Complementos de Bases de Dados

2019/2020

Licenciatura em Engenharia Informática

IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

Laboratório 5 – Transações e Concorrência

Objetivos:

- Demonstração dos níveis de isolamento de transações no SQL Server

Enunciado:

Crie a tabela Customer:

```
SELECT * INTO Customer FROM SalesLT.Customer WHERE CustomerID < 1000
```

I – READ UNCOMMITTED e COMMITTED

É possível obter os valores que ainda estão bloqueados ou não confirmados por alterações de outra transação.

Abra duas janelas e execute as seguintes funções:

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

   ```
   BEGIN TRANSACTION
   UPDATE  dbo.Customer
   SET     EmailAddress = 'new@estsetubal.ips.pt.pt'
   WHERE   CustomerId = 5
   ```

2. Janela #2
   Tente obter a leitura do valor:

   ```
   SELECT  EmailAddress
   FROM    dbo.Customer
   WHERE   CustomerId = 5
   ```

   Não consegue obter nenhuma leitura? Porquê?

3. 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.

4. Termine a transação na Janela #1

```
ROLLBACK
```

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

Abra duas janelas e execute as seguintes **<u>funções de seguida</u>**:

5. 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
```

6. 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
```

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

II – REPEATABLE READ

Este nível de isolamento ultrapassa as limitações dos isolamentos anteriores

Abra duas janelas e execute as seguintes **<u>funções de seguida</u>**:

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

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

2. Janela #2
   Alterar o valor do EmailAddress enquanto o procedimento anterior está em execução

```
BEGIN TRAN
UPDATE  dbo.Customer
SET     EmailAddress = 'update@estsetubal.ips.pt.pt'
WHERE   CustomerId = 5
COMMIT
```

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

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

5. 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.

III – SERIALIZABLE

Para ultrapassar o problema das leituras fantasma, é necessário um nível de isolamento SERIALIZABLE. Este nível, para além de assegurar o isolamento do READ COMMITED e REPEATABLE READ, permite também que transações concorrentes executem como se fosse em série. Contudo, o preço a pagar é a redução da concorrência do SGBD e portanto, menor performance.

1. Altere a alínea anterior e onde se lê:

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

colocar:

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
```

2. Comente o resultado obtido.

IV – CONTROLO DE SESSÕES

1. Views de sistema sobre concorrência

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

```
SELECT -- use * to explore
  request_session_id            AS spid,
  resource_type                 AS restype,
  resource_database_id          AS dbid,
  resource_description          AS res,
  resource_associated_entity_id AS resid,
  request_mode                  AS mode,
  request_status                AS status
FROM sys.dm_tran_locks;
```

1.2. Observe e explore o resultado da seguinte query:

```
SELECT -- use * to explore
  session_id AS spid,
  blocking_session_id,
  command,
  sql_handle,
  database_id,
  wait_type,
  wait_time,
  wait_resource
FROM sys.dm_exec_requests
WHERE blocking_session_id > 0;
```

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

```
SELECT session_id, text
FROM sys.dm_exec_connections
  CROSS APPLY sys.dm_exec_sql_text(most_recent_sql_handle) AS ST
WHERE session_id IN( X, Y);
```

1.4. 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.

(fim de enunciado)