

Complementos de Bases de Dados – Transações e Concorrência –

Engenharia Informática

2º Ano / 1º Semestre

Cláudio Miguel Sapateiro

claudio.sapateiro@estsetubal.ips.pt

Sumário

- Introdução & Contexto
- Transações
- Controlo de Concorrência
- MS SQL T-SQL:
 - Transações & Controlo de Concorrência

Introdução

Contexto

➤ **Transações**

- As operações realizadas pelo SGBD são agrupadas segundo uma unidade de divisão de trabalho a **Transação**
 - Algumas transações agrupam de forma implícita e transparente certas operações levadas a cabo pelo SGBD na satisfação das solicitações
 - É contudo possível pelo utilizador definir transações e as operações que as compõem

➤ **Concorrência**

- Considerando que:
 - O SGBD é multi-utilizador
 - Necessita de tempos de resposta reduzidos
 - ❖ Com informação coerente!!
- ❖ A sua disponibilidade e performance é conseguida à custa de uma execução intercalada das transações (*multi-tasking*)

Transações

Operações

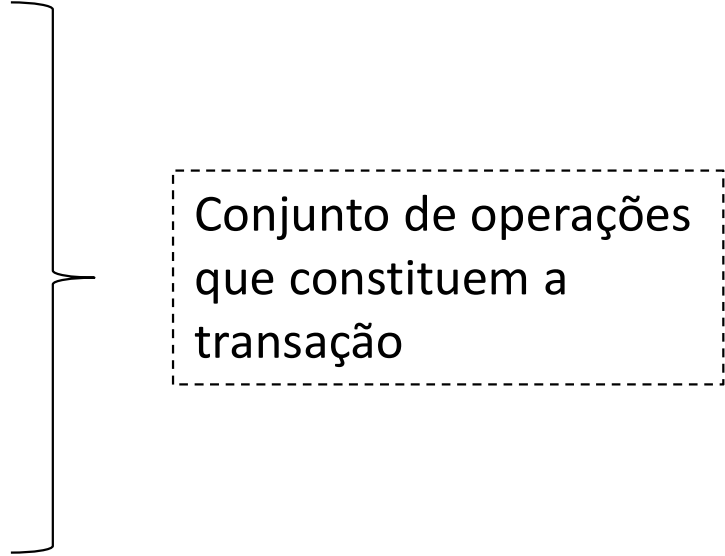
- De forma abstrata e simplificada, uma transação T_i pode ser vista como constituída por um conjunto de operações de leitura e escrita de dados
- **ler(X)**
transfere o dado X da base de dados para a memória alocada ao SGBD ficando disponível à transação que solicitou a operação de leitura (read).
- **escrever(X)**
transfere o dado X de memória para os ficheiros da base de dados em disco, cumprindo a solicitação de escrita (write) efetuada pela transação

Transações

Exemplo

Transação T_i transfere 50€ da conta A para a conta B

```
 $T_i$ : ler(A);  
      A:=A-50;  
      escrever(A);  
      ler(B);  
      B:=B+50;  
      escrever(B)
```



Conjunto de operações
que constituem a
transação

Transações

Estado(s)

- O estado de uma transação é sempre monitorizado pelo SGBD
 - que operações foram efetuadas?
 - as operações foram concluídas? Devem ser desfeitas?
- Estados de uma transação:
 - Ativa;
 - Em Efetivação;
 - Efetivada;
 - A desfazer;
 - Concluída.



Transações

Propriedades [ACID]

- **Atomicidade**
“*Tudo ou Nada*”; indivisível segundo vista externa
- **Consistência**
Uma transação conduz sempre a BD de um estado consistente para outro estado consistente
- **Isolamento**
Num contexto de transações concorrentes, a execução de uma transação T_i deve acontecer como se T_i se executasse isoladamente e não sofrer interferências de outras transações executando concorrentemente
 - ❖ a execução simultânea das operações de duas transações distintas (*escalonamento*) resulta numa situação equivalente ao estado obtido se uma das transações tivesse sido executada após a outra (*serialização*), independentemente da ordem
- **Durabilidade**
É garantido que as alterações efetuadas por uma transação, que concluiu com sucesso, persistem na BD (nenhuma falha posterior ocasionará a perda dessas alterações)

Transações

Exemplo

Escalonamento:

sequencia por ordem cronológica de execução das operações das transações

T ₁	T ₂
ler(A) A = A - 50	ler(A) A = A + A * 0.1 escrever(A) ler(B)
escrever(A) ←	
ler(B) B = B + 50 escrever(B)	
	B = B - A escrever(B) ←

*Qual constitui um
escalonamento
válido?*

T₁ interfere
em T₂

T₂ interfere
em T₁

Escalonamento Inválido

T ₁	T ₂
ler(A) A = A - 50 escrever(A)	ler(A) A = A + A * 0.1 escrever(A)
ler(B) B = B + 50 escrever(B)	
	ler(B) B = B - A escrever(B)

Escalonamento Válido

Transações

Delimitação de Transações (T-SQL)

- A DML possui instruções para delimitar o conjunto de operações que constituirão uma transação
- Por omissão os “comandos individuais” são transações (com *autocommit*)
- Em T-SQL a definição de transações (explícitas), suporta-se em:
 - BEGIN TRANSACTION (inicia uma transação)
 - COMMIT TRANSACTION (confirma as operações de uma transação)
 - ROLLBACK TRANSACTION (desfaz as operações de uma transação)

Transações

Delimitação de Transações (T-SQL)

```
BEGIN TRY
    BEGIN TRANSACTION
        -- update first account and debit from it
        -- update second account and credit in it
    COMMIT TRANSACTION
END TRY
BEGIN CATCH
    ROLLBACK TRANSACTION
END CATCH
```

```
USE AdventureWorks;
GO
BEGIN TRANSACTION;

BEGIN TRY
    -- Generate a constraint violation error.
    DELETE FROM Production.Product
    WHERE ProductID = 980;
END TRY
BEGIN CATCH
    SELECT
        ERROR_NUMBER() AS ErrorNumber
        ,ERROR_SEVERITY() AS ErrorSeverity
        ,ERROR_STATE() AS ErrorState
        ,ERROR_PROCEDURE() AS ErrorProcedure
        ,ERROR_LINE() AS ErrorLine
        ,ERROR_MESSAGE() AS ErrorMessage;

    IF @@TRANCOUNT > 0
        ROLLBACK TRANSACTION;
END CATCH;

IF @@TRANCOUNT > 0
    COMMIT TRANSACTION;
GO
```

Concorrência

Controlo de Concorrência

- Suportando o SGBD diversos utilizadores com acesso simultâneo aos dados, serão inevitavelmente geradas múltiplas transações concorrentes
- **Como é garantido o *Isolamento* das transações?**
 - *Solução Ineficiente:*
executar uma transação (completa) de cada vez!
» escalonamento *serializado* de transações
 - *Solução melhor:*
execução concorrente de transações preservando o isolamento,
através de um escalonamento (*schedule*)
» mas assegurando a integridade dos dados
- ✓ A execução concorrente permite um maior aproveitamento de recursos e melhoria dos tempos de espera face a transações (longas) serializadas

Concorrência

Exemplo

Serializado

T1	T2
	ler(A)
	A = A - 50
	escrever(A)
T1	T2
ler(A)	
A = A - 50	
escrever(A)	
ler(B)	
B = B + 50	
escrever(B)	
	ler(A)
	temp:=A*0,1;
	A:=A-temp
	escrever(A)
	ler(B)
	B:=B+temp
	escrever(B)

Concorrente

T1	T2
ler(A)	
A = A - 50	
	ler(A)
	temp:=A*0,1;
	A:=A-temp
	escrever(A)
	ler(B)
escrever(A)	
ler(B)	
B = B + 50	
escrever(B)	

- ❖ Execução concorrente de transações com baixo nível de isolamento, podem resultar num estado final inconsistente.
- ❖ O estado final da escala apresentada é inconsistente, pois a soma dos saldos de A e B não é preservada após a execução das duas transações

Concorrência

Exemplo

Concorrente

T1	T2
ler(A)	
A = A - 50	
	ler(A)
	temp := A * 0,1;
	A := A - temp
	escrever(A)
	ler(B)
escrever(A)	
ler(B)	
B = B + 50	
escrever(B)	



*Considerando operações
chave e esquemas de
recuperação
na definição do isolamento*

T ₁	T ₂
read(A) A := A - 50 write(A)	
	read(A) temp := A * 0.1 A := A - temp write(A)
read(B) B := B + 50 write(B) commit	
	read(B) B := B + temp write(B) commit

Concorrência

Na ausência de escalonamento Integral

- Efeitos na correção dos dados originados por escalonamentos sem isolamento integral:
 - atualização perdida (*lost-update*)
 - leitura suja (*dirty-read*)
 - análise inconsistente (*nonrepeatable read*)
 - leitura fantasma (*phantom read*)

Concorrência

lost-update

a transação T2 sobrescreve um registo atualizado anteriormente pela transação T1

T1	T2
ler(A)	
A = A - 50	
	ler(C)
	A = C + 10
escrever(A)	
ler(B)	
	escrever(A)
B = A + 30	
escrever(B)	

a atualização de A
por T₁ foi perdida!

Concorrência

dirty-read

T1 atualiza um registo A, acedido posteriormente por outra transação T2 .
T2 leu um valor incorreto de A pois T1 falhou.

A transação T_1 falha e por isso deve ser colocado em A o seu valor inicial.

T1	T2
ler(A)	
$A = A - 20$	
escrever(A)	
	ler(A)
	$A = A + 10$
	escrever(A)
ler(Y)	
ROLLBACK	

T_2 leu um valor de A que acabará por não ser válido.

Concorrência

nonrepeatable-read

T1 atualiza um registo A várias vezes. Após cada alteração o registo é acedido por outra transação T2 que não obtém um valor idêntico de A

A transação T_1 escreve várias vezes em A.

T1	T2
ler(A)	
$A = A - 20$	
escrever(A)	
	ler(A)
$A = A - 10$	
escrever(A)	
	ler(A)

T_2 leu um valor de A diferente do anteriormente lido.

Concorrência

nonrepeatable-read

Example

- A Web site offering goods for sale may list an item as being in stock,
- yet by the time a user selects the item and goes through the checkout process, that item might no longer be available;
- Viewed from a database perspective, this would be a ***nonrepeatable*** read

Concorrência

phantom-read

T1 insere/apaga registros num intervalo de A-D.

Após a interseção/remoção de registros, o intervalo é acedido por outra transação T2 que não obtém um valor idêntico para o intervalo A-D .

T1	T2
	ler(A-D)
escrever(C)	
	ler(A-D)

T₂ leu um valor do intervalo A-D diferente do anteriormente lido, porque T₁ inseriu o registro C.

Concorrência

MS SQL

O MS SQLServer suporta os níveis de isolamento do standard SQL através da instrução:

SET TRANSACTION ISOLATION LEVEL *nível*

nível pode ser:

- **SERIALIZABLE:** *Ti* executa com completo isolamento
- **REPEATABLE READ:** *Ti* só lê dados efetivados mas outras transações podem criar dados no intervalo lido por *Ti*
- **READ COMMITTED (default):** *Ti* só lê dados efetivados, mas outras transações podem escrever em dados lidos por *Ti*
- **READ UNCOMMITTED:** *Ti* pode ler dados que ainda não sofreram efetivação

Concorrência

MS SQL

Efeitos secundários admitidos por nível de isolamento

Isolation level	Dirty read	Nonrepeatable read	Phantom
Read uncommitted	Yes	Yes	Yes
Read committed	No	Yes	Yes
Repeatable read	No	No	Yes
Snapshot	No	No	No
Serializable	No	No	No

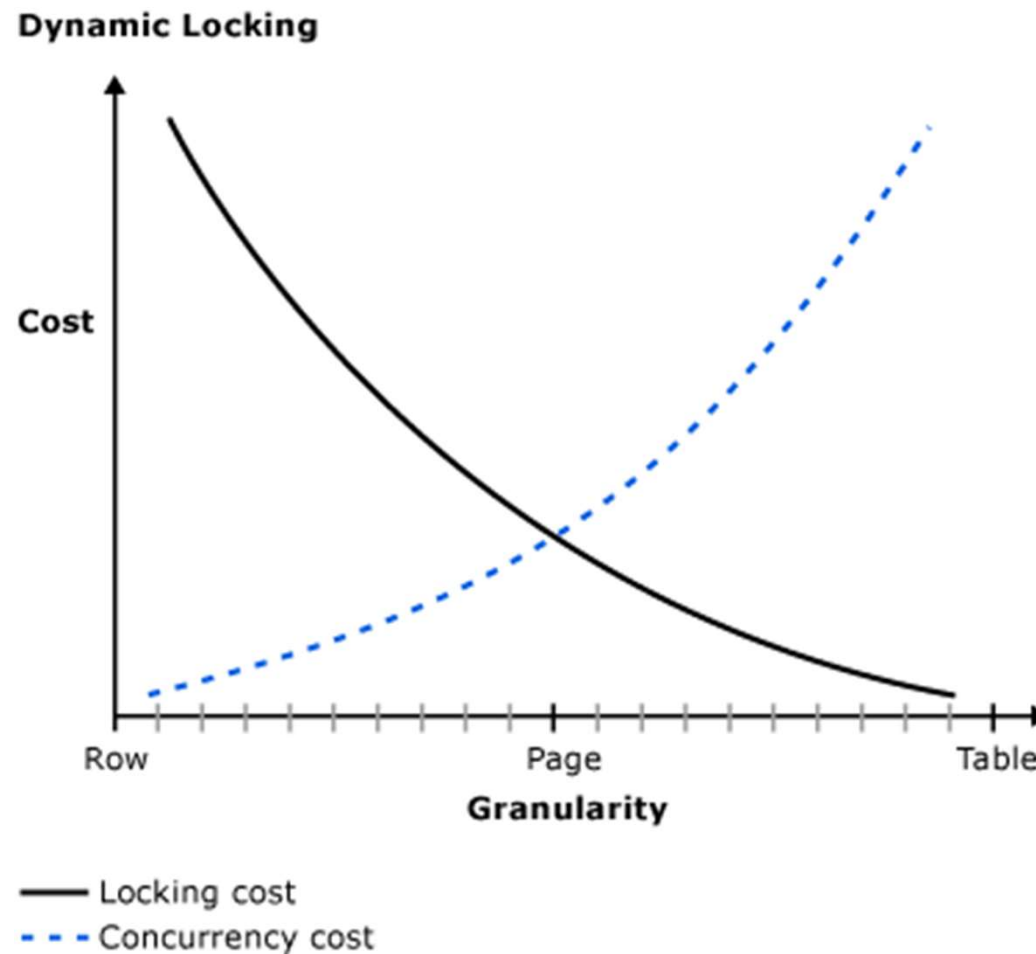
Concorrência

Implementação do controlo de concorrência

Mecanismos

- Locking
bloqueio temporário aos dados em causa (hierarquia e tipos de locks)
- Row versioning / snapshot
possibilidade de permitir acesso a uma cópia temporária de dados

Concorrência



Concorrência

Níveis de Isolamento

Exemplo

```
BEGIN TRAN
```

```
UPDATE IsolationTests SET Col1 = 2
```

```
--Simulate having some intensive processing here with a wait
```

```
 $\Delta t$  WAITFOR DELAY '00:00:10'
```

```
ROLLBACK
```

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
```

```
SELECT * FROM IsolationTests
```

dirtyRead

	Id	Col1	Col2	Col3
1	1	2	2	3
2	2	2	2	3
3	3	2	2	3
4	4	2	2	3
5	5	2	2	3
6	6	2	2	3
7	7	2	2	3
8	8	2	2	3

Δt

	Id	Col1	Col2	Col3
1	1	1	2	3
2	2	1	2	3
3	3	1	2	3
4	4	1	2	3
5	5	1	2	3
6	6	1	2	3
7	7	1	2	3
8	8	1	2	3

Concorrência

Níveis de Isolamento

Exemplo

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
```

```
BEGIN TRAN
```

```
SELECT * FROM IsolationTests
```

```
WAITFOR DELAY '00:00:10'
```

```
SELECT * FROM IsolationTests
```

```
UPDATE IsolationTests SET Col1 = -1
```

```
select * from IsolationTests
```

phantomRead

Results				
	Id	Col1	Col2	Col3
1	1	1	2	3
2	2	1	2	3
3	3	1	2	3
4	4	1	2	3
5	5	1	2	3
6	6	1	2	3
7	7	1	2	3
8	8	1	2	3

	Id	Col1	Col2	Col3
1	1	1	2	3
2	2	1	2	3
3	3	1	2	3
4	4	1	2	3
5	5	1	2	3

Δt

Δt

Results				
	Id	Col1	Col2	Col3
1	1	-1	2	3
2	2	-1	2	3
3	3	-1	2	3
4	4	-1	2	3
5	5	-1	2	3
6	6	-1	2	3
7	7	-1	2	3
8	8	-1	2	3

mini Sumário

1. Conceito de transação
2. Propriedades ACID
3. Tipos de erros e Níveis de isolamento

10:00



Exercícios

1. Porque motivo pode ser aceitavel colocar um nivel de isolamento mais baixo?
2. Qual a consequencia de um nivel de isolamento elevado?
3.
 - a. Que niveis de isolamento permitem evitar todas as ocorrências de erro abordadas?
 - b. Fazem-no da mesma maneira?

Complementos de Bases de Dados – Transações e Concorrência –

Engenharia Informática

2º Ano / 1º Semestre

Cláudio Miguel Sapateiro

claudio.sapateiro@estsetubal.ips.pt