DATA DICTIONARY CUSTOM PROCEDURES

How to access and add to the internal documentation of the database

v.2 09/09/2020 UPDATE 05/05/2021

Dave Babler

Contents

[STARTING INFO 2](#_Toc71097555)

[TABLE LEVEL INFORMATION 2](#_Toc71097556)

[Add Comment (description) To a Table 2](#_Toc71097557)

[Table commenting best practice 2](#_Toc71097558)

[Show Table Metadata With Comments. 3](#_Toc71097559)

[Show Just The Table Comment (description) 3](#_Toc71097560)

[Find Table With Name Like 3](#_Toc71097561)

[Determine If Table Exists 3](#_Toc71097562)

[COLUMN LEVEL INFORMATION 4](#_Toc71097563)

[Add Comment (description) To A Column 4](#_Toc71097564)

[Column commenting best practice 4](#_Toc71097565)

[Reminder 5](#_Toc71097566)

[Show Single Column’s Comments 5](#_Toc71097567)

[Determine If Column Exists 5](#_Toc71097568)

[~~Column Compare~~ 5](#_Toc71097569)

[MISC. TOOLS 5](#_Toc71097570)

[Suppress Procedural Output 5](#_Toc71097571)

[THE PROCEDURES FOR SUPPRESSING OUTPUT 7](#_Toc71097572)

DATA DICTIONARY PROCEDURES

# STARTING INFO

All procedures in this must be called with a fully qualified name unless system level synonyms are implemented. That means each must be called as Utility.DD.[*TheProc/FuncYouWantToCall*].

You must make calls to other objects using a fully qualified name with the exception of your organization’s default database and its default schema.

For example, if you need a DESCRIBE on AdventureWorks’s Person table in the Person Schema you’d call:

EXEC Utility.DD.Describe 'AdventureWorks.Person.Person';

But if your organization’s default database is Galactic and its default schema is dbo then either of these will work fine for checking out the Repairs table.

EXEC Utility.DD.Describe 'Galactic.dbo.Repairs';

EXEC Utility.DD.Describe 'Repairs';

# TABLE LEVEL INFORMATION

## Add Comment (description) To a Table

-Most columns and *all* tables should have comments in plain-concise-English what the table is for. The cap has been set to 320 characters to make sure that only relevant information is added.

Tables often represent one of the following:

* Real world tangible objects (list of people).
* Real world common concept (a join table is an example of this, how a list of Rx’s relates to a list of patients).

### Table commenting best practice

Comments on tables should describe either how the concept applies to the code, or how the real-life object applies to the code. Example:

* A veterinary database has a table “Patients” it has this comment: *“This contains data about patients including the relation to the patient’s owner/parent along with immutable biological characteristics such as species, breed, and date of birth”.*
* A table used to hold data for a report in a financial database has this comment: ‘*This table will hold the data that is generated from [prc\_Asset\_Liability\_Asset]*’
* A table used to join VINs to Owner contains just two columns VIN & OwnerID has this comment: ‘*This table is used to join the vehicle table to the customer table, OwnerID joins to Customer.CustomerID & VIN joins to Vehicle.VIN*’.

The Syntax is as follows:

EXEC Utility.DD.AddTableComment'TABLE\_NAME', 'COMMENT';

## Show Table Metadata With Comments.

To show a table’s metadata with comments on the columns use the following:

EXEC Utility.DD.Describe 'TableName';

This shows:

1. The top line of the table which is just the table name and the overall comment on the table.
2. The rest of the lines show all the most needed metadata for the columns including:
   1. Datatype
   2. Scale & Precision
   3. If it is a primary or foreign key.
   4. If it is a foreign key, then the name of the parent.
   5. Any column comments. (Called Descriptions).

## Show Just The Table Comment (description)

If you want to know what a table is used for you may use the following query to retrieve the table’s description (comment).

EXEC Utility.DD.ShowTableComment 'TableName';

## Find Table With Name Like

If you can only partially remember the name of a table this procedure will help you find the actual name of the table by searching by partial name. As a bonus, this also shows the comment on the table, so you know what the table is for! Note: casing does not matter, the procedure drops what you enter in, and all the table names to lowercase for searching. Try using the word ‘fund’ as the partial table name in our database to get a good look at how this works.

EXEC Utility.DD.TableNameLike 'partialname';

## Determine If Table Exists

This data dictionary procedure would likely only be useful called from other stored procedures. There are times when you need to check to see if a live table exists before you can proceed this procedure will help with that.

DECLARE @boolTableExists BIT

    , @strOutputMessage VARCHAR(400);*--strOutputMessage THIS IS OPTIONAL YOU CAN EXECUTE THE PROC WITHOUT IT.*

EXEC Utility.DD.TableExist 'tablename'

    , @boolTableExists OUTPUT

    , @strOutputMessage OUTPUT;

SELECT @boolTableExists

    , @strOutputMessage;

# COLUMN LEVEL INFORMATION

## Add Comment (description) To A Column

*Most* columns should have comments even though *all* column headers should be self-documenting in name.

Columns typically represent:

* A representation of a real-world data object such as: a VIN, a person’s age, the number of cows per acre.
* Data that is inferred, calculated, or defined by other columns such as: A sub-total column calculated on insert/update trigger.
* A conceptual object that points to or refers to another object such as: a foreign key value, a pointer to a URL where a file is stored.
* A file object such[[1]](#footnote-2) as: a column in a Patient table at a Vet that holds a picture of the pet.

### Column commenting best practice

Comments on columns should describe either how the date applies to the code, or how the real-life object applies to the code. Example:

* Veterinary Database, Patient.Gender column: *‘Contains the gender of the animal the constraints are M, F, MN, FN, (N for neuter), and O for pets like sea-anemones that do not really have gender in the way mammals do’.*
* State Prison Database Inmate.ImageRecent column: ‘*Contains an image of the convict updated monthly’*.
* Financial Database: PL\_Policy.PolicyNumber column: ‘*Contains the policy number as supplied by the client ‘Pacific Life’ it is also a foreign key do to a recursive parent-child relationship to Policy\_SF’*.

Note how the comments are concise while remaining descriptive, direct, and easily understood.

Syntax:

EXEC Utility.DD.AddColumnComment 'TABLE\_NAME', 'ColumnName', 'Comment';

### Reminder

You can get much information about a column from using the Utility.DD.DESCRIBE procedure on the table.

## Show Single Column’s Comments

To see the comments (description) of a specific single column use:

EXEC Utility.DD.ShowColumnComment 'TABLE\_NAME', 'columnName';

## Determine If Column Exists

This data dictionary procedure would likely only be useful called from other stored procedures. There are times when you need to check to see if a column exists within a table before you can proceed this procedure will help with that.

DECLARE @boolColumnExists BIT

    , @strOutputMessage VARCHAR(400);*--strOutputMessage THIS IS OPTIONAL YOU CAN EXECUTE THE PROC WITHOUT IT.*

EXEC Utility.DD.ColumnExist 'TABLE\_NAME'

    , 'ColumnName'

    , @boolColumnExists OUTPUT

    , @strOutputMessage OUTPUT;

SELECT @boolColumnExists

    , @strOutputMessage;

## ~~Column Compare~~

~~During the creation of stored procedures or ad-hoc queries that are run frequently (such as those in reports) joining tables is a frequent activity. This procedure will help you determine if you can join the two columns. It will also describe the data types and precision of both columns. Even if the join will work, should the datatypes or precisions not match then it needs to be made into a high priority for the DBA to correct, especially if used in a report. Please inform the DBA upon discovery of any mismatched datatypes, lengths, or precisions of needed JOIN columns.~~

~~EXEC Utility.DD.ColumnCompare 'TABLE\_NAME1', 'ColumnName1', 'TABLE\_NAME2', 'ColumnName2';~~

Temporarily depreciated until a future date.

# MISC. TOOLS

This section involves tools that access the data dictionary for purposes outside of the direct use of meta-data.

## Suppress Procedural Output

If you have one procedure that does:

* A large amount of calculations,
* CRUD activity that may need to be repeated in other procedures,
* Creates an OUTPUT parameter to pass to another procedure,
* Any time you output anything to view as a SELECT.

If any of the above applies please use the function made for suppressing output to a procedure.

1. First make sure you wrap your SELECT or PRINT output in an IF statement that refers to the function that creates the Boolean 1/0.
2. Call the function in the inner procedure.
3. Create the temp table in the outer procedure.
4. Drop the temp table after the outer procedure calls the inner procedure.

The function name is:

 Utility.UTL.fn\_SuppressOutput();

It must be assigned to a variable.

    DECLARE @boolSuppressVisualOutput BIT;

    SELECT @boolSuppressVisualOutput = dbo.fn\_SuppressOutput();

The temporary table in the calling (outer procedure) must be declared as:

        CREATE TABLE #\_\_suppress\_results (col1 INT);

And of course, *do not forget to drop it at the end of the procedure.*

APPENDIX

# THE PROCEDURES FOR SUPPRESSING OUTPUT

#### The outer procedure

*-- =============================================*

*-- Author:      Dave Babler*

*-- Create date: 9/10/2020*

*-- Description: Demonstrates Suppress Output*

*-- Subprocedures: 1. [fn\_SuppressOutput]*

*-- =============================================*

CREATE OR ALTER PROCEDURE TEMP\_SUPPOUT1

*-- Add the parameters for the stored procedure here*

    @strInVal VARCHAR(80) = 0

    , @strOutMessage VARCHAR(160) = NULL OUTPUT

AS

BEGIN TRY

    SET NOCOUNT ON;

*-- SET NOCOUNT ON added to prevent extra result sets from*

*-- interfering with SELECT statements.*

    DECLARE @boolSuppressVisualOutput BIT;

    SELECT @boolSuppressVisualOutput = dbo.fn\_SuppressOutput();

    SET @strOutMessage = @strInVal + ' Team';

    IF @boolSuppressVisualOutput = 0

    BEGIN

        SELECT @strInVal + ' I should only show when output is not suppressed';

    END

END TRY

BEGIN CATCH

    INSERT INTO dbo.DB\_EXCEPTION\_TANK

    VALUES (

        SUSER\_SNAME()

        , ERROR\_NUMBER()

        , ERROR\_STATE()

        , ERROR\_SEVERITY()

        , ERROR\_PROCEDURE()

        , ERROR\_LINE()

        , ERROR\_MESSAGE()

        , GETDATE()

        );

END CATCH;

#### The inner procedure

*-- =============================================*

*-- Author:      Dave Babler*

*-- Create date: 9/10/2020*

*-- Description: Shows output suppression*

*-- Subprocedures: 1. [TEMP\_SUPPOUT1]*

*-- =============================================*

CREATE OR ALTER PROCEDURE TEMP\_SUPPOUT2

*-- Add the parameters for the stored procedure here*

    @strFirstPartMess VARCHAR(80)

    , @strSecondPartMess VARCHAR(80)

AS

BEGIN TRY

    DECLARE @strProc1Output VARCHAR(80);

    DECLARE @strFinalMess VARCHAR(200);

*-- SET NOCOUNT ON added to prevent extra result sets from*

*-- interfering with SELECT statements.*

    SET NOCOUNT ON;

*--CREATE A FAKE TEMP TABLE TO SUPPRESS RESULTS*

    CREATE TABLE #\_\_suppress\_results (col1 INT);

    EXEC TEMP\_SUPPOUT1 @strFirstPartMess

        , @strProc1Output OUTPUT;

    SET @strFinalMess = @strProc1Output + ' ' + @strSecondPartMess;

    SELECT @strFinalMess;

    DROP TABLE IF EXISTS #\_\_suppress\_results;*-- ALWAYS DROP IT!*

END TRY

    BEGIN CATCH

INSERT INTO CustomLog.ERR.DB\_EXCEPTION\_TANK (

[DatabaseName]

, [UserName]

, [ErrorNumber]

, [ErrorState]

, [ErrorSeverity]

, [ErrorLine]

, [ErrorProcedure]

, [ErrorMessage]

, [ErrorDateTime]

)

VALUES (

DB\_NAME()

, SUSER\_SNAME()

, ERROR\_NUMBER()

, ERROR\_STATE()

, ERROR\_SEVERITY()

, ERROR\_LINE()

, ERROR\_PROCEDURE()

, ERROR\_MESSAGE()

, GETDATE()

);

END CATCH;

1. In my professional opinion, Oracle is the only DB that does a really good job with BLOB types; please do not use BLOBs here at Mezrah without talking to the DBA. [↑](#footnote-ref-2)