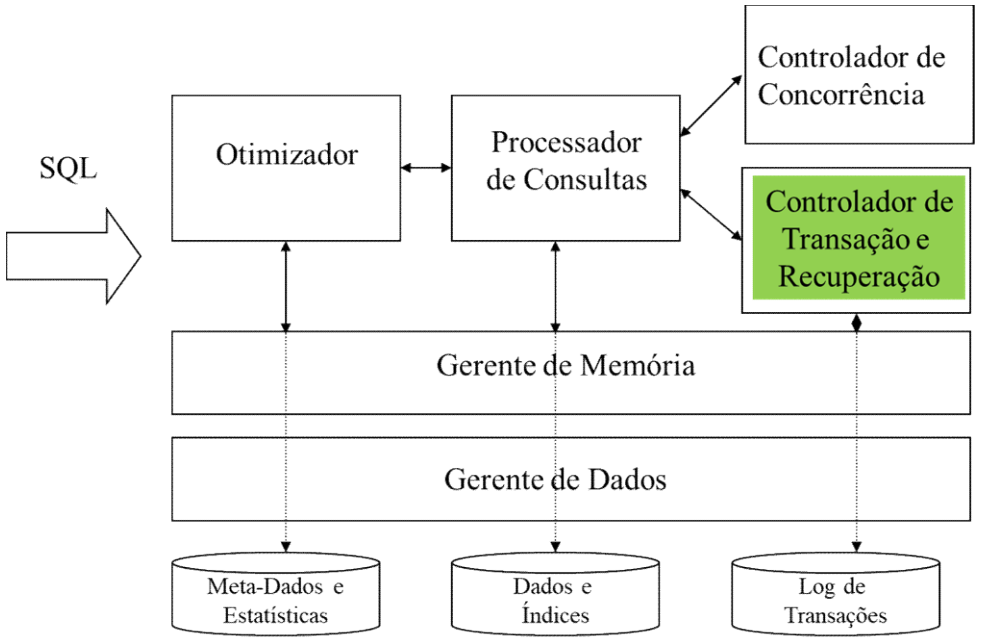
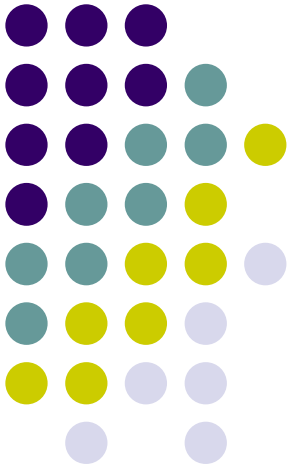
Bancos de Dados II

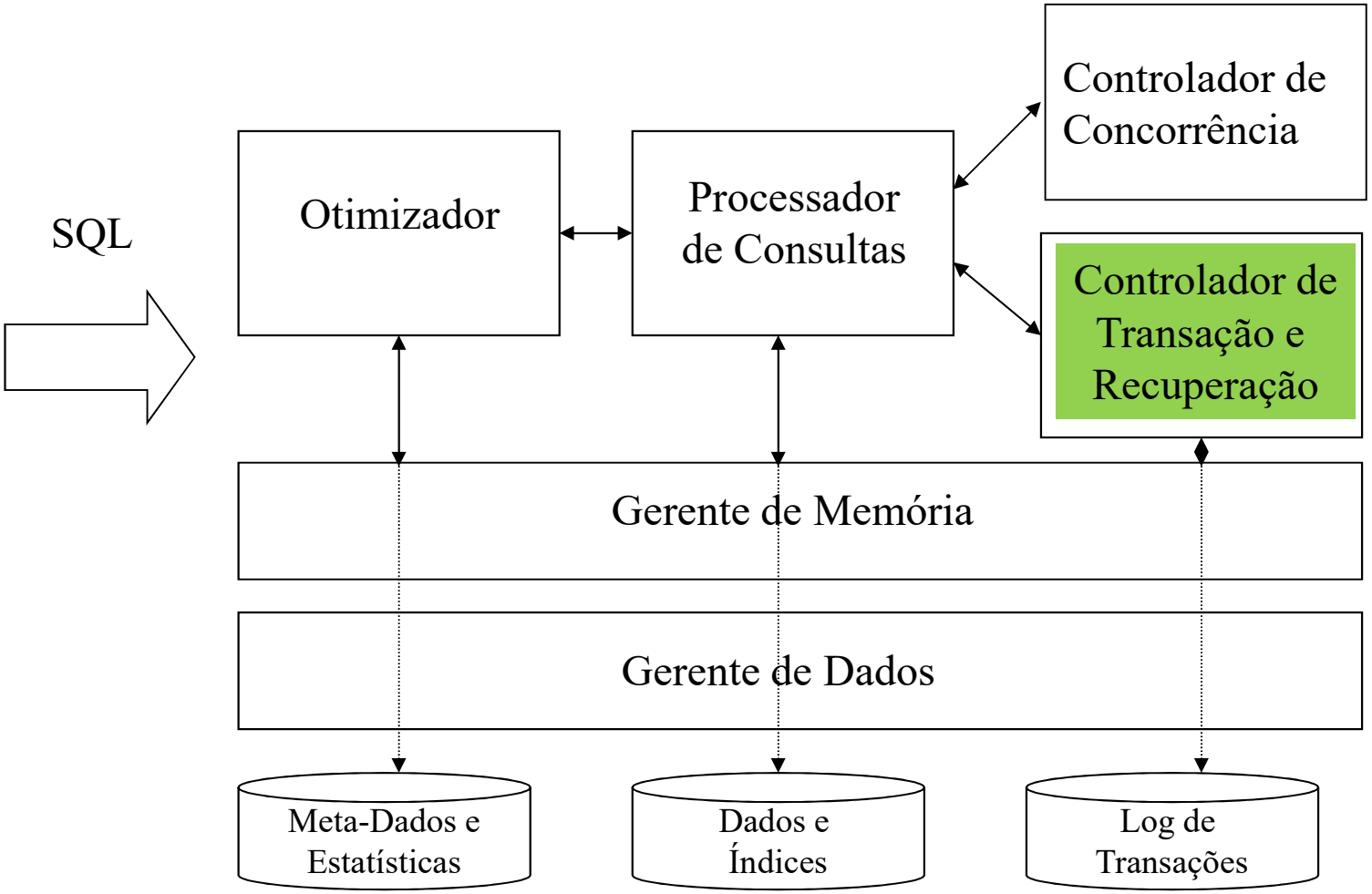
Controle de Transações e Recuperação





Arquitetura Funcional

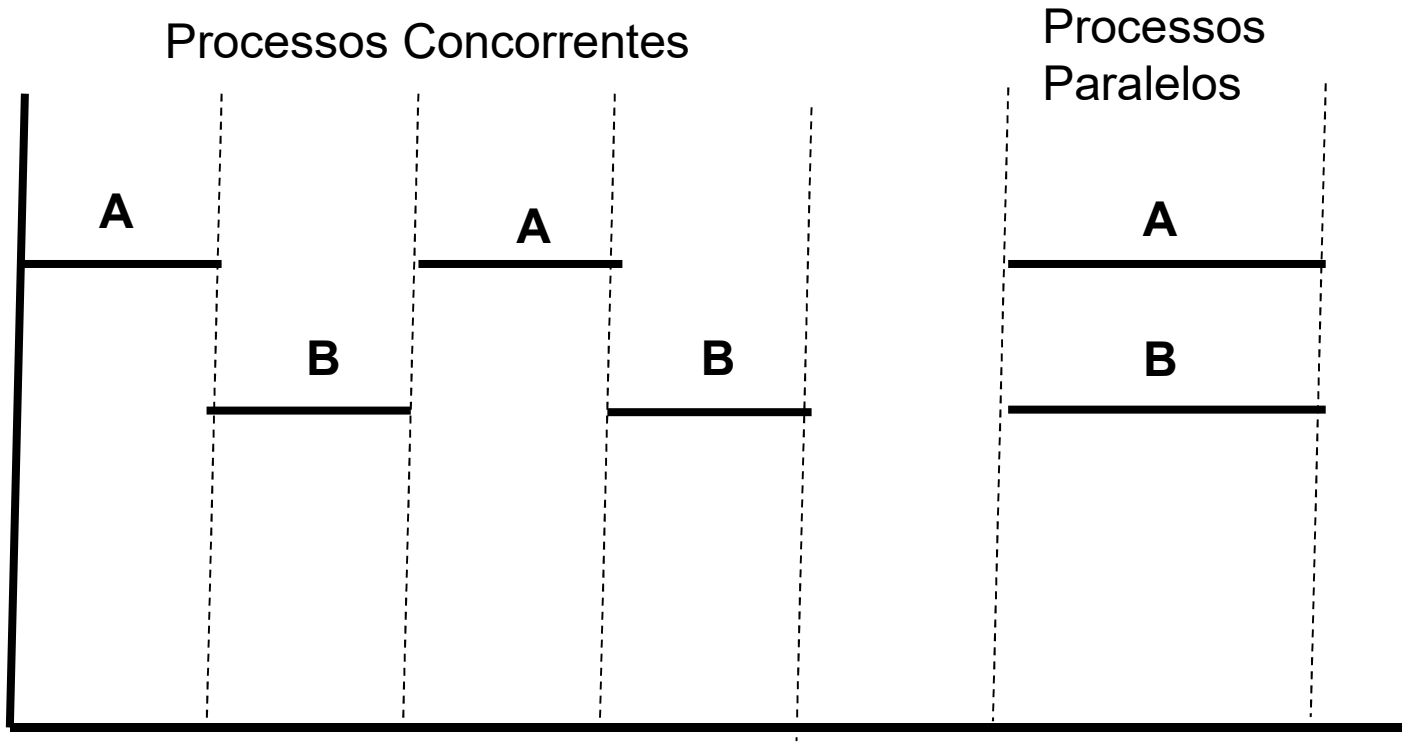
Componentes de um SGBD



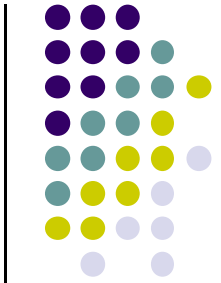
Transações

Critérios de classificação de bancos de dados:

- Número de usuários: Monousuário x Multiusuário
- Número de processos em execução (S.O.): Monoprogramação x Multiprogramação



Transações



- Transação: programa em execução que forma uma unidade lógica de processamento.

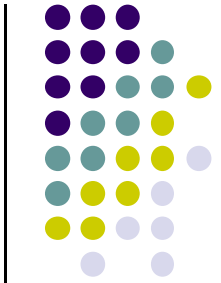
- Uma transação inclui uma ou mais operações de acesso ao banco de dados.

- Uma transação pode estar embutida em um programa ou pode ser especificada interativamente, via SQL.

- Uma forma de delimitar uma transação seria explicitar o seu início e fim vis declarações do tipo:
- BEGIN TRANSACTION e END TRANSACTION.

- Um programa de aplicação pode conter mais de uma transação.

Transações



Operações que compõem uma transação:

ler_item(x)

- Encontrar o bloco no disco que contém o item X
- Copiar o bloco para o buffer de memória (se já não estiver lá)
- Copiar o item X para a variável do programa

e

gravar_item(x)

- Encontrar o bloco no disco que contém o item X
- Copiar o bloco para o buffer de memória (se já não estiver lá)
- Copiar a variável do programa que contém o item para o buffer
- Armazenar o bloco no disco

Transações

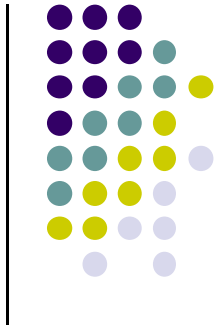
Uma transação compreende operações de **ler_item e gravar_item.**

Ex:

ler_item(X)	escrever_item(X)
X=X-N	X=X+M
escrever_item(X)	escrever_item(X)
ler_item(Y)	
Y=Y+N	
escrever_item(Y)	
Transação T1	Transação T2

O **controle de concorrência** e os **mecanismos de restauração** estão relacionados com os **comandos de acesso a banco de dados de uma transação.**

Transações e controle de concorrência



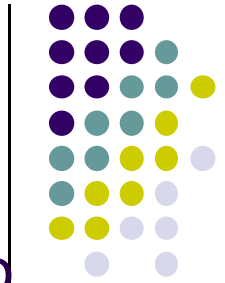
Por que o controle de concorrência é necessário?
Para evitar os problemas:

Atualização perdida – duas transações acessam os mesmos itens do banco com operações intercaladas, tornando os valores do banco incorretos. Por ex:

Transação T1	Transação T2
ler_item(X)	
X=X-N	
	ler_item(X)
	X=X+M
escrever_item(X)	
ler_item(Y)	
	escrever_item(X)
Y=Y+N	
escrever_item(Y)	

Tempo

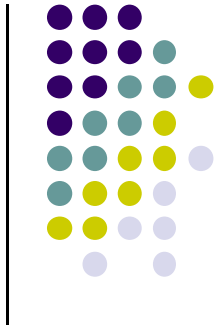
Transações e controle de concorrência



Problema: T2 lerá o valor de X antes de T1 mudá-lo no banco

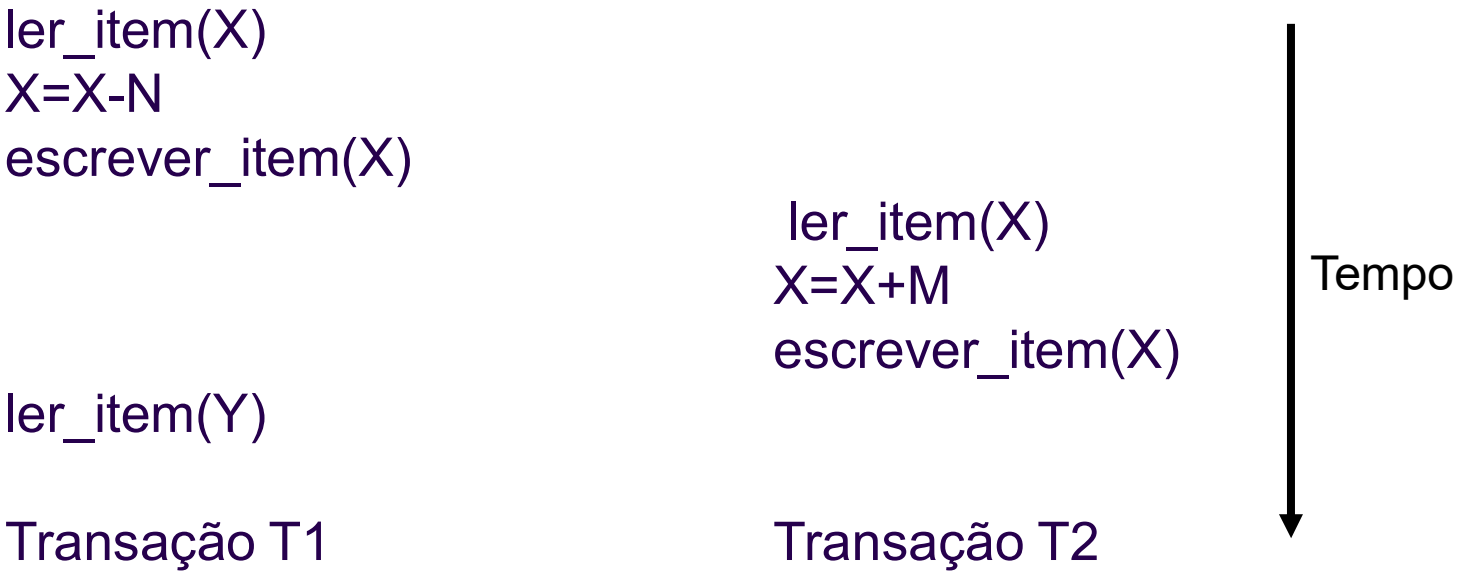
Transação T1	Transação T2
ler_item(X)	
X=X-N	
	ler_item(X)
	X=X+M
escrever_item(X)	
ler_item(Y)	
	escrever_item(X)
Y=Y+N	
escrever_item(Y)	

Transações e controle de concorrência

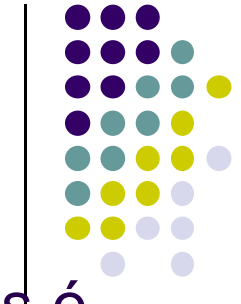


Por que o controle de concorrência é necessário?
Para evitar os problemas:

Leitura suja – uma transação atualiza o banco de dados e falha em seguida por alguma razão. Por ex:



Transações e controle de concorrência

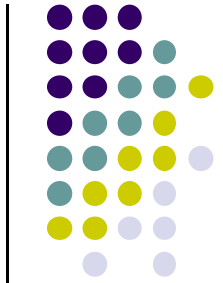


Problema: O valor lido em T2 foi atualizado em T1 mas é considerado sujo porque a transação T1 não foi efetivada.

Transação T1	Transação T2
ler_item(X)	
X=X-N	
escrever_item(X)	
	ler_item(X)
	X=X+M
	escrever_item(X)
ler_item(Y)	

Tempo

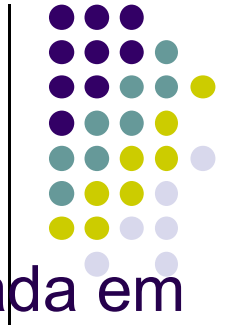
Transações e controle de concorrência



Por que o controle de concorrência é necessário?
Para evitar os problemas:

Atualização temporária – se uma transação aplicar uma função agregada para sumarizar o número de registros enquanto outra estiverem atualizando estes registros, a função agregada vai calcular valores diferentes (antes e depois da atualização). Por ex:

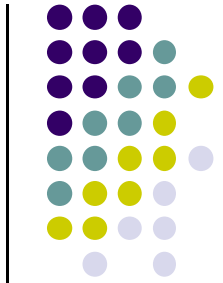
Transações e controle de concorrência



Problema: O resultado de T3 não contabilizará N (somada em T1)

Transação T1	Transação T3
	surr=0
	ler_item(A)
	sum=sum+A
ler_item(X)	
X=X-N	
escrever_item(X)	
	ler_item(X)
	sum=sum+X
	ler_item(Y)
	sum=sum+Y
ler_item(Y)	
Y=Y+N	
escrever_item(Y)	

Transações e controle de concorrência



Por que o controle de concorrência é necessário?
Para evitar os problemas:

Leitura sem repetição– ocorre quando a transação T lê um item duas vezes e entre as duas leituras o item foi mudado. Portante, T terá lido valores diferentes.

Transações

Uma transação compreende operações de **ler_item e gravar_item.**

Ex:

ler_item(X)	escrever_item(X)
X=X-N	X=X+M
escrever_item(X)	escrever_item(X)
ler_item(Y)	
Y=Y+N	
escrever_item(Y)	
Transação T1	Transação T2

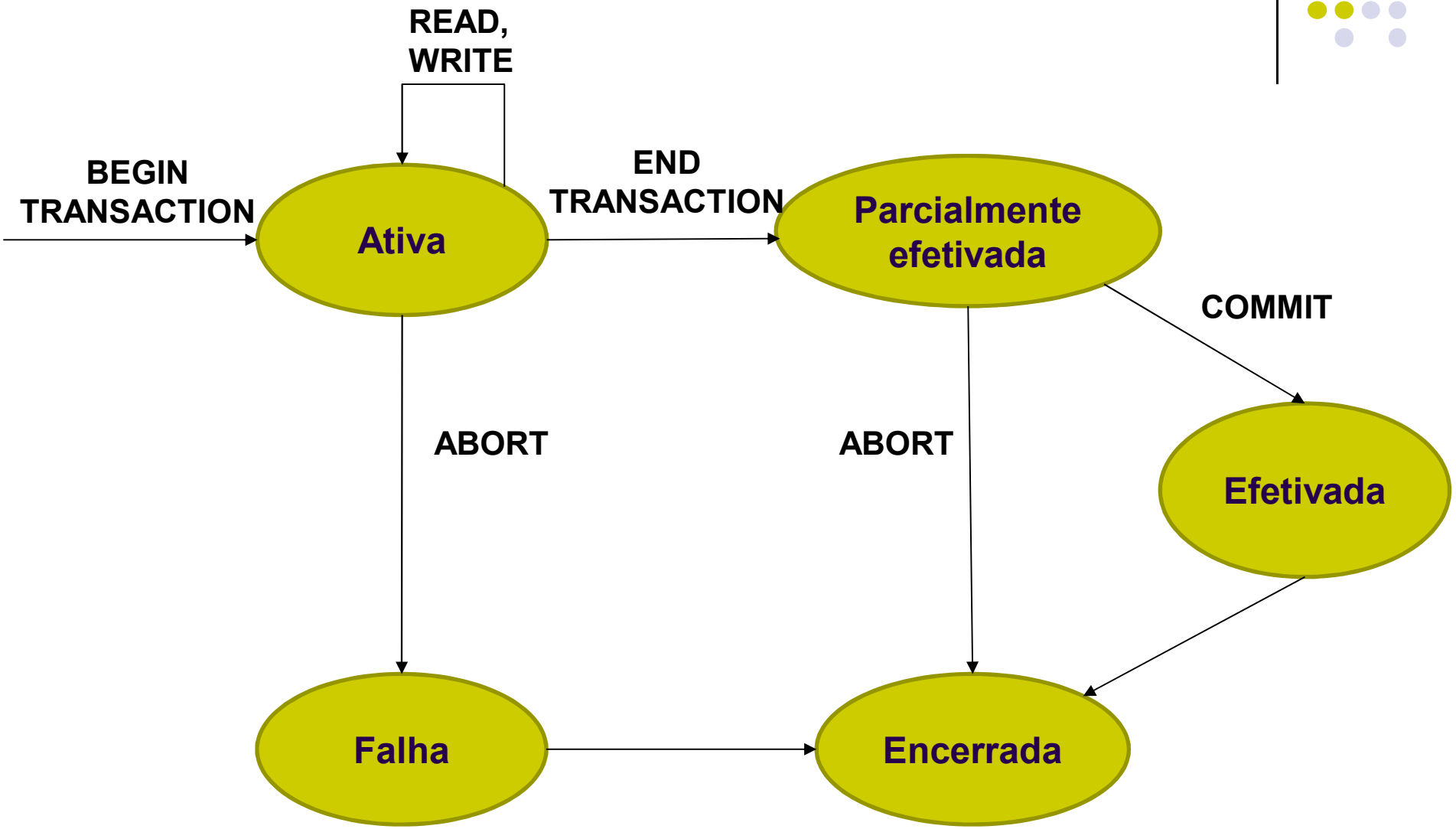
O **controle de concorrência** e os **mecanismos de restauração** estão relacionados com os **comandos de acesso a banco de dados de uma transação.**

Transações e restauração

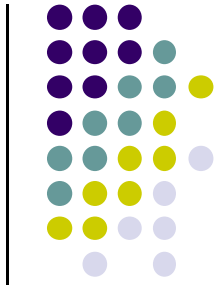
Por que a restauração é necessária?
Porque podem ocorrer falhas. Geralmente as falhas são classificadas como de transação, sistema ou mídia. São elas:

1. O computador falha - crash de sistema, falha de hardware, por ex: memória.
2. Erro de transação ou sistema - overflow de inteiro, divisão por zero, erros de lógica na programação, etc.
3. Erros locais ou condições de exceção na transação – dados não encontrados por ex.
4. Imposição do controle de concorrência – deadlock em transações, por exemplo.
5. Falha de disco – perda de dados em blocos do disco ou falha no cabeçote de leitura e gravação.
6. Problemas físicos e catástrofes – falta de energia, fogo, furto, sabotagem, sobregravação de dados em disco ou fita por falha do operador, entre outras.

Estados da Transação



O Log do Sistema (Registro de Ocorrências)

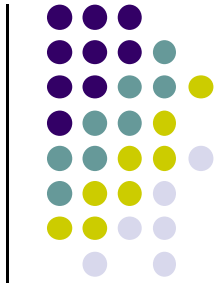


Para poder se recuperar de falhas o sistema mantém um log das transações.

O log é mantido em disco, para não ser afetado por qualquer tipo de falha (exceto as falhas de disco)

Transações falhas ou interrompidas podem ser reiniciadas mais tarde, automaticamente ou re-submetidas pelo usuário.

O Log do Sistema (Registro de Ocorrências)



Registros do log:

- **[start transaction T]**
  Transação T inicia execução
- **[escrever_item, T, X, valor antigo, valor novo]:**
  Transação T mudou o valor de X
- **[ler item, T, X]**
  Transação T leu o valor de X
- **[commit T]**
  Transação T completada com sucesso, seus efeitos podem ser efetivados no banco de dados
- **[abort T]**
  Transação T foi interrompida

O Log do Sistema (Registro de Ocorrências)

Se o sistema entrar em colapso podemos recuperar o banco de dados para um estado consistente examinando o log.

Como o log contém um registro para cada operação write que altera o valor de algum item do banco, é possível desfazer (undo) o efeito dessas operações, voltando no log e alterando as operações de write para os valores antigos.

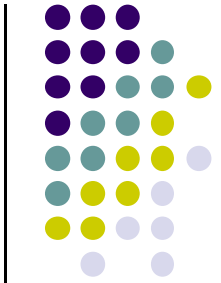
Pode ser necessário refazer (redo) as operações do log automaticamente (SGBD) ou por re-submissão do usuário.

Ponto de efetivação de uma transação (commit point)

A transação alcança o ponto de efetivação quando todas operações feitas no banco foram bem sucedidas e seus efeitos foram gravados no log.

Após alcançar o ponto de efetivação é gravado no log um registro [commit T]. Se ocorrer uma falha durante uma transação deve-se procurar no log um registro [start_transaction T] sem o correspondente registro de commit. Essas transações devem sofrer rollback para que seus efeitos sejam desfeitos no processo de restauração.

Definindo Planos de execução serializáveis



Para um plano de execução ser serializável é necessário que:

1. Sejam executadas todas as operações da transação T1 (em sequência) seguidas por todas as operações da transação T2 (em sequência).

2. Sejam executadas todas as operações da transação T2 (em sequência) seguidas por todas as operações da transação T1 (em sequência).

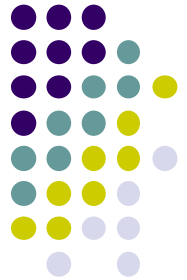
3. Um plano S com n transações é serializável se ele for equivalente a um plano serial com as mesmas transações.

Testando o conflito de serialidade em um plano

Para verificar se um plano de execução é serializável basta construir um grafo de precedência da seguinte forma:

1. Para cada transação Ti participante de um plano S, criar um nó rotulado Ti no grafo de precedência.
2. Para cada caso em S em que Tj executar um ler_item(X) depois que uma transação Ti executar um escrever_item(X), criar uma seta (Ti->Tj) no grafo.
3. Para cada caso em S em que Tj executar um escrever_item(X) depois que uma transação Ti executar um ler_item(X), criar uma seta (Ti->Tj) no grafo.
4. Para cada caso em S em que Tj executar um escrever_item(X) depois que uma transação Ti executar um escrever_item(X), criar uma seta (Ti->Tj) no grafo.
5. O plano S será serializável se não contiver ciclos.

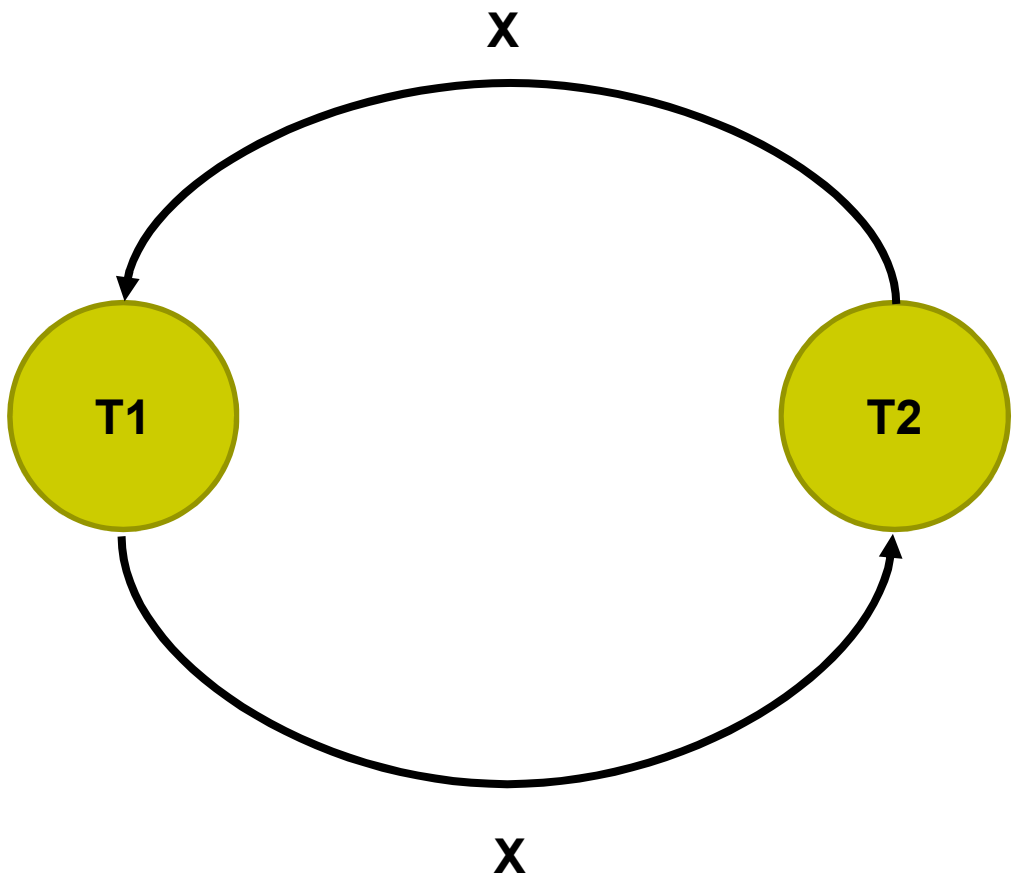
Testando o conflito de serialidade em banco incorretos. um plano



As transações T1 e T2 abaixo refletem um plano não serial:

Transação T1	Transação T2
ler_item(X)	
X=X-N	
	ler_item(X)
	X=X+M
escrever_item(X)	
ler_item(Y)	
	escrever_item(X)
Y=Y+N	
escrever_item(Y)	

Testando o conflito de serialidade em banco incorretos. um plano



Propriedades das Transações

Atomicidade: Uma transação é uma unidade atômica de processamento; ou ela será executada em sua totalidade ou não será executada.

Consistência: Uma transação é consistente se sua execução completa fizer o banco passar de um estado consistente para outro.

Isolamento: Uma transação deve ser executada como se estivesse isolada das demais, não deve sofrer interferência de quaisquer outras transações concorrentes.

Durabilidade: As mudanças aplicadas por uma transação efetivada devem persistir no banco de dados. Não devem ser perdidas em razão de falhas.

Suporte de transações em SQL

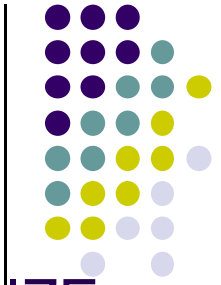
Em SQL não há transação BEGIN_TRANSACTION explícita. O início da transação é implícito e ocorre quando declarações SQL são encontradas.

Toda transação precisa ter uma declaração explícita de término: COMMIT ou ROLLBACK.

Toda transação tem determinadas características que são especificadas pela declaração SET TRANSACTION em SQL. Essas características são: o modo de acesso, o tamanho da área de diagnóstico e o nível de isolamento.

Modos de acesso: READ_ONLY ou READ_WRITE. O padrão é READ_WRITE a menos que seja especificado o nível de isolamento READ UNCOMMITED, quando será assumido READ_ONLY.

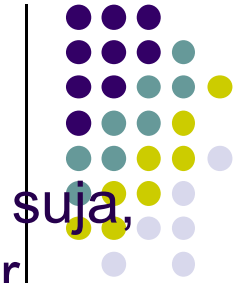
Suporte de transações em SQL



A opção **tamanho da área de diagnóstico**, DIANOSTIC SIZE n, indica o número de condições que podem ser manipuladas simultaneamente na área de diagnóstico e fornecem informações sobre as condições de execução (erros ou exceções) ao usuário ou ao programa.

A opção **nível de isolamento** é especificada pela declaração ISOLATION LEVEL <isolamento> onde o valor de isolamento pode ser:
READ UNCOMMITED (leitura não efetivada),
READ COMMITED (leitura efetivada),
REPEATABLE READ (leitura repetível) ou
SERIALIZABLE (serializável).
O nível padrão é SERIALIZABLE, entretanto, alguns sistemas tem como padrão o nível READ COMMITED.

Suporte de transações em SQL



O nível SERIALIZABLE não permite violações do tipo leitura suja, leitura não repetível e leitura fantasma. Se uma transação for executada em um nível de isolamento mais baixo que o SERIALIZABLE podem ocorrer as 3 violações.

Leitura suja: Uma transação T1 pode ler uma atualização ainda não efetivada de T2. Se T2 falhar ou for abortada, T1 lerá um valor incorreto.

Leitura não repetível: Uma transação T1 pode ler um valor em uma tabela. Se outra transação T2 atualizar este valor e T1 lê-lo novamente, terá um valor diferente.

Leitura fantasma: Uma transação T1 pode ler um conjunto de linhas sobre uma tabela, baseada em alguma condição na cláusula WHERE SQL. Se T2 inserir nova linha que também satisfaça a condição e se T1 for repetida, ela verá um fantasma, uma linha que não existia anteriormente.

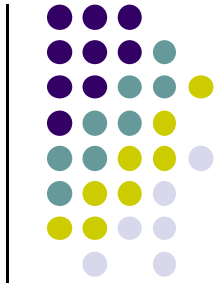
Suporte de transações em SQL

O nível SERIALIZABLE não permite violações do tipo leitura suja, leitura não repetível e leitura fantasma. Se uma transação for executada em um nível de isolamento mais baixo que o SERIALIZABLE podem ocorrer as 3 violações:

Nível de isolamento	Leitura suja	Não repetível	Leitura fantasma
Leitura não efetivada	SIM	SIM	SIM
Leitura efetivada	NÃO	SIM	SIM
Leitura Repetitiva	NÃO	NÃO	SIM
Serializável	NÃO	NÃO	NÃO

Perguntas?

Perguntas?

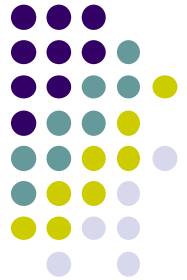


1. Para que serve o log de transações?

2. Por que é necessário o módulo do SGBD de controle de concorrência? Dê 2 exemplos de problemas que podem ocorrer.

3. Qual é objetivo das técnicas de controle de concorrência no contexto de execução de transações concorrentes em um SGBD? O que se pretende aumentar/maximizar com tais controles? Que problemas podem existir caso tais controles não sejam utilizados?

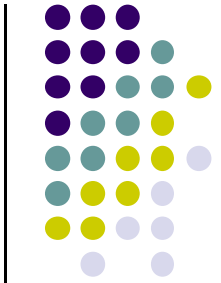
Perguntas?



4. Dadas as transações T1, T2, T3 abaixo, testar o conflito de serialidade. Existe plano serial equivalente?

T1	T2	T3
		ler_item(Y)
		ler_item(Z)
Ler_item(X)		
Escrever_item(X)		
		escrever_item(Y)
		escrever_item(Z)
	ler_item(Z)	
Ler_item(Y)		
Escrever_item(Y)		
	ler_item(Y)	
	escrever_item(Y)	
	ler_item(X)	
	escrever_item(X)	

Perguntas?



5. Dadas as transações T1, T2, T3 abaixo, testar o conflito de serialidade. Existe plano serial equivalente?

T1	T2	T3
	ler_item(Z)	
	ler_item(Y)	
	escrever_item(Y)	
		ler_item(Y)
		ler_item(Z)
ler_item(X)		
escrever_item(X)		
		escrever_item(Y)
		escrever_item(Z)
	ler_item(X)	
ler_item(Y)		
escrever_item(Y)		
	escrever_item(X)	