About Me

Anthony Mason

MCSE SQL Server 2012 Data Platform

Director of Software Engineering

Therapy Management Corporation

Hattiesburg, MS

Anthony.Mason1@gmail.com

ADVANCED T-SQL & SQL SERVER FEATURES

Topics

- CTEs
- Cross / Outer Applies
- Windowing Functions
- Lag / Lead
- Statistics Options
- Grouping Sets
- Degrees of Parallelism
- SQL CLR
- Transparent Data Encryption
- Change Data Capture (CDC)

Common Table Expressions

- A temporary query with a scope of a single SELECT, UPDATE, DELETE, or INSERT statement
- Can be used to organize and simplify complex SQL queries
- Pull common elements (subqueries) into a CTE can often improve performance and simplify execution plans

Common Table Expressions

- Let's take a simple example

```
SELECT SalesPersonID, Count(SalesOrderID) as TotalSales, YEAR(OrderDate) AS SalesYear
FROM Sales.SalesOrderHeader
WHERE SalesPersonID IS NOT NULL
Group by YEAR(OrderDate),SalesPersonID
ORDER BY SalesPersonID,SalesYear
```

Common Table Expressions

- Let's simplify it to be easier to read

```
WITH Sales_CTE
AS
(
    SELECT SalesPersonID, SalesOrderID, YEAR(OrderDate) AS SalesYear
    FROM Sales.SalesOrderHeader
    WHERE SalesPersonID IS NOT NULL
)
SELECT SalesPersonID, COUNT(SalesOrderID) AS TotalSales, SalesYear
FROM Sales_CTE
GROUP BY SalesYear, SalesPersonID
ORDER BY SalesPersonID, SalesYear;
```

Chaining CTEs

```
with averagesByMonth
as
(
select
avg(unitprice) as averageOrders,
month(orderdate) as month,
year(orderdate) as year
from sales.salesorderdetail sod
inner join  sales.salesorderheader soh on soh.salesorderid = sod.salesorderid
group by year(orderdate), month(orderdate)
),
averagesByYear as
(
select avg(unitprice) as averageOrders, year(orderdate) as year from sales.salesorderdetail sod
inner join  sales.salesorderheader soh on soh.salesorderid = sod.salesorderid
group by year(orderdate)
)
select m.AverageOrders as AverageByMonth,
y.AverageOrders as AverageByYear, m.Month, m.year  from averagesByMonth m
inner join averagesByYear y on m.year = y.year
 order by m.year, m.month
```

Recursive CTEs

- Extremely handy for hierarchy queries and dynamic date ranges
- Recursive CTEs are by default blocked at 100 levels of recursion, but you by modify this OPTION(MAXRECURSION N)

```sql
WITH OrgPath (BusinessEntityID, ManagerID, lv)
AS (
    -- Anchor
      SELECT BusinessEntityID, ManagerID, 1
        FROM HumanResources.Employee
        WHERE ManagerID IS NULL -- should only be EmployeeID 1
        -- WHERE EmployeeID = 1 -- the CEO

    -- Recursive Call
    UNION ALL
      SELECT E.BusinessEntityID, E.ManagerID, lv + 1
        FROM HumanResources.Employee E
          JOIN OrgPath
            ON E.ManagerID = OrgPath.BusinessEntityID
    )
SELECT Emp.BusinessEntityID, Emp.JobTitle,
    C.FirstName + ' ' + C.LastName AS [Name],
    M.FirstName + ' ' + M.LastName AS [Manager], Lv
  FROM HumanResources.Employee Emp
    JOIN OrgPath
      ON Emp.BusinessEntityID = OrgPath.BusinessEntityID
    JOIN Person.Person AS C
      ON C.BusinessEntityID = Emp.BusinessEntityID
    Left Join Person.Person AS M
      ON Emp.ManagerID = M.BusinessEntityID
  ORDER BY Lv
```

Great for report date ranges

```
DECLARE @Start datetime = '1/1/2014'
DECLARE @End datetime = '12/31/2014'
;WITH Months as
  (
  SELECT convert(datetime,convert(varchar(2),MONTH(@Start)) + '/1/' + convert(varchar(4),YEAR(@Start))) as MonthStart
  UNION ALL
  SELECT DATEADD(MONTH,1,MonthStart) as MonthStart from Months
  WHERE MonthStart <= @End
  ),q as
  (
  select MonthStart,DATEADD(DAY,-1,DATEADD(MONTH,1,MonthStart)) as MonthEnd from Months
  )
  select * from q where MonthStart between @Start and @End
  OPTION(MAXRECURSION 150)
```

Cross Apply

- Think of a cross apply as an INNER JOIN on a table valued function.

```
select * from dbo.ufnGetContactInformation(5)

--nope!!!
select * from Person.Person p
inner join dbo.ufnGetContactInformation(p.BusinessEntityID) contactInfo
where BusinessEntityID = 5

select p.*,contactInfo.* from Person.Person p
cross apply(select * from dbo.ufnGetContactInformation(p.BusinessEntityID)) contactInfo
where p.BusinessEntityID between 5 and 10
```

Outer Apply

- Equivalent of a LEFT JOIN on a table valued function (or inner query)

```
select p.*,contactInfo.* from Person.Person p
outer apply(select * from dbo.ufnGetContactInformation(p.BusinessEntityID)) contactInfo
where p.BusinessEntityID between 5 and 50
```

Windowing Functions

- Windowing functions give you greater control over aggregates and ranking functions
- Aggregates
- ROW_NUMBER
- RANK
- DENSE_RANK
- NTILE

Windowing Aggregates

```sql
SELECT
  TerritoryGroup,
  TerritoryName,
  SalesLastYear,
  COUNT(SalesLastYear) OVER(PARTITION BY TerritoryName) AS SalesCnt,
  SUM(SalesLastYear) OVER(PARTITION BY TerritoryName) AS SalesTtl,
  AVG(SalesLastYear) OVER(PARTITION BY TerritoryName) AS SalesAvg
FROM
  Sales.vSalesPerson
WHERE
  TerritoryGroup IS NOT NULL;
```

ROW_NUMBER

```
SELECT
  BusinessEntityID AS SalesID,
  FirstName + ' ' + LastName AS FullName,
  SalesLastYear,
  ROW_NUMBER() OVER(ORDER BY SalesLastYear ASC) AS RowNumber
FROM
  Sales.vSalesPerson;
```

00 %

Results | Messages

	SalesID	FullName	SalesLastYear	RowNumb...
1	274	Stephen Jiang	0.00	1
2	284	Tete Mensa-Annan	0.00	2
3	285	Syed Abbas	0.00	3
4	287	Amy Alberts	0.00	4

RANK

```
SELECT
  BusinessEntityID AS SalesID,
  FirstName + ' ' + LastName AS FullName,
  SalesLastYear,
  RANK() OVER(ORDER BY SalesLastYear ASC) AS SalesRank
FROM
  Sales.vSalesPerson;
```

%

Results | Messages

SalesID	FullName	SalesLastYear	SalesRank
274	Stephen Jiang	0.00	1
284	Tete Mensa-Annan	0.00	1
285	Syed Abbas	0.00	1
287	Amy Alberts	0.00	1
288	Rachel Valdez	1307949.7917	5
283	David Campbell	1371635.3158	6
276	Linda Mitchell	1439156.0291	7
278	Garrett Vargas	1620276.8966	8

DENSE_RANK

```
SELECT
  BusinessEntityID AS SalesID,
  FirstName + ' ' + LastName AS FullName,
  SalesLastYear,
  DENSE_RANK() OVER(ORDER BY SalesLastYear ASC) AS DenseRank
FROM
  Sales.vSalesPerson;
```

0 %

Results | Messages

SalesID	FullName	SalesLastYear	DenseRa...
274	Stephen Jiang	0.00	1
284	Tete Mensa-Annan	0.00	1
285	Syed Abbas	0.00	1
287	Amy Alberts	0.00	1
288	Rachel Valdez	1307949.7917	2
283	David Campbell	1371635.3158	3
276	Linda Mitchell	1439156.0291	4
278	Garrett Vargas	1620276.8966	5
289	Jae Pak	1635823.3967	6
275	Michael Blythe	1750406.4785	7

NTILE

```
SELECT
  BusinessEntityID AS SalesID,
  FirstName + ' ' + LastName AS FullName,
  SalesLastYear,
  NTILE(4) OVER(ORDER BY SalesLastYear ASC) AS NTileRank
FROM
  Sales.vSalesPerson;
```

Results | Messages

SalesID	FullName	SalesLastYear	NTileRa...
274	Stephen Jiang	0.00	1
284	Tete Mensa-Annan	0.00	1
285	Syed Abbas	0.00	1
287	Amy Alberts	0.00	1
288	Rachel Valdez	1307949.7917	1
283	David Campbell	1371635.3158	2
276	Linda Mitchell	1439156.0291	2
278	Garrett Vargas	1620276.8966	2
289	Jae Pak	1635823.3967	2
275	Michael Blythe	1750406.4785	3
279	Tsvi Reiter	1849640.9418	3
280	Pamela Ansman-Wolfe	1927059.178	3

PARTITION BY

- Resets the windowing function

```
SELECT
  BusinessEntityID AS SalesID,
  TerritoryGroup,
  SalesLastYear,
  ROW_NUMBER() OVER(PARTITION BY TerritoryGroup
    ORDER BY SalesLastYear ASC) AS RowNumber,
  RANK() OVER(PARTITION BY TerritoryGroup
    ORDER BY SalesLastYear ASC) AS SalesRank,
  DENSE_RANK() OVER(PARTITION BY TerritoryGroup
    ORDER BY SalesLastYear ASC) AS DenseRank,
  NTILE(2) OVER(PARTITION BY TerritoryGroup
    ORDER BY SalesLastYear ASC) AS NTileRank
FROM
  Sales.vSalesPerson;
```

100 %

Results | Messages

	SalesID	TerritoryGroup	SalesLastYear	RowNum...	SalesRa...	DenseRa...	NTileRa...
1	274	NULL	0.00	1	1	1	1
2	285	NULL	0.00	2	1	1	1
3	287	NULL	0.00	3	1	1	2
4	288	Europe	1307949.7917	1	1	1	1
5	289	Europe	1635823.3967	2	2	2	1
6	290	Europe	2396539.7601	3	3	3	2
7	284	North America	0.00	1	1	1	1
8	283	North America	1371635.3158	2	2	2	1
9	276	North America	1439156.0291	3	3	3	1
10	278	North America	1620276.8966	4	4	4	1
11	275	North America	1750406.4785	5	5	5	1

Lag & Lead

- New in SQL Server 2012
- More efficient way of getting previous / next row values for subtotals

```
--Lag / Lead

SELECT TerritoryName, BusinessEntityID, SalesYTD,
       LAG (SalesYTD, 1, 0) OVER (PARTITION BY TerritoryName ORDER BY SalesYTD DESC) AS PrevRepSales
FROM Sales.vSalesPerson
WHERE TerritoryName IN (N'Northwest', N'Canada')
ORDER BY TerritoryName;
```

100 %

Results | Messages

	TerritoryNa...	BusinessEntit...	SalesYTD	PrevRepSales
1	Canada	282	2604540.7172	0.00
2	Canada	278	1453719.4653	2604540.7172
3	Northwest	284	1576562.1966	0.00
4	Northwest	283	1573012.9383	1576562.1966
5	Northwest	280	1352577.1325	1573012.9383

Statistics Options

- SET STATISTICS TIME
- Handy for looking at millisecond runtimes while rewriting or optimizing queries

```
--Lag / Lead
set statistics time on;

SELECT TerritoryName, BusinessEntityID, SalesYTD,
       LAG (SalesYTD, 1, 0) OVER (PARTITION BY TerritoryName ORDER BY SalesYTD DESC) AS PrevRepSales
FROM Sales.vSalesPerson
WHERE TerritoryName IN (N'Northwest', N'Canada')
ORDER BY TerritoryName;
```

```
00 %
Results  Messages
SQL Server parse and compile time:
   CPU time = 7 ms, elapsed time = 7 ms.

(5 row(s) affected)

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
```

Statistics Options

- SET STATISTICS IO ON
- Handy for examining disk i/o and reducing physical reads through indexes, CTEs, or temp tables

```
--Lag / Lead
set statistics io on;

SELECT TerritoryName, BusinessEntityID, SalesYTD,
       LAG (SalesYTD, 1, 0) OVER (PARTITION BY TerritoryName ORDER BY SalesYTD DESC) AS PrevRepSales
FROM Sales.vSalesPerson
WHERE TerritoryName IN (N'Northwest', N'Canada')
ORDER BY TerritoryName;
```

```
0 %
Results  Messages

(5 row(s) affected)
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob rea
Table 'PersonPhone'. Scan count 5, logical reads 10, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob
Table 'EmailAddress'. Scan count 5, logical reads 10, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob
Table 'Person'. Scan count 0, logical reads 15, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-
Table 'BusinessEntityAddress'. Scan count 5, logical reads 10, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical rea
Table 'SalesTerritory'. Scan count 0, logical reads 28, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, l
Table 'SalesPerson'. Scan count 1, logical reads 2, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob r
```

Grouping Sets

- Allows you to eliminate unions and subqueries when creating subtotals and grand totals

```
SELECT      [Group], [Name], SUM(SalesYTD) AS 'Total Sales'
FROM        dbo.Sales
GROUP BY [Group], [Name]

UNION ALL

SELECT      [Group], NULL, SUM(SalesYTD) AS 'Total Sales'
FROM        dbo.Sales
GROUP BY [Group]

UNION ALL

SELECT      NULL, NULL, SUM(SalesYTD) AS 'Total Sales'
FROM        dbo.Sales
```

Result Set:

Group	Name	Total Sales
North America	Northwest	123237.00
South America	Northwest	37534.00
North America	Southwest	164232.00
South America	Southwest	39667.00
North America	NULL	287469.00
South America	NULL	77201.00
NULL	NULL	364670.00

Grouping Sets

- Allows you to eliminate unions and subqueries when creating subtotals and grand totals
- Can also give you the results as GROUP BY ROLLUP, GROUP BY CUBE, but with cleaner syntax (personal preference)

```
-- WITH ROLLUP Equivalent
SELECT      [Group], [Name], SUM(SalesYTD) AS 'Total Sales'
FROM        dbo.Sales
GROUP BY GROUPING SETS (([Group], [Name]), ([Group]), ())
```

MAX DOP

- Maximum Degrees of Parallelism
- Gives you custom control over how many threads (cores) a particular query can split off to, according to its execution plan
- Can set it on a server wide basis, or on a specific query

MAX DOP

- Changing global server setting:

```
--turn on advanced options
EXEC dbo.sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO

exec dbo.sp_configure


EXEC dbo.sp_configure 'max degree of parallelism',8;
GO
RECONFIGURE;
GO
```

MAX DOP

- On a query basis:

```
WITH OrgPath (BusinessEntityID, ManagerID, lv)
AS (
   -- Anchor
      SELECT BusinessEntityID, ManagerID, 1
        FROM HumanResources.Employee
        WHERE ManagerID IS NULL -- should only be EmployeeID 1
        -- WHERE EmployeeID = 1 -- the CEO

    -- Recursive Call
    UNION ALL
      SELECT E.BusinessEntityID, E.ManagerID, lv + 1
        FROM HumanResources.Employee E
          JOIN OrgPath
            ON E.ManagerID = OrgPath.BusinessEntityID
    )
SELECT Emp.BusinessEntityID, Emp.JobTitle,
    C.FirstName + ' ' + C.LastName AS [Name],
    M.FirstName + ' ' + M.LastName AS [Manager], Lv
  FROM HumanResources.Employee Emp
    JOIN OrgPath
      ON Emp.BusinessEntityID = OrgPath.BusinessEntityID
    JOIN Person.Person AS C
      ON C.BusinessEntityID = Emp.BusinessEntityID
    Left Join Person.Person AS M
      ON Emp.ManagerID = M.BusinessEntityID
  ORDER BY Lv
  OPTION(MAXDOP 3)
```

SQL Server CDC
Change Data Capture

Change Data Capture

- An Enterprise-Only level Feature (as of 2012)
- Allows you to accomplish the traditional “audit log” in a matter of minutes
- Does not depend on triggers
- High performance, low-impact

Change Data Capture

```
--enable our database!
EXEC sys.sp_cdc_enable_db

--enable a specific table
exec sys.sp_cdc_enable_table @source_schema = 'HumanResources',@source_name = 'Employee',@role_name = null
;

Update HumanResources.Employee
set JobTitle = 'Test Value!' where BusinessEntityID = 5

--lets look at the changes!
select * from cdc.HumanResources_Employee_CT
```

Change Data Capture

- Don’t forget to set your retention period!

```
--what is the retention period?
--its in minutes!  Default is 3 days!
SELECT [retention]
  FROM [msdb].[dbo].[cdc_jobs]
  WHERE [database_id] = DB_ID()
  AND [job_type] = 'cleanup'

EXEC sp_cdc_change_job @job_type='cleanup', @retention=525600


--this is going to DELETE all your audit data!!!  Backup first!
EXEC sys.sp_cdc_disable_db
```

SQL Server CLR

SQL Server CLR (C#)

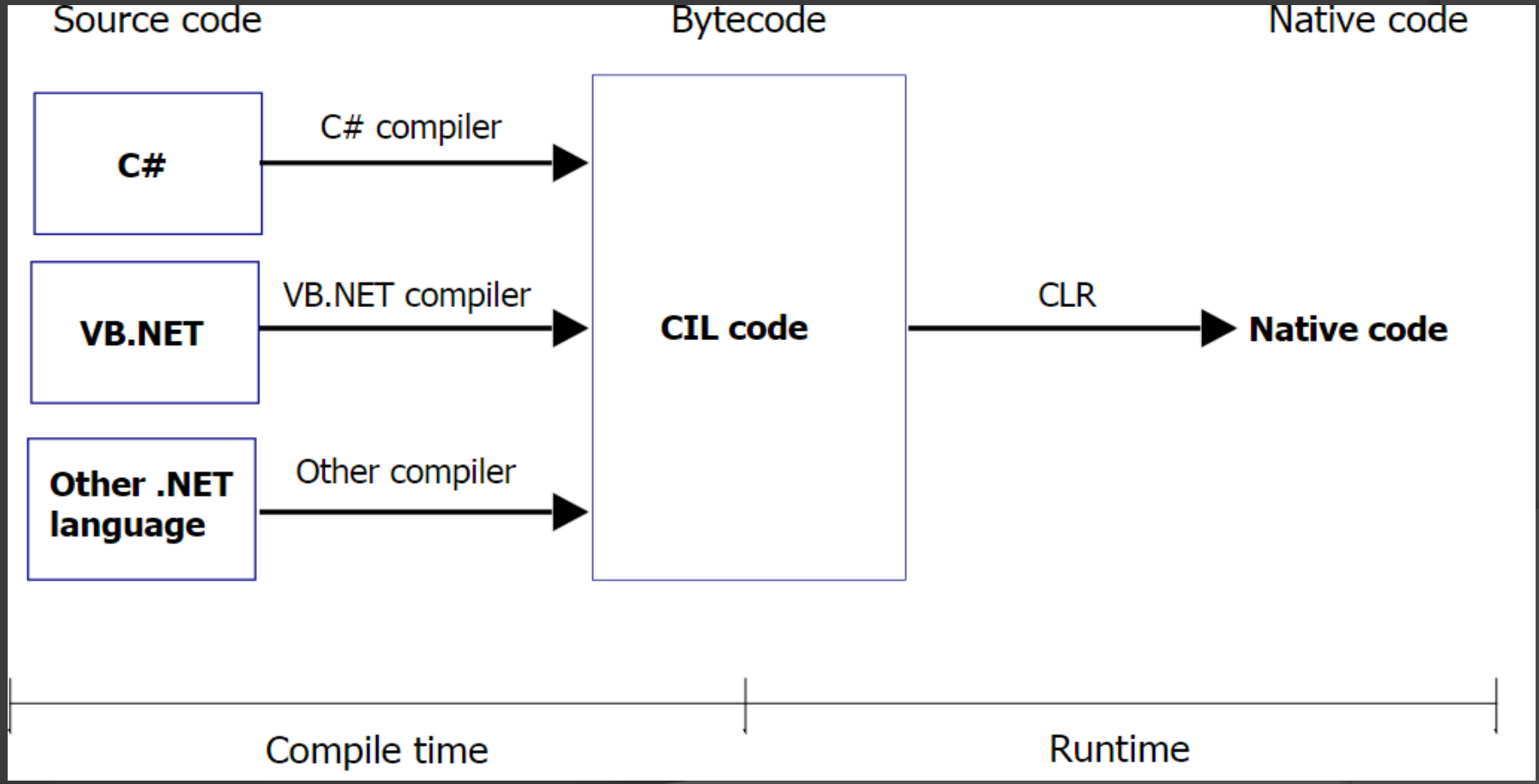
- Allows you to use a .Net Assembly (i.e. C# / VB.Net code) in SQL
  - User Defined Functions
  - Stored Procedures
  - Custom Data Types
  - Triggers

SQL Server CLR

Should be used conservatively and cautiously – can easily lead to performance issues, particularly with data-access CLR code, but can still offer performance gains over scalar UDF's or row-by-row (non-batch) queries.

.Net CLR

Microsoft's implementation of the Common Language Infrastructure
Converted "compiled" .Net Code (bytecode) to Machine Code



Example

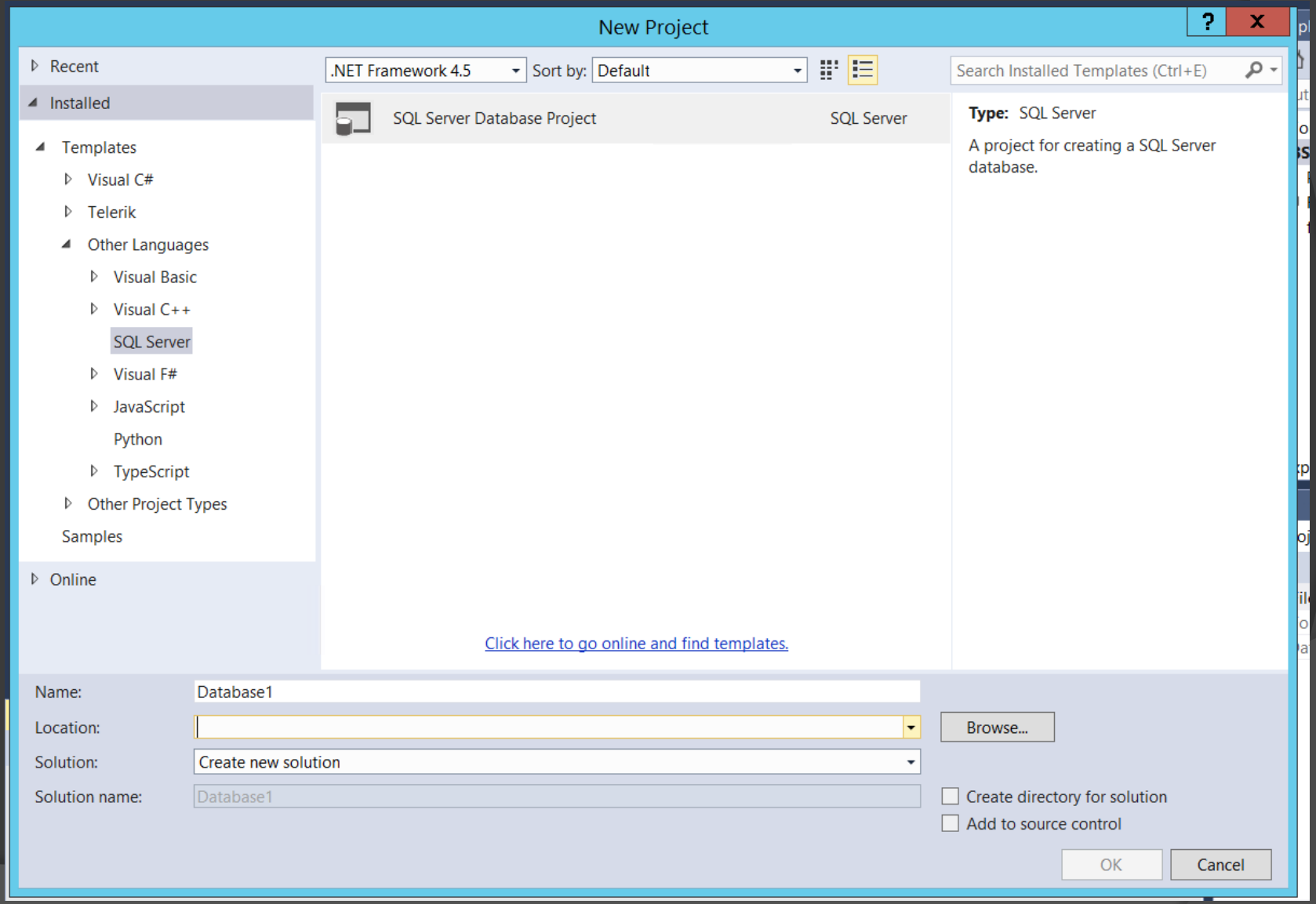
SQL Server has a LIKE statement which is powerful and fast, but maybe we just want to use an old fashioned regular expression to pass directly to a report.

Step 1: Enabling CLR

CLR is not enabled by default!

```
-- Enable CLR

--tell SQL Server to allow you to list the more advanced server options
EXEC sp_configure 'show advanced options' , '1'
GO
RECONFIGURE
GO

--turn CLR on

EXEC sp_configure 'clr enabled' , '1'
GO
RECONFIGURE
GO

--hide the advanced options again
EXEC sp_configure 'show advanced options' , '0';
GO
```

Step 2: Creating Your Project



Step 3: Write your code!

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;

public partial class UserDefinedFunctions
{
    [Microsoft.SqlServer.Server.SqlFunction]
    public static SqlBoolean DoesRegExMatch(string stringToInspect,string StandBackImGonnaUseRegex)
    {
        return System.Text.RegularExpressions.Regex.IsMatch(stringToInspect, StandBackImGonnaUseRegex);
    }
}
```

Step 4: Bring Your Assembly Into SQL Server

- Your assembly (dll) needs to be on the same machine as your SQL Server service

```
--Put your assembly somewhere on your SQL Server where you can find it
--run this command so you have a reference to that assembly

-- Install Assembly!
CREATE ASSEMBLY MyAwesomeAssembly FROM 'C:\MyAssemblies\OriginalTestProjectName.dll'
GO
```

Step 5: Tie your SQL Server object to your assembly

```
-- Create function from your assembly
CREATE FUNCTION dbo.[fnMatchRegex](@InputString NVARCHAR(MAX),@RegExpression NVARCHAR(MAX))
RETURNS bit
WITH EXECUTE AS CALLER
AS
EXTERNAL NAME MyAwsesomeAssembly.UserDefinedFunctions.DoesRegExMatch;
GO
```

Tips – Handy related SQL

- You can't drop an assembly from SQL Server without first dropping its dependencies

```
--ASSEMBLY DEPENDENCIES
SELECT  am.object_id, am.assembly_id, am.assembly_class, am.assembly_method, o. name, o.type, o.type_desc
FROM SYS.ASSEMBLY_MODULES am
JOIN SYS.OBJECTS o ON am.object_id = o.object_id
```

100 %

Results | Messages

	object_id	assembly...	assembly_class	assembly_meth...	name	ty...	type_desc
1	1536932747	65537	UserDefinedFunctions	fnRegEx	fnMatchRegex	FS	CLR_SCALAR_FUNCTION

Query Assemblies Registered w/ SQL Server

```
--LIST ASSEMBLIES
SELECT * FROM SYS.assemblies
```

```
DROP ASSEMBLY MyAwesomeAssembly
GO
```

Step 6: Use it!

```
select * from AdventureWorks2012.Person.Address a
where dbo.fnMatchRegex(a.AddressLine1,'(\d)\1\1\1') = 1
```

100 %

Results | Spatial results | Messages

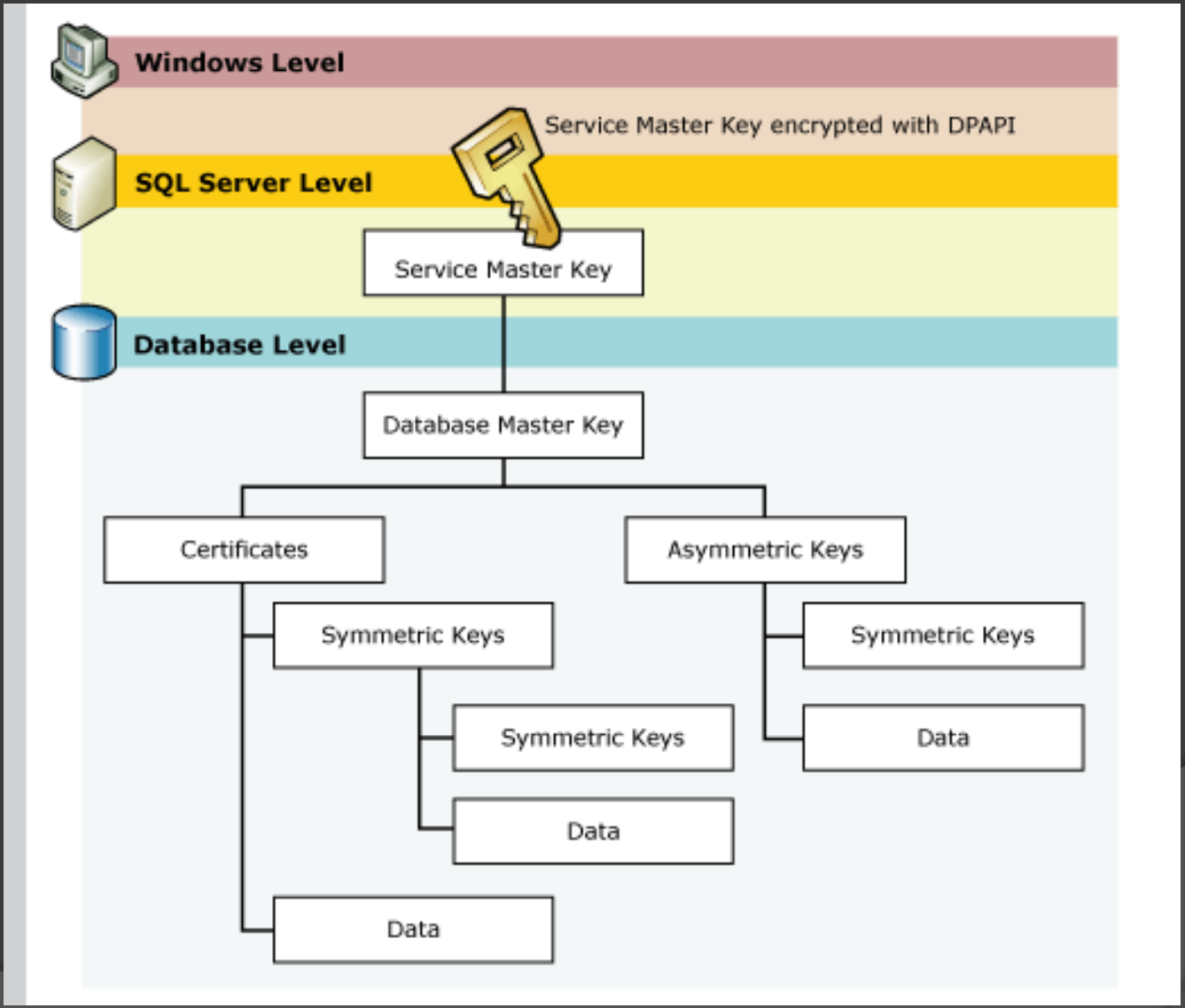
	Address...	AddressLine1	AddressLin...	City	StateProvinc...	PostalCo...
1	195	4444 Pepper Way	NULL	Sammamish	79	98074
2	276	3333 Madhatter Circle	NULL	Issaquah	79	98027
3	789	250000 Eight Mile Road	NULL	Detroit	35	48226
4	909	3333 Micro Drive	NULL	Millington	72	38054

Transparent Data Encryption (TDE)

SQL Server Encryption (TDE)

- “Data at Rest” Encryption
- Encrypts the entire database on the disk
- SQL Server Enterprise Edition Feature
- Requires No Modification to Applications or Code

SQL Server Encryption Keys



Create Master Level Key

```
USE master
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'SQLServerInstanceLevelMasterPassword';
GO
CREATE CERTIFICATE MasterCertForTDE WITH SUBJECT = 'Master Cert for TDE Sample'
GO
```

Create DB Level Key & Enable

```
use AdventureWorks2012
CREATE DATABASE ENCRYPTION KEY WITH ALGORITHM = AES_256 ENCRYPTION BY SERVER CERTIFICATE MasterCertForTDE
GO
ALTER DATABASE AdventureWorks2012 SET ENCRYPTION ON
GO
```

Check on Encryption Status

```
--To monitor encryption progress you can use this query
SELECT db_name(database_id), encryption_state,   percent_complete, key_algorithm, key_length
FROM sys.dm_database_encryption_keys
```

100 %

Results | Messages

	(No column name)	encryption_state	percent_compl...	key_algorit...	key_len...
1	tempdb	3	0	AES	256
2	AdventureWorks2012	3	0	AES	256

Questions & Beer Time.