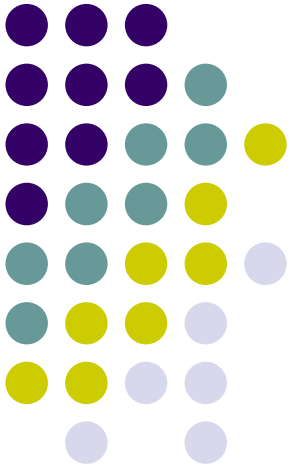
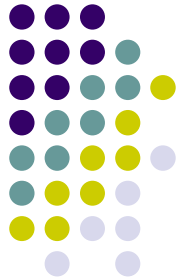
Processamento Transacional Distribuído

Universidade Federal do ABC

Prof. Dr. Francisco **Isidro** Massetto

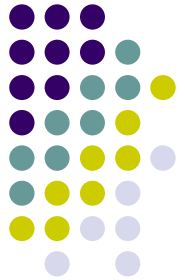


Introdução



- Exclusão mútua pode ser incômodo ao programador
  - Baixo nível na programação
- Desejável – maior nível de abstração
- Cenário: processo comunica aos demais que deseja fazer um acesso a um recurso compartilhado
  - Todos os participantes executam várias operações
  - Participam de uma transação
  - A transação só é concluída quando todos conseguem completar suas atribuições (commit)
  - Caso contrário, todo o trabalho é desfeito e volta para um estado anterior à execução (rollback)

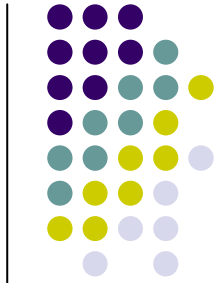
Exemplo



- Transações Bancárias – Transferência de Valores
  - Retira(conta1, valor)
  - Deposita(conta2, valor)

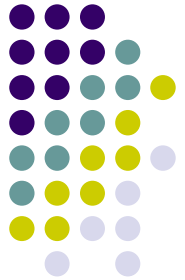
- E se houvesse uma falha entre a operação de retirada e a de depósito?
  - Para onde iria o dinheiro?

Solução – Transações Atômicas



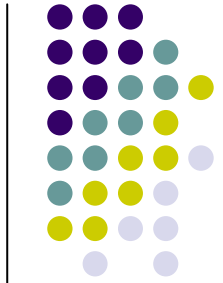
- Algumas Primitivas
  - BEGIN_TRANSACTION
  - END_TRANSACTION
    - Também conhecida como **commit**
  - ABORT_TRANSACTION
    - Também conhecida com **rollback**
  - READ
  - WRITE

Propriedades



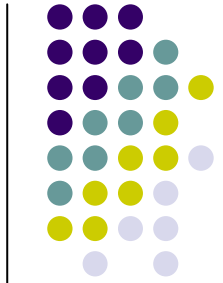
- Atomicidade
  - A transação deve apresentar-se como indivisível
- Consistência
  - A base de dados manipulada deve sair de um estado consistente para outro estado consistente
- Isolamento
  - Várias transações ocorrendo simultaneamente não interferem umas nas outras
- Durabilidade
  - Após a conclusão, as alterações tornam-se permanentes

Implementação de Transações



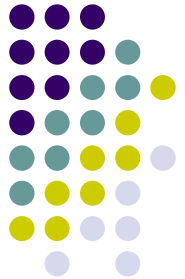
- Espaço de Trabalho privado
  - Cópias locais das tabelas envolvidas são feitas para os participantes
  - Falhas durante a execução não irão refletir na transação como um todo
  - Toda a operação é feita localmente e depois reescrita na versão original
  - Problema: overhead de cópias inviabiliza a implementação

Implementação



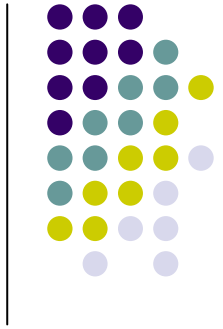
- Log – Lista de Inteções
  - Cada operação é escrita em um meio estável (arquivo)
    - Tipo da operação, valores originais e alterados
  - Quando a operação é concluída, o log pode ser descartado
  - Quando a operação falha, o log é lido na ordem inversa e as alterações revertidas

Implementação



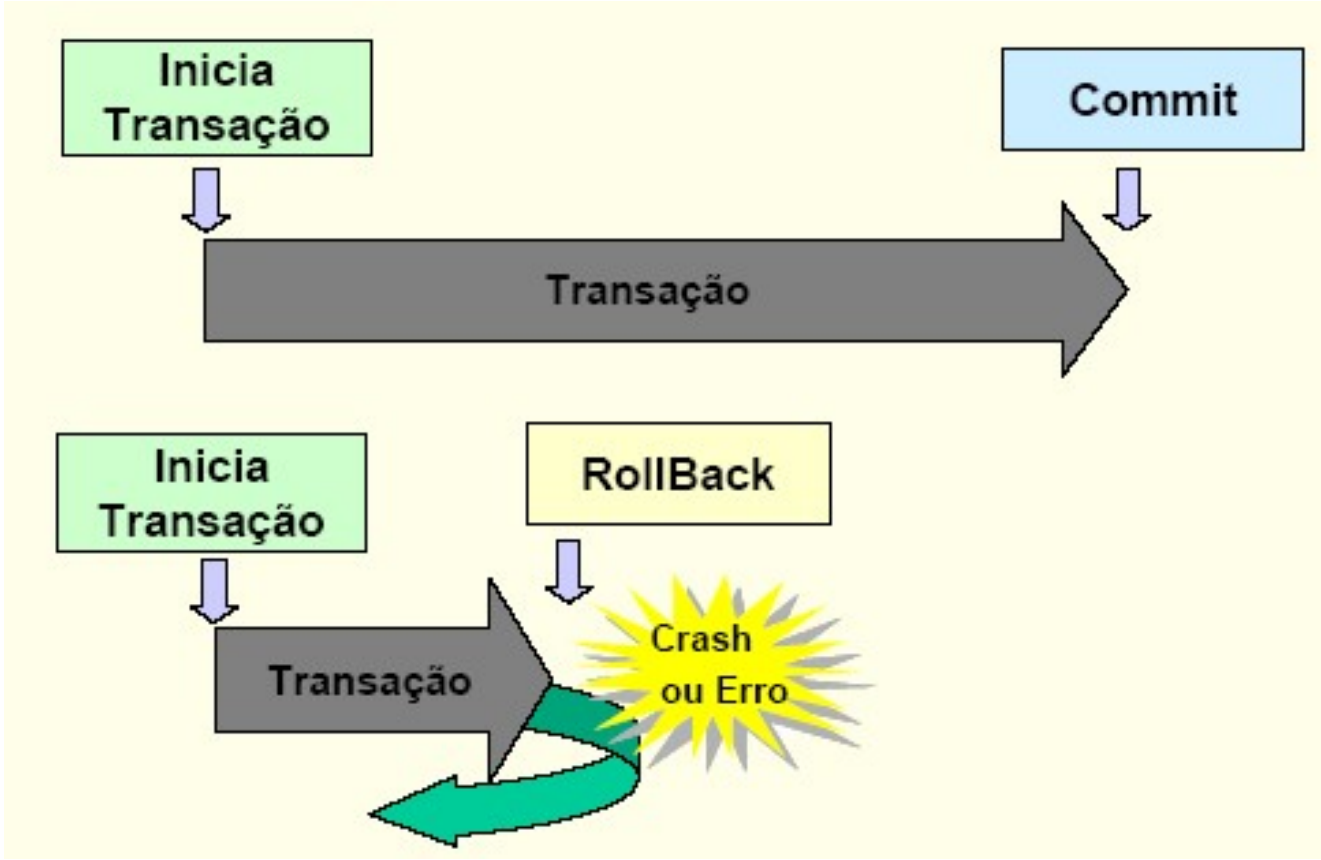
- Two-Phase Commit Protocol
  - Um dos processos é considerado o coordenador da transação
  - Escreve no Log que iniciou o protocolo e solicita a todos os demais que confirmem suas operações
  - Cada subordinado escreve seu resultado no log e comunica o coordenador
  - O coordenador recolhe todos os resultados e decide:
    - Se todos confirmaram, envia uma mensagem confirmando toda a operação aos participantes
    - Se algum participante falhou, envia uma mensagem de rollback aos demais, para que desfaçam suas alterações

Modelos de Transações

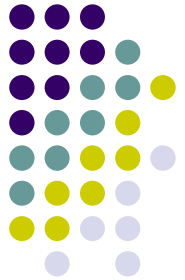


- Simples
  - Flat transactions
- Encadeadas
  - Syncpoints
  - Chained transactions
  - SAGA
- Aninhadas
  - Nested Transactions

Flat Transactions

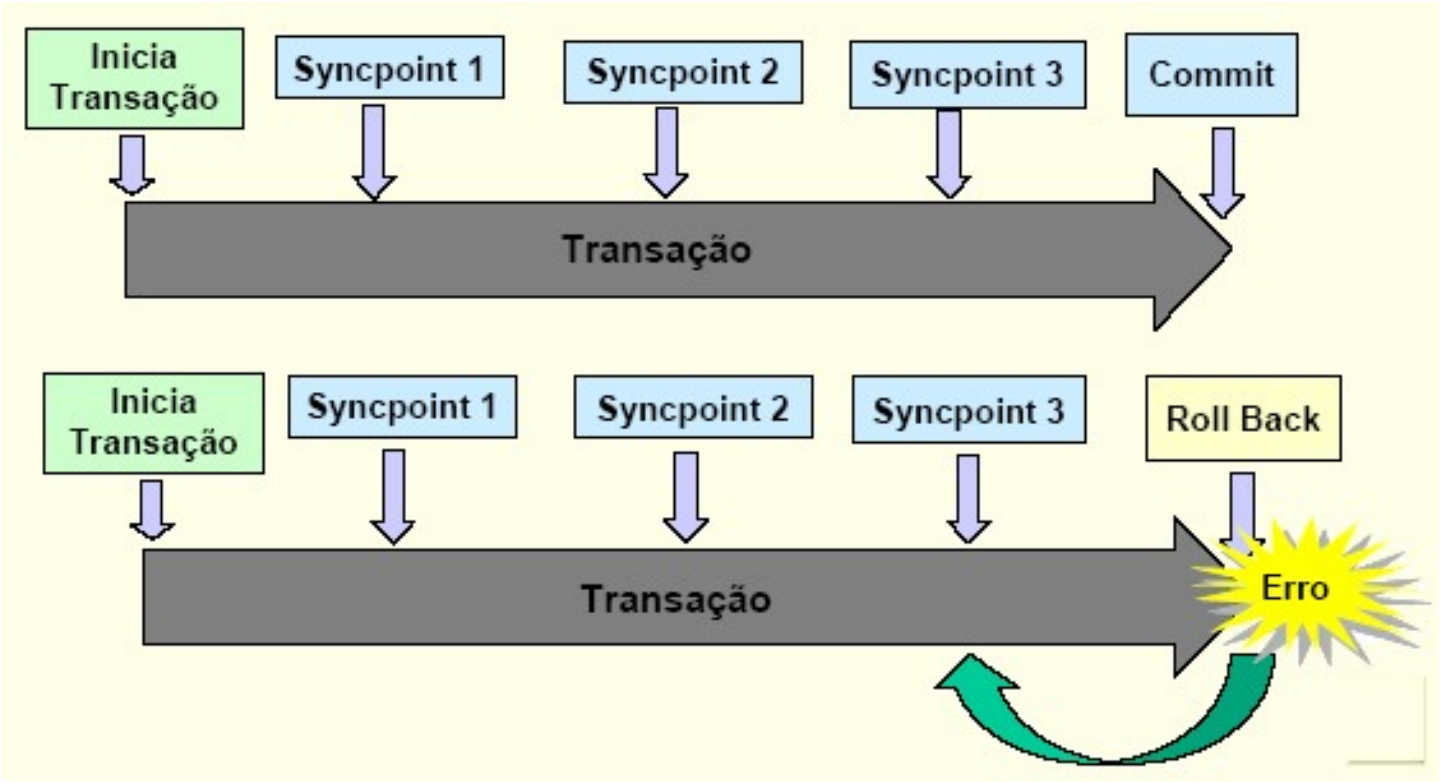


Flat Transactions

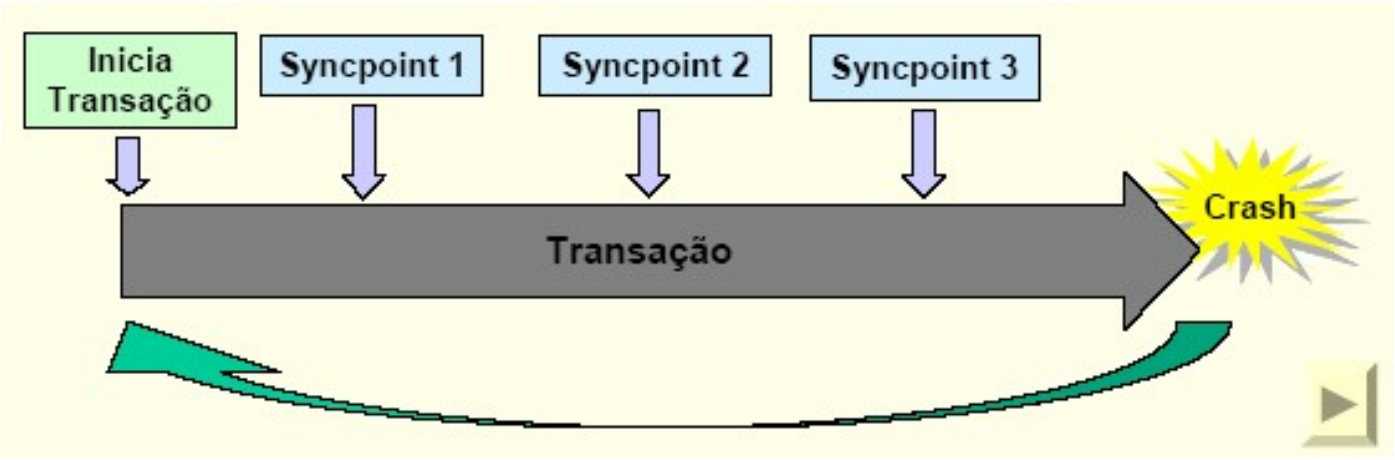


- Mais utilizada
  - Em torno de 90% das aplicações
- Para transações com tempo curto de execução
- Devem ser utilizadas com poucas operações para não monopolizarem recursos
  - Ver gráfico de alocação e liberação

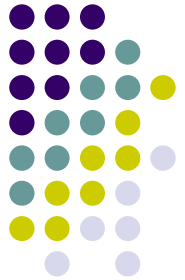
SyncPoints


Inicia Transação
Syncpoint 1
Syncpoint 2
Syncpoint 3
Commit
Transação
Inicia Transação
Syncpoint 1
Syncpoint 2
Syncpoint 3
Roll Back
Transação
Erro

SyncPoints

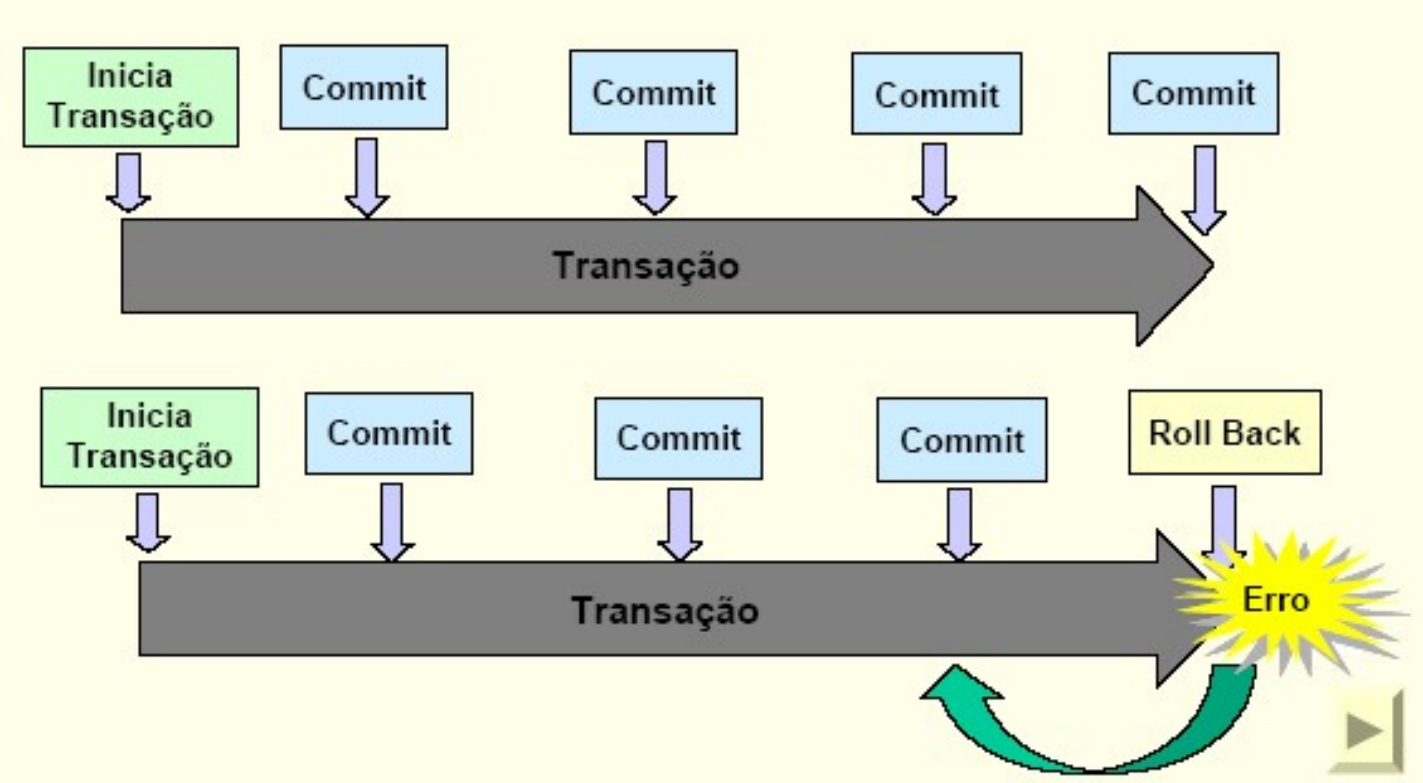

Inicia Transação
Syncpoint 1
Syncpoint 2
Syncpoint 3
Transação
Crash

SyncPoints



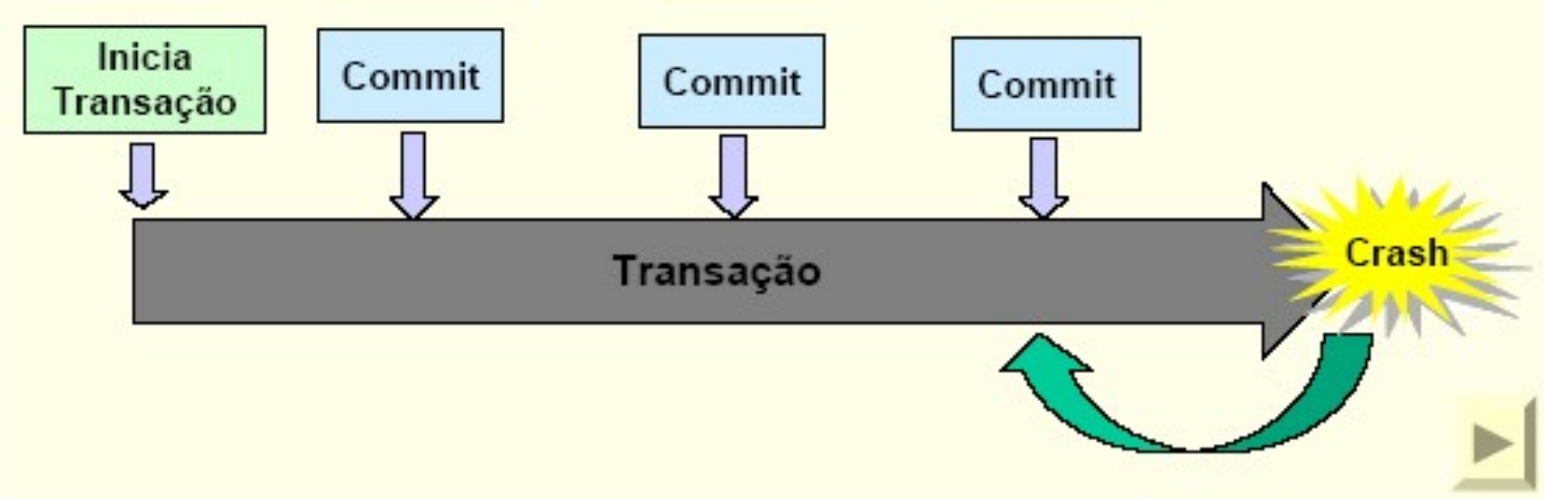
- SyncPoints são voláteis
- Commits são permanentes
- Em caso de erros → retorna ao último SyncPoint
- Em caso de falhas → retorna ao início da transação

Chained Transactions

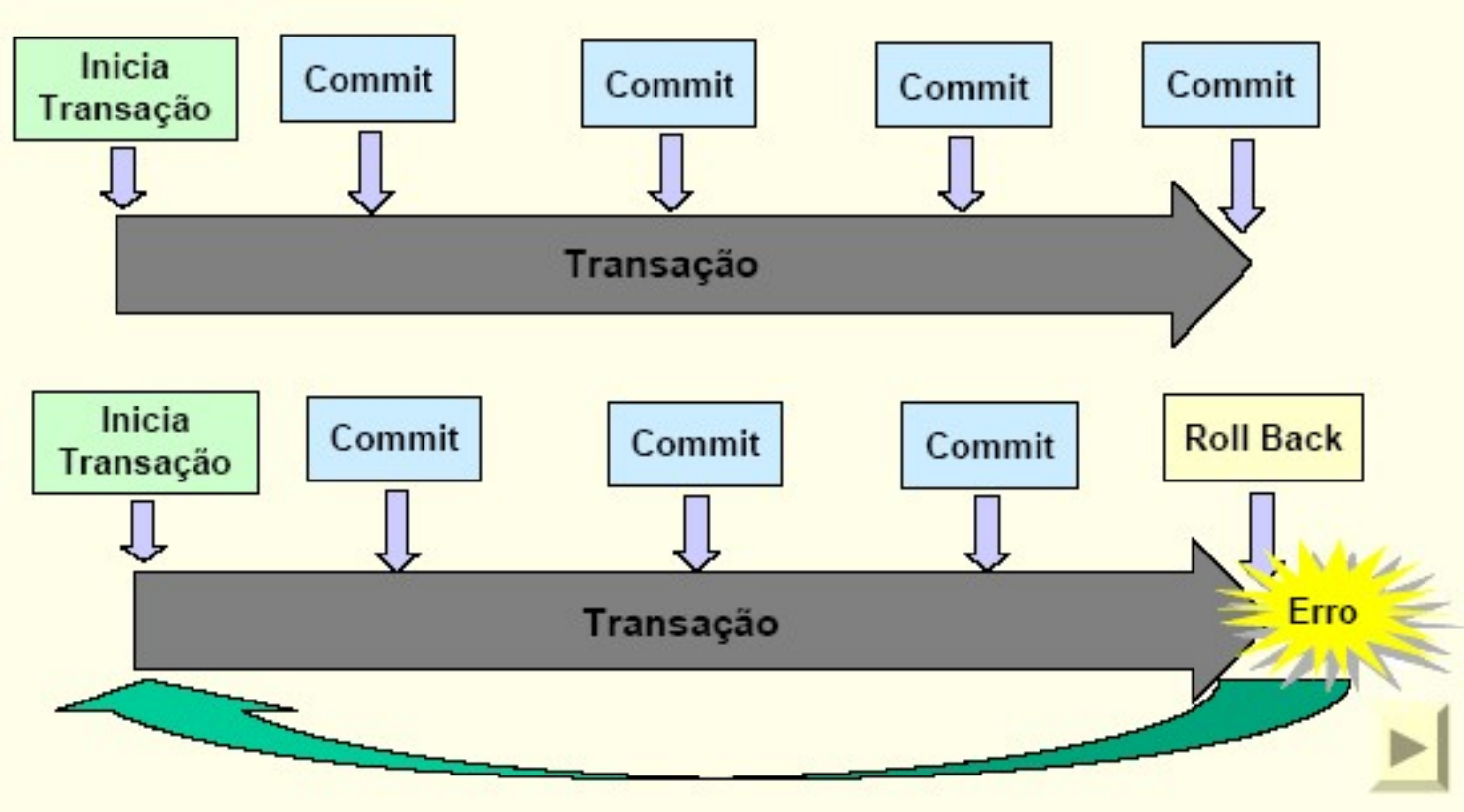


Chained Transactions

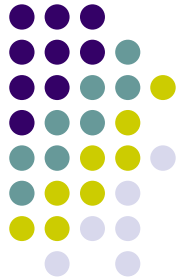
- SyncPoints são substituídos por commits
- Casos de erros ou de crashs são tratados da mesma forma
  - Retorna ao último commit realizado



SAGA

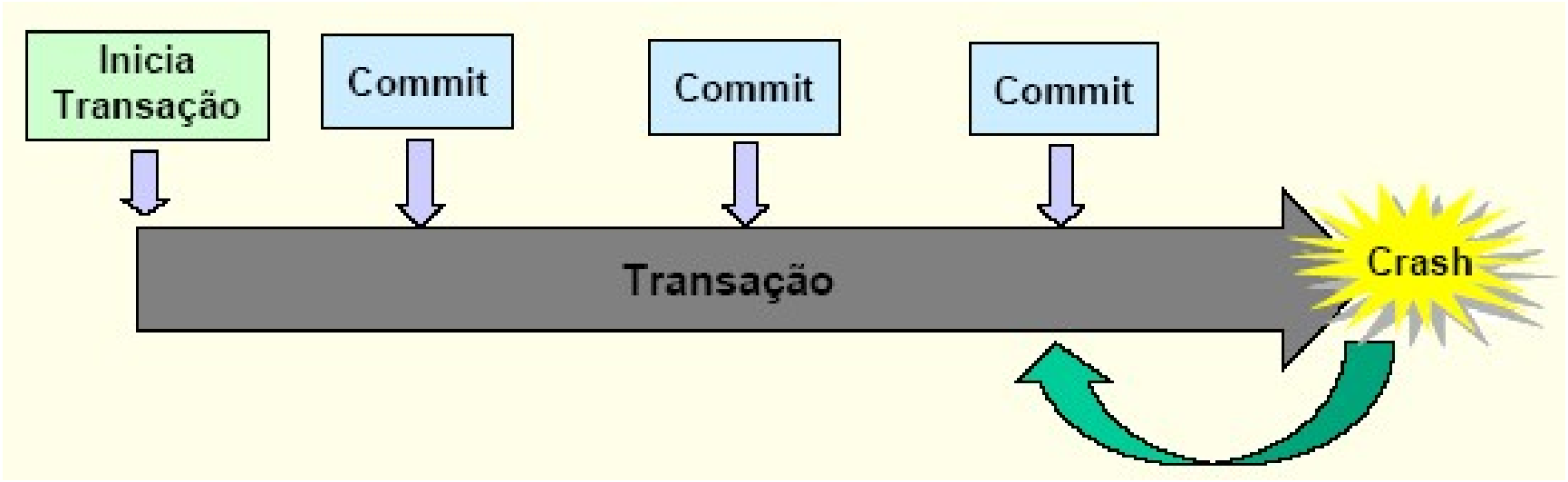

Inicia Transação
Commit
Commit
Commit
Commit
Transação
Inicia Transação
Commit
Commit
Commit
Roll Back
Transação
Erro

SAGA



- Situações e Erro e Crash são tratadas de formas diferentes
  - Erros são decorrentes de problemas na aplicação
    - Retorna ao início da transação
  - Falhas são eventos de anormalidade no sistema
    - Retorna ao último commit realizado

SAGA



Nested Transactions