Anfitrionas: Hablemos de tecnología W4TT

Cristina Tarabini-Castellani Ciordia

tSQLt y SQL Server: Una pareja bien avenida

Database Engineer

Agenda

- ¿Qué es tSQLt?
- ¿Por qué usarlo?
- Algunos conceptos
- ¿Cómo usarlo?

Introducción

Las pruebas ayudan a aumentar la calidad de nuestro código T-SQL

¿Qué se necesita para probar código T-SQL?

Tiempo

Datos

Conocimiento

¿Cómo conseguirlo sin morir en el intento?

Es posible con …

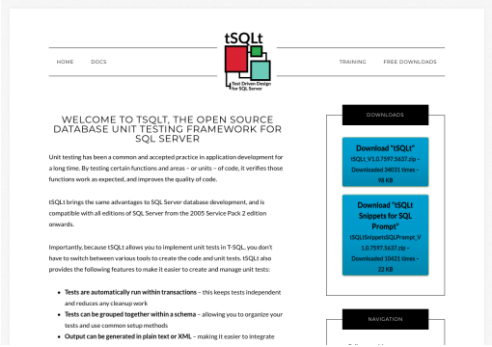
¿Qué es tSQLt?

Framework para pruebas unitarias exclusivo para SQL Server

Desarrollo liderado por Dr. Sebastian Meine y Dennis Lloyd Jr.

Open Source: https://github.com/tSQLt-org/tsqlt

Más información: https://tsqlt.org/



¿Qué es una prueba unitaria (unit test)?

"Es una forma de comprobar el correcto funcionamiento de una unidad de código"
"Asegura que cada unidad funcione correctamente"

En nuestro caso *unidad* puede ser:

- Procedimiento almacenado
- Función

Requisitos para que tenga calidad suficiente:

- Automatizable: Sin interacción manual para su integración continua
- Completas: Deben ofrecer la máxima cobertura de código
- Reutilizables
- Independientes
- Profesionales: Deben considerarse parte del código y de la documentación

¿Por qué tSQLt?

Escritos en T-SQL

No se necesita conocer otro lenguaje

Se puede trabajar con SSMS

Aunque hay herramientas más amigables

Ejecutables como SPs

Versionables

Almacenables en un repositorio de código

Repetibles y reutilizables

Pueden ejecutarse todas veces que se necesite

Aislados cada uno en una transacción

Instalación tSQLt

Descargar archivo comprimido de https://tsqlt.org/downloads/

Instalación por cada base de datos

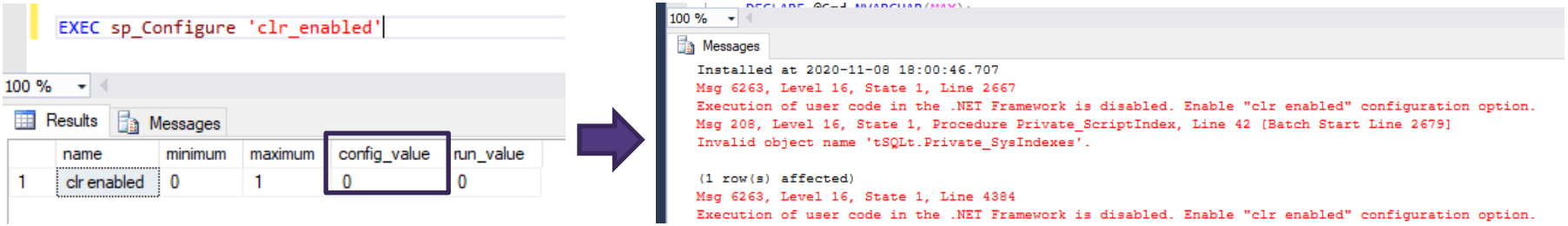
PRODUCCIÓN

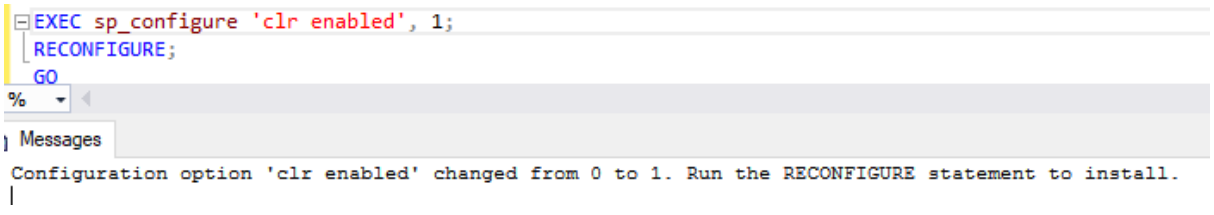
Muchos **bloqueos**

Posible **pérdida** de datos

Instalación tSQLt : Pre-requisitos (I)

- *CLR Enabled* a 1





Instalación tSQLt : Pre-requisitos (II)

- Propiedad *TRUSTWORTHY* a *ON*

```
select name,is_trustworthy_on from sys.databases
```

100 %

Results | Messages

	name	is_trustworthy_on
1	master	0
2	tempdb	0
3	model	0
4	msdb	1
5	AdventureWorks2016	0

```
ALTER DATABASE AdventureWorks2016 SET TRUSTWORTHY ON;
select name,is_trustworthy_on from sys.databases
```

100 %

Results | Messages

	name	is_trustworthy_on
1	master	0
2	tempdb	0
3	model	0
4	msdb	1
5	AdventureWorks2016	1

DEMO

Instalación tSQLt

Conceptos: Clase

- Categoriza y agrupa los test
- Es un *esquema* en la BD
- Aporta significado al conjunto. Todos ellos deberían cubrir:
    - Ejecución correcta
    - Excepciones o errores
    - Límites y extremos

TIP

Normalizar criterios para su nomenclatura y distinción de los esquemas productivos

DEMO

Creación Clase

Conceptos: Test unitario

Obligatorio que empiece por **'Test'**

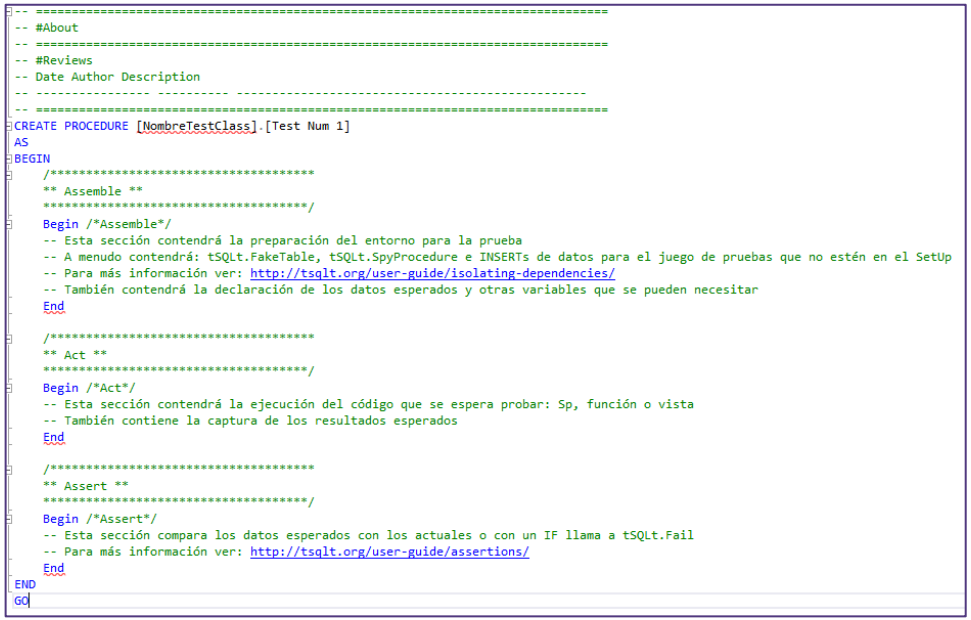
Cada test es un procedimiento almacenado

Un requisito o pregunta

Independiente del resto de test de su clase

Tiene tres áreas (Las 3 As):

- Preparación/Organización — **A**rrange
- Actuación — **A**ct
- Afirmación — **A**ssert



```
-- ****************************************************************************
-- #About
-- ****************************************************************************
-- #Reviews
-- Date Author Description
-- ---------- ---------- ----------------------------------------------------
-- ****************************************************************************
CREATE PROCEDURE [NombreTestClass].[Test Num 1]
AS
BEGIN
    /*************************************
    ** Assemble **
    *************************************/
    Begin /*Assemble*/
    -- Esta sección contendrá la preparación del entorno para la prueba
    -- A menudo contendrá: tSQLt.FakeTable, tSQLt.SpyProcedure e INSERTs de datos para el juego de pruebas que no estén en el SetUp
    -- Para más información ver: http://tsqlt.org/user-guide/isolating-dependencies/
    -- También contendrá la declaración de los datos esperados y otras variables que se pueden necesitar
    End

    /*************************************
    ** Act **
    *************************************/
    Begin /*Act*/
    -- Esta sección contendrá la ejecución del código que se espera probar: Sp, función o vista
    -- También contiene la captura de los resultados esperados
    End

    /*************************************
    ** Assert **
    *************************************/
    Begin /*Assert*/
    -- Esta sección compara los datos esperados con los actuales o con un IF llama a tSQLt.Fail
    -- Para más información ver: http://tsqlt.org/user-guide/assertions/
    End
END
GO
```

Conceptos: SetUp

Prepara los datos comunes a toda la clase

Es ejecutado previo a cada test unitario

```
-- ****************************************************************************
-- #About
-- ****************************************************************************
-- #Reviews
-- Date Author Description
-- ---------------- ---------- ----------------------------------------------
-- ****************************************************************************
CREATE PROCEDURE [NombreTestClass].[SetUp]
AS
BEGIN
    -- Ejemplo:
    -- EXEC tSQLt.PrepareTableForFaking @TableName ='Boletos', @SchemaName = 'Boletos'
    -- EXEC tSQLt.FakeTable 'Boletos.Boletos' ,@identity = 1 , @ComputedColumns = 1, @Defaults = 1--quedará vacía
END
```

Ejecución de los test unitarios

`Exec tSQLt.RunAll` → Ejecuta todos los test

`Exec tSQLt.Run 'NombreTestClass'` → Ejecuta la clase 'NombreTestClass'

`Exec tSQLt.Run 'NombreTestClass.Test1'` → Ejecuta sólo el 'Test1' de la clase 'NombreTestClass'

`Exec tSQLt.Run` → Ejecuta el último test ejecutado

tSQLt Isolation Dependencies Objects

Asegura el aislamiento de cada test y prepara los datos del entorno

- FakeTable
- ApplyConstraint
- ApplyTrigger
- FakeFunction
- SpyProcedure
- RemoveObject
- RemoveObjectIfExists

https://tsqlt.org/user-guide/isolating-dependencies/

FakeTable (I)

Copia de una tabla para que no se modifique la real

- Campos nullables
- Sin constraints ni triggers (aplicar luego *ApplyTrigger*, *ApplyConstraint*)
- Vacía

Limitaciones:

- No temporales (#tabla o ##tabla)
- No In-Memory ni Schemabinding
- No se pueden publicar
- ¿Actualizar cuando cambia la tabla? (No siempre)
- No soporta todos los tipos de datos

Sintaxis:

```
tSQLt.FakeTable [@TableName = ] 'table name'
, [[@Identity = ] 'preserve identity']
, [[@ComputedColumns = ] 'preserve computed columns']
, [[@Defaults = ] 'preserve default constraints’]
```

Por defecto sin identities, ni columnas calculadas ni valores por defecto. Poner a 1 si se desea.

FakeTable (II)

Recuperación de los datos originales:

```
SELECT OriginalName, SCHEMA_NAME(schema_id) +'.' + name AS [Name of Renamed Table], create_date
FROM tSQLt.Private_RenamedObjectLog JOIN sys.objects ON objectid = object_id;
```

En ocasiones con tablas *complejas*, se necesita *preparar la tabla*

```
(4 filas afectadas)
[TestEjemplos].[Test FakeTable without PrepareTableForFaking] failed: (Error) Error de .NET Framework durante la ejecución de la rutina o agregado
definido por el usuario "SuppressOutput":
System.Data.SqlClient.SqlException: No se puede cambiar el nombre del objeto '[Esquema].[Tabla]' porque participa en dependencias forzadas.
System.Data.SqlClient.SqlException:
en System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction)
```

TIP

PrepareTableForFaking Implementado fuera de tsqlt.org y disponible en GitHub. Elimina las limitaciones del *FakeTable*

SpyProcedure

Aísla el SP que se está probando cuando existen SPs anidados

- Recoge los parámetros que recibe en la tabla *NombreSP_SpyProcedureLog*
- Devuelve como resultado lo que se indique en el param *@CommandToExecute*

Limitaciones:

- Máximo 1020 parámetros
- El nombre del SP no puede contener 'SpyProcedure'
- No procedimientos temporales *(que empiecen por #)*

Sintaxis:

```
tSQLt.SpyProcedure [@ProcedureName = ] 'procedure name'
[, [@CommandToExecute = ] 'command' ]
```

DEMO

Preparación (Arrange)

tSQLt Assertions

Comprobación de resultados

- AssertEmptyTable
- AssertEquals
- AssertEqualsString
- AssertEqualsTable
- AssertEqualsTableSchema
- AssertLike
- AssertNotEquals
- AssertObjectDoesNotExist
- AssertObjectExists
- AssertResultSetsHaveSameMetaData
- Fail

https://tsqlt.org/user-guide/assertions/

AssertEqualsTable

Compara los datos de dos tablas **temporales**

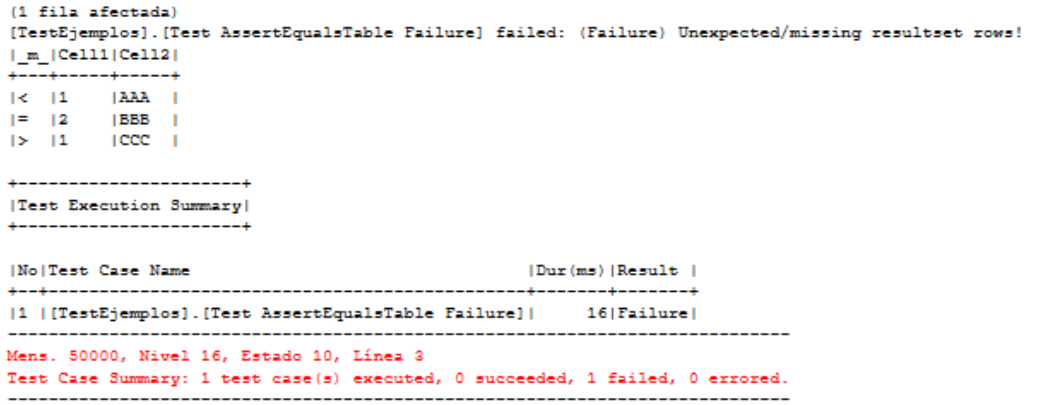
Sintaxis:

```
tSQLt.AssertEqualsTable [@Expected = ] 'expected table name'
, [@Actual = ] 'actual table name'
[, [@FailMsg = ] 'message' ]
```

@FailMsg [Optativo] Mostraría el mensaje declarado en caso de error

En caso de *failure* los resultados se muestran atendiendo a la siguiente leyenda:

- "=": El registro se corresponde
- "<": Registro en *Expected* pero no en *Actual*
- ">": Registro en *Actual* pero no en *Expected*



```
(1 fila afectada)
[TestEjemplos].[Test AssertEqualsTable Failure] failed: (Failure) Unexpected/missing resultset rows!
|_m_|Cell1|Cell2|
+---+-----+-----+
|<  |1    |AAA  |
|=  |2    |BBB  |
|>  |1    |CCC  |

+----------------------+
|Test Execution Summary|
+----------------------+

|No|Test Case Name                                  |Dur(ms)|Result |
+--+------------------------------------------------+-------+-------+
|1 |[TestEjemplos].[Test AssertEqualsTable Failure]|     16|Failure|
------------------------------------------------------------------------------
Mens. 50000, Nivel 16, Estado 10, Línea 3
Test Case Summary: 1 test case(s) executed, 0 succeeded, 1 failed, 0 errored.
------------------------------------------------------------------------------
```

DEMO

Comprobación (Assert)

tSQLt Expectations

Comprueba las excepciones que se generan o no se generan

¡Ojo! Se ponen antes de la ejecución

- ExpectNoException

```
tSQLt.ExpectNoException [ [@Message= ] 'supplemental fail message’]
```

- ExpectException

```
tSQLt.ExpectException
[ [@ExpectedMessage= ] 'expected error message']
[, [@ExpectedSeverity= ] 'expected error severity']
[, [@ExpectedState= ] 'expected error state']
[, [@Message= ] 'supplemental fail message']
[, [@ExpectedMessagePattern= ] 'expected error message pattern']
[, [@ExpectedErrorNumber= ] 'expected error number’]
```

Todos los parámetros son opcionales

https://tsqlt.org/user-guide/expectations/

DEMO

Comprobación (Expectations)

Conclusiones

Con tSQLt…

- ✓ Es más sencillo crear test unitarios que perduren en el tiempo
- ✓ Tienes control sobre los juegos de datos
- ✓ Conociendo la entrada, conoces la salida
- ✓ Tienes herramientas para probar excepciones, resultados, resultsets, SPs anidados…
- ✓ Puedes automatizar las pruebas

¡Aporta más calidad a tu código!

¡Muchas gracias!

cristina.tarabini@gmail.com

@tarabiquetevi