# ლაბორატორია 11 ფუნქციები

## SQL Server - ფუნქციები

SQL Server-ის ფუნქციები სხვა პროგრამირების ენების ფუნქციების მსგავსია. SQL Server-ის ფუნქციები შეიცავს SQL ბრძანებებს რომლებიც ასრულებენ გარკვეულ დავალებებს. ფუნქციებს შეიძლება ჰქონდეს შემავალი პარამეტრები და უნდა დააბრუნოს ერთი მნიშვნელობა ან რამდენიმე ჩანაწერი მოთხოვნის სახით.

თუ თქვენი მრავალჯერ იყენებთ SQL ბრძნაბების ერთსა და იმავე ნაკრებს, კონკრეტული დავალებების შესასრულებლად, მაშინ შეიძლება მათი ფუნქციად გამოცხადება.

## ფუნქციების სახეები

SQL სერვერზე გვხვდება ფუნქციების ორი ტიპი:

* **სისტემური ფუნქციები (System Functoins) :** ეს არის ჩაშენებული ფუნქციები, რომლებიც ხელმისაწვდომია ყველა მონაცემთა ბაზაში.
* **მომხმარებლის მიერ განსაზღვრული ფუნქციები (UserDefinedFunctions):** ფუნქციები, რომლებიც შექმნილია მონაცემთა ბაზის მომხმარებლის მიერ კონკრეტულ მონაცემთა ბაზაში. ასეთი ფუნქციები თავისმხრივ ორი ტიპისაა:
  + **სკალარული ფუნქციები** (Scalar-Valued Functions): ფუნქციები რომლებიც აბრუნებენ მონაცემთა ნებისმიერი ტიპის მხოლოდ ერთ მნიშვნელობას (სკალარს).
  + **ცხრილის დონის ფუნქციები** (Table-Valued Functions): ფუნქციები, რომლებიც აბრუნებენ მრავალ ჩანაწერს, შეკრულს ცხრილის სახით. დასაბრუნებელი შედეგი შეიძლება იყოს Select მოთხოვნის ნებისმიერი ნაკრები.

ქვემოთ მოცემულია მომხმარებლის მიერ განსაზღვრული ფუნქციის გამარტივებული სინტაქსი SQL Server-ში.

სინტაქსი: მომხმარებლის მიერ განსაზღვრული ფუნქცია

CREATE OR ALTER FUNCTION [schema\_name.]function\_name(@parameter\_name parameter\_data\_type,...)

RETURNS <data\_type>

[WITH <function\_options>]

AS

BEGIN

<function\_body>

RETURN <value or select\_statement>

END

## სკალარული ფუნქციები

სკალარული ფუნქცია ყოველთვის აბრუნებს ერთ მნიშვნელობას. შემდეგი სკალარული ფუნქცია GetLecturerCount აბრუნებს საგანზე დამატებული ლექცორების რაოდენობას საგნის ნომრის მიხედვით:

CREATE FUNCTION GetLecturerCount(@SubjectId INT)

RETURNS INT

AS

BEGIN

DECLARE @LecturerCount INT = 0; --აღვწერთ მთელი ტიპის ცვლადს შედგის შესანახად.

--მოთხოვნით ვპოულობთ ლექცორების საგანზე მიბმული ლექტორების რაოდენობას კავშირის ცხრილში

SELECT @LecturerCount = COUNT ([LecturerId]) FROM [SubjectLecturers]

WHERE @SubjectId = @SubjectId

RETURN @LecturerCount; --ვაბრუნებთ პასუხს

END

განვიხილოთ მაგალითი:

* CREATE FUNCTION ის ნაცვლად შეიძლება ყოფილიყო CREATE or ALTER FUNCTION რაც მიუთითებს ახალი ფუნქციის შექმნაზე ან ფუნქციის არსებობის შემთხვევაში შეცვლაზე ასევე შესაძლებელია ALTER FUNCTION ოპერატორის გამოყენება თუ გვსურს ფუნქციის ცვლილება. გასათვალისწინებელია რომ პროცედურის მსგავსად ცვლილების შემთხვევაში ფუნქციაც იწერება სრულად.
* GetLecturerCount არის ფუნქციის სახელი. ფუნქციას მომხმარებელი ურჩევს სახელს.
* @SubjectId არის INT ტიპის შემავალი პარამეტრი. საჭიროებისამებრ აქ მრგვალ ფრჩხილებში შეიძლება იყოს პაამეტრების ჩამონათვალი.
* RETURNS INT გვეუბენბა რომ ფუნქცია დააბრუნებს ერთ მნიშვნელობას(სკალარს) და იგი იქნება INT ტიპის.
* AS ოპერატორი ასრულებს გამყოფ მოვალეობას ფუნქციის თავსა და ტანს შორის
* თავად ფუნქციის ტანი მოთავსებულია BEGIN და END ოპერატორებს შორის.
* RETURN ოპერატორი აბრუნებს @LecturerCount ცვლადში შენახულ მნიშვნელობას.
* ფუნქციის ცვლილება ხდება ALTER ბრძანებით, ხოლო წაშლა DROP-ით.

სკალარული ფუნქციის გამოძახება შესაძლებელია [SELECT](https://www.tutorialsteacher.com/sqlserver/select-query) მოთხოვნაში როგორც ეს ნაჩვენებია ქვემოთ:

SELECT dbo.GetLecturerCount(1)

თუმცა გარდა აღნიშნული მეთოდისა ფუნქციის გამოძახება და გამოყენება შესაძლებელია ნებისმერი მოქმედების შიგნით, სადაც შესაძლებელია მისი დასაბრუნებელი ტიპის მონაცემის გამოყენება. მარტივად რომ ვთქვათ ფუნქცია რომელიც აბრუნებს მთელ რიცხვს შეიძლება გამოვიყენოთ არითმეტიკულ ოპერაციებში არგუმენტად, ხოლო ფუნქცია რომელიც აბრუნებს სტრიქონს შეიძლება გამოვიძახოთ და გამოვიყენოთ სტრიქონულ ოპერაციებში. ფუნქციის გამოძახება შესაძლებელია პროცედურების შიგნითაც კი.

ვინაიდან პროცედურები ვახსენეთ სავარაუდოდ გაგვიჩნდებოდა კითხვა: “რა სხვაობაა შენახულ პროცედურებსა და ფუნქციებს შორის?“ ან „რა შემთხვევაშია საჭირო შენახული პროცედურა და რა შემთხვევაში ფუნქცია?“ აღნიშნულ კითვებზე ამომწურავი პასუხის გასაცემად დეტალურად განვიხილოთ სხვაობა ბაზის ამ ორ ობიექტებს შორის:

|  |  |
| --- | --- |
| [**User Defined function**](http://sqlhints.com/2011/08/14/user-defined-functions/) | [**Stored Procedure**](http://sqlhints.com/2011/08/14/stored-procedure/) |
| ფუნქცია აუცილებლად აბრუნებს რაღაც მნიშვნელობას | პროცედურით შესაძლებელია არ მოხდეს კონკრეტული მნიშვნელობის დაბრუნება |
| ფუნქცია იყენებს Select წინადადებას, მასში არ ხდება DML ბრძანებების გამოყენება | პროცედურა ერთნაირად იყენებს როგორც select, ასევე DML-ის ბრძანებებს, როგორიცაა insert, update, delete |
| იყენებს მხოლოდ input პარამეტრს, არ აქვს მხარდაჭერა output პარამტრთან. | აქვს ორივე input და output პარამტრები |
| არ იძლევა try-catch ბლოკების გამოყენების საშუალებას. | For exception handling we can use try catch blocks. |
| ტრანზაქციები არ არის დაშვებული ფუნქციის გამოყენების ფარგლებში | დაიშვება ტრანზაქციების გამოყენება |
| გამოიყენება მხოლოდ ცხრილური ტიპის ცვლადები, დროებითი ცხრილების გამოყენება არ ხდება. | გამოიყენება ორივე: ცხრილური ტიპის ცვლადებიც და დროებითი ცხრილებიც. |
| ფუნქციიდან არ ხდება პროცედურის გამოძახება | პროცედურით შესაძლებელია ფუნქციის გამოძახება |
| ფუნქცია შესაძლებელია გამოძახებული იქნას select წინადადებით | პროცედურა შესაძლებელია გამოძახებულ იქნას Select/Where/Having წინადადებებით. მის გამოსაძახებლად და გასაშვებად გამოიყენება Execute/Exec |
| შესაძლებელია გამოყენებული იქნეს შეერთების ბრძანებაში, როგორც შედეგი | პროცედურის გამოყენება შეერთების ბრძანებაში არ ხდება |

## ლაბორატორიული მეცადინეობის სამუშაო ბაზა

CREATE TABLE [product]

(

[pr\_id] INT PRIMARY KEY,

[pr\_name] NVARCHAR(20),

[price] INT

);

CREATE TABLE [shop]

(

[shop\_id] INT PRIMARY KEY,

[shop\_name] NVARCHAR(15)

);

CREATE TABLE [pr\_shop]

(

[pr\_is] INT

FOREIGN KEY REFERENCES [product] ([pr\_id]) ON DELETE CASCADE ON UPDATE CASCADE,

[shop\_id] INT

FOREIGN KEY REFERENCES [shop] ([shop\_id]) ON DELETE CASCADE ON UPDATE CASCADE,

[pr\_num] INT

);

INSERT INTO [product]

VALUES

(1, 'printer', 230),

(2, 'FAX', 450),

(3, 'USB\_8', 23),

(4, 'USB\_16', 30),

(5, 'xerox', 520),

(6, 'UPS', 120);

INSERT INTO [shop]

VALUES

(101, 'shop1'),

(102, 'shop2'),

(103, 'shop3'),

(104, 'shop4'),

(105, 'shop5'),

(106, 'shop6'),

(107, 'shop7');

INSERT INTO [pr\_shop]

VALUES

(1, 101, 5),

(1, 102, 6),

(2, 101, 5),

(1, 103, 7),

(3, 101, 4),

(2, 103, 5),

(1, 104, 5),

(3, 104, 10),

(4, 107, 12),

(1, 107, 5);

## მაგალითები:

--შევქმნათ სკალარული ფუნქცია:რომელიც დააბრუნებს კონკრეტული საქონლის მთლიან თანხას

create function GetSummpr

(@pr\_name varchar(20))

returns int

begin

declare @summ int

select @summ=price\*pr\_num

from product inner join pr\_shop on product.pr\_id=[pr\_is]

inner join shop on pr\_shop.shop\_id=shop.shop\_id

where pr\_name=@pr\_name

return @summ

end

select dbo.GetSummpr ('printer')as summ

--ფუნქცია მივანიჭოთ ცვლადს

declare @summpr int

set @summpr=dbo.GetSummpr ('printer')

print N'პროდუქციის საერთო ღირებულება არის ' +str(@summpr)

-- დავითვალოთ მაღაზიებში გაცემული საქონლის რაოდენობა

create function prnum

(@pr\_id int)

returns int

as

begin

declare @prnum int

select @prnum=sum(pr\_num) from pr\_shop

where pr\_is=@pr\_id

return @prnum

end

select dbo.prnum (3)

--შევქმნათ უპარამეტრო ფუნქცია, რომელიც აბრუნებს ცხრილს:

დავითვალოთ გაცემული პროდუქტის საერთო ღირებულება

create function getprice ()

returns table

as

return

select pr\_name, price\*pr\_num as totalcost

from product inner join pr\_shop on product.pr\_id=[pr\_is]

inner join shop on pr\_shop.shop\_id=shop.shop\_id

select \* from getprice()

--დავამატოთ პროდუქციის ცხრილს ველი და შევავსოთ პროცედურით:

alter table product

add

total int

select \* from product

create procedure inserttotal

@pr\_id int,

@total int

as

begin

update product

set

total =@total

where pr\_id=@pr\_id

end

exec inserttotal 3,40

select \* from product

-------------------------------------------

select pr\_name, total-pr\_num as darchenili

from product inner join pr\_shop on product.pr\_id=[pr\_is]

inner join shop on pr\_shop.shop\_id=shop.shop\_id

--შევქმნათ ფუნქცია, რომელიც დააბრუნებს კონკრეტული საქონლის საწყობში დარჩენილ რაოდენობას

create function getnumber ()

returns table

as

return

select pr\_name, total-pr\_num as darchenili

from product inner join pr\_shop on product.pr\_id=[pr\_is]

inner join shop on pr\_shop.shop\_id=shop.shop\_id

select \* from getnumber()

--შევქმნათ ფუნქცია, რომელიც დააბრუნებს კონკრეტული საქონლის საწყობში დარჩენილ რაოდენობას და სახელს.

create function getnumber1 ()

returns table

as

return

select pr\_name, total-pr\_num as darchenili, pr\_num

from product inner join pr\_shop on product.pr\_id=[pr\_is]

inner join shop on pr\_shop.shop\_id=shop.shop\_id

select \* from getnumber1()