Лабораторна робота № 2-3

ДОСЛІДЖЕННЯ СЕРВЕРІВ ТА РЕВЕРС-ІНЖЕНЕРІЯ БАЗ ДАНИХ

Мета роботи: дослідження системного каталогу та його представлень для отримання інформації про існуючі сервери та бази даних MS SQL Server.

Завдання на лабораторну роботу

Виконати дослідження інформаційної інфраструктури за наступним планом:

СЕРВЕРА

1. Інформація о серверах.
2. Список усіх БД

БАЗИ ДАНИХ

3. Розташування файлів БД

ТАБЛИЦІ

4. Кількість записів у таблиці
5. Пошук кластерних індексів и куч (таблиць без кластерних індексів)
6. Активність в таблице

ПРЕДСТАВЛЕННЯ

ПРОЦЕДУРИ, ЩО ЗБЕРІГАЮТЬСЯ

ФУНКЦІЇ

7. Стовпці
8. Значення за замовченням
9. Стовпці, що обчислюються
10. Стовпці identity
11. Ключі та індекси
12. Зовнішні ключі
13. Залежності

СЕРВЕРА

1. Інформація о серверах.

```
-- Имена сервера и экземпляра
Select @@SERVERNAME as [Server\Instance];
-- версия SQL Server
Select @@VERSION as SQLServerVersion;
-- экземпляр SQL Server
Select @@ServiceName AS ServiceInstance;
 -- Текущая БД (БД, в контексте которой выполняется запрос)
Select DB_NAME() AS CurrentDB_Name;
```

Як довго ваш SQL Server працює після останнього перезапуску? Пам'ятайте, що системна база даних tempdb заново створюється при кожному перезапуску SQL Server. Ось один з методів визначення **часу останнього перезапуску сервера.**

```
SELECT  @@Servername AS ServerName ,
       create_date AS  ServerStarted ,
```

```
    DATEDIFF(s, create_date, GETDATE()) / 86400.0 AS DaysRunning ,
    DATEDIFF(s, create_date, GETDATE()) AS SecondsRunnig
FROM   sys.databases
WHERE  name = 'tempdb';
GO
```

2. Список усіх БД

На будь-якому сервері є чотири або п'ять системних баз даних (master, model, msdb, tempdb та distribution (для реплікації)). Ви, напевно, захочете виключити ці бази в наступних запитах. Дуже просто побачити список баз даних у SSMS, але ці запити будуть нашими «будівельними блоками» для складніших запитів.

Є кілька шляхів для отримання списку всіх баз даних на T-SQL і нижче ви побачите деякі з них. Кожен метод повертає схожий результат, але з деякими відмінностями.

```
EXEC sp_helpdb;
--OR
EXEC sp_Databases;
--OR
SELECT  @@SERVERNAME AS Server ,
    name AS DBName ,
    recovery_model_Desc AS RecoveryModel ,
    Compatibility_level AS CompatiblityLevel ,
    create_date ,
    state_desc
FROM   sys.databases
ORDER BY Name;
--OR
SELECT  @@SERVERNAME AS Server ,
    d.name AS DBName ,
    create_date ,
    compatibility_level ,
    m.physical_name AS FileName
FROM   sys.databases d
    JOIN sys.master_files m ON d.database_id = m.database_id
WHERE  m.[type] = 0 -- data files only
ORDER BY d.name;
GO
```

БАЗЫ ДАННЫХ

Можна зібрати інформацію про об'єкти у всіх БД, використовуючи різні представлення каталогу та dmv. Більшість із запитів, поданих у цьому розділі, дивляться «всередину» лише однієї БД, тому не забувайте вибирати потрібну БД у SSMS або за допомогою команди use database. Також пам'ятайте, що ви завжди можете подивитися в контексті, якої БД буде виконаний запит, за допомогою

select db_name().

Системна таблиця sys.objects одна з ключових для збору інформації про об'єкти, що становлять вашу модель даних.

```
-- В прикладі в WHERE U - таблиці
USE MyDatabase;
GO
SELECT  *
FROM    sys.objects
WHERE  type = 'U';
```

Завдання. Перелік та розшифровку значень type оформити у вигляді таблиці, дивіться документацію на sys.objects у MSDN sys.objects.type

Інші представлення каталогу, такі як sys.tables та sys.views, звертаються до sys.objects та надають інформацію про конкретний тип об'єктів. З цими представленнями плюс функцією OBJECTPROPERTY, ми можемо отримати величезну кількість інформації по кожному з об'єктів, що складають нашу схему БД.

3. Розташування файлів баз даних

Фізичне розташування вибраної БД, включаючи основний файл даних (mdf), та файл журналу транзакцій (ldf), можуть бути отримані за допомогою цих запитів.

```
EXEC sp_Helpfile;
--OR
SELECT  @@Servername AS Server ,
        DB_NAME() AS DB_Name ,
        File_id ,
        Type_desc ,
        Name ,
        LEFT(Physical_Name, 1) AS Drive ,
        Physical_Name ,
        RIGHT(physical_name, 3) AS Ext ,
        Size ,
        Growth
FROM    sys.database_files
ORDER BY File_id;
GO
```

ТАБЛИЦІ

Звичайно, Object Explorer у SSMS показує повний список таблиць у вибраній БД, але частину інформації за допомогою GUI отримати складніше, ніж за допомогою скриптів.

Стандарт ANSI передбачає звернення до представлення INFORMATION_SCHEMA, але вони не нададуть інформацію про об'єкти, які не є частиною стандарту (такі як тригери, extended procedures тощо), тому краще використовувати уявлення каталогу SQL Server.

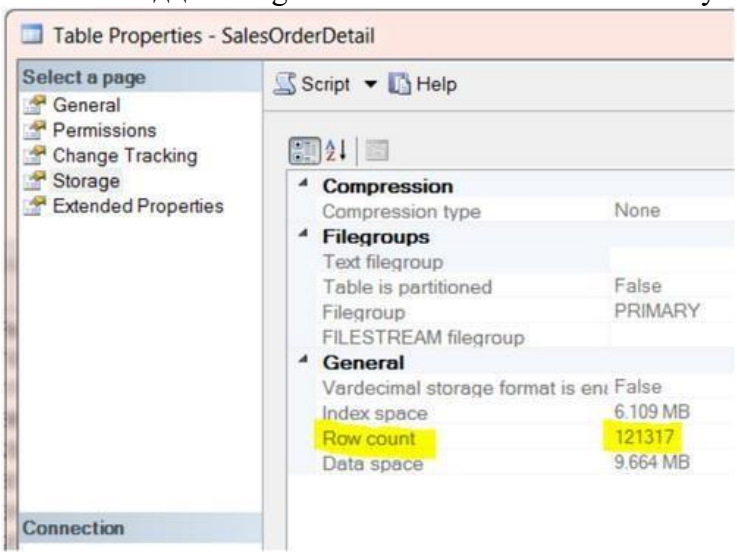
```
EXEC sp_tables; -- Пам'ятайте, що цей метод поверне і таблиці, і представлення
SELECT  @@Servername AS ServerName ,
        TABLE_CATALOG ,
        TABLE_SCHEMA ,
        TABLE_NAME
FROM    INFORMATION_SCHEMA.TABLES
WHERE TABLE_TYPE = 'BASE TABLE'
ORDER BY TABLE_NAME ;
--OR
```

```
SELECT  @@Servername AS ServerName ,
        DB_NAME() AS DBName ,
        o.name AS 'TableName' ,
        o.[Type] ,
        o.create_date
FROM    sys.objects o
WHERE o.Type = 'U' -- User table
ORDER BY o.name;
--OR
SELECT  @@Servername AS ServerName ,
        DB_NAME() AS DBName ,
        t.Name AS TableName,
        t.[Type],
        t.create_date
FROM    sys.tables t
ORDER BY t.Name;
GO
```

4. Кількість записів у таблиці

Якщо ви нічого не знаєте про таблицю, то всі таблиці однаково важливі. Чим більше ви дізнаєтеся про таблиці, тим більше ви їх поділите на умовно важливіші і умовно менш важливі. У цілому, таблиці з великою кількістю записів частіше надають серйозний вплив на продуктивність.

У SSMS ми можемо натиснути правою кнопкою миші будь-яку таблицю, відкрити властивості вкладці Storage і побачити кількість записів у таблиці.



Досить важко зібрати вручну цю інформацію про всі таблиці. Знову ж таки, якщо ми пишемо SELECT COUNT(*) FROM TABLENAME для кожної таблиці, нам доведеться дуже багато друкувати.

Набагато зручніше використовувати T-SQL для створення скрипта. Скрипт, наведений нижче, створює набір інструкцій T-SQL для отримання кількості рядків у кожній таблиці поточної бази даних.

```
SELECT  'Select '' + DB_NAME() + '.' + SCHEMA_NAME(SCHEMA_ID) + '.'
```

```
      + LEFT(o.name, 128) + '" as DBName, count(*) as Count From ' +
SCHEMA_NAME(SCHEMA_ID) + '.' + o.name
      + ';' AS ' Script generator to get counts for all tables'
FROM    sys.objects o
WHERE  o.[type] = 'U'
ORDER BY o.name;
```

Sp_msforeachtable – це недокументована функція, яка проходить по всіх таблицях в БД і виконує запит, підставляючи замість '?' ім'я поточної таблиці. Також існує схожа функція sp_msforeachdb, що працює на рівні баз даних.

Відомо кілька проблем із цією недокументованою функцією, наприклад, використання спецсимволів в іменах об'єктів. Тобто, якщо ім'я таблиці або бази даних містить знак '-', процедура, що зберігається, листинг якої нижче, завершиться з помилкою.

```
CREATE TABLE #rowcount
   ( Tablename VARCHAR(128) ,
     Rowcnt INT );
EXEC sp_MSforeachtable ' insert into #rowcount select "?", count(*) from ?'
SELECT  *
FROM    #rowcount
ORDER BY Tablename ,
      Rowcnt;
DROP TABLE #rowcount;
```

5. Пошук кластерних індексів та куч (таблиць без кластерних індексів)

Всі попередні методи використовували COUNT(*), який повільно працює, якщо в таблиці більше 500K записів. Найшвидший спосіб отримання кількості записів у таблиці – отримувати кількість записів у кластерному індексі чи кучі. Пам'ятайте, що хоч цей метод і найшвидший, MS каже, що інформація про кількість записів індексу та реальну кількість рядків у таблиці може не співпадати через те, що на оновлення інформації потрібний час. У більшості випадків ці значення або однакові, або дуже близькі і незабаром стануть однаковими.

```
      -- Найшвидший шлях отримання кількості записів
      -- Hint: отримуйте з індексу, а не таблиць
      SELECT  @@ServerName AS Server ,
      DB_NAME() AS DBName ,
      OBJECT_SCHEMA_NAME(p.object_id) AS SchemaName ,
      OBJECT_NAME(p.object_id) AS TableName ,
      i.Type_Desc ,
      i.Name AS IndexUsedForCounts ,
      SUM(p.Rows) AS Rows
FROM    sys.partitions p
      JOIN sys.indexes i ON i.object_id = p.object_id
                        AND i.index_id = p.index_id
WHERE   i.type_desc IN ( 'CLUSTERED', 'HEAP' )
                       -- This is key (1 index per table)
      AND OBJECT_SCHEMA_NAME(p.object_id) <> 'sys'
GROUP BY p.object_id ,
```

```
        i.type_desc ,
        i.Name
ORDER BY SchemaName ,
        TableName;
-- OR
-- Схожий метод отримання кількості записів, але з використанням DMV dm_db_partition_stats
SELECT  @@ServerName AS ServerName ,
        DB_NAME() AS DBName ,
        OBJECT_SCHEMA_NAME(ddps.object_id) AS SchemaName ,
        OBJECT_NAME(ddps.object_id) AS TableName ,
        i.Type_Desc ,
        i.Name AS IndexUsedForCounts ,
        SUM(ddps.row_count) AS Rows
FROM    sys.dm_db_partition_stats ddps
        JOIN sys.indexes i ON i.object_id = ddps.object_id
                              AND i.index_id = ddps.index_id
WHERE  i.type_desc IN ( 'CLUSTERED', 'HEAP' )
                              -- This is key (1 index per table)
        AND OBJECT_SCHEMA_NAME(ddps.object_id) <> 'sys'
GROUP BY ddps.object_id ,
        i.type_desc ,
        i.Name
ORDER BY SchemaName ,
        TableName;
GO
```

Робота з кучами – це як робота з плоским файлом замість бази даних. Якщо ви бажаєте гарантовано отримувати повне сканування таблиці під час виконання будь-якого запиту, використовуйте кучи. Зазвичай я рекомендую додавати primary key до всіх таблиць-куч.

```
-- Кучи (метод 1)
SELECT  @@Servername AS ServerName ,
        DB_NAME() AS DBName ,
        t.Name AS HeapTable ,
        t.Create_Date
FROM    sys.tables t
        INNER JOIN sys.indexes i ON t.object_id = i.object_id
                                AND i.type_desc = 'HEAP'
ORDER BY t.Name
--OR
-- Кучи (Метод 2)
SELECT  @@Servername AS ServerName ,
        DB_NAME() AS DBName ,
        t.Name AS HeapTable ,
        t.Create_Date
FROM    sys.tables t
WHERE    OBJECTPROPERTY(OBJECT_ID, 'TableHasClustIndex') = 0
ORDER BY t.Name;
--OR
```

```
-- Кучи (Метод 3) + количество записей
SELECT  @@ServerName AS Server ,
        DB_NAME() AS DBName ,
        OBJECT_SCHEMA_NAME(ddps.object_id) AS SchemaName ,
        OBJECT_NAME(ddps.object_id) AS TableName ,
        i.Type_Desc ,
        SUM(ddps.row_count) AS Rows
FROM    sys.dm_db_partition_stats AS ddps
        JOIN sys.indexes i ON i.object_id = ddps.object_id
                              AND i.index_id = ddps.index_id
WHERE  i.type_desc = 'HEAP'
        AND OBJECT_SCHEMA_NAME(ddps.object_id) <> 'sys'
GROUP BY ddps.object_id ,
        i.type_desc
ORDER BY TableName;
```

6. Активність у таблиці

При роботах з оптимізації продуктивності, дуже важливо знати, які таблиці активно читаються, а в які йде активний запис. Раніше ми дізналися скільки записів у наших таблицях, зараз подивимося, як часто в них пишуть і читають.

Пам'ятайте, що ця інформація з dmv очищається при кожному перезапуску SQL Server. Чим довше сервер працює, тим надійніша статистика. Я почуваюся набагато впевненіше зі статистикою, зібраною за 30 днів, ніж зі статистикою, зібраною за тиждень.

```
-- Чтение/запись таблицы
-- Кучи не рассматриваются, у них нет индексов
-- Только те таблицы, к которым обращались после запуска SQL Server
SELECT  @@ServerName AS ServerName ,
        DB_NAME() AS DBName ,
        OBJECT_NAME(ddius.object_id) AS TableName ,
        SUM(ddius.user_seeks + ddius.user_scans + ddius.user_lookups)
                                                AS  Reads ,
        SUM(ddius.user_updates) AS Writes ,
        SUM(ddius.user_seeks + ddius.user_scans + ddius.user_lookups
          + ddius.user_updates) AS [Reads&Writes] ,
        ( SELECT    DATEDIFF(s, create_date, GETDATE()) / 86400.0
          FROM      master.sys.databases
          WHERE     name = 'tempdb'
        ) AS SampleDays ,
        ( SELECT    DATEDIFF(s, create_date, GETDATE()) AS SecoundsRunnig
          FROM      master.sys.databases
          WHERE     name = 'tempdb'
        ) AS SampleSeconds
FROM    sys.dm_db_index_usage_stats ddius
        INNER JOIN sys.indexes i ON ddius.object_id = i.object_id
                                AND i.index_id = ddius.index_id
WHERE    OBJECTPROPERTY(ddius.object_id, 'IsUserTable') = 1
```

```
      AND ddius.database_id = DB_ID()
GROUP BY OBJECT_NAME(ddius.object_id)
ORDER BY [Reads&Writes] DESC;
GO
```

Набагато більш просунута версія цього запиту представлена курсором, що збирає інформацію про всі таблиці всіх баз даних на сервері. Курсори мають невисоку продуктивність, але переміщення різними базами даних – це відмінне застосування їм.

```
-- Операции чтения и записи
-- Кучи пропущены, у них нет индексов
-- Только таблицы, использовавшиеся после перезапуска SQL Server
-- В запросе используется курсор для получения информации во всех БД
-- Единый отчёт, хранится в tempdb
DECLARE DBNameCursor CURSOR
FOR
    SELECT  Name
   FROM    sys.databases
   WHERE    Name NOT IN ( 'master', 'model', 'msdb', 'tempdb',
                          'distribution' )
   ORDER BY Name;
DECLARE @DBName NVARCHAR(128)
DECLARE @cmd VARCHAR(4000)
IF OBJECT_ID(N'tempdb..TempResults') IS NOT NULL
    BEGIN
      DROP TABLE tempdb..TempResults
   END
CREATE TABLE tempdb..TempResults
   (
     ServerName NVARCHAR(128) ,
     DBName NVARCHAR(128) ,
     TableName NVARCHAR(128) ,
     Reads INT ,
     Writes INT ,
     ReadsWrites INT ,
     SampleDays DECIMAL(18, 8) ,
     SampleSeconds INT
   )
OPEN DBNameCursor
FETCH NEXT FROM DBNameCursor INTO @DBName
WHILE @@fetch_status = 0
   BEGIN
--------------------------------
-- Print @DBName
      SELECT  @cmd = 'Use ' + @DBName + '; '
      SELECT @cmd = @cmd + ' Insert Into tempdb..TempResults
SELECT @@ServerName AS ServerName,
DB_NAME() AS DBName,
object_name(ddius.object_id) AS TableName ,
```

```
SUM(ddius.user_seeks
+ ddius.user_scans
+ ddius.user_lookups) AS Reads,
SUM(ddius.user_updates) as Writes,
SUM(ddius.user_seeks
+ ddius.user_scans
+ ddius.user_lookups
+ ddius.user_updates) as ReadsWrites,
(SELECT datediff(s,create_date, GETDATE()) / 86400.0
FROM sys.databases WHERE name = "tempdb") AS SampleDays,
(SELECT datediff(s,create_date, GETDATE())
FROM sys.databases WHERE name = "tempdb") as SampleSeconds
FROM sys.dm_db_index_usage_stats ddius
INNER JOIN sys.indexes i
ON ddius.object_id = i.object_id
AND i.index_id = ddius.index_id
WHERE objectproperty(ddius.object_id,"IsUserTable") = 1 --True
AND ddius.database_id = db_id()
GROUP BY object_name(ddius.object_id)
ORDER BY ReadsWrites DESC;'
--PRINT @cmd
      EXECUTE (@cmd)
----------------------------------
      FETCH NEXT FROM DBNameCursor INTO @DBName
   END
CLOSE DBNameCursor
DEALLOCATE DBNameCursor
SELECT  *
FROM    tempdb..TempResults
ORDER BY DBName ,
      TableName;
--DROP TABLE tempdb..TempResults;
```

ПРЕДСТАВЛЕННЯ

У SQL Server, у деяких випадках, ми можемо оновлювати дані з використанням представлення. Щоб отримати представлення «лише для читання», можна використовувати SELECT DISTINCT під час його створення. Дані «через» представлення можна змінювати лише у тому випадку, якщо кожному рядку представлення відповідає лише один рядок у «базовій» таблиці. Будь-яке представлення, яке відповідає цьому критерію, тобто побудоване на кількох таблицях, або з використанням угруповань, агрегатних функцій та обчислень, буде доступним лише для читання.

```
SELECT  @@Servername AS ServerName ,
      DB_NAME() AS DBName ,
      o.name AS ViewName ,
      o.[Type] ,
      o.create_date
```

```
FROM    sys.objects o
WHERE o.[Type] = 'V' -- View
ORDER BY o.NAME
--OR
SELECT  @@Servername AS ServerName ,
        DB_NAME() AS DBName ,
        Name AS ViewName ,
        create_date
FROM    sys.Views
ORDER BY Name
--OR
SELECT  @@Servername AS ServerName ,
        TABLE_CATALOG ,
        TABLE_SCHEMA ,
        TABLE_NAME ,
        TABLE_TYPE
FROM    INFORMATION_SCHEMA.TABLES
WHERE  TABLE_TYPE = 'VIEW'
ORDER BY TABLE_NAME
--OR
-- CREATE VIEW Code
SELECT  @@Servername AS ServerName ,
        DB_NAME() AS DB_Name ,
        o.name AS 'ViewName' ,
        o.Type ,
        o.create_date ,
        sm.[DEFINITION] AS 'View script'
FROM    sys.objects o
        INNER JOIN sys.sql_modules sm ON o.object_id = sm.OBJECT_ID
WHERE  o.Type = 'V' -- View
ORDER BY o.NAME;
GO
```

ПРОЦЕДУРИ, ЩО ЗБЕРІГАЮТЬСЯ

Процедури, що зберігаються – це група скриптів, які компілюються на єдиний план виконання. Ми можемо використовувати представлення каталогу, щоб визначити, які процедури створені, які дії вони виконують і над якими таблицями.

```
-- Хранимые процедуры
SELECT  @@Servername AS ServerName ,
        DB_NAME() AS DBName ,
        o.name AS StoredProcedureName ,
        o.[Type] ,
        o.create_date
FROM    sys.objects o
WHERE o.[Type] = 'P' -- Stored Procedures
ORDER BY o.name
--OR
```

```
-- Дополнительная информация о ХП
SELECT  @@Servername AS ServerName ,
        DB_NAME() AS DB_Name ,
        o.name AS 'ViewName' ,
        o.[type] ,
        o.Create_date ,
        sm.[definition] AS 'Stored Procedure script'
FROM    sys.objects o
        INNER JOIN sys.sql_modules sm ON o.object_id = sm.object_id
WHERE  o.[type] = 'P' -- Stored Procedures
        -- AND sm.[definition] LIKE '%insert%'
        -- AND sm.[definition] LIKE '%update%'
        -- AND sm.[definition] LIKE '%delete%'
        -- AND sm.[definition] LIKE '%tablename%'
ORDER BY o.name;
GO
```

Додавши просту умову в WHERE, ми можемо отримати інформацію тільки про ті процедури, що зберігаються, які, наприклад, виконують операції INSERT.

```
WHERE  o.[type]  = 'P'  -- Stored Procedures
        AND sm.definition LIKE '%insert%'
ORDER BY o.name
…
```

Трохи модифікувавши умову в WHERE, ми можемо зібрати інформацію про процедуру, що проводять оновлення, видалення або звертаються до певних таблиць.

ФУНКЦІЇ

Функції зберігаються в SQL Server, приймають будь-які параметри і виконують певні дії або обчислення, після чого повертають результат.

```
-- Функции
SELECT  @@Servername AS ServerName ,
        DB_NAME() AS DB_Name ,
        o.name AS 'Functions' ,
        o.[Type] ,
        o.create_date
FROM    sys.objects o
WHERE o.Type = 'FN' -- Function
ORDER BY o.NAME;
--OR
-- Дополнительная информация о функциях
SELECT  @@Servername AS ServerName ,
        DB_NAME() AS DB_Name ,
        o.name AS 'FunctionName' ,
        o.[type] ,
        o.create_date ,
        sm.[DEFINITION] AS 'Function script'
FROM    sys.objects o
        INNER JOIN sys.sql_modules sm ON o.object_id = sm.OBJECT_ID
```

```
WHERE o.[Type] = 'FN' -- Function
ORDER BY o.NAME;
GO
```

ТРИГЕРИ

Тригер – це щось подібне до процедури, що зберігаються, яка виконується у відповідь на певні дії з тієї таблицею, якій цей тригер належить. Наприклад, ми можемо створити INSERT, UPDATE та DELETE тригери.

```
-- Триггеры
SELECT  @@Servername AS ServerName ,
        DB_NAME() AS DBName ,
        parent.name AS TableName ,
        o.name AS TriggerName ,
        o.[Type] ,
        o.create_date
FROM    sys.objects o
        INNER JOIN sys.objects parent ON o.parent_object_id = parent.object_id
WHERE  o.Type = 'TR' -- Triggers
ORDER BY parent.name ,
        o.NAME
--OR
SELECT  @@Servername AS ServerName ,
        DB_NAME() AS DB_Name ,
        Parent_id ,
        name AS TriggerName ,
        create_date
FROM    sys.triggers
WHERE parent_class = 1
ORDER BY name;
--OR
-- Дополнительная информация о триггерах
SELECT  @@Servername AS ServerName ,
        DB_NAME() AS DB_Name ,
        OBJECT_NAME(Parent_object_id) AS TableName ,
        o.name AS 'TriggerName' ,
        o.Type ,
        o.create_date ,
        sm.[DEFINITION] AS 'Trigger script'
FROM    sys.objects o
        INNER JOIN sys.sql_modules sm ON o.object_id = sm.OBJECT_ID
WHERE  o.Type = 'TR' -- Triggers
ORDER BY o.NAME;
GO
```

CHECK-обмеження

CHECK-обмеження – це непоганий засіб для реалізації бізнес-логіки у базі даних. Наприклад, деякі поля мають бути позитивними, або від'ємними, або дата в одному стовпці повинна бути більшою за дати в іншому.

```
        -- Check Constraints
SELECT  @@Servername AS ServerName ,
        DB_NAME() AS DBName ,
        parent.name AS 'TableName' ,
        o.name AS 'Constraints' ,
        o.[Type] ,
        o.create_date
FROM    sys.objects o
        INNER JOIN sys.objects parent
            ON o.parent_object_id = parent.object_id
WHERE o.Type = 'C' -- Check Constraints
ORDER BY parent.name ,
        o.name
--OR
--CHECK constriant definitions
SELECT  @@Servername AS ServerName ,
        DB_NAME() AS DBName ,
        OBJECT_SCHEMA_NAME(parent_object_id) AS SchemaName ,
        OBJECT_NAME(parent_object_id) AS TableName ,
        parent_column_id AS  Column_NBR ,
        Name AS  CheckConstraintName ,
        type ,
        type_desc ,
        create_date ,
        OBJECT_DEFINITION(object_id) AS CheckConstraintDefinition
FROM    sys.Check_constraints
ORDER BY TableName ,
        SchemaName ,
        Column_NBR
GO
```

МОДЕЛЬ ДАНИХ

7. Стовпці

Наступний скрипт визначає таблиці та стовпці з усієї бази даних. Результат цього запиту можна скопіювати в Excel, де можна налаштувати фільтри та сортування і добре розібратися з типами даних, що використовуються в БД. Також зверніть увагу на стовпці з однаковими іменами, але різними типами даних.

```
ELECT  @@Servername AS Server ,
        DB_NAME() AS DBName ,
        isc.Table_Name AS TableName ,
        isc.Table_Schema AS SchemaName ,
        Ordinal_Position AS  Ord ,
        Column_Name ,
        Data_Type ,
        Numeric_Precision AS  Prec ,
        Numeric_Scale AS  Scale ,
        Character_Maximum_Length AS LEN , -- -1 means MAX like Varchar(MAX)
        Is_Nullable ,
```

```
        Column_Default ,
        Table_Type
FROM    INFORMATION_SCHEMA.COLUMNS isc
        INNER JOIN  information_schema.tables ist
            ON isc.table_name = ist.table_name
--      WHERE Table_Type = 'BASE TABLE' -- 'Base Table' or 'View'
ORDER BY DBName ,
        TableName ,
        SchemaName ,
        Ordinal_position;
-- Имена столбцов и количество повторов
-- Используется для поиска одноимённых столбцов с разными типами данных/длиной
SELECT  @@Servername AS Server ,
        DB_NAME() AS DBName ,
        Column_Name ,
        Data_Type ,
        Numeric_Precision AS  Prec ,
        Numeric_Scale AS  Scale ,
        Character_Maximum_Length ,
        COUNT(*) AS Count
FROM    information_schema.columns isc
        INNER JOIN  information_schema.tables ist
            ON isc.table_name = ist.table_name
WHERE Table_type = 'BASE TABLE'
GROUP BY Column_Name ,
        Data_Type ,
        Numeric_Precision ,
        Numeric_Scale ,
        Character_Maximum_Length;
-- Информация по используемым типам данных
SELECT  @@Servername AS ServerName ,
        DB_NAME() AS DBName ,
        Data_Type ,
        Numeric_Precision AS  Prec ,
        Numeric_Scale AS  Scale ,
        Character_Maximum_Length AS [Length] ,
        COUNT(*) AS COUNT
FROM    information_schema.columns isc
        INNER JOIN  information_schema.tables ist
            ON isc.table_name = ist.table_name
WHERE Table_type = 'BASE TABLE'
GROUP BY Data_Type ,
        Numeric_Precision ,
        Numeric_Scale ,
        Character_Maximum_Length
ORDER BY Data_Type ,
        Numeric_Precision ,
        Numeric_Scale ,
```

```
      Character_Maximum_Length
-- Large object data types or Binary Large Objects(BLOBs)
-- Помните, что индексы по этим таблицам не могут быть перестроены в режиме "online"
SELECT  @@Servername AS ServerName ,
      DB_NAME() AS DBName ,
      isc.Table_Name ,
      Ordinal_Position AS  Ord ,
      Column_Name ,
      Data_Type AS  BLOB_Data_Type ,
      Numeric_Precision AS  Prec ,
      Numeric_Scale AS  Scale ,
      Character_Maximum_Length AS [Length]
FROM    information_schema.columns isc
      INNER JOIN  information_schema.tables ist
            ON isc.table_name = ist.table_name
WHERE  Table_type = 'BASE TABLE'
      AND ( Data_Type IN ( 'text', 'ntext', 'image', 'XML' )
          OR ( Data_Type IN ( 'varchar', 'nvarchar', 'varbinary' )
              AND Character_Maximum_Length = -1
            )
        ) -- varchar(max), nvarchar(max), varbinary(max)
ORDER BY isc.Table_Name ,
      Ordinal_position;
```

8. Значення за замовчуванням

Значення за замовченням – це значення, яке буде збережено, якщо жодного значення для стовпця не буде встановлено під час вставлення. Найчастіше, для стовпців, що зберігають дату, ставлять get_date(). Також значення за замовчуванням використовуються для аудиту – вставляється system_user для визначення облікового запису користувача, що вчинив певну дію.

```
-- Table Defaults
SELECT  @@Servername AS ServerName ,
      DB_NAME() AS DBName ,
      parent.name AS TableName ,
      o.name AS Defaults ,
      o.[Type] ,
      o.Create_date
FROM    sys.objects o
      INNER JOIN sys.objects parent
            ON o.parent_object_id = parent.object_id
WHERE  o.[Type] = 'D' -- Defaults
ORDER BY parent.name ,
      o.NAME
--OR
-- Column Defaults
SELECT  @@Servername AS ServerName ,
      DB_NAME() AS DB_Name ,
      OBJECT_SCHEMA_NAME(parent_object_id) AS SchemaName ,
```

```
        OBJECT_NAME(parent_object_id) AS TableName ,
        parent_column_id AS  Column_NBR ,
        Name AS DefaultName ,
        [type] ,
        type_desc ,
        create_date ,
        OBJECT_DEFINITION(object_id) AS Defaults
FROM    sys.default_constraints
ORDER BY TableName ,
        Column_NBR
--OR
-- Column Defaults
SELECT  @@Servername AS ServerName ,
        DB_NAME() AS DB_Name ,
        OBJECT_SCHEMA_NAME(t.object_id) AS SchemaName ,
        t.Name AS TableName ,
        c.Column_ID AS Ord ,
        c.Name AS Column_Name ,
        OBJECT_NAME(default_object_id) AS DefaultName ,
        OBJECT_DEFINITION(default_object_id) AS Defaults
FROM    sys.Tables t
        INNER JOIN sys.columns c ON t.object_id = c.object_id
WHERE   default_object_id <> 0
ORDER BY TableName ,
        SchemaName ,
        c.Column_ID
GO
```

9. Стовпці, що обчислюються

Стовпці, що обчислюються, – це стовпці, значення яких обчислюються виходячи, зазвичай, зі значень з інших стовпців таблиці.

```
-- Вычисляемые столбцы
SELECT  @@Servername AS ServerName ,
        DB_NAME() AS DBName ,
        OBJECT_SCHEMA_NAME(object_id) AS SchemaName ,
        OBJECT_NAME(object_id) AS Tablename ,
        Column_id ,
        Name AS  Computed_Column ,
        [Definition] ,
        is_persisted
FROM    sys.computed_columns
ORDER BY SchemaName ,
        Tablename ,
        [Definition];
--Or
-- Computed Columns
SELECT  @@Servername AS ServerName ,
```

```
        DB_NAME() AS DBName ,
        OBJECT_SCHEMA_NAME(t.object_id) AS SchemaName,
        t.Name AS TableName ,
        c.Column_ID AS Ord ,
        c.Name AS Computed_Column
FROM    sys.Tables t
        INNER JOIN sys.Columns c ON t.object_id = c.object_id
WHERE   is_computed = 1
ORDER BY t.Name ,
        SchemaName ,
        c.Column_ID
GO
```

10. Стовпці identity

Стовпці IDENTITY автоматично заповнюються системою унікальними значеннями. Зазвичай використовуються для зберігання порядкового номера запису у таблиці.

```
SELECT  @@Servername AS ServerName ,
        DB_NAME() AS DBName ,
        OBJECT_SCHEMA_NAME(object_id) AS SchemaName ,
        OBJECT_NAME(object_id) AS TableName ,
        Column_id ,
        Name AS  IdentityColumn ,
        Seed_Value ,
        Last_Value
FROM    sys.identity_columns
ORDER BY SchemaName ,
        TableName ,
        Column_id;
GO
```

11. Ключі та індекси

Наявність первинного ключа та відповідного індексу у таблиці – це одна із best practice. Ще одна best practice полягає в тому, що зовнішні ключі також повинні мати індекс, побудований по стовпцам, що входять до зовнішнього ключа. Індекси, побудовані «за зовнішніми ключами» добре підходять для з'єднання таблиць. Ці індекси також добре позначаються на продуктивності при видаленні записів.

Скрипт для пошуку всіх індексів у всіх таблицях поточної бази даних.

```
SELECT  @@Servername AS ServerName ,
        DB_NAME() AS DB_Name ,
        o.Name AS TableName ,
        i.Name AS IndexName
FROM    sys.objects o
        INNER JOIN sys.indexes i ON o.object_id = i.object_id
WHERE   o.Type = 'U' -- User table
        AND LEFT(i.Name, 1) <> '_' -- Remove hypothetical indexes
```

```
ORDER BY o.NAME ,
      i.name;
GO
```

На підставі раніше виконуваних запитів, SQL Server надає інформацію про відсутні індекси в БД, створення яких може збільшити продуктивність.

Не додавайте ці індекси наосліп. Використання включених стовпців, наприклад, може призвести до серйозного збільшення об’єму.

```
-- Отсутствующие индексы из DMV
SELECT  @@ServerName AS ServerName ,
      DB_NAME() AS DBName ,
      t.name AS 'Affected_table' ,
      ( LEN(ISNULL(ddmid.equality_columns, N'')
          + CASE WHEN ddmid.equality_columns IS NOT NULL
                      AND ddmid.inequality_columns IS NOT NULL THEN ','
                 ELSE ''
            END) - LEN(REPLACE(ISNULL(ddmid.equality_columns, N'')
                          + CASE WHEN ddmid.equality_columns
                                              IS NOT NULL
                                      AND ddmid.inequality_columns
                                              IS NOT NULL
                                 THEN ','
                                 ELSE ''
                            END, ',', '')) ) + 1 AS K ,
      COALESCE(ddmid.equality_columns, '')
      + CASE WHEN ddmid.equality_columns IS NOT NULL
                 AND ddmid.inequality_columns IS NOT NULL THEN ','
            ELSE ''
        END + COALESCE(ddmid.inequality_columns, '') AS Keys ,
      COALESCE(ddmid.included_columns, '') AS [include] ,
      'Create NonClustered Index IX_' + t.name + '_missing_'
      + CAST(ddmid.index_handle AS VARCHAR(20))
      + ' On ' + ddmid.[statement] COLLATE database_default
      + ' (' + ISNULL(ddmid.equality_columns, '')
      + CASE WHEN ddmid.equality_columns IS NOT NULL
                 AND ddmid.inequality_columns IS NOT NULL THEN ','
            ELSE ''
        END + ISNULL(ddmid.inequality_columns, '') + ')'
      + ISNULL(' Include (' + ddmid.included_columns + ');', ';')
                                      AS sql_statement ,
      ddmigs.user_seeks ,
      ddmigs.user_scans ,
      CAST(( ddmigs.user_seeks + ddmigs.user_scans )
      * ddmigs.avg_user_impact AS BIGINT) AS 'est_impact' ,
      avg_user_impact ,
      ddmigs.last_user_seek ,
      ( SELECT    DATEDIFF(Second, create_date, GETDATE()) Seconds
        FROM      sys.databases
```

```
        WHERE     name = 'tempdb'
      ) SecondsUptime
FROM    sys.dm_db_missing_index_groups ddmig
      INNER JOIN sys.dm_db_missing_index_group_stats ddmigs
            ON ddmigs.group_handle = ddmig.index_group_handle
      INNER JOIN sys.dm_db_missing_index_details ddmid
            ON ddmig.index_handle = ddmid.index_handle
      INNER JOIN sys.tables t ON ddmid.OBJECT_ID = t.OBJECT_ID
WHERE  ddmid.database_id = DB_ID()
ORDER BY est_impact DESC;
GO
```

12. Зовнішні ключі

Зовнішні ключі визначають зв'язок між таблицями та використовуються для контролю цілісності посилання. На діаграмі сутність зв'язок лінії між таблицями позначають зовнішні ключі.

```
       -- Foreign Keys
SELECT  @@Servername AS ServerName ,
      DB_NAME() AS DB_Name ,
      parent.name AS 'TableName' ,
      o.name AS 'ForeignKey' ,
      o.[Type] ,
      o.Create_date
FROM    sys.objects o
      INNER JOIN sys.objects parent ON o.parent_object_id = parent.object_id
WHERE  o.[Type] = 'F' -- Foreign Keys
ORDER BY parent.name ,
      o.name
--OR
SELECT  f.name AS ForeignKey ,
      SCHEMA_NAME(f.SCHEMA_ID) AS SchemaName ,
      OBJECT_NAME(f.parent_object_id) AS TableName ,
      COL_NAME(fc.parent_object_id, fc.parent_column_id) AS ColumnName ,
      SCHEMA_NAME(o.SCHEMA_ID) ReferenceSchemaName ,
      OBJECT_NAME(f.referenced_object_id) AS ReferenceTableName ,
      COL_NAME(fc.referenced_object_id, fc.referenced_column_id)
                                    AS ReferenceColumnName
FROM    sys.foreign_keys AS f
      INNER JOIN sys.foreign_key_columns AS fc
            ON f.OBJECT_ID = fc.constraint_object_id
      INNER JOIN sys.objects AS o ON o.OBJECT_ID = fc.referenced_object_id
ORDER BY TableName ,
      ReferenceTableName;
GO
```

Пропущені індекси за зовнішніми ключами

Бажано мати індекс, збудований по стовпцях, що входять до зовнішнього ключа. Це значно прискорює з'єднання таблиць, які зазвичай все одно з'єднуються по зовнішньому ключу. Ці індекси також значно прискорюють операції видалення. Якщо такого індексу немає, SQL Server буде створювати table scan зв'язаній таблиці, при кожному видаленні запису з «першої» таблиці.

```
-- Foreign Keys missing indexes
-- Помните, что этот скрипт работает только для создания индексов по одному столбцу
-- Внешние ключи, состоящие более чем из одного столбца, не отслеживаются
SELECT  DB_NAME() AS DBName ,
        rc.Constraint_Name AS FK_Constraint ,
-- rc.Constraint_Catalog AS FK_Database,
-- rc.Constraint_Schema AS FKSch,
        ccu.Table_Name AS FK_Table ,
        ccu.Column_Name AS FK_Column ,
        ccu2.Table_Name AS ParentTable ,
        ccu2.Column_Name AS ParentColumn ,
        I.Name AS IndexName ,
        CASE WHEN I.Name IS NULL
            THEN 'IF NOT EXISTS (SELECT * FROM sys.indexes
                                 WHERE object_id = OBJECT_ID(N'''
                + RC.Constraint_Schema + '.' + ccu.Table_Name
                + ''') AND name = N''IX_' + ccu.Table_Name + '_'
                + ccu.Column_Name + ''') '
                + 'CREATE NONCLUSTERED INDEX IX_' + ccu.Table_Name + '_'
                + ccu.Column_Name + ' ON ' + rc.Constraint_Schema + '.'
                + ccu.Table_Name + '( ' + ccu.Column_Name
                + ' ASC ) WITH (PAD_INDEX = OFF,
                                STATISTICS_NORECOMPUTE = OFF,
                                SORT_IN_TEMPDB = ON, IGNORE_DUP_KEY = OFF,
                                DROP_EXISTING = OFF, ONLINE = ON);'
            ELSE ''
        END AS SQL
FROM    information_schema.referential_constraints RC
        JOIN INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE ccu
         ON rc.CONSTRAINT_NAME = ccu.CONSTRAINT_NAME
        JOIN INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE ccu2
         ON rc.UNIQUE_CONSTRAINT_NAME = ccu2.CONSTRAINT_NAME
        LEFT JOIN sys.columns c ON ccu.Column_Name = C.name
                            AND ccu.Table_Name = OBJECT_NAME(C.OBJECT_ID)
        LEFT JOIN sys.index_columns ic ON C.OBJECT_ID = IC.OBJECT_ID
                                    AND c.column_id = ic.column_id
                                    AND index_column_id = 1
                                     -- index found has the foreign key
                                    --  as the first column
        LEFT JOIN sys.indexes i ON IC.OBJECT_ID = i.OBJECT_ID
                                AND ic.index_Id = i.index_Id
WHERE I.name IS NULL
ORDER BY FK_table ,
```

```
    ParentTable ,
    ParentColumn;
GO
```

13. Залежності

Існує три різні методи для «реверс-інжинірингу» залежностей у БД.
Перший метод - використовувати процедуру sp_msdependecies, що зберігається.
Другий метод – системні таблиці, пов'язані із зовнішніми ключами.
Третій метод - використовувати CTE (Common Table Expression).

Sp_msdependencies – це недокументована процедура, що зберігається, яка може бути дуже корисна для розбору складних взаємозалежностей таблиць.

```
EXEC sp_msdependencies '?' -- Displays Help
sp_MSobject_dependencies name = NULL, type = NULL, flags = 0x01fd
 name:  name or null (all objects of type)
 type:  type number (see below) or null
      if both null, get all objects in database
 flags is a bitmask of the following values:
      0x10000  = return multiple parent/child rows per object
      0x20000  = descending return order
      0x40000  = return children instead of parents
      0x80000  = Include input object in output result set
      0x100000 = return only firstlevel (immediate) parents/children
      0x200000 = return only DRI dependencies
      power(2, object type number(s))  to return in results set:
            0 (1       - 0x0001)     - UDF
            1 (2       - 0x0002)     - system tables or MS-internal objects
            2 (4       - 0x0004)     - view
            3 (8       - 0x0008)     - user table
            4 (16      - 0x0010)     - procedure
            5 (32      - 0x0020)     - log
            6 (64      - 0x0040)     - default
            7 (128     - 0x0080)     - rule
            8 (256     - 0x0100)     - trigger
            12 (1024   - 0x0400) - uddt
      shortcuts:
            29  (0x011c) - trig, view, user table, procedure
            448  (0x00c1) - rule, default, datatype
            4606 (0x11fd) - all but systables/objects
            4607 (0x11ff) – all
```

Якщо ми виведемо всі залежності, використовуючи sp_msdependencies, ми матимемо чотири стовпці: Type, ObjName, Owner(Schema), Sequence.

Зверніть увагу на номер послідовності (Sequence) – він починається з 1 та послідовно збільшується. Sequence - це "порядковий номер" залежності.

Я кілька разів використовував цей метод, коли мені потрібно було виконати архівування або видалення дуже великої БД. Якщо ви знаєте залежності таблиці, значить, у вас є «дорожня карта» — в якому порядку вам потрібно архівувати або видаляти дані. Почніть з таблиці з найбільшим значенням у стовпці Sequence і рухайтеся від нього у зворотному порядку – від більшого до меншого. Таблиці з однаковим значенням Sequence можна видалити одночасно.

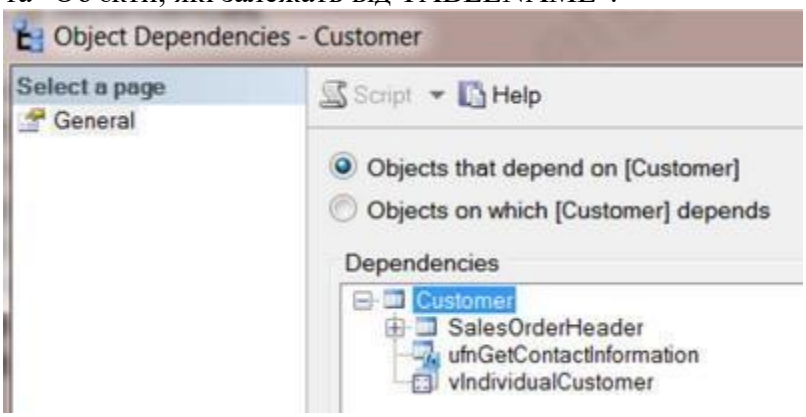
Цей метод не порушує жодного обмеження зовнішніх ключів і дозволяє перенести/видалити записи без тимчасового видалення та перебудови обмежень (constraints).

```
EXEC sp_msdependencies NULL -- Все зависимости в БД
EXEC sp_msdependencies NULL, 3 -- Зависимости определённой таблицы
```

Results | Messages

	oType	oObjName	oOwner	oSequence
2	8	AddressType	Person	1
3	8	AWBuildVersion	dbo	1
4	8	BusinessEntity	Person	1
5	8	ContactType	Person	1

У SSMS, якщо ви натиснете правою кнопкою миші на ім'я таблиці, ви зможете вибрати "View Dependencies" та "Об'єкти, які залежать від TABLENAME":

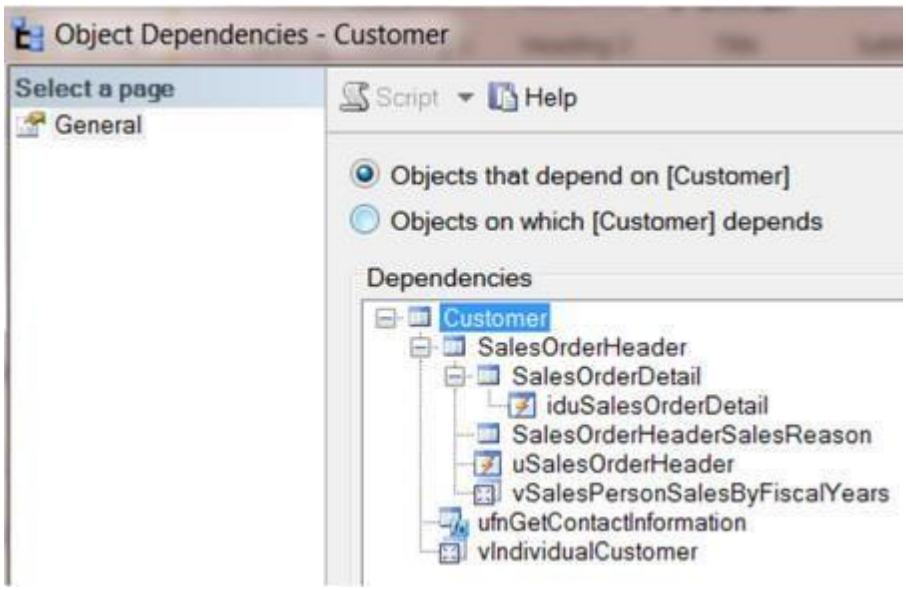


Ми також можемо отримати цю інформацію в такий спосіб:

```
-- sp_MSdependencies — Только верхний уровень
-- Объекты, которые зависят от указанного объекта
EXEC sp_msdependencies N'Sales.Customer',null, 1315327 -- Change Table Name
```

	oType	oObjName	oOwner	oSequence
1	1	ufnGetContactInformation	dbo	1
2	4	vIndividualCustomer	Sales	1
3	8	SalesOrderHeader	Sales	1
4	256	iduSalesOrderDetail	Sales	1

Якщо у SSMS у вікні перегляду залежностей, вибрати «Об'єкти, які залежать від TABLENAME», а потім розкрити всі рівні, ми побачимо наступне:

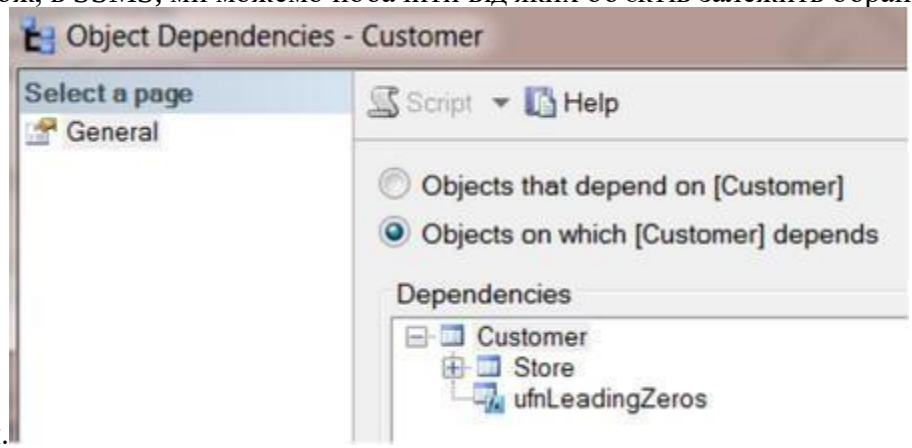


Таку саму інформацію поверне sp_msdependencies.

```
-- sp_MSdependencies - Все уровни
-- Объекты, которые зависят от указанного объекта
EXEC sp_MSdependencies N'Sales.Customer', NULL, 266751 -- Change Table Name
```

	oType	oObjName	oOwner	oSequence
1	1	ufnGetContactInformation	dbo	1
2	4	vIndividualCustomer	Sales	1
3	8	SalesOrderHeader	Sales	1
4	4	vSalesPersonSalesByFiscalYears	Sales	2
5	8	SalesOrderDetail	Sales	2
6	8	SalesOrderHeaderSalesReason	Sales	2
7	256	uSalesOrderHeader	Sales	2
8	256	iduSalesOrderDetail	Sales	3

Також, в SSMS, ми можемо побачити від яких об'єктів залежить обрана таблиця.



Наступний запит, з використанням msdependencies, повертає ту саму інформацію.

```
-- Объекты, от которых зависит указанный объект
EXEC sp_MSdependencies N'Sales.Customer', null, 1053183 -- Change Table
```

	oType	oUDDTName	oOwner	oSequence
1	4096	Name	dbo	1
2	4096	NameStyle	dbo	1

	oType	oObjName	oOwner	oSequence
1	1	ufnLeadingZeros	dbo	2
2	8	Person	Person	2
3	8	SalesTerritory	Sales	2
4	8	Store	Sales	2

Якщо ви хочете отримати список залежностей, ви можете використовувати тимчасову таблицю, щоб відфільтрувати залежності за типом.

```
CREATE TABLE #TempTable1
   (
    Type INT ,
    ObjName VARCHAR(256) ,
    Owner VARCHAR(25) ,
    Sequence INT
   );
INSERT  INTO #TempTable1
       EXEC sp_MSdependencies NULL
SELECT  *
FROM    #TempTable1
WHERE Type = 8 --Tables
ORDER BY Sequence ,
       ObjName
DROP TABLE #TempTable1;
```

Запити до системних представлень каталогу

Другий метод «реверс-інжинірингу» залежностей у вашій БД – це запити до системних преставлень каталогу, пов'язаних із зовнішніми ключами.

```
--Independent tables
SELECT Name AS InDependentTables
FROM    sys.tables
WHERE  object_id NOT IN ( SELECT  referenced_object_id
                          FROM  sys.foreign_key_columns )
                                -- Check for parents
       AND object_id NOT IN ( SELECT parent_object_id
                              FROM  sys.foreign_key_columns )
                                -- Check for Dependents
ORDER BY Name
-- Tables with dependencies.
SELECT DISTINCT
       OBJECT_NAME(referenced_object_id) AS ParentTable ,
       OBJECT_NAME(parent_object_id) AS DependentTable ,
       OBJECT_NAME(constraint_object_id) AS ForeignKeyName
FROM    sys.foreign_key_columns
ORDER BY ParentTable ,
       DependentTable
-- Top level of the pyramid tables. Tables with no parents.
SELECT DISTINCT
```

```
        OBJECT_NAME(referenced_object_id) AS TablesWithNoParent
FROM    sys.foreign_key_columns
WHERE    referenced_object_id NOT IN ( SELECT  parent_object_id
                                FROM    sys.foreign_key_columns )
ORDER BY 1
-- Bottom level of the pyramid tables.
-- Tables with no dependents. (These are the leaves on a tree.)
SELECT DISTINCT
        OBJECT_NAME(parent_object_id) AS TablesWithNoDependents
FROM    sys.foreign_key_columns
WHERE  parent_object_id NOT IN ( SELECT  referenced_object_id
                            FROM    sys.foreign_key_columns )
ORDER BY 1
-- Tables with both parents and dependents.
-- Tables in the middle of the hierarchy
SELECT DISTINCT
        OBJECT_NAME(referenced_object_id) AS MiddleTables
FROM    sys.foreign_key_columns
WHERE    referenced_object_id IN ( SELECT  parent_object_id
                            FROM    sys.foreign_key_columns )
        AND parent_object_id  NOT IN ( SELECT  referenced_object_id
                                FROM    sys.foreign_key_columns )
ORDER BY 1;
-- in rare cases, you might find a self-referencing dependent table.
-- Recursive (self) referencing table dependencies.
SELECT DISTINCT
        OBJECT_NAME(referenced_object_id) AS ParentTable ,
        OBJECT_NAME(parent_object_id) AS ChildTable ,
        OBJECT_NAME(constraint_object_id) AS ForeignKeyName
FROM    sys.foreign_key_columns
WHERE    referenced_object_id = parent_object_id
ORDER BY 1 ,
        2;
```

Використання CTE

Третій метод для отримання ієрархії залежностей – використання рекурсивного CTE.

```
-- How to find the hierarchical dependencies
-- Solve recursive queries using Common Table Expressions (CTE)
WITH    TableHierarchy (  ParentTable, DependentTable, Level )
        AS (
-- Anchor member definition (First level group to start the process)
            SELECT DISTINCT
                    CAST(NULL AS INT) AS ParentTable ,
                    e.referenced_object_id AS DependentTable ,
                    0 AS Level
            FROM    sys.foreign_key_columns AS e
            WHERE    e.referenced_object_id NOT IN (
                    SELECT  parent_object_id
```

```
                    FROM    sys.foreign_key_columns )
-- Add filter dependents of only one parent table
-- AND Object_Name(e.referenced_object_id) = 'User'
            UNION ALL
-- Recursive member definition (Find all the layers of dependents)
            SELECT --Distinct
                    e.referenced_object_id AS ParentTable ,
                    e.parent_object_id AS DependentTable ,
                    Level + 1
            FROM    sys.foreign_key_columns AS e
                    INNER JOIN TableHierarchy AS d
                        ON ( e.referenced_object_id ) =
                                        d.DependentTable
          )
   -- Statement that executes the CTE
SELECT DISTINCT
      OBJECT_NAME(ParentTable) AS ParentTable ,
      OBJECT_NAME(DependentTable) AS DependentTable ,
      Level
FROM    TableHierarchy
ORDER BY Level ,
      ParentTable ,
      DependentTable;
```

Вимоги до звітності:

1. Звіт зі скриптами оформити наступним чином:

Процедура (назва чи код)	**Джерело**	**Використані поля**	**Призначення**	**З'єднання**
Перелік БД сервра	sys.databases	d.name, create_date , compatibility_level	Відомості про БД	sys.databases d JOIN sys.master_files m ON d.database_id = m.database_id
	sys.master_files	physical_name	Ім'я файлу, де розміщуєтсья	

Results | Messages

	Server	DBName	create_date	compatibility_level	FileName
1	FILIFIONKA	AdventureWorks	2021-02-25 12:36:44.673	140	d:\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQ...
2	FILIFIONKA	AdventureWorks2014	2020-10-09 12:22:45.820	120	d:\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQ...
3	FILIFIONKA	AdventureWorksDW	2020-12-23 15:52:48.210	140	d:\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQ...
4	FILIFIONKA	AdventureWorksDW2017	2020-09-30 16:36:03.093	140	d:\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQ...
5	FILIFIONKA	AmtrastTest	2020-09-09 05:00:46.257	140	d:\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQ...
6	FILIFIONKA	AMTrustTest	2020-04-07 16:39:59.363	140	d:\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQ...

2. Файл із скриптами та файл із скриптом бази даних.