José Antônio da Cunha

CEFET-RN

Natal, 10 de maio de 2013

Auditando operações DML com OUPUT

Imagine o seguinte cenário: você possui uma tabela muito importante em seu banco de dados e gostaria de implantar uma auditoria de modificação de dados para armazenar todas as informações referente a operação DML (Data Manipulation Language), executadas sobre a tabela. O objetivo é manter um histórico dos dados alterados e excluídos da tabela e principalmente quem executou a ação.

Bom, uma técnica bastante utilizada por quem administra o SQL Server 2000 ou superior é o Mirror Table ou "Tabela espelho".

O problema dessa técnica é que exige a criação sobre a tabela auditada, de uma trigger que monitore as operações de Insert, Update e Delete, e dependendo da utilização da tabela o desempenho de seu sistema pode ser significativamente prejudicado.

Para resolver este problema o SQL Server 2005 possui a nova cláusula OUTPUT, que pode ser usada em conjunto com os comandos DML de Insert, Update ou Delete, para retornar os registros afetados pela operação.

Conhecendo o método de espelhamento de tabela

A técnica Mirror Table consiste em criar duas tabelas idênticas e criar na tabela principal, trigger de Insert, Update e Delete para que na execução de uma dessas operações, a trigger seja disparada e armazene na tabela espelho os dados afetados pela operação.

Para demonstrar como essa técnica funciona, vamos supor que precisamos monitorar todas as alterações realizadas sobre a tabela Products do banco de dados Northwind. O primeiro passo é criar uma tabela espelho que tenha exatamente as mesmas colunas da tabela Products.

O Script da **Listagem 1** criar a tabela Products_Espelho.

Listagem 1

```
CREATE TABLE [dbo].[Products_Espelho](
        [ProductID] [int] IDENTITY(1,1) NOT NULL,
        [ProductName] [nvarchar](40) NOT NULL,
        [SupplierID] [int] NULL,
        [CategoryID] [int] NULL,
```

```
    [QuantityPerUnit] [nvarchar](20) NULL,
    [UnitPrice] [money] NULL,
    [UnitsInStock] [smallint] NULL,
    [UnitsOnOrder] [smallint] NULL,
    [ReorderLevel] [smallint] NULL,
    [Discontinued] [bit] NOT NULL
) ON [PRIMARY]
GO
```

Após a execução do Script temos então uma tabela idêntica à Products para armazenar as informações auditadas. No entanto, a auditoria não fará sentido se não tiver informações sobre o comando executado, quem o executou e a data em que a operação foi executada.

Sendo assim, utilize o seguinte script para adicionar mais três colunas na tabela:

```
ALTER TABLE [dbo].[Products_Espelho]
ADD DataAlteracao datetime,
   ComanndoDML varchar(6),
   Usuario varchar(50)
```

Criada a tabela espelho, vamos então criar uma trigger sobre a tabela Products, conforme a **Listagem 2**.

Listagem 2

```
CREATE TRIGGER Products_Update_Delete ON Products
FOR UPDATE, DELETE
AS
--Pega os novos registros
INSERT Products_Espelho SELECT *, getdate(), 'INSERT', suser_sname()
FROM INSERTED

--Pega os registros antigos ou excluídos
```

```
INSERT Products_Espelho SELECT *, getdate(), 'DELETE', suser_sname()

FROM DELETED

RETURN

GO
```

Para demonstrar o funcionamento da auditoria, vamos inserir alguns registros na tabela products.

```
INSERT INTO Products (ProductName, SupplierID, CategoryID, QuantityPerUnit,
UnitPrice, UnitsInStock, UnitsOnOrder, ReorderLevel, Discontinued)
VALUES ('Batata doce',1,1,'10 caixas',100,50,0,0,0)
```

Se você executar o comando select a seguir, verá que o comando INSERT acima não foi auditada, como esperado, uma vez que o trigger é de UPDATE ou DELETE.

```
select * from Products_Espelho
```

Com o script a seguir, vamos verificar o comportamento do trigger, alterando o preço unitário do produto de R$100,00 para R$95,00.

```
UPDATE Products SET UnitPrice = 95
WHERE ProductName = 'Batata doce'
GO
```

Ao efetuarmos um SELECT na tabela espelho (Products_Espelho), podemos ver o resultado da auditoria para uma operação de Update.

Para demonstrar o comportamento da trigger durante uma operação de Delete execute o seguinte script:

```
DELETE FROM Products WHERE ProductName = 'Batata doce'
```

Como vimos, a técnica de auditoria com tabela espelho é bastante simples e pode ser muito útil em algumas situações. No entanto, você deve ficar atento, pois o uso de triggers em aplicações que possuem muitas operações de escritas, normalmente, geram freqüentes bloqueios de conexões, ocasionando possível lentidão para o usuário.

Por outro lado, esse tipo de auditoria provoca uma sobrecarga extra. Note que para cada operação de Update está sendo executadas duas operações Insert sobre a tabela espelho. Temos então duas vezes mais sobrecarga para cada transação de modificação de dados.

Para um sistema que possui uma alta utilização,isso pode causar um significativo impacto no desempenho do servidor.

Auditando operações DML com a cláusula OUTPUT

No SQL Server a partir da versão 2005, temos uma nova cláusula OUTPUT que oferece uma alternativa mais inteligente para auditar operações DML executadas sobre uma tabela. Com a nova cláusula, podemos monitorar as operações de Insert, Update e Delete sem fazer uso do trigger.

Para demonstrar a utilização da cláusula OUTPUT, vamos criar uma nova tabela no banco de dados AdventureWorks e inserir alguns registros de teste. Veja o script da **Listagem 4**.

Listagem 4

```
CREATE TABLE dbo.TB_Salario
(
 SalarioID int identity(1,1) not null primary key,
 FuncionarioID int not null,
 Salario money not null,
 UltimaAtualizacao datetime not null default getdate()
)
GO
INSERT INTO TB_Salario VALUES (1245, 4800.00,'15/01/2007')
INSERT INTO TB_Salario VALUES (4512, 6700.00,'15-01-2007 17:30:00')
INSERT INTO TB_Salario VALUES (2543, 3200.00,'15-01-2007 17:30:00')
INSERT INTO TB_Salario VALUES (2544, 3400.00,'15-01-2007 17:30:00')
INSERT INTO TB_Salario VALUES (2545, 3800.00,'15-01-2007 17:30:00')
```

Criada a tabela a ser auditada, utilize o script da **Listagem 5** para criar a tabela espelho.

Listagem 5

```
CREATE TABLE dbo.TB_Salario_Espelho
(
 AuditoriaID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
 ComandoDML varchar(10),
 Insert_SalarioID int NULL,
 Delete_SalarioID int NULL,
 Insert_FuncionarioID int NULL,
 Delete_FuncionarioID int null,
 Insert_Salario money NULL,
 Delete_Salario money NULL,
 Insert_UltimaAtualizacao datetime NULL,
 Delete_UltimaAtualizacao datetime NULL,
 Insert_Usuario varchar(10) NULL DEFAULT SUSER_SNAME(),
 Delete_usuario varchar(10) NULL DEFAULT SUSER_SNAME()
)
GO
```

Uma observação muito importante sobre essa tabela é que para receber as linhas da cláusula OUTPUT ela não pode possuir triggers, Check Constraints ou qualquer Foreign Key referenciando suas colunas ou colunas de outra tabela.

Como você deve ter notado no script de criação da tabela TB_Salario_Espelho, essa possui duas versões para as colunas da tabela TB_Salario, a coluna com o prefixo INSERT armazenará os dados inseridos e a coluna com o prefixo DELETE os dados excluídos. Outro ponto de observação são as duas últimas colunas do script, que servem para armazenar o nome do usuário.

Agora que você já tem toda a estrutura necessária para fazer auditoria com a cláusula OUTPUT, então escreva os scripts para fazer auditoria para cada comando de INSERT, UPDATE e DELETE na tabela tb_Salario usando a cláusula OUTPUT do SQL Server.

1 - Script para o comando INSERT

Resposta:

2 - Script para o comando UPDATE

Resposta:

3 - Script para o comando DELETE