Developing Database

04 – Transações e Controle de Fluxo I

Sand Onofre

Sand.Onofre@FaculdadeImpacta.com.br

Sumário

- Transações
  - Alta Concorrência
  - Unidade Lógica de Trabalho
  - Propriedades ACID
- Controle de Fluxo I
  - IF ... ELSE
  - BEGIN ... END
  - GOTO
  - RETURN
- Exercícios

Alta Concorrência - Overview

- Sistemas de banco de dados devem permitir que um grande número de conexões simultâneas faça uso de seus dados para leitura ou escrita de informações.

- Um dos objetivos dos sistemas de gerenciamento de banco de dados é fornecer a cada usuário a ilusão, sempre que possível, que é o único usuário no sistema.

- Sistemas de banco de dados lutam com a necessidade de equilibrar a consistência e simultaneidade. Muitas vezes há um trade-off ("perde-e-ganha") direto entre estes dois objetivos. O desafio é reduzir o impacto da concorrência, mantendo a consistência suficiente.

Transação

- Uma transação é uma sequência de operações executadas como uma única unidade lógica de trabalho.

```
INSERT INTO Veiculo VALUES (1, 'Strada')
```

```
UPDATE Veiculo SET Descricao = 'Strada Adventure'
WHERE id = 1

INSERT INTO Veiculo VALUES (2, 'Montana')
INSERT INTO Veiculo VALUES (3, 'Saveiro')

DELETE Veiculo WHERE id = 1
```

Propriedades das Transações - ACID

- Transações devem apresentar quatro propriedades que são conhecidas como ACID, responsáveis por assegurar que várias modificações de dados são processadas como uma unidade. Por exemplo, uma transação bancária pode creditar uma conta e debitar outra. Ambas as etapas devem ser concluídas em conjunto ou não concluídas (desfazendo qualquer modificação realizada). SGBDs devem suportar o processamento de transações para gerenciar múltiplas transações.

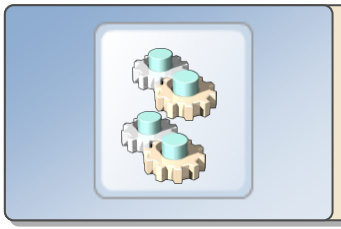
Atomicidade

Consistência

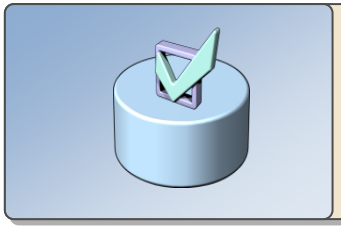
I solamento

Durabilidade

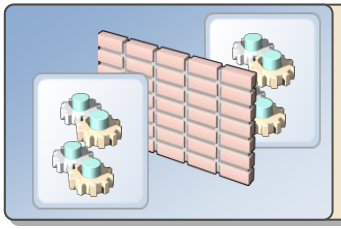
ACID



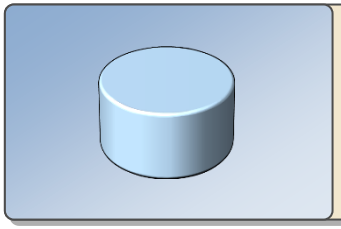
Uma transação é uma unidade de trabalho Atômica



Uma transação deixa os dados num estado Consistente



Uma transação é Isolada de outras transações correntes



Uma transação é Durável

Transações

- Transações usam LOCK (travas) para impedir que outros usuários alterem ou leiam os dados em uma transação que não foi concluída.

- O BLOCK (bloqueio) é necessário no processamento de transações online (OLTP) para sistemas multi-usuário.

T-SQL: Controle de Fluxo

Estas são as palavras-chave de controle de fluxo:

- BEGIN...END
- BREAK
- GOTO
- CONTINUE
- IF...ELSE
- WHILE
- RETURN
- WAITFOR

```
IF Boolean_expression
BEGIN
   { sql_statement | statement_block }
END
ELSE
   { sql_statement | statement_block }
```

T-SQL: Controle de Fluxo

- IF ... ELSE

```
IF Boolean_expression
   { sql_statement | Declaração SQL }
ELSE
   { sql_statement | Declaração SQL }
```

- Não importando qual seja a expressão boleana sempre retornará um dos 3 valores possíveis: TRUE, FALSE ou UNKNOWN.
- Se TRUE, executará a declaração imediatamente abaixo da cláusula IF.
- Se FALSE ou UNKNOWN, executará a declaração imediatamente abaixo da cláusula ELSE.

T-SQL: Controle de Fluxo

- IF ... ELSE

```
IF 1 = 1
   PRINT 'VERDADEIRO'
ELSE
   PRINT 'FALSO ou DESCONHECIDO'

IF 1 <> 1
   PRINT 'VERDADEIRO'
ELSE
   PRINT 'FALSO ou DESCONHECIDO'

IF 1 = null
   PRINT 'VERDADEIRO'
ELSE
   PRINT 'FALSO ou DESCONHECIDO'
```

T-SQL: Controle de Fluxo

- IF ... ELSE

```
/*
Rotina Teste para o Controle de Fluxo IF ... ELSE
*/

-- Declarando Variável e ajustando valor inicial
Declare @qtd smallint
Set @qtd = 200

-- Verificação do Valor da quantidade
IF @qtd >= 200 -- Se Maior ou Igual a 200
   Set @qtd += 50 --Somar 50 unidades
ELSE
   Set @qtd -= 20 --Reduzir 20 unidades

Select @qtd --Independente do caminho tomando no fluxo, retorna o valor @qtd
```

T-SQL: Controle de Fluxo

- IF ... ELSE

```
IF Boolean_expression
BEGIN
   { statement_block | Bloco de Declarações }
END
ELSE
BEGIN
   { statement_block | Bloco de Declarações }
END
```

- As cláusulas BEGIN ... END, auxiliam a determinar um bloco de declarações que serão executadas, caso o fluxo se dê através do IF ou mesmo através do ELSE.

- Como boas práticas, independentemente se temos uma declaração ou um bloco de declarações, o código fica mais claro utilizando o BEGIN ... END para determinar o que será executado no fluxo.

T-SQL: Controle de Fluxo

- IF ... ELSE

```
-- Declarando Variáveis e Ajustando valores Iniciais
declare @qtd smallint = 200, @taxa decimal(3,2) = 0.05

-- Aplicação das taxas
if @qtd * @taxa > 20
begin
	set @qtd += @qtd * 0.1 -- Aumentar em 10 %
	set @taxa *= 2 -- Duplicar a taxa
end
else
begin
	set @qtd -= @qtd * 0.15 -- Reduzir em 15 %
	set @taxa *= 2 -- Duplicar a taxa
end

select @qtd as 'Valor da Quantidade', @taxa as 'Valor da Taxa'
```

T-SQL: Controle de Fluxo

- IF ... ELSE

```
IF Boolean_expression
     { sql_statement | Declaração SQL }
ELSE
BEGIN
   { statement_block | Bloco de Declarações }
END
```

- Na prática é possível combinar declaração e bloco de declarações com o uso do BEGIN ... END ou sem sua utilização, tanto na cláusula IF, quanto na cláusula ELSE.

- Esse tipo de combinação é muito comum em códigos legados e apesar das boas práticas sugerirem sempre a utilização do BEGIN ... END em qualquer um dos casos (declaração ou bloco de declarações), não é considerado erro este tipo de combinação.

T-SQL: Controle de Fluxo

- IF ... ELSE

```
-- Declarando Variáveis e Ajustando valores Iniciais
declare @qtd smallint = 200, @taxa decimal(3,2) = 0.05

-- Aplicação das taxas
if @qtd * @taxa > 20
begin
        set @qtd += @qtd * 0.1 -- Aumentar em 10 %
        set @taxa *= 2 -- Duplicar a taxa
end
else  -- Reduzir em 15 % e Duplicar a taxa
        select @qtd -= @qtd * 0.15, @taxa *= 2

select @qtd as 'Valor da Quantidade', @taxa as 'Valor da Taxa'
```

T-SQL: Controle de Fluxo

❑ RETURN *[integer_expression]*

- Sai incondicionalmente de uma consulta ou procedimento. RETURN é imediato e completo e pode ser usado em qualquer ponto para sair de um procedimento, lote ou bloco de instruções. As instruções posteriores a RETURN não são executadas.

integer_expression

✓ É opcional e representa um valor inteiro que é retornado. Os procedimentos armazenados podem retornar um valor inteiro a uma chamada de execução. Quando usado com um procedimento armazenado, RETURN não pode retornar um valor nulo.

T-SQL: Controle de Fluxo

- RETURN

```
-- Declarando Variáveis e Ajustando valores Iniciais
declare @qtd smallint = 200, @taxa decimal(3,2) = 0.05

-- Verificando valores
if @qtd * @taxa >= 10
begin
        select 'Valor da Quantidade x Taxa = '
                + cast(@qtd * @taxa as varchar) + ' !'
        return
end

select @qtd as 'Valor da Quantidade', @taxa as 'Valor da Taxa'
```

T-SQL: Controle de Fluxo

- GOTO *Rotulo*

- Altera o fluxo de execução para um rótulo (label). A instrução ou as instruções Transact-SQL que seguem GOTO são ignoradas e o processamento continua no rótulo. As instruções GOTO e os rótulos podem ser usados em qualquer lugar em um procedimento, lote ou bloco de instruções.

- O comando GOTO faz um salto de um ponto A para um ponto B, podendo ser para adiante da linha do código que contém o GOTO ou para linhas anteriores no mesmo batch. O mesmo não costuma ser usado em programação estruturada, mas pode ser útil em algumas situações.

T-SQL: Controle de Fluxo

- GOTO *Rotulo*

```
-- Declarando Variáveis e Ajustando valores Iniciais
declare @qtd smallint = 200, @taxa decimal(3,2) = 0.05

goto MarcacaoDeUmRotulo -- Salta diretamente para este rótulo

set @qtd = @qtd + 100
set @taxa = @taxa + 1

MarcacaoDeUmRotulo: -- Definição do Rótulo
       select @qtd as 'Valor da Quantidade’
              , @taxa as 'Valor da Taxa'
```

T-SQL: Controle de Fluxo

```
if (select count(*) from sys.objects) > 50
        goto Objetos_Maior_50 -- Forçar desvio de Fluxo
else
        select 'Quantidade de OBJETOS menor que 50 !'

Return


Objetos_Maior_50: -- Desvio para o número de sys.objects
        select  count(*) as 'Quantidade de Objetos encontrados !'
        from    sys.objects
```

T-SQL: Controle de Fluxo

```
-- Variável já com valor inicial de objetos de sistema e usuários
declare @qtd int = (select count(*) from sys.objects)

-- Verificando o número de objetos na base de dados atual
if @qtd > 50
        goto Objetos_Maior_50 -- Forçar desvio de Fluxo
return -- Forçar a saída imediata do script

/*
Como o número de objetos é maior que 50, só chegará a este ponto do BATCH se houver
algum outro desvio de fluxo
*/
Desvio_Nao_Estruturado: -- Pegar o número de tabelas de usuários
        select @qtd = count(*) from sys.tables
        select 'Quantidade de TABELAS =  ‘ + cast(@qtd as varchar) + ' ! ‘
                as 'Select no Desvio_Nao_Estruturado:’
        return -- Forçar a saída imediata do script

Objetos_Maior_50: -- Só chega neste ponto do BATCH por desvio de fluxo
        select concat('Quantidade de OBJETOS = ', cast(@qtd as varchar)
                , ' ! ‘) as 'Select no Objetos_Maior_50:’
        goto Desvio_Nao_Estruturado -- Desvio pode ser para qualquer ponto do BATCH
```

Exercícios

Construa um bloco T-SQL com as seguintes características:

I. Criar variáveis saldoInicial e saldoTotal, numéricas, de 2 bytes e com valores iniciais respectivamente 100 e 200.

II. Criar variáveis dtAtual e dtFutura, de 3 bytes e armazenar com valores iniciais em 19-06-2015 e 25-08-2015.

III. Utilizando funções built-in, adicione ao saldoInicial a diferença em dias obtida entre as datas anteriores.

IV. Da mesma forma que a instrução anterior, multiplique saldoTotal pela diferença em meses destas datas.

Exercícios

V. Faça um controle de fluxo verificando se o saldoInicial é maior ou igual ao saldoTotal, Se TRUE:

   I. Escreva (select) a seguinte mensagem:

      “Seu Saldo em ” + dtAtual + espaço(1) + “ é de ” + saldoInicial + “.”

   II. Faça com que a dtAtual seja somada em 21 dias

VI. Se FALSE, escreva:

“Seu Saldo em ” + dtFutura + espaço(1) + “ é de ” + saldoTotal + “.”

VII. Gere a saída de todas as variáveis numa única linha, colocando um ALIAS em cada coluna de forma que saibamos o que cada coluna representa.

Exercícios

- Através de comandos ad hoc, descubra as quantidade de registros das seguintes tabelas no banco de dados Temp: sys.objetcs, sys.tables, sys.columns

- Examine também o conteúdo da tabela sys.columns

- A partir da investigação anterior, crie um script com as seguintes características

1. Criar variáveis com os **menores tipos de dados** possíveis que armazenem:
   a. As **quantidades** de dados das tabelas mencionadas.
   b. Os **Maiores** e **Menores** valores dos seguintes campos da tabela **sys.columns**: **Object_id, name, system_type_id, user_type_id, is_nullable**
   c. A **média** (com **2 casas decimais**) do campo **column_id** da tabela **sys.columns**

Exercícios

2. Após o armazenamento, construa o seguinte controle de fluxo:

   a. Caso o menor valor de **system_type_id** for menor ou igual ao menor valor de **user_type_id**, escreva um texto que mostre isso. Em seguida **some 10 unidades** em cada uma destas variáveis.
   b. Senão, escreva um texto que mostre que são divergentes.

   c. Caso o menor valor de **object_id** seja **negativo**, desvie o fluxo para um rótulo chamado **DESVIO_DE_FLUXO**, nas últimas linhas do script. No rótulo mencionado gere um **SELECT** que descreva este desvio, qual a variável e o valor responsável pelo desvio.
   d. Senão, verifique se os maiores valores em **system_type_id** e **user_type_id** são diferentes. Se forem, escreva um texto que descreva isso, mostrando os 2 valores.

   e. Force o término do script antes de chegar no rótulo **DESVIO_DE_FLUXO**.

Obrigado !

Sand Onofre
Sand.Onofre@FaculdadeImpacta.com.br