

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

Институт кибербезопасности и цифровых технологий

Кафедра КБ-2 «Информационно-аналитические системы кибербезопасности»

Отчет по практической работе № 5

по дисциплине «Безопасность систем баз данных»

«Среда управления SQL Server Management Studio. ЯзыкTransact-SQL. Компоненты SQL»

Выполнил студент 3 курса по специальности (направлению подготовки):

10.05.04 «Информационно-аналитические системы безопасности»

Зубарев Василий Сергеевич

|  |  |
| --- | --- |
|  | Проверил:  старший преподаватель Войтенков Д.В.  Отметка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Москва 2025

**Цели работы:**

1. Процедурные расширения
2. Хранимые процедуры
3. Определяемые пользователем функции

В этой работе мы познакомимся с пакетами и подпрограммами.

**Пример 1. Исполнение блока инструкций посредством инструкции if**

use sample\_Zub

IF (SELECT COUNT(\*)

FROM works\_on

WHERE project\_no = 'p1'

GROUP BY project\_no) > 3

BEGIN

PRINT 'The number of employees in the project p1 is 4 or more';

END

ELSE

BEGIN

PRINT 'The following employees work for the project p1';

SELECT e.emp\_fname, e.emp\_lname

FROM employee e

JOIN works\_on w ON e.emp\_no = w.emp\_no

WHERE w.project\_no = 'p1';

END

В примере 1 демонстрируется использование блока инструкций внутри инструкции if. Следующее далее логическое выражение инструкции if

(SELЕCT COUNT(\*)

FROM works\_on

WHERЕ projectno = ’p1’

GROP BY projectno) > 3

возвращает значение true (истина) для базы данных sample. Поэтому будет выполняться инструкция print, входящая в часть инструкции if. Обратите внимание на то обстоятельство, что в этом примере используется подзапрос, чтобы возвратить число строк (посредством агрегатной функции count), удовлетворяющих условию предложения where (project\_no=’p1’).

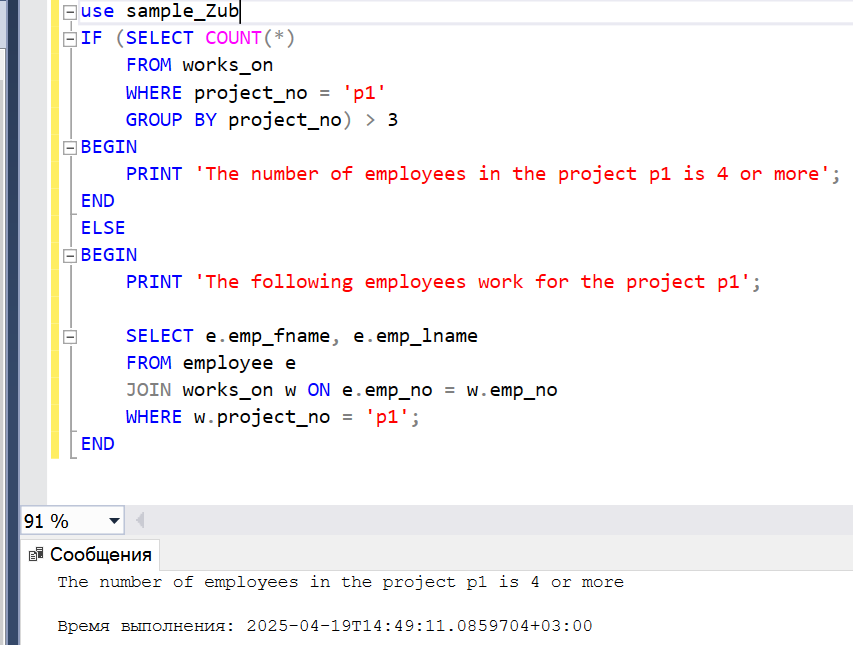


Рисунок 1 - Выполнение примера 1

**Пример 2. Использование инструкции WHILE**

use sample\_Zub

WHILE (SELECT SUM(budget) FROM project) < 500000

BEGIN

UPDATE project SET budget = budget \* 1.1;

IF (SELECT MAX(budget) FROM project) > 240000

BREAK;

ELSE

CONTINUE; -- Необязательная строка

END

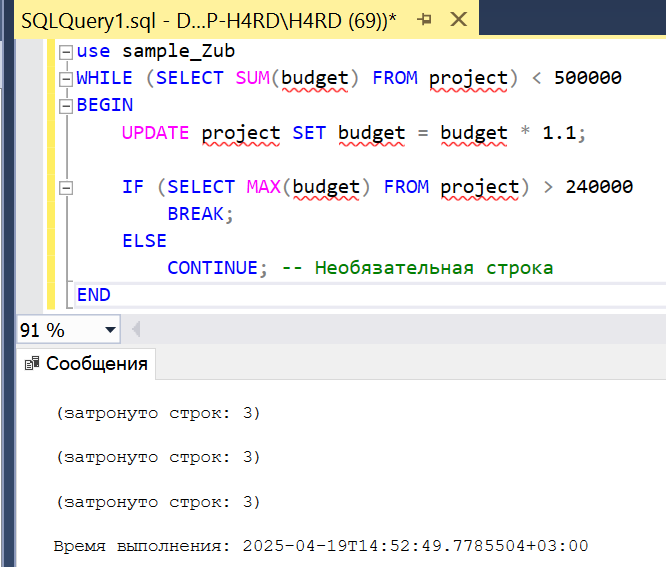


Рисунок 2 - Выполнение примера 2

**Пример 3. Присвоение значения локальным переменным**

use sample\_Zub

DECLARE @avg\_budget MONEY, @extra\_budget MONEY;

SET @extra\_budget = 15000;

SELECT @avg\_budget = AVG(budget)

FROM project;

IF (SELECT budget FROM project WHERE project\_no = 'p1') < @avg\_budget

BEGIN

UPDATE project

SET budget = budget + @extra\_budget

WHERE project\_no = 'p1';

PRINT 'Budget for p1 increased by ' + CAST(@extra\_budget AS VARCHAR(10));

END

ELSE

BEGIN

PRINT 'Budget for p1 unchanged';

END

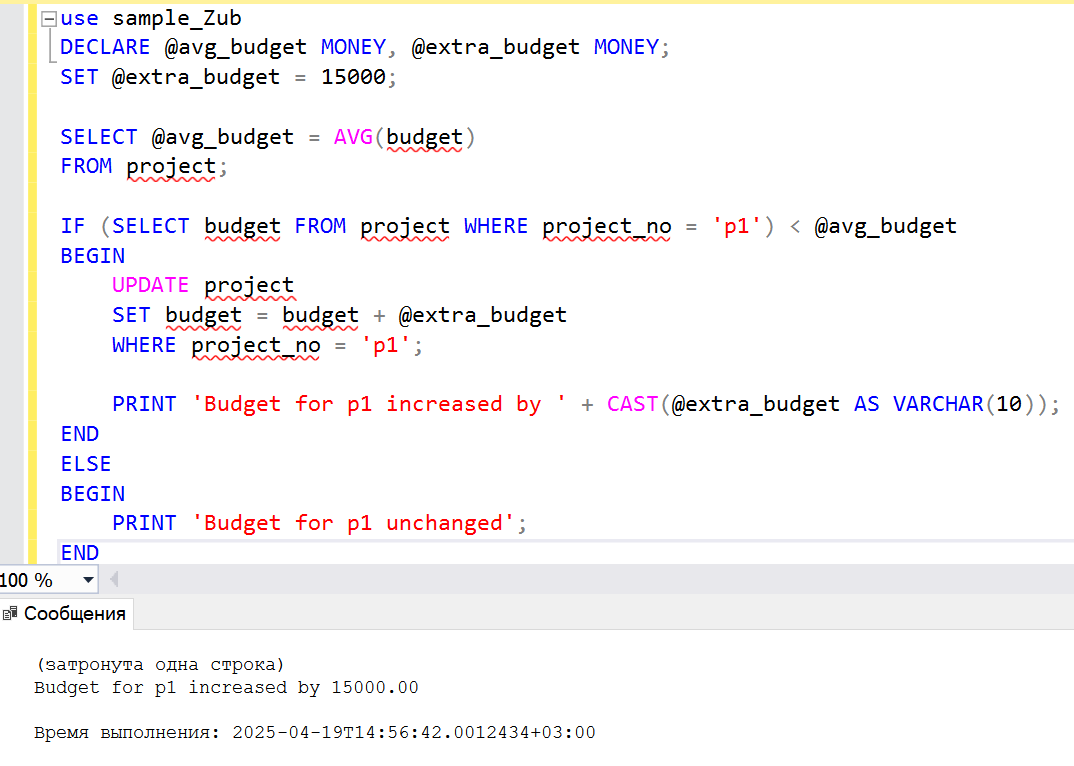


Рисунок 3 - Выполнение примера 3

Пример 4. **Добавление ограничений внешнего ключа.**

Используйте инструкцию ALTER TABLE employee для добавления ограничений внешнего ключа в базе данных sample.

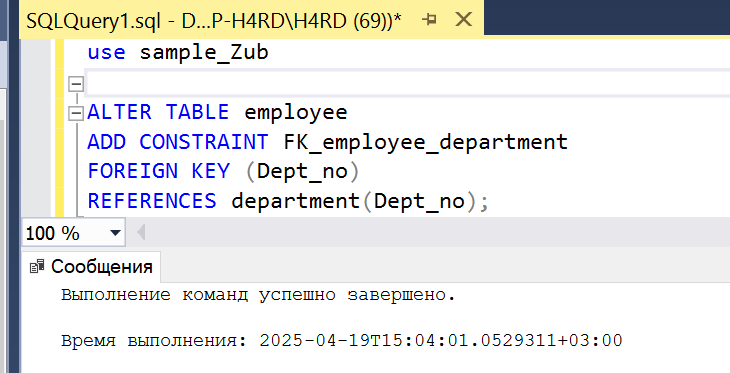


Рисунок 4 - Выполнение задания 4

**Пример 5 Применение инструкций TRY, CATCH и THROW для обработки исключений**

use sample\_Zub

BEGIN TRY

BEGIN TRANSACTION;

INSERT INTO employee VALUES (11111, 'Ann', 'Smith', 'd2');

INSERT INTO employee VALUES (22222, 'Matthew', 'Jones', 'd4');

INSERT INTO employee VALUES (33333, 'John', 'Barrimore', 'd2');

COMMIT TRANSACTION;

PRINT 'Transaction committed';

END TRY

BEGIN CATCH

IF @@TRANCOUNT > 0

ROLLBACK TRANSACTION;

PRINT 'Transaction rolled back';

THROW;

END CATCH

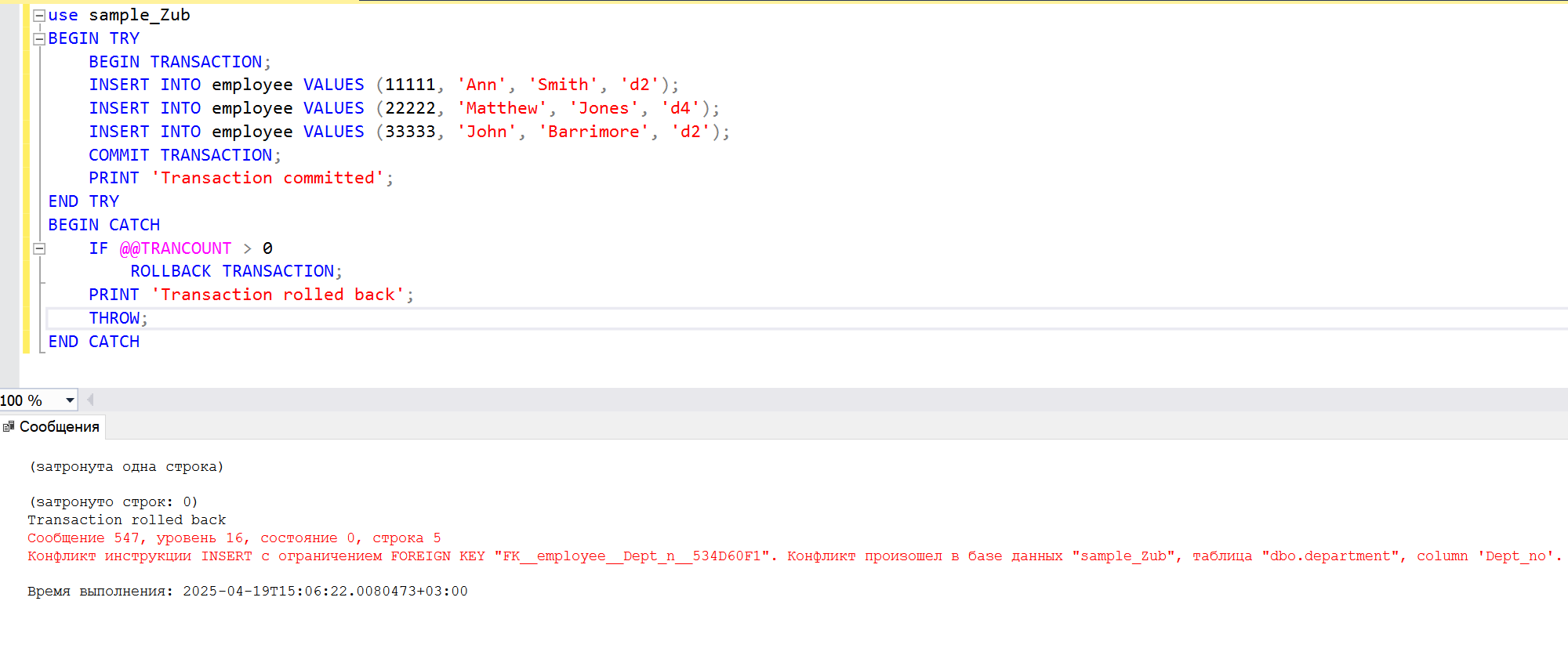


Рисунок 5 - Выполнение задания 5

**Пример 6. Создание хранимой процедуры для изменения данных таблицы**

use sample\_Zub

GO

CREATE PROCEDURE increase\_budget

@percent INT = 5

AS

BEGIN

SET NOCOUNT ON;

UPDATE project

SET budget = budget + (budget \* @percent / 100.0);

PRINT CONCAT('All project budgets increased by ', @percent, '%');

END;

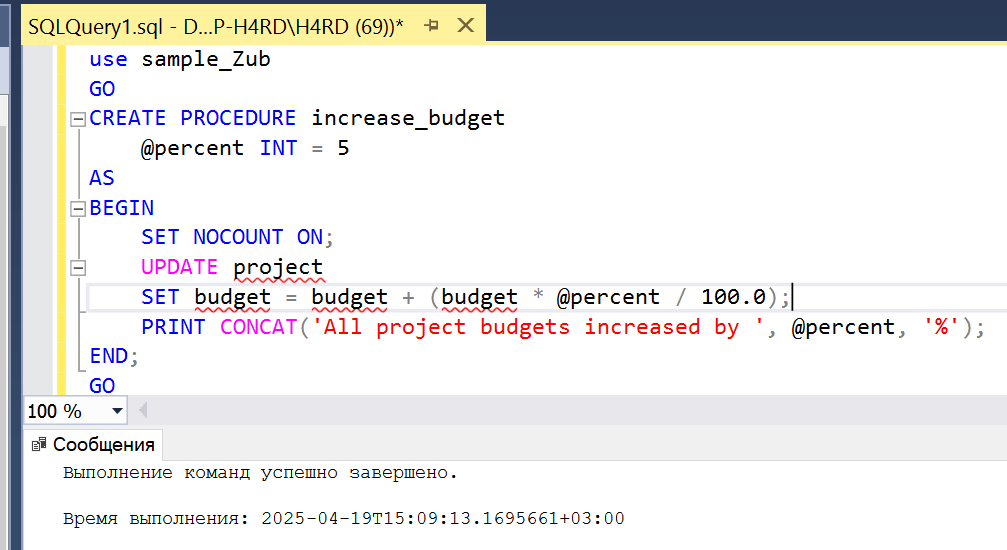
GO  


Рисунок 6 - Выполнение примера 6

**Пример 7. Применение инструкции EXECUTE** ;

USE sample\_Zub;

GO

EXEC increase\_budget 10;

Инструкция execute в примере 7 выполняет хранимую процедуру increase\_budget (см. Пример 6), которая увеличивает бюджете всех проектов на 10%.

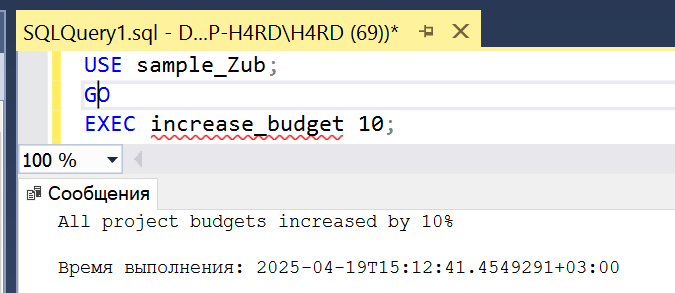


Рисунок 7 - Выполнение задания 7

**Пример 8. Создание хранимой процедуры для обработки таблиц EMPLOYEE и WORKS ON**

USE sample\_Zub;

GO

CREATE PROCEDURE modify\_empno

@old\_no INT,

@new\_no INT

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

BEGIN TRANSACTION;

IF NOT EXISTS (SELECT 1 FROM employee WHERE emp\_no = @old\_no)

BEGIN

RAISERROR('Сотрудник с номером %d не найден', 16, 1, @old\_no);

RETURN;

END

IF EXISTS (SELECT 1 FROM employee WHERE emp\_no = @new\_no)

BEGIN

RAISERROR('Номер %d уже занят другим сотрудником', 16, 1, @new\_no);

RETURN;

END

UPDATE employee

SET emp\_no = @new\_no

WHERE emp\_no = @old\_no;

UPDATE works\_on

SET emp\_no = @new\_no

WHERE emp\_no = @old\_no;

COMMIT TRANSACTION;

PRINT CONCAT('Номер сотрудника успешно изменен с ', @old\_no, ' на ', @new\_no);

END TRY

BEGIN CATCH

IF @@TRANCOUNT > 0

ROLLBACK TRANSACTION;

PRINT 'Ошибка при изменении номера сотрудника: ' + ERROR\_MESSAGE();

THROW;

END CATCH

END;

GO

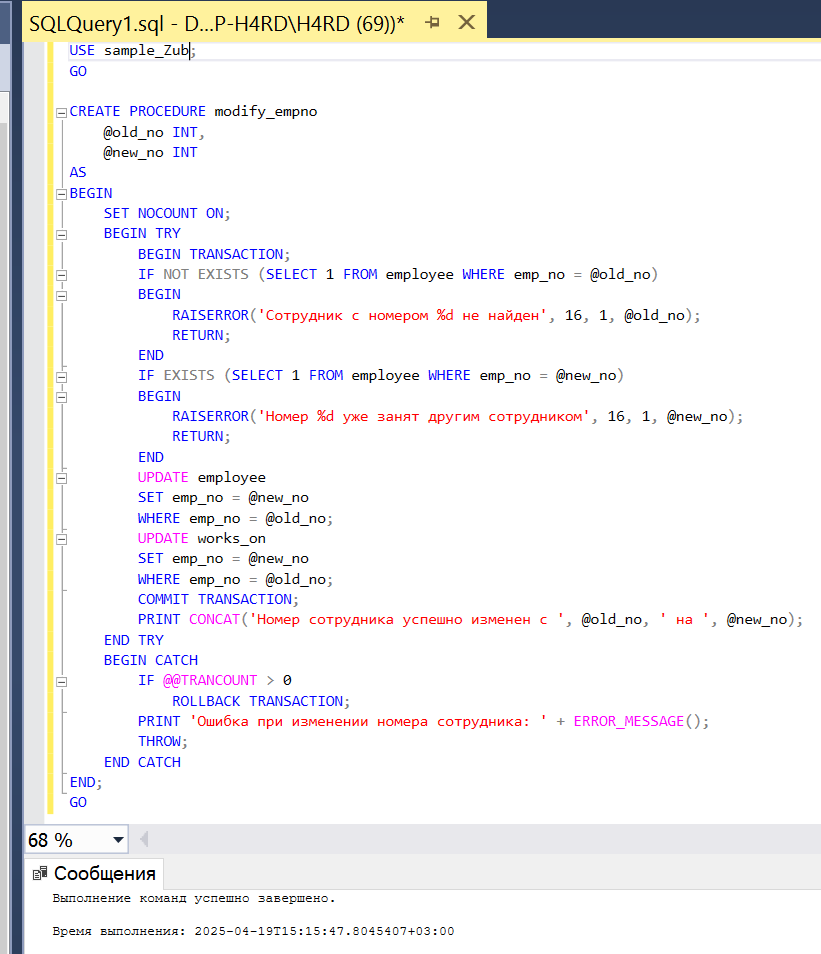


Рисунок 8 - Выполнение задания 8

**Пример 9. Использование в хранимой процедуре предложения OUTPUT**

USE sample;

GO

CREAT PROCEDURE delete\_emp @employee\_no INT, @counter INT OUTPUT AS

SELECT @counter = COUNT(\*)

FROМ works\_on

WHERE emp\_no = @employee\_no

DELETE FROM employee

WHERE emp\_no = @employee\_no

DELETE FROM works\_on WНERE emp\_no = @employee\_no

Данную хранимую процедуру можно запустить на выполнение посредством следующих инструкций:

DECLARE @quantity INT

EXECUTE delete еmp @employee\_no=28559, @counter=@quantity OUTPUT

**Пример 10. Применение инструкции EXECUTE с предложением WITH RESULT SETS**

USE sample;

EХEC employeesindept 'd1'

WITH RESULT SETS

(([EMPLOYEE NUMBER] INT NOT NULL,

[NAME OF EMPLOYEE] CHAR(20) NOT NULL));

**Пример 11. Создание определяемой пользователем функции**

* Эта функция вычисляет возникающие дополнительные обшие затраты,
* при увеличении бюджетов проектов

USE sampl;

GO

CREAT FUNCTION compute\_costs (@percent INT =10)

RETURNS DEСIMAL(16, 2)

BEGIN

DECLARE @additional\_costs DEC (14,2), @sum\_budget dec(16,2)

SЕLECT @sum\_budget = SUM (budget)

FROM project

SET @additional\_costs = @sum\_budget \* @pecent/100

REТURN @additional\_costs

END

**Пример 12. Вызов ОПФ в инструкции select**

USE sample;

SELECT project no, project name

FROM project

WНERE budget < dbo.cоmpute\_costs(25)

**Пример 13. Создание возвращающей табличное значение функции**

USE sample;

GO

CREATE FUNCTION employees\_in\_project (@pr\_number CHAR(4))

RETURNS TABLE AS

RETURN (SELECT emp\_fname, emp\_lname

FROM works\_on, employee

WHERE employee.emp\_no = works\_on.emp\_no AND project\_no = @pr\_number)

**Пример 14. Использование возвращающей табличное значение функции**

USE sample;

SELECT \*

FROM employees\_in\_project(’p3’)

**Пример 15. Создание возвращающей табличное значение функции**

CREATE FUNCТION dbo.fn\_getjob(@empid AS INT)

REТURNS TABLE AS

RETURN SELECT job

FROM works\_on

WHERE emp\_no = @empid AND job IS NОT NULL AND project\_no = ’p1’;

**Пример 16. "Соединение" двух таблиц посредством предложения APPLY**

— используется CROSS APPLY

SELECT E.emp\_no, emp\_fname, emp\_lname, job

FROM employee as E CROSS APPLY dbo.fn\_getjob(E.emp\_no) AS A

— используется OUTER APPLY

SELECT E.emp\_no, emp\_fname, emp\_lname, job

FROM employee as E OUTER APPLY dbo.fn\_getjob(E.emp\_no) AS A

Результатом выполнения этих двух функций будут следующие две таблицы.

**Пример 24. Применение возвращающего табличное значение параметра**

USE sample;

GО

CREATETYPE departmentType AS TABLE

(dept\_no CHAR(4),

dеpt\_name CHAR(25),

location CHAR(30));

GO

CREATE TABLE #dallasTable

(dept\_no CHAR(4),

dept\_nаme CHAR(25),

location CHAR(30));

GO

CREATE PROCEDURE insertРroc @Dallas departmentType READONLY AS

SET NOCOUNT ON

INSERТ INTO #dallasTable (dept\_no, dept\_name, location)

SELECT \* FROM @Dallas

GO

DECLARE @Dallas AS departmentType;

INSERT INTO @Dallas( dept\_no, dept\_name, location)

SELECT \* FROM department WHERE location = 'Dallas'

EХEC insertProc @Dallas;

**Упражнения**

**Упражнение 1**

Создайте пакет для вставки 3000 строк в таблицу employee. Значения столбца emp\_no должны быть однозначными в диапазоне от 1 до 3000. Всем ячейкам столбцов emp\_lname, emp\_fname и dept\_no присваиваются значения "Jane", "Smith" и "d1" соответственно.

**Упражнение 2**

Измените пакет из упражнения 1 таким образом, чтобы генерировать случайные значения для столбца emp\_no, используя функцию rand. (Подсказка: для получения случайных значений используйте системные функции datepart и GETDATE.)