Class 21 Intermediate Stored Procedures 2

# Using Table Valued Functions

## STRING\_SPLIT

To this point we've been using functions independently – pass value A, get value B. But it is more common that we will want to use the results of our functions in conjunction with other data. To see how this works, we're going to use a system function called STRING\_SPLIT.

<https://docs.microsoft.com/en-us/sql/t-sql/functions/string-split-transact-sql?view=sql-server-ver15>

This function can take delimited strings as input and split results into separate rows. Let's review in a simple example:

USE TestDB;

GO

DECLARE @singleValue NCHAR(7) = 'A,B,C,D'

SELECT \*

FROM STRING\_SPLIT(@singleValue,',');

Note the output returns several rows in a single column called 'value'.

This simple example helps illustrate the use of STRING\_SPLIT, but we won't usually want to apply this to a single value. We want to apply this function to every row in a table. However, unlike scalar functions, this table valued function might return multiple output rows for each input row.

Consider the following example. We want to split a delimited list of classes for each student, creating a new row for each student-class.

-- NOTE: This is just a different way of declaring a table variable

DECLARE @studentClasses TABLE ( FullName NVARCHAR(100), Courses NVARCHAR(250) );

INSERT INTO @studentClasses

VALUES ( 'Finn Mertens', 'Advanced Databases,Client Server,Data Structures' ),

( 'Simon Petrikov', 'Client Server,Advanced Databases,Object Oriented Programming,Data Structures,Networking' ),

( 'Marceline Abadeer', 'Advanced Databases' ),

( 'Minerva Cambell', 'Communications,Employment Prep' ),

( 'Warren Ampersand', '' );

-- See what is in the table:

Select \* from @studentClasses;

GO

-- Why does this fail?

SELECT sc.FullName, sc.Courses, STRING\_SPLIT(sc.Courses, ',')

FROM @studentClasses sc

Why does our code fail? Why can't we use STRING\_SPLIT in this way?

If we tried again, selecting from the TVF like we did in other examples, we would run into an issue with our join condition:

SELECT sc.FullName, sc.Courses, splt.value

FROM @studentClasses sc

INNER JOIN STRING\_SPLIT(sc.Courses, ',') splt

ON -- What would I join on here?

The only return from this function is value, which doesn't match any of the columns I have available in the original data.

So how do we join this data?

## CROSS APPLY

What we are really looking to do here is join each record from STRING\_SPLIT with the value that was used to create it. But without that value in the output, we can't use a regular join to accomplish this.

For cases like this, we use CROSS APPLY.

<https://docs.microsoft.com/en-us/sql/t-sql/queries/from-transact-sql?view=sql-server-ver15>

Let's rewrite our join using CROSS APPLY instead. Note that there is no join condition required – each record from the output is joined to the record that created it.

DECLARE @studentClasses TABLE ( FullName NVARCHAR(100), Courses NVARCHAR(250) );

INSERT INTO @studentClasses

VALUES ( 'Finn Mertens', 'Advanced Databases,Client Server,Data Structures' ),

( 'Simon Petrikov', 'Client Server,Advanced Databases,Object Oriented Programming,Data Structures,Networking' ),

( 'Marceline Abadeer', 'Advanced Databases' ),

( 'Minerva Cambell', 'Communications,Employment Prep' ),

( 'Warren Ampersand', '' );

SELECT val.FullName,

val.Courses AS OriginalCourseList,

splt.value

FROM @studentClasses val

CROSS APPLY STRING\_SPLIT(val.Courses,',') splt;

Does this return what you expected?

Notice that CROSS APPLY can be used in conjunction with other kinds of joins. Perhaps we wanted to separate this delimited list so we could return a list of the professors that these students had this semester. Splitting this list and using the value column returned by STRING\_SPLIT might allow us to do that:

DECLARE @studentClasses TABLE ( FullName NVARCHAR(100), Courses NVARCHAR(250) );

INSERT INTO @studentClasses

VALUES ( 'Finn Mertens', 'Advanced Databases,Client Server,Data Structures' ),

( 'Simon Petrikov', 'Client Server,Advanced Databases,Object Oriented Programming,Data Structures,Networking' ),

( 'Marceline Abadeer', 'Advanced Databases' ),

( 'Minerva Cambell', 'Communications,Employment Prep' ),

( 'Warren Ampersand', '' );

DECLARE @professorClasses TABLE ( FullName NVARCHAR(100), Course NVARCHAR(100) );

INSERT INTO @professorClasses

VALUES ( 'Eleanor Shellstrop', 'Advanced Databases' ),

( 'Chidi Anagonye', 'Client Server' ),

( 'Jason Mendoza', 'Data Structures' ),

( 'Tahani Al-Jamil', 'Object Oriented Programming' ),

( 'Michael', 'Networking' ),

( 'Good Janet', 'Communications' ),

( 'Bad Janet', 'Employment Prep' );

SELECT val.FullName,

val.Courses AS OriginalCourseList,

splt.value AS CourseName,

prof.FullName AS ProfessorName

FROM @studentClasses val

CROSS APPLY STRING\_SPLIT(val.Courses,',') splt

JOIN @professorClasses prof

ON splt.value = prof.Course; -- Join the output of STRING\_SPLIT to the course to get the professor's name

# Homework

## Understand

1. Use STRING\_SPLIT to convert the string "A B C D" into four separate rows with one letter each.
2. Using WideWorldImporters, create a table variable to store delimited strings of customer IDs along with an column called "GroupName". Create a few groups (e.g. GroupA, GroupB, GroupC) and for each, pick a few customer ids to add as a delimited string (e.g. "801,802"). Using this:
   1. Use STRING\_SPLIT and CROSS APPLY to get a list of customer IDs and the group they belong to
   2. Use an inner join to retrieve each customer's details from the sales.Customers table while also displaying the group the customer is in
   3. Use an outer join to retrieve any orders the customer might have in the sales.Orders table