(T37)討論Dependency  
CourseGUID: e48417fc-9db5-4e99-822c-706c5ccef6cc  
=======================================================================  
(T37)討論Dependency  
=======================================================================  
0. Summary

1. Create Sample Data

-----------

2. sys.dm\_sql\_referencing\_entities, dm\_sql\_referencing\_entities, sp\_depends

2.1. sys.dm\_sql\_referencing\_entities V.S. Execute sp\_depends 'ObjectName'

2.2. sys.dm\_sql\_referenced\_entities V.S. Execute sp\_depends 'ObjectName'

2.3. Clean up

-----------

3. sys.dm\_sql\_referencing\_entities, dm\_sql\_referencing\_entities, sp\_depends

3.1. Create Sample Data

3.2. Find dependencies

3.3. Drop the table and then re-create it.

3.4. Find dependencies.

3.5. Clean up  
=======================================================================

0. Summary

sys.dm\_sql\_referencing\_entities

sys.dm\_sql\_referenced\_entities

sp\_depends

--Referencing entity V.S. Referenced entity

--Schema-bound dependency V.S. Non-schema-bound dependency

--sys.dm\_sql\_referencing\_entities V.S. sys.dm\_sql\_referenced\_entities

--sp\_depends

1.

Referencing entity V.S. Referenced entity

1.1.

In sammary,

Referencing entity depends on Referenced entity

1.2.

--CREATE VIEW VwBookType  -- referencing entity

--AS

--    SELECT  \*

--    FROM    BookType;  --referenced entity

--GO

VwBookType is referencing entity

BookType is referenced entity

referencing entity depends on referenced entity.

By default, this is Non-schema-bound dependency which is

a relationship between two entities

that does NOT prevent the referenced entity from being dropped or modified.

Before Modify or drop the referenced entity,

you have to ensure its referencing entity can still work properly.

--------------------------------------------------------------------------

2.

Schema-bound dependency V.S. Non-schema-bound dependency

Reference:

[https://technet.microsoft.com/en-us/library/ms345449(v=sql.105).aspx](https://technet.microsoft.com/en-us/library/ms345449%28v=sql.105%29.aspx)

2.0.

In sammary,

When Table1 depends on Table2

E.g.

--TypeId INT foreign key references BookType(TypeId)

By defualt, its will create Schema-bound dependency.

When Views and Functions depends on Table1.

By defualt, it will create Non-Schema-bound dependency.

But we can use WITH SCHEMABINDING clause

to create Schema-bound dependency.

However, for some reason, In SSMS

ObjectName --> Right click --> View dependencies

will always dispaly they are still Non-Schema-bound dependency

even if you use WITH SCHEMABINDING clause

But the truth is they are actually

a Schema-bound dependency when using WITH SCHEMABINDING.

---------------------------------------------------------

2.1.

Schema-bound dependency

2.1.1.

A schema-bound dependency is a relationship

between two entities that prevents the referenced entity

from being dropped or modified when the referencing entity exists.

2.1.2.

A schema-bound dependency is created when a view or

user-defined function is created by using the WITH SCHEMABINDING clause.

2.1.3.

A schema-bound dependency can also be created

when a table references another entity,

such as a Transact-SQL user-defined function,

user-defined type, or XML schema collection,

in a CHECK or DEFAULT constraint or

in the definition of a computed column.

---------------------------------------------------------

2.2.

Non-schema-bound dependency

A non-schema-bound dependency is a relationship between two entities

that does not prevent the referenced entity from being dropped or modified.

---------------------------------------------------------

2.3.

2.3.1.

E.g.1.

--CREATE VIEW VwBookType    --Referencing entity

--WITH SCHEMABINDING --Schema-bound dependency Keyword

--AS

--    SELECT  \*

--    FROM    BookType;    --Referenced entity

--GO

VwBookType is referencing entity

BookType is referenced entity

VwBookType depends on BookType with Non-Schema-bound dependency by default.

Thus, you can drop or modify BookType when Book exists.

but you use "SchemaBinding" keyword to create Schema-bound dependency.

--------------------------------

2.3.2.

E.g.

--CREATE FUNCTION fnGetBookById ( @Id int )    --Referencing entity

--RETURNS nvarchar(20)

--    WITH SchemaBinding --Schema-bound dependency Keyword

--AS

--    BEGIN

--        RETURN (

--                    SELECT BookName

--                    FROM dbo.Book    --Referenced entity

--                    WHERE BookId = @Id

--             );

--    END;

--GO

fnGetBookById is referencing entity

Book is referenced entity

fnGetBookById depends on Book with Non-Schema-bound dependency by default.

Thus, you can drop or modify Book when fnGetBookById exists.

but you use "SchemaBinding" keyword to create Schema-bound dependency.

--------------------------------

2.3.3.

E.g.

--CREATE TABLE Book    --Referencing entity

--(

--  BookId INT IDENTITY(1, 1)

--             PRIMARY KEY ,

--  BookName NVARCHAR(50) ,

--  TypeId INT foreign key references BookType(TypeId)    --Schema-bound dependency keyword, Referenced entity

--);

Book is referencing entity.

BookType is referenced entity.

Book depends on BookType with Schema-bound dependency by default.

Thus, you can not drop or modify BookType when Book exists.

--------------------------------------------------------------------------

3.

Find object dependencies

--sys.dm\_sql\_referencing\_entities V.S.

--sys.dm\_sql\_referenced\_entities V.S.

--Execute sp\_depends 'ObjectName'

---------------------------------------------------------

3.0.

In summary,

3.0.1.

Don't use sp\_depends.

We need to Specify an object using a two-part (schema\_name.object\_name) name

for both dynamic management functions,

sys.dm\_sql\_referencing\_entities and sys.dm\_sql\_referenced\_entities

to find object dependencies.

If Table1 depends on Table2,

both dynamic management functions will

Not display the object dependencie between tables.

The best way to find object dependencies is using SSMS.

In SSMS, Object --> right click --> View dependencies

The following explain the reason.

3.0.2.

We create BookType table,

and then create the view VwBookType which depends on BookType table.

Drop BookType table and recreated it.

Both sys.dm\_sql\_referencing\_entities and

sys.dm\_sql\_referenced\_entities are dynamic management functions

which will still work fine.

sp\_depends is NOT a dynamic management function which

will NOT work fine.

It means we know the view VwBookType still depends on BookType table.

But sp\_depends does not report this dependency,

as the BookType table is dropped and recreated.

---------------------------------------------------------

3.1.

--sys.dm\_sql\_referencing\_entities V.S.  Execute sp\_depends 'ObjectName'

Returns all referecing objects except Table objects that depend on dbo.Book table.

--------------------------------

3.1.1.

--SELECT  \*

--FROM    sys.dm\_sql\_referencing\_entities('dbo.Book', 'Object');

--GO

Returns all referecing objects except Table objects that depend on dbo.Book table.

--------------------------------

3.1.2.

--sp\_depends 'Book'

--GO

When parameter is a Table object,

then it will return all referecing objects except Table objects that depend on dbo.Book table.

---------------------------------------------------------

3.2.

--sys.dm\_sql\_referenced\_entities V.S.  Execute sp\_depends 'ObjectName'

Returns referenced entity objects except table objects,

which the stored procedure spGetAllBooks depends on.

--------------------------------

3.2.1.

--SELECT  \*

--FROM    sys.dm\_sql\_referenced\_entities('dbo.spGetAllBooks',

--                                       'Object');

--GO

Returns referenced entity objects except table objects,

which the stored procedure spGetAllBooks depends on.

--------------------------------

3.2.2.

sp\_depends 'spGetAllBooks'

--GO

When parameter is Not a Tabe Object,

then it will return return referenced entity objects except table objects,

which the stored procedure spGetAllBooks depends on.

---------------------------------------------------------

3.3.

--sys.dm\_sql\_referencing\_entities V.S.

--sys.dm\_sql\_referenced\_entities V.S.

--Execute sp\_depends 'ObjectName'

--------------------------------

3.3.1.

Both sys.dm\_sql\_referencing\_entities and

sys.dm\_sql\_referenced\_entities are dynamic management functions.

sp\_depends is NOT a dynamic management function.

E.g.

We create BookType table,

and then create the view VwBookType which depends on BookType table.

Drop BookType table and recreated it.

Both sys.dm\_sql\_referencing\_entities and

sys.dm\_sql\_referenced\_entities are dynamic management functions

which will still work fine.

sp\_depends is NOT a dynamic management function which

will NOT work fine.

It means we know the view VwBookType still depends on BookType table.

But sp\_depends does not report this dependency,

as the BookType table is dropped and recreated.

-------------------

3.3.1.1.

E.g.

--CREATE TABLE BookType

--(

--  TypeId INT IDENTITY(1, 1)

--             PRIMARY KEY ,

--  TypeName NVARCHAR(50)

--);

-------------------

3.3.1.2.

--CREATE VIEW VwBookType

--AS

--    SELECT  TypeId, TypeName

--    FROM    dbo.BookType;

-------------------

3.3.1.3.

--SELECT  \*

--FROM    sys.dm\_sql\_referenced\_entities('dbo.VwBookType',

--                                       'Object');

-------------------

--Execute sp\_depends 'VwBookType'

Both will return referenced entity objects except table objects,

which the View VwBookType depends on.

-------------------

3.3.1.4.

--SELECT  \*

--FROM    sys.dm\_sql\_referencing\_entities('dbo.BookType', 'Object');

OR

--Execute sp\_depends 'BookType'

Both will return all referecing objects except Table objects that depend on dbo.BookType table.

-------------------

3.3.1.5.

--Drop table BookType

-------------------

3.3.1.6.

--CREATE TABLE BookType

--(

--  TypeId INT IDENTITY(1, 1)

--             PRIMARY KEY ,

--  TypeName NVARCHAR(50)

--);

-------------------

3.3.1.7.

--SELECT  \*

--FROM    sys.dm\_sql\_referenced\_entities('dbo.VwBookType',

--                                       'Object');

OR

--Execute sp\_depends 'VwBookType'

dm\_sql\_referenced\_entities will return referenced entity objects except table objects,

which the View VwBookType depends on.

We know the view VwBookType still depends on BookType table.

But sp\_depends does not report this dependency,

as the BookType table is dropped and recreated.

-------------------

3.3.1.8.

--SELECT  \*

--FROM    sys.dm\_sql\_referencing\_entities('dbo.BookType', 'Object');

OR

--Execute sp\_depends 'BookType'

dm\_sql\_referencing\_entities will return all referecing objects except Table objects

that depend on dbo.BookType table.

We know the view VwBookType still depends on BookType table.

But sp\_depends does not report this dependency,

as the BookType table is dropped and recreated.

--------------------------------

3.3.1.

Both sys.dm\_sql\_referencing\_entities and

sys.dm\_sql\_referenced\_entities are dynamic management functions.

sp\_depends is NOT a dynamic management function.

We need to Specify an object using a two-part (schema\_name.object\_name) name

for both dynamic management functions, sys.dm\_sql\_referencing\_entities and sys.dm\_sql\_referenced\_entities

sp\_depends does not need a two-part (schema\_name.object\_name) name.

E.g.

--SELECT  \*

--FROM    sys.dm\_sql\_referenced\_entities('dbo.VwBookType',

--                                       'Object');

OR

--Execute sp\_depends 'VwBookType'

E.g.

--SELECT  \*

--FROM    sys.dm\_sql\_referencing\_entities('dbo.BookType', 'Object');

OR

--Execute sp\_depends 'BookType'

===========================================================================

1. Create Sample Data

--===================================================================================

--T037\_01\_CREATE Sample Data

--===================================================================================

-----------------------------------------------

--Create Table

--Drop Function if it exists.

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fnGetBookById2' ) )

    BEGIN

        DROP FUNCTION fnGetBookById2;

    END;

GO -- Run the previous command and begins new batch

--Drop View if it exists.

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'VwBookType2' ) )

    BEGIN

        DROP VIEW VwBookType2;

    END;

GO -- Run the previous command and begins new batch

--Drop Table if it exists.

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Book' ) )

    BEGIN

        TRUNCATE TABLE dbo.Book;

        DROP TABLE Book;

    END;

GO -- Run the previous command and begins new batch

--Drop Table if it exists.

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'BookType' ) )

    BEGIN

        TRUNCATE TABLE dbo.BookType;

        DROP TABLE BookType;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE BookType

(

  TypeId INT IDENTITY(1, 1)

             PRIMARY KEY ,

  TypeName NVARCHAR(50)

);

GO -- Run the previous command and begins new batch

CREATE TABLE Book

(

  BookId INT IDENTITY(1, 1)

             PRIMARY KEY ,

  BookName NVARCHAR(50) ,

  TypeId INT FOREIGN KEY REFERENCES BookType ( TypeId )

);

GO -- Run the previous command and begins new batch

-----------------------------------------------

--ReCreate stored procedure

--Drop stored procedure if it exists.

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spGetAllBooks' ) )

    BEGIN

        DROP PROCEDURE spGetAllBooks;

    END;

GO -- Run the previous command and begins new batch

--Drop stored procedure if it exists.

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spGetAllBooksWithType' ) )

    BEGIN

        DROP PROCEDURE spGetAllBooksWithType;

    END;

GO -- Run the previous command and begins new batch

CREATE PROCEDURE spGetAllBooks

AS

    BEGIN

        SELECT  \*

        FROM    Book;

    END;

GO -- Run the previous command and begins new batch

CREATE PROCEDURE spGetAllBooksWithType

AS

    BEGIN

        SELECT  b.BookId ,

                b.BookName ,

                b.TypeId ,

                t.TypeName

        FROM    dbo.Book b

                JOIN dbo.BookType t ON b.TypeId = t.TypeId;

    END;

GO -- Run the previous command and begins new batch

-----------------------------------------------

--Create VIEW

--Drop VIEW if it exists.

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'VwBookType' ) )

    BEGIN

        DROP VIEW VwBookType;

    END;

GO -- Run the previous command and begins new batch

CREATE VIEW VwBookType

--WITH SCHEMABINDING --Schema-bound dependency Keyword

AS

    SELECT  TypeId, TypeName

    FROM    dbo.BookType;

GO -- Run the previous command and begins new batch

CREATE VIEW VwBookType2

WITH SCHEMABINDING --Schema-bound dependency Keyword

AS

    SELECT  TypeId, TypeName

    FROM    dbo.BookType;

GO -- Run the previous command and begins new batch

-----------------------------------------------

--Create FUNCTION

--Drop FUNCTION if it exists.

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fnGetBookById' ) )

    BEGIN

        DROP FUNCTION fnGetBookById;

    END;

GO -- Run the previous command and begins new batch

CREATE FUNCTION fnGetBookById ( @Id int )

RETURNS nvarchar(20)

--    WITH SchemaBinding --Schema-bound dependency Keyword

AS

    BEGIN

        RETURN (

                    SELECT BookName

                    FROM dbo.Book

                    WHERE BookId = @Id

             );

    END;

GO -- Run the previous command and begins new batch

CREATE FUNCTION fnGetBookById2 ( @Id int )

RETURNS nvarchar(20)

    WITH SchemaBinding --Schema-bound dependency Keyword

AS

    BEGIN

        RETURN (

                    SELECT BookName

                    FROM dbo.Book

                    WHERE BookId = @Id

             );

    END;

GO -- Run the previous command and begins new batch

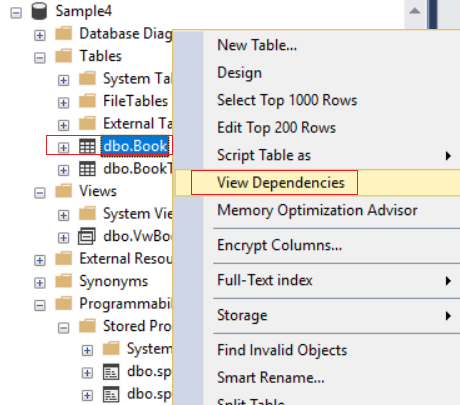
Graphical user interface, application

Description automatically generated

E.g.

To See the dbo.Book Table Dependencies

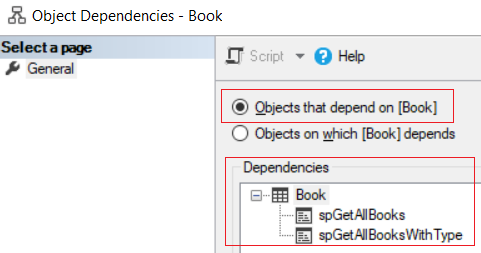
dbo.Book --> Right Click --> View Dependencies -->



There are some stored procedure is depending on this table.

If you want to modify the table structure or drop the table,

you need to double check the query of all these stored procedure.



Graphical user interface, application

Description automatically generated

===========================================================================

2. sys.dm\_sql\_referencing\_entities, dm\_sql\_referencing\_entities, sp\_depends

--===================================================================================

--T037\_02\_sys.dm\_sql\_referencing\_entities, dm\_sql\_referencing\_entities, sp\_depends

--===================================================================================

--Referencing entity V.S. Referenced entity

--Schema-bound dependency V.S. Non-schema-bound dependency

--sys.dm\_sql\_referencing\_entities V.S.

--sys.dm\_sql\_referenced\_entities V.S.

--Execute sp\_depends 'ObjectName'

2.1. sys.dm\_sql\_referencing\_entities V.S. Execute sp\_depends 'ObjectName'

--===================================================================================

--T037\_02\_01

--sys.dm\_sql\_referencing\_entities V.S.

--Execute sp\_depends 'ObjectName'

SELECT  \*

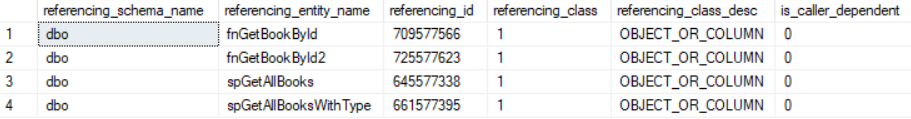
FROM    sys.dm\_sql\_referencing\_entities('dbo.Book', 'Object');

GO -- Run the previous command and begins new batch

/\*

returns all the objects except table objects that depend on dbo.Book table.

\*/



sp\_depends 'Book'

GO -- Run the previous command and begins new batch

/\*

When parameter is TableName,

then it will return all the objects except table objects that depend on Book table.

\*/



SELECT  \*

FROM    sys.dm\_sql\_referencing\_entities('dbo.BookType',

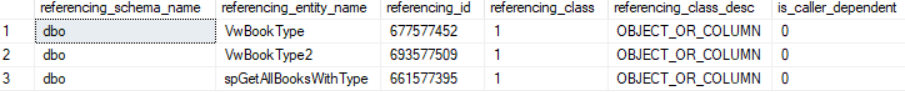
                                       'Object');

GO -- Run the previous command and begins new batch

/\*

returns all the objects except table objects that depend on dbo.BookType table.

\*/



sp\_depends 'BookType'

GO -- Run the previous command and begins new batch

/\*

When parameter is TableName,

then it will return all the objects except table objects that depend on BookType table.

\*/

Table

Description automatically generated

SELECT  \*

FROM    sys.dm\_sql\_referencing\_entities('dbo.spGetAllBooks',

                                       'Object');

GO -- Run the previous command and begins new batch

/\*

returns all the objects except table objects that depend on the stored procedure spGetAllBooks

\*/



2.2. sys.dm\_sql\_referenced\_entities V.S. Execute sp\_depends 'ObjectName'

--===================================================================================

--T037\_02\_02

--sys.dm\_sql\_referenced\_entities V.S.

--Execute sp\_depends 'ObjectName'

SELECT  \*

FROM    sys.dm\_sql\_referenced\_entities('dbo.BookType',

                                       'Object');

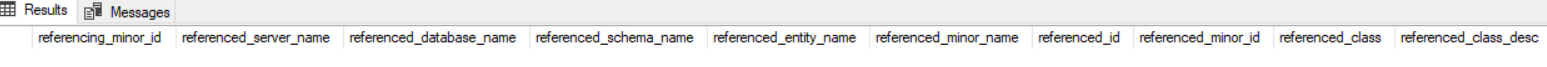
GO -- Run the previous command and begins new batch

/\*

Returns referenced entity objects except table objects,

which the table BookType depends on.

\*/



SELECT  \*

FROM    sys.dm\_sql\_referenced\_entities('dbo.Book',

                                       'Object');

GO -- Run the previous command and begins new batch

/\*

\*\*

Returns referenced entity objects except table objects,

which the table book depends on.

Book actually depends on BookType.

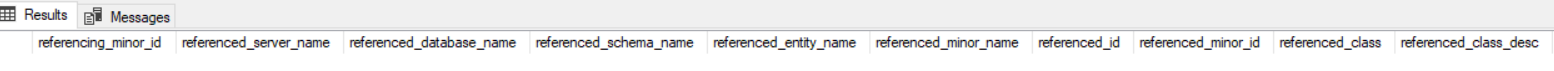
But both dynamic management functions will

Not display the object dependencie between tables.

The best way to find object dependencies is using SSMS.

In SSMS, Object --> right click --> View dependencies

\*/



SELECT  \*

FROM    sys.dm\_sql\_referenced\_entities('dbo.spGetAllBooks',

                                       'Object');

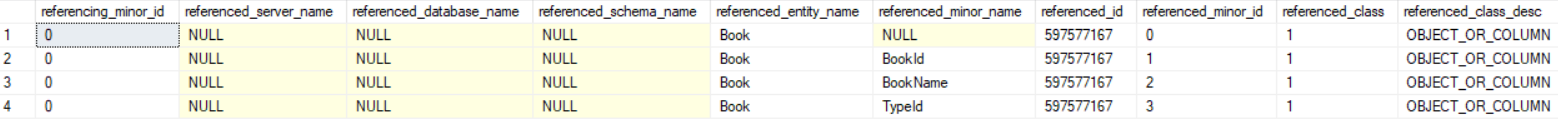
GO -- Run the previous command and begins new batch

/\*

Returns referenced entity objects except table objects,

which the stored procedure spGetAllBooks depends on.

\*/



sp\_depends 'spGetAllBooks'

GO -- Run the previous command and begins new batch

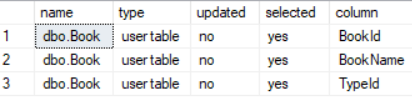
/\*

When parameter is Not TableName,

then it will return return referenced entity objects except table objects,

which the stored procedure spGetAllBooks depends on.

\*/



SELECT  \*

FROM    sys.dm\_sql\_referenced\_entities('dbo.spGetAllBooksWithType',

                                       'Object');

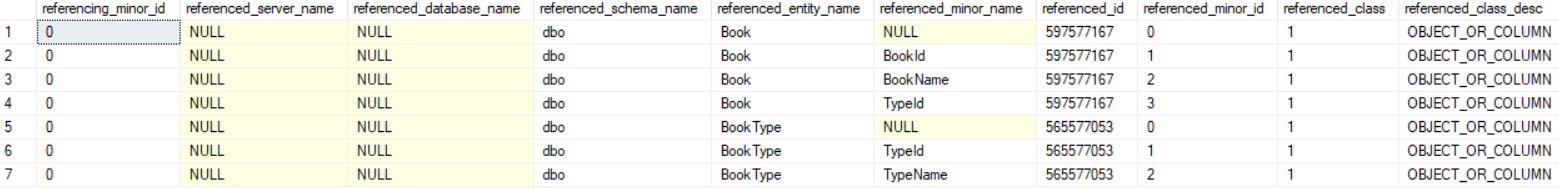
GO -- Run the previous command and begins new batch

/\*

Returns referenced entity objects except table objects,

which the stored procedure spGetAllBooksWithType depends on.

\*/



sp\_depends 'spGetAllBooksWithType'

GO -- Run the previous command and begins new batch

/\*

When parameter is Not TableName,

then it will return return referenced entity objects except table objects,

which the stored procedure spGetAllBooksWithType depends on.

\*/

Table

Description automatically generated

SELECT  \*

FROM    sys.dm\_sql\_referenced\_entities('dbo.VwBookType',

                                       'Object');

GO -- Run the previous command and begins new batch

/\*

Returns referenced entity objects except table object,

which the view VwBookType depends on.

\*/



sp\_depends 'VwBookType'

GO -- Run the previous command and begins new batch

/\*

When parameter is Not TableName,

then it will return return referenced entity objects except table object,

which the view VwBookType depends on.

\*/

Table

Description automatically generated

SELECT  \*

FROM    sys.dm\_sql\_referenced\_entities('dbo.VwBookType2',

                                       'Object');

GO -- Run the previous command and begins new batch

/\*

Returns referenced entity objects except table object,

which the view VwBookType2 depends on.

\*/



sp\_depends 'VwBookType2'

GO -- Run the previous command and begins new batch

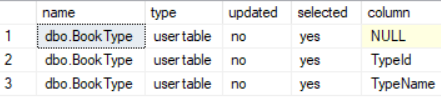
/\*

When parameter is Not TableName,

then it will return return referenced entity objects except table object,

which the view VwBookType2 depends on.

\*/



SELECT  \*

FROM    sys.dm\_sql\_referenced\_entities('dbo.fnGetBookById',

                                       'Object');

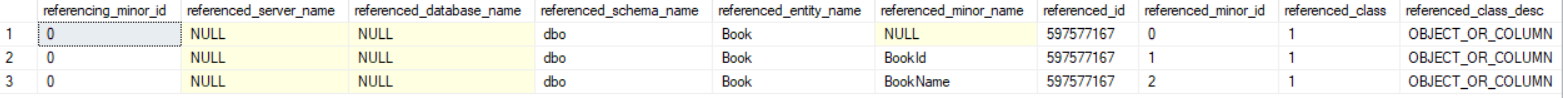
GO -- Run the previous command and begins new batch

/\*

Returns referenced entity objects except table object,

which the function fnGetBookById depends on.

\*/



sp\_depends 'fnGetBookById'

GO -- Run the previous command and begins new batch

/\*

When parameter is Not TableName,

then it will return return referenced entity objects except table object,

which the function fnGetBookById depends on.

\*/

Table

Description automatically generated

SELECT  \*

FROM    sys.dm\_sql\_referenced\_entities('dbo.fnGetBookById2',

                                       'Object');

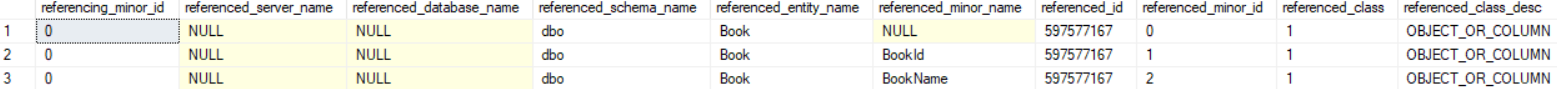
GO -- Run the previous command and begins new batch

/\*

Returns referenced entity objects except table object,

which the function fnGetBookById2 depends on.

\*/



sp\_depends 'fnGetBookById2'

GO -- Run the previous command and begins new batch

/\*

When parameter is Not TableName,

then it will return return referenced entity objects except table object,

which the function fnGetBookById2 depends on.

\*/

Table

Description automatically generated

2.3. Clean up

--===================================================================================

--T037\_02\_03

--Clean up

--If Function exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fnGetBookById2' ) )

    BEGIN

        DROP FUNCTION fnGetBookById2;

    END;

GO -- Run the previous command and begins new batch

--If View exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'VwBookType2' ) )

    BEGIN

        DROP VIEW VwBookType2;

    END;

GO -- Run the previous command and begins new batch

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Book' ) )

    BEGIN

        TRUNCATE TABLE dbo.Book;

        DROP TABLE Book;

    END;

GO -- Run the previous command and begins new batch

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'BookType' ) )

    BEGIN

        TRUNCATE TABLE dbo.BookType;

        DROP TABLE BookType;

    END;

GO -- Run the previous command and begins new batch

--If stored procedure exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spGetAllBooks' ) )

    BEGIN

        DROP PROCEDURE spGetAllBooks;

    END;

GO -- Run the previous command and begins new batch

--If stored procedure exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spGetAllBooksWithType' ) )

    BEGIN

        DROP PROCEDURE spGetAllBooksWithType;

    END;

GO -- Run the previous command and begins new batch

--If View exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'VwBookType' ) )

    BEGIN

        DROP VIEW VwBookType;

    END;

GO -- Run the previous command and begins new batch

--If function exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fnGetBookById' ) )

    BEGIN

        DROP FUNCTION fnGetBookById;

    END;

GO -- Run the previous command and begins new batch

===========================================================================

3. sys.dm\_sql\_referencing\_entities, dm\_sql\_referencing\_entities, sp\_depends

--===================================================================================

--T037\_03\_sys.dm\_sql\_referencing\_entities, dm\_sql\_referencing\_entities, sp\_depends

--===================================================================================

/\*

1.

Don't use sp\_depends.

We need to Specify an object using a two-part (schema\_name.object\_name) name

for both dynamic management functions,

sys.dm\_sql\_referencing\_entities and sys.dm\_sql\_referenced\_entities

to find object dependencies.

If Table1 depends on Table2,

both dynamic management functions will

Not display the object dependencie between tables.

The best way to find object dependencies is using SSMS.

In SSMS, Object --> right click --> View dependencies

The following explain the reason.

2.

We create BookType table,

and then create the view VwBookType which depends on BookType table.

Drop BookType table and recreated it.

Both sys.dm\_sql\_referencing\_entities and

sys.dm\_sql\_referenced\_entities are dynamic management functions

which will still work fine.

sp\_depends is NOT a dynamic management function which

will NOT work fine.

It means we know the view VwBookType still depends on BookType table.

But sp\_depends does not report this dependency,

as the BookType table is dropped and recreated.

\*/

3.1. Create Sample Data

--===================================================================================

--T037\_03\_01

--Create Sample Data

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'BookType' ) )

    BEGIN

        TRUNCATE TABLE dbo.BookType;

        DROP TABLE BookType;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE BookType

(

  TypeId INT IDENTITY(1, 1)

             PRIMARY KEY ,

  TypeName NVARCHAR(50)

);

GO -- Run the previous command and begins new batch

--If View exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'VwBookType' ) )

    BEGIN

        DROP VIEW VwBookType;

    END;

GO -- Run the previous command and begins new batch

CREATE VIEW VwBookType

AS

    SELECT  TypeId, TypeName

    FROM    dbo.BookType;

GO -- Run the previous command and begins new batch

3.2. Find dependencies

--===================================================================================

--T037\_03\_02

--Find dependencies

-------------------------------------------------------------------------------------

--T037\_03\_02\_01

--sys.dm\_sql\_referenced\_entities V.S. sp\_depends

SELECT  \*

FROM    sys.dm\_sql\_referenced\_entities('dbo.VwBookType',

                                       'Object');

Execute sp\_depends 'VwBookType'

GO -- Run the previous command and begins new batch

/\*

Both will return referenced entity objects except table objects,

which the View VwBookType depends on.

\*/

Graphical user interface, application, Word

Description automatically generated

-------------------------------------------------------------------------------------

--T037\_03\_02\_02

--sys.dm\_sql\_referencing\_entities V.S. sp\_depends

SELECT  \*

FROM    sys.dm\_sql\_referencing\_entities('dbo.BookType', 'Object');

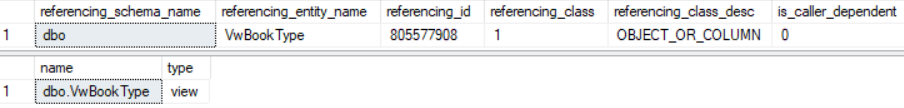
Execute sp\_depends 'BookType'

GO -- Run the previous command and begins new batch

/\*

Both will return all referecing objects except Table objects that depend on dbo.BookType table.

\*/



3.3. Drop the table and then re-create it.

--===================================================================================

--T037\_03\_03

--Drop the table and then re-create it.

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'BookType' ) )

    BEGIN

        TRUNCATE TABLE dbo.BookType;

        DROP TABLE BookType;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE BookType

(

  TypeId INT IDENTITY(1, 1)

             PRIMARY KEY ,

  TypeName NVARCHAR(50)

);

3.4. Find dependencies.

--===================================================================================

--T037\_03\_04

--Find dependencies

-------------------------------------------------------------------------------------

--T037\_03\_04\_01

--sys.dm\_sql\_referenced\_entities V.S. sp\_depends

SELECT  \*

FROM    sys.dm\_sql\_referenced\_entities('dbo.VwBookType',

                                       'Object');



Execute sp\_depends 'VwBookType'

GO -- Run the previous command and begins new batch



/\*

1.

--SELECT  \*

--FROM    sys.dm\_sql\_referenced\_entities('dbo.VwBookType',

--                                       'Object');

sys.dm\_sql\_referenced\_entities will return

referenced entity objects except table objects,

which the View VwBookType depends on.

2.

--Execute sp\_depends 'VwBookType'

Logic Error

--Object does not reference any object, and no objects reference it.

\*/

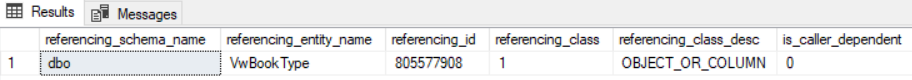
-------------------------------------------------------------------------------------

--T037\_03\_04\_02

--sys.dm\_sql\_referencing\_entities V.S. sp\_depends

SELECT  \*

FROM    sys.dm\_sql\_referencing\_entities('dbo.BookType', 'Object');



Execute sp\_depends 'BookType'

GO -- Run the previous command and begins new batch



/\*

1.

--SELECT  \*

--FROM    sys.dm\_sql\_referencing\_entities('dbo.BookType', 'Object');

sys.dm\_sql\_referencing\_entities will return

all referecing objects except Table objects

that depend on dbo.BookType table.

2.

--Execute sp\_depends 'BookType'

Logic Error

--Object does not reference any object, and no objects reference it.

\*/

3.5. Clean up

--===================================================================================

--T037\_03\_05

--Clean up

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'BookType' ) )

    BEGIN

        TRUNCATE TABLE dbo.BookType;

        DROP TABLE BookType;

    END;

GO -- Run the previous command and begins new batch

--If View exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'VwBookType' ) )

    BEGIN

        DROP VIEW VwBookType;

    END;

GO -- Run the previous command and begins new batch