(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
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'

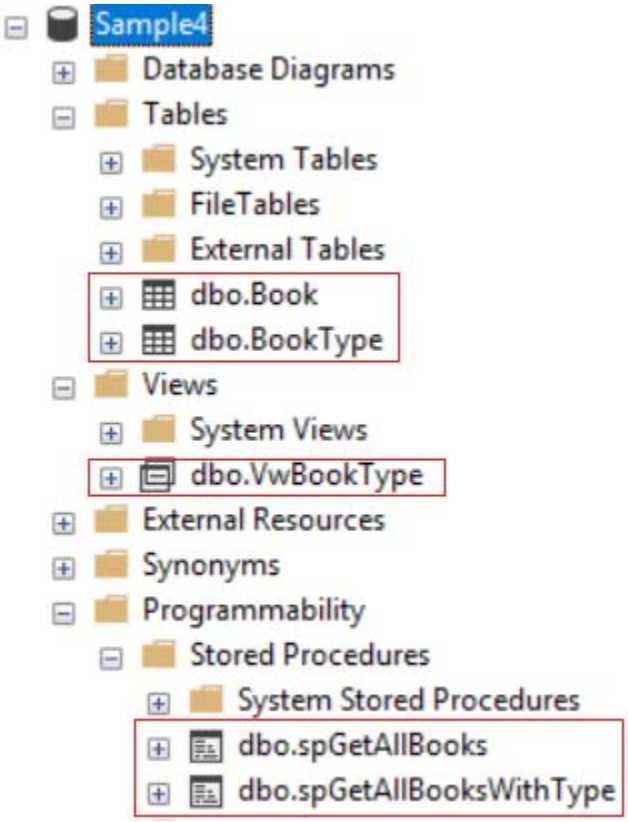
==========

1. Create Sample Data

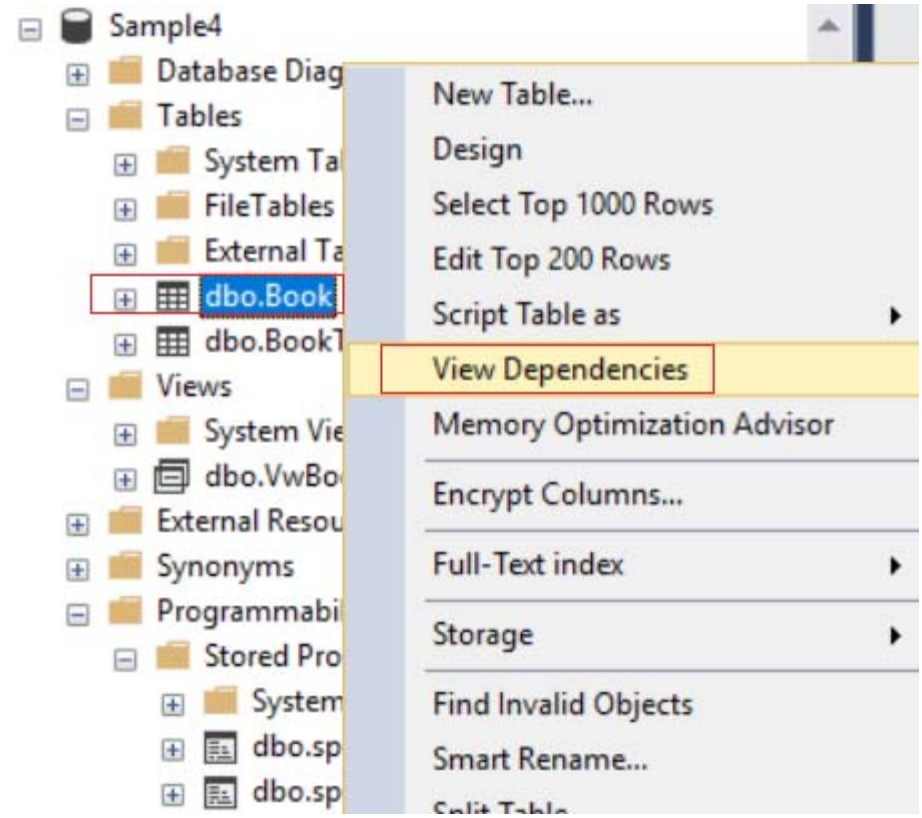
```
--============================================================================================
--T037_01_CREATE Sample Data
--============================================================================================
----------------------------------------------
--Create Table
--Drop Function 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;
```

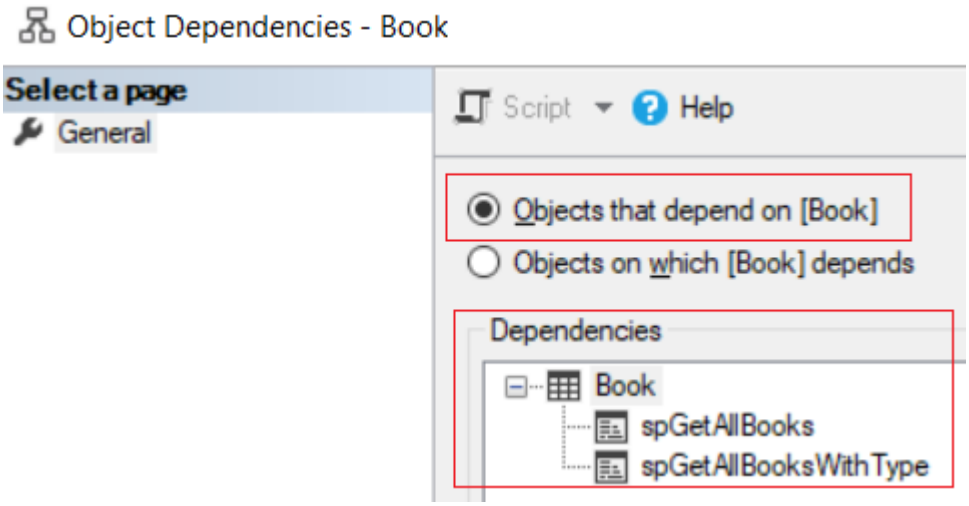
```
GO -- Run the previous command and begins new batch
CREATE VIEW VwBookType
--WITH SCHEMABINDING --Schema-bound dependency Keyword
AS
    SELECT  TypeId, TypeName
    FROM    dbo.BookType;
GO -- Run the previous command and begins new batch

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
```

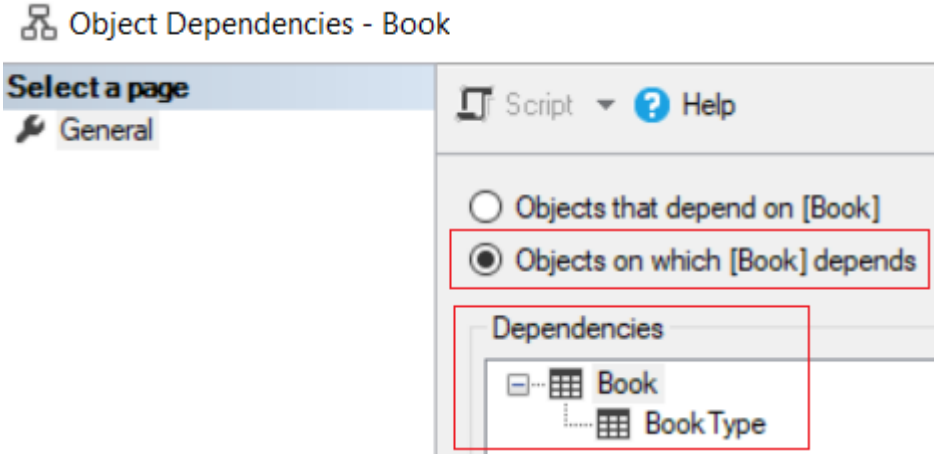


E.g.
To See the dbo.Book Table Dependencies
dbo.Book --> Right Click --> View Dependencies -->



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.





==========

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.
*/
```

	referencing_schema_name	referencing_entity_name	referencing_id	referencing_class	referencing_class_desc	is_caller_dependent
1	dbo	fnGetBookById	709577566	1	OBJECT_OR_COLUMN	0
2	dbo	fnGetBookById2	725577623	1	OBJECT_OR_COLUMN	0
3	dbo	spGetAllBooks	645577338	1	OBJECT_OR_COLUMN	0
4	dbo	spGetAllBooksWithType	661577395	1	OBJECT_OR_COLUMN	0

```
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.
*/
```

	name	type
1	dbo.fnGetBookById	scalar function
2	dbo.fnGetBookById2	scalar function
3	dbo.spGetAllBooks	stored procedure
4	dbo.spGetAllBooksWithType	stored procedure

```
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.
*/
```

	referencing_schema_name	referencing_entity_name	referencing_id	referencing_class	referencing_class_desc	is_caller_dependent
1	dbo	VwBookType	677577452	1	OBJECT_OR_COLUMN	0
2	dbo	VwBookType2	693577509	1	OBJECT_OR_COLUMN	0
3	dbo	spGetAllBooksWithType	661577395	1	OBJECT_OR_COLUMN	0

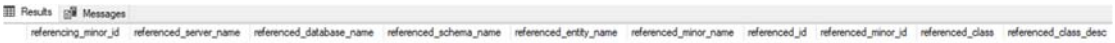
```
sp_depends 'BookType'
GO -- Run the previous command and begins new batch
/*
When parameter is TableName,
then it will return all the objects except table objects that depend on BookType table.
*/
```

	name	type
1	dbo.VwBookType	view
2	dbo.VwBookType2	view
3	dbo.spGetAllBooksWithType	stored procedure

```
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.
*/
```

	referencing_minor_id	referenced_server_name	referenced_database_name	referenced_schema_name	referenced_entity_name	referenced_minor_name	referenced_id	referenced_minor_id	referenced_class	referenced_class_desc
1	0	NULL	NULL	NULL	Book	NULL	597577167	0	1	OBJECT_OR_COLUMN
2	0	NULL	NULL	NULL	Book	BookId	597577167	1	1	OBJECT_OR_COLUMN
3	0	NULL	NULL	NULL	Book	BookName	597577167	2	1	OBJECT_OR_COLUMN
4	0	NULL	NULL	NULL	Book	TypeId	597577167	3	1	OBJECT_OR_COLUMN

```
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.
*/
```

	name	type	updated	selected	column
1	dbo.Book	user table	no	yes	BookId
2	dbo.Book	user table	no	yes	BookName
3	dbo.Book	user table	no	yes	TypeId

```
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.
*/
```

	referencing_minor_id	referenced_server_name	referenced_database_name	referenced_schema_name	referenced_entity_name	referenced_minor_name	referenced_id	referenced_minor_id	referenced_class	referenced_class_desc
1	0	NULL	NULL	dbo	Book	NULL	597577167	0	1	OBJECT_OR_COLUMN
2	0	NULL	NULL	dbo	Book	BookId	597577167	1	1	OBJECT_OR_COLUMN
3	0	NULL	NULL	dbo	Book	BookName	597577167	2	1	OBJECT_OR_COLUMN
4	0	NULL	NULL	dbo	Book	TypeId	597577167	3	1	OBJECT_OR_COLUMN
5	0	NULL	NULL	dbo	BookType	NULL	565577053	0	1	OBJECT_OR_COLUMN
6	0	NULL	NULL	dbo	BookType	TypeId	565577053	1	1	OBJECT_OR_COLUMN
7	0	NULL	NULL	dbo	BookType	TypeName	565577053	2	1	OBJECT_OR_COLUMN

```
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.
*/
```

	name	type	updated	selected	column
1	dbo.BookType	user table	no	yes	TypeId
2	dbo.BookType	user table	no	yes	TypeName
3	dbo.Book	user table	no	yes	BookId
4	dbo.Book	user table	no	yes	BookName
5	dbo.Book	user table	no	yes	TypeId

```
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.
*/
```

	referencing_minor_id	referenced_server_name	referenced_database_name	referenced_schema_name	referenced_entity_name	referenced_minor_name	referenced_id	referenced_minor_id	referenced_class	referenced_class_desc
1	0	NULL	NULL	dbo	BookType	NULL	565577053	0	1	OBJECT_OR_COLUMN
2	0	NULL	NULL	dbo	BookType	TypeId	565577053	1	1	OBJECT_OR_COLUMN
3	0	NULL	NULL	dbo	BookType	TypeName	565577053	2	1	OBJECT_OR_COLUMN

```
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.
*/
```

	name	type	updated	selected	column
1	dbo.BookType	user table	no	yes	TypeId
2	dbo.BookType	user table	no	yes	TypeName

```
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.
*/
```

	referencing_minor_id	referenced_server_name	referenced_database_name	referenced_schema_name	referenced_entity_name	referenced_minor_name	referenced_id	referenced_minor_id	referenced_class	referenced_class_desc
1	0	NULL	NULL	dbo	BookType	NULL	565577053	0	1	OBJECT_OR_COLUMN
2	0	NULL	NULL	dbo	BookType	TypeId	565577053	1	1	OBJECT_OR_COLUMN
3	0	NULL	NULL	dbo	BookType	TypeName	565577053	2	1	OBJECT_OR_COLUMN

```
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.
*/
```

	name	type	updated	selected	column
1	dbo.BookType	user table	no	yes	NULL
2	dbo.BookType	user table	no	yes	TypeId
3	dbo.BookType	user table	no	yes	TypeName

```
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.
*/
```

	referencing_minor_id	referenced_server_name	referenced_database_name	referenced_schema_name	referenced_entity_name	referenced_minor_name	referenced_id	referenced_minor_id	referenced_class	referenced_class_desc
1	0	NULL	NULL	dbo	Book	NULL	597577167	0	1	OBJECT_OR_COLUMN
2	0	NULL	NULL	dbo	Book	BookId	597577167	1	1	OBJECT_OR_COLUMN
3	0	NULL	NULL	dbo	Book	BookName	597577167	2	1	OBJECT_OR_COLUMN

```
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.
*/
```

	name	type	updated	selected	column
1	dbo.Book	user table	no	yes	BookId
2	dbo.Book	user table	no	yes	BookName

```
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.
*/
```

	referencing_minor_id	referenced_server_name	referenced_database_name	referenced_schema_name	referenced_entity_name	referenced_minor_name	referenced_id	referenced_minor_id	referenced_class	referenced_class_desc
1	0	NULL	NULL	dbo	Book	NULL	597577167	0	1	OBJECT_OR_COLUMN
2	0	NULL	NULL	dbo	Book	BookId	597577167	1	1	OBJECT_OR_COLUMN
3	0	NULL	NULL	dbo	Book	BookName	597577167	2	1	OBJECT_OR_COLUMN

```
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.
*/
```

	name	type	updated	selected	column
1	dbo.Book	user table	no	yes	NULL
2	dbo.Book	user table	no	yes	BookId
3	dbo.Book	user table	no	yes	BookName

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.
*/
```

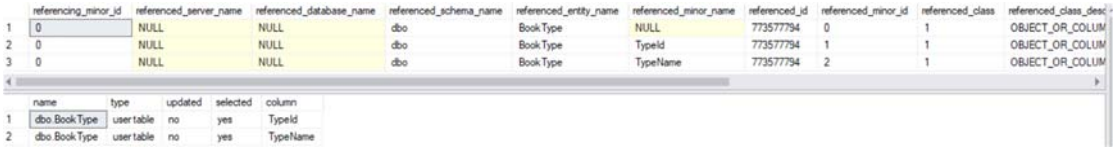
3.1. Create Sample Data

```
--============================================================================
--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.
*/
```



	referencing_minor_id	referenced_server_name	referenced_database_name	referenced_schema_name	referenced_entity_name	referenced_minor_name	referenced_id	referenced_minor_id	referenced_class	referenced_class_desc
1	0	NULL	NULL	dbo	BookType	NULL	773577794	0	1	OBJECT_OR_COLUM
2	0	NULL	NULL	dbo	BookType	TypeId	773577794	1	1	OBJECT_OR_COLUM
3	0	NULL	NULL	dbo	BookType	TypeName	773577794	2	1	OBJECT_OR_COLUM

	name	type	updated	selected	column
1	dbo.BookType	user table	no	yes	TypeId
2	dbo.BookType	user table	no	yes	TypeName

```
--------------------------------------------------------------------------------
--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.
*/
```

	referencing_schema_name	referencing_entity_name	referencing_id	referencing_class	referencing_class_desc	is_caller_dependent
1	dbo	VwBookType	805577908	1	OBJECT_OR_COLUMN	0

	name	type
1	dbo.VwBookType	view

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');
```

	referencing_minor_id	referenced_server_name	referenced_database_name	referenced_schema_name	referenced_entity_name	referenced_minor_name	referenced_id	referenced_minor_id	referenced_class	referenced_class_desc
1	0	NULL	NULL	dbo	BookType	NULL	821577965	0	1	OBJECT_OR_COLUMN
2	0	NULL	NULL	dbo	BookType	TypeId	821577965	1	1	OBJECT_OR_COLUMN
3	0	NULL	NULL	dbo	BookType	TypeName	821577965	2	1	OBJECT_OR_COLUMN

```
Execute sp_depends 'VwBookType'
GO -- Run the previous command and begins new batch
```

Messages

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

```
/*
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');
```

Results Messages

	referencing_schema_name	referencing_entity_name	referencing_id	referencing_class	referencing_class_desc	is_caller_dependent
1	dbo	VwBookType	805577908	1	OBJECT_OR_COLUMN	0

```
Execute sp_depends 'BookType'
GO -- Run the previous command and begins new batch
```

Messages

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

```
/*
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
```