Bases de Dados 2007/2008

Aula 9

Sumário

Referências

http://msdn2.microsoft.com/en-us/library/ms189826.aspx (linguagem t-sql)
http://www.di.ubi.pt/~pprata/bd/BD0405-Proc.sql (procedimentos)
http://www.di.ubi.pt/~hugomcp/bd2/t05.pdf (procedimentos)
http://www.di.ubi.pt/~pprata/bd/BD0405-Triggers.sql (triggers)
http://doc.ddart.net/mssql/sql70/create_8.htm (triggers)
http://www.java2s.com/Code/SQLServer/CatalogSQLServer.htm (exemplos vários)

SQL Server 2000 para Profissionais, Orlando Belo, FCA ISBN 972-722-505-5
SQL - Structured Query Language, Luís Manuel Dias Damas, FCA ISBN 972-722-443-1

1 Try Catch Processamento de erros e excepções em T-SQL

Os erros podem ser processados usando blocos de instruções TRY e CATCH semelhante aos utilizados em outras linguagens como Java e C. Temos então dois blocos, o bloco TRY e o bloco CATCH. Quando um erro é detectado no interior do bloco TRY o controlo é passado para o bloco CATCH onde o erro pode ser processado. O bloco TRY começa com a instrução BEGIN TRY e termina com END TRY. Um bloco CATCH começa com BEGIN CATCH e termina com END CATCH.

Sintaxe

```
BEGIN TRY
{ sql_statement | statement_block }
END TRY

BEGIN CATCH
{ sql_statement | statement_block }
END CATCH
```

No exemplo seguinte temos duas instruções SELECT intercaladas por um cursor não declarado no bloco TRY. A execução vai gerar um erro no cursor não executando o último select e passando a execução para o bloco CATCH.

```
BEGIN TRY
SELECT 'Apareceu !!!' AS aparece
-- erro de cursor não declarado
open CursorXPTO;
SELECT 'Este select já não aparece!!!' AS escondida
END TRY
BEGIN CATCH
IF ERROR_NUMBER() = 16916
Print 'Erro, o cursor não está declarado'
   END CATCH;
```

As funções do SGBD que disponibilizam informação sobre erros no âmbito de instruções TRY e CATCH são os seguintes:

ERROR_NUMBER()	indica o número do erro encontrado.
ERROR_SEVERITY()	mostra a gravidade do erro. Erros com gravidades entre 11 e 16 podem ser corrigidos pelo utilizador.

ERROR_STATE()	mostra o estado do erro.
ERROR_PROCEDURE()	mostra o nome do procedimento ou trigger onde o erro ocorreu.
ERROR_LINE()	mostra o número da linha onde ocorreu o erro.
ERROR_MESSAGE()	mostra a descrição completa do erro.

Exemplos de tipos de erros referenciados por ERROR_NUMBER()

16905	abertura de um cursor já aberto	16917	acesso a um cursor que não está aberto
8134	divisão por zero	16916	acesso a um cursor que não declarado

Exemplo erro divisão por zero que é passado para o CATCH.

```
BEGIN TRY
-- provocar uma divisão por zero
SELECT 1/0;
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;
END CATCH;
```

Resultado obtido

```
8134     16    1     NULL  3     Divide by zero error encountered.
```

2 RAISERROR

O Raiserror permite gerar excepções. No caso de a gravidade do erro estar compreendida entre 11 e 16 a excepção pode ser processada pelo Catch. Este pode também ser usado para enviar as mensagens de erro a aplicações clientes.

Exemplo com inserção de valores que violam uma restrição UNIQUE previamente definida, sendo gerado neste caso uma mensagem de erro.

```
BEGIN TRY
SELECT * FROM aluno
DECLARE @num INT
SET @num=1
WHILE @num < 20
BEGIN
INSERT INTO aluno VALUES (300,'Ana','Covilhã')
SET @num = @num+1
END
SELECT 'Não se vê este select'
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;

RAISERROR 2147483647 'Violação da restrição UNIQUE KEY da constraint morada';
select * from aluno
END CATCH
```

Exemplo com erro na actualização que violam uma restrição previamente definida.

```
BEGIN TRY
INSERT INTO aluno VALUES (1,'Pedro','covilhã')
```

```
INSERT INTO aluno VALUES (2,'Ana','covilhã')
UPDATE aluno SET numaluno= 100 WHERE morada='covilhã'
END TRY
BEGIN CATCH
DECLARE @erro NVARCHAR(1000);
DECLARE @severidade INT;
DECLARE @estado INT;
SET @erro = ERROR_MESSAGE();
SET @severidade = ERROR_SEVERITY();
SET @estado = ERROR_STATE();
RAISERROR ( @erro, @severidade, @estado)
END CATCH
SELECT * FROM aluno
```

3 Triggers

Um Trigger é um objecto que é associado a uma tabela na base de dados. Em muitos aspectos, é semelhante a um procedimento armazenado, sendo desencadeado automaticamente quando ocorrem determinadas condições pré estabelecidas (operações do tipo INSERT UDATE e DELETE).

Sintaxe

```
CREATE TRIGGER trigger_name
ON { ALL SERVER | DATABASE }
[ WITH <ddl_trigger_option> [ ,...n ] ]
{ FOR | AFTER } { event_type | event_group } [ ,...n ]
AS { sql_statement [ ; ] [ ...n ] | EXTERNAL NAME < method specifier > [ ; ]
}
```

Podemos aceder aos valores que estão a ser modificados por um trigger através do acesso às tabelas: `Inserted` e `Deleted`. Estas duas tabelas são processadas do mesmo modo que as tabelas da base de dados. Deve salvaguardar o facto que não se pode alterar o seu conteúdo.

Exemplo de um Trigger que escreve uma string antes de executar uma inserção na ta bela aluno.

```
CREATE TRIGGER alerta
 ON aluno FOR INSERT AS

    PRINT 'Iniciar introdução de novos alunos'
```

Exemplo de um Trigger que não permite apagar os registos da tabela aluno utilizando ROLLBACK TRAN para cancelar a operação de DELETE efectuada inicialmente.

```
SELECT * FROM aluno
DELETE FROM aluno WHERE numaluno > 2
/* -- criar o trigger e comentar novamente
CREATE TRIGGER anulatransaccao ON aluno
FOR DELETE
AS
BEGIN
 ROLLBACK TRAN
PRINT 'Não permito DELETE nesta tabela.'
END
*/
GO
SELECT * FROM aluno
```

Exemplo de uma inserção através de um procedimento armazenado. Esta desencadeia através de um trigger a escrita da data da inserção na primeira tabela na segunda.

```
/* --criar as tabelas o procedimento e o trigger primeiro e depois comentar
CREATE TABLE tabela1 (num INT)
CREATE TABLE tabela2 (data DATETIME)
CREATE PROC ins @num int AS
INSERT INTO tabela1 VALUES (@num)
```

```
CREATE TRIGGER noc ON tabela1 FOR INSERT AS
 INSERT INTO tabela2 VALUES (GETDATE());
 RAISERROR 13122 'inserido valor na tabela 1 directamente e inseri data na
outra tabela usando o trigger';
*/
exec ins 1;
select *from tabela1;
select *from tabela2;
```

Para detectar a acção a realizar pode ser necessário verificar quais as tabelas (inserted, deleted) é que têm registos. É possível também verificar se um determinado campo foi alterado ou inserido através da função UPDATE() Ex.

```
IF UPDATE (ArtigoID) ...
```

Apagar um trigger

Sintaxe

```
DROP TRIGGER <nome_trigger>;
```

Activar um trigger

Sintaxe

```
ENABLE TRIGGER {trigger_name [ ,...n ] | ALL }
ON { object_name | DATABASE | ALL SERVER }
[ ; ]
```

Exemplo da activação de um Trigger

```
ENABLE TRIGGER omeutrigger ON aluno
```

Desactivar um trigger

Sintaxe

```
DISABLE TRIGGER {trigger_name [ ,...n ] | ALL }
ON { object_name | DATABASE | ALL SERVER }
[ ; ]
```

Exemplo da desactivação de um Trigger

```
DISABLE TRIGGER omeutrigger ON aluno
```

4 Exercícios

4.1 Alterar os exercícios seguintes de modo a incluir o processamento de erros com TRY e CATCH e RAISERRROR nos exercícios efectuados na aula passada:

4.2 Criar o procedimento insere_disciplina para inserir novas disciplinas.

4.3 Criar o procedimento insere_aluno para inserir novos alunos.

4.4 Criar o procedimento insere_nota para inserir as notas dos alunos.

4.5 Criar o procedimento melhor_nota para mostrar a melhor nota de uma disciplina.

4.6 Criar o procedimento melhores_notas para mostrar as melhor nota de todas as disciplinas.

4.7 Criar o procedimento para pior_nota para mostrar a pior nota de uma disciplina.

4.8 Criar o procedimento melhor_disciplina que deve devolver o código da disciplina com a melhor média.

4.9 Criar o procedimento alterar_nota para alterar a nota de um aluno a uma disciplina.

4.10 Criar um procedimento media_disciplina para mostrar a média das notas a todas as disciplinas.

4.11 Criar uma tabela de inscrição dos alunos na escola, nesta deve incluir a data (datetime) e estabelecer a ligação à tabela alunos.

4.12 Faça um trigger para quando efectuar uma inscrição mostre o número de alunos existentes na escola.

4.13 Faça um trigger que preencha o registo na tabela de inscrição automaticamente usando os dados da tabela alunos.

4.14 Faça um trigger para mostrar um alerta sempre que uma disciplina tenha média negativa.

4.15 Faça um trigger para cancelar os registos efectuados na tabela disciplinas que não respeitem as regras de preenchimento. Estas regras são as seguintes: Disciplinas do 1ºano começam com 1 até 199, as do 2ºano começam no 200 e terminam no 299 e as do 3º ano começam no 300 e terminam no 399, todos os outros códigos são para serem ignorados.

4.16 Faça um trigger para alertar automaticamente sempre que uma disciplina tenha mais de 5 alunos com mais de 18 valores.

4.17 Faça um trigger para alertar quando o número de alunos da Covilhã atinja 30%.