Хранимые процедуры

#Синтаксис

```
CREATE PROCEDURE имя_процедуры

[ {@имя_параметра тип_данных} ] [= по умолчанию] [OUTPUT] [,...,n]

AS оператор(ы) t-sql
```

Сценарии (Scripts)

Сценарии содержат набор SQL-операторов в текстовом файле, который можно многократно запускать. Основные операторы сценария включают:

- **USE** – выбор текущей базы данных для работы.

  ```
  USE MYDB;
  ```

- **DECLARE** – объявление локальных переменных, которые используются в сценарии.

  ```
  DECLARE @Ident int;
  ```

- **INSERT INTO** – добавление данных в таблицу.

  ```
  INSERT INTO Orders (CustomerNo, OrderDate, EmployeeID) VALUES
  (1, GETDATE(), 1);
  ```

- **SELECT** – используется для присваивания значения переменной или получения данных.

```
SELECT @Ident = @@IDENTITY;
```

Пример сценария для вставки строки:

```
USE MYDB;
DECLARE @Ident int;
INSERT INTO Orders (CustomerNo, OrderDate, EmployeeID) VALUES (1,
GETDATE(), 1);
SELECT @Ident = @@IDENTITY;
INSERT INTO OrdersDetails (OrderID, PartNo, Description,
UnitPrice, Qty) VALUES (@Ident, '2R2416', 'Cylinder Head', 1300,
2);
```

Пакеты (Batches)

Пакет – это логическая группа операторов, выполняемая как единое целое. Операторы в пакете обрабатываются SQL Server вместе.

- **Оператор GO** используется для разделения пакетов.

```
-- Команда GO завершает текущий пакет и запускает его
выполнение
GO
```

Операторы управления ходом выполнения

IF...ELSE

Условный оператор для выполнения действий при истинности или ложности условия.

```
IF (@condition = 1)
    PRINT 'Condition is true';
ELSE
    PRINT 'Condition is false';
```

CASE

Позволяет обрабатывать несколько условий в одном выражении.

```
SELECT CASE
          WHEN @value = 1 THEN 'One'
          WHEN @value = 2 THEN 'Two'
          ELSE 'Other'
       END AS Result;
```

GOTO

Оператор безусловного перехода, который передает выполнение к указанной метке.

```
BEGIN
    PRINT 'Starting...';
    GOTO Label;
    PRINT 'This will not be printed';
Label:
    PRINT 'Jumped to Label';
END;
```

WHILE

Организация цикла с проверкой условия.

```
DECLARE @Counter int = 0;
WHILE @Counter < 5
```

```
BEGIN
    PRINT 'Counter: ' + CAST(@Counter AS varchar);
    SET @Counter = @Counter + 1;
END;
```

TRY...CATCH

Для обработки ошибок.

```
BEGIN TRY
    -- Код с возможной ошибкой
    SELECT 1 / 0; -- ошибка деления на ноль
END TRY
BEGIN CATCH
    PRINT 'An error occurred: ' + ERROR_MESSAGE();
END CATCH;
```

Хранимые процедуры (Stored Procedures)

Хранимая процедура – это набор SQL-операторов, который компилируется и хранится на сервере. Они позволяют выполнять многократные задачи и возвращать результаты.

Создание процедуры (CREATE PROCEDURE)

Пример простой процедуры для выбора данных:

```
USE Northwind;
GO
CREATE PROCEDURE GetLateShipments
AS
    SELECT RequiredDate, ShippedDate
    FROM Orders
    WHERE ShippedDate > RequiredDate;
GO
```

Входные параметры

Для передачи данных в процедуру используются входные параметры, перед именем которых ставится `@`.

```
CREATE PROCEDURE GetOrderById @OrderId int
AS
    SELECT * FROM Orders WHERE OrderID = @OrderId;
GO
```

Выходные параметры (OUTPUT)

Процедура может возвращать значения через выходные параметры.

```
CREATE PROCEDURE GetUnitPrice @ProductId int, @UnitPrice money
OUTPUT
AS
    SELECT @UnitPrice = UnitPrice FROM Products WHERE ProductID =
@ProductId;
GO

DECLARE @Price money;
EXEC GetUnitPrice 1, @UnitPrice = @Price OUTPUT;
PRINT 'Unit Price: ' + CAST(@Price AS varchar);
```

Изменение процедуры (ALTER PROCEDURE)

Используется для изменения существующей процедуры.

```
ALTER PROCEDURE GetLateShipments
AS
    SELECT RequiredDate, ShippedDate, CompanyName
    FROM Orders, Shippers
    WHERE Orders.ShipVia = Shippers.ShipperID AND ShippedDate >
```

```
RequiredDate;
GO
```

Удаление процедуры (DROP PROCEDURE)

Для удаления процедуры используется оператор `DROP PROCEDURE`.

```
DROP PROCEDURE GetLateShipments;
```

Примеры процедур с условиями и циклами

1. **Процедура с циклом и условием:**

```
CREATE PROCEDURE InsertRows @StartValue int
AS
    DECLARE @Counter int = 0;
    WHILE @Counter < 5
    BEGIN
        INSERT INTO MyTable (Column1, Column2) VALUES
(@StartValue + @Counter, 'NewRow');
        SET @Counter = @Counter + 1;
    END;
GO
```

2. **Процедура с проверкой входного значения:**

```
CREATE PROCEDURE CheckProductPrice @ProductId int
AS
    IF @ProductId IS NULL
    BEGIN
        PRINT 'Product ID is missing!';
        RETURN;
    END
    ELSE
    BEGIN
```

```
        SELECT UnitPrice FROM Products WHERE ProductID =
@ProductId;
    END;
GO
```