SchemaBinding In SQL Server

**What is SchemaBinding**

The SchemaBinding option indicates that your UDF or View will be strictly bound to database objects. Not only helps to bind database objects strictly but also improve performance of query execution plan. This option can be provided at the time of creating UDF/View.

Let’s see how it can be applied in UDF and View:

**In Function**

See the following query  for creating a UDF with SchemaBinding option.

1. **CREATE** **FUNCTION** dbo.fnGetUserID (@**name** **varchar**(10))
2. **RETURNS** **INT**
3. **WITH** **RETURNS** NULL **ON** NULL INPUT,
4. SCHEMABINDING **AS**
5. **BEGIN**
6. **DECLARE** @tempID **INT**
7. **SELECT** @tempID = ID
8. **FROM** dbo.tblUser
9. **WHERE** FirstName = @**name**;
11. **RETURN** @tempID;
12. **END**;

Once query executed successfully, the function created and strictly bound to database objects. As it is strictly bound to objects any kind of operation like drop or modify table will throw an error. Let’s see!  
  
In above example we have created the UDF which depends on tblUser table and now we will try to drop the table directly. See the following **Figure1**it is throwing an error and saying that it can’t be dropped since it is referenced in fnGetUserID.

**Figure 1:** Error in Delete Table  
  
Let’s rename the column of tblUser table. See the following **Figure2,**it isthrowing error since it can’t be applied because function fnGetUserID is bound to this column.

**Figure 2:** Error when Rename a Column  
  
Let's see how SchemaBinding can be implemented in Views.

**In Views**

Same way we can use with SchemaBinding option while creating a View. See the following query:

1. **CREATE** **VIEW** vw\_Test **WITH** SCHEMABINDING **AS**
2. **SELECT** ID **FROM** dbo.tblUser;

Here we create view based on table *tblUser*. Now let’s drop the table tblUser and see what happens. Strange it is throwing an error. Ohh, our view is created with SchemaBinding option.

**Figure 3**: Error when delete a table

**Note1**: You can’t use “\*” in views when you use SchemaBinding. See the following query, if you will execute, it will throw error.

1. **CREATE** **VIEW** vw\_Test **WITH** SCHEMABINDING **AS**
2. **SELECT** \* **FROM** dbo.tblUser;

**Note 2**: If you want to create index on view then your view must be Schema bound.  
  
From above discussion we come to know that UDF/View is strictly bound with database objects when SchemaBinding option is turned on.  
  
Secondly, SchemaBinding helps to improve performance of UDFs and Views. When a object is SchemaBound, query optimizer doesn’t generate unnecessary spool operators for Query execution plan.  
  
Spool operators helps query optimizer to avoid logical problems and perform queries better. Spool reads data and save it in out pre-defined TempDB database. This process is useful when a column volume is high or perform any complex calculation. Spool helps to store the result and use it in future purpose to improve performance. For more on Spooling, please visit [here](https://www.simple-talk.com/sql/learn-sql-server/operator-of-the-week---spools,-eager-spool/).  
  
Take a look at the following example to create a simple UDF:

1. **CREATE** **FUNCTION** dbo.ComputeNum(@i **int**)
2. **RETURNS** **int**
3. **BEGIN**
4. **RETURN** @i \* 2 + 50
5. **END**

In above UDF we didn’t provide SchemaBinding option. In that function we are not accessing any database objects (tables). So do we really need to add SchemaBinding option in this scenario? Yes, we need to add SchemaBinding option because when a function is not SchemaBinding there is no way to ensure that the underlying schema (including the schema of any underlying UDFs or Views that this UDF may call) did not change since its creation. This means SQL Engine has to derive these properties at runtime during every execution of the UDF. To avoid this performance penalty, we mark the UDF as SchemaBinding for safe side data access and do not attempt to derive these properties at runtime which leads to improve performance.  
  
**Advantages**

1. It helps to improve query execution plan better.
2. It checks dependency objects before drop a table/view in database. Suppose a function is SchemaBinding to table. And you are trying to delete the table now; it will throw you an error because of schema binding.