**Лабораторная работа № 13. Разработка и использование функций**

**КУРНОСЕНКО СОФЬЯ**

**Функция** – это объект БД, представляющий собой поименованный код T-SQL. Для создания, удаления и изменения функций надо использовать операторы CREATE, DROP и ALTER соответственно. Отличие функций от хранимых процедур в ограничениях, накладываемых на код функции, в форме представления результата работы, а также в способе вызова. В функции не допускается применение DDL-операторов, DML-операторов, изменяющих БД (INSERT, DELETE, UPDATE), конструкций TRY/CATCH, а также использование транзакций.

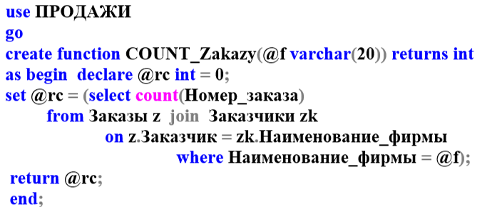
Результатом выполнения функции является возвращаемое к точке вызова значение. Если функция возвращает единственное значение (число, строка, дата, время и пр.), то она называется *скалярной*. Функция, возвращающая таблицу, называется *табличной*. В зависимости от структуры кода, различают *встроенные* функции и *многооператорные* табличные функции.

1. Разработать *скалярную* функцию с именем **COUNT\_STUDENTS**, которая вычисляет количество студентов на факультете, код которого задается параметром типа VARCHAR(20) с именем **@faculty**. Использовать внутреннее соединение таблиц FACULTY, GROUPS, STUDENT. Опробовать работу функции.

Внести изменения в текст функции с помощью оператора ALTER с тем, чтобы функция принимала второй параметр **@prof** типа VARCHAR(20),обозначающий специальность студентов. Для параметров определить значения по умолчанию NULL. Опробовать работу функции с помощью SELECT-запросов.

***→ Теория (методичка)***

Пусть требуется подсчитать количество товаров, заказанных фирмой:



При вызове функции надо указывать ее имя с точностью до схемы БД. Если при создании функции имя схемы не указано, то она размещается по умолчанию в схеме **DBO**.



Другой запрос:

**select Наименование\_фирмы,**

**dbo.COUNT\_Zakazy(Наименование\_фирмы)**

**from Заказчики;**

***→ Выполнение заданий***

Первая часть задания:

go

-- создание функции

create function COUNT\_STUDENTS

(@faculty varchar(20))

returns int

as

begin

declare @amountOfStudents int =

(select count(\*)

from FACULTY inner join GROUPS

on FACULTY.FACULTY = GROUPS.FACULTY

inner join STUDENT

on GROUPS.IDGROUP = STUDENT.IDGROUP

where FACULTY.FACULTY = @faculty);

return @amountOfStudents;

end;

-- выполнение

declare @fac varchar(20) = 'ХТиТ';

declare @amountOfFacStud int = dbo.COUNT\_STUDENTS(@fac);

print 'Кол-во студентов на кафедре ' + @fac + ': ' + cast(@amountOfFacStud as varchar);



Изменение функции COUNT\_STUDENTS: функция принимает два параметра с установленными значениями по умолчанию:

go

alter function dbo.COUNT\_STUDENTS

(@faculty varchar(20) = null,

@prof varchar(20) = null)

returns int

as

begin

declare @amountOfStudents int =

(select count(\*)

from FACULTY f inner join PROFESSION p on f.FACULTY=p.FACULTY

inner join GROUPS g on p.PROFESSION=g.PROFESSION

inner join STUDENT s on g.IDGROUP=s.IDGROUP

where f.FACULTY=isnull(@faculty, f.FACULTY)

and

p.PROFESSION=isnull(@prof, p.PROFESSION));

return @amountOfStudents;

end;

Вызов функции с передачей обоих параметров:

-- вызов функции при помощи select

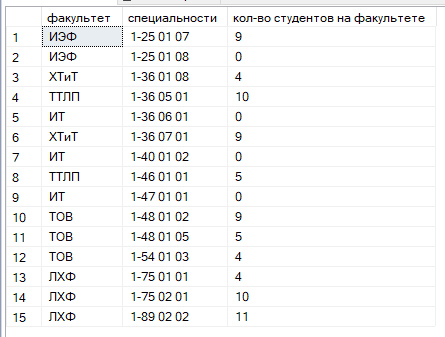
select distinct f.FACULTY as [факультет],

p.PROFESSION as [специальности],

dbo.COUNT\_STUDENTS(f.FACULTY, p.PROFESSION) as [кол-во студентов на факультете]

from FACULTY f inner join PROFESSION p on f.FACULTY=p.FACULTY

group by f.FACULTY, p.PROFESSION



Вызов функции, когда второй параметр оставлен со значением по умолчанию:

select distinct FACULTY as [факультет],

dbo.COUNT\_STUDENTS(FACULTY, default) as [кол-во студентов на факультете]

from FACULTY

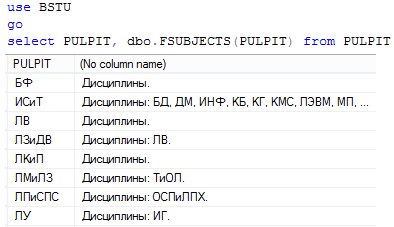


2. Разработать *скалярную* функцию с именем **FSUBJECTS,** принимающую параметр **@p** типа VARCHAR(20), значение которого задает код кафедры (столбец **SUBJECT.PULPIT**).

Функция должна возвращать строку типа VARCHAR(300) с перечнем дисциплин в отчете.

Создать и выполнить сценарий, который создает отчет, аналогичный представленному ниже.

Примечание: использовать локальный статический курсор на основе SELECT-запроса к таблице **SUBJECT**.



***→ Теория (методичка)***

Пусть требуется создать отчет о заказанных товарах тех фирм, информация о которых имеется в таблице **Заказчики**,с помощью скалярной функции **FZakazy**:

**create FUNCTION FZakazy(@tz char(20)) returns char(300)**

**as**

**begin**

**declare @tv char(20);**

**declare @t varchar(300) = 'Заказанные товары: ';**

**declare ZkTovar CURSOR LOCAL**

**for select Наименование\_товара from Заказы**

**where Заказчик = @tz;**

**open ZkTovar;**

**fetch ZkTovar into @tv;**

**while @@fetch\_status = 0**

**begin**

**set @t = @t + ', ' + rtrim(@tv);**

**FETCH ZkTovar into @tv;**

**end;**

**return @t;**

**end;**

Обращение к функции:

**select Наименование\_фирмы, dbo.FZakazy**

**(Наименование\_фирмы) from Заказчики;**

***→ Выполнение заданий***

go

create function FSUBJECTS (@p varchar(20)) returns varchar(300)

as

begin

declare @subject varchar(10),

@allSubjects varchar(300) = 'Дисциплины: ';

declare subjectsOnPulpitCur cursor local static

for

select SUBJECT

from SUBJECT

where PULPIT=@p;

open subjectsOnPulpitCur;

fetch subjectsOnPulpitCur into @subject;

while @@fetch\_status = 0

begin

set @allSubjects=@allSubjects + ', ' + rtrim(@subject);

fetch subjectsOnPulpitCur into @subject;

end;

close subjectsOnPulpitCur;

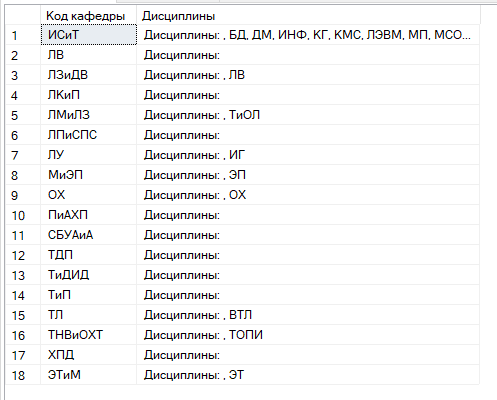
return @allSubjects;

end;

select PULPIT as [Код кафедры],

dbo.FSUBJECTS(PULPIT) as [Дисциплины]

from PULPIT



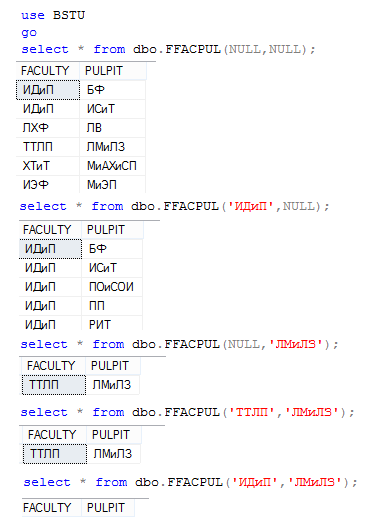
3. Разработать *табличную* функцию **FFACPUL**, результаты работы которой продемонстрированы на рисунке ниже.

Функция принимает два параметра, задающих код факультета (столбец **FACULTY.FACULTY**) и код кафедры (столбец **PULPIT.PULPIT**). Использует SELECT-запрос c левым внешним соединением между таблицами **FACULTY** и **PULPIT**.

Если оба параметра функции равны NULL, то она возвращает список всех кафедр на всех факультетах.

Если задан первый параметр (второй равен NULL), функция возвращает список всех кафедр заданного факультета.

Если задан второй параметр (первый равен NULL), функция возвращает результирующий набор, содержащий строку, соответствующую заданной кафедре.

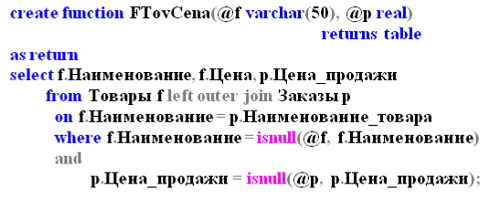
****

Если заданы два параметра, функция возвращает результирующий набор, содержащий строку, соответствующую заданной кафедре на заданном факультете.

Если по заданным значениям параметров невозможно сформировать строки, функция возвращает пустой результирующий набор.

***→ Теория (методичка)***

Встроенная *табличная* функция **FTovCena** выводит инфор-мацию об исходных ценах и ценах продажи, используя таблицы **Товары** и **Заказы**:

****

Различные варианты обращения к функции позволяют проанализировать ее работу:

**select \* from dbo.FTovCena(NULL, NULL);**

**select \* from dbo.FTovCena('Стул', NULL);**

**select \* from dbo.FTovCena(NULL, 400);**

**select \* from dbo.FTovCena('Шкаф', 340);**

***→ Выполнение заданий***

go

create function FFACPUL (@fac varchar(20), @pul varchar(20)) returns table

as return

select PULPIT as [Кафедра],

FACULTY as [Факультет]

from PULPIT

where PULPIT = isnull(@pul, PULPIT)

and

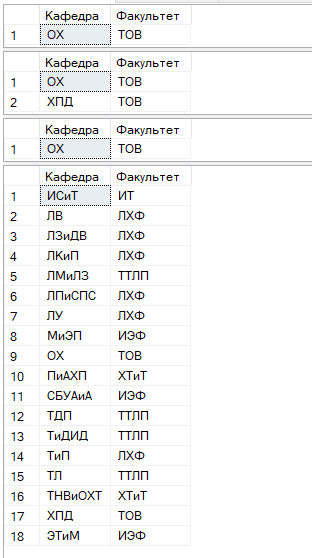
FACULTY = isnull(@fac, FACULTY);

select \* from dbo.FFACPUL('ТОВ', 'ОХ')

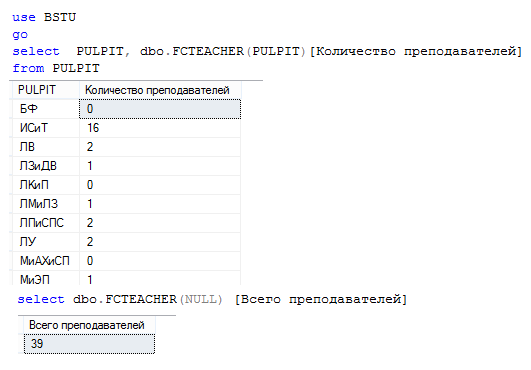
select \* from dbo.FFACPUL('ТОВ', null)

select \* from dbo.FFACPUL(null, 'ОХ')

select \* from dbo.FFACPUL(null, null)



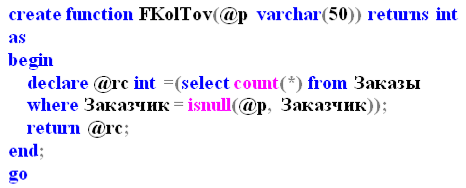
4. На рисунке ниже показан сценарий, демонстрирующий работу *скалярной* функции **FCTEACHER**. Функция принимает один параметр, задающий код кафедры. Функция возвращает количество преподавателей на заданной параметром кафедре. Если параметр равен NULL, то возвращается общее количество преподавателей.

****

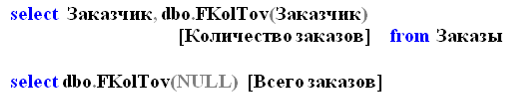
Разработать функцию **FCTEACHER**.

***→ Теория (методичка)***

Скалярная функция **FKolTov** выводит информацию о товарах, заказанных конкретным заказчиком, используя табли-цу **Заказы**. Если передается в функцию параметр, равный NULL, то выводится сообщение об общем количестве заказов:

****

Обращения к функции:

****

***→ Выполнение заданий***

go

create function FCTEACHER (@pul varchar(20)) returns int

as

begin

return

(select count(\*)

from TEACHER

where PULPIT=isnull(@pul, PULPIT))

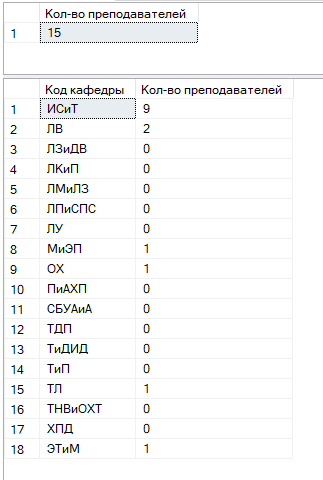
end;

select dbo.FCTEACHER(null) as [Кол-во преподавателей];

select PULPIT as [Код кафедры],

dbo.FCTEACHER(PULPIT) as [Кол-во преподавателей]

from PULPIT;



6. Проанализировать *многооператорную табличную* функцию **FACULTY\_REPORT**, представленную ниже:

**create function FACULTY\_REPORT(@c int) returns @fr table**

**( [Факультет] varchar(50), [Количество кафедр] int, [Количество групп] int,**

**[Количество студентов] int, [Количество специальностей] int )**

**as begin**

**declare cc CURSOR static for**

**select FACULTY from FACULTY**

**where dbo.COUNT\_STUDENTS(FACULTY, default) > @c;**

**declare @f varchar(30);**

**open cc;**

**fetch cc into @f;**

**while @@fetch\_status = 0**

**begin**

**insert @fr values( @f, (select count(PULPIT) from PULPIT where FACULTY = @f),**

**(select count(IDGROUP) from GROUPS where FACULTY = @f), dbo.COUNT\_STUDENTS(@f, default),**

**(select count(PROFESSION) from PROFESSION where FACULTY = @f) );**

**fetch cc into @f;**

**end;**

**return;**

**end;**

Изменить эту функцию так, чтобы количество кафедр, количество групп, количество студентов и количество специальностей вычислялось отдельными скалярными функциями.

***→ Выполнение заданий***

Пояснение функции:

create function FACULTY\_REPORT(@c int) returns @fr table

( [Факультет] varchar(50), [Количество кафедр] int, [Количество групп] int,

[Количество студентов] int, [Количество специальностей] int )

as begin

declare cc CURSOR static for

-- выбираем из таблицы FACULTY все факультеты, на к-рых кол-во студентов больше параметра @c данной функции

select FACULTY from FACULTY

where dbo.COUNT\_STUDENTS(FACULTY, default) > @c;

declare @f varchar(30);

open cc;

fetch cc into @f;

while @@fetch\_status = 0

begin

-- вставляем записи в таблицу, к-рую будем возвращать

insert @fr values(

@f, -- факультет, где студентов больше @c

(select count(PULPIT) from PULPIT where FACULTY = @f), -- кол-во кафедр на факультете @f

(select count(IDGROUP) from GROUPS where FACULTY = @f), -- кол-во кафедр на групп @f

dbo.COUNT\_STUDENTS(@f, default), -- кол-во студентов на факультете @f

(select count(PROFESSION) from PROFESSION where FACULTY = @f) -- кол-во специальностей на факультете @f

);

fetch cc into @f;

end;

return;

end;

Перенесем логику в скалярные функции:

-- скалярная функция для подсчета кол-ва кафедр на факультете @fac

go

create function countPulpits(@fac varchar(20)) returns int

as

begin

return (select count(PULPIT) from PULPIT where FACULTY = @fac)

end;

-- скалярная функция для подсчета кол-ва групп на факультете @fac

go

create function countGroups(@fac varchar(20)) returns int

as

begin

return (select count(IDGROUP) from GROUPS where FACULTY = @fac)

end;

-- скалярная функция для подсчета кол-ва студентов на факультете @fac

go

create function countStudents(@fac varchar(20)) returns int

as

begin

return (

select count(IDSTUDENT)

from STUDENT inner join GROUPS

on STUDENT.IDGROUP=GROUPS.IDGROUP

where FACULTY = @fac

)

end;

-- скалярная функция для подсчета кол-ва специальностей на факультете @fac

go

create function countProf(@fac varchar(20)) returns int

as

begin

return (select count(PROFESSION) from PROFESSION where FACULTY = @fac)

end;

-- вызываем вышесозданные функции в данной

go

create function FACULTY\_REPORT(@c int) returns @fr table

( [Факультет] varchar(50), [Количество кафедр] int, [Количество групп] int,

[Количество студентов] int, [Количество специальностей] int )

as begin

declare cc CURSOR static for

-- выбираем из таблицы FACULTY все факультеты, на к-рых кол-во студентов больше параметра @c данной функции

select FACULTY from FACULTY

where dbo.countStudents(FACULTY) > @c;

declare @f varchar(30);

open cc;

fetch cc into @f;

while @@fetch\_status = 0

begin

-- вставляем записи в таблицу, к-рую будем возвращать

insert @fr values(

@f, -- факультет, где студентов больше @c

dbo.countPulpits(@f), -- кол-во кафедр на факультете @f

dbo.countGroups(@f), -- кол-во групп на факультете @f

dbo.countStudents(@f), -- кол-во студентов на факультете @f

dbo.countProf(@f) -- кол-во специальностей на факультете @f

);

fetch cc into @f;

end;

return;

end;

select \* from dbo.FACULTY\_REPORT(0);

