**Лабораторная работа № 8. Основы программирования на T-SQL**

**КУРНОСЕНКО СОФЬЯ**

Использование программ на языке T-SQL позволяет расширить круг решаемых задач, возникающих при работе с базами данных.

1. Разработать T-SQL-скрипт, в котором:

− объявить переменные типа char, varchar, datetime, time, int, smallint, tinint, numeric(12, 5);

− первые две переменные проинициализировать в операторе объявления;

− присвоить произвольные значения следующим двум переменным с помощью оператора SET, одной из этих переменных присвоить значение, полученное в результате запроса SELECT;

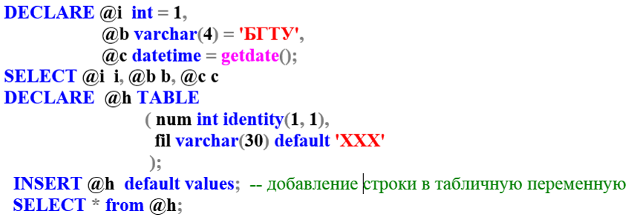
− одну из переменных оставить без инициализации и не присваивать ей значения, оставшимся переменным присвоить некоторые значения с помощью оператора SELECT;

− значения одной половины переменных вывести с помощью оператора SELECT,значения другой половины переменных распечатать с помощью оператора PRINT.

Проанализировать результаты.

***→ Теория (методичка)***

Для объявления переменных, используемых в программах, предназначен оператор DECLARE:



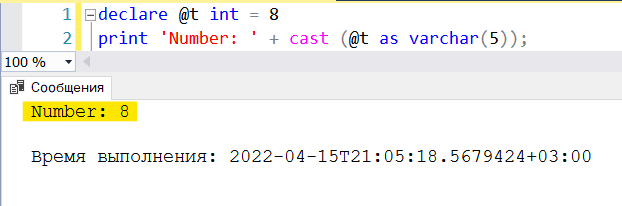
Имя переменной должно начинаться с символа @.

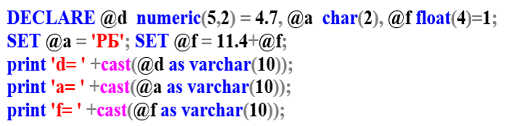
С помощью оператора SET можно переменной присвоить значение и выполнять вычисления. Оператор SELECT позволяет нескольким переменным присвоить значения.

Здесь **@h** − переменная типа TABLE. Этот тип позволяет создавать таблицы в памяти и использовать их для хранения промежуточных данных.

*Вывод данных* в T-SQL возможен двумя способами: оператором SELECT можно сформировать выходной результирующий набор и с помощью оператора PRINT можно вывести строку в стандартный выходной поток.

Если одновременно выводятся данные, сформированные оператором SELECT и данные, сформированные оператором PRINT, то посмотреть последние можно на вкладке Messages.

****

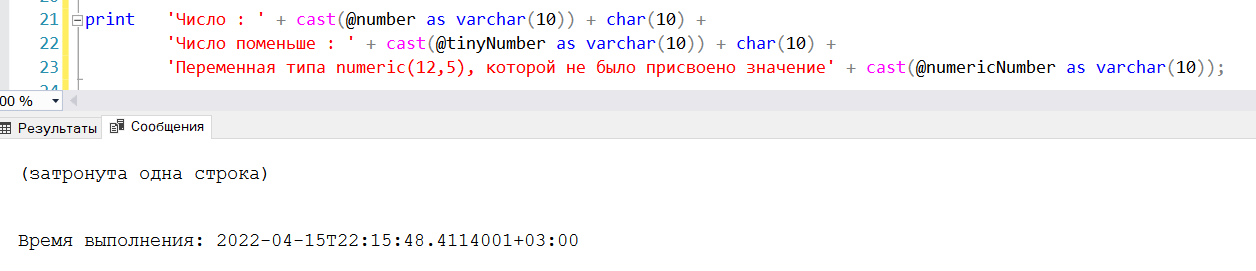


Функция CAST используется для преобразования типов.



***→ Выполнение заданий***

При попытке вывести значение переменной, которой не присвоено значение, ничего для соответствующего print вообще не отображается:



declare @symbol char(1) = '\*',

@str nvarchar(65) = 'Я решил жить, как падший ангел, то есть развращенным',

@currentDatetime datetime,

@currentHour time,

@number int,

@tinyNumber tinyint,

@numericNumber numeric(12,5);

set @currentDatetime = getdate();

set @number = (select count(\*) from FACULTY);

select @currentHour = cast(datepart(hour, getdate()) as datetime),

@tinyNumber = 1;

select @symbol as [символ],

@str as [строка],

@currentDatetime as [текущие дата и время],

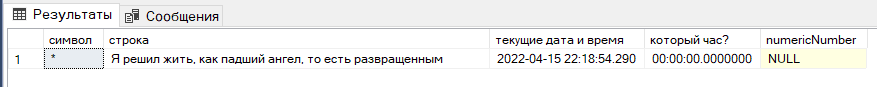
@currentHour as [который час?],

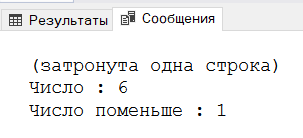
@numericNumber numericNumber;

-- char(10) - перенос строки

print 'Число : ' + cast(@number as varchar(10)) + char(10) +

'Число поменьше : ' + cast(@tinyNumber as varchar(10)) + char(10);





Для того, чтобы перенести текст на новую строку с возвратом каретки в MS SQL Server предусмотрены функции ([ссылка на офиц. сайт](https://remotehelper.ru/goto/https:/docs.microsoft.com/ru-ru/sql/t-sql/functions/char-transact-sql?view=aps-pdw-2016)):

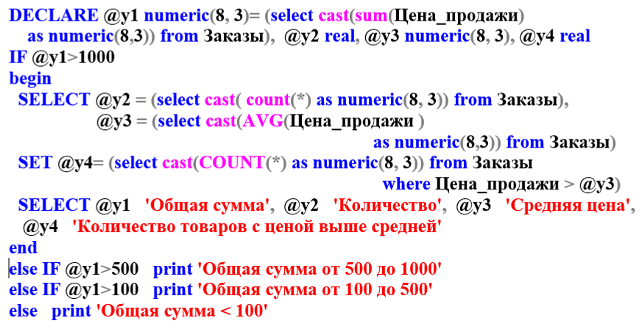
|  |  |
| --- | --- |
| Перевод строки | char(10) |
| Возврат каретки | char(13) |

2. Разработать скрипт, в котором определяется общая вместимость аудиторий. Когда общая вместимость превышает 200, то вывести количество аудиторий, среднюю вместимость аудиторий, количество аудиторий, вместимость которых меньше средней, и процент таких аудиторий. Когда общая вместимость аудиторий меньше 200, то вывести сообщение о размере общей вместимости.

***→ Теория (методичка)***

Пусть требуется определить общую сумму заказанных товаров, их количество, среднюю цену продажи, количество товаров, цены которых превышают среднюю и процент таких товаров в том случае, когда общая сумма превышает 1000.

В противном случае вывести сообщение о размере общей суммы:



***→ Выполнение заданий***

declare

@totalAuditoriumCapacity int = (

select sum(AUDITORIUM\_CAPACITY)

from AUDITORIUM

),

@amountOfAudit int,

@avgAuditCapacity float(10),

@lessAvgAuditAmount int,

@lessAvgPercent float(10);

IF @totalAuditoriumCapacity > 200

BEGIN

set @amountOfAudit = (

select count(\*) as [кол-во ауд.]

from AUDITORIUM

);

set @avgAuditCapacity = (

select cast(avg(AUDITORIUM\_CAPACITY) as float(10)) as [сред. вместимость ауд.]

from AUDITORIUM

);

set @lessAvgAuditAmount = (

select count(\*) as [кол-во ауд. вместимости меньше сред.]

from AUDITORIUM

where AUDITORIUM\_CAPACITY < @avgAuditCapacity

);

set @lessAvgPercent = cast((@lessAvgAuditAmount \* 100 / @amountOfAudit) as float(10));

-- вывод полученных значений

print 'Общая вместимость всех аудиторий : ' + cast(@totalAuditoriumCapacity as varchar);

print 'Кол-во аудиторий : ' + cast(@amountOfAudit as varchar);

print 'Средняя вместимость всех аудиторий : ' + cast(@avgAuditCapacity as varchar);

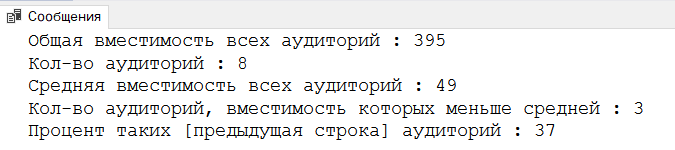
print 'Кол-во аудиторий, вместимость которых меньше средней : ' + cast(@lessAvgAuditAmount as varchar);

print 'Процент таких [предыдущая строка] аудиторий : ' + cast(@lessAvgPercent as varchar);

END

ELSE

print 'Общая вместимость : ' + cast(@totalAuditoriumCapacity as varchar);



1. Разработать T-SQL-скрипт, который выводит на печать глобальные переменные:

− @@ROWCOUNT (число обработанных строк);

− @@VERSION (версия SQL Server);

− @@SPID (возвращает системный идентификатор процесса, назначенный сервером текущему подключению);

− @@ERROR (код последней ошибки);

− @@SERVERNAME (имя сервера);

− @@TRANCOUNT (возвращает уровень вложенности транзакции);

− @@FETCH\_STATUS (проверка результата считывания строк результирующего набора);

− @@NESTLEVEL (уровень вложенности текущей процедуры).

Проанализировать результат.

***→ Выполнение заданий***

print 'Число обработанных строк : ' + cast(@@ROWCOUNT as varchar);

print 'Версия SQL Server : ' + cast(@@VERSION as varchar);

print 'Системный идентификатор процесса, назначенный сервером текущему подключению : ' + cast(@@SPID as varchar);

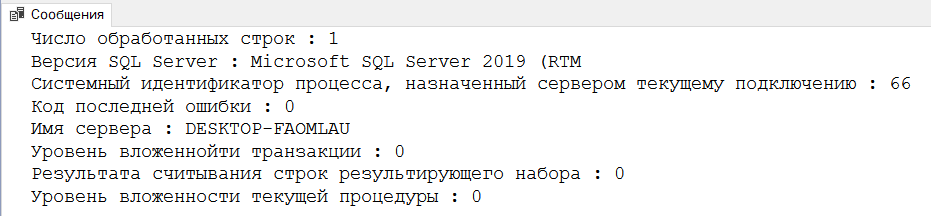
print 'Код последней ошибки : ' + cast(@@ERROR as varchar);

print 'Имя сервера : ' + cast(@@SERVERNAME as varchar);

print 'Уровень вложеннойти транзакции : ' + cast(@@TRANCOUNT as varchar);

print 'Результата считывания строк результирующего набора : ' + cast(@@FETCH\_STATUS as varchar);

print 'Уровень вложенности текущей процедуры : ' + cast(@@NESTLEVEL as varchar);



4. Разработать T-SQL-скрипты, выполняющие:

− вычисление значений переменной **z**



для различных значений исходных данных;

− преобразование полного ФИО студента в сокращенное (например, Макейчик Татьяна Леонидовна в Макейчик Т. Л.);

− поиск студентов, у которых день рождения в следующем месяце, и определение их возраста;

− поиск дня недели, в который студенты некоторой группы сдавали экзамен по СУБД.

***→ Теория (методичка)***

MSS поддерживает широкий набор встроенных математических функций, которые можно применять в сценариях T-SQL для вычислений:

**print 'Округление : '+ cast(round(12345.12345, 2) as varchar(12));**

**print 'Нижнее целое : '+ cast(floor(24.5) as varchar(12));**

**print 'Возведение в степень: '+ cast(power(12.0, 2) as varchar(12));**

**print 'Логарифм : '+ cast(log(144.0) as varchar(12));**

**print 'Корень квадратный : '+ cast(sqrt(144.0) as varchar(12));**

**print 'Экпонента : '+ cast(exp(4.96981) as varchar(12));**

**print 'Абсолютное значение : '+ cast(abs(-5) as varchar(12));**

**print 'Cинус : '+ cast(sin(pi()) as varchar(12));**

**print 'Подстрока : '+ substring('1234567890', 3,2);**

**print 'Удалить пробелы справа : '+ rtrim('12345 ') +'X';**

**print 'Удалить пробелы слева : '+ 'X'+ ltrim(' 67890');**

**print 'Нижний регистр : '+ lower ('ВЕРХНИЙ РЕГИСТР');**

**print 'Верхний регистр : '+ upper ('нижний регистр');**

**print 'Заменить : '+ replace('1234512345', '5', 'X');**

**print 'Строка пробелов : '+ 'X'+ space(5) +'X';**

**print 'Повторить строку : '+ replicate('12', 5);**

**print 'Найти по шаблону : '+ cast (patindex ('%Y\_Y%', '123456YxY7890') as varchar(5));**

**DECLARE @t time(7) = sysdatetime(), @dt datetime = getdate();**

**print 'Текущее время : '+ convert (varchar(12), @t);**

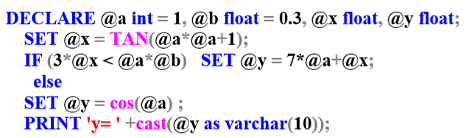
**print 'Текущая дата : '+ convert (varchar(12), @dt, 103);**

**print '+1 день : '+ convert(varchar(12), dateadd(d, 1, @dt), 103);**

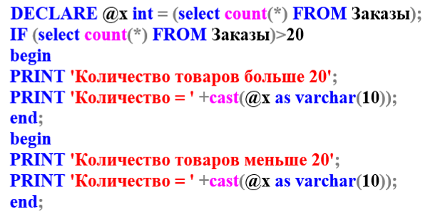
Функция CONVERT также используется для преобразования типов. Третий параметр этой функции применяется при преобразованиях, использующих типы данных для хранения даты и времени, и задает стиль представления этих данных.

Пусть требуется вычислить значение **y**:





С помощью операторных скобок BEGIN END можно объединять операторы в группы. Пусть надо проанализировать количество заказанных товаров:



***→ Выполнение заданий***

-- 4.1

declare @t int = 10,

@x int = 7,

@z float(10);

if(@t > @x)

set @z = power(sin(@t), 2);

else if(@t < @x)

set @z = 4 \* (@t + @x);

else

set @z = 1 - exp(@x -2);

print 'Значение переменной z : ' + convert(varchar(10), @z);



-- 4.2

declare @fio varchar(50) = 'Курносенко Софья Андреевна';

set @fio = replace(@fio, 'Софья','С.');

set @fio = replace(@fio, 'Андреевна','А.');

print 'ФИО : ' + @fio;



-- 4.3

declare @currentDatePlusOneMonth datetime = dateadd(month, 1, getdate());

select IDSTUDENT,

NAME,

BDAY,

-- datediff(year, BDAY, @currentDatePlusOneMonth) только отнимает от года даты currentDatePlusOneMonth год даты BDAY

-- мы получаем не возраст человека, а сколько ему лет должно исполнится в этом году

-- часть

--

-- case

-- when datepart(dayofyear, @currentDatePlusOneMonth) > datepart(dayofyear, BDAY) then 0

-- else 1

-- end

-- отвечает за то, чтобы отнять от результата единицу, если ДР человека ещё не было в этом году

datediff(

year,

BDAY,

@currentDatePlusOneMonth) -

case

when datepart(dayofyear, @currentDatePlusOneMonth) >= datepart(dayofyear, BDAY) then 0

else 1

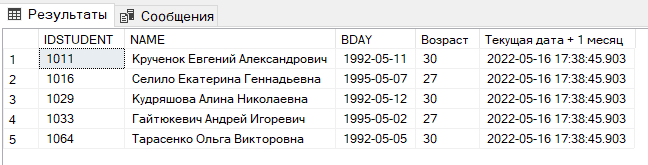
end

as Возраст,

@currentDatePlusOneMonth as [Текущая дата + 1 месяц]

from STUDENT

where datepart(month, BDAY) = datepart(month, @currentDatePlusOneMonth)



-- 4.4

select distinct

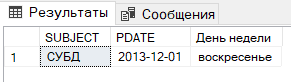
SUBJECT,

PDATE,

datename(weekday, PDATE) as [День недели]

from PROGRESS

where SUBJECT = 'СУБД';



5. Продемонстрировать конструкцию IF… ELSE на примере анализа данных таблиц базы данных Х\_UNIVER.

-- 5

declare @avgAudCap float(10) = (

select sum(AUDITORIUM\_CAPACITY)

from AUDITORIUM

);

if(@avgAudCap < 15)

print 'Средняя вместимость аудиторий меньше 15'

else if(@avgAudCap = 15)

print 'Средняя вместимость аудиторий равна 15'

else

print 'Средняя вместимость аудиторий больше 15'

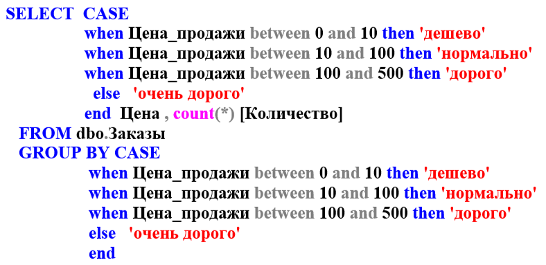


6. Разработать сценарий, в котором с помощью CASE анализируются оценки, полученные студентами некоторого факультета при сдаче экзаменов.

***→ Теория (методичка)***

Выражение CASE, как правило, применяется в SELECT-списке, секциях WHERE, ORDER BY, HAVING и служит для формирования одного из нескольких возможных значений.

Например, для анализа цен товаров можно использовать следующие операторы:



***→ Выполнение заданий***

-- BETWEEN value1 AND value2 - от value1 до value2, порядок значений важен!

select \*

from(

select case

when PROGRESS.NOTE between 8 and 10 then 'Is everything ok with your health, sunny?'

when PROGRESS.NOTE between 4 and 7 then 'Good kids'

when PROGRESS.NOTE < 4 then 'Damn, bitch, you live like this (I don''t judge)'

end as [Mark],

count(\*) [Amount of students]

from PROGRESS

group by case

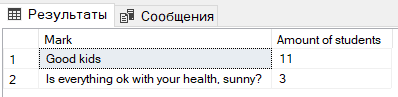
when PROGRESS.NOTE between 8 and 10 then 'Is everything ok with your health, sunny?'

when PROGRESS.NOTE between 4 and 7 then 'Good kids'

when PROGRESS.NOTE < 4 then 'Damn, bitch, you live like this (I don''t judge)'

end

) as T



7. Создать временную локальную таблицу из трех столбцов и 10 строк, заполнить ее и вывести содержимое. Использовать оператор WHILE.

***→ Теория (методичка)***

Основное отличие *временных* таблиц от постоянных в том, что они хранятся в системной базе данных TEMPDB и не могут иметь внешние ключи.

*Локальные* временные таблицы имеют имена, начинающиеся с символа # и доступны только пользователю, их создавшему.

Сформировать временную локальную таблицу можно с помощью оператора CREATE:

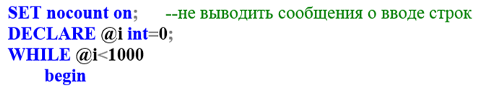
**CREATE table #EXPLRE**

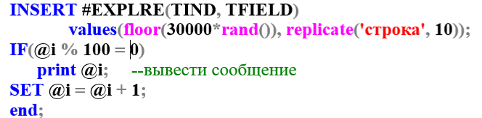
**( TIND int,**

**TFIELD varchar(100)**

**);**

Заполнить таблицу можно, например, с помощью следующего скрипта:





Функция RAND генерирует случайное число.

*Глобальные* временные таблицы имеют имена, начинающиеся с символов **##** и доступны всем пользователям, подключенным к серверу.

***→ Выполнение заданий***

set nocount on; -- не выводить сообщения о затронутых строках

create table #tempLocTable(

number1 int,

dateTimeNow datetime,

string varchar(10)

);

declare @i int = 0;

while @i < 10

begin

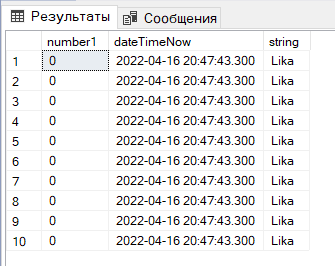
insert #tempLocTable

values(convert(int, rand())%10000, getdate(), 'Lika');

set @i += 1;

end;

select \* from #tempLocTable;

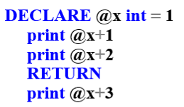


Оператор SET NOCOUNT ON предотвращает появление сообщения, показывающего количество строк, затронутых операторами запроса T-SQL. Это используется в хранимых процедурах и триггерах, чтобы избежать отображения сообщения о затронутых строках. Использование SET NOCOUNT ON внутри хранимой процедуры может повысить производительность хранимой процедуры.

8. Разработать скрипт, демонстрирующий использование оператора RETURN.

***→ Теория (методичка)***

Оператор RETURN служит для немедленного завершения работы пакета:



***→ Выполнение заданий***

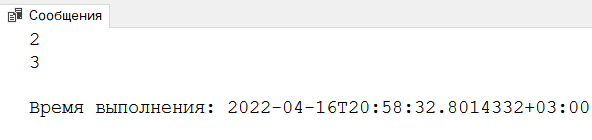
declare @value int = 1;

print @value + 1;

print @value + 2;

RETURN

print @value + 3;

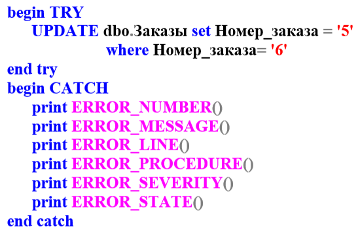


9. Разработать сценарий с ошибками, в котором используются для обработки ошибок блоки TRY и CATCH. Применить функции ERROR\_NUMBER (код последней ошибки), ERROR\_MESSAGE (сообщение об ошибке), ERROR\_LINE (номер строки возникновения ошибки), ERROR\_PROCEDURE (имя процедуры или NULL), ERROR\_SEVERITY (уровень серьезности ошибки), ERROR\_ STATE (номер состояния ошибки). Проанализировать результат.

***→ Теория (методичка)***

Для обработки ошибок выполнения в сценарии T-SQL предусмотрена конструкция, состоящая из двух блоков: TRY и CATCH. Блок TRY содержит код T-SQL, в котором могут возникнуть ошибки, а блок CATCH – код, предназначенный для обработки ошибок.

Ошибка, возникающая в охраняемом коде, приводит к передаче управления в блок обработки ошибок:



***→ Выполнение заданий***

declare @num int = 9;

begin try

print 'Число : ' + @num;

end try

begin catch

print 'Код ошибки : ' + convert(varchar, ERROR\_NUMBER());

print 'Сообщение об ошибке : ' + convert(varchar, ERROR\_MESSAGE());

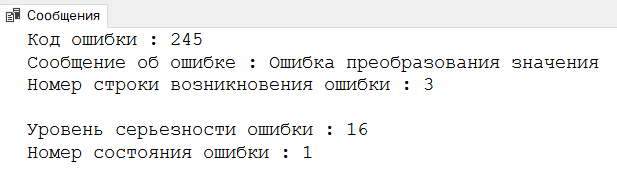
print 'Номер строки возникновения ошибки : ' + convert(varchar, ERROR\_LINE());

print 'Имя процедуры или NULL : ' + convert(varchar, ERROR\_PROCEDURE());

print 'Уровень серьезности ошибки : ' + convert(varchar, ERROR\_SEVERITY());

print 'Номер состояния ошибки : ' + convert(varchar, ERROR\_STATE());

end catch

******