Первое задание включает в себя создание хранимой процедуры – рисунок 1.1

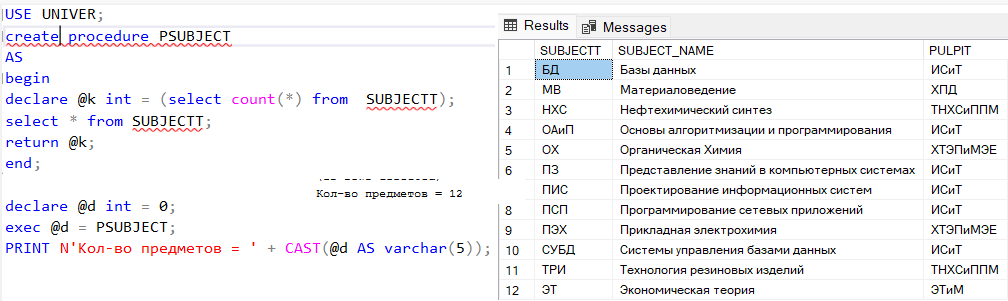


Рисунок 1.1 – Создание процедуры PSUBJECT

Мы создаём процедуру, которая выводит все строки таблицы SUBJECTT, а также возвращает значение, равное числу строк в таблице.

Во втором задании необходимо было изменить процедуру с помощью команды ALTER так, чтобы она принимала один входной параметр – название кафедры, и один выходной параметр - кол-во предметов на кафедре. Процедура должна возвращать те дисциплины, которые содержатся на указанной кафедре – рисунок 1.2.

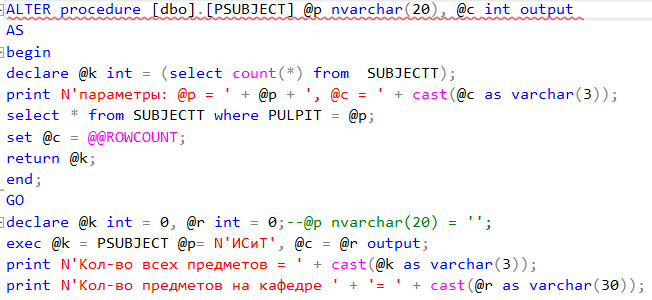


Рисунок 1.2 – Модификация процедуры PSUBJECT

В следующем задании необходимо было создать временную таблицу «#SUBJECT», и изменить процедуру, использованную ранее так, чтобы она производила выборку по указанной кафедре в таблице SUBJECT. В дальнейшем это необходимо для её использования в INSERT, чтобы тот результирующий набор из процедуры был занесён в нашу временную таблицу – рисунок 1.3.

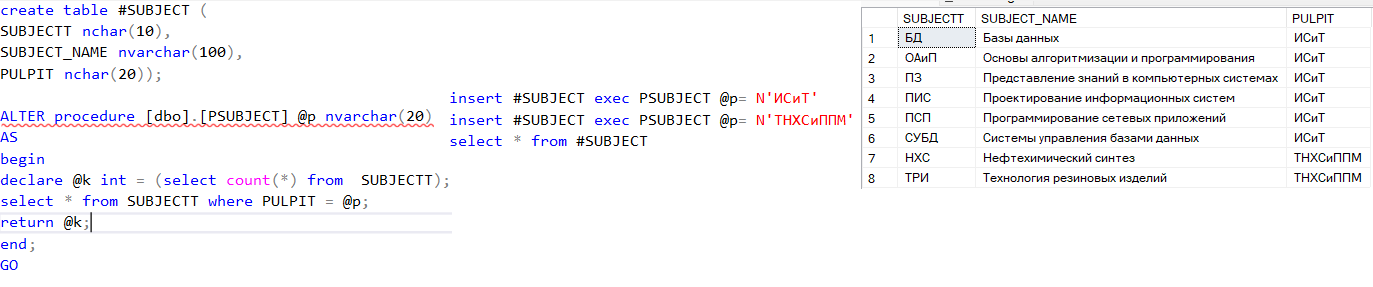


Рисунок 1.3 – Заполнение временной таблицы с помощью процедуры

Следующее задание - Разработать процедуру с именем **PAUDITORIUM\_INSERT,** которая будет принимать 4 параметра, соответствующие столбцам таблицы AUDITORIUM, и будет производить вставку в эту самую таблицу, плюс ко всему будет реализовывать блок TRY/CATCH. Процедура должна возвращать к точке вызова значение **-1** в том случае, если произошла ошибка и **1**, если выполнение успешно – рисунок 1.4.



Рисунок 1.4 – Процедура PAUDITORIUM\_INSERT

В пятом задании требовалось 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 – рисунок 1.5.

В том случае, если по заданному значению @p невозможно определить код кафедры, процедура должна генерировать ошибку с сообщением ошибка в параметрах.

alter procedure SUBJECT\_REPORT @p nchar(20)

as

declare @rc int = 0;

begin try

declare @tv nchar(20), @t nchar(300) = '';

declare SUBREP cursor for

select SUBJECTT from SUBJECTT where PULPIT = @p;

IF not exists (select SUBJECTT from SUBJECTT where PULPIT = @p)

raiserror(N'ошибка', 11, 1);

else open SUBREP;

FETCH SUBREP into @tv;

print N'Предметы кафедры: ';

while @@FETCH\_STATUS = 0

begin

set @t = RTRIM(@tv) +', ' + @t;

set @rc = @rc + 1;

fetch SUBREP into @tv;

end;

print @t;

close SUBREP;

deallocate SUBREP;

return @rc;

end try

begin catch

print N'ошибка в параметре ' + @p

IF ERROR\_PROCEDURE() is not null

print N'имя процедуры:' + error\_procedure();

return @rc;

end catch;

Код 1.1 – Процедура SUBJECT\_REPORT

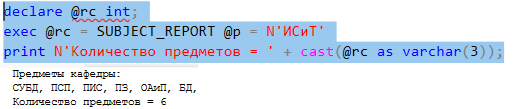


Рисунок 1.5 – Выполнение процедуры SUBJECT\_REPORT

Шестое задание заключало в себе создание процедуры PAUDITORIUM\_INSERTX, которое использует процедуру PAUDITORIUM\_INSERT, но также принимает дополнительные параметры, чтобы осуществить вставку в таблицу AUDITORIUM\_TYPE – рисунок 1.6. Необходимо учесть, что добавление строки в таблицу **AUDITORIUM\_TYPE** и вызов процедуры **PAUDITORIUM\_INSERT** должны выполняться в рамках одной явной транзакции с уровнем изолированности SERIALIZABLE для разрешение конфликтов при параллельных транзакциях.

Далее применим изученные знания для создания процедур в базе данных Л\_Mybase.

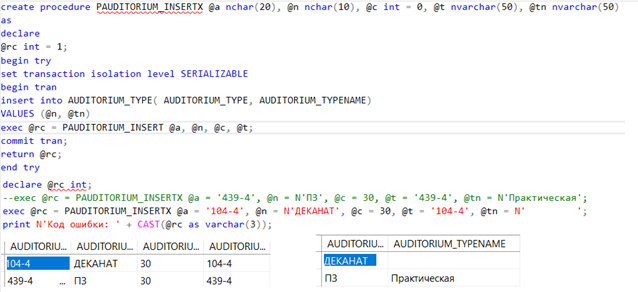


Рисунок 1.6 – Процедура PAUDITORIUM\_INSERTX

**Вывод:**

В результате данной лабораторной работе были получены теоретические знания о процедурах – объектах базы данных, которая выполняется через команду EXEC. Решены типовые задания, возникающие при необходимости использования процедур.