UNIT TESTS ÎN SQL SERVER

STARINSCHI Felicia, LICA Ion, SARANCIUC Dorian

Universitatea Tehnică a Moldovei

Rezumat: *În articol este dedicat descrierii testelor modulare (unit tests) în SQL Server, o parte componentă a instrumentului SQL Server Data Tools. Este prezintată descrierea generală instrumentului. Sînt descrise componentele de bază ale unui test modular, ordinea executării acestora într-un test de unitate. Sînt enumerate tipurile de script-uri şi rolul acestora într-un test modular. Este efectuată o analiză comparativă între utilizarea condiţiilor de test şi folosirea aserţiunilor.*

Cuvinte Cheie: *SQL Server Data Tools, unit tests, script, Transact-SQL, validare test, post-action, pre-action, test-action.*

1. Privire generală asupra SQL Server Data Tools

SQL Server Data Tools (SSDT) transformă modelarea bazelor de date prin introducerea unui model nou, declarativ care cuprinde toate fazele de dezvoltare a bazelor de date în interiorul Visual Studio. Este posibilă utilizarea capacităţilor de proiectare SSDT Transact-SQL pentru a construi, depana, menţine, şi reconstrui bazele de date. Se poate lucra cu un proiect de bază de date, sau direct cu o instanţă de bază de date.

Dezvoltatorii pot folosi instrumente familiare Visual Studio pentru dezvoltarea bazelor de date. Instrumente disponibile sînt: navigarea de cod, IntelliSense, suport pentru limbajul paralel ceea ce este disponibil pentru C # şi Visual Basic, depanarea şi editarea declarativă în editorul Transact-SQL. SSDT oferă, de asemenea un tabel Designer vizual pentru crearea şi editarea tabelelor fie în proiecte de baze de date sau instanţe de baze de date conectate. În timp ce se lucrează la proiecte de bază de date într-un mediu bazat pe echipe, puteţi folosi controlul de versiune pentru toate fişierele, pentru a le repartiza printre membri şi pentru a spori calitatea creării bazelor de date.

SQL Server Object Explorer în Visual Studio oferă o vedere a obiectelor bazei de date similar ca şi în SQL Server Management Studio. SQL Server Object Explorer uşurează administrarea şi proiectarea bazelor de date uşor. Puteţi crea, edita, redenumi şi şterge tabele, proceduri stocate, tipuri şi funcţii. Deasemenea, putem edita datele din tabel, compara scheme, sau executa interogări cu ajutorul meniurilor contextuale chiar de la SQL Server Object Explorer.

2. Testele de unitate în SQL Server

Testele de unitate (Unit Tests) în SQL Server se rulează pentru a verifica modificările aduse de unul sau mai multe obiecte, introduse în schema bazei de date a sistemului nostru, pentru a vedea dacă există nereguli de funcţionare ale aplicaţiei, ce lucrează cu o anumită bază de date existentă. Aceste teste completează setul de teste unitare, elaborate de către dezvoltatorii softului. Astfel, pentru a verifica comportamentul aplicaţiei, trebuie să se efectuieze ambele tipuri de teste.

Comportamentul fiecărui obiect din schema bazei de date este verificat prin adăugarea unui test unitar SQL Server şi prin adăugarea unui script Transact-SQL pentru testarea acestui obiect. De asemenea, se poate genera automat un script-dop Transact-SQL, dacă este nevoie de testat comportamentul unei anumite funcţii, a unui trigger, sau a unei anumite proceduri.

Crearea unui test gol, adăugarea unui anumit cod în acesta ca şi rularea lui este posibilă fără a avea acces la o anumită bază de date. Dar generarea automată a unui dop Transact-SQL care testează o funcţie, un trigger, sau o procedură memorată fără a deschide proiectul care conţine acest obiect este imposibilă.

3. SQL Server Unit Test Files

La fel ca şi testele unitare pentru un cod gestionat, teste de unitate SQL Server se amplasează în proiectele de testare. Putem vedea elementele care compun o unitate de testare SQL Server în ierarhia unui proiect de testare în Solution Explorer.

O unitate de testare SQL Server este formată din mai multe elemente, care sunt cuprinse în mai multe fişiere (tab.1).

Tabelul 1 – Fişiere ale unitîţii de testare

Fișier	Descriere
.cs sau **.vb**	Acest fișier cod-sursă conține o clasă care este completată cu atribute a [TestClass]. Această clasă are o singură metodă de testare pentru fiecare modul de test prevăzut în SQL Server. Aceste metode sunt decorate cu atributele [TestMethod].
.resx	Acest fișier de resurse conține scripturile Transact-SQL pentru toate testele în fișierele asociate .cs/.vb. Acest grup de scripturi include scriptul testului preliminar, scriptul testului și scriptul post-testului. Fișierul de resurse conține XML, care poate fi modificat. Fișierul resursă este compilat în interiorul ansamblatorului de test.
app.config	Acest fișier stochează conexiunile dintre bazele de date pentru proiectul de test și alte setări suplimentare de configurare a testului dat.

4. Script-uri în SQL Server Unit Tests

Fiecare Unit Test în SQL Server conține un pre-test action, test action și post-test action. Fiecare dintre aceste acțiuni, la rândul său conțin următoarele elemente:

- un script Transact-SQL care se execută pe o bază de date.
- zero sau mai multe condiții de teste care evaluează rezultatele returnate de executarea script-lui.

Scriptul de test Transact-SQL în test action este singura componenta care trebuie inclusă în fiecare unitate de testare SQL Server. În plus față de scriptul de test propriu zis putem să specificam condițiile de testare pentru a verifica dacă scriptul de test a returnat valoarea sau setul de valori așteptat. Testele de unitate antrenează sau schimbă un anumit obiect în baza de date și apoi evaluează această schimbare.

Pentru fiecare test, putem să includem o acțiune pre-test și o acțiune post-test. Similar cu acțiunea de testare, fiecare acțiune pre-test și fiecare acțiune post-test conține un script Transact-SQL și zero sau mai multe condiții de testare. Putem utiliza o acțiune pre-test pentru a ne asigura că baza de date este într-o stare validă ca să permită unei acțiuni de test să ruleze și să întoarcă rezultate semnificative. De exemplu, putem seta o acțiune pre-test care verifică dacă un tabel conține date înainte ca scriptul de test să efectueze o operațiune pe aceste date. După ce acțiunea pre-test pregătește baza de date și acțiunea de testare returnează rezultate semnificative, acțiunea post-test poate fi folosită pentru a reveni la starea inițială a bazei de date înainte de a fi efectuată acțiunea pre-test.

În unele cazuri, putem folosi o acțiune post-test pentru a valida rezultatele acțiunii de testare. Acest lucru se datorează faptului că acțiunea post-test poate avea privilegii mai mari asupra bazei de date decât acțiunea de testare.

Suplimentar, acestor trei acțiuni, există, două scripturi de testare (menționate la scripturi comune), care rulează înainte și după fiecare test unitar SQL Server. Ca urmare, până la cinci scripturi Transact-SQL pot fi rulate în timpul execuției unui singur Unit Test SQL Server. Astfel este nevoie obligatoriu de scriptul Transact-SQL care este în cadrul acțiunii de testare; scripturile comune, script-urile pre-test și post-test fiind opționale.

Initialization and Cleanup Scripts

Scripturile date sunt folosite pentru pregătirea bazei de date pentru testare, și totodată revenirea bazei de date la starea inițială după ce a fost executat însuși testul.

Scripturi Pre-test și post-test

Script-urile care sunt asociate cu acțiuni pre-test și post-test sunt capabile să varieze de la un test unitate la altul. Astfel, putem folosi aceste script-uri pentru a stabili modificări incrementale la baza de date și apoi curăța aceste schimbări.

5. Utilizarea Test Conditions în SQL Server Unit Tests

Într-un test de unitate SQL Server, sunt executate unul sau mai multe scripturi de testare Transact-SQL. Rezultatele pot fi evaluate în cadrul scriptului Transact-SQL, unde se pot specifica condiții ca THROW sau RAISERROR pentru a returna o eroare și a invalida testul, sau se pot specifica condițiile de testare ce vor evalua rezultatele. Testul returnează o instanță a clasei SqlExecutionResult. Instanța acestei clase conține una sau mai multe seturi de date, timpul de execuție, precum și rândurile afectate de script-ul executat. Toate aceste informații sunt colectate în timpul executării scriptului. Aceste rezultate pot fi evaluate prin folosirea condițiilor de testare. SQL Server Data Tools oferă un set de condiții de testare predefinite.

Condițiile de testare, numite Test Condition, care sunt predefinite, sunt următoarele: Data Checksum, Empty ResultSet, Execution Time, Expected Schema, Row Count, Scalar Value. În continuare prezentăm o descriere succintă a fiecărei condiții de test:

– *Data CheckSum:* are loc cînd suma de control pentru setul de rezultate returnat corespunde cu suma de control aşteptată;

– *Empty ResultSet:* eşuează atunci când setul de rezultate returnat nu este null.

– *Execution Time:* eşuează dacă scriptul Transact-SQL se execută mai mult decât timpul aşteptat. Timpul de execuţie implicit este de 30 secunde. Timpul de execuţie se aplică doar la scriptul de test propriu-zis.

– *Expected Schema:* eşuează dacă coloanele şi tipurile de date a setului de rezultate nu corespunde cu cele specificate pentru condiţia de test.

– *Inconclusive:* condiţia de bază a fiecărui test. Aceasta condiţie este inclusă pentru a indica că verificarea testului nu a fost implementată.

– *Row Count:* eşuează dacă setul returnat de tupluri nu corespunde cu cel iniţial.

De asemenea putem crea teste de unitate a cărui comportament este conceput de la bun început să dea eroare de executare. Aceste tipuri de erori aşteptate fac parte din testarea negativa. Exemple unde se poate folosi acest tip de testare pot fi:

– verificarea eşuării unei proceduri stocate care şterge datele unui client care are un ID de client nevalid.

– verificarea eşuării unei proceduri stocate care umple un ordin, nu dacă ordinul nu a fost plasat sau dacă ordinul a fost deja umplut.

6. Utilizarea ”Transact-SQL Assertions” în SQL Server Unit Tests

Într-un unit test SQL Server, un script de test Transact-SQL se execută şi returnează un rezultat. Uneori, rezultatele sunt returnate ca un set de rezultate. Putem valida rezultatele folosind condiţiile de testare. De exemplu, avem posibilitatea să utilizăm o condiţie de test pentru a verifica cît de multe rînduri au fost returnate într-un set de rezultate specifice sau pentru a verifica cît de mult timp un anumit test a rulat.

Deseori, în loc de a folosi condiţiile de testare, putem utiliza, de asemenea, afirmaţii Transact-SQL, care sunt THROW şi RAISERROR. În anumite circumstanţe, se preferă de utilizat o declaraţie Transact-SQL în loc de o condiţie de test.

Criteriile de utilizare a Transact-SQL Assertions

Trebuie de luat în considerare următoarele puncte înainte de a decide validarea datelor, fie prin utilizarea afirmaţiilor Transact-SQL sau prin utilizarea condiţiilor de test.

Performanţă. Este mai rapid pentru a rula o afirmaţie Transact-SQL pe server decât să transferi datele pe un computer client şi apoi să le manipulezi la nivel local.

Familiarizarea cu limbajul. S-ar putea prefera un anumit limbaj bazat pe experienţa curentă şi, prin urmare, se pot utiliza afirmaţii Transact-SQL, Visual C # sau Visual Basic precum condiţii de testare.

Validare complicată. În unele cazuri, pot fi construite mai multe teste complexe, în Visual C # sau Visual Basic şi validate testele pe client.

Simplitate. Este de multe ori mai simplu de a utiliza o pre-condiţie definită de testare decât să se scrie scenariul echivalent în Transact-SQL.

Biblioteci de validare Legacy. Dacă există deja codul care efectuează validarea, îl putem folosi într-o unitate de testare SQL Server în loc de a folosi condiţii de testare.

7. Privire generală asupra ”Connection Strings and Permissions”

Pentru a rula unit testele SQL Server, trebuie să ne conectăm la un server cu baza de date, utilizînd unul sau două şiruri de conectare specifice, numite connection strings. Fiecare şir de conexiune reprezintă un cont care are permisiuni specifice pentru a efectua sarcina sau un set de sarcini într-un anumit script, ca parte a testului. Puteţi specifica aceste şiruri de caractere în caseta de dialog SQL Server Test Configuration sau prin editarea manuală a fişierului app.config pentru proiectul dvs. de test.

Cînd se specifică şiruri de conexiune, trebuie de ales între autentificarea Windows sau autentificarea SQL. Un motiv pentru a alege autentificarea Windows este că suportă utilizarea testelor de către o echipă mai bine decât autentificarea SQL Server. Dacă alegem autentificarea SQL Server, şirurile de caractere de conectare sunt criptate, folosind API-ul pentru Protecţia Datelor (DPAPI), în funcţie de datele de utilizator. Acest lucru înseamnă că testele din acest proiect de test vor fi rulate doar de o singură persoană, nu de toţi membrii echipei.

8. Concluzii

Instrumentul SQL Server Data Tools oferă un set larg de posibilități pentru proiectarea și gestionarea bazelor de date. O funcționalitate care oferă siguranță și permite testarea bazelor de date pentru identificarea erorilor sunt testele modulare. Acestea permit depistarea erorilor și corectarea lor la o stare incipientă. Verificarea tuplurilor, coloanelor, schemei unei baze de date ne asigură funcționarea corectă și stabilă a acesteia.

Bibliografie

1. SQL Server Data Tools. [Resursă electronică]. Regim de acces: https://msdn.microsoft.com/en-us/library/hh272686(v=vs.103).aspx

2. Creating and Defining SQL Server Unit Tests. [Resursă electronică]. Regim de acces: https://msdn.microsoft.com/en-US/library/jj851203(v=vs.103).aspx

3. Andrey Zavadskiy. Модульное тестирование с помощью SQL Server Data Tools. [Resursă electronică]. Regim de acces: http://www.sqlpass.org/24hours/2015/russian/%D0%94%D0%BE%D0%BA%D0%BB%D0%B0%D0%B4%D1%8B/Details.aspx?sid=7667