

**Transações e Concorrência Ficheiro**

Grupo Nº 6

Alexandre Coelho, Nº 190221093

Sérgio Veríssimo, Nº 190221128

**Licenciatura em Engenharia Informática**

**COMPLEMENTOS de BASES DE Dados**

**laboratório 5**

**Índice**

[I – READ UNCOMMITTED e COMMITTED 2](#_Toc25737931)

[II – REPEATABLE READ 2](#_Toc25737932)

[III – SERIALIZABLE 2](#_Toc25737933)

[IV – CONTROLO DE SESSÕES 2](#_Toc25737934)

# I – READ UNCOMMITTED e COMMITTED

Janela #1 - Execute a seguinte transação, sem a finalizar:

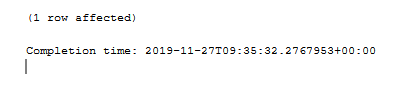
BEGIN TRANSACTION

UPDATE dbo.Customer

SET EmailAddress = 'new@estsetubal.ips.pt.pt'

WHERE CustomerId = 5

A execução deste código deu o resultado de uma linha afetada com sucesso. Como se pode ver a baixo:



Janela #2 - Tente obter a leitura do valor:

SELECT EmailAddress

FROM dbo.Customer

WHERE CustomerId = 5

Não consegue obter nenhuma leitura? Porquê?

Não, pois existe uma transação ainda não efetuada na tabela em questão e devido ao nível de isolamento do SQL Server estar definido por defeito como READ COMMITTED não é possível aceder a essa tabela pois a mesma esta num estado de LOCK.

Janela #2 - Tente novamente obter o valor, mas executando a seguinte instrução:

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

SET NOCOUNT ON

GO

SELECT EmailAddress

FROM dbo.Customer

WHERE CustomerId = 5

OU

SELECT EmailAddress

FROM dbo.Customer (NOLOCK)

WHERE CustomerId = 5

Comente este procedimento.

Ao alterar o nível de isolamento do SQL Server para READ UNCOMMITED é possível obter os dados da tabela, no entanto estes poderão estar num estado de inconsistência pois a transação nessa tabela ainda não esta completa.

Termine a transação na Janela #1

ROLLBACK

Este comando no caso de haver algum tipo de erro ou inconformidade irá desfazer as operações da transação de update da Janela #1. Quando é executado o código da Janela #2 este irá devolver o seguinte resultado:



Pois não tendo a transação da janela #1 sido efetuada e tendo o código da Janela #2 sido executada, o rollback irá reverter essa transação não sendo realizado o update.

As seguintes instruções pretendem demonstrar a importância das transações em execuções concorrenciais.

Abra duas janelas e execute as seguintes funções de seguida:

Janela #1 - Efetue 2 queries identificas, separadas por 10 segundos, à mesma tabela.

SET TRANSACTION ISOLATION LEVEL READ COMMITTED

SET NOCOUNT ON

GO

BEGIN TRAN

SELECT EmailAddress

FROM dbo.Customer

WHERE CustomerId = 5

WAITFOR DELAY '00:00:10'

SELECT EmailAddress

FROM dbo.Customer

WHERE CustomerId = 5

COMMIT TRAN

Janela #2 - Alterar o EmailAddress enquanto o procedimento anterior está em execução

BEGIN TRAN

UPDATE dbo.Customer

SET EmailAddress = 'new@estsetubal.ips.pt.pt'

WHERE CustomerId = 5

COMMIT

Qual o resultado das 2 queries na 1ª Janela? Comente

Na primeira execução foi devolvido o valor antes do update pois o mesmo ainda não teria sido efetuado, já na segunda query o valor devolvido foi o valor após o update, pois durante esse intervalo existiu uma transação completada com sucesso e o nível de isolamento do SQL Server definido era READ UNCOMMITTED.

# II – REPEATABLE READ

1. Qual o resultado das 2 queries na 1ª Janela? Comente.

O resultado das duas queries foi “lucy0@adventure-works.com”.

Não efetuou uma alteração, devido à transação da primeira janela bloquear qualquer alteração aos dados (updates), o que vai bloquear as operações da segunda janela, que consistem numa alteração do email para "update@estsetubal.ips.pt.pt" onde o CustomerId = 5;

Em que situações se deve utilizar o nível de isolamento REPEATABLE READ?

Devemos utilizar o nível de isolamento REPEATABLE READ quando ocorre o fenómeno de leitura fantasma (phantom-read).

Altere o comando do ponto 2 para inserir uma nova linha na mesma tabela.

INSERT INTO dbo.Customer VALUES (0, 'Mr.', 'FirstName', null, 'LastName', null, 'CompanyName', 'SalesPerson','EmailAdress','Phone','','',NEWID(),

GETDATE());

Execute o ponto 1 e 2 novamente e comente o resultado.

Com esta alteração, a query presente já segunda janela, já executa e não fica bloqueada. Não existe bloqueio, porque o REPEATABLE READ apenas não permite atualizações (updates) em simultâneo com uma operação de leitura aos mesmos dados. Neste caso, foi alterado o código na Janela 2, de uma atualização, para uma inserção.

# III – SERIALIZABLE

1. Altere a alínea anterior e onde se lê:

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

**colocar:**

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

Comente o resultado obtido.

A transação que tem declarado o nível de isolamento SERIALIZABLE (Janela 1), bloqueia devido a existir uma inserção em simultâneo com a execução da transação da janela 1. Este bloqueio, ocorre devido ao nível de isolamento SERIALIZABLE, bloquear todas as transações (escrita e leitura), quando existe uma variação nos dados.

# IV – CONTROLO DE SESSÕES

1. Identifique o número de sessão associado às janelas #1 e #2 do ponto da Etapa I. Deverá executar de novo ambas e numa terceira janela este código:

Observe e explore o resultado da seguinte query:

Com esta execução da view de sistema sobre concorrência, é possível verificarmos que a query com id de sessão(session id) 53 está a bloquear a execução da query com id de sessão 52.

Execute a query seguinte substituindo X e Y, pelos números de sessão identificados no anterior ponto 1 (V.2 – 1)

Esta query permite verificar a query que está a tentar ser executada.

Execute a query no ponto I.1 e I.2 e identifique os processos bloqueados utilizando o procedimento sp\_who2. Parar a sessão que está a criar o bloqueio utilizando o método KILL.

O processo bloqueado (identificado pelo procedimento sp\_who2) , foi destruída, devido à paragem de sessão onde este processo estava a ser executado. O processo pendente como foi destruído sem ser efetivado, o resultado, diz-se equivalente a uma aplicação de um ROLLBACK.