Sintaxe e Comparação de FUNCTION, PROCEDURE e TRIGGER

1 Introdução

No T-SQL, `FUNCTIONS`, `PROCEDURES` e `TRIGGERS` são três tipos de objetos que permitem a modularização e automação de tarefas. Cada um tem sua própria sintaxe e características específicas. Este documento destaca a sintaxe e as principais diferenças entre eles.

2 FUNCTION

As funções (FUNCTIONS) retornam um valor e podem ser usadas em consultas SQL. Elas não devem modificar o estado do banco de dados.

2.1 Sintaxe Geral de uma Função

```
CREATE FUNCTION [nome_da_funcao]
(
    @param1 [tipo],
    @param2 [tipo]
)
RETURNS [tipo_de_retorno]
AS
BEGIN
    -- Declaração de variáveis locais, se necessário
    DECLARE @variavel [tipo];

    -- Lógica da função
    SET @variavel = [expressão];

    -- Retorno do valor
    RETURN @variavel;
END;
```

2.2 Componentes da Sintaxe

- `CREATE FUNCTION [nome_da_funcao]`: Define o nome da função a ser criada.
- `(@param1 [tipo], @param2 [tipo])`: Declara os parâmetros da função. Eles são usados para passar valores para a função e devem ser especificados com um tipo.
- `RETURNS [tipo_de_retorno]`: Especifica o tipo de dado que a função retornará.

- `AS BEGIN ... END`: Define o bloco de código que será executado quando a função for chamada. O código dentro deste bloco pode incluir declarações de variáveis, lógica e cálculos.

2.3 Exemplo

```sql
CREATE FUNCTION dbo.NomeDaFuncao (@param1 INT, @param2 INT)
RETURNS INT
AS
BEGIN
    RETURN @param1 + @param2;
END;
```

2.4 Uso

Funções podem ser usadas diretamente em consultas:

```sql
SELECT dbo.NomeDaFuncao(5, 10);
```

3 PROCEDURE

Procedimentos armazenados (PROCEDURES) são blocos de código que podem executar operações complexas e modificar o banco de dados.

3.1 Sintaxe Geral de uma Procedure

```sql
CREATE PROCEDURE [nome_da_procedure]
    @param1 [tipo] [opcional] = [valor_default] [OUTPUT se for valor de retorno],
    @param2 [tipo] [opcional] = [valor_default] [OUTPUT se for valor de retorno]
AS
BEGIN
    -- Corpo da procedure
    -- Código T-SQL que será executado quando a procedure for chamada

    -- Exemplos de comandos:
    -- SELECT * FROM [nome_da_tabela];
    -- UPDATE [nome_da_tabela] SET coluna = valor WHERE condição;
END;
```

3.2 Componentes da Sintaxe

- `CREATE PROCEDURE [nome_da_procedure]`: Define o nome da procedure a ser criada.
- `@param1 [tipo] [opcional] = [valor_default]`: Define um parâmetro para a procedure. Parâmetros são usados para passar valores para a procedure. Eles podem ter um tipo definido e um valor padrão opcional.

- `AS BEGIN ... END`: Define o bloco de código que será executado quando a procedure for chamada. O código T-SQL dentro deste bloco pode incluir qualquer comando SQL válido.

3.3 Exemplo

```
CREATE PROCEDURE NomeDoProcedimento
    @param1 FLOAT,
    @param2 INT OUTPUT
AS
BEGIN
    UPDATE Tabela
    SET Coluna = @param1;

    SET @param2 = (SELECT COUNT(*) FROM Tabela);
END;
```

3.4 Uso

Procedimentos são executados usando:

```
DECLARE @param2 INT;
EXEC NomeDoProcedimento 5, @param2 OUTPUT;
PRINT(@param2);
```

4 TRIGGER

Trigers são procedimentos especiais que são automaticamente executados em resposta a eventos de modificação de dados.

4.1 Sintaxe Geral de uma Trigger

```
CREATE TRIGGER [nome_da_trigger]
ON [nome_da_tabela]
AFTER | INSTEAD OF [evento]
AS
BEGIN
    -- Corpo da trigger
    -- Código T-SQL que será executado quando a trigger for disparada
END;
```

4.2 Componentes da Sintaxe

- `CREATE TRIGGER [nome_da_trigger]`: Define o nome da trigger a ser criada.

- `ON [nome_da_tabela]`: Especifica a tabela ou visão sobre a qual a trigger será criada.
- `FOR | AFTER | INSTEAD OF [evento]`: Define o tipo de evento que disparará a trigger. `FOR` indica que a trigger disparará juntamente ao evento. `AFTER` indica que a trigger será executada após o evento especificado, enquanto `INSTEAD OF` indica que a trigger substituirá o evento especificado. Os eventos podem incluir `INSERT`, `UPDATE`, ou `DELETE`.
- `BEGIN ... END`: Define o bloco de código que será executado quando a trigger for disparada.

4.3 Exemplo

```
CREATE TRIGGER trg_AposInsert
ON Tabela
AFTER INSERT
AS
BEGIN
    INSERT INTO LogTabela (DataHora, Descricao)
    VALUES (GETDATE(), 'Novo registro inserido.');
END;
```

5 Tabelas Virtuais em Triggers

No T-SQL, as triggers utilizam duas tabelas virtuais principais para acessar os valores antes e depois da modificação:

- **INSERTED**: Contém as linhas que foram inseridas ou atualizadas com novos valores. Usada em triggers `AFTER INSERT` e `AFTER UPDATE`.
- **DELETED**: Contém as linhas que foram removidas ou atualizadas com valores antigos. Usada em triggers `AFTER DELETE` e `AFTER UPDATE`.

6 Exemplos de Uso

6.1 Trigger AFTER INSERT

```
CREATE TRIGGER trg_AfterInsert
ON dbo.tb_serie_temporal
AFTER INSERT
AS
BEGIN
    -- Exemplo: Log de inserção
    INSERT INTO dbo.Log (Action, Value, LogDate)
    SELECT 'Inserted', jun, SYSDATETIME()
    FROM INSERTED;
END;
```

6.2 Trigger AFTER DELETE

```
CREATE TRIGGER trg_AfterDelete
ON dbo.tb_serie_temporal
AFTER DELETE
AS
BEGIN
    -- Exemplo: Log de exclusão
    INSERT INTO dbo.Log (Action, Value, LogDate)
    SELECT 'Deleted', jun, SYSDATETIME()
    FROM DELETED;
END;
```

6.3 Trigger AFTER UPDATE

```
CREATE TRIGGER trg_AfterUpdate
ON dbo.tb_serie_temporal
AFTER UPDATE
AS
BEGIN
    -- Exemplo: Log de atualização
    INSERT INTO dbo.UpdateLog (OldValue, NewValue, UpdateDate)
    SELECT d.jun, i.jun, SYSDATETIME()
    FROM DELETED d
    JOIN INSERTED i ON d.id = i.id;
END;
```

6.4 Uso

Triggers são acionados automaticamente quando ocorre um evento especificado (como um `INSERT`, `UPDATE`, ou `DELETE`) na tabela associada.

7 Comparação

Na Tabela 1, é feita uma comparação entre os módulos personalizáveis do T-SQL.

8 Conclusão

Cada um desses objetos no T-SQL tem suas aplicações específicas. Funções são úteis para cálculos e manipulação de dados dentro de consultas, procedimentos armazenados são ideais para tarefas complexas e modificações de dados, e triggers são utilizados para automatizar reações a eventos no banco de dados.

Aspecto	**FUNCTION**	**PROCEDURE**	**TRIGGER**
Retorno de Valor	Sempre retorna um valor	Não retorna valores diretamente	Não retorna valores diretamente
Uso em Consultas	Pode ser usada em consultas SQL	Não pode ser usada diretamente em consultas	Não é usada em consultas SQL
Modificação de Dados	Não modifica dados	Pode modificar dados	Modifica dados em resposta a eventos
Parâmetros	Aceita parâmetros de entrada	Aceita parâmetros de entrada e saída	Não usa parâmetros explicitamente

Table 1: Comparação entre FUNCTION, PROCEDURE e TRIGGER