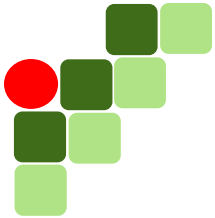
INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus Araraquara

Banco de Dados II

Cristiane Yaguinuma
cristiane.yaguinuma@ifsp.edu.br

- Processamento de transações

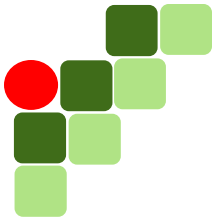


Roteiro da aula

- Conceito de transação
- Propriedades ACID
- Estrutura de uma transação
- Comandos para controle de transação
- Exercícios

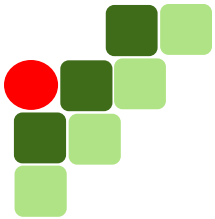
O que é uma transação?

- Uma transação é um programa em execução que forma uma unidade lógica de processamento de banco de dados
- Inclui uma ou mais operações de acesso ao banco de dados
  - inserção, exclusão, modificação ou consulta
- Utilizada para agrupar operações logicamente relacionadas e assegurar a consistência



Exemplo: sistema bancário

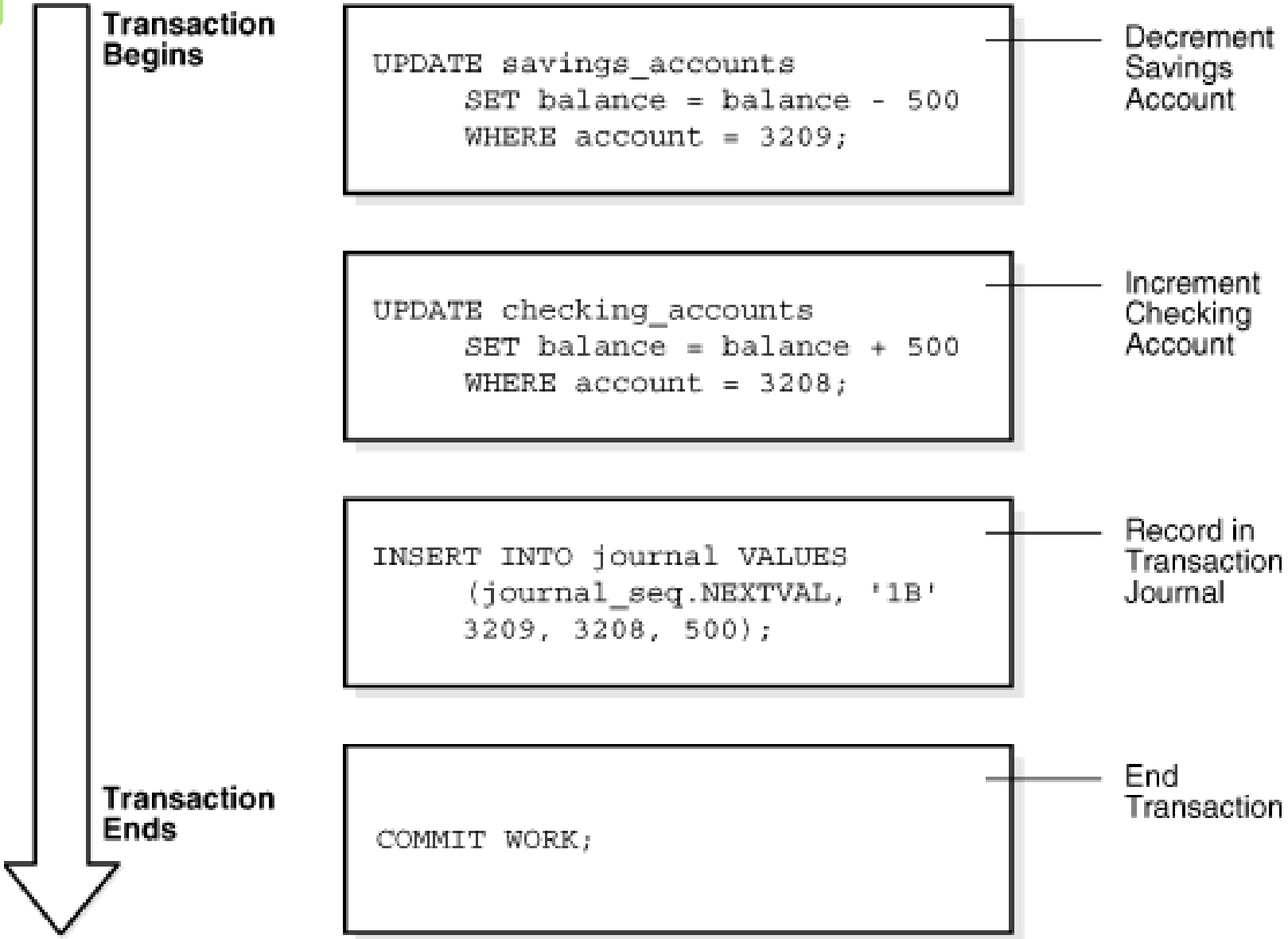
- Quando um cliente transfere $ da conta poupança para a conta corrente, a transação deve conter 3 operações:
  - Decrementar o valor da conta poupança
  - Incrementar o valor na conta corrente
  - Registrar a transação numa tabela de auditoria

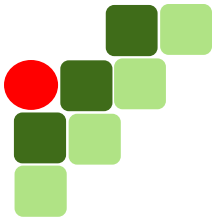


Exemplo: sistema bancário

- Para manter a consistência, as 3 operações devem ocorrer de forma atômica
- Se acontecer um problema como
  - saldo insuficiente
  - número de conta inválido ou
  - uma falha de hardware
- Então o SGBD deve desfazer a transação por inteiro de modo que o saldo de todas as contas seja correto

Exemplo: sistema bancário

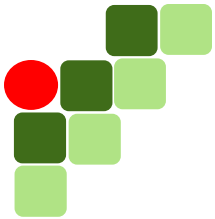




Importância das transações

- Controle de concorrência
  - Transações submetidas por diversos usuários podem interferir uma nas outras e produzir resultados incorretos

- Recuperação em caso de falhas
  - Transações interrompidas por falhas no sistema devem ser tratadas para evitar erros ou inconsistências



Propriedades ACID

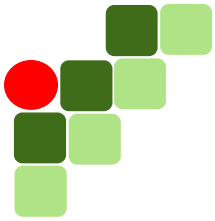
- Atomicidade
  - Uma transação é uma unidade de processamento atômica – deve ser executada em sua totalidade ou não ser realizada de forma alguma

- Preservação da Consistência
  - Uma transação deve levar o BD de um estado consistente para outro também consistente

Propriedades ACID

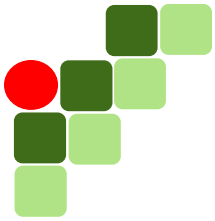
- Isolamento
  - A execução de uma transação não deve ser interferida por quaisquer outras transações que acontecem simultaneamente
  - O efeito de uma transação não é visível para outras transações até que a transação seja confirmada (COMMIT)

- Durabilidade ou permanência
  - As alterações aplicadas por uma transação confirmada devem persistir no BD



Estrutura de uma transação

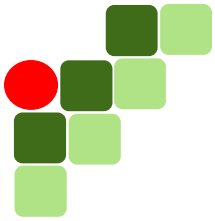
- Uma transação consiste de um ou mais comandos SQL
  - Um ou mais comandos DML que juntos constituem uma alteração atômica no BD
  - Um comando de DDL
  - Um comando DCL (GRANT, REVOKE)



Estrutura de uma transação

- Início de uma transação
  - Primeiro comando DML executado

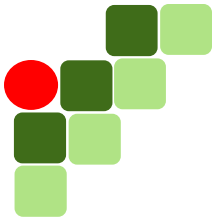
- Fim de uma transação
  - Comandos COMMIT ou ROLLBACK
  - Um comando DDL (COMMIT implícito antes e depois)
  - Fim da conexão com o SGBD
  - Falha do sistema



Controle de transação

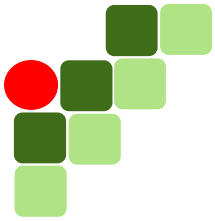
- COMMIT
  - Finaliza a transação corrente e torna permanentes todas as alterações realizadas na transação

- ROLLBACK
  - Desfaz as alterações realizadas na transação atual – faz com que todas as alterações realizadas desde o início da transação sejam descartadas



Comandos para transação em SQL

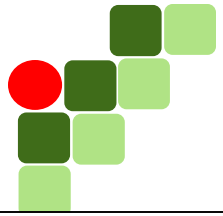
- Para tornar um conjunto de instruções SQL atômico → usar comandos para transações
  - COMMIT
  - ROLLBACK
  - SAVEPOINT
  - ROLLBACK TO SAVEPOINT



Prática (preparação)

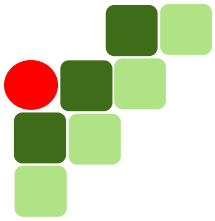
- Criar usuário USER1 que deve ter uma tabela employees contendo uma cópia do conteúdo da tabela hr.employees
  - Escreva os comandos SQL necessários

- Criar usuário USER2 com privilégios para consultar e modificar os dados de USER1.EMPLOYEES
  - Escreva os comandos SQL necessários



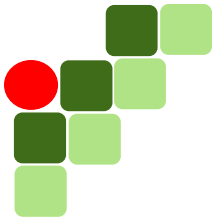
Estrutura de uma transação

```
-- executar como usuário SYSTEM
SELECT XID, STATUS, START_TIME FROM V$TRANSACTION;

-- executar como usuário user1
SELECT * FROM EMPLOYEES WHERE EMPLOYEE_ID IN (100, 110);

UPDATE EMPLOYEES          -- início da transação
SET SALARY = SALARY - 1000
WHERE EMPLOYEE_ID = 100;

UPDATE EMPLOYEES
SET SALARY = SALARY + 1000
WHERE EMPLOYEE_ID = 110;

COMMIT;                   -- fim da transação
```



Estrutura de uma transação

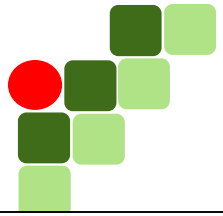
```
UPDATE employees          -- início da transação
SET salary = 20000;

-- executar como usuário SYSTEM
SELECT XID, STATUS, START_TIME FROM V$TRANSACTION;

ROLLBACK;                 -- fim da transação
```



Controle de transação

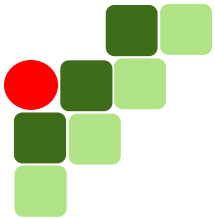
- SAVEPOINT
  - Identifica um ponto marcador em uma transação para o qual é possível realizar ROLLBACK

- ROLLBACK TO SAVEPOINT <nome_savepoint>
  - desfaz as alterações realizadas a partir do SAVEPOINT <nome_savepoint>
  - não desfaz a transação inteira
  - Não finaliza a transação



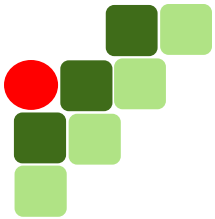
Controle de transação

```
UPDATE employees SET salary = 7000
    WHERE last_name = 'Banda';

SAVEPOINT after_banda_salary;

UPDATE employees SET salary = 12000
    WHERE last_name = 'Greene';

ROLLBACK TO SAVEPOINT after_banda_salary;

COMMIT;
```



Transações ativas

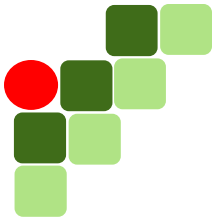
- Transação ativa: iniciada mas ainda não confirmada (COMMIT) nem cancelada (ROLLBACK)
  - As mudanças feitas são temporárias
- Antes da transação terminar, o estado dos dados é:
  - SGBD mantém os valores antigos dos dados modificados pela transação (tablespace UNDO)
  - As modificações são armazenadas em buffers para serem escritas definitivamente no BD se houver COMMIT
  - As linhas afetadas pelas modificações são bloqueadas. Outros usuários não podem alterar os dados nas linhas afetadas nem podem visualizar as mudanças não confirmadas



Transações ativas

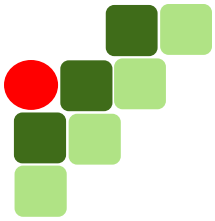
- Enquanto uma transação não termina, as linhas afetadas ficam bloqueadas (LOCK)

- Outros usuários não podem alterar dados nas linhas bloqueadas, até que a transação que as bloqueou seja terminada
  - Mas podem fazer consultas SELECT obtendo os valores anteriores à transação ativa



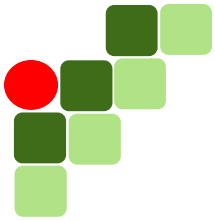
Regras para bloqueios

- Uma linha somente é bloqueada quando modificada por alguma transação
  - Quando um comando atualiza um linha, a transação adquire o LOCK somente para esta linha
  - O bloqueio em nível de linha minimiza contenção de dados



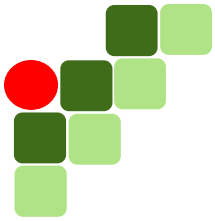
Exemplo

T	USER1	USER2
t0	UPDATE employees SET SALARY = SALARY + 1000 WHERE EMPLOYEE_ID = ‘100’;	
t1		UPDATE USER1.employees SET SALARY = SALARY + 1000 WHERE EMPLOYEE_ID = ‘101’;
t2	COMMIT;	
t3		COMMIT;



Regras para bloqueios

- Uma transação de escrita em uma linha bloqueia outra transação de escrita concorrente na mesma linha
  - Se uma transação está modificando uma linha, então o bloqueio evita que uma transação diferente modifique a mesma linha simultaneamente

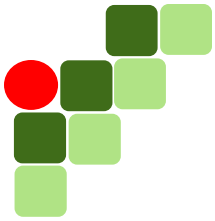


Exemplo

T	USER1	USER2
t0	UPDATE employees SET SALARY = SALARY + 1000 WHERE EMPLOYEE_ID = '100';	
t1		UPDATE USER1.employees SET SALARY = SALARY + 2000 WHERE EMPLOYEE_ID = '100';
t2	COMMIT;	
t3		COMMIT;

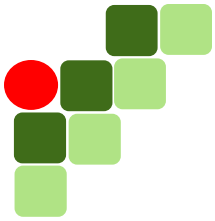
Regras para bloqueios

- Uma transação de escrita nunca bloqueia um comando de leitura
  - Quando uma linha está sendo modificada por uma transação, o SGBD permite que outras transações a consultem, considerando valores anteriores ao início da transação de escrita
  - Quando é feito o COMMIT as mudanças são visíveis para outras transações de consulta



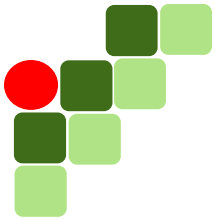
Exemplo

T	USER1	USER2
t0	SELECT EMPLOYEE_ID, SALARY FROM EMPLOYEES WHERE EMPLOYEE_ID = '100';	
t1	UPDATE employees SET SALARY = SALARY + 1000 WHERE EMPLOYEE_ID = '100';	
t2		SELECT * FROM USER1.EMPLOYEES WHERE EMPLOYEE_ID = '100';
t3	COMMIT;	
t4		SELECT * FROM USER1.EMPLOYEES WHERE EMPLOYEE_ID = '100';



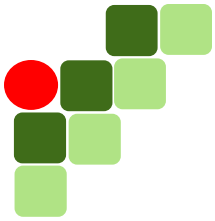
Regras para bloqueios

- Um comando de leitura nunca bloqueia uma transação de escrita
  - Como um comando de leitura de uma linha não a bloqueia, uma transação de escrita pode modificar essa linha
  - A única exceção é o comando SELECT ... FOR UPDATE, que bloqueia as linhas sendo consultadas



Exemplo

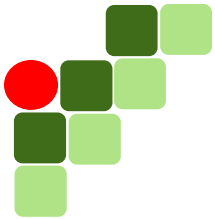
T	USER1	USER2
t0	SELECT EMPLOYEE_ID, SALARY FROM EMPLOYEES WHERE EMPLOYEE_ID = '100';	
t1		UPDATE USER1.employees SET SALARY = SALARY + 2000 WHERE EMPLOYEE_ID = '100';
t2	SELECT EMPLOYEE_ID, SALARY FROM EMPLOYEES WHERE EMPLOYEE_ID = '100';	
t3		COMMIT;
t4	SELECT * FROM EMPLOYEES WHERE EMPLOYEE_ID = '100';	SELECT * FROM USER1.EMPLOYEES WHERE EMPLOYEE_ID = '100';



LOCKs e DEADLOCKs

- Oracle automaticamente detecta DEADLOCKs e os resolve ao desfazer um dos comandos que gerou o DEADLOCK, liberando a outra transação que estava bloqueada
- Exemplo de Table 9-5

T	USER1	USER2
t0	UPDATE employees SET salary = salary*1.1 WHERE employee_id = 100;	UPDATE USER1.employees SET salary = salary*1.1 WHERE employee_id = 200;
t1	UPDATE employees SET salary = salary*1.1 WHERE employee_id = 200;	UPDATE USER1.employees SET salary = salary*1.1 WHERE employee_id = 100;



Exercício

- Considerando os comandos para controle de transação e as regras para bloqueio, analise os comandos ilustrados na tabela (próximo slide) e responda:
  1. Quantas transações estão envolvidas? Quais os comandos de início e fim de cada transação e os instantes em que ocorrem?
  2. Alguma transação fica bloqueada em algum momento? Por quê?
  3. Qual o resultado final do phone_number do employee com id = 118? Justifique sua resposta.

T	USER1	USER2
t0	SELECT employee_id, email, phone_number FROM employees WHERE employee_id=118;	
t1		SELECT employee_id, email, phone_number FROM USER1.employees WHERE employee_id=118;
t2	UPDATE employees SET phone_number='515.555.1234' WHERE employee_id=118 ;	
t3		UPDATE USER1.employees SET phone_number='515.555.0000' WHERE employee_id=118;
t4	COMMIT;	
t5		COMMIT;

Exercício extraído de Table 9-4

Referências

- Material e figuras extraídos de:
  - ELMASRI, R.; NAVATHE, S. Sistemas de Banco de Dados. 6. ed. Pearson, 2011.
  - Oracle Database Concepts
    - Transactions
    - Overview of the Oracle Database Locking Mechanism