Passing a Table to a Stored Procedure

Introduction

SQL Server 2005 and previous versions do not support passing a table variable to a stored procedure. In one of my previous articles, I had presented a way to pass a table to a stored procedure. There had been a large number of excellent comments in the discussion forum on this subject and a few alternate methods were discussed.

This article introduces the new feature added to SQL Server 2008, which supports passing a TABLE to a stored procedure or function.

The CODE

This article is based on SQL Server 2008 CTP 3. Some of the information may change by the time the product is finally released.

Before we create a Function or Stored Procedure that accepts a TABLE variable, we need to define a User Defined TABLE Type. SQL Server 2008 introduced a new User defined TABLE type. A TABLE type represents the structure of a table that can be passed to a stored procedure or function.

So the first step is to create a User Defined TABLE type. The following TSQL code creates a User defined TABLE type named "ItemInfo".

```
CREATE TYPE ItemInfo AS TABLE
(
    ItemNumber VARCHAR(50),
    Qty INT
)
```

You can use the system view SYS.TYPES to see the type that you have just created. The following query returns all the types defined in the system.

```
SELECT * FROM SYS.TYPES

/*
    If you just need to find information about the TABLE types, you could find it from
    the following TSQL query.
*/

SELECT * FROM SYS.TYPES WHERE is_table_type = 1

/*
    There is another view, which is handy to find information about TABLE types.
*/

SELECT * FROM SYS.TABLE_TYPES
```

We have created a TABLE type that we need. Now let us see how it works. Let us create a variable of type "ItemInfo" and try to insert a few records to it. Then lets query the table variable to see if the information is correctly inserted. [code]

```
/*
```

```
    Let us declare a variable of type ItemInfo which is a TABLE Type
*/
DECLARE @items AS ItemInfo

/*
    Insert values to the variable
*/

INSERT INTO @Items (ItemNumber, Qty)
    SELECT '11000', 100 UNION ALL
    SELECT '22000', 200 UNION ALL
    SELECT '33000', 300

/*
    Lets check if the values are correctly inserted or not
*/
SELECT * FROM @Items

/*
OUTPUT:

ItemNumber                                          Qty
-------------------------------------------------- -----------
11000                                              100
22000                                              200
33000                                              300
*/
```

Now let us create a stored procedure that accepts a TABLE variable. Let us create a very simple stored procedure which accepts a TABLE variable and SELECTs contents of the table.

```
CREATE PROCEDURE TableParamDemo
(
    @Items ItemInfo
)

AS

SELECT *
FROM @Items
```

Well, this would generate the following error:

```
/*
Msg 352, Level 15, State 1, Procedure TableParamDemo, Line 1
The table-valued parameter "@Items" must be declared with the READONLY option.
*/
```

A table variable that is passed to a stored procedure or function should be marked as READONLY. The "callee" cannot modify the table being passed into it. Here is the correct code.

```
CREATE PROCEDURE TableParamDemo
(
    @Items ItemInfo READONLY
)

AS

SELECT *
FROM @Items
```

Now let us execute the stored procedure we just created. Run the following code.

```
/*
    declare the variable
*/
DECLARE @items AS ItemInfo

/*
```

```
    Insert values to the variable
*/

INSERT INTO @Items (ItemNumber, Qty)
    SELECT '11000', 100 UNION ALL
    SELECT '22000', 200 UNION ALL
    SELECT '33000', 300

/*
    Execute the procedure
*/
EXECUTE TableParamDemo @Items

/*
OUTPUT:

ItemNumber                                          Qty
--------------------------------------------------- -----------
11000                                               100
22000                                               200
33000                                               300

*/
```

You cannot modify the TABLE parameter passed into the stored procedure. If you try to do so, you will get an error as shown in the following example.

```
CREATE PROCEDURE TableParamDemo
(
    @Items ItemInfo READONLY
)

AS

SELECT *
FROM @Items

INSERT INTO @Items (ItemNumber, Qty)
    SELECT '1001', 20

/*
OUTPUT:

Msg 10700, Level 16, State 1, Procedure TableParamDemo, Line 11
The table-valued parameter "@Items" is READONLY and cannot be modified.
*/
```

Conclusions

The support for TABLE variables is very interesting. While working with User Defined TABLE Type, please note that you cannot use it as a column of a table. Please also note that, once created, you cannot alter the structure of the TABLE.