UEA
UNIVERSIDADE
DO ESTADO DO
AMAZONAS

Controle de Transações

Banco de Dados

André Luiz do Vale Soares

Adaptações: Márcia Lima

Transações de Banco de Dados

- O que são transações em BDs ?

  – São um conjunto de instruções SQL, tratadas como uma UNIDADE, ou seja, todas as instruções são executadas ou nenhuma delas é executada, caso uma apresente algum tipo de problema.

  – Servem para garantir CONSISTÊNCIA no banco de dados.

  – Exemplo:

    –Um banco transfere dinheiro entre duas contas (saque + deposito)

    –Se qualquer uma das operações falhar, a transferência será cancelada

Transações de Banco de Dados

- Consistem de uma das seguintes instruções:
  - Instruções DML que fazem uma alteração consistente nos dados (select, insert, update, delete,....)
  - Uma instrução DDL (create, alter,...)

Modelo de transações ACID

- Atomicidade
  - Uma transação é uma unidade atômica de processamento; ou ela será executada em sua totalidade ou não será executada de modo nenhum.
  - Exemplo:
    - Transferência de valores entre contas
    - Ou o Saque+Depósito são registrados com sucesso ou nenhum dos dois será
    - Em caso de falha o BD permanece inalterado

Modelo de transações ACID

- Consistência
  - Uma transação será preservadora de consistência se sua execução completa fizer o banco de dados passar de um estado consistente para outro.
  - Exemplo:
    - Apenas dados válidos são gravados
    - A Transação não pode quebrar regras de integridade e respeita: Chaves primárias, estrangeiras e únicas
    - Transação não pode encerrar com chave duplicada

Modelo de transações ACID

- Isolamento
  - Uma transação deve ser executada como se estivesse isolada das demais.
  - Isto é, a execução de uma transação não deve sofrer interferência de quaisquer outras transações concorrentes.

Modelo de transações ACID

- Durabilidade
  - As mudanças aplicadas ao banco de dados por uma transação devem persistir no banco de dados, ou seja, as mudanças não devem ser perdidas em razão de uma falha.
  - Exemplo: Após concluída a transferência, a energia falha. Ao retornar, os dados continuam íntegros e registrados conforme o momento imediatamente anterior à falha

Serialização de Transações

- Sejam duas transações T1 e T2;
- Quando o SGBD consegue intercalar a execução de T1 e T2, de modo que a execução concorrente e a sequencial produzam os mesmos resultados, dizemos que há um escalonamento serializável de T1 e T2.

Serialização de Transações

- Quando a intercalação da execução de T1 e T2 de maneira concorrente não assegura a produção do mesmo resultado que a execução sequencial, dizemos que o escalonamento é não serializável.

Serialização de Transações

- Exemplo:

T_1	
1	Ler(X)
2	X := X + 10
3	Escrever(X)
4	Ler(Y)
5	Y := Y - 20
6	Escrever(Y)

T_2	
1	Ler(Y)
2	Y := Y + 10
3	Escrever(Y)
4	Ler(X)
5	X := X - 20
6	Escrever(X)

Serialização de Transações

- Exemplo Escalonamento Serializável:

T_1	
1	Ler(X)
2	X := X + 10
3	Escrever(X)

T_2

1	Ler(Y)
2	Y := Y + 10
3	Escrever(Y)

4	Ler(Y)
5	Y := Y - 20
6	Escrever(Y)

4	Ler(X)
5	X := X - 20
6	Escrever(X)

Serialização de Transações

- Escalonamento Não Serializável:

T_1	
1	Ler(X)
2	X := X + 10
3	Escrever(X)
4	Ler(Y)
5	Y := Y - 20
6	Escrever(Y)

T_2	
1	Ler(Y)
2	Y := Y + 10
3	Escrever(Y)
4	Ler(X)
5	X := X - 20
6	Escrever(X)

Linguagens de Transações de Banco de Dados

- TCL (Transaction Control Language)
  - São as instruções utilizadas para controlar as transações no banco de dados.
  - Podem e DEVEM ser utilizadas em aplicações de banco de dados.
  - Instruções TCL podem ser:
    - COMMIT (padrão em todos os SGBDs)
    - ROLLBACK (padrão em todos os SGBDs)
    - SAVEPOINT (disponível em alguns SGBDs, como Oracle)

UEA
UNIVERSIDADE
DO ESTADO DO
AMAZONAS

GOVERNO DO ESTADO DO
AMAZONAS

Linguagens de Transações de Banco de Dados

•TCL (Transaction Control Language)

- COMMIT: instrução TCL utilizada para confirmar no SGBD as instruções SQL enviadas até o momento.
- ROLLBACK: instrução TCL utilizada para desfazer no SGBD as instruções SQL enviadas até o momento.
- SAVEPOINT: instrução disponível em alguns SGBDs, como Oracle, que criam um ponto de referência que pode ser utilizado para criar uma sub-transação, manipulável por uma instrução “rollback”.

• Uma instrução “COMMIT” ou “ROLLBACK” sempre finaliza uma transação no banco de dados. Qualquer outra instrução SQL que venha após uma das duas instruções TCL, implica no início de uma NOVA transação no banco de dados.

Linguagens de Transações de Banco de Dados

Linguagens de Transações de Banco de Dados

- Transactions Behavior:
  - Transações garantem a consistência e intregridade do banco de dados
  - Todas as modificações da transação são "temporárias"
  - Modificações são "persistidas" apenas após o Commit
  - A qualquer momento (Antes do commit) as modificações podem ser canceladas através de um Rollback
  - Todas as operações são executadas como uma unidades (ou nenhuma será)

UEA
UNIVERSIDADE
DO ESTADO DO
AMAZONAS

Transações de Banco de Dados

–Começa quando for executada a primeira instrução SQL executável

–Termina com um dos seguintes eventos:

- •COMMIT ou ROLLBACK é emitida
- •Instrução DDL é executada (commit automático)
- •O usuário sai (COMMIT automático)
- •O sistema cai (ROLLBACK automático)

Linguagens de Transações de Banco de Dados

- TCL
  - Exemplo 1:

```
update T1 set C1 = 'A' where C1 is null;
Insert into T2 values (12,sysdate);
rollback;
Insert into T3 values ('XYZ',450.12,sysdate);
commit;
```

Esta instrução TCL "cancela" as instruções DML enviadas anteriormente, finalizando a transação.

Esta instrução TCL confirma a execução de todas as instruções da transação.

Ao fim, apenas a tabela T3 sofrerá a inclusão de uma nova tupla.

Linguagens de Transações de Banco de Dados

- TCL
  - Exemplo 2:

update T1 set C1 = 'A' where C1 is null;

savepoint a

Insert into T2 values (12,sysdate);

savepoint b;

Insert into T3 values ('XYZ',450.12,sysdate);

Neste ponto, você pode emitir TCLs do tipo:

rollback to a; : as inclusões em T2 e T3 serão desfeitas. Assim, pode-se confirmar a atualização em T1 (commit) ou cancelá-la (rollback)

rollback to b; : a inclusão em T3 será cancelada. Assim, pode-se confirmar a atualização em T1 e inclusão em T2 (commit) ou cancelás-la (rollback)

rollback; : Todas as operações são canceladas

commit; : Todas as operações são confirmadas e os "savepoints" são então removidos.

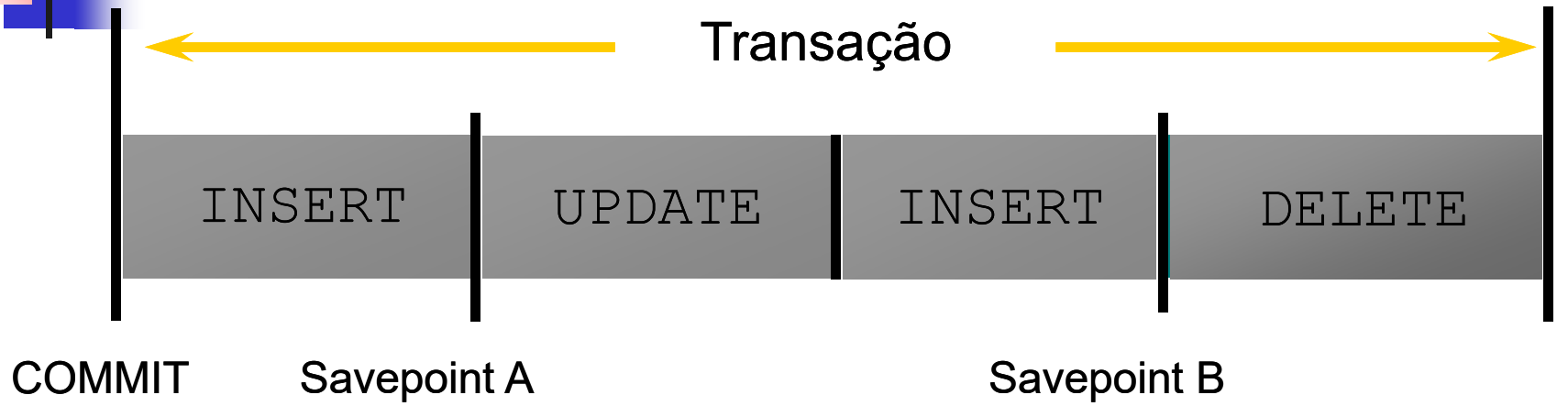
Vantagens das Instruções COMMIT e ROLLBACK

–Garantir consistência de dados

–Visualizar alterações nos dados antes de fazer as alterações permanentemente

–Agrupar operações relacionadas logicamente

Controlando Transações



ROLLBACK para Savepoint B

ROLLBACK para Savepoint A

ROLLBACK

Processando Transações Implícitas

–Um commit automático ocorre sob as seguintes circunstâncias:

- A instrução DDL é emitida
- A instrução TCL é emitida
- A saída normal do SQL*Plus, sem emitir explicitamente COMMIT ou ROLLBACK

–Um rollback automático ocorre quando há uma finalização anormal do SQL*Plus ou queda do sistema.

Estado dos Dados Antes de COMMIT ou ROLLBACK

–O estado anterior dos dados pode ser recuperado.

–O usuário atual pode revisar os resultados das operações DML usando a instrução SELECT.

–Outros usuários *não poderão* ver os resultados das instruções DML do usuário atual.

–As linhas afetadas são *bloqueadas*, outros usuários não poderão alterar os dados dentro das linhas afetadas.

Estado dos Dados Após COMMIT

–As alterações nos dados são feitas permanentemente no banco de dados.

–O estado anterior dos dados é perdido permanentemente.

–Todos os usuários podem ver os resultados.

–As linhas afetadas são desbloqueadas, essas linhas estão disponíveis para serem manipuladas por outros usuários.

–Todos os *savepoints* são apagados.

Submetendo Dados a Commit

Fazer as alterações.

```
SQL> UPDATE   emp
  2  SET      deptno = 10
  3  WHERE    empno = 7782;
1 row updated.
```

Submeter alterações a commit.

```
SQL> COMMIT;
Commit complete.
```

Estado dos Dados Após ROLLBACK

•Descarte todas as alterações pendentes usando a instrução ROLLBACK.

–As alterações nos dados são desfeitas.

–O estado anterior dos dados é restaurado.

–As linhas afetadas são desbloqueadas.

```
SQL> DELETE FROM     employee;
14 rows deleted.
SQL> ROLLBACK;
Rollback complete.
```

Fazendo Roll Back de Alterações para um Marcador

–Crie um marcador em uma transação atual usando a instrução SAVEPOINT.

–Faça roll back do marcador usando a instrução ROLLBACK TO SAVEPOINT.

```
SQL> UPDATE...
SQL> SAVEPOINT update_done;
Savepoint created.
SQL> INSERT...
SQL> ROLLBACK TO update_done;
Rollback complete.
```

Rollback no Nível da Instrução

–Se uma única instrução DML falhar durante a execução, será feito roll back somente dessa instrução.

–O Oracle Server implementa um savepoint implícito.

–Todas as outras alterações são mantidas.

–O usuário deve finalizar as transações explicitamente usando uma instrução COMMIT ou ROLLBACK.

Bloqueando – Técnica de Locking

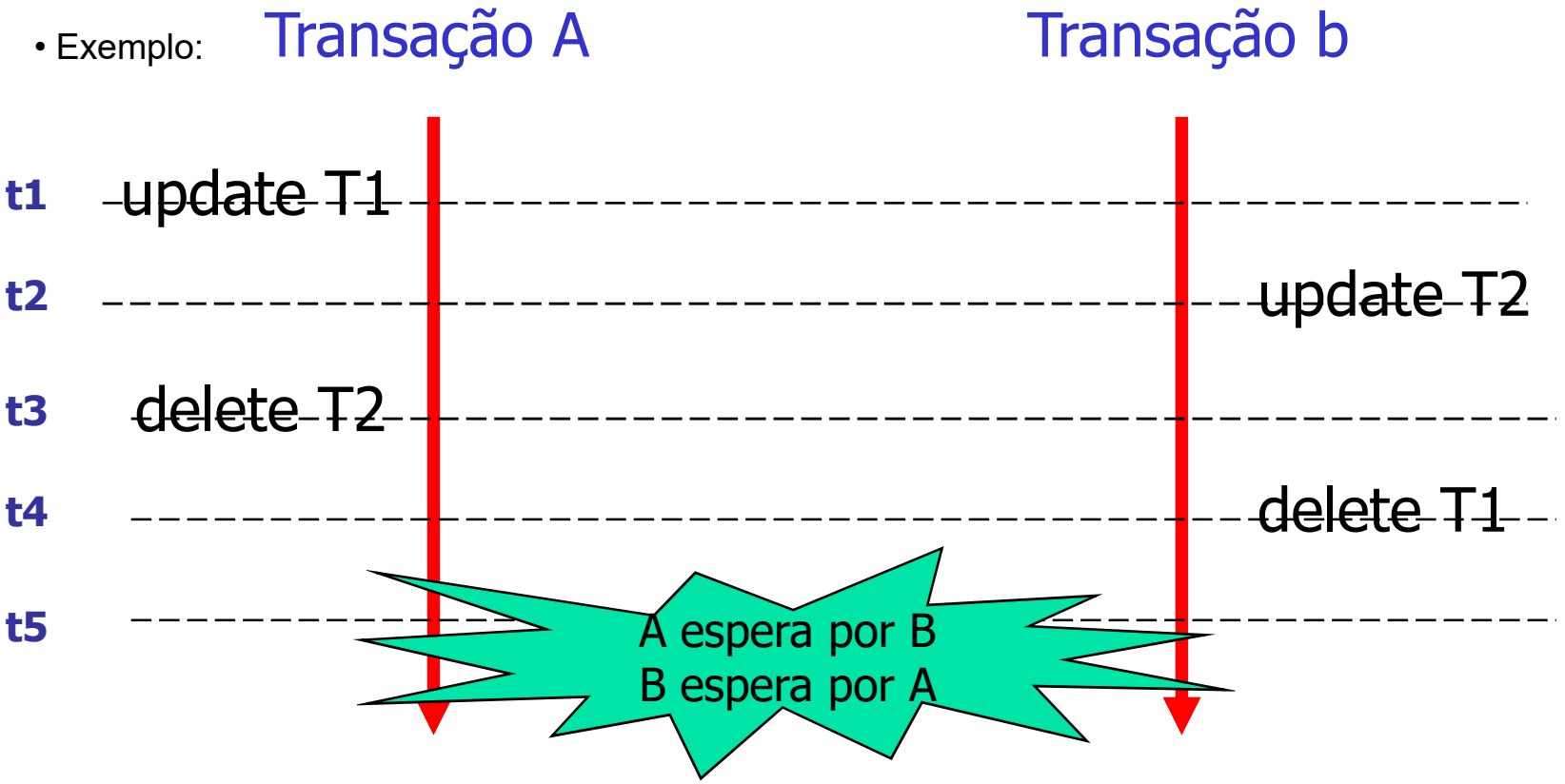
- Bloqueios em SGBDs:
  - Impedem a interação destrutiva entre transações simultâneas
  - Não requerem ação do usuário
  - São mantidos durante a duração da transação

Deadlock

- Ocorre quando:
  - Duas ou mais transações bloqueiam uma à outra permanentemente, sendo que cada uma tem o bloqueio de um recurso, que a outra transação está tentando bloquear.

Deadlock

- Exemplo:



Deadlock

- A transação A não pode terminar até que a transação B termine, mas a transação B está bloqueada pela transação A.
- Essa condição é também chamada de dependência cíclica: a transação A tem uma dependência da transação B, e a transação B fecha o círculo tendo uma dependência da transação A
- Ambas as transações em um deadlock esperarão indefinidamente, a menos que o deadlock seja quebrado por um processo externo.

Deadlock

•O monitor de deadlock do SGBD verifica periodicamente as tarefas que estão em um deadlock. Se for detectada uma dependência cíclica, ele escolhe uma das transação como “vítima” e a termina com um erro. Isso permite que a outra transação seja completada.

• O aplicativo com a transação que terminou com um erro pode repetir a transação, a qual normalmente é concluída depois que a outra transação em deadlock é encerrada.

UEA
UNIVERSIDADE
DO ESTADO DO
AMAZONAS

GOVERNO DO ESTADO DO
AMAZONAS

Deadlock

- Por exemplo:
  - 2 processos querem gravar em CD um documento obtido pelo scanner.
  - O processo A está usando o scanner, enquanto o processo B, que é programado diferentemente, está usando o gravador de CD.
  - Então, o processo A pede para usar o gravador de CD, mas a solicitação é negada até que o processo B o libere.
  - Porém, ao invés de liberar o gravador de CD, o processo B pede para usar o scanner.
  - Nesse momento, ambos os processos ficam bloqueados e assim ficarão para sempre.

Deadlock

- Como evitá-los ?

  – Basicamente a forma mais eficiente para evitar deadlocks é construindo aplicações que gerem transações de bancos de dados CURTAS.

  – Isso diminui a probabilidade de uma transação que requer um bloqueio exclusivo sobre certos recursos ser iniciada em um mesmo instante em que outra transação que requer os mesmos bloqueios sobre os mesmos recursos estiver em aberto.

  – Assim, transações GRANDES podem ser diminuídas através da inclusão de “COMMITs” em alguns pontos, fazendo com que as mesmas sejam DIVIDIDAS em transações menores.

  – As transações também devem ter o menor tempo possível de duração. Para isto, é necessário otimizar o tempo de resposta das instruções SQL utilizadas pelas mesmas.