

Laboratório de Bases de Dados



Prof. José Fernando Rodrigues Júnior

Aula 10 – Transações

Material: Profa. Elaine Parros Machado de Sousa



Transações

- **Transação**: Unidade lógica de trabalho
 - abrange um conjunto de operações de manipulação de dados que executam uma única tarefa

Conecta ao Banco de Dados

Começa transação

Operações de consulta/atualização

...

Finaliza transação

Começa transação

Operações de consulta/atualização

...

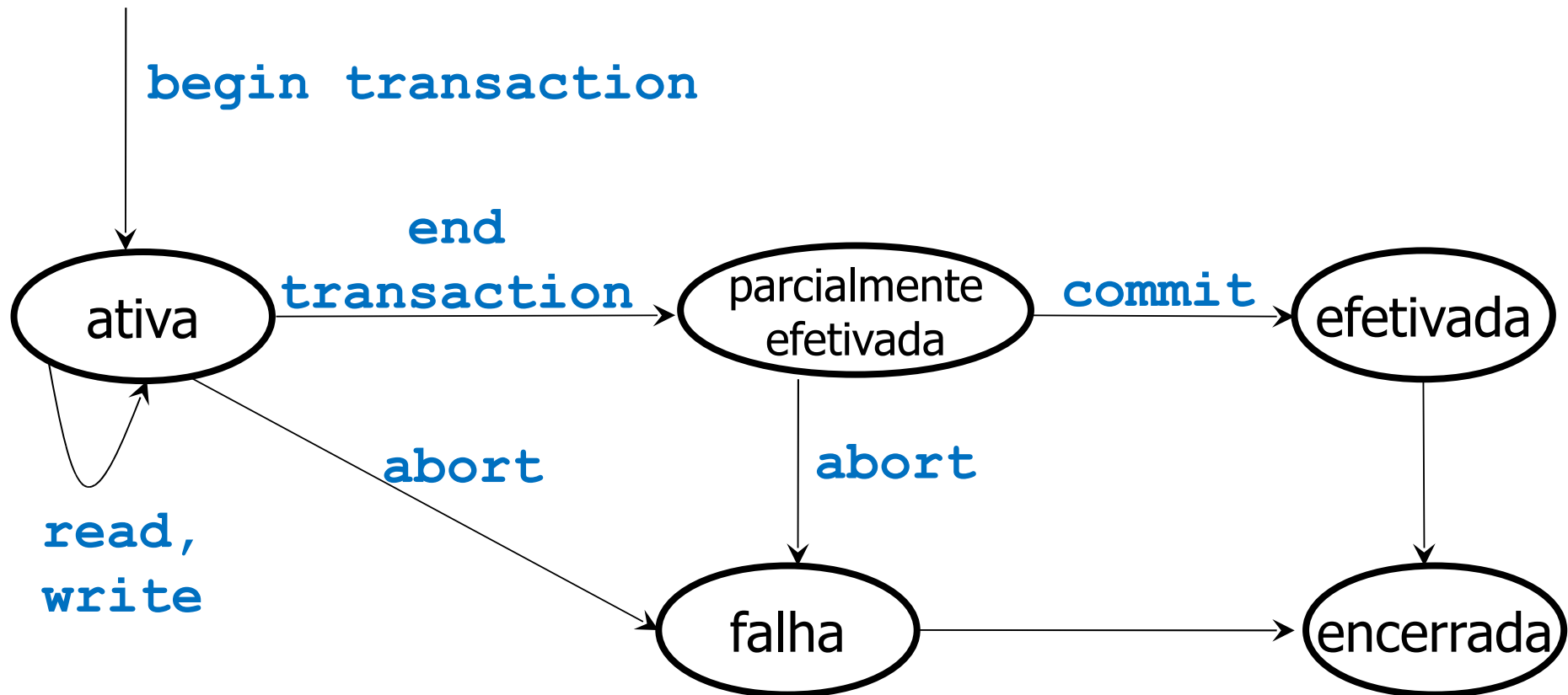
Finaliza transação

Desconecta



Transações

- Transição de Estados de Execução





Transações

- O que acontece se a energia acabar no meio de uma transação, ou se houver um problema com o disco?
 - O que acontece quando duas transações executam simultaneamente manipulando o mesmo dado?
- ⇒ O banco de dados pode ser levado a um estado inconsistente...



Falhas

| | Tipo de falha | Abrangência | Como evitar/recuperar |
|--------|-----------------------------|----------------------------------|--|
| Global | Local | Apenas a transação corrente | Registro de log/controle de concorrência |
| | De sistema (soft crash) | Todas as transações em andamento | Registro de log |
| | De meio físico (hard crash) | Todas as transações em andamento | Backup |

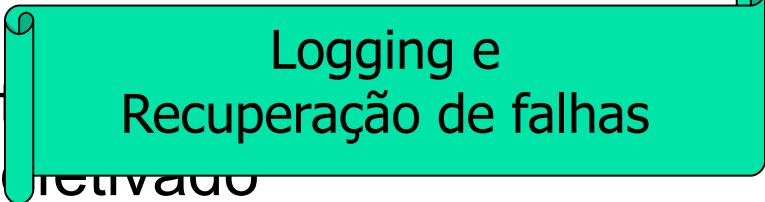
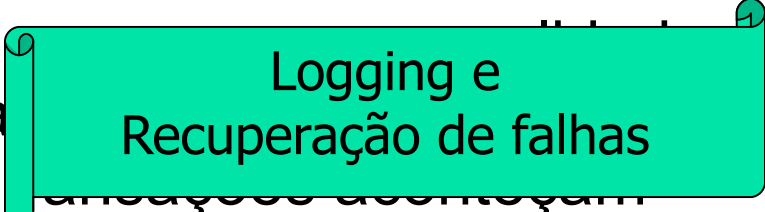


Transações – Propriedades ACID

- **Atomicidade:** todas as operações de uma transação devem ser efetivadas; ou, na ocorrência de uma falha, nada deve ser efetivado
 - “**tudo ou nada**” – não se admite parte de uma operação
- **Consistência:** transações preservam a consistência da base
 - Estado inicial consistente \Rightarrow Estado final consistente
- **Isolamento:** a maneira como várias transações em paralelo interagem (o que pode ser lido e o que pode ser escrito por cada uma) deve ser bem definida
- **Durabilidade:** uma vez consolidada (committed) a transação, suas alterações permanecem no banco até que outras transações aconteçam



Transações – Propriedades ACID

- **Atomacidade:** uma transação deve ser efetuada ou não deve ser efetuada. 
 - “**tudo ou nada**” – não se admite parte de uma operação
- **Consistência:** transações preservam a consistência da base
 - Estado inicial consistente \Rightarrow Estado final consistente
- **Isolamento:** a maneira como várias transações em paralelo interagem (o que pode ser lido e o que pode ser escrito por cada uma) deve ser bem definida
- **Durabilidade:** uma transação (committed) a transação, sua alteração permanece no banco até que outras transações aconteçam. 



Transações – Propriedades ACID

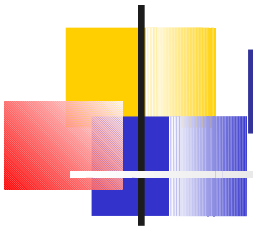
- **Atomacidade:** Uma transação deve ser efetuada ou nada deve ser efetivado.
Recuperação de falhas via log
■ “**tudo ou nada**” – não se admite parte de uma operação
- **Consistência:** Consistência da base
Controle de Concorrência via Locks
■ Estado inicial consistente \Rightarrow Estado final consistente
- **Isolamento:** Transações em paralelo interagem e o que pode ser escrito por cada uma deve ser bem definida
Controle de Concorrência via Locks
- **Durabilidade:** (committed) a transação, sua alteração ficam no banco até que outras alterações aconteçam
Recuperação de falhas via log



Controle de Concorrência

- **Execução Serial (sequencial):** diversas transações executadas em sequência
 - deixa a base de dados em estado correto e consistente
- **Execução Intercalada:** comandos de diversas transações são intercalados
 - pode levar a inconsistências

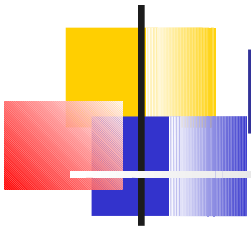
| | Isolamento | Concorrência | Chance de inconsistências |
|-------------|------------|--------------|---------------------------|
| Serial | ↑ | ↓ | ↓ |
| Intercalada | ↓ | ↑ | ↑ |



Execução Serial X Intercalada

- **Execução serial**

- estado inicial correto e consistente \Rightarrow estado final correto e consistente
- estado final pode variar dependendo da ordem das transações
 - mas **todos são estados corretos e consistentes**

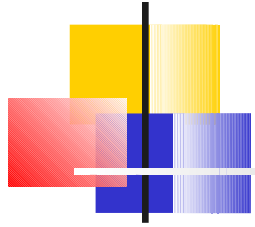


Execução Serial X Intercalada

- **Execução Intercalada**

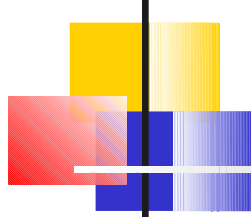
- Toda execução serial é consistente
- Mas uma execução intercalada só é **consistente se for igual ao resultado de uma execução em sequência (em ordem conhecida)**
 - esta execução é dita **serializável**

Problemas de Execução Intercalada



- Ocorrência de anomalias
 1. leitura inválida
 2. leitura não repetível
 3. leitura fantasma

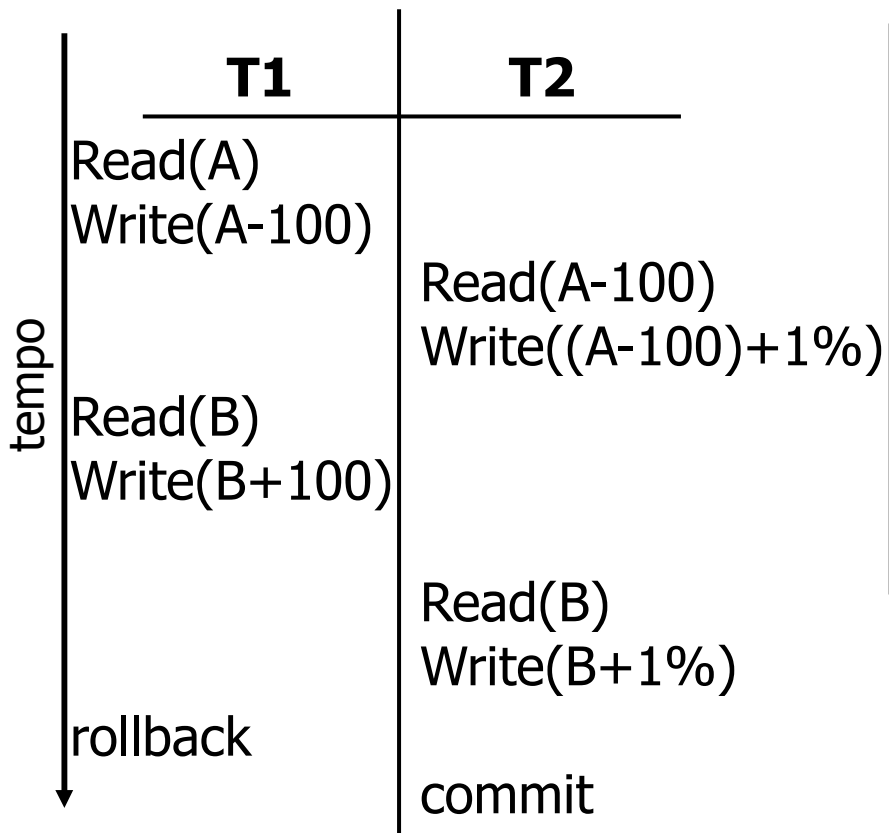
Problemas de Execução Intercalada



- 1) **Leitura inválida (*Dirty Read*):**
 - transação T' lê um dado modificado por uma transação T que ainda não terminou;
 - permite que outras transações possam ver os dados que ainda não foram completados (committed), isto é, mudanças que podem ser descartadas em seguida, por causa de uma instrução ROLLBACK WORK por exemplo.

Problemas de Execução Intercalada

■ Ex: Leitura inválida (*Dirty Read*):

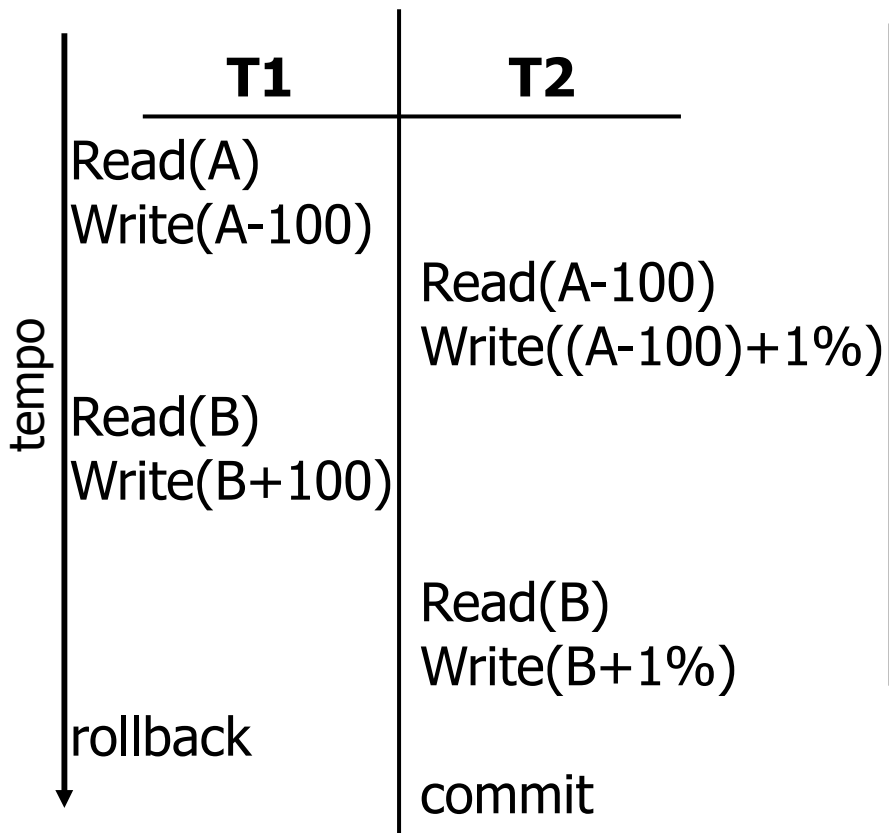


Exemplo 1:

- **Transação T_1** : transfere R\$100,00 da conta A para a conta B.
- **Transação T_2** : incrementa A e B em 1% (juros).

Problemas de Execução Intercalada

■ Ex: Leitura inválida (*Dirty Read*):



Exemplo 1:

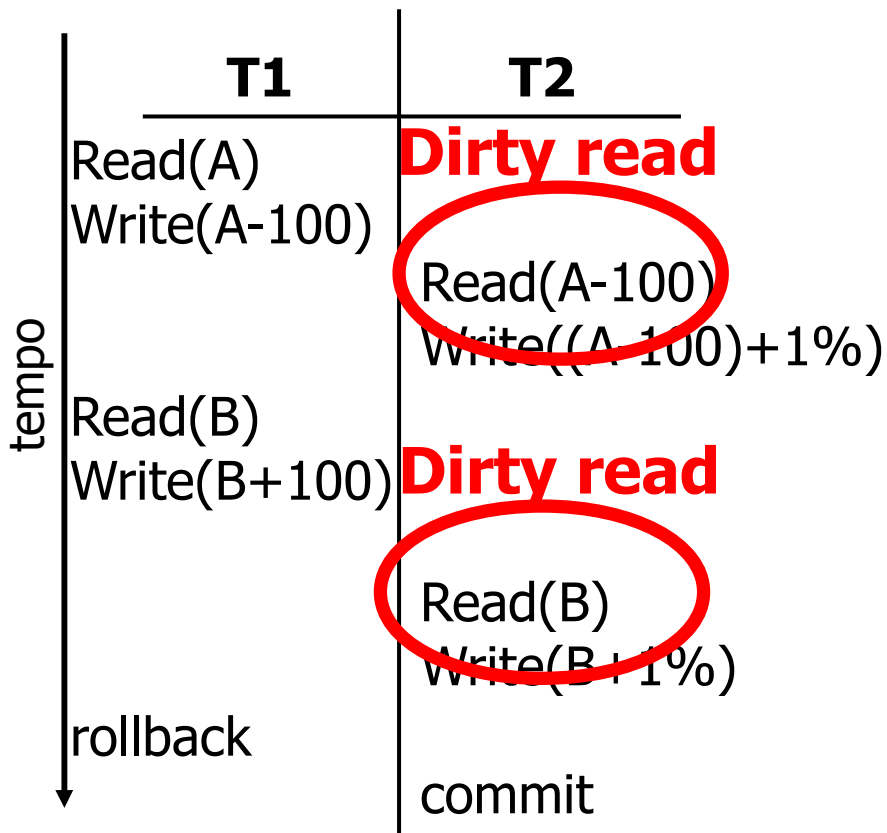
Supõe que inicialmente $A = 500$ e $B = 600$

- resultado esperado: T1 (rollback) seguido de T2 é $A = 505$ ($A+1\%$) e $B = 606$ ($B+1\%$)

- no entanto, com T1 e T2 em paralelo, o que se tem é: $A = 404$ ($A - 100 + 1\%$) e $B = 707$ ($B+100 + 1\%$)

Problemas de Execução Intercalada

■ Ex: Leitura inválida (*Dirty Read*):



Exemplo 1:

Supõe que inicialmente $A = 500$ e $B = 600$

- resultado esperado: T1 (rollback) seguido de T2 é $A = 505$ ($A+1\%$) e $B = 606$ ($B+1\%$)

- no entanto, com T1 e T2 em paralelo, o que se tem é: $A = 404$ ($A - 100 + 1\%$) e $B = 707$ ($B+100 + 1\%$)

Problemas de Execução

Interlocks

- **Ex:** **Dirty read**:

Resultado: a transferência de R\$ 100 de T1 foi sobrescrita e T2 considerou valores errados como base se cálculo de 1%. Ambos, A e B estão errados.

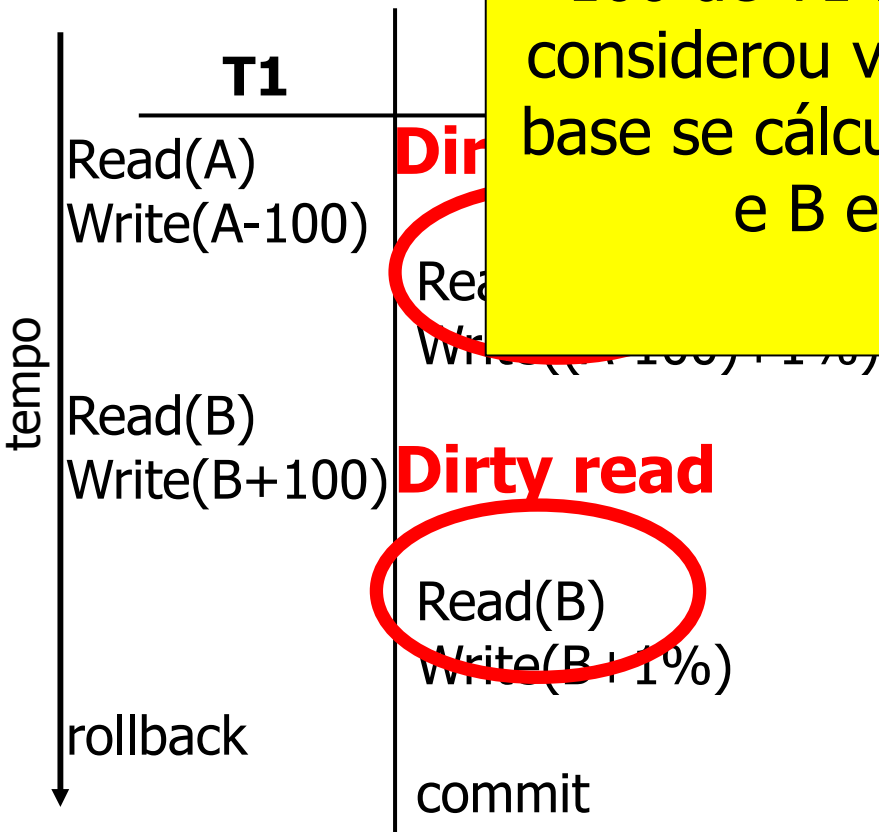
... e $A = 500$ e $B =$

... T1 (rollbacked)

... $(A+1\%)$ e $B =$

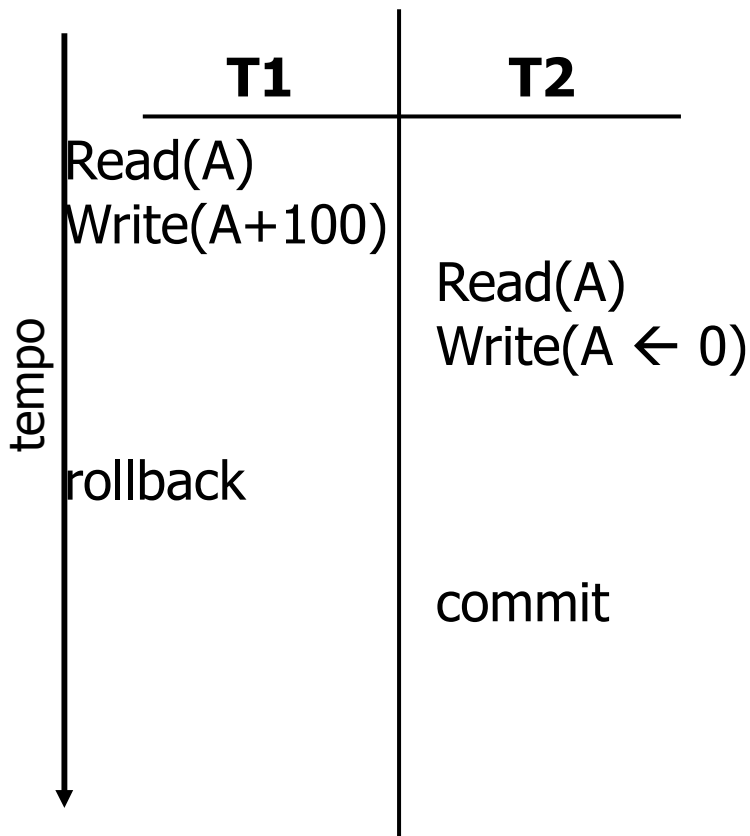
606 ($B+1\%$)

- no entanto, com T1 e T2 em paralelo, o que se tem é: $A = 404$ ($A - 100 + 1\%$) e $B = 707$ ($B+100 + 1\%$)



Problemas de Execução Intercalada

■ Ex: Leitura inválida (*Dirty Read*):



Exemplo 2:

- **Transação T_1** : deposita R\$100,00 na conta A.
- **Transação T_2** : saca tudo de A.
- T_1 é cancelada

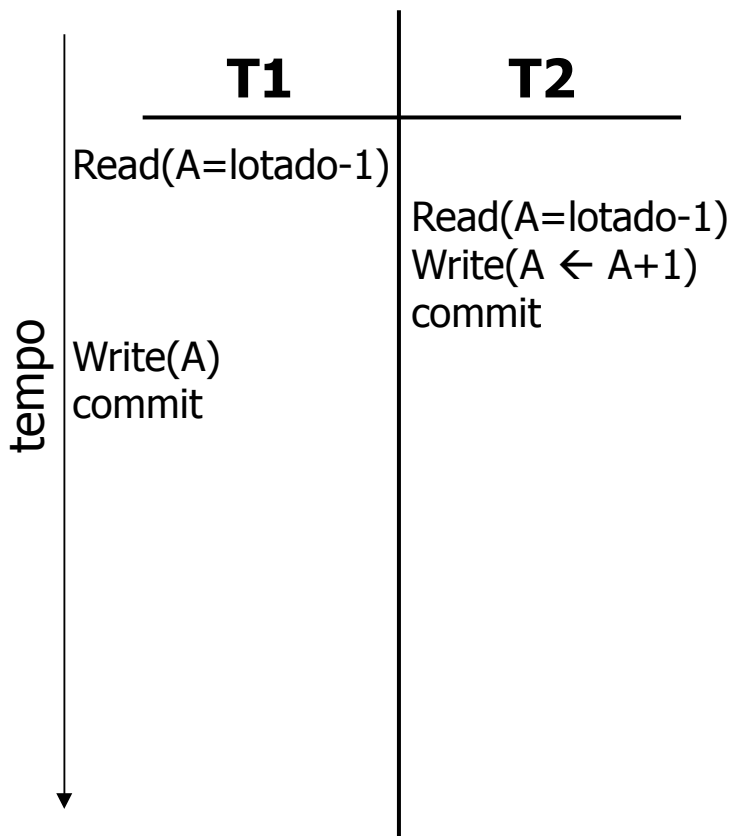


Problemas de Execução Intercalada

- **2) Leitura não repetível (*Nonrepeatable Read*):**
 - transação T lê um dado
 - esse dado é modificado por uma transação T' que começou depois de T
 - T é efetivada
 - se T' tentar reler o mesmo dado, obterá valores diferentes (*nonrepeatable read*)

Problemas de Execução Intercalada

- **Ex:** Leitura não repetível (*Nonrepeatable Read*):

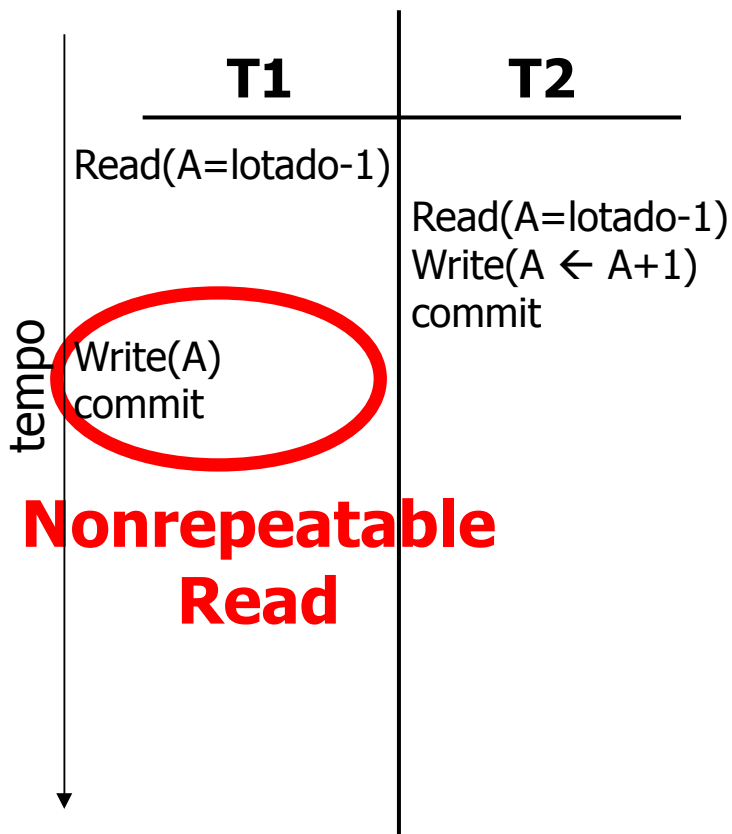


Exemplo:

- **Transação T_1** : lê reservas de um voo e verifica que há apenas um lugar disponível.
- **Transação T_2** : lê a mesma coisa
- T_2 reserva o último lugar e é efetivada.
- T_2 tenta reservar o lugar e

Problemas de Execução Intercalada

- **Ex: Leitura não repetível (*Nonrepeatable Read*):**



Exemplo:

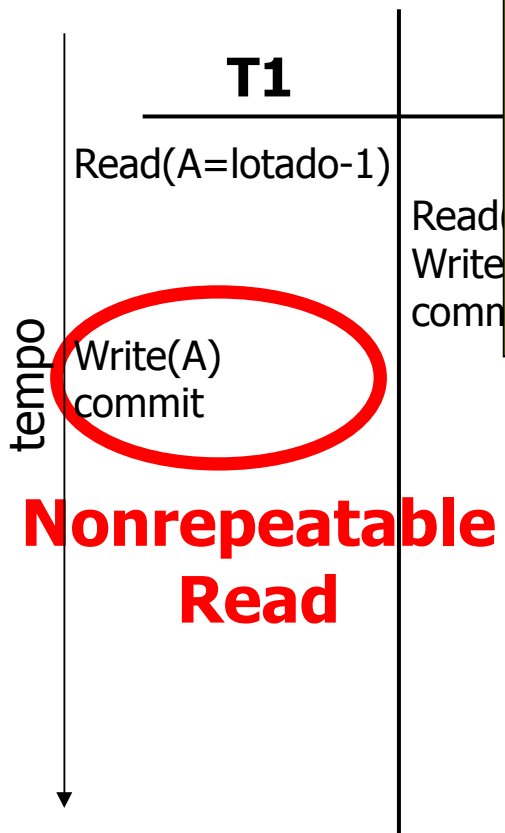
- **Transação T_1** : lê reservas de um voo e verifica que há apenas um lugar disponível.
- **Transação T_2** : lê a mesma coisa
- T_2 reserva o último lugar e é efetivada.
- T_1 tenta reservar o lugar e

Problemas de Execução

Interpretação

■ Ex: Leitura

Resultado: um usuário foi informado de que ainda havia lugares. Após preencher o cadastro, clicou confirma e recebeu um erro de que o voo estava lotado.



um lugar disponível.

- **Transação T_2** : lê a mesma coisa
- **T_2** reserva o último lugar e é efetivada.
- **T_1** tenta reservar o lugar e

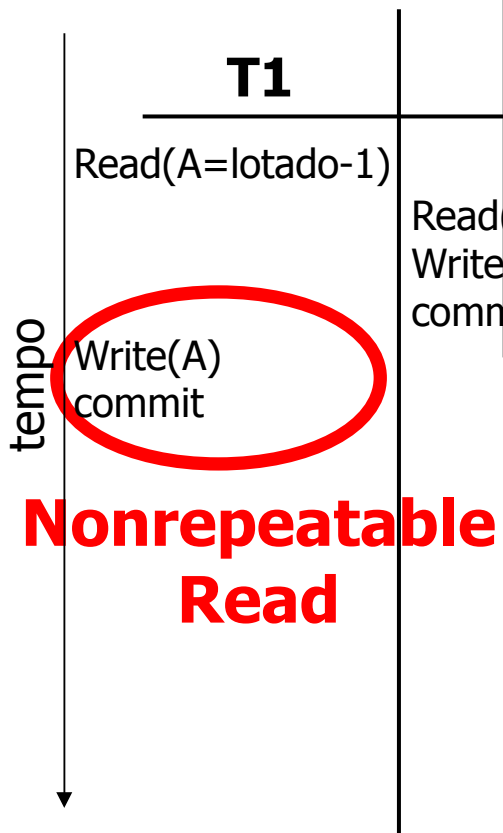
Problemas de Execução

Interpretação

■ Ex: Leitura

Repeatable read? Mas não está ocorrendo uma nova leitura, mas sim uma escrita.

Na verdade, há uma nova leitura pelo SGBD que validará a entrada dos dados antes de inserí-los.



um lugar disponível.

- **Transação T_2** : lê a mesma coisa
- **T_2** reserva o último lugar e é efetivada.
- **T_1** tenta reservar o lugar e



Problemas de Execução Intercalada

- **3) Leitura fantasma (*Phantom Read*):**
 - transação T lê um conjunto de tuplas que atendam a uma condição de consulta
 - transação T' **insere/remove/atualiza** uma tupla que atenderia a essa condição e é efetivada
 - se T refizer a mesma consulta, obterá um conjunto diferente de tuplas (*phantom read*)

Problemas de Execução Intercalada

■ 3)

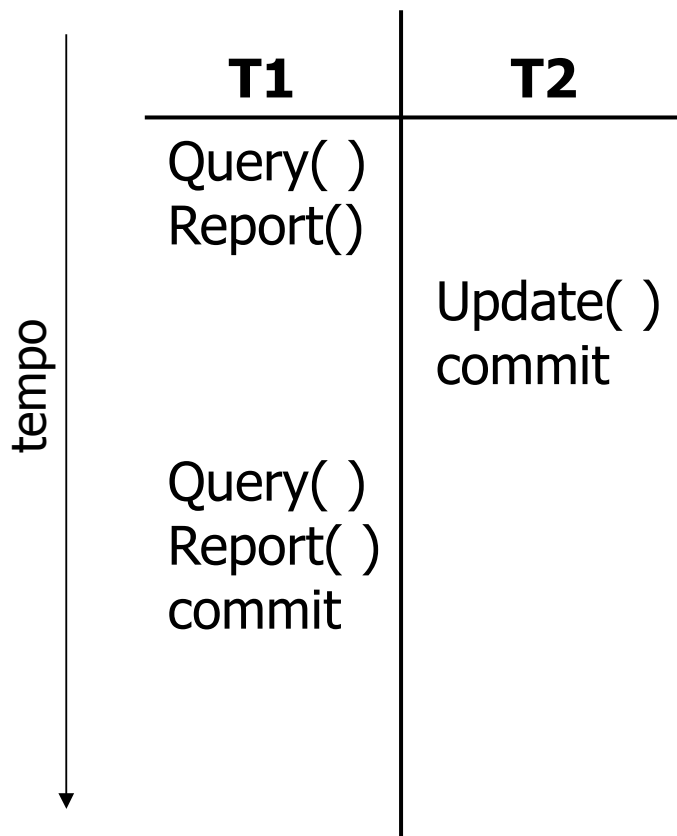
Repeatable read vs Phantom read

- t
a
■ t
C
s
c
- Repeatable read: lê valores diferentes de um mesmo dado que ainda está lá, mas foi alterado

- Phantom read: lê conjuntos de dados diferentes, sendo que um dos conjuntos possui dados que não existem no(s) outro(s) conjunto(s) – fantasmas

Problemas de Execução Intercalada

■ Ex: Leitura fantasma (*Phantom Read*):



Exemplo:

- **Transação T_1** : faz uma consulta que retorna a média geral dos alunos que têm média ponderada acima de 5.0, e gera um relatório
- **Transação T_2** : atualiza as notas de alguns alunos e é efetivada
- **T_1** refaz a consulta para gerar relatório com nro de alunos por faixa de média

→ relatórios inconsistentes



Problemas de Execução Intercalada → Isolamento

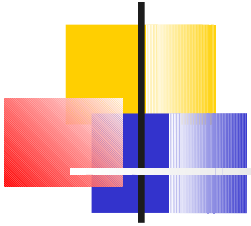
- Ocorrência de anomalias

1. leitura inválida
2. leitura não repetível
3. leitura fantasma

- Solução via isolamento em diferentes graus

- Read uncommitted
- Read committed
- Repeatable read
- Serializable

Níveis de Isolamento em SQL99



| Nível de isolamento | Anomalias que PODEM ocorrer | | |
|---------------------|-----------------------------|--------------------------|---------------------|
| | 1) Leitura inválida | 2) Leitura não repetível | 3) Leitura fantasma |
| Read uncommitted | Sim | Sim | Sim |
| Read committed | Não | Sim | Sim |
| Repeatable read | Não | Não | Sim |
| Serializable | Não | Não | Não |



Interpretação

| Nível de isolamento | Anomalias que PODEM ocorrer | | |
|--|-----------------------------|--------------------------|---------------------|
| | 1) Leitura inválida | 2) Leitura não repetível | 3) Leitura fantasma |
| Leitura mesmo do que NÃO FOI committed | Sim | Sim | Sim |
| Leitura apenas do que FOI committed | Não | Sim | Sim |
| Leitura apenas se a leitura repetida for garantida | Não | Não | Sim |
| Torna a execução equivalente à execução em série | Não | Não | Não |



Como implementar os níveis de isolamento?

→ Locks: ou travas; uma transação T determina que se uma outra transação T' precisar daquele dado – para escrita ou leitura – ela deverá esperar

| Isolamento \ Travas | Lock de escrita (write lock) | Lock de leitura (read lock) | Lock de leitura de conjunto (range lock) |
|---------------------|---------------------------------|--------------------------------|---|
| Read uncommitted | IGNORA | IGNORA | IGNORA |
| Read committed | CONSIDERA | IGNORA | IGNORA |
| Repeatable read | CONSIDERA | CONSIDERA | IGNORA |
| Serializable | CONSIDERA | CONSIDERA | CONSIDERA |



Transações em Oracle



Transação em ORACLE

- Comando **SET TRANSACTION**

SET TRANSACTION

READ ONLY | READ WRITE | ISOLATION LEVEL
{SERIALIZABLE | READ COMMITTED} |
USE ROLLBACK SEGMENT rollback_segment
NAME 'nome_da_transacao';



Níveis de Isolamento de SQL99 no ORACLE

| Nível de isolamento | Anomalias que podem ocorrer | | |
|-------------------------|---|-----------------------|------------------|
| | Leitura inválida | Leitura não repetível | Leitura fantasma |
| Read uncommitted | Nunca permitido em Oracle. | | |
| Read committed (padrão) | Não | Sim | Sim |
| Repeatable read | Não suportado especificamente (abrangido por serializable). | | |
| Serializable | Não | Não | Não |



Transações em ORACLE

- Transações explícitas **iniciam** com a cláusula SET TRANSACTION
- E **terminam**:
 - explicitamente com **commit** ou **rollback**
 - implicitamente quando um processo de usuário é finalizado
 - com sucesso – ex: disconnect (**commit**)
 - sem sucesso – ex: falha de sistema (**rollback**)
 - execução de comando DDL
- Possuem quatro possibilidades:
 - **Modo de leitura**: para transações apenas leitura
 - READ-ONLY, e READ-WRITE
 - **Modo de isolamento**: para transações com atualizações
 - READ committed, e SERIALIZABLE



Transação em ORACLE

- Comando **COMMIT**
 - termina a transação
 - torna permanente as ações da transação
 - libera os recursos bloqueados



Transação em ORACLE

- Comando **ROLLBACK**

- desfaz operações realizadas dentro de uma transação não efetivada
- *rollback* sem *savepoint*
 - desfaz todas as operações da transação
 - libera todos os recursos bloqueados
 - termina transação
- *rollback* com *savepoint*
 - desfaz operações realizadas depois do *savepoint*
 - *savepoints* depois de um ponto de *rollback* são perdidos
 - libera recursos bloqueados depois do *savepoint*
 - transação permanece ativa, **mas não efetivada**



Transações em ORACLE

- **Modo de leitura:** aplica-se a transações que não possuam comandos de atualização, APENAS LEITURA
 - **READ WRITE (padrão):** a transação que se segue considera todos os dados, inclusive o que for committed DURANTE sua execução
 - **READ ONLY:** a transação que se segue considera apenas os dados consolidados (committed) no seu início – ela trabalha com o snapshot do banco no momento em que é definida



Transações em ORACLE

- A transação abaixo considera apenas o snapshot da base iniciado no momento quando o comando SET TRANSACTION foi declarado

```
DECLARE
```

```
    daily_order_total  NUMBER(12,2);
```

```
    weekly_order_total NUMBER(12,2);
```

```
    monthly_order_total NUMBER(12,2);
```

```
BEGIN
```

```
    SET TRANSACTION READ ONLY NAME 'Calculate Order Totals';
```

```
        SELECT SUM (order_total) INTO daily_order_total FROM orders
```

```
            WHERE order_date = SYSDATE;
```

```
        SELECT SUM (order_total) INTO weekly_order_total FROM orders
```

```
            WHERE order_date = SYSDATE - 7;
```

```
        SELECT SUM (order_total) INTO monthly_order_total FROM orders
```

```
            WHERE order_date = SYSDATE - 30;
```

```
    COMMIT; -- ends read-only transaction
```

```
END;
```

Transações em ORACLE

- Transações READ-ONLY aceitam apenas os seguintes tipos de comandos:

- SELECT INTO
- OPEN
- FETCH
- CLOSE
- LOCK TABLE
- COMMIT
- ROLLBACK

- Não permitem INSERT, UPDATE e DELETE

```
WHERE order_date = SYSDATE - 7;
```

```
SELECT SUM (order_total) INTO monthly_order_total FROM orders
```

```
WHERE order_date = SYSDATE - 30;
```

```
COMMIT; -- ends read-only transaction
```

```
END;
```

momento



Transações em ORACLE

- Modo de isolamento: para transações com atualizações
 - READ committed (padrão):
 - antes de uma operação, a transação aguarda até que quaisquer tuplas sendo atualizadas sejam liberadas e prossegue
 - a transação “vê” apenas dados consolidados (committed) antes do início de uma dada **operação**
 - SERIALIZABLE:
 - caso uma tupla seja alterada após o início da transação serializable, se houver uma tentativa de alteração desta tupla, será jogada a exceção:
ORA-08177: Can't serialize access for this transaction.
 - ou seja, o Oracle informa que não é capaz de tornar a concorrência semelhante a um processamento em série
 - a transação “vê” apenas dados modificados pela própria transação e dados efetivados antes do início da transação



Transações em ORACLE

- Modo de isolamento: para transações com atualizações
 - READ committed (padrão):
 - antes de uma operação, a transação aguarda até que quaisquer tuplas sendo

Idéia de Snapshot dos dados antes de uma operação – vários snapshots do banco.

- Serializable
 - caso uma tupla seja alterada após o início da transação serializable, se houver uma tentativa de alteração desta tupla, será jogada a exceção:
ORA-08177: Can't serialize access for this transaction.

Idéia de Snapshot dos dados antes de uma transação inteira – um único snapshot.



Transações em ORACLE

- Em caso de erro “**ORA-08177**: Can't serialize access for this transaction” deve-se fazer uma opção:
 - **Desfazer (rollback)** toda a transação, terminando assim a transação
 - **Validar (commit)** o que foi feito até aquele ponto, terminando assim a transação
 - **Continuar** a transação com outras operações adicionais – usando o recurso de ***savepoint***

Transações em ORACLE

| | | T2 | | | | |
|----|--------------------------|--------|-------------------------------------|-------------------------------------|-------------------------------------|----------------------------|
| | | SELECT | INSERT | UPDATE | DELETE | |
| T1 | T1 INSERT não finalizado | Não vê | Espera | Não vê | Espera | READ COMMITTED em T2 |
| | T1 INSERT finalizado | Vê | Vê | Vê | Vê | |
| | T1 UPDATE não finalizado | Não vê | Espera | Espera | Espera | |
| | T1 UPDATE finalizado | Vê | Vê | Vê | Vê | |
| | T1 DELETE não finalizado | Não vê | Espera | Espera | Espera | |
| | T1 DELETE finalizado | Vê | Vê | Vê | Vê | |
| | T1 INSERT não finalizado | Não vê | Espera | Espera | Espera | SERIALIZABLE T2 em |
| | T1 INSERT finalizado | Não vê | Vê | Não vê | Não vê | |
| | T1 UPDATE não finalizado | Não vê | Espera | Espera | Espera | |
| | T1 UPDATE finalizado | Não vê | Vê | Rollback-OK Commit- ORA-08177 | Rollback-OK Commit- ORA-08177 | |
| | T1 DELETE não finalizado | Não vê | Espera | Espera | Espera | |
| | T1 DELETE finalizado | Não vê | Rollback-OK Commit- ORA-08177 | Rollback-OK Commit- ORA-08177 | Rollback-OK Commit- ORA-08177 | |

Transações em ORACLE

| | | T2 | | | | |
|----|--------------------------|--------|---------------------------------|---------------------------------|---------------------------------|-------------------------|
| | | SELECT | INSERT | UPDATE | DELETE | |
| T1 | T1 INSERT não finalizado | Não vê | Espera | Não vê | Espera | T2 em READ COMMITTED |
| | T1 INSERT finalizado | Vê | Vê | Vê | Vê | |
| | T1 UPDATE não finalizado | Não vê | Espera | Espera | Espera | |
| | T1 UPDATE finalizado | Vê | Vê | Vê | Vê | |
| | T1 DELETE não finalizado | Não vê | Espera | Espera | Espera | |
| | T1 DELETE finalizado | Não vê | Espera | Espera | Espera | T2 em SERIALIZABLE |
| | T1 INSERT não finalizado | Não vê | Espera | Não vê | Espera | |
| | T1 INSERT finalizado | Vê | Vê | Vê | Vê | |
| | T1 UPDATE não finalizado | Não vê | Espera | Espera | Espera | |
| | T1 UPDATE finalizado | Não vê | Vê | Rollback-OK Commit-ORA-08177 | Rollback-OK Commit-ORA-08177 | |
| | T1 DELETE não finalizado | Não vê | Espera | Espera | Espera | |
| | T1 DELETE finalizado | Não vê | Rollback-OK Commit-ORA-08177 | Rollback-OK Commit-ORA-08177 | Rollback-OK Commit-ORA-08177 | |

INSERT/UPDATE: não vê que há novos dados

DELETE: não vê que houve remoção (as tuplas permanecem)

Exemplo

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
```

```
/*bloco PL/SQL*/
```

```
BEGIN
```

```
UPDATE Turma SET NAlunos = NAlunos+1  
           WHERE SIGLA = 'SCE518';
```

```
INSERT INTO Matricula  
           VALUES ('SCE518', 1, 777, 2006, 0);
```

```
COMMIT;
```

```
EXCEPTION /* tratamento de exceções simplificado! */
```

```
/* Ex: se aluno 777 não existe*/
```

```
WHEN .... THEN BEGIN  
           dbms_output.put_line('Erro - rollback');  
           ROLLBACK;
```

```
END;
```

```
END;
```

Exemplo

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

/*bloco PL/SQL*/
BEGIN
    UPDATE Matricula SET Nota = 5.0
        WHERE Nota BETWEEN 4.9 AND 5.0;
    SAVEPOINT atualiza_matricula;

    UPDATE Turma SET NAlunos = NAlunos+1
        WHERE SIGLA = 'SCE518';
    INSERT INTO Matricula
        VALUES ('SCE518', 1, 777, 2006, 0);

    COMMIT;

    EXCEPTION /* tratamento de exceções simplificado! */
        /* Ex: se aluno 777 não existe*/
        WHEN .... THEN BEGIN
            ROLLBACK TO atualiza_matricula;
            COMMIT;

        END;

    END;
```



Transação em ORACLE

- Transações X blocos PL/SQL
 - não há relação entre início/fim de bloco e início/fim de transação. Ex:

```
UPDATE Aluno SET idade = 20 WHERE nusp = 222;  
DECLARE  
    v_idade Aluno.idade%Type;  
BEGIN  
    v_idade := 21;  
    UPDATE aluno SET idade = v_idade WHERE nusp = 222;  
    ROLLBACK;  
END;
```



Transações em ORACLE

- **Transações autônomas**

- são inicializadas dentro de outras transações (chamadas transações pai), mas são independentes delas
- podem ser efetivadas ou revertidas independente do estado da transação pai
- não permitem *rollback* para *savepoint* da transação pai
- definidas dentro de blocos PL/SQL

Exemplo

```
CREATE OR REPLACE PROCEDURE Teste_TAutonoma AS
PRAGMA AUTONOMOUS_TRANSACTION; -- sempre na seção declarativa
BEGIN
    DELETE FROM LBD08_Matricula WHERE nrousp = 7 and coddisc = 'SSC0740'
    and ano = 2008;
    commit; -- deve terminar com commit ou rollback, no próprio bloco
END;

set transaction isolation level read committed name 'transacao_pai';
BEGIN
    Teste_TAutonoma();
    DELETE FROM LBD08_Matricula WHERE nrousp = 7 and coddisc = 'SSC0740'
    and ano = 2008;
    ROLLBACK;
END;
```

O que aconteceu na base de dados???

Exemplo

```
CREATE OR REPLACE PROCEDURE Teste_TAutonoma AS
PRAGMA AUTONOMOUS_TRANSACTION; -- sempre na seção declarativa
```

```
BEGIN
```

```
DELETE FROM
```

```
and ano =
```

```
commit;
```

```
END;
```

```
set transaction
```

```
BEGIN
```

```
Teste_TAu
```

```
DELETE FROM
```

```
and ano =
```

```
ROLLBACK;
```

```
END;
```

SSC0740'

bloco

_pai';

SSC0740'

O comando de rollback não surtiu efeito, pois uma remoção semelhante ocorreu definida por uma transação autônoma (finalizada) de outro procedimento.

O que aconteceu na base de dados???



Transações em ORACLE

- Transações autônomas podem ser definidas em:
 - blocos anônimos de primeiro nível
 - subprogramas locais (empacotados e independentes)
 - *triggers*
 - métodos de um tipo de objeto



Transações e Controle de Concorrência

- Referências

- *Oracle Database Concepts*
- *OracleSQL Reference*
- Elmasri e Navathe. *Fundamentals of Database Systems*



PRÁTICA 10
