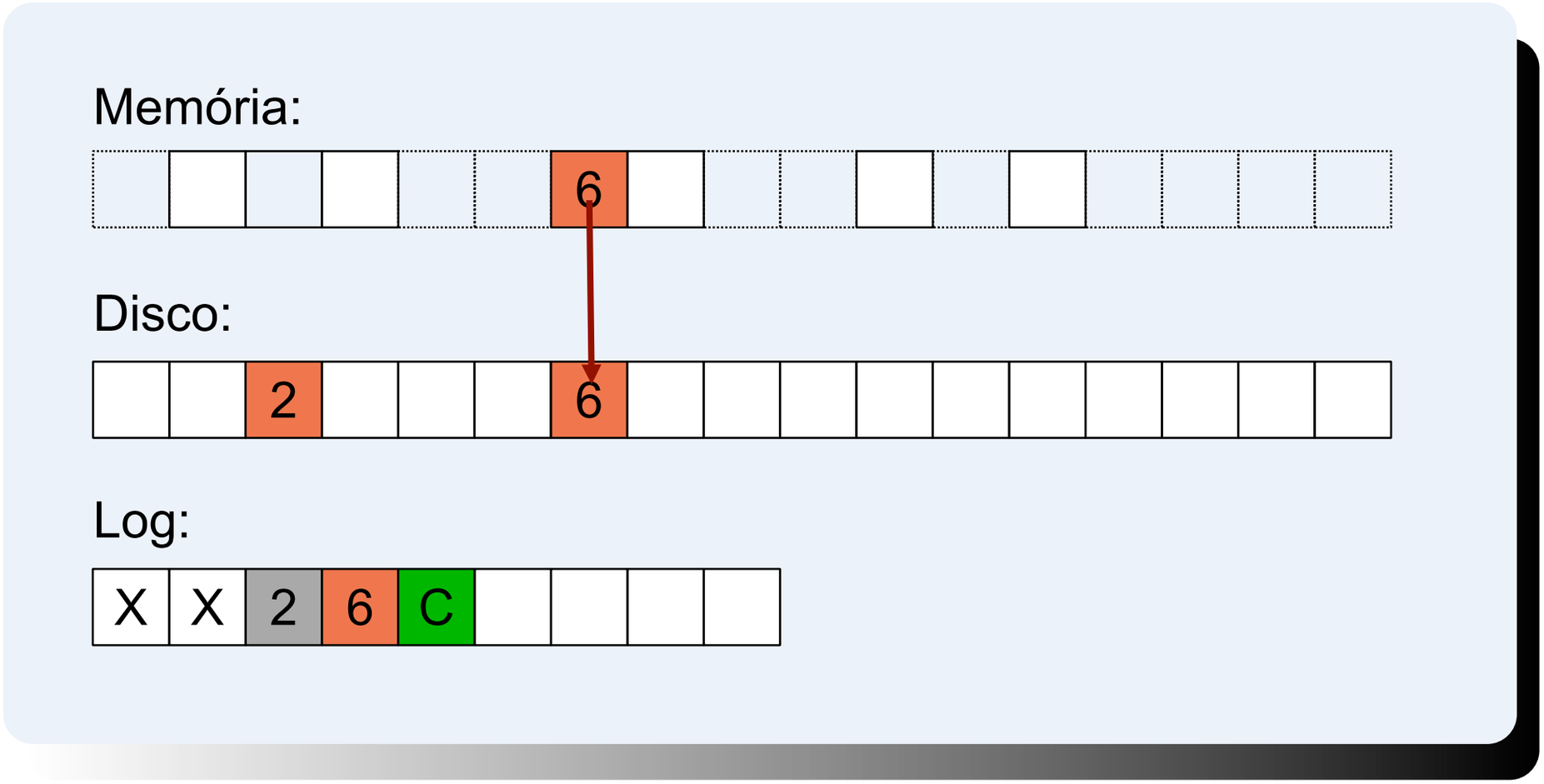
Método "Undo-Redo"

- O marcador de confirmação é acrescentado:



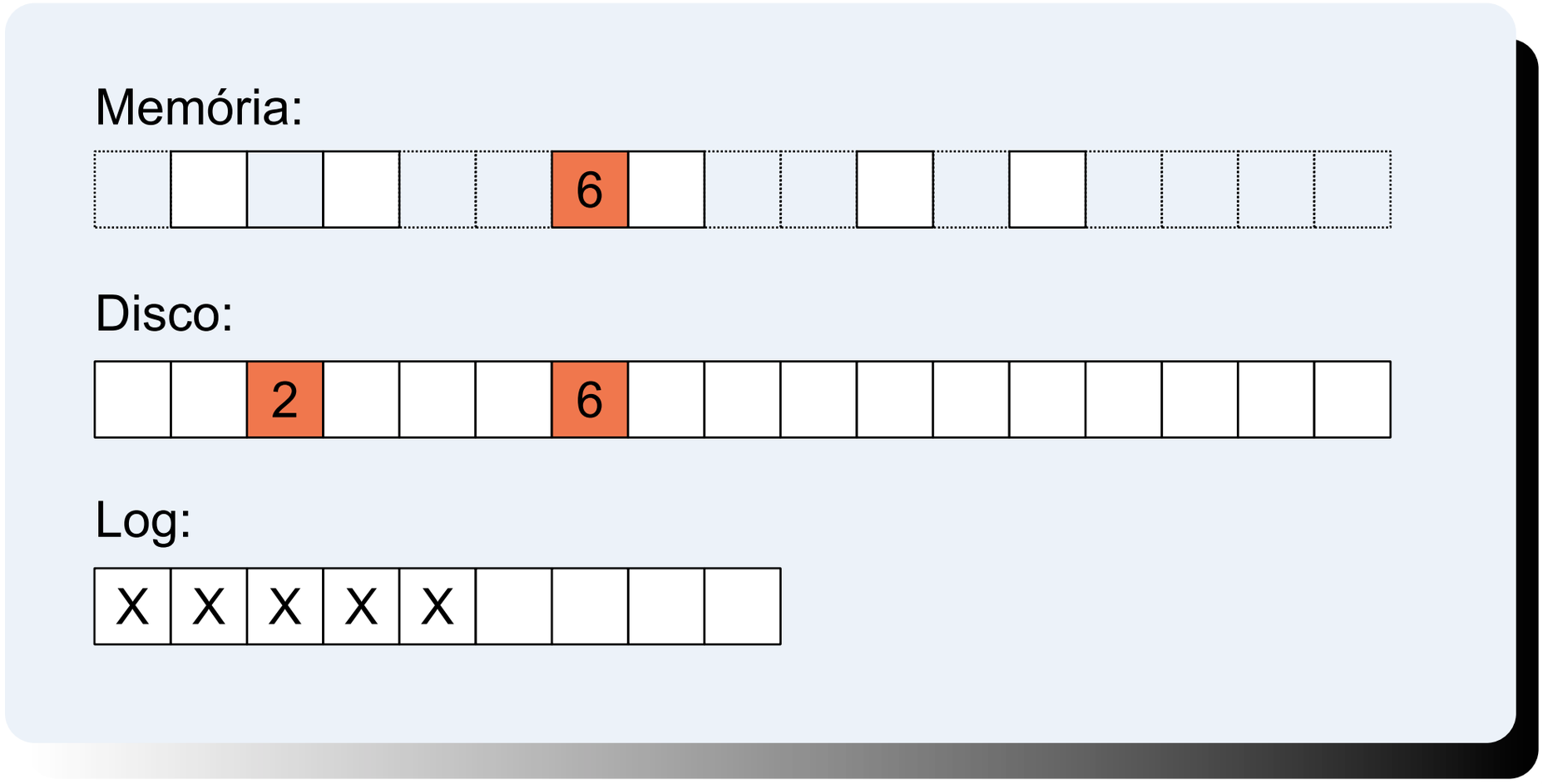
Método "Undo-Redo"

- Quando oportuno, os restantes items modificados são escritos na BD em disco:



Método "Undo-Redo"

- Finalmente, o log é apagado:



Método "Undo-Redo"

- Processo de recuperação:
  - Copiar de volta os valores originais dos items “Undo” sem marcador de confirmação
  - Os items “Redo” no log com marcador de confirmação são copiados para a BD
- Vantagens:
  - Items modificados podem ser escritos para a BD (steal) ou apenas armazenados no log (no-force) antes da confirmação
  - Assume apenas que o log é suficientemente grande para guardar todas as modificações

Transacções Distribuídas

- As propriedades transaccionais de atomicidade e durabilidade são atraentes para lidar com falhas em sistemas distribuídos:
  - Lidar com falhas parciais
  - Permitir recuperação independente

Transacções Distribuídas

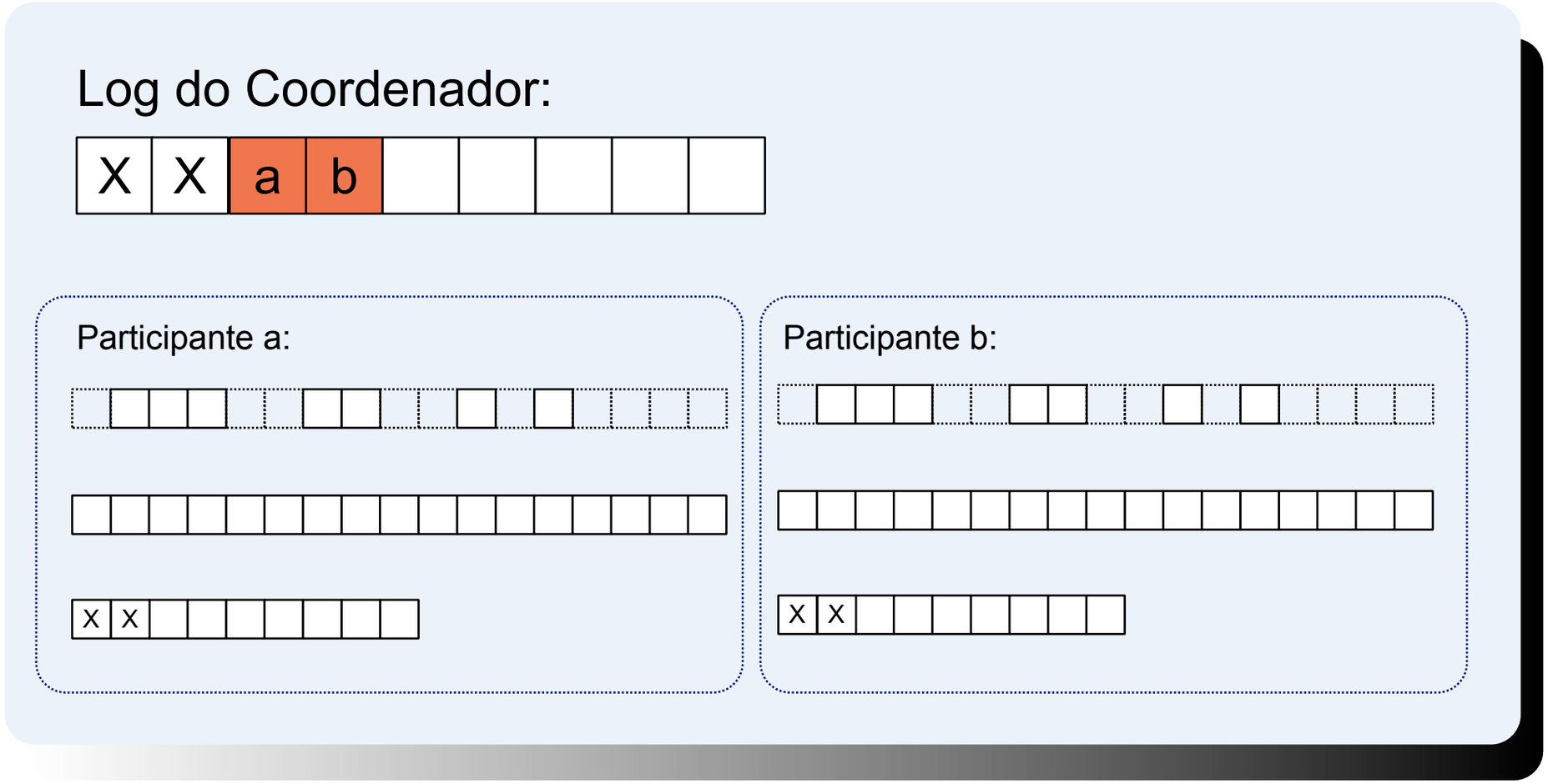
- Abordagem simplista:
  - Manter um log num coordenador
  - Executar um dos métodos conhecidos usando invocações remotas
- Problemas:
  - Tráfego de rede excessivo
  - Não suporta falhas parciais

Execução

- Durante a fase de execução os participantes cooperam na realização de uma tarefa
  - Utilizando invocações remotas
  - Utilizando primitivas de passagem de mensagens
- O coordenador é avisado sempre que aparece um novo participante
- Quando terminada a tarefa, o coordenador é avisado para dar inicio ao protocolo de acordo em duas fases

Execução

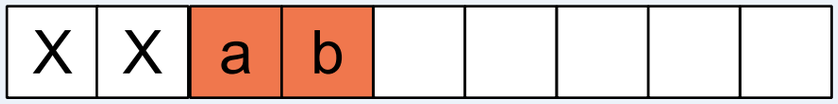
- Durante a execução, o coordenador regista no log todos os participantes:



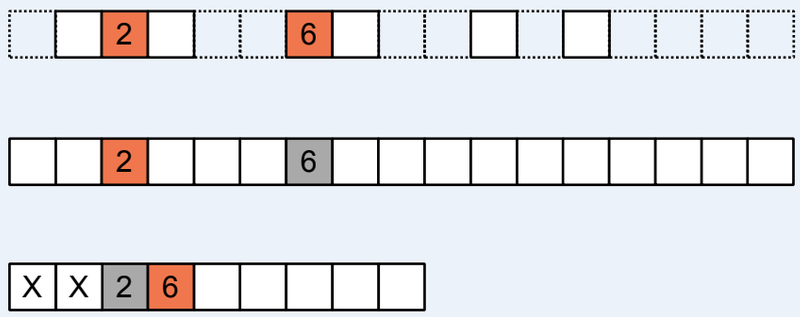
Execução

- Durante a execução, cada participante usou o seu método preferido de log:

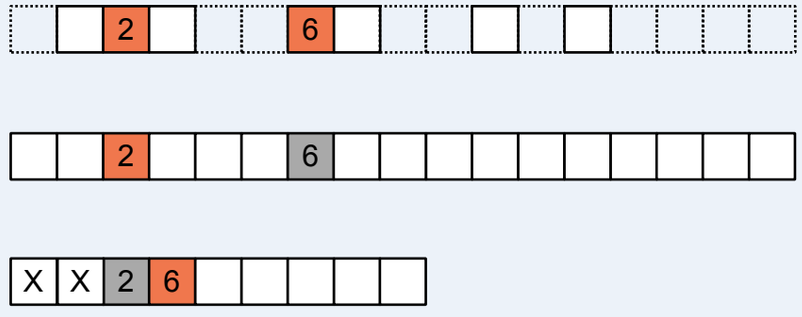
Log do Coordenador:



Participante a:

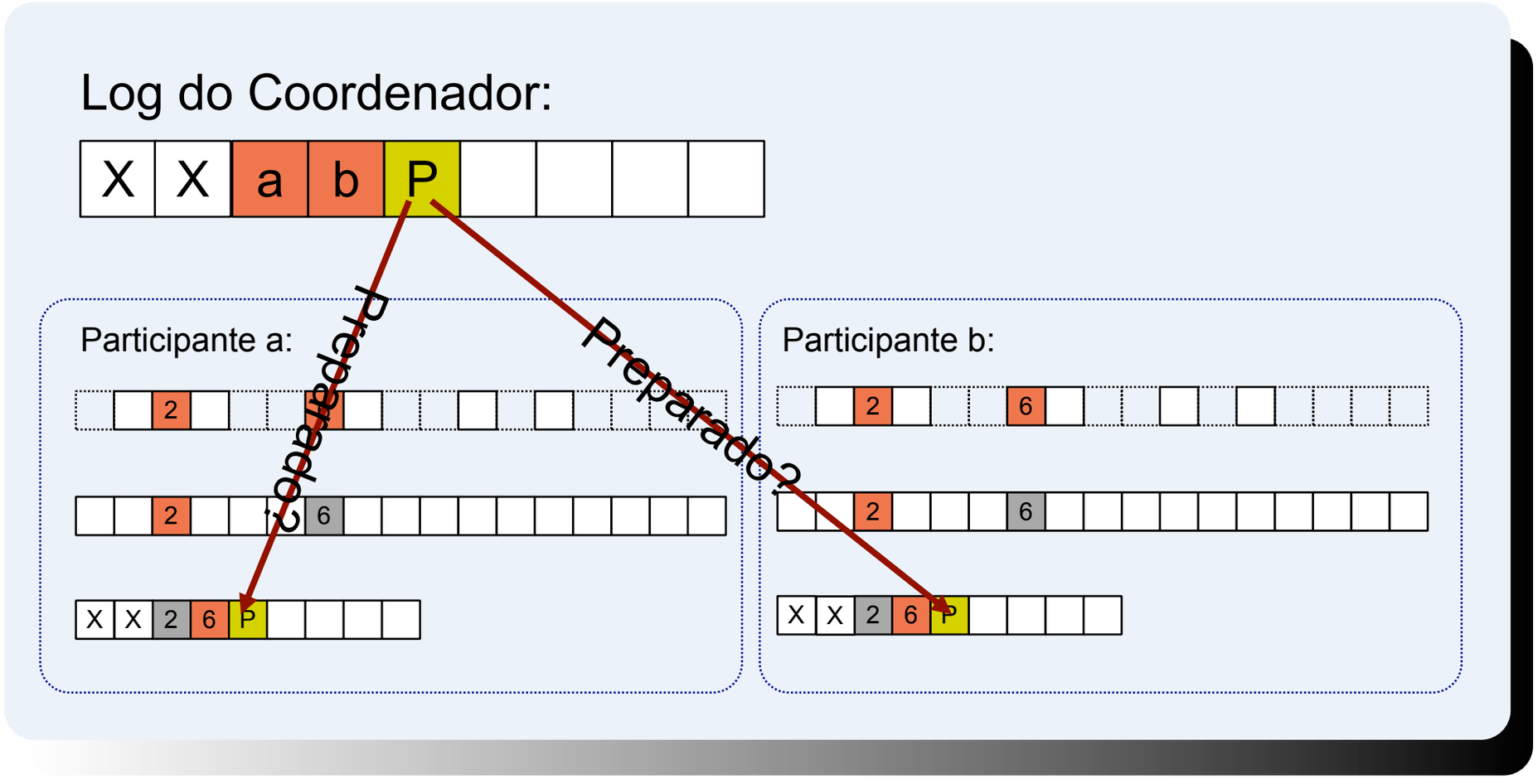


Participante b:



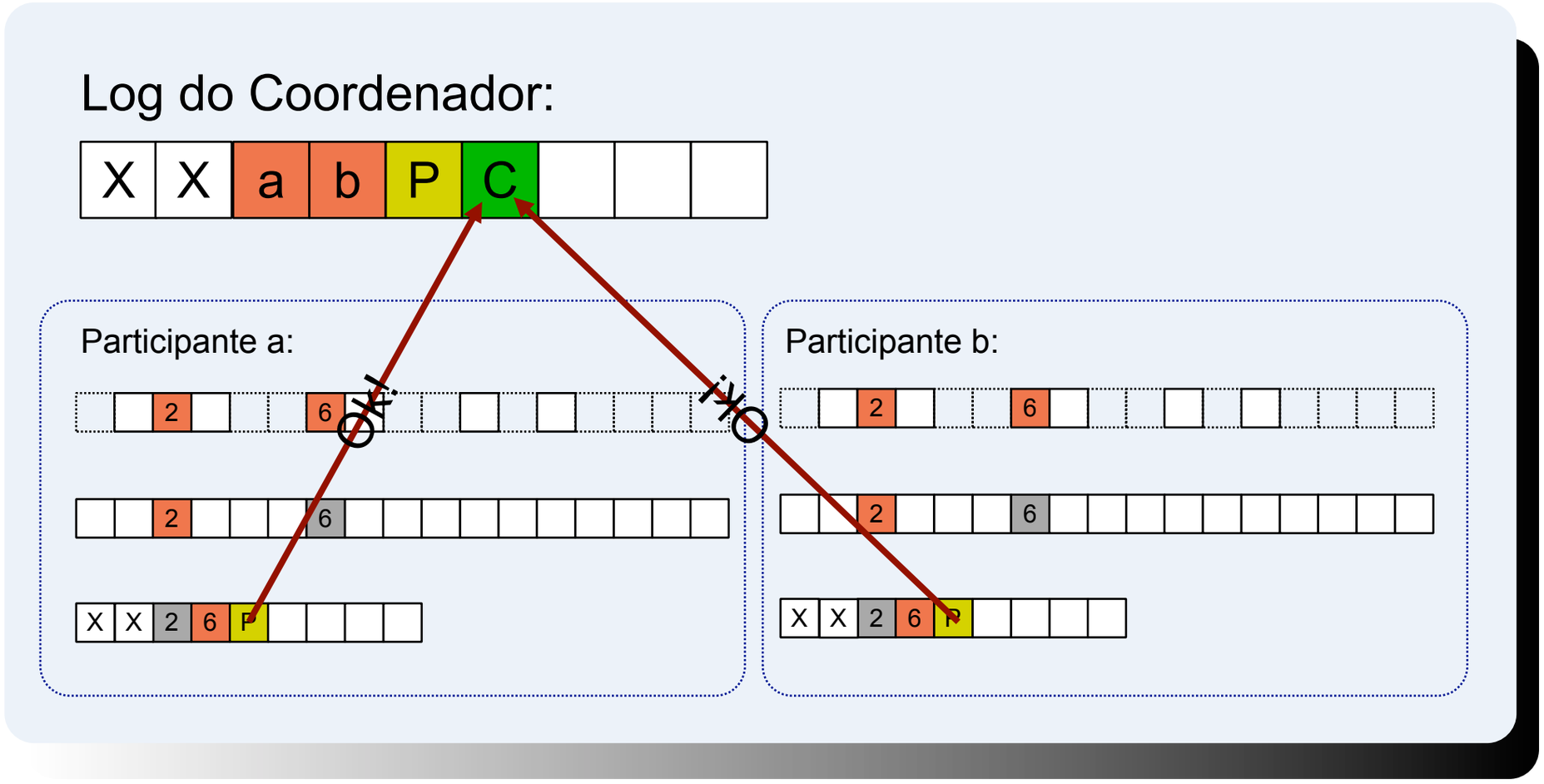
2PC: Fase 1

- O coordenador marca o seu log e avisa cada um dos participantes para se preparar:



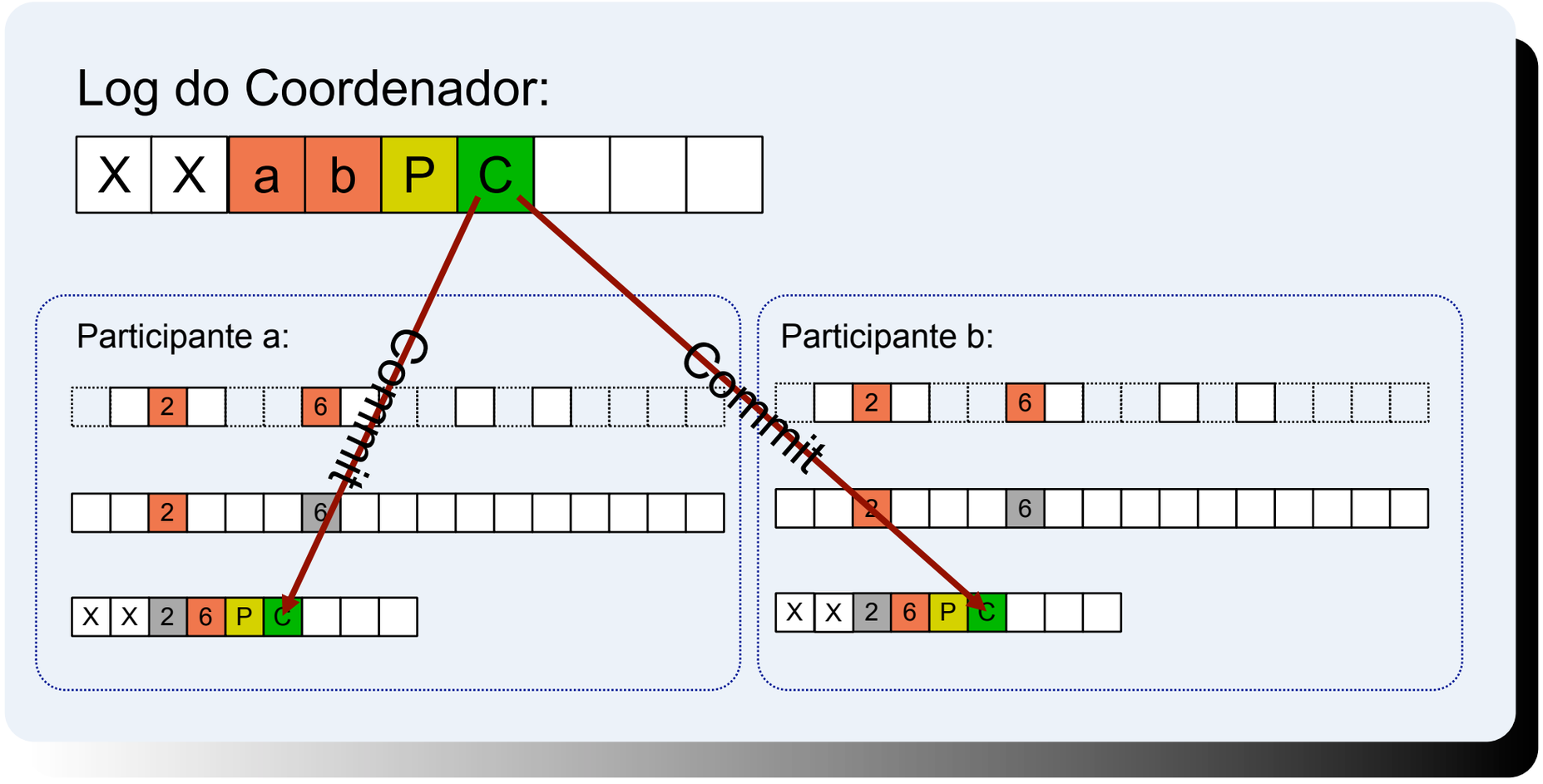
2PC: Fase 1 Completa

- Se todos os participantes concordam, o coordenador marca como confirmada



2PC: Fase 2

- Os participantes são informados da decisão e registam-na nos seus logs:

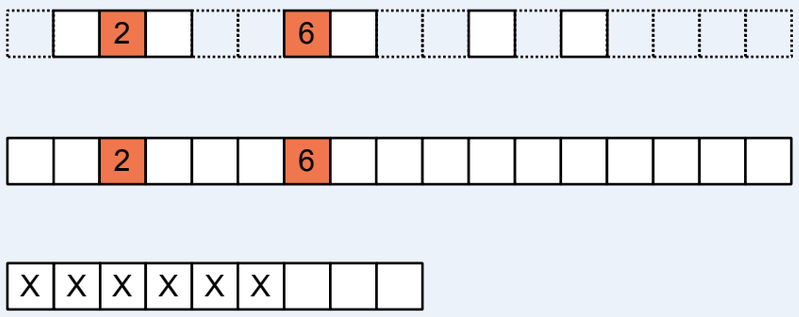


2PC Completo

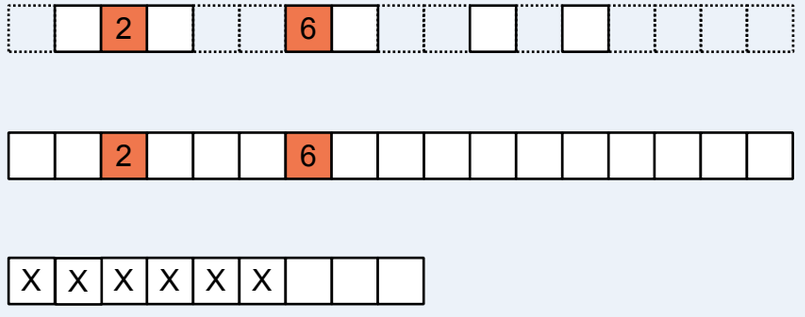
- Os logs dos participantes podem então ser esvaziados:

Log do Coordenador:

Participante a:

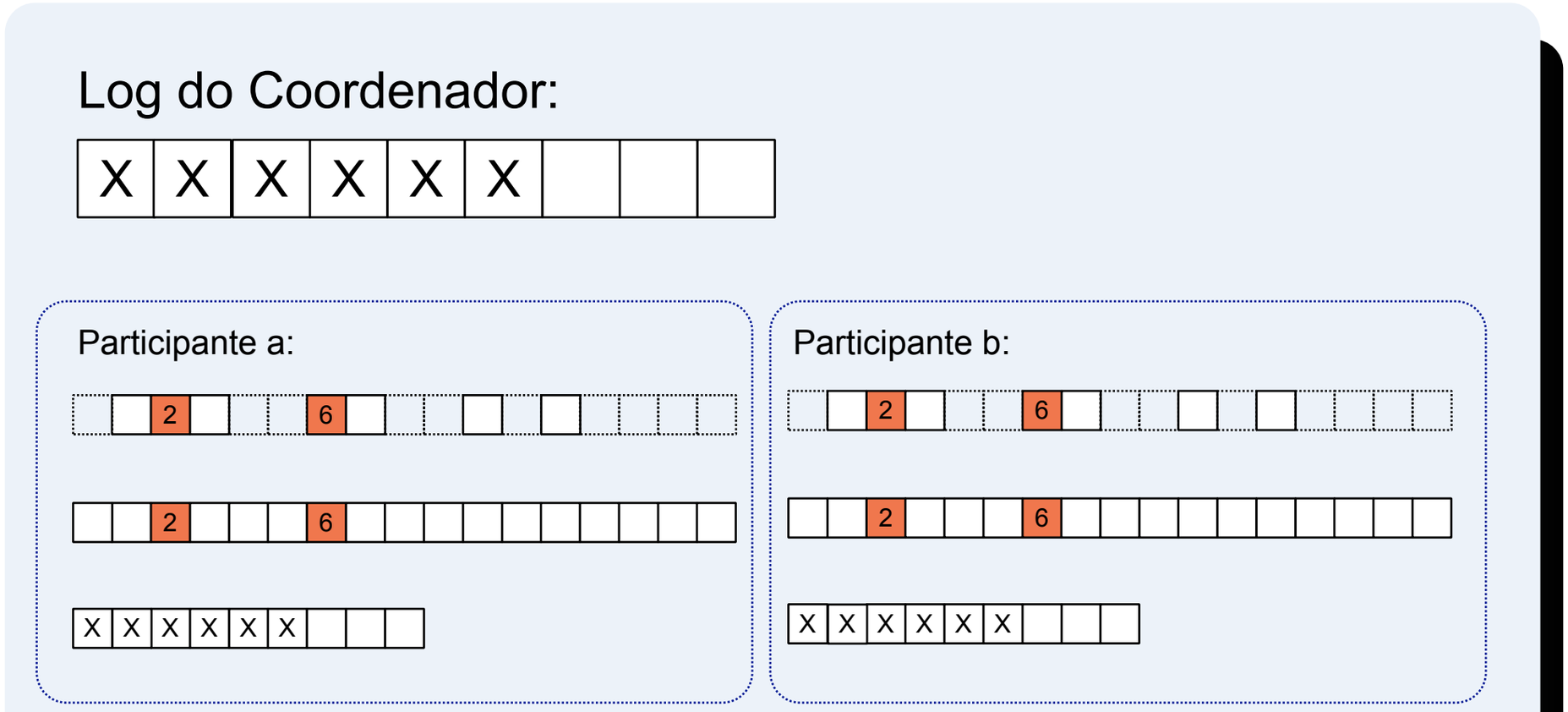


Participante b:



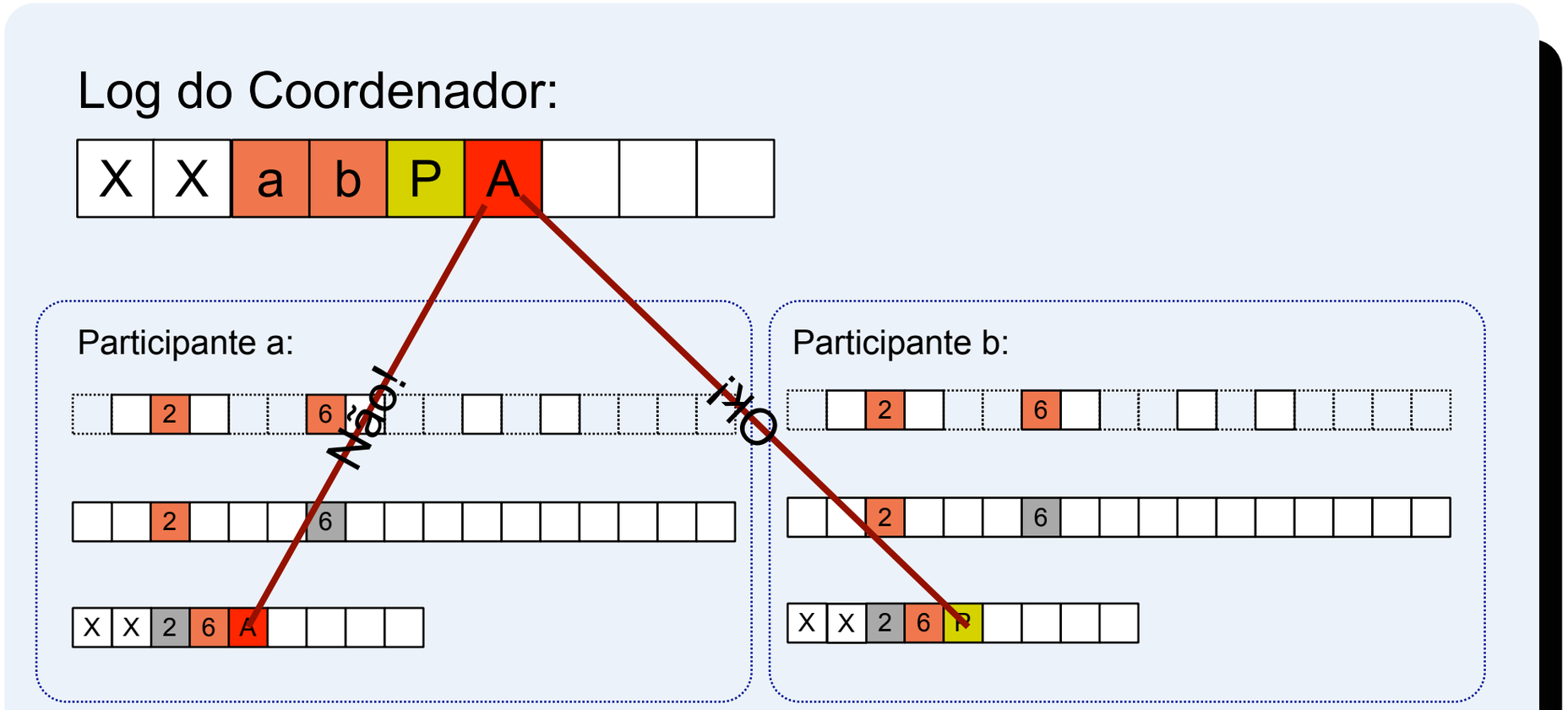
2PC Completo

- E o log do coordenador pode então ser esvaziado



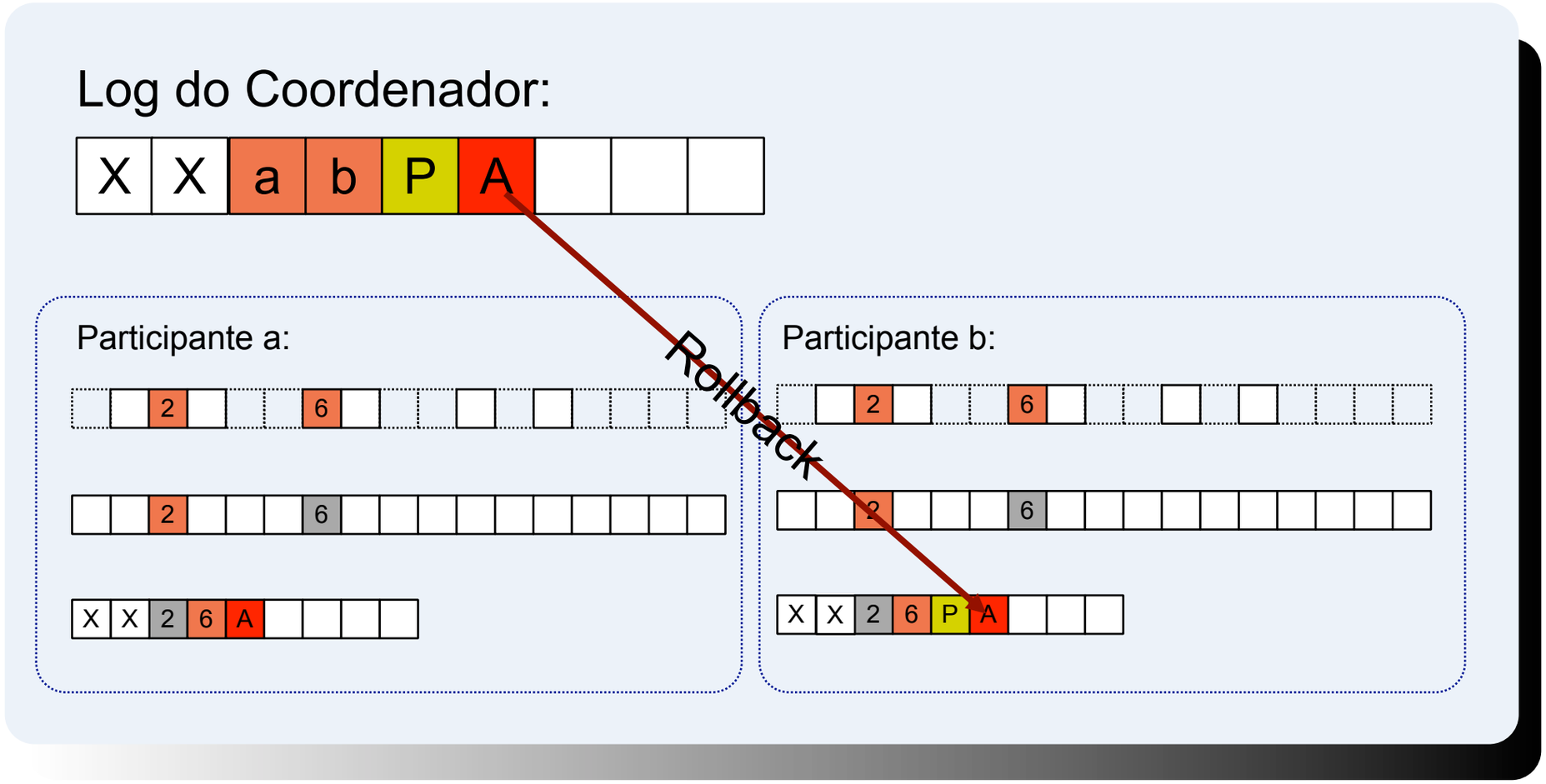
2PC: Fase 1 Abortada

- Basta que um participante não confirme, para toda a transacção ser desfeita:



2PC: Fase 2 Abortada

- Todos os participantes são então avisados:

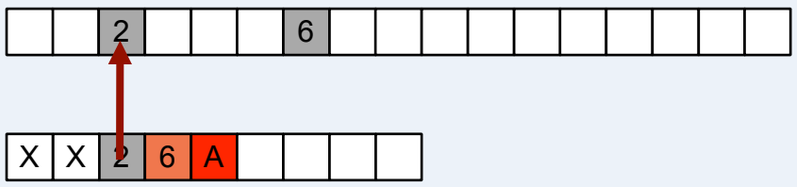


2PC: Fase 2 Abortada

- Cada participante recupera então o estado anterior:

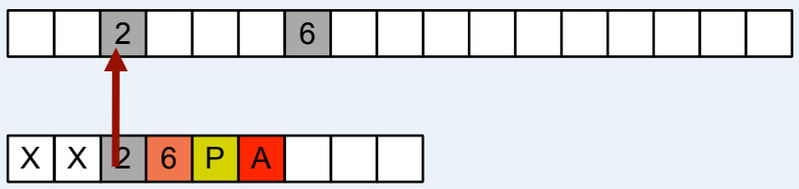
Log do Coordenador:

Participante a:



Participante b:

A cópia em memória é eliminada ou refrescada.

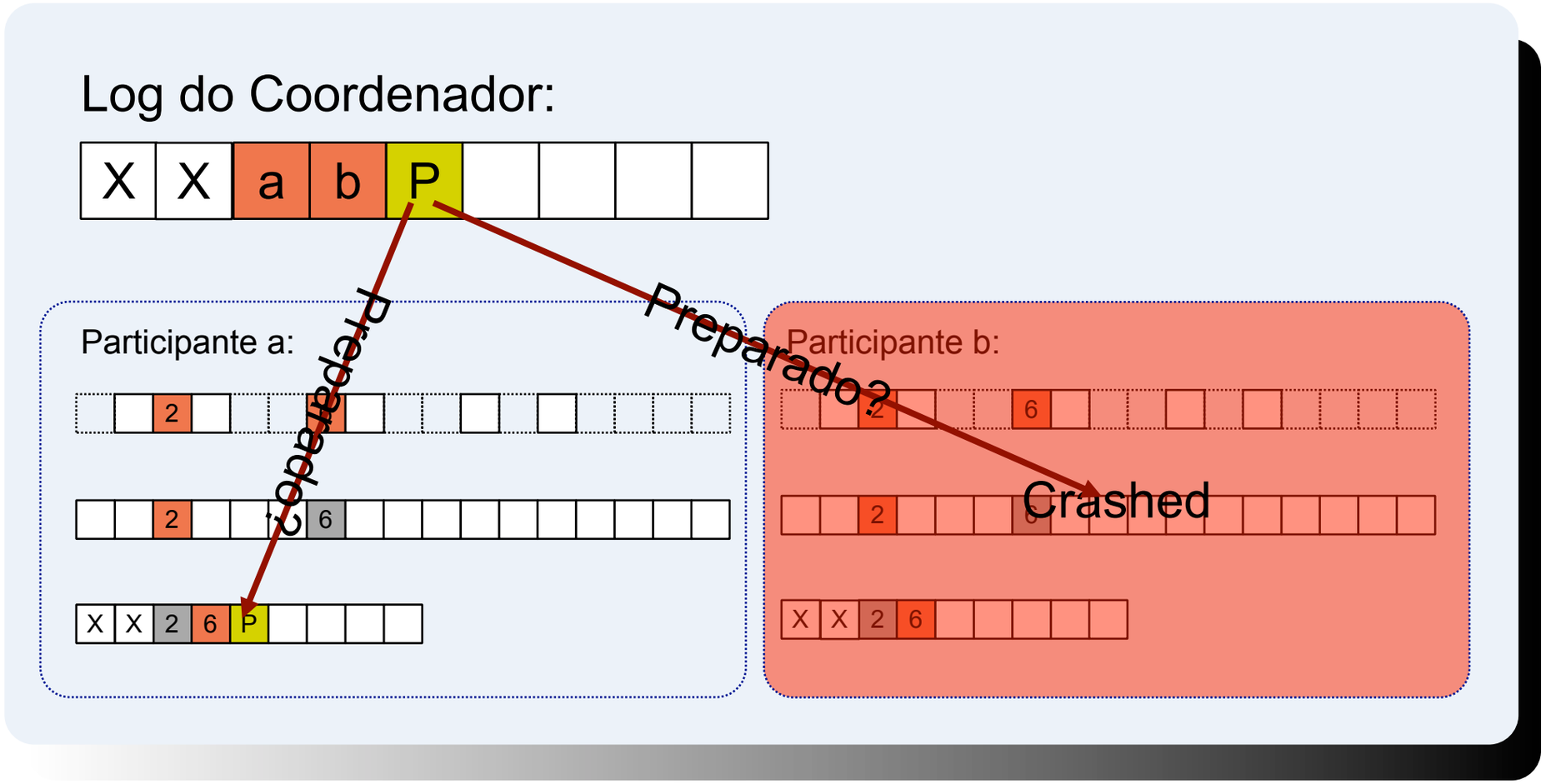


2PC: Recuperação

- Reinicio do coordenador:
  - Durante a execução: Abortar
  - Antes da decisão: Reinicio da fase 1
  - Decidida: Reinicio da fase 2
- Reinicio de um participante:
  - Ainda não votou: Aborta localmente e como consequência, toda a transacção
  - Já votou: Espera pela retransmissão da fase 2 pelo coordenador

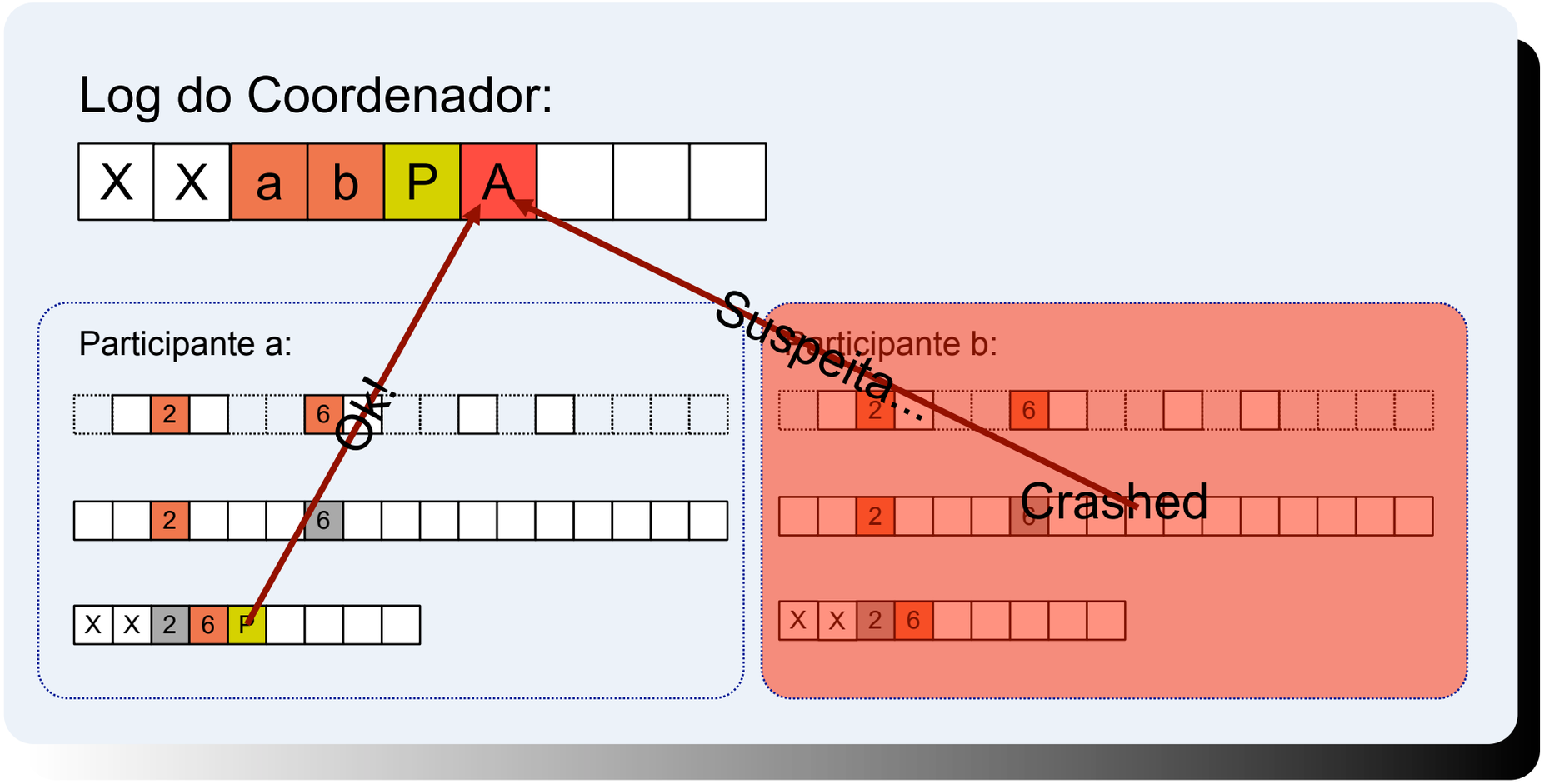
2PC: Falha Durante a Fase 1

- A fase 1 é ignorada por participantes que falharam:

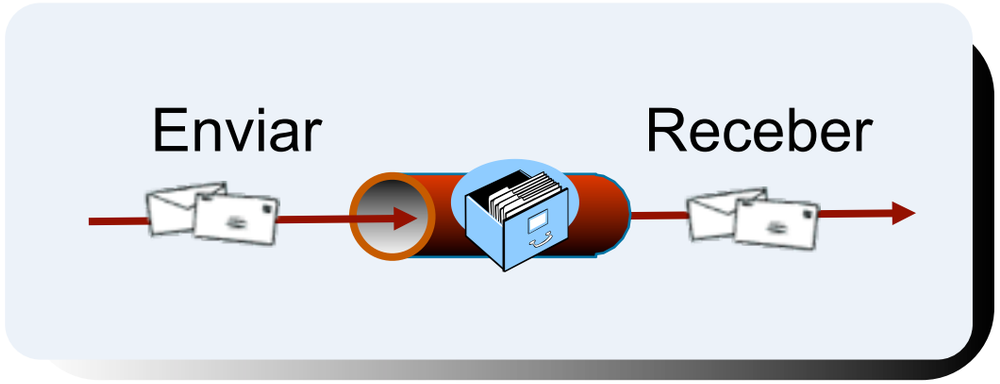


2PC: Falha Durante a Fase 1

- Quando o coordenador suspeita que um participante falhou, considera-o falhado:



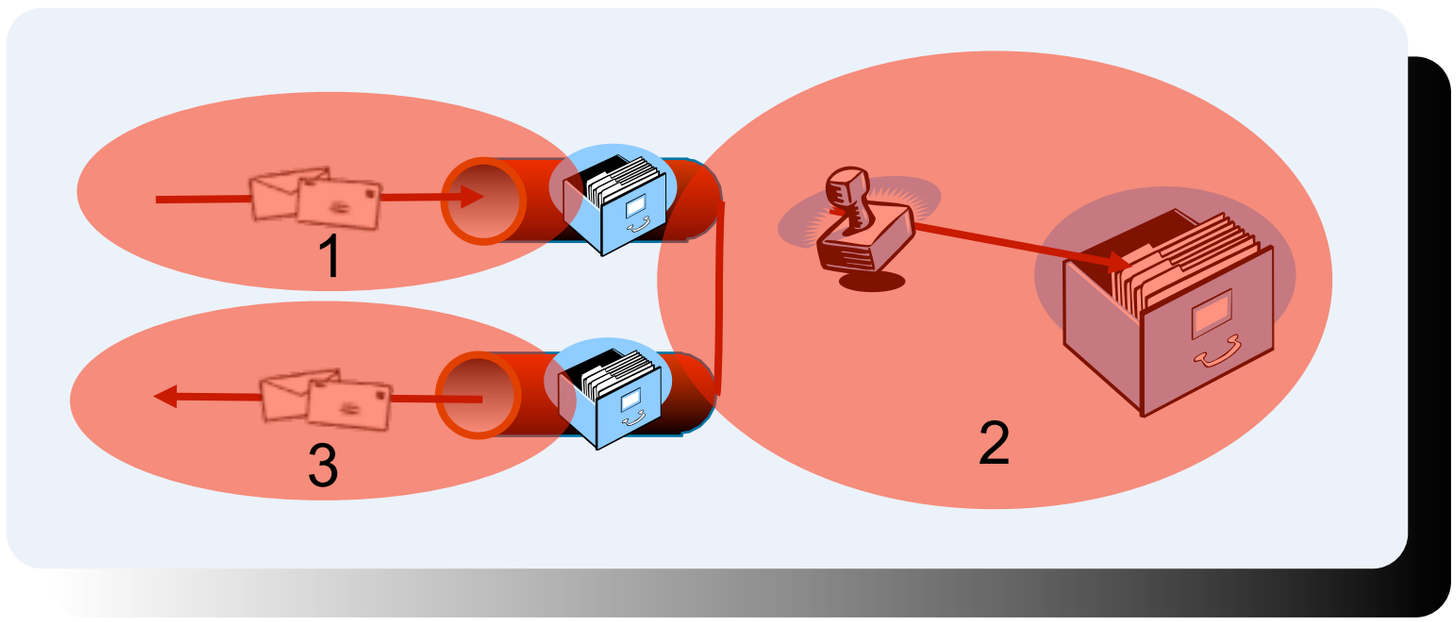
Comunicação Transaccional



- Canal de comunicação com armazenamento persistente das mensagens:
  - Enviar == INSERT INTO ... ;
  - Receber == SELECT ... ; DELETE FROM ... ;
- Operações de envio e recepção podem participar em transacções distribuídas

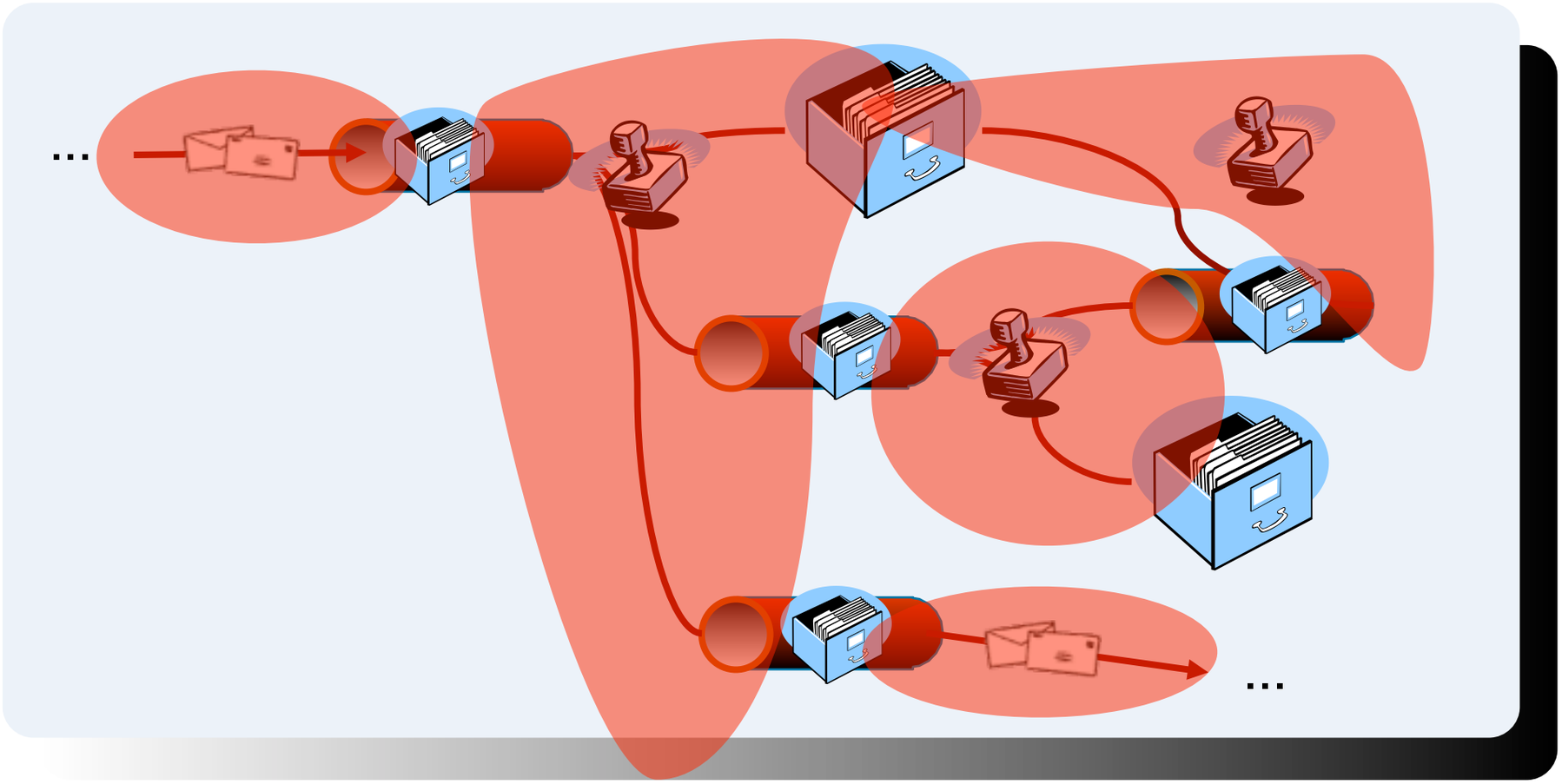
Store & Forward

- Padrão de pedido/processamento/ resposta:
  - Separa interacção com clientes e processamento em diferentes transacções
  - Assegura que é dado seguimento ao pedido



Store & Forward

- Funciona em procedimentos complexos:



Conclusões

- Transacções como método de tolerância a faltas
- Transacções distribuídas para tolerância de faltas parciais em sistemas distribuídos
- Comunicação transaccional suporta sistemas de “workflow”