Mecanismos de Recuperação

Banco de Dados: Teoria e Prática

André Santanchè
Instituto de Computação – UNICAMP
Outubro 2019

Mecanismos de Recuperação

Propósito

- Restaurar o BD ao seu último estado consistente antes de uma falha
- Preservar as propriedades ACID: **A**tomicidade, **C**onsistência, **I**solamento e **D**urabilidade

Exercício 1

T1	T2
ler(X)	
X = X – N	
gravar(X)	
	ler(X)
	X = X + M
	gravar(X)
ler(Y)	
Y = Y + N	
gravar(Y)	
** crash **	

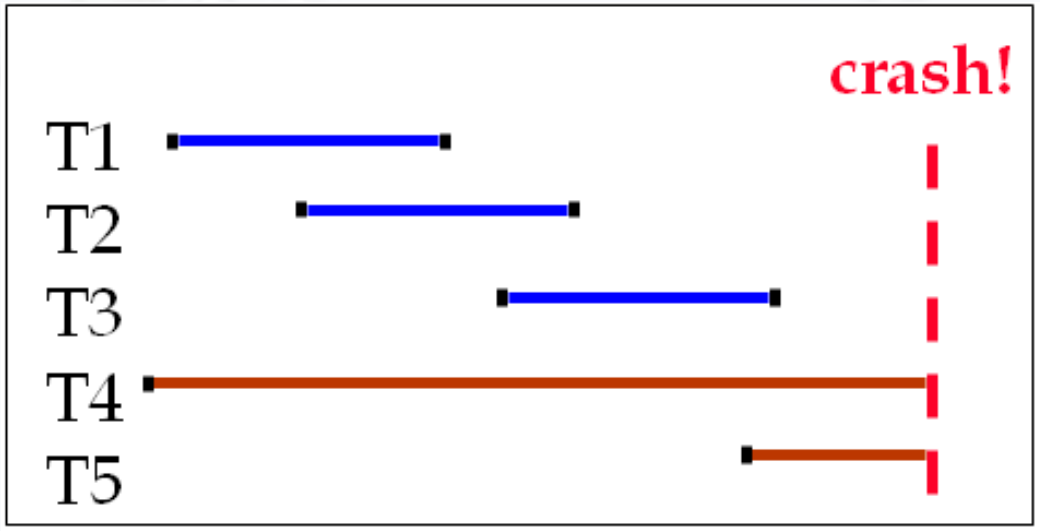
- Considerando que as operações à esquerda foram registradas em um banco de dados, proponha uma estratégia para garantir a atomicidade (T1 e T2 não realizaram COMMIT).

Propriedades ACID

- **Atomicidade**: todas as operações da transação acontecem ou nenhuma acontece
- Preservação de **Consistência:** a execução completa de uma transação faz o BD passar de um estado consistente para outro
- **Isolamento**: uma transação deve ser executada como se estivesse isolada das demais
- **Durabilidade** ou permanência: se uma transação é efetivada, seu efeito persiste

Gerenciamento de Recuperação

- O gerenciamento de recuperação garante a Atomicidade e Durabilidade
  - **Atomicidade**: transações podem reverter (*rollback*)
  - **Durabilidade**: o que fazer se o SGBD parar?
- Exemplos:
  - T1, T2 & T3 tem que permanecer
  - T4 & T5 devem ser revertidas



(Ramakrishnan, 2003b)

Falha

- Tipos de Falha:
  - Sem dano físico ao BD:
    - O computador falhar (crash ou queda de sistema)
    - Um erro de transação ou sistema
    - Erros locais ou condições de exceção detectadas pela transação
    - Imposição do controle de concorrência
  - Com dano físico ao BD:
    - Falha de disco
    - Problemas físicos e catástrofes

Cache

- Cache do SGBD
  - Baseado em páginas de disco mantidas pelo SO
  - SGBD chama rotinas de baixo nível do SO

Dados por Página do Cache

- Bit sujo
  - 0 → página não alterada
  - 1 → página alterada
- Bit preso-solto
  - 0 → página pode ser gravada
  - 1 → página ainda não pode ser gravada (e.g., espera de commit)
- Transações que modificaram a página

Cache
Atualização Shadow x In-place

- Shadow: versão modificada de um item gravada em nova localização de disco
- In-place: versão modificada de item sobrescreve a anterior (recuperação por Log)

Log

- Mantém o registro sequencial das operações de transação que afetam itens do BD
- Estes dados podem ser necessários para:
  - desfazer ações de uma transação “abortada”
  - recuperar o sistema de falhas
  - Auditoria
- O Log é mantido em disco
  - afetado apenas pelas falhas em disco ou catastróficas
  - recomenda-se um disco separado

Protocolo Write-Ahead Logging (WAL)

- Grava registro de operação no disco de log antes que a modificação do item seja gravada em disco
  - garante atomicidade
- Todas as operações de uma transação são gravadas no disco de log antes do commit
  - garante durabilidade

(Ramakrishnan, 2003b)

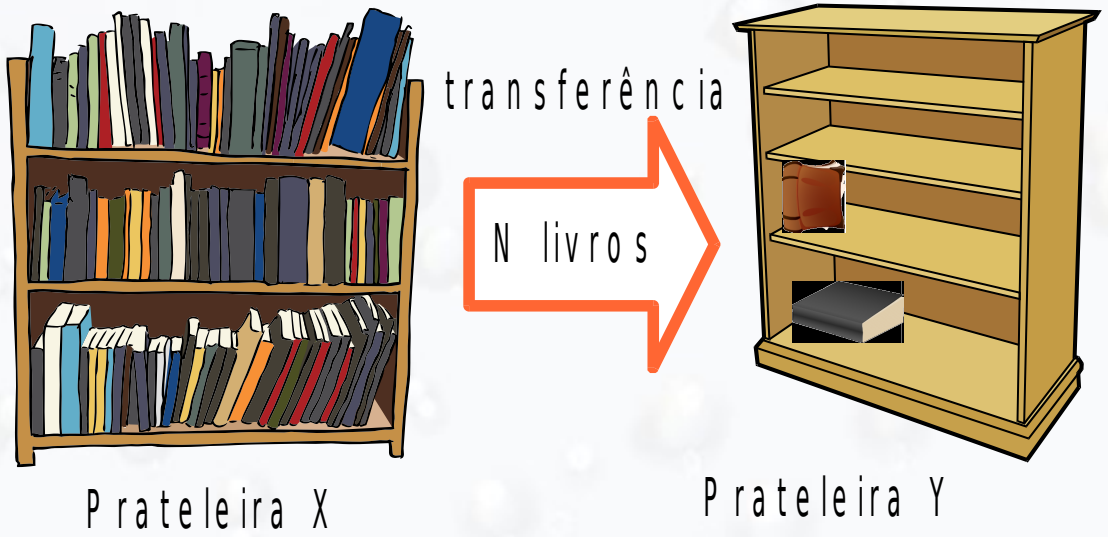
Tipos de Registro do Log

- [**start_transaction**,T]
- [**write_item**,T,X,valor_antigo,novo_valor]
- [**read_item**,T,X]
- [**commit**,T]
- [**abort**,T]
- [**checkpoint**]

Exemplo de Log

Transação 1: Transferência

T1
ler(X)
X = X - N
gravar(X)
ler(Y)
Y = Y + N
gravar(Y)



Exemplo de Log
Transação 2: Aquisição

T2

```
ler(X)
X = X + M
gravar(X)
```

Plano de Execução

T1	T2
início	
ler(X)	
X = X – N	
	início
gravar(X)	
	ler(X)
	X = X + M
ler(Y)	
	gravar(X)
Y = Y + N	
gravar(Y)	
commit	
	commit

Plano de Execução

S
início
ler(X)
X = X – N
início
gravar(X)
ler(X)
X = X + M
ler(Y)
gravar(X)
Y = Y + N
gravar(Y)
commit
commit

Log

- Cada transação T tem um identificador único gerada automaticamente pelo sistema
- Campos para recuperação (UNDO e REDO):
  - BFIM (Before Image): estado antes da alteração
    - usado para UNDO
  - AFIM (After Image): estado depois da alteração
    - usado para REDO

Exemplo de Log

X = 50 **Y = 110**
N = 20 **M = 40**

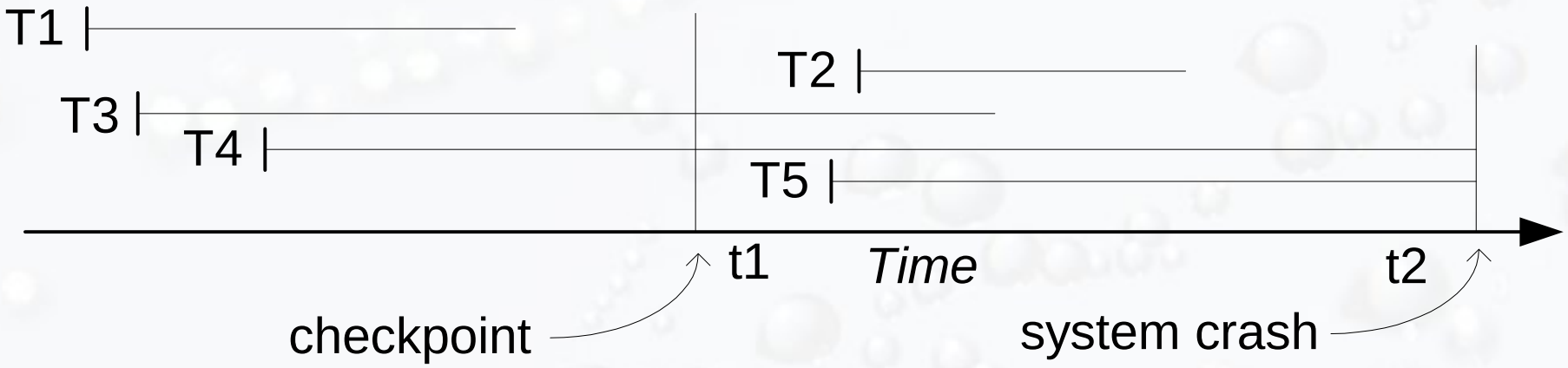
Plano	
início	
ler(X)	ler(50)
X = X – N	X = 50 – 20
início	
gravar(X)	gravar(30)
ler(X)	ler(30)
X = X + M	X = 30 + 40
ler(Y)	ler(110)
gravar(X)	gravar(70)
Y = Y + N	Y = 110 + 20
gravar(Y)	gravar(130)
commit	
commit	

LOG							
	Trans.	^Trás	^Frente	Op.	Item	BFIM	AFIM
1	T1	0	2	start			
2	T1	1	4	read	X		
3	T2	0	5	start			
4	T1	2	6	write	X	50	30
5	T2	3	7	read	X		
6	T1	4	9	read	Y		
7	T2	5	10	write	X	30	70
8	T1	6	9	write	Y	110	130
9	T1	9	-	commit			
10	T2	7	-	commit			

Checkpoint

- Entrada no LOG gravada periodicamente
- Indica gravação de todos os dados modificados do buffer em disco

Recuperação



(Elmasri, 2004)

Checkpoint Fuzzy

- Usa [begin_checkpoint] no início do processo e libera para outros processos
- Usa [end_checkpoint] no final
  - Não válido enquanto não alcança este ponto

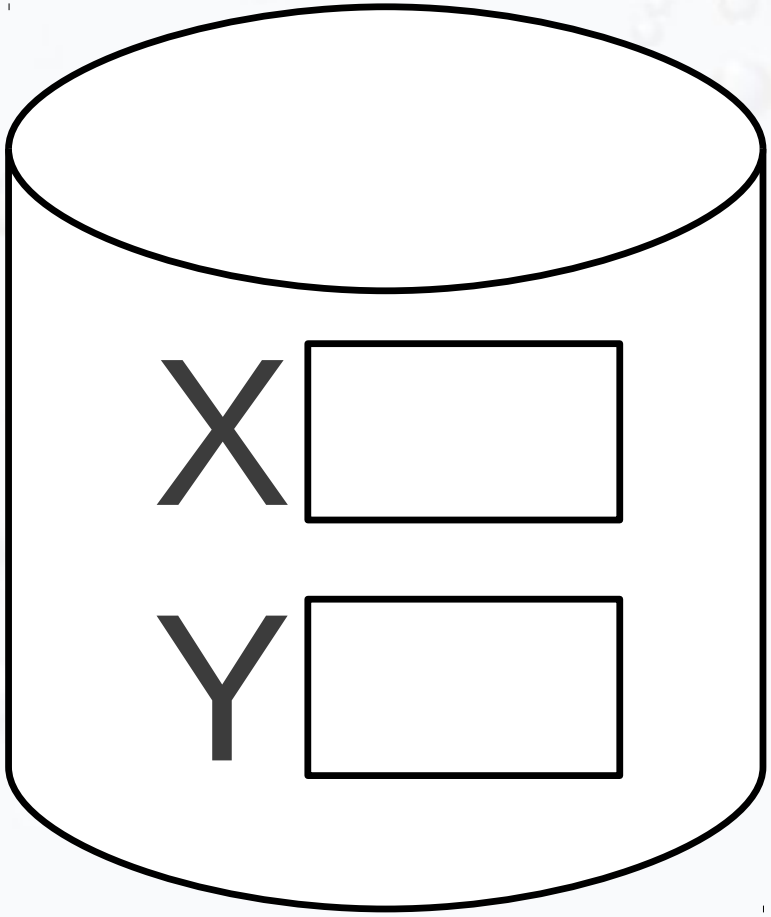
Gravando Cache no Disco

- Forçar gravação dos itens alterados no disco quando alterados (antes do commit)?
  - Sim: Force
  - Não: No-force
- Permitir alguma gravação antes do commit?
  - Sim: Steal
  - Não: No-steal

Force

S	LOG
início	□
ler(X)	□
X = X - N	□
início	□
gravar(X)	□
ler(X)	□
X = X + M	□
ler(Y)	□
gravar(X)	□
Y = Y + N	□
gravar(Y)	□
commit	□
commit	□
	□

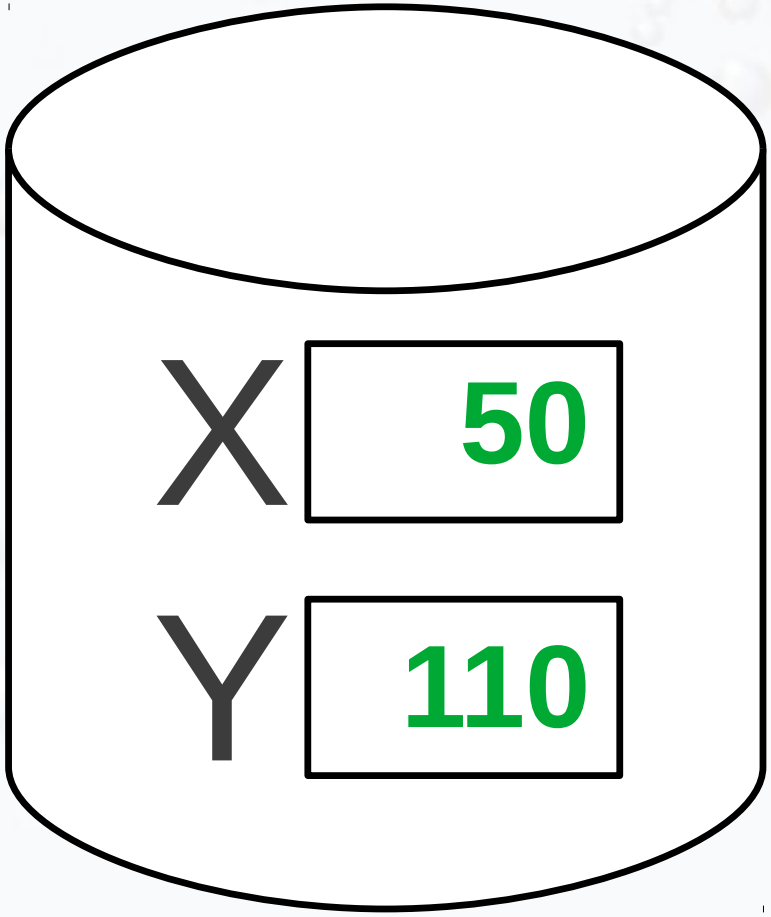
X=50 N=20
Y=110 M=40



Force

S	LOG
início	☒
ler(X)	☒
X = X - N	☐
início	☒
gravar(X)	☐
ler(X)	☐
X = X + M	☐
ler(Y)	☐
gravar(X)	☐
Y = Y + N	☐
gravar(Y)	☐
commit	☐
commit	☐

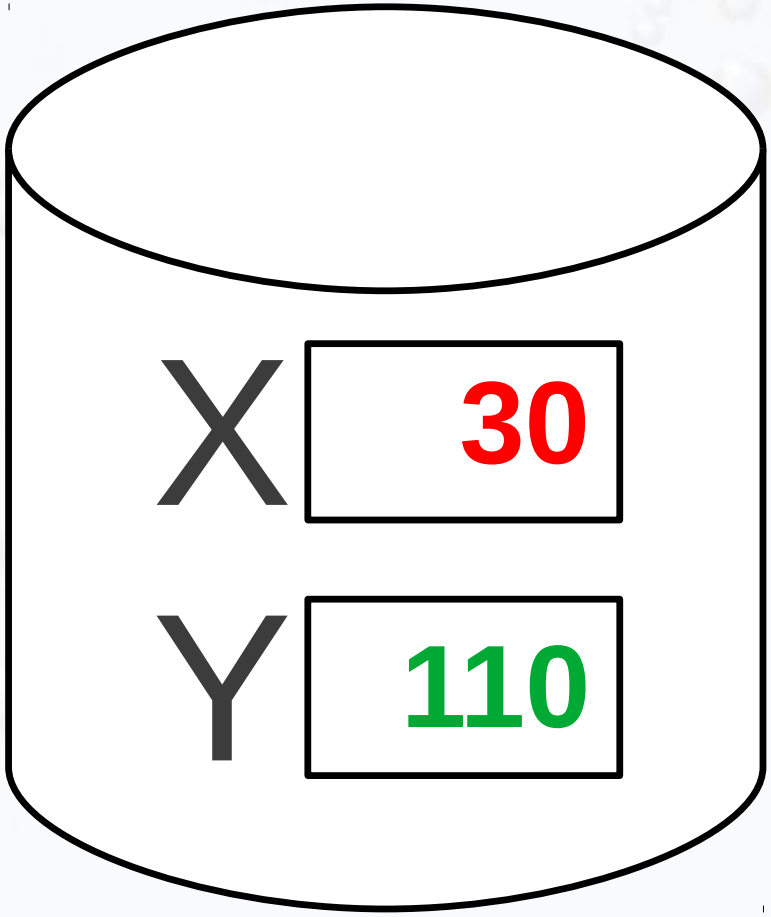
X=50 N=20
Y=110 M=40



Force

S	LOG
início	☒
ler(X)	☒
X = X – N	☐
início	☒
gravar(X)	☒
ler(X)	☐
X = X + M	☐
ler(Y)	☐
gravar(X)	☐
Y = Y + N	☐
gravar(Y)	☐
commit	☐
commit	☐
	☐

X=50 N=20
Y=110 M=40

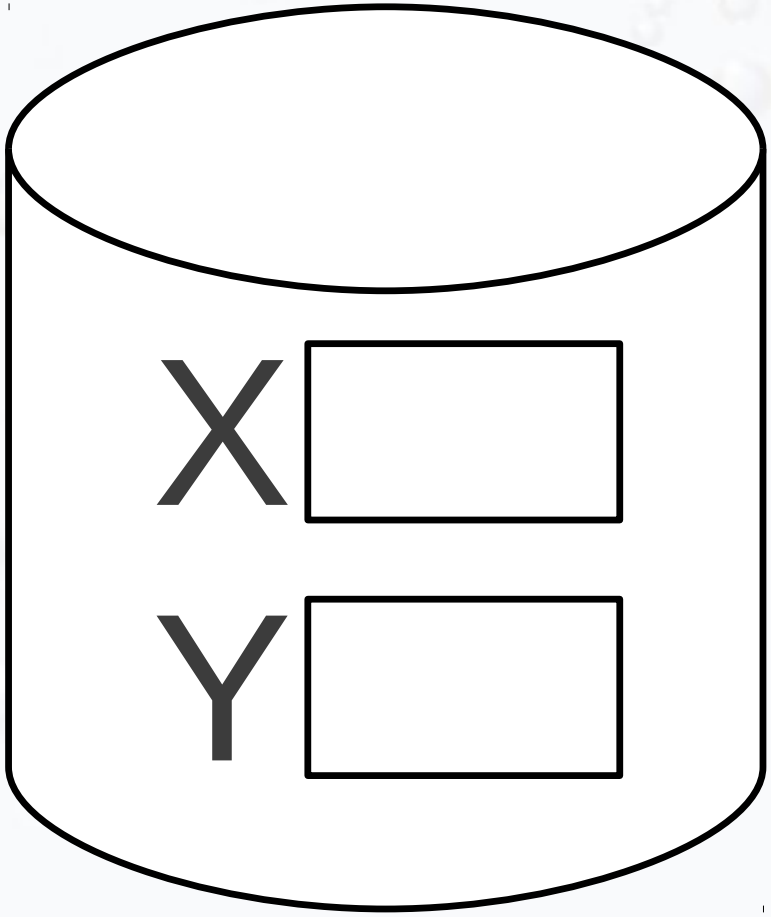


Force → No-REDO

No-Steal

S	LOG
início	□
ler(X)	□
X = X - N	□
início	□
gravar(X)	□
ler(X)	□
X = X + M	□
ler(Y)	□
gravar(X)	□
Y = Y + N	□
gravar(Y)	□
commit	□
commit	□
	□

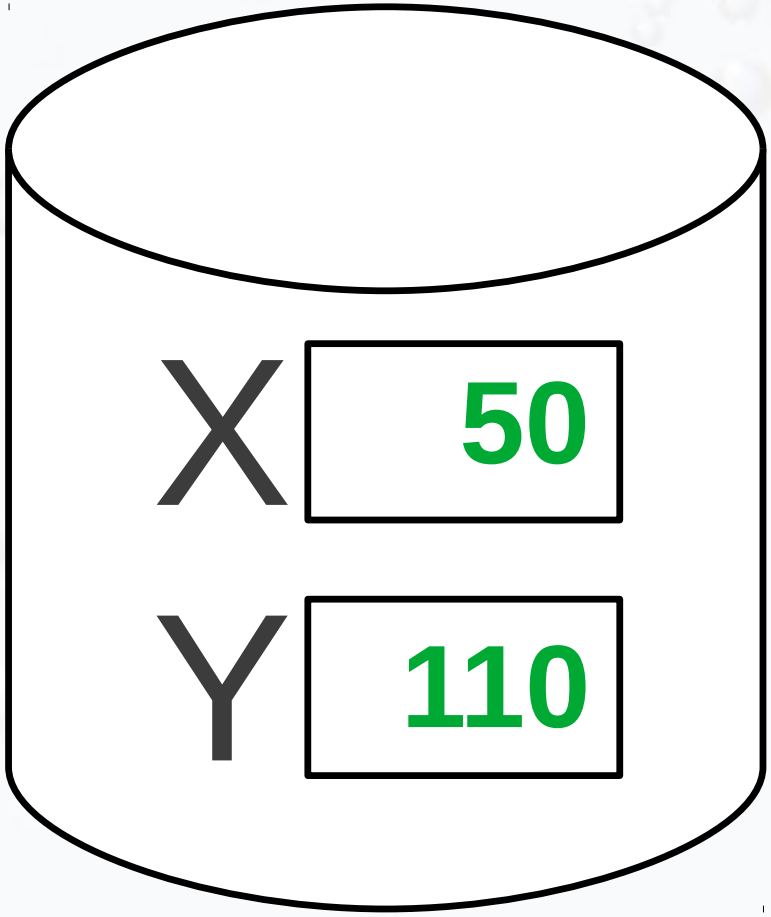
X=50 N=20
Y=110 M=40



Force

S	LOG
início	⊗
ler(X)	⊗
X = X - N	☐
início	⊗
gravar(X)	⊗
ler(X)	⊗
X = X + M	☐
ler(Y)	⊗
gravar(X)	⊗
Y = Y + N	☐
gravar(Y)	☐
commit	☐
commit	☐
	☐

X=50 N=20
Y=110 M=40



No-Steal → No-UNDO

Recuperação e Steal/No-Steal x Force/No-Force

- Steal/No-Force (Undo/Redo)
- Steal/Force (Undo/No-redo)
- No-Steal/No-Force (Redo/No-undo)
- No-Steal/Force (No-undo/No-redo)

André Santanchè
http://www.ic.unicamp.br/~santanche

Referências

- Elmasri, Ramez; Navathe, Shamkant B. (2005) **Sistemas de Bancos de Dados**. Addison-Wesley, 4ª edição em português.
- Elmasri, Ramez; Navathe, Shamkant B. (2010) **Sistemas de Banco de Dados**. Pearson, 6ª edição em português.
- Ramakrishnan, Raghu; Gehrke, Johannes (2003) **Database Management Systems.** McGraw-Hill, 3rd edition.
- Ramakrishnan, Raghu; Gehrke, Johannes (2003b) Database Management Systems. McGraw-Hill, 3rd edition (companion slides).

Licença

- Estes slides são concedidos sob uma Licença Creative Commons. Sob as seguintes condições: Atribuição, Uso Não-Comercial e Compartilhamento pela mesma Licença.
- Mais detalhes sobre a referida licença Creative Commons veja no link: http://creativecommons.org/licenses/by-nc-sa/3.0/
- Agradecimentos: fotografia da capa e fundo por Ben Collins -http://www.flickr.com/photos/graylight/. Ver licença específica em http://www.flickr.com/photos/graylight/261480919/