# **Лабораторна робота №9. Збережені процедури**

## Створення та виконання процедур

Нерідко операція з даними представляє набір інструкцій, які необхідно виконати в певній послідовності. Наприклад, при додаванні купівлі товару необхідно внести дані в таблицю замовлень. Однак перед цим треба перевірити, а чи є товар, що купується в наявності. Можливо, при цьому знадобиться перевірити ще ряд додаткових умов. Тобто фактично процес покупки товару охоплює кілька дій, які повинні виконуватися в певній послідовності. І в цьому випадку більш оптимально буде инкапсулировать всі ці дії в один об'єкт - збережену процедуру (stored procedure).

Тобто по суті збережені процедури представляє набір інструкцій, які виконуються як єдине ціле. Тим самим збережені процедури дозволяють спростити комплексні операції і винести їх в єдиний об'єкт. Чи зміниться процес покупки товару, відповідно буде змінити код процедури. Тобто процедура також спрощує управління кодом.

Також збережені процедури дозволяють обмежити доступ до даних в таблицях і тим самим зменшити ймовірність навмисних або неусвідомлених небажаних дій щодо цих даних.

І ще один важливий аспект - продуктивність. Збережені процедури зазвичай виконуються швидше, ніж звичайні SQL-інструкції. Все тому що код процедур компілюється один раз при першому її запуску, а потім зберігається в компільованою формі.

Для створення збереженої процедури застосовується команда CREATE PROCEDURE або CREATE PROC.

Таким чином, процедура, що зберігається має три ключові особливості: спрощення коду, безпеку і продуктивність.

Наприклад, нехай в базі даних є таблиця, яка зберігає дані про товари:

|  |
| --- |
| CREATE TABLE Products  (  Id INT IDENTITY PRIMARY KEY,  ProductName NVARCHAR (30) NOT NULL,  Manufacturer NVARCHAR (20) NOT NULL,  ProductCount INT DEFAULT 0,  Price MONEY NOT NULL  ); |

Створимо збережену процедуру для отримання даних з цієї таблиці:

|  |
| --- |
| USE productsdb;  GO  CREATE PROCEDURE ProductSummary AS  SELECT ProductName AS Product, Manufacturer, Price  FROM Products |

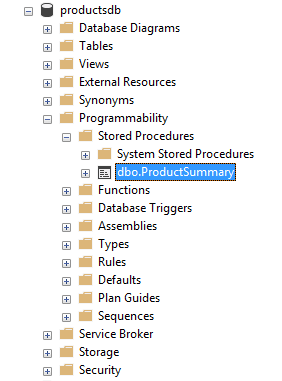
Оскільки команда CREATE PROCEDURE повинна викликатися в окремому пакеті, то після команди USE, яка встановлює поточну базу даних, використовується команда GO для визначення нового пакета.

Після імені процедури повинно йти ключове слово AS.

Для відділення тіла процедури від іншої частини скрипта код процедури нерідко поміщається в блок BEGIN ... END:

|  |
| --- |
| USE productsdb;  GO  CREATE PROCEDURE ProductSummary AS  BEGIN  SELECT ProductName AS Product, Manufacturer, Price  FROM Products  END; |

Після додавання процедури ми її можемо побачити в вузлі бази даних в SQL Server Management Studio в Підвузли Programmability -> Stored Procedures:

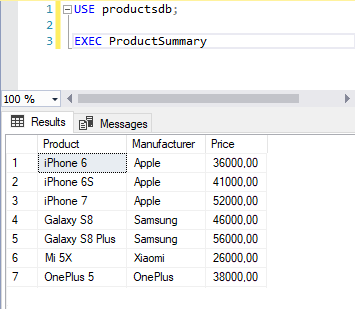


І ми зможемо управляти процедурою також і через візуальний інтерфейс.

**Виконання процедури**

Для виконання процедури викликається команда EXEC або EXECUTE:

|  |
| --- |
| EXEC ProductSummary |



**Видалення процедури**

Для видалення процедури застосовується команда DROP PROCEDURE:

|  |
| --- |
| DROP PROCEDURE ProductSummary |

## Параметри в процедурах

Процедури можуть приймати і передавати параметри. Параметри бувають вхідними - з їх допомогою в процедуру можна передати деякі значення. І також параметри бувають вихідними - вони дозволяють повернути з процедури деяке значення.

Наприклад, нехай в базі даних буде наступна таблиця Products:

|  |
| --- |
| USE productsdb;  CREATE TABLE Products  (  Id INT IDENTITY PRIMARY KEY,  ProductName NVARCHAR (30) NOT NULL,  Manufacturer NVARCHAR (20) NOT NULL,  ProductCount INT DEFAULT 0,  Price MONEY NOT NULL  ); |

Визначимо процедуру, яка буде додавати дані в цю таблицю:

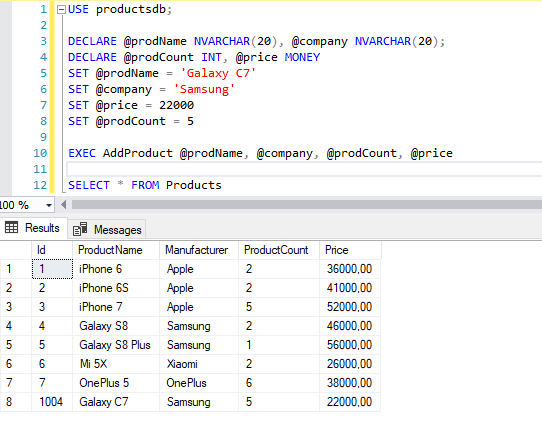
|  |
| --- |
| USE productsdb;  GO  CREATE PROCEDURE AddProduct  @name NVARCHAR (20),  @manufacturer NVARCHAR (20),  @count INT,  @price MONEY  AS  INSERT INTO Products (ProductName, Manufacturer, ProductCount, Price)  VALUES (@name, @manufacturer, @count, @price) |

Після назви процедури йде список вхідних параметрів, які визначаються також як і змінні - назва починається з символу @, а після назви йде тип змінної. І за допомогою команди INSERT значення цих параметрів будуть передаватися в таблицю Products.

Використовуємо цю процедуру:

|  |
| --- |
| USE productsdb;  DECLARE @prodName NVARCHAR (20), @company NVARCHAR (20);  DECLARE @prodCount INT, @price MONEY  SET @prodName = 'Galaxy C7'  SET @company = 'Samsung'  SET @price = 22000  SET @prodCount = 5  EXEC AddProduct @prodName, @company, @prodCount, @price  SELECT \* FROM Products |

Тут передані в процедуру значення визначаються через змінні. При виклику процедури їй через кому передаються значення. При цьому значення передаються параметрам процедури по позиції. Так як першим визначено параметр @name, то йому буде передаватися перше значення - значення змінної @prodName. Другому параметру - @manufacturer передається друге значення - значення змінної @company і так далі. Головне, щоб між переданими значеннями і параметрами процедури було відповідність за типом даних.



Також можна було б передати безпосередньо значення:

|  |
| --- |
| EXEC AddProduct 'Galaxy C7', 'Samsung', 5, 22000 |

Також значення параметрам процедури можна передавати по імені:

|  |
| --- |
| USE productsdb;  DECLARE @prodName NVARCHAR (20), @company NVARCHAR (20);  SET @prodName = 'Honor 9'  SET @company = 'Huawei'  EXEC AddProduct @name = @ProdName,  @ Manufacturer = @ company ,  @count = 3,  @price = 18000 |

При передачі параметрів по імені параметру процедури присвоюється деяке значення.

**Необов'язкові параметри**

Параметри можна відзначати як необов'язкові, привласнюючи їм деяке значення за замовчуванням. Наприклад, в разі вище ми можемо автоматично встановлювати для кількості товару значення 1, якщо відповідне значення не передана в процедуру:

|  |
| --- |
| USE productsdb;  GO  CREATE PROCEDURE AddProductWithOptionalCount  @name NVARCHAR (20),  @manufacturer NVARCHAR (20),  @price MONEY,  @count INT = 1  AS  INSERT INTO Products (ProductName, Manufacturer, ProductCount, Price)  VALUES (@name, @manufacturer, @count, @price) |

При цьому необов'язкові параметри поміщати в кінці списку параметрів процедури.

|  |
| --- |
| DECLARE @prodName NVARCHAR (20), @company NVARCHAR (20), @price MONEY  SET @prodName = 'Redmi Note 5A'  SET @company = 'Xiaomi'  SET @price = 22000  EXEC AddProductWithOptionalCount @prodName, @company, @price  SELECT \* FROM Products |

І в цьому випадку для параметра @count в процедуру можна не передавати значення.

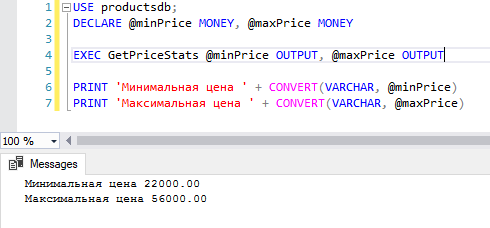
## Вихідні параметри і повернення результату

Вихідні параметри дозволяють повернути з процедури деякий результат. Вихідні параметри визначаються за допомогою ключового слова OUTPUT. Наприклад, визначимо ще одну процедуру:

|  |
| --- |
| USE productsdb;  GO  CREATE PROCEDURE GetPriceStats  @minPrice MONEY OUTPUT,  @maxPrice MONEY OUTPUT  AS  SELECT @minPrice = MIN (Price), @maxPrice = MAX (Price)  FROM Products |

При виклику процедури для вихідних параметрів передаються змінні з ключовим словом OUTPUT:

|  |
| --- |
| USE productsdb;  DECLARE @minPrice MONEY, @maxPrice MONEY  EXEC GetPriceStats @minPrice OUTPUT, @maxPrice OUTPUT  PRINT 'Мінімальна ціна' + CONVERT (VARCHAR, @minPrice)  PRINT 'Максимальна ціна' + CONVERT (VARCHAR, @maxPrice) |



Також можна поєднувати вхідні і вихідні параметри. Наприклад, визначимо процедуру, яка додає новий рядок в таблицю і повертає її id:

|  |
| --- |
| USE productsdb;  GO  CREATE PROCEDURE CreateProduct  @name NVARCHAR (20),  @manufacturer NVARCHAR (20),  @count INT,  @price MONEY,  @id INT OUTPUT  AS  INSERT INTO Products (ProductName, Manufacturer, ProductCount, Price)  VALUES (@name, @manufacturer, @count, @price)  SET @id = @@ IDENTITY |

За допомогою глобальної змінної @@ IDENTITY можна отримати ідентифікатор доданої записи.

При виклику цієї процедури їй також по позиції передаються всі вхідні і вихідні параметри:

|  |
| --- |
| USE productsdb;  DECLARE @id INT  EXEC CreateProduct 'LG V30', 'LG', 3, 28000, @id OUTPUT  PRINT @id |

**Повернення значення**

Крім передачі результату виконання через вихідні параметри збережена процедура також може повертати якесь значення за допомогою оператора RETURN. Хоча дана можливість багато в чому знівельована використанням вихідних параметрів, через які можна повертати результат, тим не менш, якщо треба повернути з процедури одне значення, то цілком можна використовувати оператор RETURN.

Наприклад, повернемо середню ціну на товари:

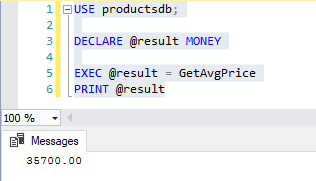
|  |
| --- |
| USE productsdb;  GO  CREATE PROCEDURE GetAvgPrice AS  DECLARE @avgPrice MONEY  SELECT @avgPrice = AVG (Price)  FROM Products  RETURN @avgPrice; |

Після оператора RETURN вказується значення, що повертається. В даному випадку це значення змінної @avgPrice.

Викличемо дану процедуру:

|  |
| --- |
| USE productsdb;  DECLARE @result MONEY  EXEC @result = GetAvgPrice  PRINT @result |

Для отримання результату процедури її значення зберігається в змінну (в даному випадку в змінну @result):



# **Завдання**

1. Створити процедуру, яка автоматизує наповнення таблиці Products даними (дані для INSERT можна взяти в роботі №3)

2. Створити процедуру, яка дозволить вивести Ім’я користувача та всі його замовлення на основі його id, яке буде передаватись до процедури.

3. Створити процедуру, яка дозволить отримати id користувача, з найбільшою кількістю замовлень у місяці MMM року YYY (MMM та YYY передавати при викликанні процедури).

4. За допомогою виклику процедури з завдання 3 як параметра процедури з завдання 2 дізнатись всі дані про користувача найбільшою кількістю замовлень за жовтень 2021 року.

5\*. Зробити завдання 4. З минулої лабораторної роботи (4. Вивести за допомогою циклу суму замовлень за січень, лютий, ..., грудень 2021 року) через одну процедуру у циклі.

6\*. Створити процедуру, яка буде запускати скрипт з задачі 6 Лабораторної роботи № 8 (6. Написати код, який створить таблицю (перед створенням перевірити її наявність та видалити, якщо наявна), в яку вставить 1000 випадкових цілих значень A, B, C та float X1, X2, де A, B, C параметри квадратного тричлену (Ax2 + Bx + C = 0), X1 та X2 – корені (NULL – якщо відсутній корінь))

7\*. Створити процедуру, яка наповнить таблицю Orders випадковими значеннями замовлень на основі таблиць Products та Customers (Для цього створити процедури додавання записів в таблицю товарів та користувачів).

8\*. Написати скрипт, який запустить процедуру з п.7 100000 разів.