**Лабораторна робота №8. Змінні і керуючі конструкції**

## Змінні в T-SQL

Змінна представляє іменований об'єкт, який зберігає деяке значення. Для визначення змінних застосовується вираз DECLARE, після якого вказується назва і тип змінної. При цьому назва локальної змінної має починатися з символу @:

|  |
| --- |
| DECLARE @Назва\_змінної тип\_даних |

Наприклад, визначимо змінну name, яка буде мати тип NVARCHAR:

|  |
| --- |
| DECLARE @name NVARCHAR (20) |

Також можна визначити через кому відразу кілька змінних:

|  |
| --- |
| DECLARE @name NVARCHAR (20), @age INT |

За допомогою виразу SET можна привласнити змінної деяке значення:

|  |
| --- |
| DECLARE @name NVARCHAR (20), @age INT;  SET @name = 'Tom';  SET @age = 18; |

Так як @name надає тип NVARCHAR, тобто рядок, то цієї змінної відповідно і присвоюється рядок. А змінної @age присвоюється число, так як вона являє тип INT.

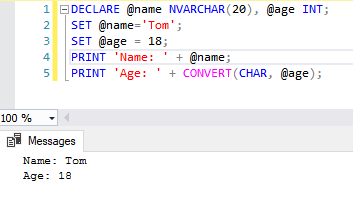
Вираз PRINT повертає повідомлення клієнту. наприклад:

|  |
| --- |
| PRINT 'Hello World' |

І з його допомогою ми можемо вивести значення змінної:

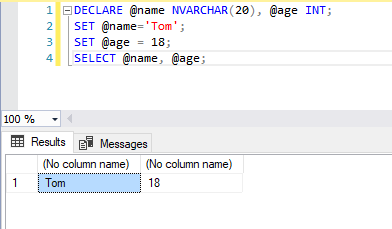
|  |
| --- |
| DECLARE @name NVARCHAR (20), @age INT;  SET @name = 'Tom';  SET @age = 18;  PRINT 'Name:' + @Name;  PRINT 'Age:' + CONVERT (CHAR, @age); |

При виконанні скрипта внизу SQL Server Management Studio відобразиться значення змінних:



Також можна використовувати для отримання значення команду SELECT:

|  |
| --- |
| DECLARE @name NVARCHAR (20), @age INT;  SET @Name = 'Tom';  SET @age = 18;  SELECT @name, @age; |



## Змінні в запитах

Через змінні ми можемо передавати дані в запити. І також ми можемо отримувати дані, які є результатом запитів, в змінні. Наприклад, при вибірці з таблиць за допомогою команди SELECT ми можемо витягувати дані в змінну за допомогою наступного синтаксису:

|  |
| --- |
| SELECT @Змінна\_1 = специфікація\_стовбчика\_1,  @Змінна\_2 = специфікація\_стовбчика\_2,  ......................................  @Змінна\_N = специфікація\_стовбчика\_N |

Крім того, в вираженні SET значення, що привласнюється змінної, також може бути результатом команди SELECT.

Наприклад, нехай у нас будуть такі таблиці:

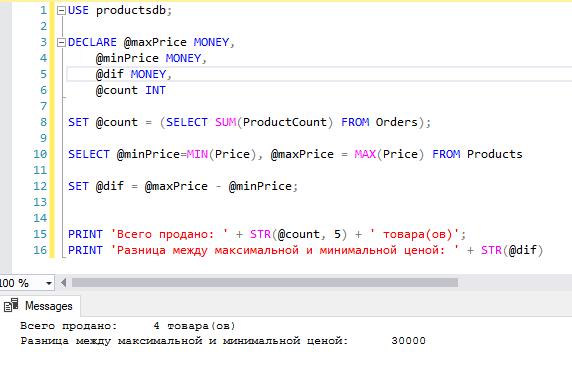
|  |
| --- |
| CREATE TABLE Products  (  Id INT IDENTITY PRIMARY KEY,  ProductName NVARCHAR (30) NOT NULL,  Manufacturer NVARCHAR (20) NOT NULL,  ProductCount INT DEFAULT 0,  Price MONEY NOT NULL  );  CREATE TABLE Customers  (  Id INT IDENTITY PRIMARY KEY,  FirstName NVARCHAR (30) NOT NULL  );  CREATE TABLE Orders  (  Id INT IDENTITY PRIMARY KEY,  ProductId INT NOT NULL REFERENCES Products (Id) ON DELETE CASCADE,  CustomerId INT NOT NULL REFERENCES Customers (Id) ON DELETE CASCADE,  CreatedAt DATE NOT NULL,  ProductCount INT DEFAULT 1,  Price MONEY NOT NULL  ); |

Використовуємо змінні при отриманні даних:

|  |
| --- |
| DECLARE @maxPrice MONEY,  @minPrice MONEY,  @dif MONEY,  @count INT  SET @count = (SELECT SUM (ProductCount) FROM Orders);  SELECT @MinPrice = MIN (Price), @maxPrice = MAX (Price) FROM Products  SET @dif = @maxPrice - @minPrice;  PRINT 'Всього продано:' + STR (@count, 5) + 'товарa (ів)';  PRINT 'Різниця між максимальною і мінімальною ціною:' + STR(@dif) |

В даному випадку змінна @count буде містити суму всіх значень з шпальти ProductCount таблиці Orders, тобто загальна кількість проданих товарів.

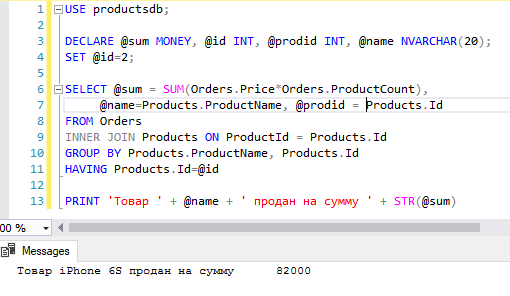
Змінні @min і @max зберігають відповідно мінімальне і максимальне значення стовпця Price з таблиці Products, а змінна @dif - різницю між цими значеннями. І подібно простим значенням, змінні також можуть брати участь в операціях.



Інший приклад:

|  |
| --- |
| DECLARE @sum MONEY, @id INT, @prodid INT, @name NVARCHAR (20);  SET @Id = 2;  SELECT @sum = SUM (Orders.Price \* Orders.ProductCount),  @Name = Products.ProductName, @prodid = Products.Id  FROM Orders  INNER JOIN Products ON ProductId = Products.Id  GROUP BY Products.ProductName, Products.Id  HAVING Products.Id=@id  PRINT 'Товар' + @name + 'Проданий на суму' + STR (@sum) |

Тут добуваються дані з двох таблиць Products і Orders групуються за стовпцями Id і ProductName з таблиці Products. Потім дані фільтруються за стовпцем Id з Products. А витягнуті дані потрапляють в змінні @sum, @name, @prodid.



## Табличні змінні

Табличні змінні (table variable) дозволяють зберегти вміст цілої таблиці. Формальний синтаксис визначення подібної змінної багато в чому схожий на створення таблиці:

|  |
| --- |
| DECLARE @ Таблична\_змінна TABLE  (Стовбчик\_1 тип\_даних [атрибути\_стовбця],  стовбчик\_2 тип\_даних [атрибути\_стовбця] ....)  [Атрибути\_таблиці] |

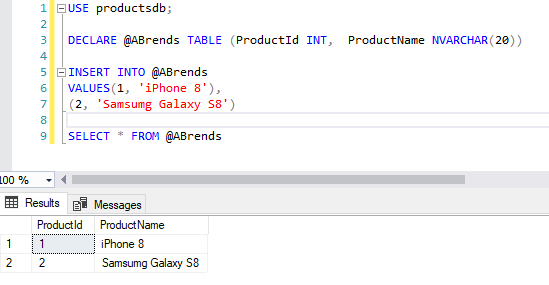
наприклад:

|  |  |
| --- | --- |
| 1 | DECLARE @ABrends TABLE (ProductId INT, ProductName NVARCHAR (20)) |

В даному випадку змінна @ABrends буде містити два стовпці.

Надалі ми зможемо працювати з цієї змінної як зі звичайною таблицею, тобто додавати в неї дані, змінювати, видаляти і витягувати їх:

|  |
| --- |
| DECLARE @ABrends TABLE (ProductId INT, ProductName NVARCHAR (20))  INSERT INTO @ABrends  VALUES (1, 'iPhone 8'),  (2, 'Samsumg Galaxy S8')  SELECT \* FROM @ABrends |



Однак слід враховувати, що такі змінні не повністю еквівалентні таблицями. Вони живуть в межах одного пакета, після завершення роботи якого вони видаляються. Тобто вони носять тимчасовий характер, і фізично їх дані ніде не зберігаються на жорсткому диску.

## Умовні вирази

Для виконання дій за умовою використовується вираз IF ... ELSE. SQL Server обчислює вираз після ключового слово IF. І якщо воно істинне, то виконуються інструкції після ключового слова IF. Якщо умова помилкова, то виконуються інструкції після ключового слова ELSE.

Якщо після IF або ELSE має блок інструкцій, то цей блок укладається між ключовими словами BEGIN і END:

|  |
| --- |
| IF умова  {Інструкція | BEGIN ... END}  [ELSE  {Інструкція | BEGIN ... END}] |

Вираз ELSE є необов'язковим, і його можна опускати.

Нехай у нас є такі таблиці (УВАГА! Якщо ви виконували всі роботи підряд, то у вас ці таблиці вже є):

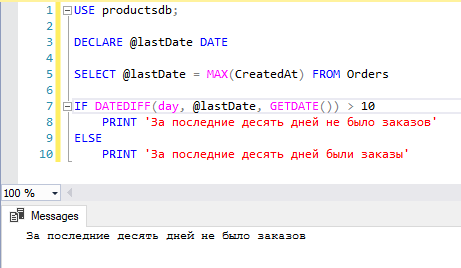
|  |
| --- |
| CREATE TABLE Products  (  Id INT IDENTITY PRIMARY KEY,  ProductName NVARCHAR (30) NOT NULL,  Manufacturer NVARCHAR (20) NOT NULL,  ProductCount INT DEFAULT 0,  Price MONEY NOT NULL  );  CREATE TABLE Customers  (  Id INT IDENTITY PRIMARY KEY,  FirstName NVARCHAR (30) NOT NULL  );  CREATE TABLE Orders  (  Id INT IDENTITY PRIMARY KEY,  ProductId INT NOT NULL REFERENCES Products (Id) ON DELETE CASCADE,  CustomerId INT NOT NULL REFERENCES Customers (Id) ON DELETE CASCADE,  CreatedAt DATE NOT NULL,  ProductCount INT DEFAULT 1,  Price MONEY NOT NULL  ); |

Таблиця Orders представляє замовлення, а стовпець CreatedAt - дату замовлень. Дізнаємося, чи були замовлення за останні 10 днів:

|  |
| --- |
| DECLARE @lastDate DATE  SELECT @lastDate = MAX (CreatedAt) FROM Orders  IF DATEDIFF (day, @lastDate, GETDATE ())> 10  PRINT 'За останні десять днів не було замовлень' |

Додамо вираз ELSE:

|  |
| --- |
| DECLARE @lastDate DATE  SELECT @lastDate = MAX (CreatedAt) FROM Orders  IF DATEDIFF (day, @lastDate, GETDATE ())> 10  PRINT 'За останні десять днів не було замовлень'  ELSE  PRINT 'За останні десять днів були замовлення' |



Якщо після IF або ELSE йдуть дві і більше інструкцій, то вони полягають в блок BEGIN ... END:

|  |
| --- |
| DECLARE @lastDate DATE, @count INT, @sum MONEY  SELECT @lastDate = MAX (CreatedAt),  @count = SUM (ProductCount),  @sum = SUM (ProductCount \* Price)  FROM Orders  IF @count > 0  BEGIN  PRINT 'Дата останнього замовлення:' + CONVERT (NVARCHAR, @lastDate)  PRINT 'Продано' + CONVERT (NVARCHAR, @count) + 'одиниць (и)'  PRINT 'На загальну суму' + CONVERT (NVARCHAR, @sum)  END;  ELSE  PRINT 'Замовлення в базі даних відсутні' |



## Цикли

Для виконання повторюваних операцій в T-SQL застосовуються цикли. Зокрема, в T-SQL є цикл WHILE. Цей цикл виконує певні дії, поки деякий умова істинно.

|  |
| --- |
| WHILE умова  {Інструкція | BEGIN ... END} |

Якщо в блоці WHILE необхідно розмістити кілька інструкцій, то всі вони містяться в блок BEGIN ... END.

Вирішимо задачу щодо підрахунку суми на кінець кожного року за вкладом у 10000 доларів на 5 років з 1,5% річних (з пролонгацією – автоматичне перезаключення договору на той самий відсоток)

DECLARE @rate FLOAT=0.015, @period INT=5, @sum MONEY=1000

WHILE @period > 0

BEGIN

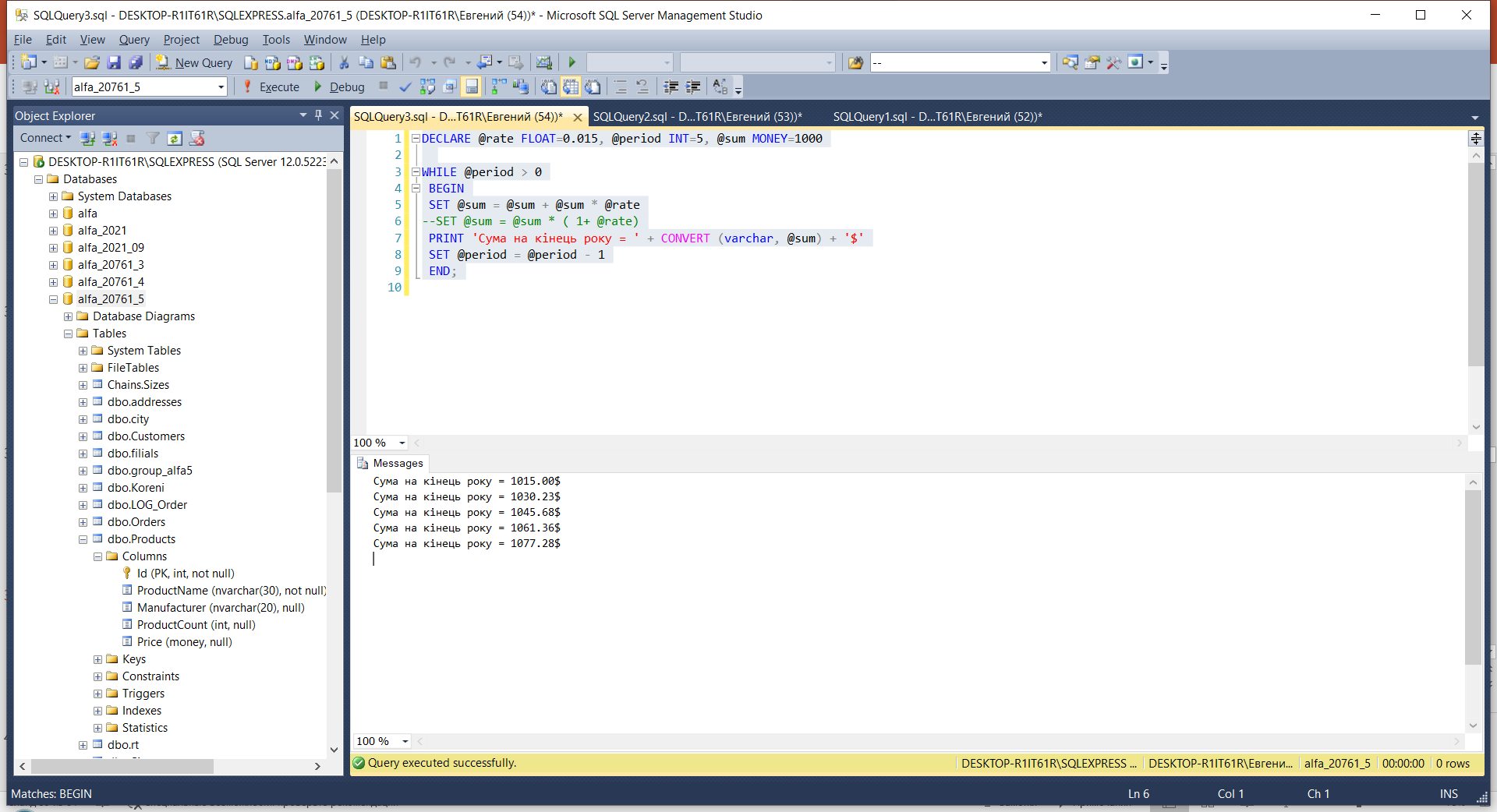
SET @sum = @sum + @sum \* @rate

PRINT 'Сума на кінець року = ' + CONVERT (varchar, @sum) + '$'

SET @period = @period - 1

END;

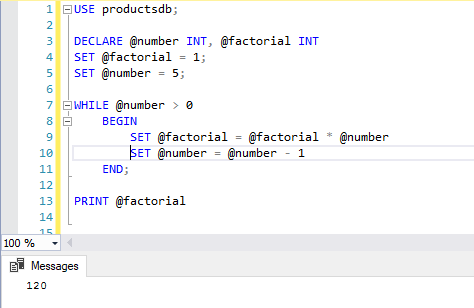
У коді SET @sum = @sum + @sum \* @rate можна замінити на SET @sum = @sum \* ( 1+ @rate).



Хто пам’ятає математику, то за допомогою циклю може легко обчислити факторіал числа (! – знак факторіалу. !6=1\*2\*3\*4\*5\*6=720):

|  |
| --- |
| DECLARE @number INT, @factorial INT  SET @factorial = 1;  SET @number = 5;  WHILE @number> 0  BEGIN  SET @factorial = @factorial \* @number  SET @number = @number - 1  END;  PRINT @factorial  PRINT 1\*2\*3\*4\*5 |

Тобто в даному випадку поки змінна @number не дорівнюватиме 0, триватиме цикл WHILE. Так як @number дорівнює 5, то цикл зробить п'ять проходів. Кожен прохід циклу називається итерацией. У кожній ітерації буде переустановлюватися значення змінних @factorial і @number.



Можемо покращити представлення інформації в прикладі про рахунок, якщо додамо рік, на який проводиться нарахування.

DECLARE @rate FLOAT=0.015, @period INT=5, @sum MONEY=1000, @year\_rah INT=1

WHILE @period > 0

BEGIN

SET @sum = @sum + @sum \* @rate

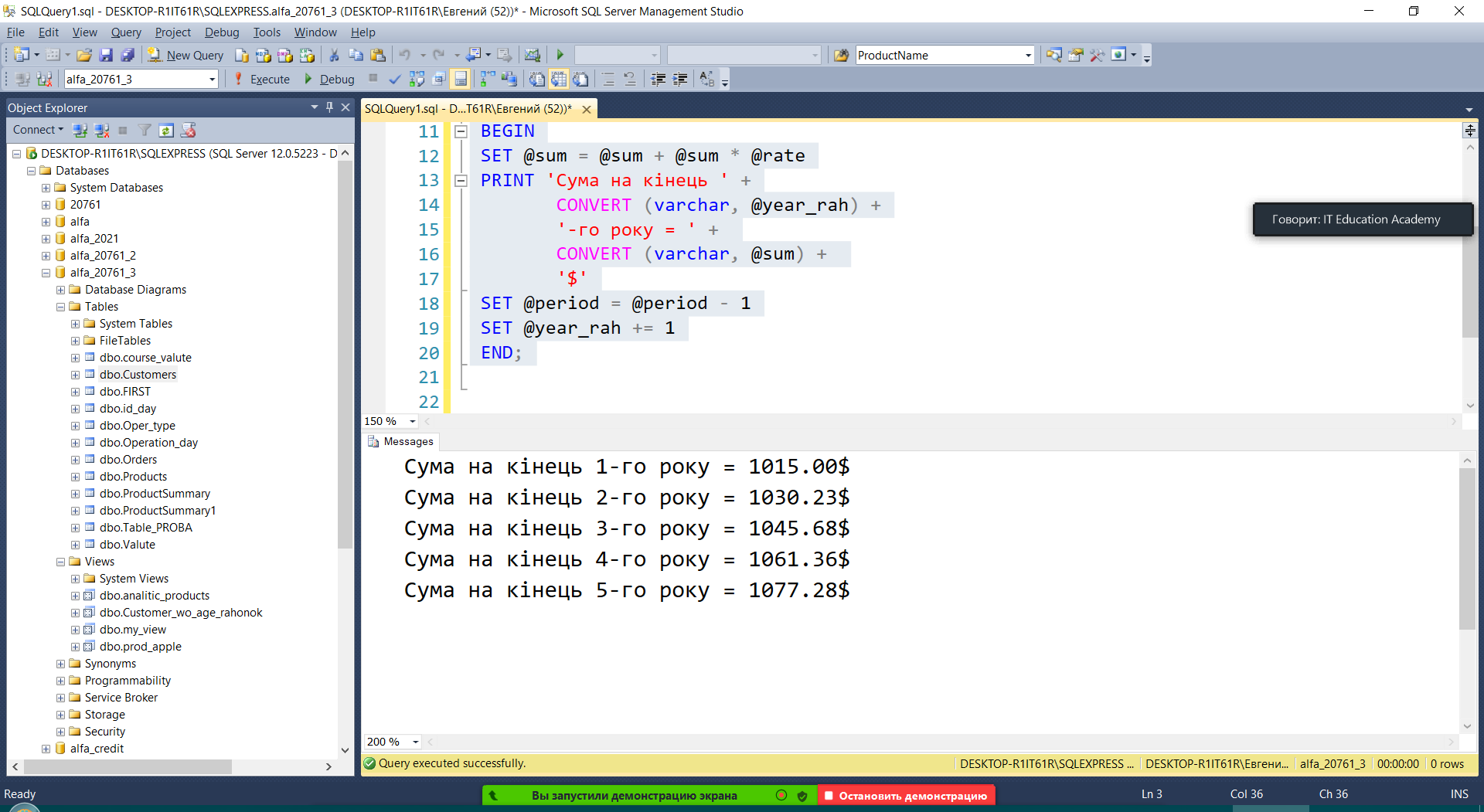
PRINT 'Сума на кінець ' + CONVERT (varchar, @year\_rah) +

'-го року = ' + CONVERT (varchar, @sum) + '$'

SET @period = @period - 1

SET @year\_rah += 1

END;



Розрахуємо баланс рахунку через кілька років з урахуванням процентної ставки і занесенням даних до тимчасової таблиці починаючи з поточної дати.

CREATE TABLE #Accounts (CreatedAt DATE, Balance MONEY)

DECLARE @rate FLOAT=0.015, @period INT=5, @sum MONEY=1000, @date\_begin DATE=GetDate()

WHILE @period > 0

BEGIN

INSERT INTO #Accounts VALUES (@date\_begin, @sum)

SET @date\_begin = DATEADD (year, 1, @date\_begin)

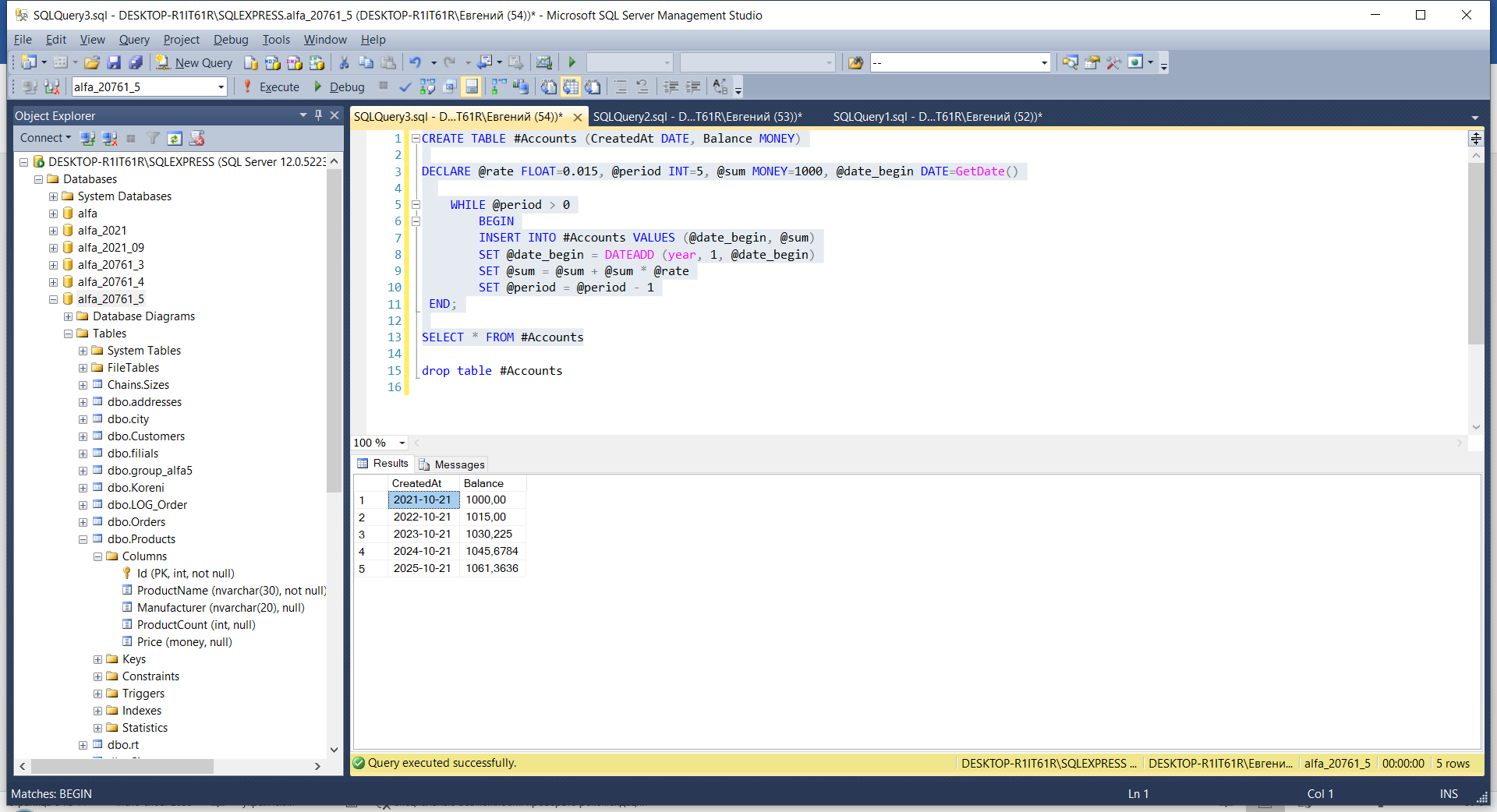
SET @sum = @sum + @sum \* @rate

SET @period = @period - 1

END;

SELECT \* FROM #Accounts

Тут створюється тимчасова таблиця #Accounts, в яку додається в циклі п'ять рядків з даними.



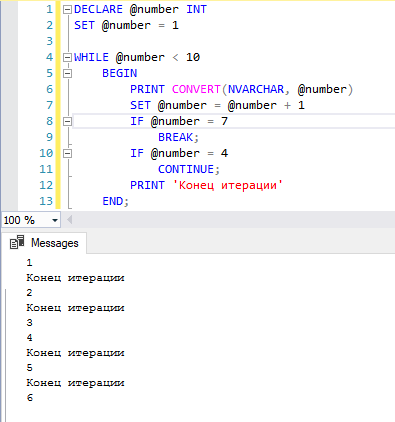
**Оператори BREAK і CONTINUE**

Оператор BREAK дозволяє завершити цикл, а оператор CONTINUE - перейти до нової ітерації.

|  |
| --- |
| DECLARE @number INT  SET @number = 1  WHILE @number <10  BEGIN  PRINT CONVERT (NVARCHAR, @number)  SET @number = @number + 1  IF @number = 7  BREAK;  IF @number = 4  CONTINUE;  PRINT 'Кінець ітерації'  END; |

Коли змінна @number стане дорівнює 4, то за допомогою оператора CONTINUE відбудеться перехід до нової ітерації, тим самим зменшується час рядокPRINT 'Кінець ітерації' не виконуватиметься, хоча цикл триватиме.

Коли змінна @number стане дорівнює 7, то оператор BREAK зробить вихід з циклу, і він завершиться.



## Обробка помилок

Для обробки помилок в T-SQL застосовується конструкція TRY ... CATCH. Вона має наступний формальний синтаксис:

|  |
| --- |
| BEGIN TRY  інструкції  END TRY  BEGIN CATCH  інструкції  END CATCH |

між виразами BEGIN TRY і END TRY поміщаються інструкції, які потенційно можуть викликати помилку, наприклад, який-небудь запит. І якщо в цьому блоці TRY виникне помилка, то управління передається в блок CATCH, де можна обробити помилку.

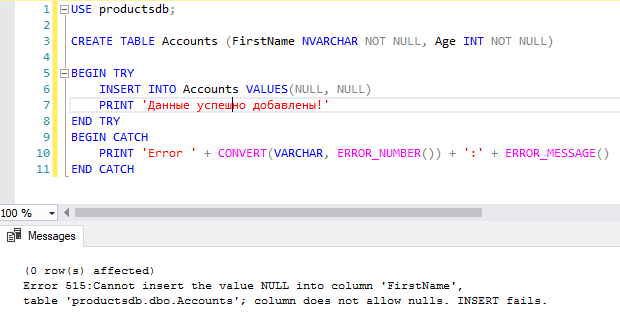
У блоці CATCH для обаботкі помилки ми можемо використовувати ряд функцій:

* **ERROR\_NUMBER ()**: Повертає номер помилки
* **ERROR\_MESSAGE ()**: Повертає повідомлення про помилку
* **ERROR\_SEVERITY ()**: Повертає ступінь серйозності помилки. Ступінь серйозності представляє числове значення. І якщо воно дорівнює 10 і менше, то така помилка розглядається як попередження і не обробляється конструкцією TRY ... CATCH. Якщо ж це значення дорівнює 20 і вище, то така помилка призводить до закриття доступу до бази даних, якщо вона не обробляється конструкцією TRY ... CATCH.
* **ERROR\_STATE ()**: Повертає стан помилки

Наприклад, додамо в таблицю дані, які не відповідають обмеженням стовпців:

|  |
| --- |
| CREATE TABLE Accounts (FirstName NVARCHAR NOT NULL, Age INT NOT NULL)  BEGIN TRY  INSERT INTO Accounts VALUES (NULL, NULL)  PRINT 'Дані успішно додані!'  END TRY  BEGIN CATCH  PRINT 'Error' + CONVERT (VARCHAR, ERROR\_NUMBER ()) + ':' + ERROR\_MESSAGE ()  END CATCH |

В даному випадку для стовпців таблиці вставляються неприпустимі дані - значення NULL, тому обробка програми перейде до блоку CATCH:



**Завдання**

1. Проробити всі приклади коду з лабораторної роботи.

2. Написати код, який без вбудованих запитів (замінити на змінну) виведе всі товари, вартість яких вище середньої вартості товарів по таблиці.

3. Вивести Ім’я користувача та всі його замовлення на основі його id, яке визначається у змінній.

4. Вивести за допомогою циклу суму замовлень за січень, лютий, ..., грудень 2021 року.

5. Написати код, який на основі значень A, B, C порахує корені квадратного тричлену (Ax2 + Bx + C = 0), якщо корені відсутні, то вивести напис «Корені відсутні».

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. Вивести дані про користувача (ім'я та телефон) та його максимальне замовлення (сума і остання дата) за його id, якщо замовлень не було, вивести фразу: "У користувача @name (код @id) замовлень не було" (де замість @id - поставити код користувача, замість @name - його ім'я)

8.\* Створити тимчасову таблицю і занести в неї дані (без використання PIVOT) з кількості проданих товарів для кожного місяця 2021 року (перелік полів: id, Name, Manufacturer, Sum\_2021\_01, Sum\_2021\_02, …, Sum\_2021\_12). Вивести цю таблицю на екран.

9\*. Виконати задачу 5 за умови унікальності трійки значень A, B, C та цілого типу значення кореня дискримінанту.

10\*. Порахувати, на скільки відрізняється число товарів, менших середньої ціни, від числа товарів, більше середньої ціни (товари, ціна яких співпадає з середньою не враховувати)

Завдання в класі

11. Вивести на екран (без використання підзапитів) виробника товару з найбільшоу вартістю (якщо є декілька таких компаній виробників, то вивести про це повідомлення) і суму, на яку продано товарів даного виробника. Інформацію виводмимо у форматі: «Виробник, товар якого найдорожчий, - це … , вже продано його товарів на суму - …» або «Багато компаній, які випускають найдорожчі продукти».