Sistema Fiep

FIEP SESI SENAI IEL

Banco de Dados

Objetos Programáveis de Banco de Dados

Objetos Programáveis de Banco de Dados

- *Views*.
- Variáveis.
- Lotes.
- Elementos de fluxo IF... ELSE e WHILE.
- *Functions.*
- *Stored Procedures.*
- *Triggers.*

Views

Uma *view* pode ser definida como uma “tabela virtual” cujo conteúdo (colunas e linhas) é definido por uma consulta. Pode ser utilizada para criar uma exibição dos dados em uma ou mais tabelas no banco de dados.

```
use DatabaseTeste
GO
create view Sales.ClientesUSA
AS
select custid, companyname, contacttitle, address, city, region
from Sales.Customers
where Sales.Customers.country = 'USA';
```

Variáveis

Sistema Fiep FIEP SESI SENAI IEL

As variáveis permitem armazenar valores de dados temporariamente para uso posterior no mesmo lote onde foram declaradas. No *SQL Server* usa-se a instrução *DECLARE* para declarar uma ou mais variáveis e use a instrução *SET* para atribuir um valor a uma única variável.

```
--Variaveis
DECLARE @inteira AS int;
SET @inteira = 10;

--Na mesma linha da declaração
DECLARE @inteira2 AS int = 10;

--Declarar uma varivel escalar, com resultado de uma expressao escalar
DECLARE @empregado AS varchar(100);
SET @empregado = (SELECT firstname + N' ' + lastname FROM HR.Employees WHERE empid = 5);
GO
SELECT @empregado AS nome_empregado;
```

Variáveis

O *SQL Server* também dá suporte a uma instrução SELECT de atribuição não padronizada, que permite consultar dados e atribuir vários valores obtidos da mesma linha para várias variáveis usando uma única instrução.

```
--Declaracao nao padronizada para instrucao SELECT
DECLARE @primeironome AS varchar(100), @ultimonome AS varchar(100);

SELECT
@primeironome = firstname,
@ultimonome = lastname
FROM HR.Employees
WHERE empid = 5;

SELECT @primeironome AS primeironome, @ultimonome AS ultimonome;
```

Variáveis

A instrução *SET* é mais segura que a instrução *SELECT*, porque requer que seja utilizada uma consulta escalar para recuperar os dados de uma tabela.

```
--Declaracao com SET
DECLARE @nomeempregado AS varchar(100);

SET @nomeempregado = (SELECT firstname + N' ' + lastname FROM HR.Employees WHERE empid = 5)

SELECT @nomeempregado AS nome_empregado;
```

Lotes

Um lote é uma ou mais instruções T-SQL enviadas por um aplicativo cliente para ser executado no SQL Server, como uma única unidade de execução. O lote passa por uma análise (verificação de sintaxe), validação (verificando a existência de objetos e colunas referenciadas, bem como as permissões) e otimização como uma unidade.

```
--Lote invalido
PRINT 'Lote 2';
select custid from Sales.Customers;
select orderid fom Sales.Orders;
GO
--Lote valido
PRINT 'Lote 3';
select empid from HR.Employees;
GO
```

Lotes e Variáveis

As variáveis são locais para os lotes em que elas são definidas. Caso for tentado fazer referencia a uma variável fora do lote onde foi definida, o SGBD retornará um erro.

```
--Lote e variaveis
--Correto
--Dentro do lote
DECLARE @numero AS INT;
SET @numero = 10;
PRINT @numero;
GO

--Falha
--Fora do lote
PRINT @numero;
GO
```

Execução assíncrona de lotes

Para uma execução assicrona do lote é preciso utilizar a opção *GO.*

```
use DatabaseTeste
GO

IF OBJECT_ID('Sales.vw_ordens', 'V') IS NOT NULL DROP VIEW Sales.vw_ordens;
GO

CREATE VIEW Sales.vw_ordens
AS
SELECT YEAR(orderdate) AS orderyear, COUNT(*) AS numorders
FROM Sales.Orders
GROUP BY YEAR(orderdate);
GO
```

Elementos de fluxo IF ... ELSE e WHILE

Os elementos de fluxo permitem controlar o fluxo de seu código. Na T-SQL é possível utilizar os elementos de fluxo IF ... ELSE e WHILE.

```
IF YEAR(CURRENT_TIMESTAMP) <> YEAR(DATEADD(day, 1, CURRENT_TIMESTAMP))
  PRINT 'Hoje é o ultimo dia do ano.'
ELSE
  PRINT 'Hoje não é o ultimo dia dos ano.'
GO

IF YEAR(CURRENT_TIMESTAMP) <> YEAR(DATEADD(day, 1, CURRENT_TIMESTAMP))
  PRINT 'Hoje é o ultimo dia do ano.'
ELSE
  IF MONTH(CURRENT_TIMESTAMP) <> MONTH(DATEADD(day, 1, CURRENT_TIMESTAMP))
    PRINT 'Hoje é o ultimo dia do mês, mas não é o ultimo dia do ano.'
  ELSE
    PRINT 'Hoje não é o ultimo dia do mês';
GO
```

Elementos de fluxo IF ... ELSE e WHILE

Sistema Fiep FIEP SESI SENAI IEL

Caso seja preciso executar mais de uma instrução nas seções *IF* ou *ELSE*, pode ser utilizado bloco de instruções, demarcando os limites do bloco com as palavras-chaves *BEGIN* e *END*.

```
-- Declaracao em bloco
IF DAY(CURRENT_TIMESTAMP) = 1
BEGIN
  PRINT 'Hoje é o primeiro dia do mês'; PRINT ' Iniciando backup full';
  BACKUP DATABASE [DatabaseTeste]
    TO DISK = 'C:\PH\Fiep\CursoBSI_2020_1\Disciplina3pBD\Aulas\Aula12_20200525\DatabaseTeste_Full.bak'
WITH INIT;
  PRINT 'Backup full finalizado';
END
ELSE
BEGIN
  PRINT 'Hoje não é o ultimo dia do mês'
  PRINT 'Iniciando backup diferencial';
  BACKUP DATABASE [DatabaseTeste]
    TO DISK = 'C:\PH\Fiep\CursoBSI_2020_1\Disciplina3pBD\Aulas\Aula12_20200525\DatabaseTeste_Diff.bak'
WITH INIT;
  PRINT 'Backup diferencial finalizado';
END
```


Elementos de fluxo IF ... ELSE e WHILE

A T-SQL fornece o elemento de fluxo *WHILE* para habilitar a execução do código em *loop,* e para sair da execução em algum ponto, pode ser usando o comando *Break* ou, se precisar continuar após uma validação com *IF*, pode usar o comando *Continue*.

```
-- Elemento de fluxo WHILE
DECLARE @numero AS INT;
SET @nu = 1;
WHILE @i <= 10
BEGIN
  PRINT @i;
  SET @i = @i + 1;
END;
GO
```

```
--Break
DECLARE @numero AS INT;
SET @numero = 1
WHILE @numero <= 7
BEGIN
  IF @numero = 6 Break;
  PRINT @numero;
  SET @numero = @numero + 1;
END;
GO
```

Elementos de fluxo IF ... ELSE e WHILE

A T-SQL fornece o elemento de fluxo *WHILE* para habilitar a execução do código em *loop,* e para sair da execução em algum ponto, pode ser usando o comando *Break* ou, se precisar continuar após uma validação com *IF*, pode usar o comando *Continue*.

```
--Continue
DECLARE @numero AS INT;
SET @numero = 0
WHILE @numero < 10
BEGIN
  SET @numero = @numero + 1;
  PRINT @numero;
  IF @numero = 6
  PRINT 'já chegou em ' + convert(char(1), @numero)
Continue;
  PRINT @numero;
END;
GO
```

Elementos de fluxo IF ... ELSE e WHILE

Sistema Fiep FIEP SESI SENAI IEL

```
-- Um exemplo usando IF com WHILE
SET NOCOUNT ON;
USE tempdb;
IF OBJECT_ID('dbo.tbl_Numeros', 'U') IS NOT NULL DROP TABLE dbo.tbl_Numeros;
CREATE TABLE dbo.tbl_Numeros(Cod INT NOT NULL PRIMARY KEY);
GO

DECLARE @Cod AS INT;
SET @Cod = 1;
WHILE @Cod <= 100
BEGIN
  INSERT INTO dbo. tbl_Numeros(Cod) VALUES(@Cod);
  SET @Cod = @Cod + 1;
END
GO
```

Functions

A finalidade de uma função definida pelo usuário é sintetizar a lógica para realização de cálculos, possivelmente baseado em parâmetros de entrada, e retornar um resultado.

```
USE [DatabaseTeste];
IF OBJECT_ID('dbo.fn_idade') IS NOT NULL DROP FUNCTION dbo.fn_idade;
GO
CREATE FUNCTION dbo.fn_idade
(
  @dtnascimento AS DATETIME, @datacalc AS DATETIME
)
RETURNS INT
AS
BEGIN
  RETURN
    DATEDIFF(year, @dtnascimento, @datacalc)
    - CASE WHEN 100 * MONTH(@datacalc) + DAY(@datacalc)
               < 100 * MONTH(@dtnascimento) + DAY(@dtnascimento)
          THEN 1 ELSE 0
      END
END
```

Para testar a função, basta “chama-la” dentro da consulta para calcular a idade de cada empregado conforme abaixo:

```
-- Teste da função fn_idade
SELECT
  empid, firstname, lastname, birthdate,
  dbo.fn_idade(birthdate, CURRENT_TIMESTAMP) AS idade
FROM HR.Employees;
```

Stored Procedures

Os *stored procedures* são rotinas do lado do servidor que utilizam o código T-SQL (no caso do *SQL Server*). Estes "procedimentos armazenados" podem ter parâmetros de entrada e de saída, podem retornar conjuntos de resultados de consultas.

```
--Usando Stored Procedure
USE [DatabaseTeste];
GO
IF OBJECT_ID('Sales.sp_RetornaPedidos', 'P') IS NOT NULL
  DROP PROC Sales.sp_RetornaPedidos;
GO

CREATE PROC Sales.sp_RetornaPedidos
  @idcliente   AS INT, @dataini AS DATETIME = '20000101', @datafin   AS DATETIME = '20200101',
  @resultado  AS INT OUTPUT
AS
SET NOCOUNT ON;

SELECT orderid, custid, empid, orderdate
FROM Sales.Orders
WHERE custid = @idcliente AND orderdate >= @dataini AND orderdate < @datafin;
```

Stored Procedures

Para testar o *stored procedure* basta executar passando os parâmetros necessário conforme o código descrito abaixo:

```
--Teste Stored Procedure
USE [DatabaseTeste];
GO
DECLARE @linhas AS INT;

EXEC Sales.sp_RetornaPedidos
  @idcliente   = 5,
  @dataini = '20070101',
  @datafin   = '20080101',
  @resultado  = @linhas OUTPUT;

SELECT @linhas AS linhas;
GO
```

Stored Procedures

Para testar o *stored procedure* basta executar passando os parâmetros necessário conforme o código descrito abaixo:

```
--Teste Stored Procedure
USE [DatabaseTeste];
GO
DECLARE @linhas AS INT;

EXEC Sales.sp_RetornaPedidos
  @idcliente   = 5,
  @dataini = '20070101',
  @datafin   = '20080101',
  @resultado  = @linhas OUTPUT;

SELECT @linhas AS linhas;
GO
```

Triggers

Sistema Fiep — FIEP SESI SENAI IEL

Trigger é um tipo especial de objeto programável que não pode ser executado explicitamente. Ele deve ser acionado a partir de um evento que ocorra no banco de dados, ou seja, está associado a um evento.

```
-- Exemplo de um trigger de auditoria para eventos DML
USE tempdb;
GO
IF OBJECT_ID('dbo.tb_Audita', 'U') IS NOT NULL DROP TABLE dbo.tb_Audita;
IF OBJECT_ID('dbo.tb_Alvo', 'U') IS NOT NULL DROP TABLE dbo.tb_Alvo;

CREATE TABLE dbo.tb_Alvo
(id  INT NOT NULL PRIMARY KEY, nome VARCHAR(10) NOT NULL);

CREATE TABLE dbo.tb_Audita
(
  audit_lsn  INT         NOT NULL IDENTITY PRIMARY KEY,
  dt         DATETIME    NOT NULL DEFAULT(CURRENT_TIMESTAMP),
  login_name sysname     NOT NULL DEFAULT(SUSER_SNAME()),
  id INT          NOT NULL,
  nome VARCHAR(10) NOT NULL
);
GO
```

```
-- Continuação do Exemplo de um trigger de auditoria para eventos DML

CREATE TRIGGER trg_tb_Alvo_insert_audit ON dbo.tb_Alvo AFTER INSERT
AS
SET NOCOUNT ON;

INSERT INTO dbo.tb_Audita(id, nome)
  SELECT id, nome FROM inserted;
GO

INSERT INTO dbo.tb_Alvo(id, nome) VALUES(100, 'abc');
INSERT INTO dbo.tb_Alvo(id, nome) VALUES(200, 'efg');
INSERT INTO dbo.tb_Alvo(id, nome) VALUES(300, 'hij');

SELECT audit_lsn, dt, login_name, id, nome
FROM dbo.tb_Audita;
GO
```

Atividades

1. Retorne os pedidos estabelecidos em junho de 2007. Banco de Dados (DatabaseTeste); Tabelas (Sales.Orders).
2. Retorne os pedidos estabelecidos no último dia do mês. Banco de Dados (DatabaseTeste); Tabelas (Sales.Orders).
3. Retorne os funcionários cujos sobrenomes contenham a letra ‘a’ duas vezes ou mais. Banco de Dados (DatabaseTeste); Tabelas (HR.Employees).
4. Retorne os pedidos com valor total (quantidade * preço unitário) maior que 10.000, classificados por valor total. Banco de Dados (DatabaseTeste); Tabelas (Sales.OrderDetails).
5. Retorne 3 países de entrega com maiores fretamentos médios de 2007. Banco de Dados (DatabaseTeste); Tabelas (Sales.Orders).
6. Estabeleça um SELECT que retorne para cada funcionário o gênero com base no título de cortecia *(titleofcourtesy).* Para ‘Ms.’ e ‘Mrs.’ retorne ‘Feminino’, para ‘Mr.’ retorne ‘Masculino’ e em todos os outros casos (por exemplo, ‘Dr.’) retorne ‘Desconhecido’. Banco de Dados (DatabaseTeste); Tabelas (HR.Employees).

Sistema Fiep

FIEP
SESI
SENAI
IEL

nosso i é de indústria.