Учреждение образования “БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ”

Лабораторная работа № 12. Разработка хранимых процедур

Выполнил:

студент 2 курса, 7 группы

Бобрович Глеб Сергеевич

Минск 2022ПроТТеоре

|  |  |
| --- | --- |
| **Задание** |  |
| 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.  2.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**). | | **Задание 1**  USE ЛР12;  go  CREATE procedure PSUBJECT  as begin  DECLARE @n int = (SELECT count(\*) from SUBJECT);  SELECT SUBJECT [КОД], SUBJECT\_NAME [ДИСЦИПЛИНА], PULPIT [КАФЕДРА] from SUBJECT;  return @n;  end;  DECLARE @k int;  EXEC @k = PSUBJECT; -- вызов процедуры  print 'Количество предметов: ' + cast(@k as varchar(3));  go  --DROP procedure PSUBJECT;    **Задание 2**  USE ЛР12;  GO  ALTER procedure PSUBJECT @p varchar(20), @c nvarchar(2) output  as begin  SELECT \* from SUBJECT where SUBJECT = @p;  set @c = cast(@@rowcount as nvarchar(2));  end;  DECLARE @k1 int, @k2 nvarchar(2);  EXEC @k1 = PSUBJECT @p = 'СУБД', @c = @k2 output;  print 'Количество предметов: ' + @k2;  go | |
| 3.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**. | | **Задание 3**  USE ЛР12;  GO  ALTER procedure PSUBJECT @p varchar(20)  as begin  SELECT \* from SUBJECT where SUBJECT = @p;  end;  CREATE table #SUBJECTs  (  Код\_предмета varchar(20),  Название\_предмета varchar(100),  Кафедра varchar(20)  );  INSERT #SUBJECTs EXEC PSUBJECT @p = 'ПСП';  INSERT #SUBJECTs EXEC PSUBJECT @p = 'СУБД';  SELECT \* from #SUBJECTs;  go  --drop table #SUBJECTs | |
| 4.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. | | **Задание 4**  USE ЛР12;  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\_TYPE, AUDITORIUM\_CAPACITY, AUDITORIUM\_NAME)  values(@a, @n, @c, @t);  --values(433-1, 'ЛК', 433-1, 100);  return 1;  end try  begin catch  print 'Номер ошибки: ' + cast(error\_number() as varchar(6));  print 'Сообщение: ' + error\_message();  print 'Уровень: ' + cast(error\_severity() as varchar(6));  print 'Метка: ' + cast(error\_state() as varchar(8));  print 'Номер строки: ' + cast(error\_line() as varchar(8));  if error\_procedure() is not null  print 'Имя процедуры: ' + error\_procedure();  return -1;  end catch;  end;  DECLARE @rc int;  EXEC @rc = PAUDITORIUM\_INSERT @a = '420-3', @n = 'ЛК', @c = 100, @t = '420-3';  print 'Код ошибки: ' + cast(@rc as varchar(3));  go  delete AUDITORIUM where AUDITORIUM='420-3'; | |
| 5.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. | | **Задание 5**  USE ЛР12;  GO  CREATE procedure SUBJECT\_REPORT @p char(10)  as begin  DECLARE @rc int;  begin try  DECLARE @sb char(10), @r varchar(100) = '';  DECLARE sbj CURSOR for  SELECT SUBJECT from SUBJECT where PULPIT = @p;  if not exists(SELECT SUBJECT from SUBJECT where PULPIT = @p)  raiserror('Ошибка', 11, 1);  else  OPEN sbj;  fetch sbj into @sb;  print 'Предметы: ';  while @@fetch\_status = 0  begin  set @r = rtrim(@sb) + ', ' + @r;  set @rc = @rc + 1;  fetch sbj into @sb;  end  print @r;  CLOSE sbj;  return @rc;  end try  begin catch  print 'Ошибка в параметрах'  if error\_procedure() is not null  print 'Имя процедуры: ' + error\_procedure();  return @rc;  end catch;  end;  DECLARE @k2 int;  EXEC @k2 = SUBJECT\_REPORT @p ='ИСиТ';  print 'Количество предметов: ' + cast(@k2 as varchar(3));  go  drop procedure SUBJECT\_REPORT; | |
| 6.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. | | **Задание 6**  USE ЛР12;  GO  CREATE procedure PAUDITORIUM\_INSERTX  @a char(20),  @n varchar(50),  @c int = 0,  @t char(10),  @tn varchar(50) --доп., для ввода в AUD\_TYPEAUD\_TYPENAME  as begin  DECLARE @rc int = 1;  begin try  set transaction isolation level serializable;  begin tran  INSERT into AUDITORIUM\_TYPE(AUDITORIUM\_TYPE, AUDITORIUM\_TYPENAME)  values(@n, @tn);  EXEC @rc = PAUDITORIUM\_INSERT @a, @n, @c, @t;  commit tran;  return @rc;  end try  begin catch  print 'Номер ошибки: ' + cast(error\_number() as varchar(6));  print 'Сообщение: ' + error\_message();  print 'Уровень: ' + cast(error\_severity() as varchar(6));  print 'Метка: ' + cast(error\_state() as varchar(8));  print 'Номер строки: ' + cast(error\_line() as varchar(8));  if error\_procedure() is not null  print 'Имя процедуры: ' + error\_procedure();  if @@trancount > 0 rollback tran ;  return -1;  end catch;  end;  DECLARE @k3 int;  EXEC @k3 = PAUDITORIUM\_INSERTX '622-3', @n = 'КГ', @c = 85, @t = '622-3', @tn = 'Комп. гласс';  print 'Код ошибки: ' + cast(@k3 as varchar(3));  delete AUDITORIUM where AUDITORIUM='622-3';  delete AUDITORIUM\_TYPE where AUDITORIUM\_TYPE='КГ';  go  drop procedure PAUDITORIUM\_INSERTX; | |