<https://www.sqlshack.com/a-complete-guide-to-t-sql-metadata-functions-in-sql-server/#:~:text=Metadata%2C%20as%20most%20of%20you,in%20SQL%20Server>.

A complete guide to T-SQL Metadata Functions in SQL Server

July 30, 2019 by [Gauri Mahajan](https://www.sqlshack.com/author/gauri-mahajan/)

[](https://www.quest.com/solutions/sql-server/?utm_source=sql_shack&utm_medium=Direct-Outbound%20Marketing&utm_campaign=FY2024_Q2_AMER_SQLServer_SQLShack_ADV&utm_term=&utm_content=banner)

In this article, we will demonstrate T-SQL metadata functions available in the SQL Server.

Introduction

Before we jump into the specifics of metadata functions in SQL Server, let’s quickly understand how exactly term metadata is beneficial for us.

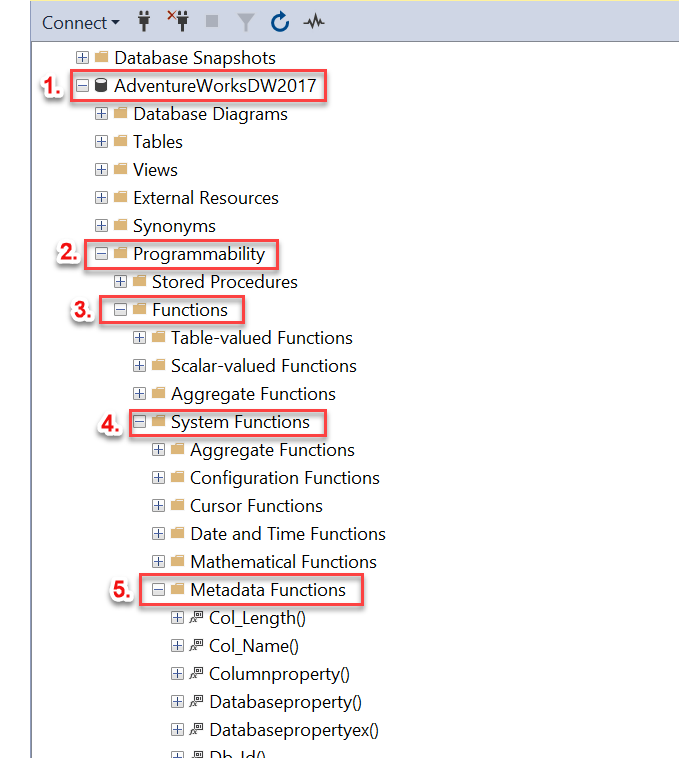
Metadata, as most of you may already know, provides the basic and relevant information about the data. Metadata functions in SQL Server return information about the database, database objects, database files, file groups etc. in SQL Server. Instead of writing complex queries to seek database information, SQL professionals can make use of T-SQL metadata functions to retrieve database info, which helps them to save a great amount of time and effort.

Additionally, they become extremely handy and useful in tracking and monitoring databases when they are used along with the system views. In case you are not familiar with the system views, I would recommend you to refer to this article, [Discovering SQL server instance information using system views](https://www.sqlshack.com/discovering-sql-server-instance-information-using-system-views/) to learn about them.

Preparing our environment

We will need SQL Server, SSMS and a SQL database to understand Metadata functions in SQL Server. Since SQL Server 2019 is not released yet and is in RC, we will run the queries against SQL Server 2017. For SSMS, we will be using the latest version v18.1, in case you don’t’ have it, you can get it from [here](https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-2017). For the sake of brevity, I will be referring to the sample database – *AdventureWorksDW2017* in this article, you can download it from [here](https://docs.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-2017)

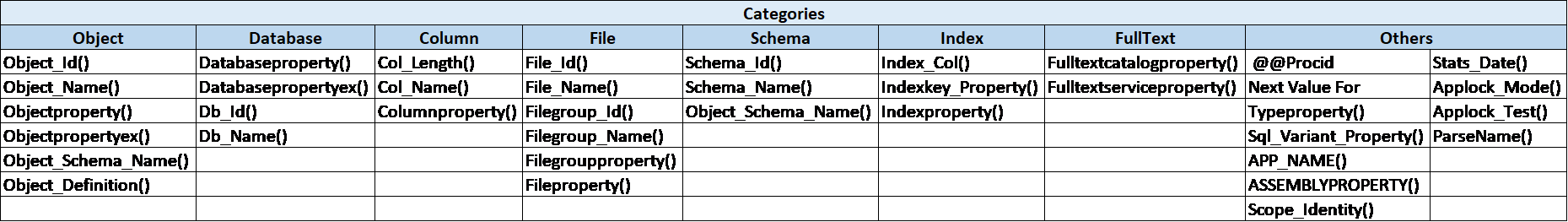
T-SQL metadata functions in SQL Server come under the umbrella of System functions in [SSMS](https://docs.microsoft.com/en-us/sql/ssms/tutorials/connect-query-sql-server?view=sql-server-2017) and can be accessed as shown below. Expand **Databases**, and navigate to **Programmability** in database AdventureWorksDW2017. Double-click **Functions** and **System Functions** and expand **Metadata** **Functions** to see the complete list of metadata functions under it.



And below is the extended list of metadata functions:



I have classified these T-SQL metadata functions in several categories as shown below, and tried to add almost all metadata functions in this consolidated list. You can consider this as a single go-to place to refer to these functions in SQL Server.

[](https://www.sqlshack.com/wp-content/uploads/2019/07/table.png)

It will be difficult for me to cover each one of them in a single article, so I will try to demonstrate the majority of them in the following broad categories:

1. Object IDs – OBJECT\_ID(), DB\_ID(), File\_ID() etc.
2. Object Names – OBJECT\_NAME(), DB\_NAME(), COL\_NAME() etc.
3. Object Property values – OBJECTPROPERTY(), DATABASEPROPERTY(), INDEXPROPERTY() etc.
4. Other metadata functions like @@procid, parsename(), applock\_test() etc.

Let’s go ahead and learn about these functions in detail. We will cover the basic definition, syntax, a few examples, output and any special feature associated with these functions.

Functions returning Object IDs

T-SQL OBJECT\_ID() metadata function

This function returns the database object ID number of a schema object and returns NULL on error. These objects can be tables, views, constraints or stored procedures etc.

Syntax of function OBJECT\_ID()

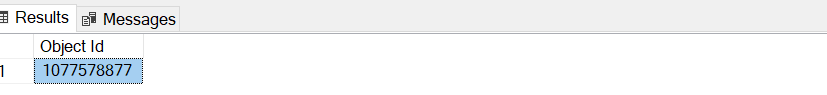
OBJECT\_ID ( ‘database\_name . schema\_name . object\_name’ )

The syntax of this function is self-explanatory.

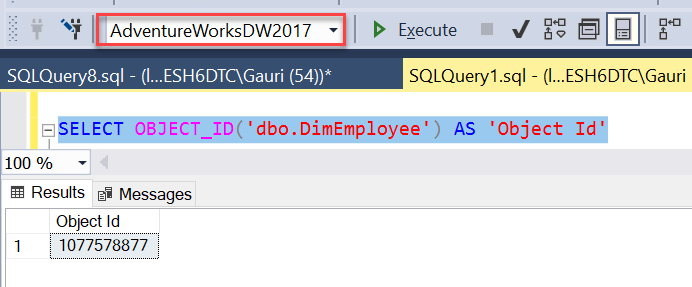
Examples of function OBJECT\_ID()

1. Execute the code below to retrieve Object Id of the table DimEmployee in the database AdventureWorksDW2017 using this function

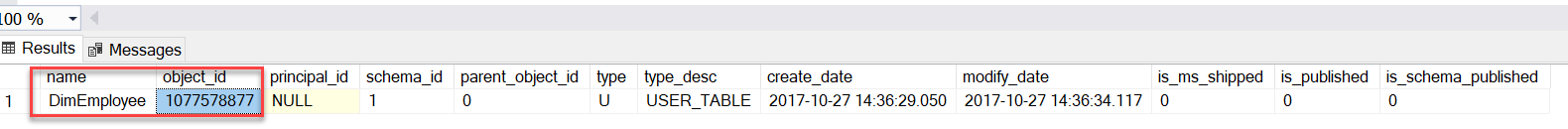
|  |  |
| --- | --- |
| 1 | SELECT OBJECT\_ID('AdventureWorksDW2017.dbo.DimEmployee') AS 'Object Id' |

1. 
2. If the current database context is AdventureWorksDW2017, you can drop database name from the syntax of the function Object\_Id() and use it like the below. Alternatively, you can execute the above code using this ‘USE AdventureWorksDW2017’, it will make sure you are pointing to the right database

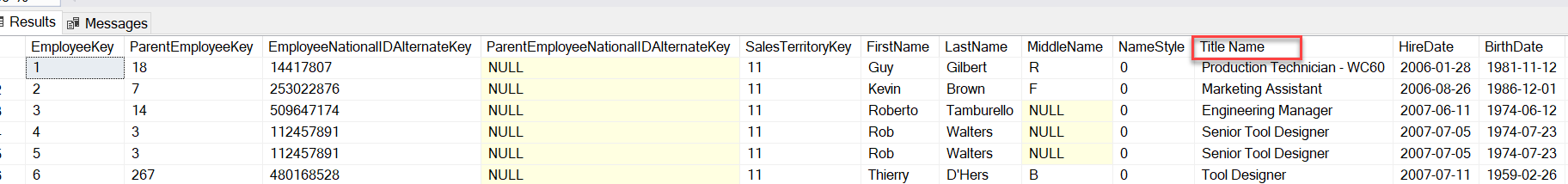
|  |  |
| --- | --- |
| 1 | SELECT OBJECT\_ID('dbo.DimEmployee') AS 'Object Id' |

1. 
2. The Object ID value returned by OBJECT\_ID() function is stored in the sys.objects view in the column object\_id. One such example is shown below, it helps to retrieve information like the name (DimEmployee), object\_id, type (U= user-defined table) etc from this

|  |  |
| --- | --- |
| 1 | SELECT \* FROM sys.objects WHERE object\_id = 1077578877 |

1. [](https://www.sqlshack.com/wp-content/uploads/2019/07/objectid-value-is-stored-in-sys-objects-view-in-ss-e1564412517518.png)
2. Metadata function OBJECT\_ID() is also useful in verifying the existence of objects in the database. And based on if the object exists, we can take further action, very common action is to drop the table
3. One major use of metadata function, OBJECT\_ID() is to verify the existence of objects, like tables in the database. In the below example, this function checks if the table ‘DimEmployee’ exists in the database AdventureWorksDW2017, if yes, it renames the column ‘Title’ to ‘Title Name’

|  |  |
| --- | --- |
| 1  2  3  4 | USE AdventureWorksDW2017;  GO  IF OBJECT\_ID (N'dbo.DimEmployee', N'U') IS NOT NULL  EXEC sp\_rename 'DimEmployee.Title', 'Title Name', 'COLUMN' |

1. [](https://www.sqlshack.com/wp-content/uploads/2019/07/example-of-metadata-function-object_id-in-sql-se-e1564412440755.png)

T-SQL DB\_ID() metadata function

As the name suggests, this function returns the ID of the specified database. When you create a database in SSMS, a new record is inserted into both sys.databases and sys.sysdatabases table with a brand new Id and other relevant information like version, filename etc. There is a metadata function, DB\_ID() that helps to retrieve this number.

Syntax of function DB\_ID ()

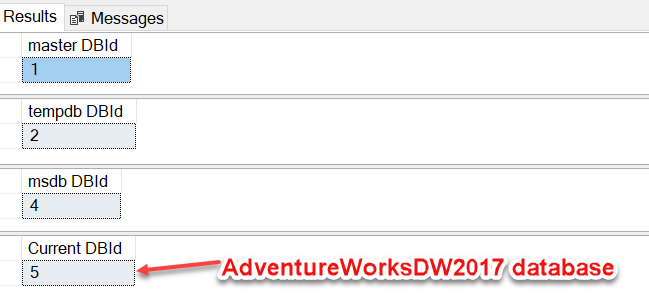
DB\_ID ( [ ‘database\_name’ ] )

To retrieve id value, database name is passed . Unlike function OBJECT\_ID(), this can be executed without passing arguments in it, in that case, it will return database identification (ID) number of the current database.

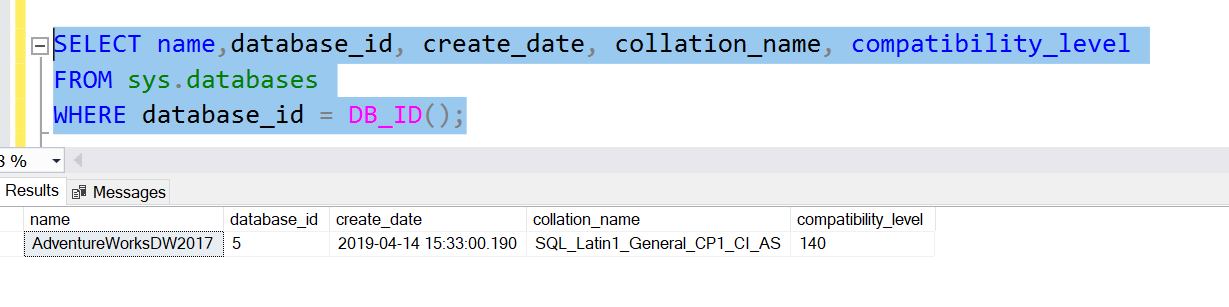
Examples of function DB\_ID()

1. Execute the below examples to see how this function works. The first three queries return the database id for system databases – master, tempdb and msdb. In the last line of code, this function is passed without any argument and it returns Id =5 for the current database AdventureWorksDW2017

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT DB\_ID('master') 'master DBId'  SELECT DB\_ID('tempdb') 'tempdb DBId'  SELECT DB\_ID('msdb') 'msdb DBId'  SELECT DB\_ID() 'Current DBId' |

1. 
2. This function helps to narrow down the data when used along with system views and dynamic management views as shown below

SELECT name,database\_id, create\_date, collation\_name, compatibility\_level FROM sys.databases WHERE database\_id = DB\_ID();



Exercise # 1 – Execute the below query and see if for yourself how results are reduced from 321 to 56 on passing function DB\_ID() as a parameter in the **DMV** [sys.dm\_db\_index\_physical\_stats](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-db-index-physical-stats-transact-sql?view=sql-server-2017)

|  |  |
| --- | --- |
| 1 | SELECT \* FROM sys.dm\_db\_index\_physical\_stats (DB\_ID(),NULL, NULL, NULL, NULL); |

T-SQL FILE\_ID()/ FILE\_IDEX() metadata function

As a database professional, I have often dealt with files like primary data files (.mdf), secondary data files (.ndf) , log files (.ldf) and filegroups in SQL Server. Function FILE\_ID() is a deprecated feature for SQL Server 2017 and FILE\_IDEX() is it’s replacement. This function returns the identification number associated with a database file of the current database.

Syntax of function FILE\_IDEX()

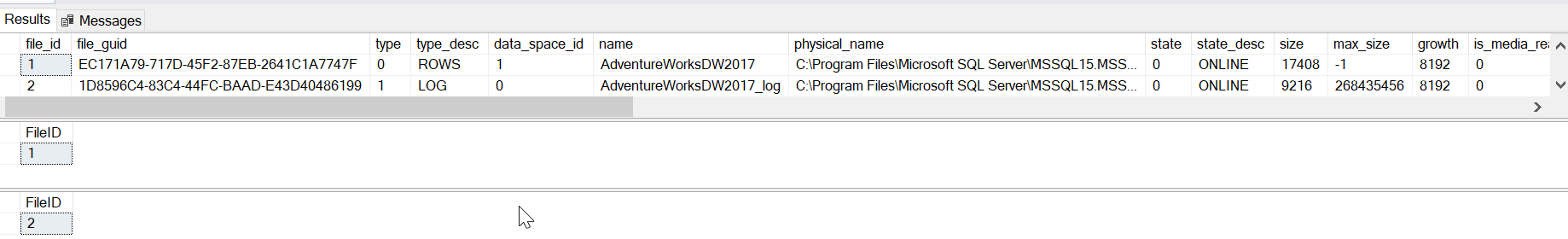
FILE\_IDEX ( file\_name )

Where, file\_name is the name of the logical database file and it returns type int.

Examples of function FILE\_IDEX()

Catalog view, [sys.database\_files](https://docs.microsoft.com/en-us/sql/relational-databases/system-catalog-views/sys-database-files-transact-sql?view=sql-server-2017) contains information of current database file names. From this table, we can retrieve the name of the file and pass it in the FILE\_IDEX() to obtain File ID.

|  |  |
| --- | --- |
| 1  2  3 | SELECT \* FROM sys.database\_files  SELECT FILE\_IDEX('AdventureWorksDW2017') AS FileID  SELECT FILE\_IDEX('AdventureWorksDW2017\_log') AS FileID |

[](https://www.sqlshack.com/wp-content/uploads/2019/07/examples-of-metadata-function-file_idex-in-sql-s-e1564412392502.png)

T-SQL SCHEMA\_ID() metadata function

As the name suggests, it returns the schema id of the schema name.

Syntax of function SCHEMA\_ID()

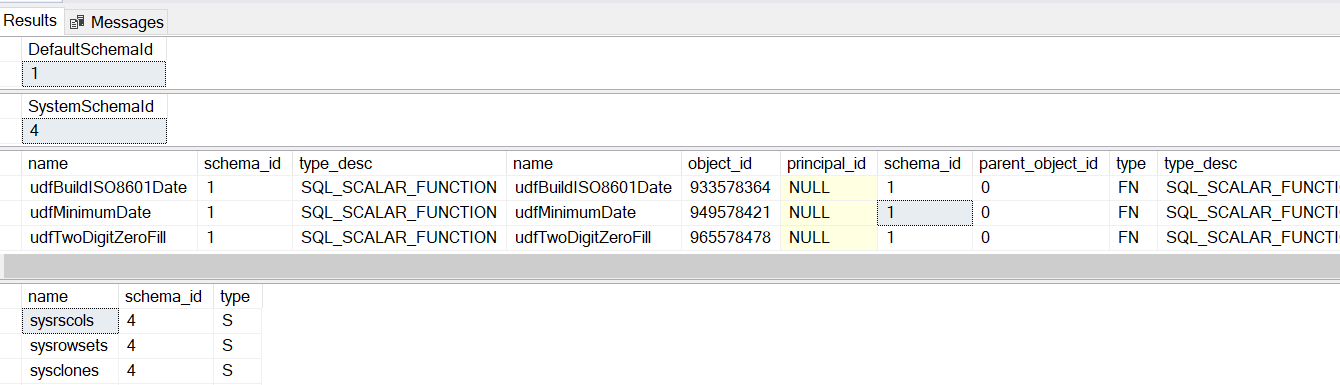
SCHEMA\_ID ( [ schema\_name ] )

Which means we can obtain schema id of an object by passing the schema name as an argument in it.

Examples of function SCHEMA\_ID()

The following first two examples return schema Id of default schema and system schema respectively using SCHEMA\_ID(). The subsequent example helps to retrieve information from the system catalog view, sys.objects for the ‘dbo’ schema and scalar functions.

|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT SCHEMA\_ID() AS DefaultSchemaId  SELECT SCHEMA\_ID('sys') AS SystemSchemaId    SELECT name, schema\_id, type\_desc,\* FROM sys.objects WHERE schema\_id = SCHEMA\_ID('dbo') AND type = 'FN';  SELECT name, schema\_id, type FROM sys.objects WHERE schema\_id = SCHEMA\_ID('sys'); |



T-SQL FILEGROUP\_ID() metadata function

SQL Server organizes data files in the collection named File Groups and there are several file group functions that helps DBA in their tasks. The function FILEGROUP\_ID() returns the file ID for the corresponding file group name.

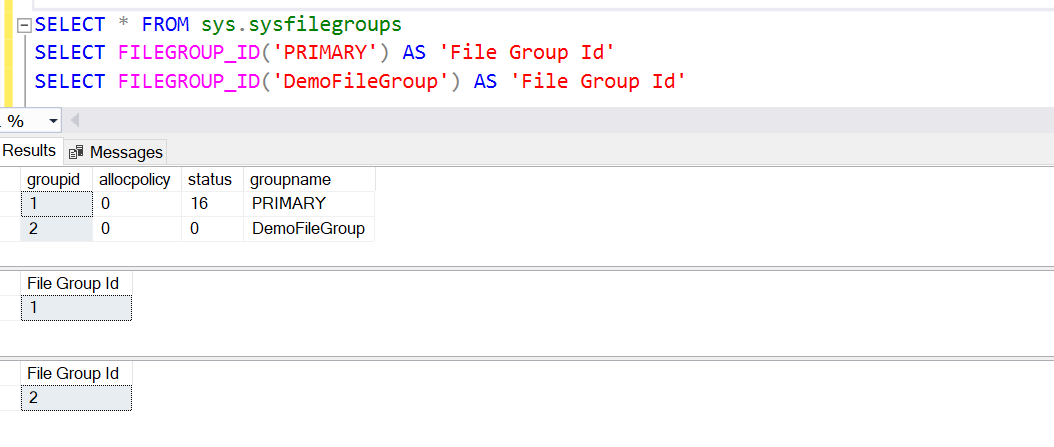
Syntax of function FILEGROUP\_ID()

FILEGROUP\_ID ( ‘filegroup\_name’ )

Where filegroup\_name represents the filegroup name for which filegroup id is returned.

Examples of function FILEGROUP\_ID()

I have created a filegroup named, DemoFileGroup to demonstrate how this function works. You can check the system view, sys.sysfilegroups to retrieve information like group id, name etc. for a database. Below queries return filegroup ids for filegroups names Primary and DemoFileGroup in the AdventureWorksDW2017 database.



Metadata Functions returning Object Names

T-SQL OBJECT\_NAME() metadata function

OBJECT\_NAME() function returns the name of the database object based on the given Object ID.

Syntax of function OBJECT\_NAME()

Below is the syntax of the function. Basically, we need to pass ObjectID in this function to get the Object name.

OBJECT\_NAME ( object\_id [, database\_id ] )

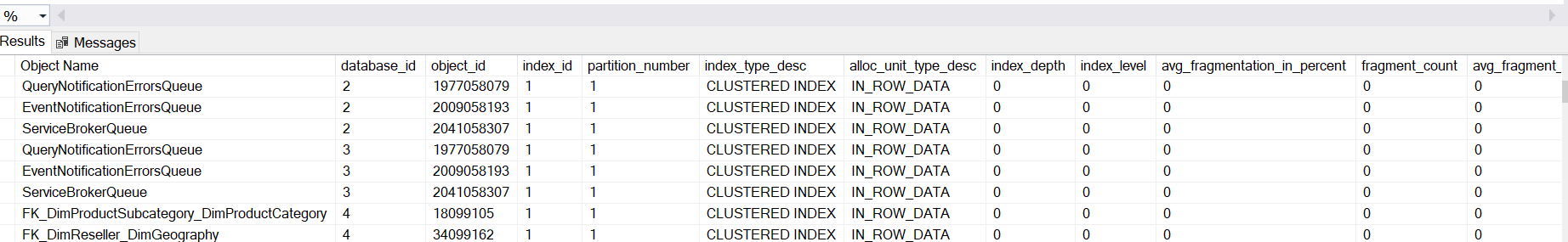
Where:

* object\_id is int and the ID of the object
* database\_id is int as well and belongs to the ID of the database where the object is expected to exist

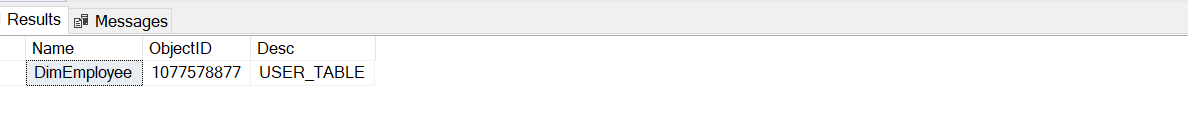
Examples of function OBJECT\_NAME()

1. The following example makes use of the function OBJECT\_NAME() to retrieve object name along with all other columns in the DMV [sys.dm\_db\_index\_physical\_stats.](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-db-index-physical-stats-transact-sql?view=sql-server-2017) for all objects in the database

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT  OBJECT\_NAME(object\_id) AS 'Object Name',\*  FROM sys.dm\_db\_index\_physical\_stats (NULL,NULL,NULL,NULL,NULL)  ORDER BY index\_type\_desc |

1. [](https://www.sqlshack.com/wp-content/uploads/2019/07/metadata-function-object_name-example-in-sql-ser-e1564412309470.png)
2. This function can also be used with system catalog views like sys.objects, sys.tables etc. and WHERE clause to return columns specified by this function in the WHERE clause as shown below. To keep things simpler for you, I will continue with the same table, DimEmployee and make sure current database context points to the database, AdventureWorksDW2017

|  |  |
| --- | --- |
| 1  2  3  4 | DECLARE @ObId int;  SET @ObId = (SELECT OBJECT\_ID('dbo.DimEmployee', 'U'));  SELECT name AS Name, object\_id AS ObjectID, type\_desc AS 'Desc' FROM sys.tables  WHERE name = OBJECT\_NAME(@ObId) |

1. 

T-SQL DB\_NAME() metadata function

This function returns the name of the database.

Syntax of function DB\_NAME ()

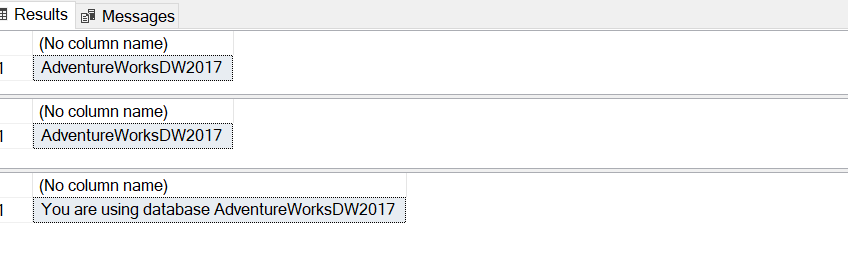
DB\_NAME ( [ database\_id ] )

Where database\_id is the id of the database to be retrieved. This function works without this id as well, in that case, the current database name is returned.

Examples of function DB\_NAME()

In the below examples, DB\_NAME() returns the name of the database in use and by passing database id as 6. This function also helps in verification, here it verifies if the current database value is AdventureWorksDW2017 or not, and accordingly it takes further action.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT DB\_NAME()  SELECT DB\_NAME(6)  IF DB\_NAME() = 'AdventureWorksDW2017'  SELECT 'You are using database ' + DB\_NAME()  ELSE  SELECT 'You are using the wrong DB' |



* *Just a small****note****– These examples might look simple and straight-forward because here we are just referring to a single object in a single database, however, in reality, SQL professionals often have to deal with a large number of tables in several databases at the same time, in such scenarios, these functions come in handy*

T-SQL COL\_NAME() metadata function

This function returns the name of a column for a corresponding table id and column id.

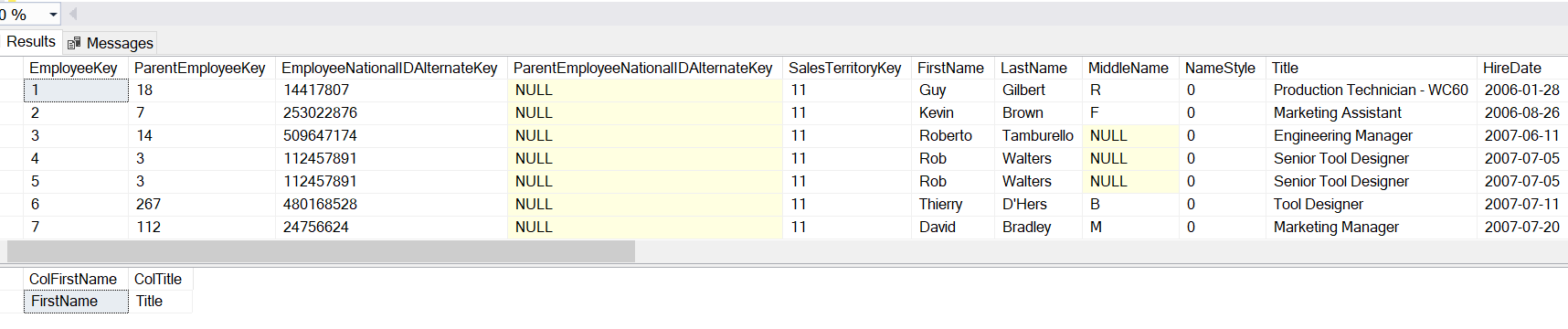
Syntax of function COL\_NAME ()

COL\_NAME ( table\_id , column\_id )

Examples of function COL\_NAME()

The below example returns the name of the 6th and 10th column in a table, dbo.DimEmployee.

|  |  |
| --- | --- |
| 1  2  3 | SELECT \* FROM DimEmployee  SELECT COL\_NAME(OBJECT\_ID('dbo.DimEmployee'), 6) AS ColFirstName,  COL\_NAME(OBJECT\_ID('dbo.DimEmployee'), 10) AS ColTitle |

[](https://www.sqlshack.com/wp-content/uploads/2019/07/example-of-metadata-function-col_name-in-sql-ser-e1564412158655.png)

T-SQL FILE\_NAME() metadata function

This function returns the name of the file for the given file ID.

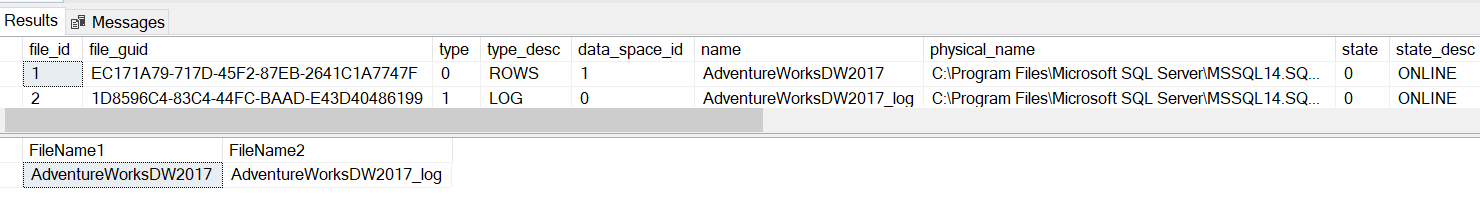
Syntax of function FILE\_NAME()

FILE\_NAME (file\_id)

Examples of function FILE\_NAME()

The below example returns the file names for ID =1 and ID=2 in the AdventureWorksDW2017 database.

|  |  |
| --- | --- |
| 1  2 | SELECT \* FROM sys.database\_files  SELECT FILE\_NAME(1) AS 'FileName1', FILE\_NAME(2) AS 'FileName2' |

[](https://www.sqlshack.com/wp-content/uploads/2019/07/example-of-metadata-function-file_name-in-sql-se-e1564412128646.png)

T-SQL FILEGROUP\_NAME() metadata function

This returns the filegroup name for the given filegroup ID.

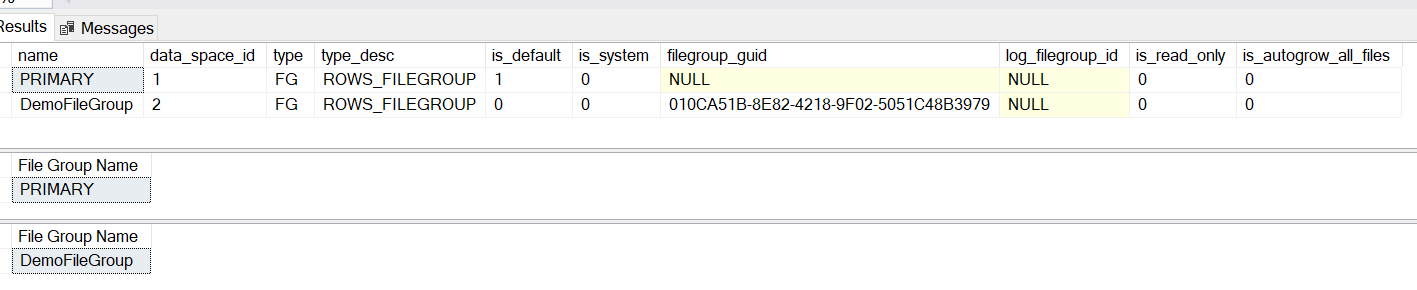
Syntax of function FILEGROUP\_NAME()

FILEGROUP\_NAME ( filegroup\_id )

Examples of function FILEGROUP\_NAME()

The following examples return filegroup name for the specified filegroup ids in the AdventureWorksDW2017 database.

|  |  |
| --- | --- |
| 1  2  3 | SELECT \* FROM sys.filegroups  SELECT FILEGROUP\_NAME(1) AS 'File Group Name'  SELECT FILEGROUP\_NAME(2) AS 'File Group Name' |

[](https://www.sqlshack.com/wp-content/uploads/2019/07/examples-of-metadata-function-filegroup_name-in-e1564412066118.png)

Functions returning Object Properties

T-SQL OBJECTPROPERTY() metadata function

OBJECTPROPERTY() metadata function returns information related to several properties about objects in the current database.

**Syntax of function OBJECTPROPERTY()**

OBJECTPROPERTY ( id , property )

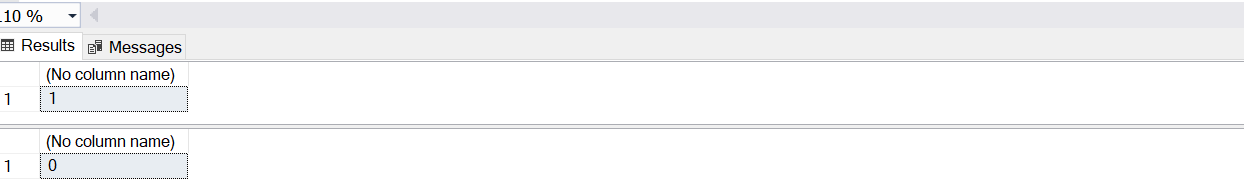
Where:

* id is the object id and is int
* property here represents the name of the property of the object specified by the id argument. There is a battalion of properties available in SQL Server and explaining them in this article is certainly out of scope here. You can go to the property name section over [here](https://docs.microsoft.com/en-us/sql/t-sql/functions/objectproperty-transact-sql?view=sql-server-2017) to understand these properties

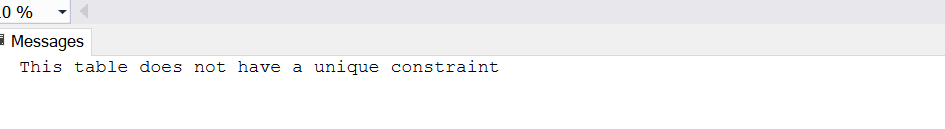
Examples of function OBJECTPROPERTY()

1. The following examples check if the table DimEmployee, has a foreign key reference and is memory- optimized. The value 1 means true and 0 means false

|  |  |
| --- | --- |
| 1  2 | SELECT OBJECTPROPERTY(OBJECT\_ID(N'AdventureWorksDW2017.dbo.DimEmployee'), 'TableHasForeignRef');  SELECT OBJECTPROPERTY(OBJECT\_ID(N'AdventureWorksDW2017.dbo.DimEmployee'), 'TableIsMemoryOptimized'); |

1. 
2. Here we are using this function to validate a property, IsUniqueCnst on the table DimEmployee and based on the results, printing the statements

|  |  |
| --- | --- |
| 1  2  3  4  5 | USE AdventureWorksDW2017  IF OBJECTPROPERTY(OBJECT\_ID('dbo.DimEmployee'), 'IsUniqueCnst') = 0    PRINT 'This table does not have a unique constraint'  ELSE    PRINT 'This table has a unique constraint'; |

1. 

T-SQL OBJECTPROPERTYEX() metadata function

This function works on similar lines as function OBJECTPROPERTY(), returning information about the properties of objects in the database. Microsoft added some extra properties in this function and prefixed this function by ex.

Syntax of function OBJECT\_PROPERTYEX()

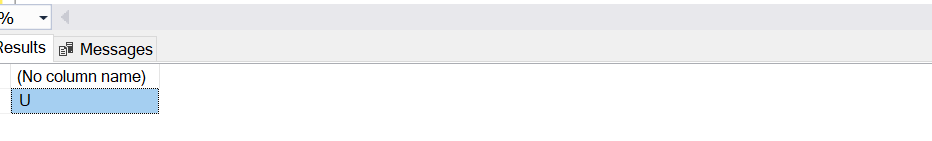
OBJECTPROPERTYEX ( id , property )

The Id here represents the id of an object that is being looked up in the database and property returns the property name. The only difference is that this function returns sql\_variant and can support non-integer properties.

Examples of function OBJECT\_PROPERTYEX()

The following example returns the base type of our table DimEmployee which is a User Table (U).

|  |  |
| --- | --- |
| 1 | SELECT OBJECTPROPERTYEX ( object\_id('dbo.DimEmployee'), N'BaseType'); |



T-SQL DATABASEPROPERTYEX() metadata function

This function returns information related to several properties of the specified database.

Syntax of function DATABASEPROPERTYEX()

DATABASEPROPERTYEX ( database , property )

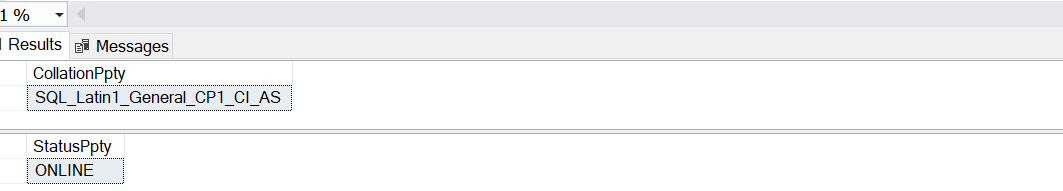
Where, database represents the name of the database for which this function will return the property information and property denotes the property that needs to be returned.

There are a lot of database properties that this metadata function supports, you can go to the property name section over [here](https://docs.microsoft.com/en-us/sql/t-sql/functions/databasepropertyex-transact-sql?view=sql-server-2017) to understand all these properties.

Examples of function DATABASEPROPERTYEX()

In the following examples, we can retrieve information like database collation and status using the below properties.

|  |  |
| --- | --- |
| 1  2 | SELECT DATABASEPROPERTYEX('AdventureWorksDW2017', 'Collation') AS CollationPpty  SELECT DATABASEPROPERTYEX('AdventureWorksDW2017', 'Status') AS StatusPpty |



T-SQL COLUMNPROPERTY() metadata function

This function returns information about a column.

Syntax of function COLUMNPROPERTY ()

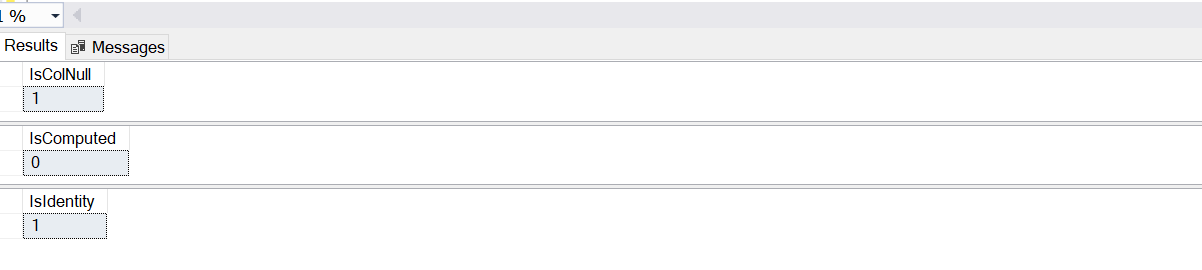
COLUMNPROPERTY ( id , column , property )

Where, ID means the id of the table or procedure, the column represents the name of the column and property here refers to the type of information that this function will return. Like objectproperty() and databaseproperty() functions, there are too many properties to cover here, please refer [here](https://docs.microsoft.com/en-us/sql/t-sql/functions/columnproperty-transact-sql?view=sql-server-2017) to learn about these properties.

Examples of function COLUMNPROPERTY ()

The following examples return properties to check if the column allows null values, is computed and is an identity column for three different columns in the DimEmployee table of AdventureWorksDW2017 database.

|  |  |
| --- | --- |
| 1  2  3 | SELECT COLUMNPROPERTY( OBJECT\_ID('DimEmployee'),'MiddleName','AllowsNull') AS 'IsColNull'  SELECT COLUMNPROPERTY( OBJECT\_ID('DimEmployee'),'EmployeeKey','IsComputed') AS 'IsComputed'  SELECT COLUMNPROPERTY( OBJECT\_ID('DimEmployee'),'EmployeeKey','IsIdentity') AS 'IsIdentity' |



Some more T-SQL metadata functions

Let’s look at some of the other metadata functions that provide equally important information about the database and its objects.

T-SQL @@PROCID metadata function

As we can make out from the name, @@procid, this function returns the ID of the current procedure, by procedure here, I mean, stored procedure, user-defined function (UDF), or a trigger.

Syntax of function @@PROCID

@@PROCID

This function returns int type.

Examples of function @@PROCID

The following code makes use of the @@PROCID function in the stored procedure and executing this procedure returns the current if of this stored procedure, DemoProcID.

|  |  |
| --- | --- |
| 1  2  3  4  5 | CREATE PROC DemoProcID  AS BEGIN  SELECT @@PROCID AS ProcID  END  Exec DemoProcID |

A screenshot of a computer

Description automatically generated

T-SQL OBJECT\_DEFINITION() metadata function

This function helps to return the T-SQL code of the object specified, it can be a constraint, stored procedure, function, view etc.

Syntax of function OBJECT\_ DEFINITION ()

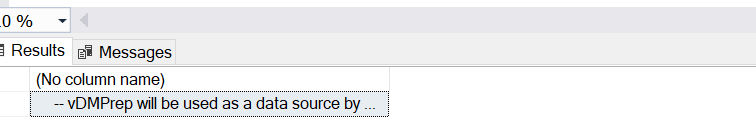
OBJECT\_DEFINITION ( object\_id )

Syntax of this function makes use of object\_id which is int, the similar way we have seen above for other OBJECT related functions. This functions returns type nvarchar(max).

Examples of function OBJECT\_ DEFINITION ()

The following example returns the definition of a view, vDMPrep in database AdventureWorksDW2017. Copy and paste the result value in a new query window to see the CREATE definition of this view.

|  |  |
| --- | --- |
| 1 | SELECT OBJECT\_DEFINITION (OBJECT\_ID(N'dbo.vDMPrep')) |



T-SQL COL\_LENGTH() metadata function

This function provides the length of the column, in bytes.

Syntax of function COL\_LENGTH()

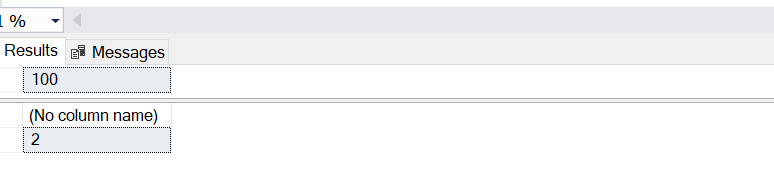
COL\_LENGTH ( ‘table’ , ‘column’ )

Where table represents table name and column is the name of the column whose length is returned by this function.

Examples of function COL\_LENGTH()

We can retrieve lengths of columns FirstName and Gender in table dimEmployee using the below examples.

|  |  |
| --- | --- |
| 1  2 | SELECT COL\_LENGTH('dbo.dimEmployee','FirstName');  SELECT COL\_LENGTH('dbo.dimEmployee','Gender'); |



T-SQL PARSENAME() metadata function

This function returns the specified part of the given object name. These parts can be database name, schema/owner name and database name etc.

Syntax of function PARSENAME()

PARSENAME ( ‘object\_name’ , object\_part )

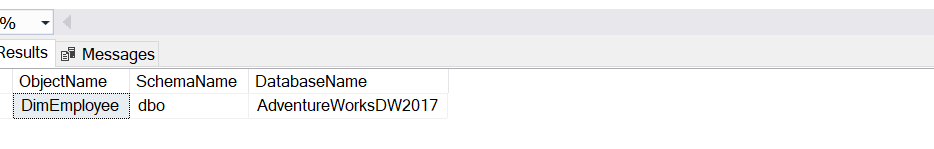
Where object\_name is the name of the object for which to return the specified object part and object\_part is the part of the object to be returned. The return type of object\_part is int and these values can be as follows:

1 – Object Name, 2 – Schema Name, 3 – Database Name, 4 – Server Name

Examples of function PARSENAME()

The examples shown below, return information like schema name and database name for DimEmployee table that we had used throughout this article in the AdventureWorksDW2017 database.

|  |  |
| --- | --- |
| 1  2  3 | SELECT PARSENAME('dbo.DimEmployee', 1) AS ObjectName ,  PARSENAME('dbo.DimEmployee', 2) AS SchemaName,  PARSENAME('AdventureWorksDW2017.dbo.DimEmployee', 3) AS DatabaseName |



Observations

We can conclude the following observations from our exercise above on the metadata functions:

* Metadata functions are both scalar and non-deterministic, implying that these functions return only single value, and also the result values vary even if we pass the same input parameters in them
* These functions are easily queried with a simple SELECT statement
* These functions return NULL on error
* These functions are available in a variety of formats:
  + Majority of them use a suffix, ‘()’, for eg. Object\_Id(), Columnproperty() etc.
  + A few are used with a prefix, ‘@@’, for eg. @@PROCID
  + Some do not make use of any prefix or suffix, for eg. – NEXT VALUE FOR

Conclusion