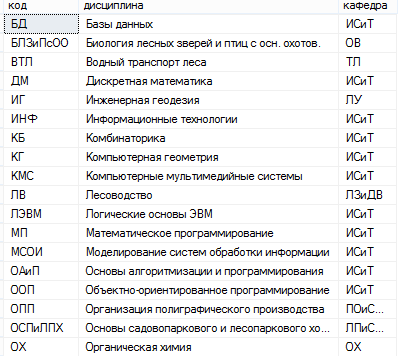
**Лабораторная работа № 12. Разработка хранимых процедур**

1. Разработать хранимую процедуру без параметров с именем **PSUBJECT**. Процедура формирует результирующий набор на основе таблицы **SUBJECT**, аналогичный набору, представленному на рисунке:

****

К точке вызова процедура должна возвращать количество строк, выведенных в результирующий набор.

use UNIVER;

set nocount on;

GO

CREATE PROCEDURE PSUBJECT AS

BEGIN

DECLARE @count int;

SELECT \* FROM dbo.SUBJECT\_T;

SET @count = (SELECT COUNT(\*) FROM SUBJECT\_T);

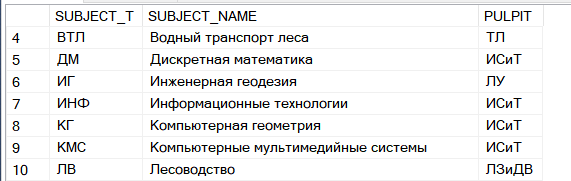
RETURN @count;

END;

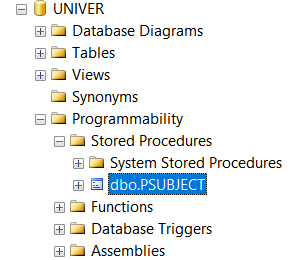
DECLARE @c int;

EXEC @c = PSUBJECT;

PRINT 'Count of rows in SUBJECT table: ' + CAST(@c as varchar(5));



1. Найти процедуру **PSUBJECT** с помощью обозревателя объектов (Object Explorer) SSMS и через контекстное меню создать сценарий на изменение процедуры оператором ALTER. Изменить процедуру **PSUBJECT**, созданную в задании 1, таким образом, чтобы она принимала два параметра с именами **@p** и **@c**. Параметр **@p** является входным, имеет тип VARCHAR(20) и значение по умолчанию NULL. Параметр **@с** является выходным, имеет тип INT. Процедура **PSUBJECT** должна формировать результирующий набор, аналогичный набору, представленному на рисунке выше, но при этом содержать строки, соответствующие коду кафедры, заданному параметром **@p**. Кроме того, процедура должна формировать значение выходного параметра **@с**, равное количеству строк в результирующем наборе, а также возвращать значение к точке вызова, равное общему количеству дисциплин (количеству строк в таблице **SUBJECT**).



USE [UNIVER]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[PSUBJECT] Script Date: 06.04.2022 16:18:28 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[PSUBJECT] @param varchar(20), @cout int output AS

BEGIN

DECLARE @count int;

SELECT \* FROM dbo.SUBJECT\_T WHERE SUBJECT\_T.PULPIT = @param;

SET @count = (SELECT COUNT(\*) FROM SUBJECT\_T WHERE SUBJECT\_T.PULPIT = @param);

PRINT 'Params: @p = ' + @param + ' @c = ' + CAST(@cout as varchar(5));

RETURN @count;

END;

GO

use UNIVER;

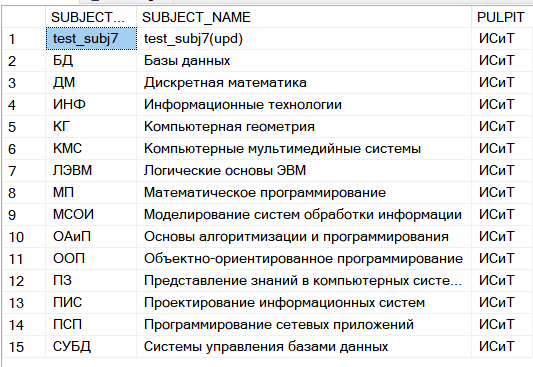
set nocount on;

DECLARE @k int, @r int, @param varchar(20);

EXEC @k = PSUBJECT @param = 'ИСиТ', @cout = @r output;

PRINT 'Count of rows in SUBJECT table: ' + CAST(@r as varchar(5));

PRINT 'Count of rows in SUBJECT table: ' + CAST(@k as varchar(5));



1. Создать временную локальную таблицу с именем **#SUBJECT**. Наименование и тип столбцов таблицы должны соответствовать столбцам результирующего набора процедуры **PSUBJECT**, разработанной в задании 2. Изменить процедуру **PSUBJECT** таким образом, чтобы она не содержала выходного параметра. Применив конструкцию INSERT… EXECUTE с модифицированной процедурой **PSUBJECT**, добавить строки в таблицу **#SUBJECT**.

use UNIVER;

GO

CREATE procedure PSUBJECT\_3 @param varchar(20) AS

BEGIN

DECLARE @count int;

SET @count = (SELECT COUNT(\*) FROM SUBJECT\_T WHERE SUBJECT\_T.PULPIT = @param);

PRINT 'Params: @p = ' + @param + ' @c = ' + CAST(@count as varchar(5));

SELECT \* FROM dbo.SUBJECT\_T WHERE SUBJECT\_T.PULPIT = @param;

END;

CREATE TABLE #SUBJECT (

SUBJ varchar(10) NOT NULL,

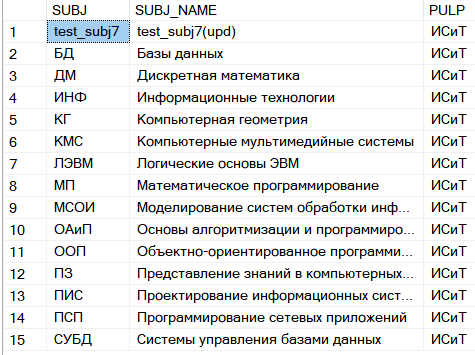
SUBJ\_NAME varchar(100),

PULP varchar(10) NOT NULL

);

INSERT #SUBJECT EXEC PSUBJECT\_3 @param = 'ИСиТ';

SELECT \* FROM #SUBJECT;



1. Разработать процедуру с именем **PAUDITORIUM\_INSERT**. Процедура принимает четыре входных параметра: **@a**, **@n**, **@c** и **@t**. Параметр **@a** имеет тип CHAR(20), параметр **@n** имеет тип VARCHAR(50), параметр **@c** имеет тип INT и значение по умолчанию **0**, параметр **@t** имеет тип CHAR(10). Процедура добавляет строку в таблицу **AUDITORIUM**. Значения столбцов **AUDITORIUM**, **AUDITORIUM\_NAME, AUDITORIUM\_CAPACITY** и **AUDITORIUM\_TYPE** добавляемой строки задаются соответственно параметрами **@a**, **@n**, **@c** и **@t**. Процедура **PAUDITORIUM\_INSERT** должна применять механизм TRY/CATCH для обработки ошибок. В случае возникновения ошибки, процедура должна формировать сообщение, содержащее код ошибки, уровень серьезности и текст сообщения в стандартный выходной поток. Процедура должна возвращать к точке вызова значение **-1** в том случае, если произошла ошибка и **1**, если выполнение успешно. Опробовать работу процедуры с различными значениями исходных данных, которые вставляются в таблицу.

use UNIVER;

SET nocount on;

GO

CREATE procedure PAUDITORIUM\_INSERT @a char(20), @n varchar(50), @c int = 0, @t char(10) AS

BEGIN

BEGIN try

INSERT into AUDITORIUM(AUDITORIUM,AUDITORIUM\_NAME,AUDITORIUM\_CAPACITY,AUDITORIUM\_TYPE)

values(@a, @n, @c, @t);

RETURN 1;

END try

BEGIN catch

print 'Error number: ' + cast(ERROR\_NUMBER() as varchar(6));

print 'Error message: ' + ERROR\_MESSAGE();

print 'Error line: ' + cast(ERROR\_LINE()as varchar(8));

if ERROR\_PROCEDURE() is not null

print 'Error procedure: ' + ERROR\_PROCEDURE();

print 'Error secerity: ' + cast(ERROR\_SEVERITY()as varchar(6));

print 'Error state: ' + cast(ERROR\_STATE()as varchar(8));

RETURN -1;

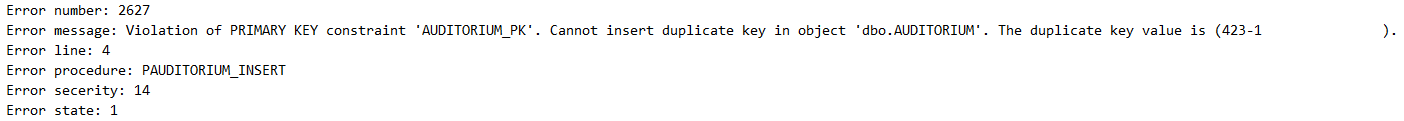
END CATCH

END;

DECLARE @paud int = 0;

EXEC @paud = PAUDITORIUM\_INSERT @a='423-1',@n='423-1',@c='90',@t='ЛК-К';

DELETE AUDITORIUM where AUDITORIUM\_NAME='423-1';



1. Разработать процедуру с именем **SUBJECT\_REPORT**, формирующую в стандартный выходной поток отчет со списком дисциплин на конкретной кафедре. В отчет должны быть выведены краткие названия (поле SUBJECT) из таблицы SUBJECT в одну строку через запятую (использовать встроенную функцию RTRIM). Процедура имеет входной параметр с именем **@p** типа CHAR(10), который предназначен для указания кода кафедры. В том случае, если по заданному значению **@p** невозможно определить код кафедры, процедура должна генерировать ошибку с сообщением **ошибка в параметрах**. Процедура **SUBJECT\_REPORT** должна возвращать к точке вызова количество дисциплин, отображенных в отчете.

CREATE procedure SUBJECT\_REPORT @pulpit varchar(20) AS

BEGIN

BEGIN try

if not exists (select \* from SUBJECT\_T where SUBJECT\_T.PULPIT = @pulpit)

RAISERROR('ошибка в параметрах', 11, 1);

else

BEGIN

DECLARE pulpit\_subjects CURSOR FOR

SELECT SUBJECT\_T.SUBJECT\_T

FROM SUBJECT\_T

WHERE SUBJECT\_T.PULPIT = @pulpit;

DECLARE @subject char(30), @subjects char(500) ='', @count int = 0;

OPEN pulpit\_subjects;

FETCH pulpit\_subjects into @subject;

PRINT 'Дисциплины кафедры ИСиТ:';

WHILE @@FETCH\_STATUS = 0

BEGIN

SET @subjects = RTRIM(@subject) +', ' + @subjects;

FETCH pulpit\_subjects into @subject;

SET @count = @count + 1;

END;

PRINT @subjects;

CLOSE pulpit\_subjects;

RETURN @count;

END

END try

BEGIN catch

print 'ошибка в параметрах.'

if ERROR\_PROCEDURE() is not null

print 'Error procedure:' + error\_procedure();

return -1;

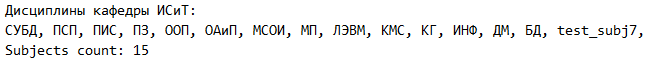
END catch;

END;

DECLARE @res\_rep int;

exec @res\_rep = SUBJECT\_REPORT @pulpit = 'ИСиТ';

print 'Subjects count: ' + cast(@res\_rep as varchar(3));



1. Разработать процедуру с именем **PAUDITORIUM\_INSERTX**. Процедура принимает пять входных параметров: **@a**, **@n**, **@c**, **@t** и **@tn**. Параметры **@a**, **@n**, **@c**, **@t** аналогичны параметрам процедуры **PAUDITORIUM\_INSERT**. Дополнительный параметр **@tn** является входным, имеет тип VARCHAR(50), предназначен для ввода значения в столбец **AUDITORIUM\_TYPE**.**AUDITORIUM\_TYPENAME**. Процедура добавляет две строки. Первая строка добавляется в таблицу **AUDITORIUM\_TYPE**. Значения столбцов **AUDITORIUM\_TYPE** и **AUDITORIUM\_ TYPENAME** добавляемой строки задаются соответственно параметрами **@t** и **@tn**. Вторая строка добавляется путем вызова процедуры **PAUDITORIUM\_INSERT**. Добавление строки в таблицу **AUDITORIUM\_TYPE** и вызов процедуры **PAUDITORIUM\_INSERT** должны выполняться в рамках одной явной транзакции с уровнем изолированности SERIALIZABLE. В процедуре должна быть предусмотрена обработка ошибок с помощью механизма TRY/CATCH. Все ошибки должны быть обработаны с выдачей соответствующего сообщения в стандартный выходной поток. Процедура **PAUDITORIUM\_INSERTX** должна возвращать к точке вызова значение **-1** в том случае, если произошла ошибка и **1**, если выполнения процедуры завершилось успешно.

CREATE procedure PAUDITORIUM\_INSERTX @a char(20), @n varchar(50), @c int = 0, @t char(10), @tn varchar(50) AS

BEGIN

BEGIN try

set transaction isolation level SERIALIZABLE;

begin transaction

INSERT into AUDITORIUM\_TYPE (AUDITORIUM\_TYPE,AUDITORIUM\_TYPENAME)

values(@t,@tn);

EXEC PAUDITORIUM\_INSERT @a=@a, @n=@n, @c=@c, @t=@t;

commit transaction;

return 1;

END try

BEGIN catch

print 'Error number: ' + cast(ERROR\_NUMBER() as varchar(6));

print 'Error message: ' + ERROR\_MESSAGE();

print 'Error line: ' + cast(ERROR\_LINE()as varchar(8));

if ERROR\_PROCEDURE() is not null

print 'Error procedure: ' + ERROR\_PROCEDURE();

print 'Error secerity: ' + cast(ERROR\_SEVERITY()as varchar(6));

print 'Error state: ' + cast(ERROR\_STATE()as varchar(8));

RETURN -1;

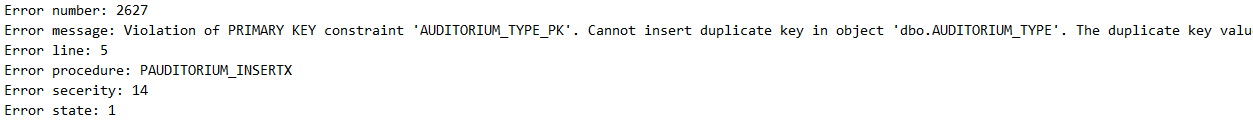
END catch

END;

GO

DECLARE @paud int = 0;

EXEC @paud = PAUDITORIUM\_INSERTX @a='423-1',@n='423-1',@c='90',@t='ЛК-К', @tn ='test auditorium name';



1. Разработать хранимые процедуры для базы данных **X\_MyBASE** и продемонстрировать их работу.

-- #1

use Dobriyan\_MyBase;

set nocount on;

GO

CREATE PROCEDURE PCOURSES AS

BEGIN

DECLARE @count int;

SELECT \* FROM dbo.Courses;

SET @count = (SELECT COUNT(\*) FROM Courses);

RETURN @count;

END;

GO

DECLARE @c int;

EXEC @c = PCOURSES;

PRINT 'Count of rows in Courses table: ' + CAST(@c as varchar(5));

-- #3

use Dobriyan\_MyBase;

GO

CREATE PROCEDURE PGROUPS @speciality int AS

BEGIN

DECLARE @count int;

SET @count = (SELECT COUNT(\*) FROM Groups WHERE Groups.Speciality = @speciality);

SELECT \* FROM dbo.Groups WHERE Groups.Speciality = @speciality;

END;

GO

CREATE TABLE #GROUP (

NUM int NOT NULL,

STUD\_COUNT int,

SPEC int NOT NULL,

DEPART varchar(20)

);

INSERT #GROUP EXEC PGROUPS @speciality = 1;

SELECT \* FROM #GROUP;

GO

-- #4

use Dobriyan\_MyBase;

SET nocount on;

GO

CREATE procedure PSPEC\_INSERT @id int, @spec varchar(10) AS

BEGIN

BEGIN try

INSERT into Specialities values(@id, @spec);

RETURN 1;

END try

BEGIN catch

print 'Error number: ' + cast(ERROR\_NUMBER() as varchar(6));

print 'Error message: ' + ERROR\_MESSAGE();

print 'Error line: ' + cast(ERROR\_LINE()as varchar(8));

if ERROR\_PROCEDURE() is not null

print 'Error procedure: ' + ERROR\_PROCEDURE();

print 'Error secerity: ' + cast(ERROR\_SEVERITY()as varchar(6));

print 'Error state: ' + cast(ERROR\_STATE()as varchar(8));

RETURN -1;

END CATCH

END;

GO

DECLARE @pspec int = 0;

EXEC @pspec = PSPEC\_INSERT @id = 5, @spec = 'test spec';

SELECT \* from Specialities;

DELETE Specialities where Specialities.Speciality = 'test spec';

GO

-- #5

use Dobriyan\_MyBase;

SET nocount on;

GO

CREATE procedure GROUPS\_REPORT @spec int AS

BEGIN

BEGIN try

if not exists (select \* from Groups where Groups.Speciality = @spec)

RAISERROR('ошибка в параметрах', 11, 1);

else

BEGIN

DECLARE spec\_average\_stud\_count CURSOR LOCAL FOR

SELECT Groups.Students\_Count

FROM Groups

WHERE Groups.Speciality = @spec;

DECLARE @stud\_count int, @stud\_counts varchar(200) = '', @count int = 0, @average int = 0;

OPEN spec\_average\_stud\_count;

FETCH spec\_average\_stud\_count into @stud\_count;

PRINT 'Количество студентов в группах определенной специальности:';

WHILE @@FETCH\_STATUS = 0

BEGIN

SET @stud\_counts = CAST(@stud\_count as varchar(10)) + ', ' + @stud\_counts;

SET @average = @average + @stud\_count;

FETCH spec\_average\_stud\_count into @stud\_count;

SET @count = @count + 1;

END;

PRINT @stud\_counts;

CLOSE spec\_average\_stud\_count;

RETURN @average / @count;

END

END try

BEGIN catch

print 'Error message: ' + ERROR\_MESSAGE();

if ERROR\_PROCEDURE() is not null

print 'Error procedure:' + error\_procedure();

return -1;

END catch;

END;

GO

DECLARE @res\_rep int;

exec @res\_rep = GROUPS\_REPORT @spec = 2;

print 'Average students count: ' + cast(@res\_rep as varchar(3));

GO

-- #6

use Dobriyan\_MyBase;

SET nocount on;

GO

CREATE procedure PSPEC\_INSERTX @id int, @spec varchar(10) AS

BEGIN

BEGIN try

set transaction isolation level SERIALIZABLE;

begin transaction

INSERT into Specialities values(@id, @spec);

EXEC PSPEC\_INSERTX @id = @id, @spec = @spec;

commit transaction;

return 1;

END try

BEGIN catch

print 'Error number: ' + cast(ERROR\_NUMBER() as varchar(6));

print 'Error message: ' + ERROR\_MESSAGE();

print 'Error line: ' + cast(ERROR\_LINE()as varchar(8));

if ERROR\_PROCEDURE() is not null

print 'Error procedure: ' + ERROR\_PROCEDURE();

print 'Error secerity: ' + cast(ERROR\_SEVERITY()as varchar(6));

print 'Error state: ' + cast(ERROR\_STATE()as varchar(8));

RETURN -1;

END catch

END;

GO

DECLARE @pspec int = 0;

EXEC @pspec = PSPEC\_INSERTX @id = 5, @spec = 'test spec';

SELECT \* from Specialities;

DELETE Specialities WHERE Speciality = 'test spec';