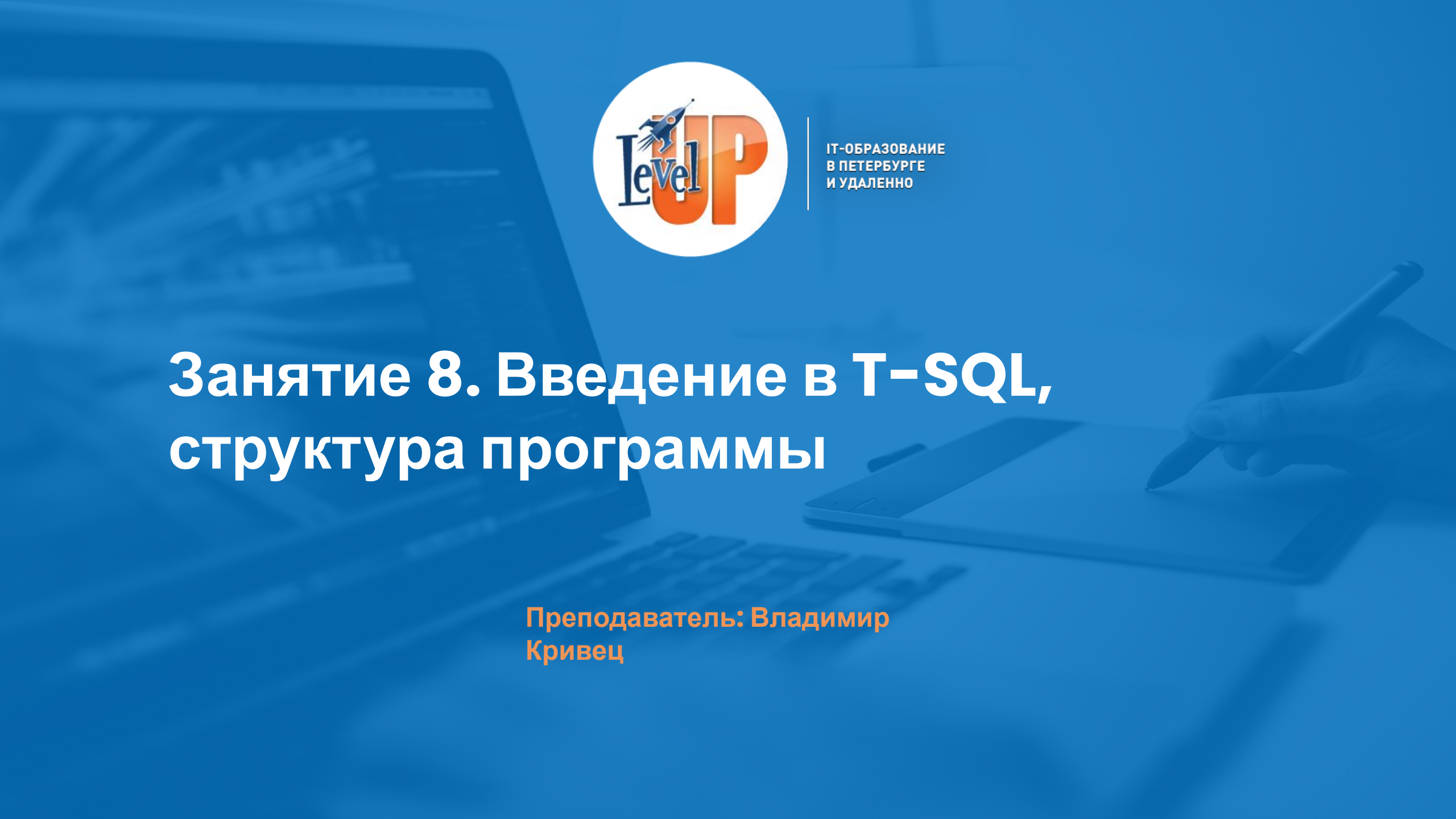

Занятие 8. Введение в T-SQL, структура программы

Преподаватель: Владимир Кривец

Вспоминаем пройденное

Занятие 7

- 3 этапа выполнения запроса
- Что такое план запроса?
- Какие операции влияют на производительность запросов больше всего?
- Что такое статистики?
- Что такое индекс, как он влияет на производительность?
- Какие типы индексов существуют в чем отличие?
- Практика

Занятие 8. Введение в T-SQL, структура программы

План

- Переменные
  - Объявление переменных
  - Присваивание значений
  - Глобальные переменные
- Использование переменных в запросе
  - В SELECT
  - В WHERE
- Условные выражения
- Циклы
- Обработка ошибок
- Практика. Создание первой программы на T-SQL

Отличие T-SQL от SQL

- SQL – декларативный язык, который предназначен для работы с множествами
  - Отсутствуют инструменты итеративной обработки
    - Циклы
    - Переменные
    - Условия
  - SQL плохо подходит для автоматизации процессов и расчетов
- T-SQL расширяет возможности SQL, добавляет:
    - Циклы
    - Переменные
    - Условия

Переменные

- Для создания процедуры применяется выражение **CREATE PROC** или **CREATE PROCEDURE** (**CREATE OR ALTER – упрощает работу с процедурами**)
- Для вызова / выполнения процедуры применяется выражение **EXEC** или **EXECUTE** (**CALL в ORACLE**)
- Для удаления процедуры применяется выражение **DROP PROC**

```
CREATE PROC FIRST_PROC
AS
BEGIN
    SELECT 'hello!' as HELLO
END
GO
EXEC FIRST_PROC
GO
DROP PROC FIRST_PROC
```

Определение значения

- После объявления значение переменных = NULL
- Чтоб изменить значение переменной можно использовать:
  - Оператор SET
    - SET @CLIENT_ID = 1
  - Оператор SELECT
    - SELECT @CLIENT_ID = 2

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

Задание:

- Создать переменные для хранения ФИО и даты рождения @FIO, @BIRTHDATE и присвоить им начальные значения
- Создать переменную @AGE для хранения возраста человека
- Написать код, который выводит сообщение:
- «Уважаемый, @FIO, вы родились @BIRTHDATE , вам сейчас @AGE лет»

Глобальные переменные

- Область видимости @переменной -все инструкции между ее объявлением и концом пакета или хранимой процедуры, где она объявлена.
  - Вы не можете объявить переменную в одном окне запросов и использовать ее в другом.
- Существуют глобальные переменные (системные переменные, системные функции)
  - Глобальные переменные начинаются с @@
  - Вы не можете создать свою глобальную переменную
  - Вы можете использовать глобальную переменную в любом месте в вашем T-SQL коде.
- Пример полезных глобальных переменных
  - `@@SERVERNAME` – имя вашего SQL Server
  - `@@SPID` – идентификатор текущей сессии
  - `@@VERSION` – версия вашего SQL Server

Использование переменных в запросе

- Через переменные мы можем передавать данные **в запросы**
- Также мы можем получать данные **из запросов**
  - И сохранять их **в переменные**
- В переменную можно «сохранить» результат только одной ячейки таблицы
  - В обычную переменную нельзя сохранить несколько строк или столбцов

```
DECLARE @CLIENT_ID int, @age INT;
SET @name='Tom';
SET @age = 18;
PRINT 'Name: ' + @name;
PRINT 'Age: ' + CONVERT(CHAR, @age);
SELECT @name, @age;
```

Использование переменных в запросе


```
--ПОДСЧЕТ КОЛ-ВА ПРОДАНОГО ТОВАРА
DECLARE @ProductName varchar(100) = 'Road-650 Black, 58';
DECLARE @ProductID int;
DECLARE @COUNT INT;

SELECT @ProductID =[ProductID]
FROM [Production].[Product]
WHERE [NAME] = @ProductName

SELECT @COUNT = sum(SOD.OrderQty)
FROM Sales.SalesOrderDetail SOD
WHERE SOD.ProductID = @ProductID

PRINT N'ПРОДУКТ "'+@ProductName + '"'
PRINT N'ВСЕГО ПРОДАНО(шт.) '+CAST(@COUNT AS
NVARCHAR(100))
```

Условные выражения

- Для выполнения действий по условию используется выражение `IF ... ELSE`.
- SQL Server вычисляет выражение после ключевого слово `IF`.
- И если оно истинно, то выполняются инструкции после ключевого слова `IF`.
- Если условие ложно, то выполняются инструкции после ключевого слова `ELSE`.

```
--ПРОВЕРКА ЗАКАЗОВ ЗА ПОСЛЕДНИЕ 10 ДНЕЙ
DECLARE @lastDate DATE

SELECT @lastDate = MAX(SOH.OrderDate) FROM Sales.SalesOrderHeader SOH

IF DATEDIFF(day, @lastDate, GETDATE()) > 10
    PRINT N'За последние десять дней не было заказов'
ELSE
    PRINT N'За последние десять дней были заказы'
```

Условные выражения

- Если после `IF` или `ELSE` идут две и более инструкций
  - То они заключаются в блок `BEGIN`...`END`:

```
--ПРОВЕРКА ЗАКАЗОВ, И ЕСЛИ ЗАКАЗЫ БЫЛИ - ВЫВОД СТАТИСТИКИ
DECLARE @lastDate DATE, @count INT, @sum MONEY
SELECT @lastDate = MAX(OrderDate),
        @count = count(SOH.SalesOrderID) ,
        @sum = SUM(SOH.SubTotal)
FROM Sales.SalesOrderHeader SOH
IF @count > 0
    BEGIN
        PRINT N'Дата последнего заказа: ' + CONVERT(NVARCHAR, @lastDate)
        PRINT N'Продано ' + CONVERT(NVARCHAR, @count) + N' единиц(ы)'
        PRINT N'На общую сумму ' + CONVERT(NVARCHAR, @sum)
    END;
ELSE
    PRINT N'Заказы в базе данных отсутствуют'
```

Циклы

- Для выполнения повторяющихся операций в T-SQL применяются цикл WHILE
- WHILE (условие)
  - BEGIN
    - T-SQL
  - END

```
--Вывод чисел от 1 до 10

DECLARE @N INT = 1

WHILE @N <= 10
    BEGIN
        PRINT @n
        SET @N= @n + 1
    END;
```

Циклы

- Задание
  - Написать код, который считает факториал числа

```
--факториал числа
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
```

Циклы

Операторы BREAK и CONTINUE

- Оператор BREAK завершает цикл
- Оператор CONTINUE переходит к следующей итерации

```
DECLARE @N INT = 1
WHILE @N <= 10
    BEGIN
PRINT @n
    SET @N= @n + 1
    IF @N= 7
            BREAK;
        IF @N= 4
            CONTINUE;
        PRINT 'END OF STEP' + cast(@N as nvarchar(100))
    END;
```

Обработка ошибок

- Для обработки ошибок в T-SQL применяется конструкция TRY…CATCH

```
BEGIN TRY
        T-SQL инструкции
END TRY
BEGIN CATCH
    T-SQL инструкции
END CATCH
```

- Между выражениями BEGIN TRY и END TRY помещаются инструкции, которые потенциально могут вызвать ошибку, например, какой-нибудь запрос.
- Если в этом блоке TRY возникнет ошибка, то управление передается в блок CATCH, где можно обработать ошибку и продолжить выполнение программы.
- Инструкции без TRY…CATCH в случае ошибки – завершают выполнение программы

Обработка ошибок

В блоке `CATCH` для обработки ошибки мы можем использовать ряд функций:

- `ERROR_NUMBER`(): возвращает номер ошибки
- `ERROR_MESSAGE`(): возвращает сообщение об ошибке
- `ERROR_SEVERITY`(): возвращает «степень серьезности» ошибки.
  - Если «степень серьезности» <=10, то такая ошибка рассматривается как предупреждение и не обрабатывается конструкцией `TRY...CATCH`.
  - Если «степень серьезности» >=20, то такая ошибка приводит к закрытию подключения к базе данных, если она не обрабатывается конструкцией `TRY...CATCH`.
- `ERROR_STATE`(): возвращает состояние ошибки

Обработка ошибок


```
--добавим в таблицу данные, которые не соответствуют ограничениям столбцов
CREATE TABLE Accounts (FirstName NVARCHAR NOT NULL, Age INT NOT NULL)

BEGIN TRY
    INSERT INTO Accounts VALUES(NULL, NULL)
    PRINT 'Данные успешно добавлены!'
END TRY
BEGIN CATCH
    PRINT 'Error ' + CONVERT(VARCHAR, ERROR_NUMBER()) + ':' + ERROR_MESSAGE()
END CATCH
```