Язык программирования T-SQL

SQL Server включает много конструкций, компонентов, функций которые расширяют возможности языка SQL стандарта ANSI, в том числе и классическое программирование, которое отличается от обычного написания запросов.

Переменные в T-SQL

Существует две разновидности переменных в T-SQL:

• ***Локальные***. Существуют только в пределах сеанса, во время которого они были созданы. Локальные переменные объявляются с помощью ключевого слова **DECLARE** и начинаются со знака @.

• ***Глобальные***. Используются для получения информации о SQL сервере или базе данных. Глобальные переменные начинаются с @@.

Например:

`@@ROWCOUNT` – хранит количество записей, обработанных предыдущей командой;

`@@ERROR` – возвращает код ошибки для последней команды;

`@@SERVERNAME` — имя локального SQL сервера;

`@@VERSION` — номер версии SQL Server;

`@@IDENTITY` — последнее значение счетчика, используемое в операции вставки (*insert*).

Для присвоения значения переменной можно использовать команды **SET** или **SELECT**.

Типы данных

- ***Точные числа*** (bit, tinyint, smallint, int, bigint, numeric (p, s), decimal (p, s), smallmoney, money);
- ***Приблизительные числа*** (float (n), real);
- ***Символьные строки не в Юникоде*** (char (n), varchar (n | max), text);
- ***Символьные строки в Юникоде*** (nchar (n), nvarchar (n | max), ntext);
- ***Дата и время*** (date, datetime, datetime2, smalldatetime, time [Точность], datetimeoffset [Точность]);
- ***Двоичные данные*** (binary (n), varbinary (n | max), image);
- ***Прочие типы данных*** (cursor, table, sql_variant, rowversion (timestamp), Xml, uniqueidentifier, hierarchyid, пространственные типы).

decimal и numeric - типы числовых данных с фиксированными точностью и масштабом

```
CREATE TABLE dbo.MyTable
(
  MyDecimalColumn DECIMAL(5,2)
,MyNumericColumn NUMERIC(10,5)

);

GO
INSERT INTO dbo.MyTable VALUES (123, 12345.12);
GO
SELECT MyDecimalColumn, MyNumericColumn
FROM dbo.MyTable;
```

Результирующий набор:

SQL

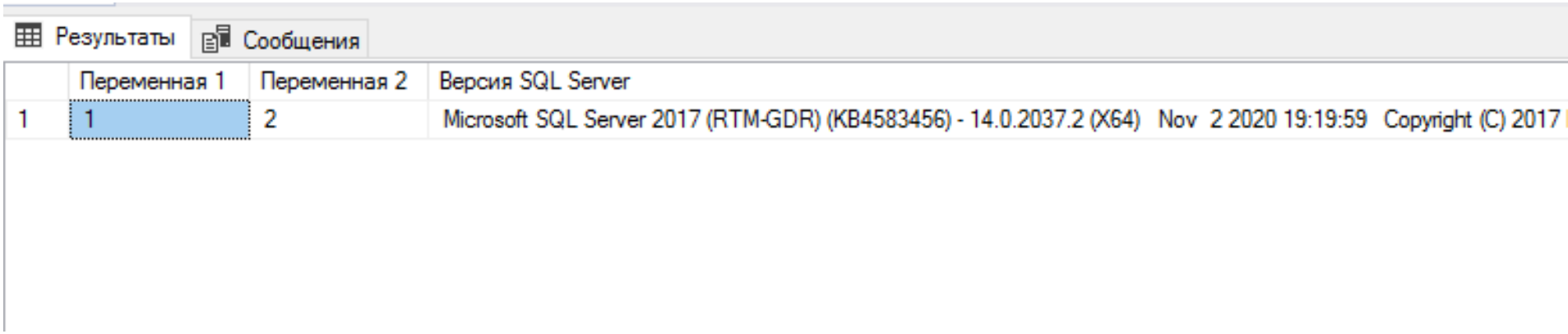
```
MyDecimalColumn                          MyNumericColumn
---------------------------------------- ----------------------------------------
123.00                                   12345.12000

(1 row(s) affected)
```

Пример

```
DECLARE @Var1 INT =1
--Объявление и присвоение значения переменной Var1
DECLARE @Var2 INT          --Объявление переменной Var2
SELECT @Var2 = 2           --Присвоение значения переменной Var2

/*Вывод значений локальных переменных Var1, Var2 и глобальной переменной
Version*/
SELECT @Var1 AS [Переменная 1], @Var2 AS
[Переменная 2], @@VERSION AS [Версия SQL Server]
```



Пакеты

Пакет в T-SQL — это команды или инструкции SQL, которые объединены в одну группу и при этом SQL сервер будет компилировать, и выполнять их как одно целое.

Для того чтобы дать понять SQL серверу, что передается пакет команд необходимо указывать ключевое слово **GO** после всех команд, которые принадлежат одному пакету.

Локальные переменные будут видны только в пределах того пакета, в котором они были созданы, если обратиться к переменной после завершения пакета SQL Server сгенерирует ошибку.

Условные конструкции

Эти конструкции подразумевают ветвление, т.е. в зависимости от выполнения или невыполнения определенных условий инструкции T-SQL будут выполняться соответствующие блоки кода.

Конструкция IF…ELSE подразумевает проверку выполнения условий и если все проверки пройдены, то выполняется команда идущая следом, если нет, то не выполняется ничего, но можно указать ключевое слово ELSE и тогда в этом случае будут выполняться операторы указанные после этого слова.

```
IF Boolean_expression
      { sql_statement | statement_block }
[ ELSE
      { sql_statement | statement_block } ]
```

Внимание! statement_block начинается с BEGIN, заканчивается END

Условные конструкции. Примеры

```
IF 1 = 2 PRINT 'Boolean_expression is true.'
ELSE PRINT 'Boolean_expression is false.' ;
GO



DECLARE @Var1 INT
DECLARE @Var2 VARCHAR(20)
SET @Var1 = 5
IF @Var1 > 0
  SET @Var2 = 'Больше 0'
ELSE
  SET @Var2 = 'Меньше 0'
SELECT @Var2 AS [Значение Var1]
```

PRINT

Для передачи служебного сообщения можно использовать команду **PRINT**. В Management Studio это сообщение отобразится на вкладке «Сообщения» (*Messages*).

```
DECLARE @Cnt INT = 10, @Var varchar(100)
IF @Cnt > 0
  SET @Var = 'Значение переменной Cnt больше 0 и
  равняется ' + CAST(@Cnt  AS VARCHAR(10))
ELSE
  SET @Var = 'Значение переменной Cnt меньше 0 и
  равняется ' + CONVERT(VARCHAR(10),@Cnt)
PRINT @Var
```

Преобразование выражения одного типа данных в другой:

CAST (expression AS data_type [(length)])

CONVERT (data_type [(length)] , expression [, style])

Expression - любое допустимое выражение.

data_type - целевой тип данных.

Length - дополнительно целое число, обозначающее длину целевого типа данных. Значение по умолчанию - 30.

Style - целочисленное выражение, определяющее, как функция CONVERT преобразует значение аргумента *expression* .

Условные конструкции. Пример

```
USE AdventureWorks2012;
GO
IF (SELECT COUNT(*) FROM Product
WHERE Name LIKE '%турист%' ) >= 5
/* Если количество записей в таблице Product, удовлетворяющих заданному условию (поле Name начинается со слов «Турист») больше 5, то …
  PRINT 'Имеется не менее 5 туристических товаров.'
ELSE
  PRINT 'Имеется меньше чем 5 туристических товаров.' ;
GO
```

IF EXISTS

Данная конструкция позволяет определить наличие записей, соответствующих определенному условию.

Пример

```
DECLARE @Var VARCHAR(20)
IF EXISTS (SELECT * FROM test_table WHERE id > 0)
  SET @Var = 'Записи есть'
ELSE
  SET @Var = 'Записей нет'
SELECT @Var AS [Наличие записей]
```

CASE

Данная конструкция используется совместно с оператором SELECT и предназначена для замены многократного использования конструкции IF. Она полезна в тех случаях, когда необходимо проверять переменную (*или поле*) на наличие определенных значений.

```
--Simple CASE expression:
CASE input_expression
WHEN when_expression THEN result_expression [...n]
     [ ELSE else_result_expression ]
END

--Searched CASE expression:
CASE
WHEN Boolean_expression THEN result_expression [...n]
     [ ELSE else_result_expression ]
END
```

Case. Пример

```
DECLARE @CountProductAll INT
DECLARE @Mess VARCHAR(20)
SELECT @CountProductAll = SUM(CountProduct) FROM
Product
SELECT @Mess = CASE @CountProductAll
               WHEN 1 THEN 'Один'
               WHEN 2 THEN 'Два'
               WHEN 3 THEN ‘Три'
               ELSE ‘Равно нулю или больше 3'
       END
SELECT @Mess AS [Суммарное количество товаров на
складе]
```

CASE. Пример

При использовании в инструкции SELECT поисковое выражение CASE позволяет заменять значения в результирующем наборе в зависимости от результатов сравнения. В следующем примере отображается список цен в виде текстового комментария, основанного на диапазоне цен для продукта.

```
USE AdventureWorks2012;
GO
SELECT ProductNumber, Name, "Price Range" =
CASE
WHEN ListPrice = 0 THEN 'Не для продажи'
WHEN ListPrice < 50 THEN 'Ниже $50'
WHEN ListPrice >= 50 and ListPrice < 250 THEN 'Ниже $250'
WHEN ListPrice >= 250 and ListPrice < 1000 THEN 'Ниже
$1000'
ELSE 'Выше $1000'
END
FROM Product;
GO
```

BEGIN…END

Эта конструкция необходима для создания блока команд.

Пример

```
DECLARE @Var1 VARCHAR(20)
DECLARE @Var2 INT
SET @Var2 = 0
IF EXISTS(SELECT * FROM test_table WHERE id > 0)
  BEGIN
      SET @Var1 = 'Записи есть'
      UPDATE test_table SET column1 = 5 WHERE id > 12
      SET @Var2 = @@ROWCOUNT     --определение количества
                                   обработанных строк
  END
ELSE
   SET @Var1 = 'Записей нет'
SELECT @Var1 AS [Наличие записей], @Var2 AS [Затронуто
строк:]
```

Циклы T-SQL

Конструкция

```
WHILE условие
{sql_statement|statement_block|BREAK|CONTINUE }
END
```

позволяет выполнять SQL-инструкцию или блок инструкций до тех пор, пока условие истинно.

Аргументы

Условие – логическое *в*ыражение, возвращающее значение TRUE или FALSE. Если логическое выражение содержит инструкцию SELECT, инструкция SELECT должна быть заключена в скобки.

{*sql_statement* | *statement_block*}

Любая инструкция или группа инструкций Transact-SQL, определенная в виде блока инструкций.

BREAK - Приводит к выходу из ближайшего цикла WHILE. Вызываются инструкции, следующие за ключевым словом END, обозначающим конец цикла.

CONTINUE - Выполняет цикл WHILE для перезагрузки, не учитывая все инструкции, следующие после ключевого слова CONTINUE.

Пример
Если количество записей=max(id)

```
DECLARE @i INT = 0, @CountRow INT
SELECT @CountRow = COUNT(*) FROM test_table
WHILE @i <= @CountRow  /*цикл, пока @i  будет меньше
или равно количеству строк в таблице test_table */
BEGIN
   SET @i += 1
   IF @i = 2 /*если количество повторений цикла достигло 2,
   то принудительно выходим из цикла */
      BREAK
   ELSE
      CONTINUE   --иначе продолжаем цикл
END
SELECT @i  AS [Количество выполнений цикла:]
```

test_table

id	name	countOfSite
1	Товар1	12
2	Товар2	10
3	Товар3	11

Пример
Если количество записей<>max(id)

```
DECLARE @i INT = 0, @MaxIDRow INT
SELECT @MaxIDRow = Max(id) FROM test_table
WHILE @i <= @MaxIDRow /*цикл выполняется, пока @i будет
меньше или равно максимальному id в таблице test_table */
BEGIN
  SET @i += 1
  IF @i > 2 /*если количество повторений цикла достигло 2,
  то принудительно выходим из цикла */
     BREAK
  ELSE
     CONTINUE   --иначе продолжаем цикл
END
SELECT @i  AS [Количество выполнений цикла:]
```

test_table

id	name	countOfSite
1	Товар1	12
13	Товар2	10
26	Товар3	11

Пример

Вывести все записи из таблицы в окно Message

```
DECLARE @i INT = 0, @CountRow INT
DECLARE @name VARCHAR(10), @countOfSite INT
SELECT @CountRow = Count(*) FROM test_table
WHILE @i <= @CountRow
BEGIN
   SET @i += 1
   SELECT @name=name, @countOfSite= countOfSite
   FROM test_table WHERE id=@i
   PRINT @name + CONVERT(VARCHAR(2), @countOfSite)
END
```

test_table

id	name	countOfSite
1	Товар1	12
2	Товар2	10
3	Товар3	11

Окно Message

Товар1 12
Товар2 10
Товар3 11

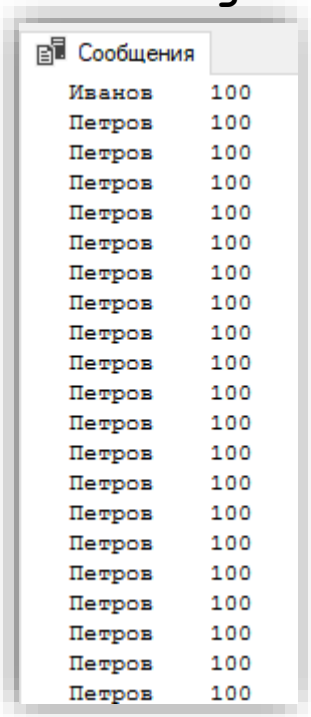
Пример
Вывести все записи из таблицы в окно Message

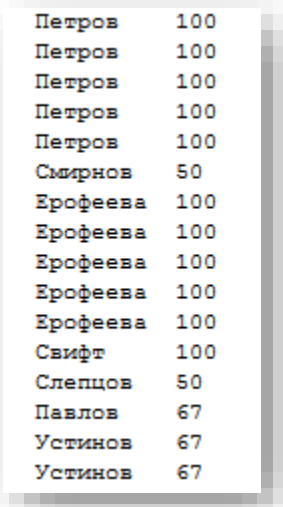
```
use test
DECLARE @i INT = 0, @MaxIDRow INT
DECLARE @name VARCHAR(10), @countryCode INT
SELECT @MaxIDRow = Max(НомерБегуна) FROM Бегун
WHILE @i <= @MaxIDRow
BEGIN
SET @i += 1
SELECT @name=Фамилия, @countryCode = countryCode FROM Бегун WHERE
НомерБегуна=@i
PRINT @name + CONVERT(VARCHAR(3), @countryCode)
END
```

Таблица "Бегун"

НомерБегуна	Фамилия	Имя	Отчество	BibNumber	CountryCode
1	Иванов ...	Иван	Петрович	12	100
2	Петров ...	Петр	Петрович	13	100
1002	Смирнов ...	Петр	Михайлович ...	20	50
1003	Ерофеева ...	Инга	Денисовна ...	12	100
1008	Свифт ...	Семен	Михайлович ...	12	100
1009	Слепцов ...	Аркадий	Степанович ...	20	50
1010	Павлов ...	Иван	Игоревич	80	67
1011	Устинов ...	Олег	Семенович ...	80	67

Окно Message





Пример

Если средняя цена продуктов ListPrice меньше чем $300, цикл WHILE удваивает цены, а затем выбирает максимальную. В том случае, если максимальная цена меньше или равна $500, цикл WHILE повторяется и снова удваивает цены. Этот цикл продолжает удваивать цены до тех пор, пока максимальная цена не будет больше чем $500, затем выполнение цикла WHILE прекращается, о чем выводится соответствующее сообщение.

```
USE AdventureWorks2012;
GO
WHILE (SELECT AVG(ListPrice) FROM Product)<300
BEGIN
  UPDATE Product SET ListPrice = ListPrice * 2
  IF (SELECT MAX(ListPrice) FROM Product)>500
    BREAK
  ELSE
    CONTINUE
END
PRINT 'Выполнение цикла прекратилось';
```

Тип данных TABLE

Тип данных TABLE — это особый тип локальной переменной, который помогает временно хранить данные, подобно временной таблице в SQL Server.

```
DECLARE @LOCAL_TABLEVARIABLE TABLE
(column_1 DATATYPE,
column_2 DATATYPE,
…
column_N DATATYPE)
```

Тип данных TABLE. Пример

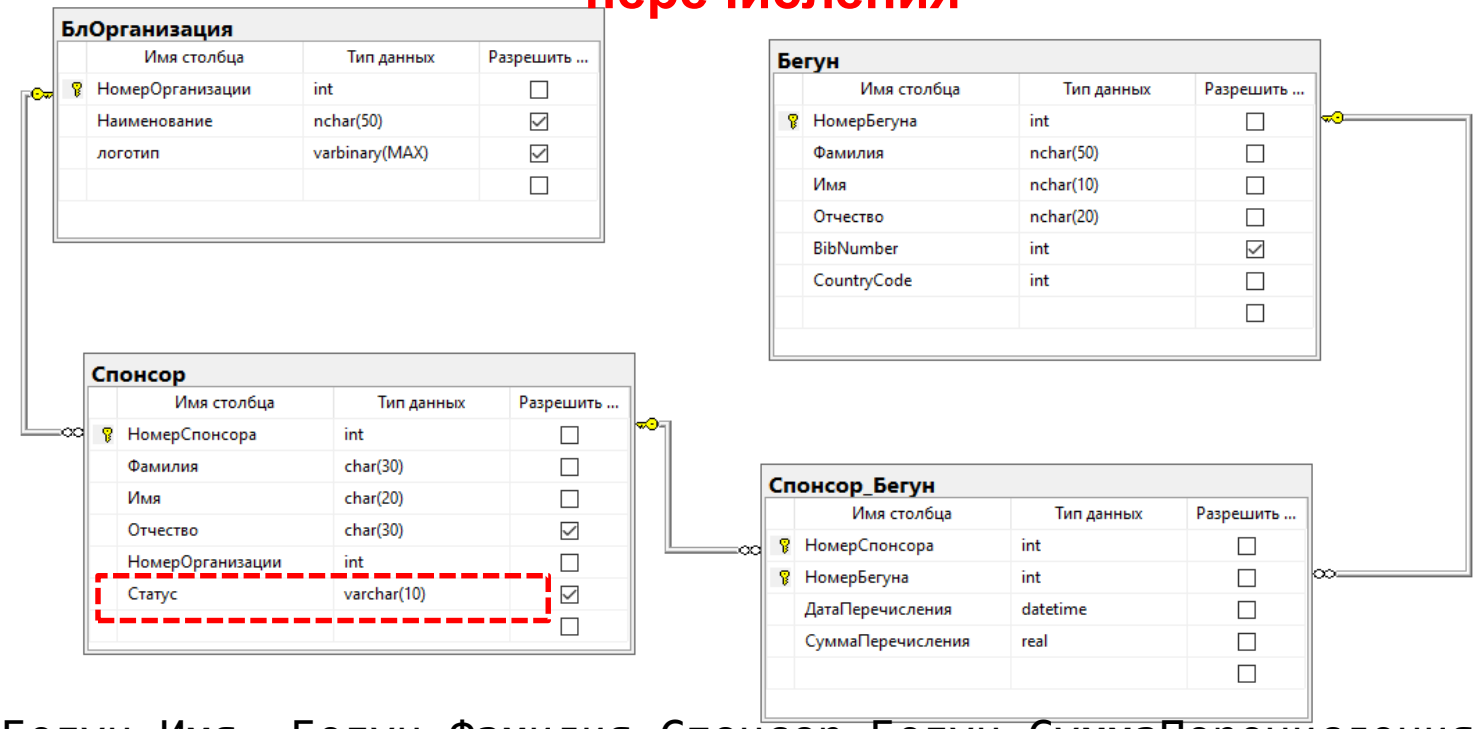
```
DECLARE @ListWeekDays TABLE(WeekDyNumber INT, WeekAbb VARCHAR(40),
WeekName VARCHAR(40))

INSERT INTO @ListWeekDays
VALUES
(1,'Mon','Monday')  ,
(2,'Tue','Tuesday') ,
(3,'Wed','Wednesday') ,
(4,'Thu','Thursday'),
(5,'Fri','Friday'),
(6,'Sat','Saturday'),
(7,'Sun','Sunday')
SELECT * FROM @ListWeekDays
```

Results | Messages

	WeekDyNumber	WeekAbb	WeekName
1	1	Mon	Monday
2	2	Tue	Tuesday
3	3	Wed	Wednesday
4	4	Thu	Thursday
5	5	Fri	Friday
6	6	Sat	Saturday
7	7	Sun	Sunday

Пример. Присвоить статус спонсору в зависимости от суммы перечисления




```
select Бегун.Имя, Бегун.Фамилия,Спонсор_Бегун.СуммаПеречисления,
Спонсор.Фамилия as Спонсор from Бегун
inner join Спонсор_Бегун on Бегун.НомерБегуна=Спонсор_Бегун.НомерБегуна
inner join Спонсор on Спонсор.НомерСпонсора=Спонсор_Бегун.НомерСпонсора
```

Результаты | Сообщения

	Имя	Фамилия	СуммаПеречисления	Спонсор
1	Иван	Иванов	1000	Катаев
2	Петр	Петров	25000	Катаев
3	Иван	Павлов	23444	Кижаев
4	Петр	Смирнов	56000	Ивлев

Пример. Присвоить статус спонсору в зависимости от суммы перечисления

```
DECLARE @table TABLE
(НомерСпонсора INT,
Спонсор VARCHAR(20),
СуммаОбщая MONEY,
Статус VARCHAR(10) NULL );

INSERT INTO @table(НомерСпонсора,Спонсор,СуммаОбщая)
SELECT Спонсор.НомерСпонсора, Спонсор.Фамилия AS
Спонсор,Sum(Спонсор_Бегун.СуммаПеречисления) AS СуммаОбщая  FROM Бегун
inner join Спонсор_Бегун ON Бегун.НомерБегуна=Спонсор_Бегун.НомерБегуна
inner join Спонсор ON Спонсор.НомерСпонсора=Спонсор_Бегун.НомерСпонсора
GROUP BY Спонсор.Фамилия, Спонсор.НомерСпонсора

SELECT * FROM @table
```

Результаты | Сообщения

	НомерСпонсора	Спонсор	СуммаОбщая	Статус
1	1	Катаев	26000,00	NULL
2	2	Кижаев	23444,00	NULL
3	3	Ивлев	56000,00	NULL

Пример. Присвоить статус спонсору в зависимости от суммы перечисления

```
UPDATE @table SET Статус='***' WHERE СуммаОбщая>50000
UPDATE @table SET Статус='**' WHERE СуммаОбщая>20000 AND
СуммаОбщая<=50000
UPDATE @table SET Статус='*' WHERE СуммаОбщая<=20000

SELECT * FROM @table
```

	НомерСпонсора	Спонсор	СуммаОбщая	Статус
1	1	Катаев	26000,00	**
2	2	Кижаев	23444,00	**
3	3	Ивлев	56000,00	***

Пример. Присвоить статус спонсору в зависимости от суммы перечисления

Если необходимо объединить две или более переменные типа TABLE друг с другом или с обычными таблицами, необходимо использовать псевдонимы для имен таблиц:

```
UPDATE Спонсор
SET Спонсор.Статус =
(SELECT Статус FROM @table AS MyTable
WHERE Спонсор.НомерСпонсора= MyTable.НомерСпонсора)
```

Я-ПК\SQLEXPRESS.test - dbo.Спонсор | Я-ПК\SQLEXPRESS....bo.Спонсор_Бегун | Я-ПК\SQLEXPRESS.test

	НомерСпонс...	Фамилия	Имя	Отчество	НомерОрган...	Статус
▶	1	Катаев ...	Иван	Васильевич ...	1	**
	2	Кижаев ...	Дмитрий	Анатольевич ...	1	**
	3	Ивлев ...	Антон	Иванович ...	2	***
*	NULL	NULL	NULL	NULL	NULL	NULL

Ограничения для типа данных TABLE

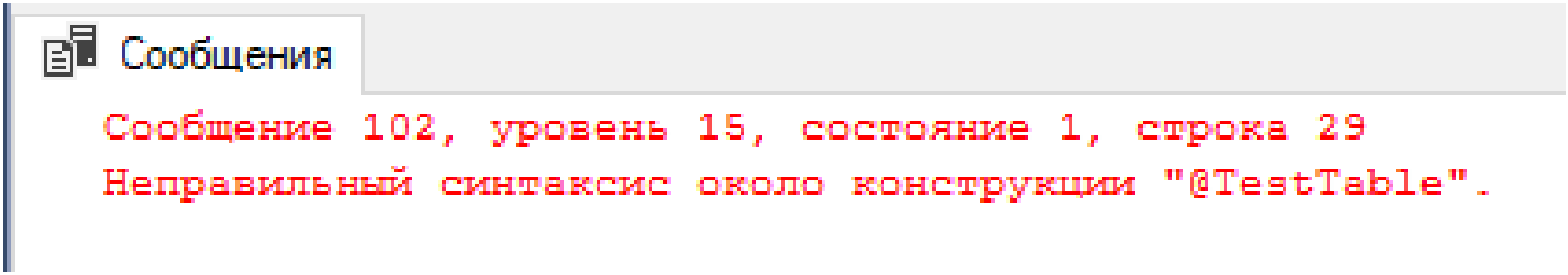
При **создании** таблицы можно использовать ограничения:

- Primary Key
- Unique
- Null
- Check

Однако изменения в структуру созданной переменной типа TABLE вносить нельзя:

```
DECLARE @TestTable TABLE
(ID INT PRIMARY KEY,
Col1 VARCHAR(40) UNIQUE,
Col2 VARCHAR(40) NOT NULL)

ALTER TABLE @TestTable
ADD Col4 INT
```



GOTO

С помощью этой команды можно перемещаться по коду к указанной метке.

```
DECLARE @A INT = 0
Metka: --Устанавливаем метку
SET @A += 1 --Прибавляем к переменной 1
if @A < 10
  GOTO Metka --Если значение меньше 10, то переходим к метке
SELECT @A AS [Значение A =]
```

WAITFOR

Блокирует выполнение пакета, хранимой процедуры или транзакции, пока не наступит указанное время, не истечет заданный интервал времени, либо заданная инструкция не изменит или не возвратит по крайней мере одну строку.

```
WAITFOR
{ DELAY 'time_to_pass'
| TIME 'time_to_execute'
| [ ( receive_statement )
| ( get_conversation_group_statement ) ]
 [ , TIMEOUT timeout ] }
```

Аргументы

DELAY - Заданный период времени (не более 24 часов), который должен пройти до выполнения пакета.

'***time_to_pass***' - Период времени ожидания (формат чч:мм[[:сс].мсс]).

TIME - Заданное время выполнения пакета.

'***time_to_execute***' - Время, в которое инструкция WAITFOR завершает работу (формат чч:мм[[:сс].мсс]).

receive_statement, get_conversation_group_statement, **TIMEOUT** ***timeout*** применяются только к сообщениям компонента Компонент Service Broker.

WAITFOR. Пример

Команда может приостановить выполнение кода на время или до наступления заданного времени. Параметр **DELAY** делает паузу заданной длины, а **TIME** приостанавливает процесс до указанного времени. Значение параметров задается в формате **hh:mi:ss**

```
DECLARE @TimeStart time, @TimeEnd time
SET @TimeStart = CONVERT (time, GETDATE()) --Узнаем
время
WAITFOR DELAY '00:00:05‘ --Пауза на 5 секунд
SET @TimeEnd = CONVERT (time, GETDATE()) --Снова
узнаем время
--Узнаем, сколько прошло времени в секундах
SELECT DATEDIFF(ss, @TimeStart, @TimeEnd) AS [Прошло
Секунд:]
```

GETDATE() - возвращает текущее системное время в формате DT_DBTIMESTAMP (Структура отметки времени, включающая год, месяц, день, час, минуты, секунды и доли секунд. Максимальный масштаб для долей секунд — 3 разряда).

DATEDIFF (едИзмерения , startdate , enddate) – возвращает временной период между startdate и enddate в указанных единицах измерения.

DATEPART (едИзмерения , date) - возвращает целое число, представляющее указанную часть даты (месяц, день, год, час …).

Например, SELECT DATEPART(month, '12/20/1974'); вернет число 12

WAITFOR. Замечания

Фактическая временная задержка может отличаться от времени, указанного в аргументах *time_to_pass*, *time_to_execute* или *timeout*, и зависит от уровня активности сервера. Счетчик времени запускается, когда запланирован поток инструкции WAITFOR. Если сервер занят, запланировать поток сразу может быть невозможно, поэтому время задержки может оказаться больше заданного.

Использование указания WAITFOR приведет к замедлению выполнения процесса SQL Server, в результате чего в приложении может открыться сообщение об истечении времени ожидания.

RETURN

Служит для безусловного выхода из запроса или процедуры. Инструкция RETURN выполняется немедленно и может использоваться в любой точке для выхода из процедуры, пакета или блока инструкций. Инструкции, следующие после RETURN, не выполняются.

RETURN [integer_expression]

integer_expression - возвращаемое целочисленное значение.

```
DECLARE @A INT = 1, @result varchar(15)
/*Если значение переменной A меньше 0, то следующие команды не выполнятся, и на экран не выведется результат выполнения запроса в виде «Результат :»*/
IF @A < 0
RETURN
SET @result = 'A больше 0'
SELECT @result AS [Результат:]
```

Задание к практической работе

1. Определить и вывести имя локального SQL сервера и номер версии.
2. Определить количество мероприятий в определенный день и вывести сообщение, превышает ли их количество 30 шт. или нет.
3. Найти студента, у которого самый низкий средний балл. Добавить к характеристике этого студента «На контроле в дирекции».
4. Приостановить процесс выполнения кода до указанного времени.
5. Определить количество «2», «3», «4» и «5», выставленных студентам. Вывести, каких оценок больше в виде «Больше всего у студентов …».
6. Если студент в течение текущего месяца получил оценок «5», больше, чем 20 шт., и у него нет оценок «4», «3» и «2», то добавить ему к стипендии 5%;

   если студент в течение текущего месяца получил оценок «5», больше, чем 10 шт., оценок

   «4» не более 5 шт. и у него нет оценок «2» и «3», то добавить ему к стипендии 3%;

   в остальных случаях стипендию не изменять.