How to Use Cross Apply Instead of Cursors in SQL Server

Please [Sign up or sign in](https://www.codeproject.com/script/Membership/LogOn.aspx?rp=%2fTips%2f654894%2fHow-to-Use-Cross-Apply-Instead-of-Cursors-in-SQL-S) to vote.

How to use cross apply along with a table valued function as an alternative to cursors

Introduction

I recently ran into a stored procedure with cursor logic, I wanted to see what would be the performance gain if cursor logic is rewritten with set theory operations.

Here is a simplified description of the stored procedure. There is a table with five columns, one column is an identity and a primary, (let's call it the ID), three columns (x, y, z) are of type integers and the fifth column (a) is a computed column. The computation is quite complex so it cannot be declared as a computed column expression. For each row, the values of the three columns (x, y, z) are passed in as parameters to a custom function (where the logic is encapsulated) which spits out a calculated value. Finally, for every row, the column "a" is updated with the calculated value from the custom function. There are around 7000 rows in this table.

Using the Code

Let's create a table called test1 with five columns. For this example, let's stick with a simple logic, the fifth column is a sum of cols x,y,z.

**Step 1**: Create the test table:

Hide   Copy Code

create table test1

(

id int not null identity(1,1),

x int,

y int,

z int,

a int null

)

**Step 2**: Insert dummy data:

Hide   Copy Code

insert into test1 values (5,5,5, NULL)

go 10

*-- (I inserted 9132 rows, took five minutes for the code to execute)*

**Step 3**: Using cursors to update the column "a":

Hide   Copy Code

declare @x1 int

declare @x2 int

declare @x3 int

declare @x4 int

declare @x5 int

declare c1 cursor local for

select id, x, y, z, a from test1

open c1

while (0=0)

begin

fetch next from c1 into

@x1, @x2,@x3,@x4,@x5

if (@@FETCH\_STATUS = -1)

break

*-- your logic*

set @x5 = @x4 + @x2 + @x3

update test1

set test1.a = @x5

from test1

where id = @x1

end

close c1

deallocate c1

*-- Execution time 01:09*

**Step 4**: Reset column "a":

Hide   Copy Code

update test1 set a = null

**Step 5**: Create a table valued function (tvf) shown below:

Hide   Copy Code

create function dbo.fnsomelogic (@x int ,@y int, @z int)

returns @val table

(

q int

)

as

begin

declare @q int

set @q = @x + @y +@z

insert into @val (q) values (@q)

return

end

*-- tvf can be invoked as shown below*

*-- select \* from dbo.fnsomelogic(1,2,3)*

**Step 6**: Use cross apply and the tvf to update column a:

Hide   Copy Code

update test1

set test1.a = c.q

from test1 b cross apply dbo.fnsomelogic(b.x,b.y,b.z) c

*--Execution Time: (9132 row(s) affected) in less than a second.*

Points of Interest

If you observe the messages, the cursor which is a row based operator, displays (1 row(s) affected) for every row it updated, unlike the cross apply which displays (9132 row(s) affected). Although the problem is screaming out use cursors, however with little observation a cross apply along with a table valued function can boost the performance significantly. Relational/Set theory concepts are deeply embedded within SQL Server.