Recuperação após Falha

Banco de Dados II

Cap. 18 (Ramakrishnan)
Cap. 23 (Elmasri)
Cap. 17 (Silberschatz)

Introdução

Tipos de falhas:

- Transação
- Sistema
- Meio de armazenamento
- Problemas físicos e catástrofes

Motivação

```
create table dados (cpf varchar(10), nome varchar(50));

#acha o local dos dados
1. select pg_relation_filepath('dados');

#mostra o dataset
2. sudo hexdump -C /var/lib/postgresql/14/main/*

#insere dados
3. insert into dados  values (‘aaa’, ‘bbb’)

#mostra o dataset novamente
4. sudo hexdump -C /var/lib/postgresql/14/main/*
```

Técnicas Baseadas em *Log*

Técnicas mais comuns de *recovery*

Utilizam um arquivo de *Log*

- registra sequencialmente as atualizações feitas por transações no BD
- write-ahead logging (WAL)
- tipos de *log*
  - *log* de UNDO
    - mantém apenas o valor antigo do dado (*before image*)
  - *log* de REDO
    - mantém apenas o valor atualizado do dado (*after image*)
  - *log* de UNDO/REDO
    - mantém os valores antigo e atualizado do dado

Notação no Registro de *Log*

Start t : indica o início de uma transação T;

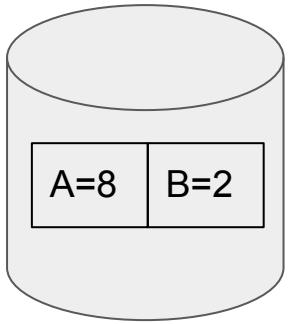
Commit t: Indica o fim de uma transação e todas as alterações feitas por *t devem ser escritas no disco (não é possível garantir quando isso será feito);*

Abort t: transação incompleta, reverter todas as ações já feitas

Transações em memória

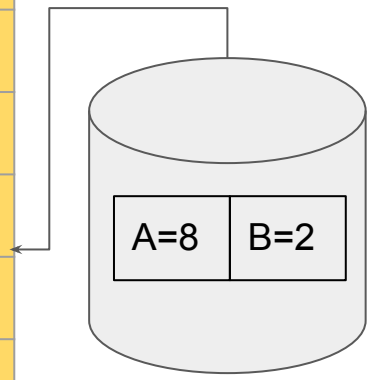
Buffer Pool

Passo	Ação	t	M-A	M-B
1	Read (A,t)	8	8	
2				
3				
4				
5				
6				



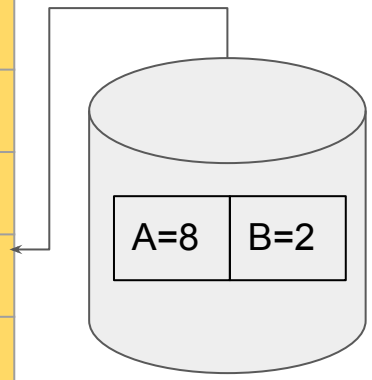
Transações em memória

Passo	Ação	t	M-A	M-B
1	Read (A,t)	8	8	
2	t= t*2	16	8	
3	write(A,t)	16	16	
4				
5				
6				



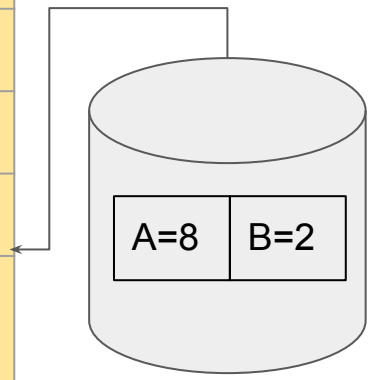
Transações em memória

Passo	Ação	t	M-A	M-B
1	Read (A,t)	8	8	
2	t= t*2	16	8	
3	write(A,t)	16	16	
4	Read(B,t)	2	16	2
5	t= t*2	4	16	2
6	write(B,t)	4	16	4



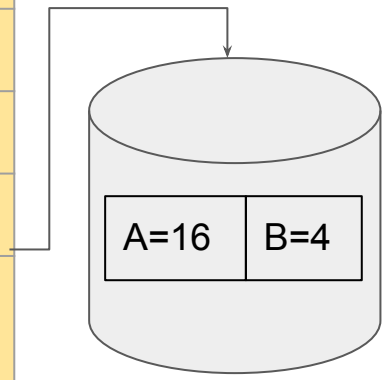
Transações em memória

Passo	Ação	t	M-A	M-B
1	Read (A,t)	8	8	
2	t= t*2	16	8	
3	write(A,t)	16	16	
4	Read(B,t)	2	16	2
5	t= t*2	4	16	2
6	write(B,t)	4	16	4



Transações em memória

Passo	Ação	t	M-A	M-B
1	Read (A,t)	8	8	
2	t= t*2	16	8	
3	write(A,t)	16	16	
4	Read(B,t)	2	16	2
5	t= t*2	4	16	2
6	write(B,t)	4	16	4
7	output(A)	4	16	4
8	output(B)	4	16	4



Redo Logging

- Ignora ações feitas parcialmente e repete alterações feitas por transações com "*commit*";

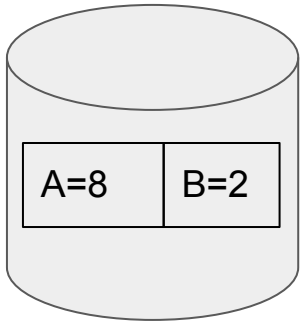
-Regra REDO:

Se uma transação modificar X, então uma entrada no log no formato <T, X,v_new> deve ser escrita no disco antes de X ser escrito. Incluindo Update e *commits*;

Conhecido como escrita adiada

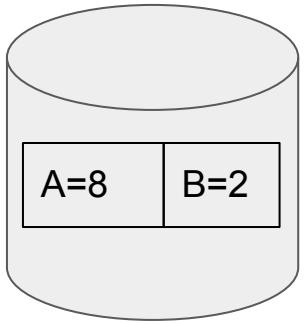
Redo- Escrita atrasada

Passo	Ação	t	M-A	M-B	Arquivo - LOG
					<start T>
1	Read (A,t)	8	8		
2	t= t*2	16	8		
3	write(A,t)	16	16		<T, A, 16>
4	Read(B,t)	2	16	2	
5	t= t*2	4	16	2	
6	write(B,t)	4	16	4	<T, B, 4>



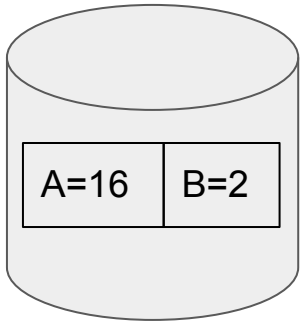
Redo- Escrita atrasada

Passo	Ação	t	M-A	M-B	Arquivo - LOG
					<start T>
1	Read (A,t)	8	8		
2	t= t*2	16	8		
3	write(A,t)	16	16		<T, A, 16>
4	Read(B,t)	2	16	2	
5	t= t*2	4	16	2	
6	write(B,t)	4	16	4	<T, B, 4>
	Commit				Commit (T)
Flush log					



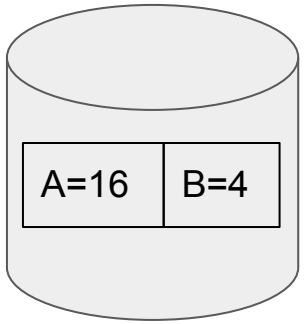
Redo- Escrita atrasada

Passo	Ação	t	M-A	M-B	Arquivo - LOG
					<start T>
1	Read (A,t)	8	8		
2	t= t*2	16	8		
3	write(A,t)	16	16		<T, A, 16>
4	Read(B,t)	2	16	2	
5	t= t*2	4	16	2	
6	write(B,t)	4	16	4	<T, B, 4>
	Commit				Commit (T)
Flush log					
10	output(A)	4	16	4	



Redo- Escrita atrasada

Passo	Ação	t	M-A	M-B	Arquivo - LOG
					<start T>
1	Read (A,t)	8	8		
2	t= t*2	16	8		
3	write(A,t)	16	16		<T, A, 16>
4	Read(B,t)	2	16	2	
5	t= t*2	4	16	2	
6	write(B,t)	4	16	4	<T, B, 4>
	Commit				Commit (T)
Flush log					
10	output(A)	4	16	4	
30	output(B)	4	16	4	



Regras Redo

Verificar o log

- ***T sem o <Commit T>***
  - Pode ser ignorado
  - T não escreveu nada no disco

- ***T com o <Commit T>***
  - Redo suas ações (Inicia no *<Start T>*)

Recovery ReDO

- Percorrer o arquivo das operações mais **antigas** para as mais **novas** (início do log até o fim);
- Identificar se o **commit** foi salvo no disco:
  - Se **sim**, temos que refazer a operação da transação;
  - Se **não**, ignorar a transação;

- Para cada operação não completa:
  escrever <abort T>

Exemplo

- Segundo as operações abaixo:
  1) <start t>;
  2) <T,A,10>;
  3) <Start U>;
  4) <U,B,20>;
  5) <T,C,30>;
  6) <U,D,40>;
  7) <Commit U>;
  8) <T,E,50>;
  9) <Commit T>

Valor atual na memória
- A=5
- B=10
- C=20
- D=30
- E=40

Após o Redo
- A=
- B=
- C=
- D=
- E=

Se a última operação a ser escrita no log foi imediatamente após uma das seguintes linhas. **Quais são as ações do recover REDO?**

A) 3
B) 7
C) 8
D) 9

Atividade 1

<start T1>
<write T1,A,10>
<start T2>
<write T2,C,55>
<commit T2>
<start T3>
<write T3,B,20>
<commit T1>
<start T4>
<write T4,C,65>
<start T5>

1 Quais transações executam o Redo?2 Qual é o valor final nas variáveis?

Valor antigo de

- A=10
- B=15
- C=45

Valor novo de

- A=
- B=
- C=

Atividade 2

<start T1>
<write T1,A,10>
<start T2>
<write T2,C,45>
<write T2,E,17>
<commit T2>
<write T1,C,55>
<start T3>
<write T3,B,20>
<commit T1>
<start T4>
<write T4,C,65>
<start T5>
….

...
<write T5,D,39>
<start T6>
<write T3,A,25>
<write T6,F,2>
<write T3,E,28>
<commit T3>
<start T7>
<write T7,B,30>
<commit T7>
<write T4,E,34>
Crash!

1 Quais transações executam o Redo? 2 Qual é o valor final nas variáveis?

Valor antigo de
- A=10
- B=15
- C=55
- D=30
- E=17
- F=1

Valor novo de
- A=
- B=
- C=
- D=
- E=
- F=

Checkpoint REDO

<start T_1>
<T_1,A,5>
<start T_2>
<Commit T_1>
<T_2,B,10>
<CKPT (T2)>
<T_2,C,15>
<start T_3>
<T_3,D,20>
<END CKPT>
<Commit T_3>
<Commit T_2>

...

- Força que as transações com "commit" antes do <CKPT> tenham suas operações salvas no disco

Atividade 3

<start T1>
<write T1,A,10>
<start T2>
<write T2,C,45>
<write T2,E,17>
<commit T2>
<write T1,C,55>
<CKPT (T1)>
<start T3>
<write T3,B,20>
<commit T1>
<start T4>
<END CKPT>
<write T4,C,65>
<start T5>
….

...
<write T5,D,39>
<start T6>+
<write T3,A,25>
<write T6,F,2>
<write T3,E,28>
<commit T3>
<start T7>
<write T7,B,30>
<commit T7>
<write T4,E,34>
Crash!

1 Quais transações executam o Redo?
2 Qual é o valor final nas variáveis?

Valor antigo de

- A=10
- B=15
- C=55
- D=30
- E=17
- F=1

Valor novo de

Atividade 4

<start T1>

<T1,1, A,30>

<start T2>

<commit T1>

< CKPT (T2)>

<T2,1, A,50>

<start T3>

<END CKPT>

<T2,2, A,100>

<CKPT (T2,T3)>

<start T4>

<T4,1, A,200>

<commit T4>

Quais variáveis serão atualizadas
e para qual valor?

Atividade 5

A=25 | B=30 | C=90 | D=40 |

E=28 | F=1 | G=10 | H=10

<start T1>

<T1,A,20>

<start T2>

<T2,C,45>

<T2,E,77>

<commit T2>

<T1,C,70>

<start T3>

<T3,B,15>

<commit T1>

<start T4>

<T4,C,90>

<start T5>

<T5,D,65>

<CKPT (T5,T4,T3)>

<commit T4>

<T5,D,40>

<start T6>

<T3,A,25>

<T6,F,2>

<T3,E,28>

<commit T3>

<T6,A,32>

<END CKPT>

<commit T5>

<start T7>

<T7,B,30>

<CKPT (T7, T6)>

<commit T6>

<start T8>

<T8,G,30>

<start T9>

<T9,H,30>

Qual é o valor final das variáveis aplicando REDO?
Redo:

Atividade 6

A=20 | B=30 | C=70 | D=50 |

E=28 | F=1 | G=30 | H=10

<start T1>

<T1,A,20>

<start T2>

<T2,C,45>

<T2,E,77>

<commit T2>

<T1,C,70>

<start T3>

<T3,B,15>

<commit T1>

<start T4>

<T4,C,90>

<start T5>

<T5,D,65>

<CKPT (T5,T4,T3)>

<commit T4>

<T5,D,40>

<start T6>

<T3,A,25>

<T6,F,2>

<T3,E,28>

<commit T3>

<T6,A,32>

<commit T5>

<start T6>

<T6,B,30>

<commit T6>

<END CKPT>

<start T7>

<T7,G,30>

<Commit T7>

Qual variável teve o valor alterado pela execução do REDO?

Atividade 7

A=20 | B=30 | C=70 | D=50 |

E=28 | F=1 | G=30 | H=10

<start T1>

<T1,A,20>

<start T2>

<T2,C,45>

<T2,E,77>

<commit T2>

<T1,C,70>

<start T3>

<T3,B,15>

<commit T1>

<start T4>

<T4,C,90>

<start T5>

<T5,D,65>

<CKPT (T5,T4,T3)>

<commit T4>

<T5,D,40>

<start T6>

<T3,A,25>

<T6,F,2>

<T3,E,28>

<commit T3>

<T6,A,32>

<commit T5>

<start T7>

<T7,B,30>

<commit T7>

<start T8>

<T8,G,30>

<Commit T8>

<END CKPT>

A=25
B=30
C=90
D=40
E=28
G=30

Qual é o valor final das variáveis aplicando REDO? E qual transação sofreu o REDO?

Undo Logging

- Faz reparos no estado do BD desfazendo ações de transações;

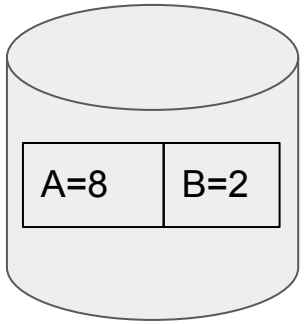
-Regras UNDO:

U1-Se uma transação modificar X, então uma entrada no log no formato <T, X,v_old> deve ser escrita no disco antes de X ser escrito no disco;

U2- Se uma transação faz um commit então um <commit > deve ser salvo no log depois das operações serem escritas no disco;

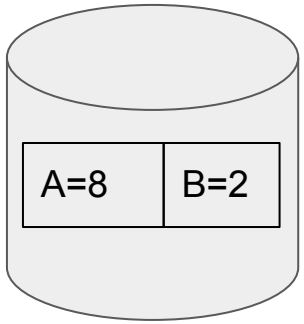
UNDO- Escrita imediata

Passo	Ação	t	M-A	M-B	Arquivo - LOG
					<start t>
1	Read (A,t)	8	8		
2	t= t*2	16	8		
3	write(A,t)	16	16		<T, A, 8>



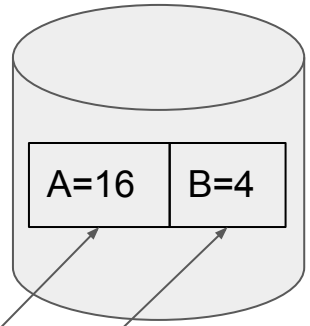
UNDO- Escrita imediata

Passo	Ação	t	M-A	M-B	Arquivo - LOG
					<start t>
1	Read (A,t)	8	8		
2	t= t*2	16	8		
3	write(A,t)	16	16		<T, A, 8>
4	Read(B,t)	2	16	2	
5	t= t*2	4	16	2	
6	write(B,t)	4	16	4	<T, B, 2>
Flush log					



UNDO- Escrita imediata

Passo	Ação	t	M-A	M-B	Arquivo - LOG
					<start t>
1	Read (A,t)	8	8		
2	t= t*2	16	8		
3	write(A,t)	16	16		<T, A, 8>
4	Read(B,t)	2	16	2	
5	t= t*2	4	16	2	
6	write(B,t)	4	16	4	<T, B, 2>
Flush log					
7	output(A)	4	16	4	
8	output(B)	4	16	4	
					Commit
Flush log					



Recovery UNDO

- Percorrer o log da última linha salva para a primeira (mais nova para mais velha)
- Identificar se o commit foi salvo no disco:
  - Se sim, transação OK;
  - Se não, desfazer transação percorrendo o arquivo de log da operação mais recente até a mais antiga;
- Escrever <Abort t> para operações abortadas;

Atividade 1

- Segundo as operações abaixo:
  1) <start t>;
  2) <T,A,10>;
  3) <Start U>;
  4) <U,B,20>;
  5) <T,C,30>;
  6) <U,D,40>;
  7) <Commit U>;
  8) <T,E,50>;
  9) <Commit T>

Se a última operação a ser escrita no log foi imediatamente uma das seguintes.
Quais são as ações do recover UNDO?

A) 3
B) 7
C) 8
D) 9

Checkpoint UNDO

<start T_1>
<T_1,A,5>
<start T_2>
<T_2,B,10>
<CKPT (T1,T2)>
<T_2,C,15>
<T_1,D,20>
<start T_3>
<Commit T_1>
<Commit T_2>
<END CKPT>
...

- Força que as transações com "commit" antes do <END CKPT> tenham suas operações salvas no disco

Checkpoint UNDO

<start T_1>
<T_1,A,5>
<start T_2>
<T_2,B,10>
<Commit T_1>

<CKPT (T2)>
<T_2,C,15>
<T_1,D,20>
<start T_3>
<Commit T_2>
<END CKPT>
...

- Força que as transações com "commit" antes do <END CKPT> tenham suas operações salvas no disco

Atividade 2

- Segundo as operações abaixo:
  1) <start t>;
  2) <T,A,10>;
  3) <Start U>;
  4) <U,B,20>;
  5) <T,C,30>;
  6) <T,D,40>;
  7) <Commit T>;
  8) <U,E,50>;
  9) <Commit U>

Suponha que uma operação de checkpoint não bloqueante tenha iniciado imediatamente depois da operação abaixo. Quando o CKPT END pode ser escrito?

A) 2
B) 3
C) 6

Atividade 3

- Segundo as operações abaixo:
  1) <start t>;
  2) <T,A,10>;
  3) <Start U>;
  4) <START CKPT(t, u)>
  5) <U,B,20>;
  6) <T,C,30>;
  7) <Commit T>;
  8) <Start X>;
  9) <Commit U>
  10) <END CKPT>
  11)

Quais os valores das variáveis no caso de bug após a linha: (possível valor novo do A=100, B=200,C=300)

A) 10

B) 6

C) 5

Atividade 4

1-Segundo as operações abaixo usando log Undo:

1) <start T1>
2) <start T2>
3) <write T1,X,1>
4) <start T3>
5) <write T2,X,7>
6) <commit T1>
7) <start T4>
8) <write T3,Y,10>
9) <write T4,Y,100>
10) <commit T3>
11) <write T2,Y,55>

Quais os valores das variáveis no caso de bug após a linha 11:

Atividade 5

Segundo as operações abaixo:

1) <start T1>
2) <start T2>
3) <write T1,X,1>
4) <START CKPT(T1,T2)>
5) <write T2,X,7>
6) <commit T1>
7) <start T4>
8) <start T3>
9) <commit T4>
10) <write T2,Y,55>
11) <commit T2>
12) <END CKPT>
13) <write T3,Y,100>

Quais os valores das variáveis no caso de bug após a linha 13:

UNDO- Qual a desvantagem?

Passo	Ação	t	M-A	M-B	D-A	D-B	Arquivo - LOG
							<start t>
1	Read (A,t)	8	8		8	2	
2	t= t*2	16	8		8	2	
3	write(A,t)	16	16		8	2	<T, A, 8>
4	Read(B,t)	2	16	2	8	2	
5	t= t*2	4	16	2	8	2	
6	write(B,t)	4	16	4	8	2	<T, B, 2>
Flush log							
7	output(A)	4	16	4	16	2	
8	output(B)	4	16	4	16	4	
							Commit
Flush log							

Leitura

- Chapter 18 Crash Recovery (Database Management Systems - third edition – Ramakrishnan & Gerhke)
  - Verificar o capítulo equivalente na versão em português.

Técnicas de Gerência de Buffer

STEAL: a alteração de uma transação pode ser descarregada do buffer pool a qualquer momento (antes do commit). Ou seja, uma transação pode “roubar” espaço do buffer pool de outra transação.

> **vantagem**: não há necessidade de manter blocos bloqueados por transações

NO STEAL: todas as alterações de transações permanecem no buffer pool até commit.

> **vantagem**: processo de recovery mais simples - evita dados de transações inacabadas sendo gravadas no BD

Gerenciamento do Buffer Pool

Force: escrever no disco as alterações a cada commit

- Baixo tempo de resposta.
- Garante a durabilidade

Steal: salvar dados não comitados, quando necessário.

- Se não, baixa performance
- Se sim, como garantir a atomicidade.

No Force e Steal

Manter um arquivo de log contendo alterações das transações

- **Assumir que o log estará armazenado em disco**
- **Log permite desfazer ou refazer ações;**

O SGBD deve escrever no arquivo de log antes de fazer qualquer alteração.

	No Steal	Steal
No Force		
Force		

Atividade prática

Abra um terminal no postgres e rode:

```
create extension if not exists pg_stat_statements;

select pg_walfile_name( pg_current_wal_lsn() ),

    pg_current_wal_lsn(),

    pg_size_pretty( pg_relation_size( 'TABLE NAME' ) );
```

Atividade prática

Abra um novo terminal e acesse como usuaŕio posgres

cd **/var/lib/postgresql/14/main/pg_wal**

/usr/lib/postgresql/14/bin/./pg_waldump ./000000010000000000000017 -f

O que acontece no caso de uma transação ser abortada?

Técnicas de Gerência de Buffer

FORCE: em cada commit, todas as páginas sujas são enviadas para o disco.

> **Vantagem**: garante a durabilidade de Tx o mais cedo possível – garante mais o REDO de Tx em caso de falha

No-Force: páginas modificadas podem continuar no buffer pool após o commit.

> **Vantagem**: blocos atualizados podem permanecer na cache e serem utilizados por outras transações, após o commit de Tx (reduz custo de acesso a disco)

Qual será a política mais implementada?

No Force e Steal

Manter um arquivo de log contendo alterações das transações

- **Assumir que o log estará armazenado em disco**
- **Log permite desfazer ou refazer ações;**

O SGBD deve escrever no arquivo de log antes de fazer qualquer alteração.

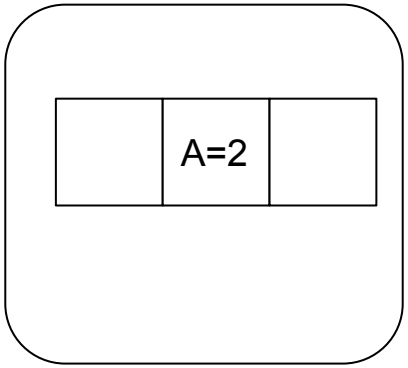
	No Steal	Steal
No Force		
Force		

Motivação

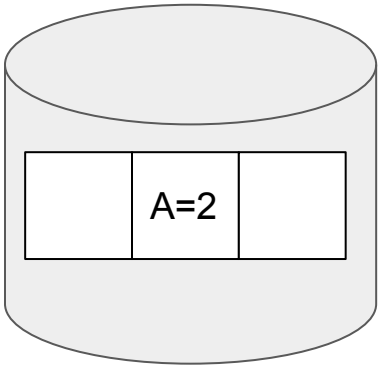
Schedule

T1
Begin
R(A)
W(A)
.
.
.
Commit()

Buffer Pool



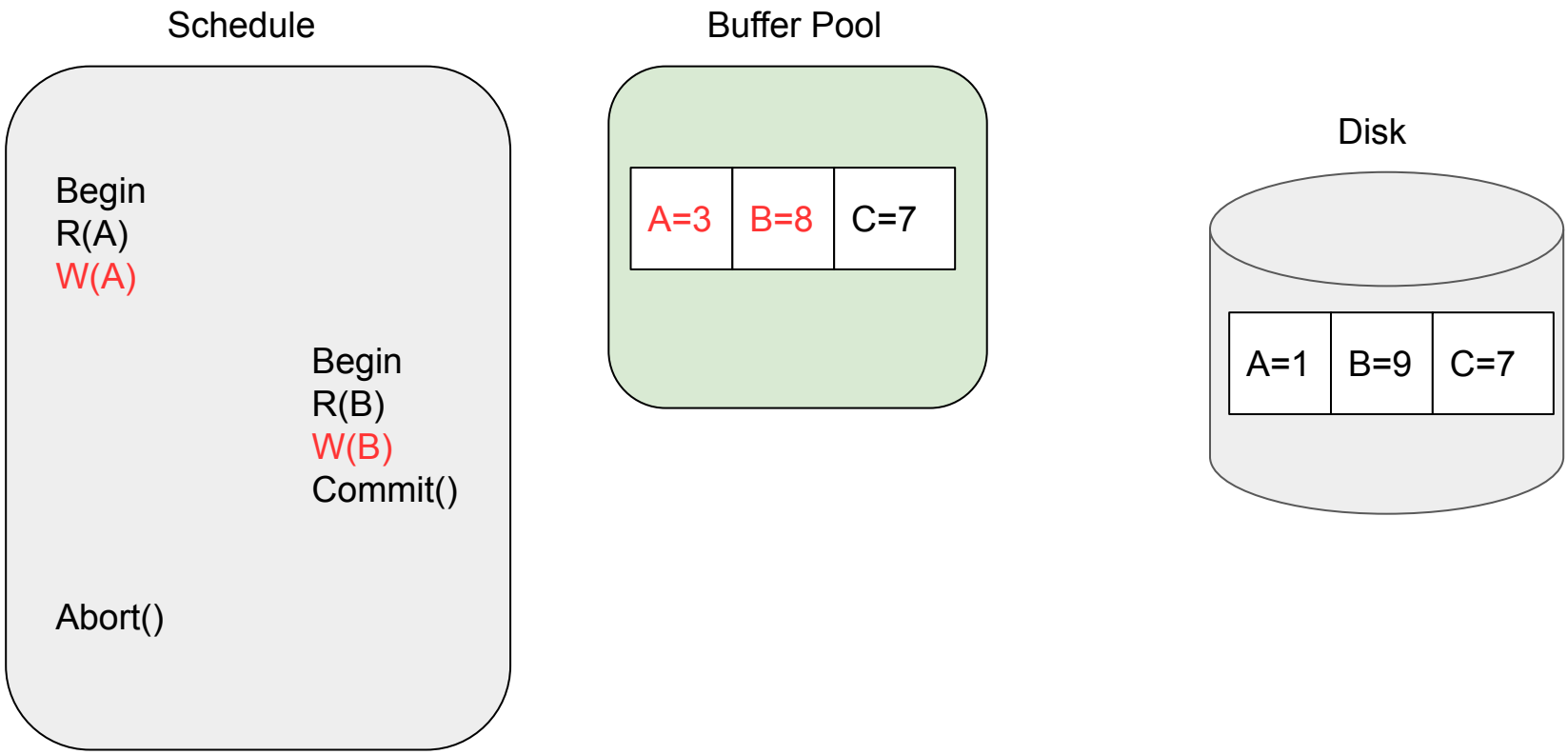
Disk



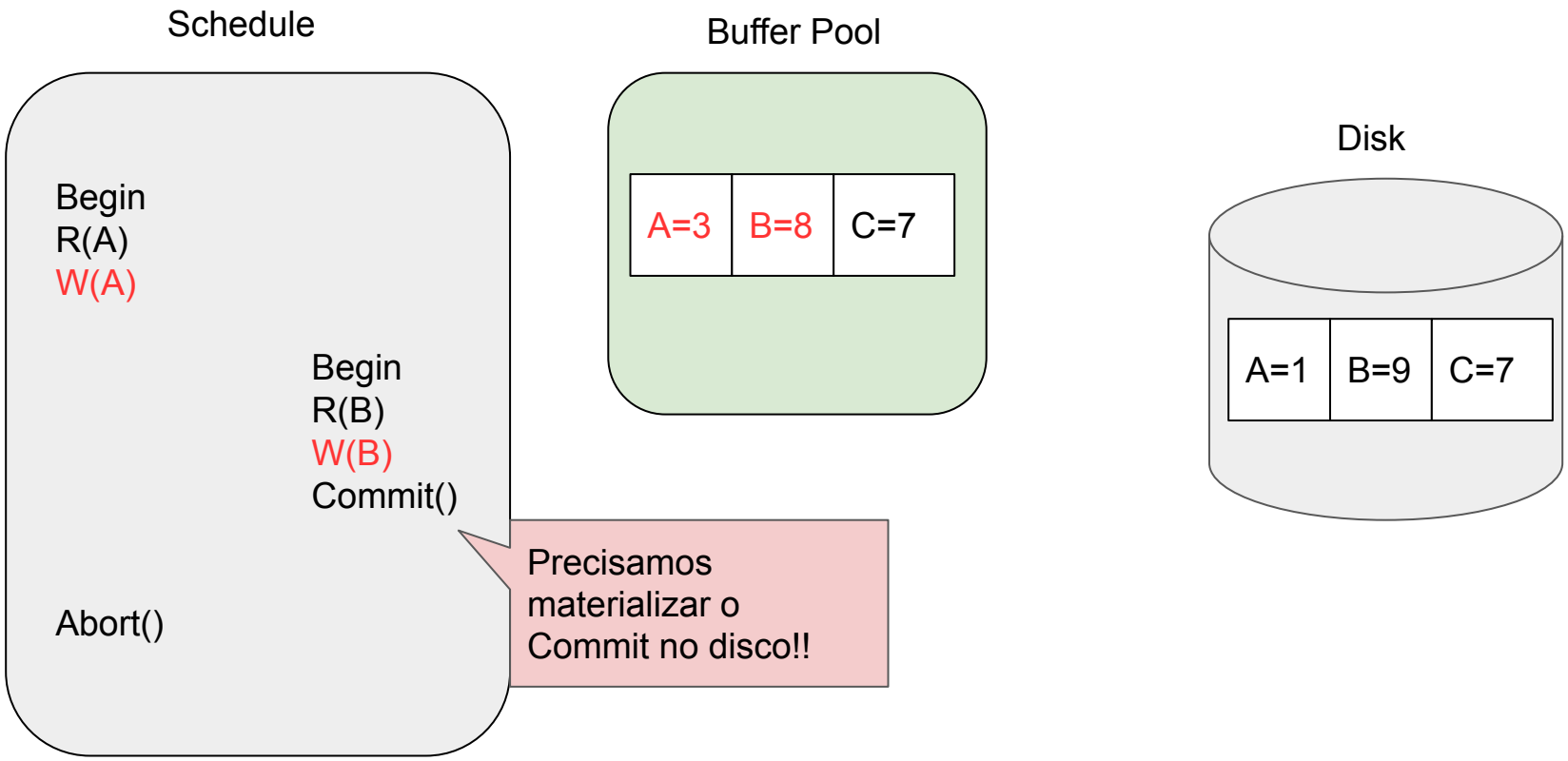
Recuperação após falha

Mecanismos de recuperação são técnicas que garantem a consistência, atomicidade e durabilidade em situações de falhas.

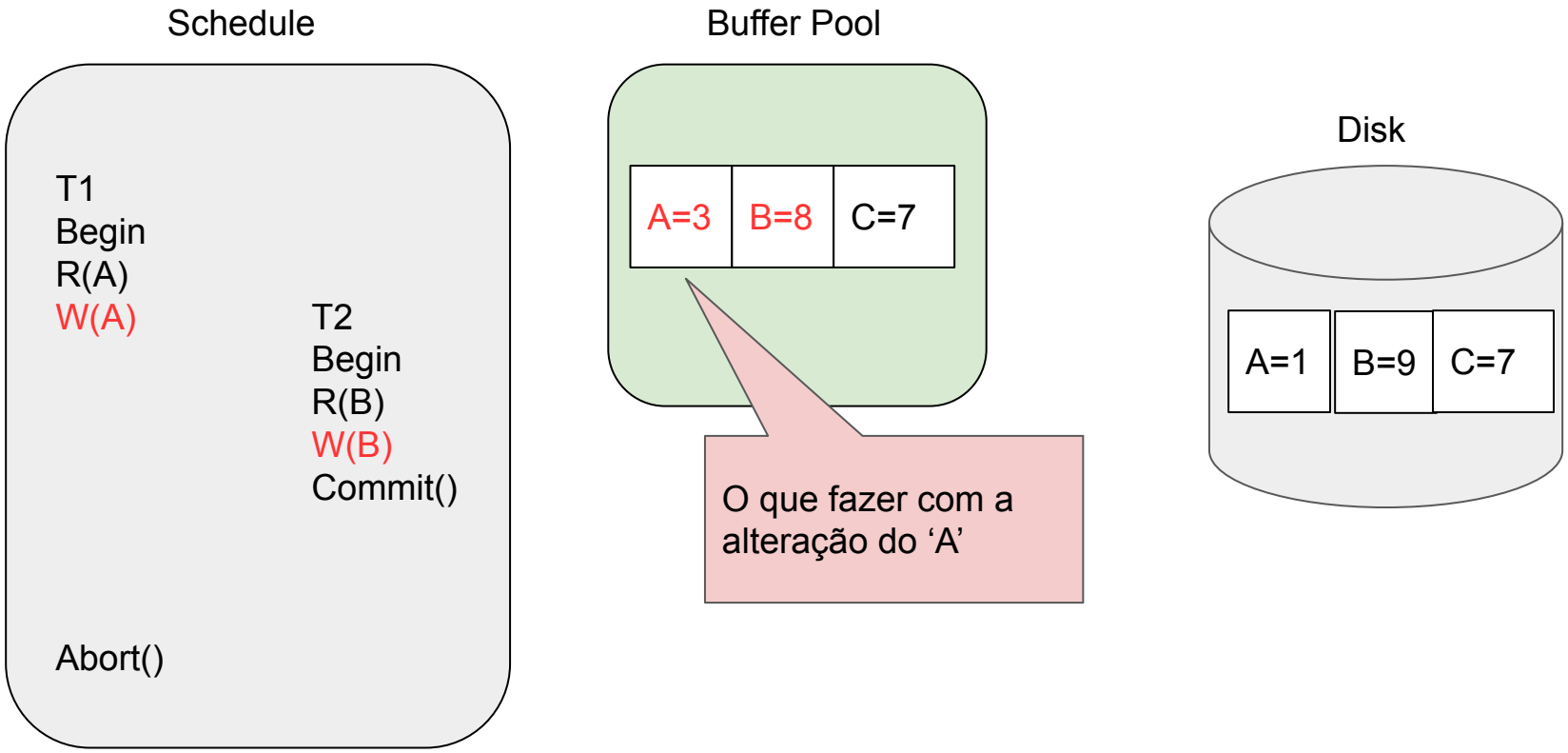
Motivação


Schedule
Begin
R(A)
W(A)
Begin
R(B)
W(B)
Commit()
Abort()
Buffer Pool
A=3
B=8
C=7
Disk
A=1
B=9
C=7

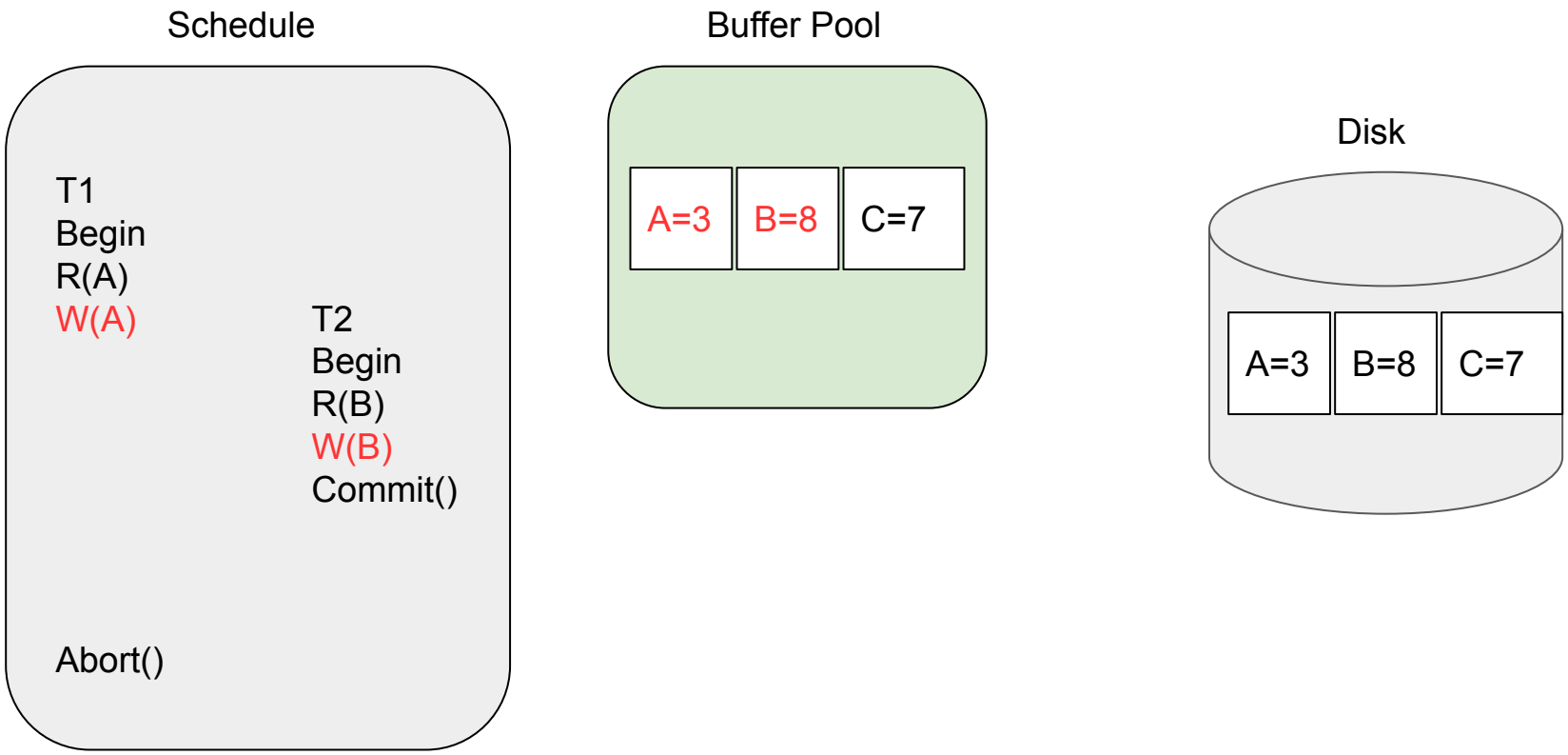
Motivação


Schedule
Begin
R(A)
W(A)
Begin
R(B)
W(B)
Commit()
Abort()
Precisamos materializar o Commit no disco!!
Buffer Pool
A=3
B=8
C=7
Disk
A=1
B=9
C=7

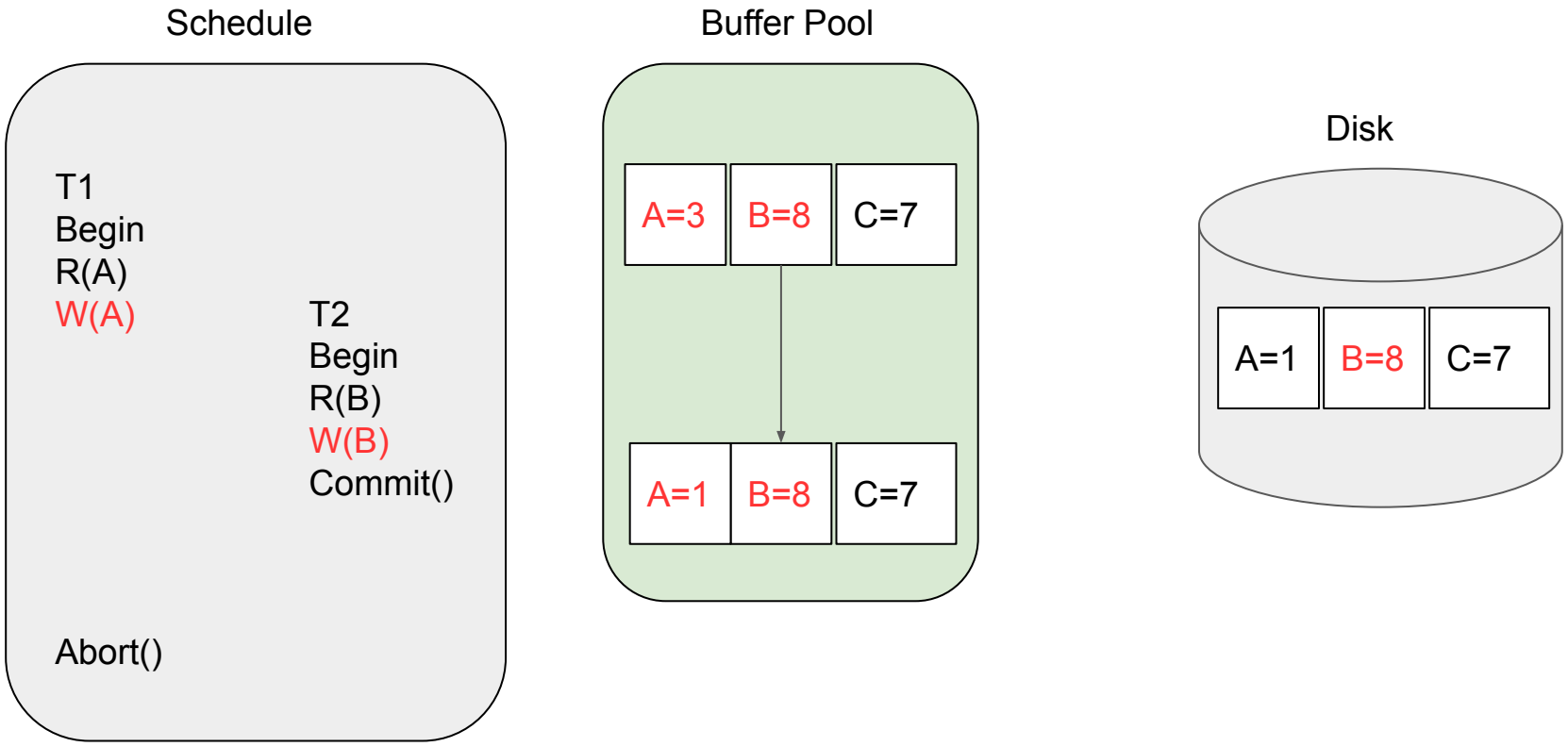
Motivação


Schedule
T1
Begin
R(A)
W(A)
T2
Begin
R(B)
W(B)
Commit()
Abort()
Buffer Pool
A=3
B=8
C=7
O que fazer com a alteração do 'A'
Disk
A=1
B=9
C=7

Motivação


Schedule
T1
Begin
R(A)
W(A)
T2
Begin
R(B)
W(B)
Commit()
Abort()
Buffer Pool
A=3
B=8
C=7
Disk
A=3
B=8
C=7

Motivação


Schedule
T1
Begin
R(A)
W(A)
T2
Begin
R(B)
W(B)
Commit()
Abort()
Buffer Pool
A=3
B=8
C=7
A=1
B=8
C=7
Disk
A=1
B=8
C=7