Database Testing with Visual Studio 2010

Amit Bansal

Sr. Consultant Trainer (PeoplewareIndia.com)

President (SQLServerGeeks.com)

About Amit Bansal

- CTO, eDominer Systems & Peopleware India
- Corporate Trainer/Consultant & Evangelist
- Conducted more than 400 workshops on SQL Server & BI for top notch IT companies world wide
- Microsoft MVP for SQL Server
- Microsoft Certified Trainer Advisory Council member
- Speaker at TechED India, TechED US & TechED Europe
- Technical Reviewer – MSL courses on SQL Server
- SME – SQL Server 2008 certifications
- President – www.SQLServerGeeks.com | @SQLServerGeeks
- www.amitbansal.net | @A_Bansal

Database Testing - Introduction

- RDBMs often persist mission-critical data
- Updated by many application, thousands of users
- DBs often implement logic/code/methods in form of functions, SPs, triggers, etc
- We need to ensure continuing quality of these pieces of code
- Focus: fully automated, continuous regression testing

Why Test an RDBMS?

- **Current approaches aren't sufficient.**
  - Data professionals to control changes to the database schemas
  - Developers to visually inspect the database during construction, and to perform some form of formal testing during the test phase at the end of the lifecycle.

    Unfortunately, none of these approaches prove effective. Application developers will often go around their organization's data management group because they find them too difficult to work with, too slow in the way they work, or sometimes they don't even know they should be working together. We all know the end result.

Why Test an RDBMS?

- **Testing provides the concrete feedback required to identify defects.**
  - How do you know how good the quality of your source data actually is without an effective test suite which you can run whenever you need to?
- **Support for evolutionary development.**
  - Many evolutionary development techniques, in particular database refactoring, are predicated upon the idea that it must be possible to determine if something in the database has been broken when a change has been made. The easiest way to do that is to simply run your regression test suite.

Why Test an RDBMS?

- **If someone is still not convinced about DB testing?**
  - If you're implementing code in the DB in the form of stored procedures, triggers, ... shouldn't you test that code to the same level that you test your app code?
  - Think of all the data quality problems you've run into over the years. Wouldn't it have been nice if someone had originally tested and discovered those problems before you did?
  - Wouldn't it be nice to have a test suite to run so that you could determine how (and if) the DB actually works?

What should we Test?

Black-Box Testing at the Interface	White/Clear-Box Testing Internally Within the Database
•O/R mappings (including the meta data) •Incoming data values •Outgoing data values (from queries, stored functions, views ...)	•Scaffolding code (e.g. triggers or updateable views) which support refactorings •Typical unit tests for your stored procedures, functions, and triggers •Existence tests for database schema elements (tables, procedures, ...) •View definitions •Referential integrity (RI) rules •Default values for a column •Data invariants for a single column •Data invariants involving several columns

What should we Test?

- **Functionality Testing in Relational Databases**
  - Stored procedures and triggers
- **Relationship Testing in Relational Databases**
  - Referential integrity (RI)

    cascading deletes

    Existence rules

What should we Test?

- **Data Quality Testing in Relational Databases**
  - Default values
  - Data invariants
  - Validate the attribute size
  - Data Formats
  - De-Duplicating source/incoming data
    - Exact De-Duplication
    - Fuzzy De-Duplication
  - Lookups for incoming data
    - Exact Lookups
    - Fuzzy Lookups

What should we Test?

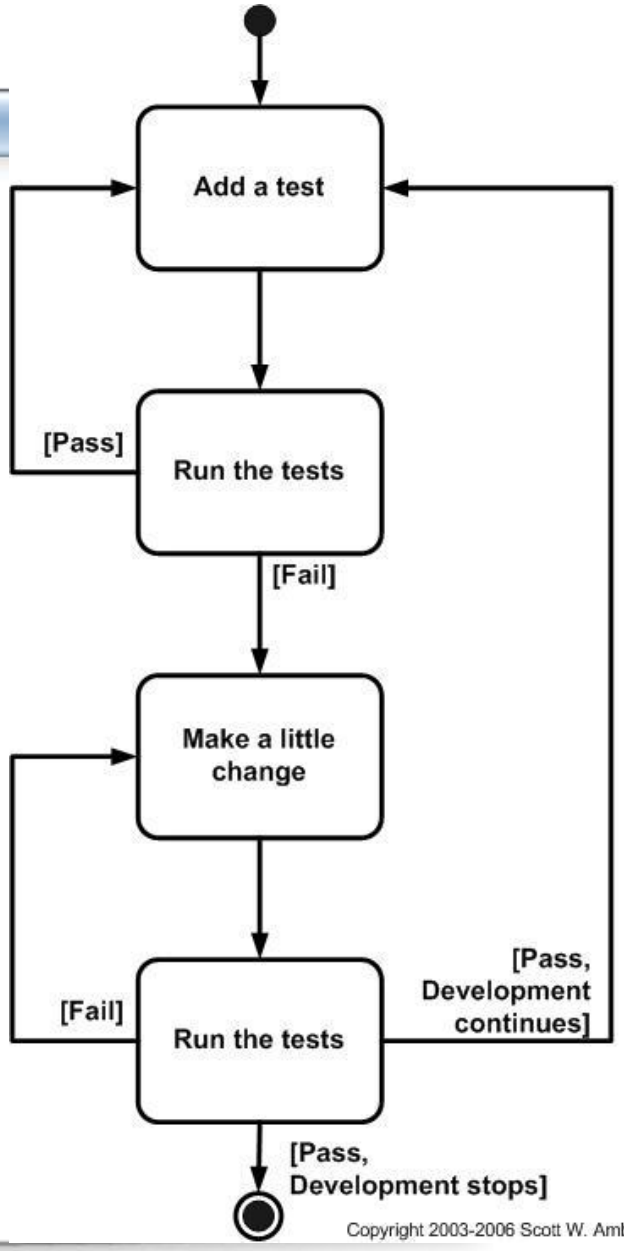
- **Performance Testing of Relational Databases**
  - Access time to read/write/delete a single row.
  - Access time for common queries returning multiple rows.
  - Access time for queries involving several tables.
  - Existence test for an index. Does the expected index exist or not?
  - Index performance?
  - IO Subsystem test
  - Stress/Load test

What should we Test?

- **Structural Testing in Relational Databases**
  - Table existence. We can check whether all the data from the application is being inserted into the database properly, or not
  - View definitions. Views often implement interesting business logic. Things to look out for include: Does the filtering/select logic work properly? Do you get back the right number of rows? Are you returning the right columns? Are the columns, and rows, in the right order?

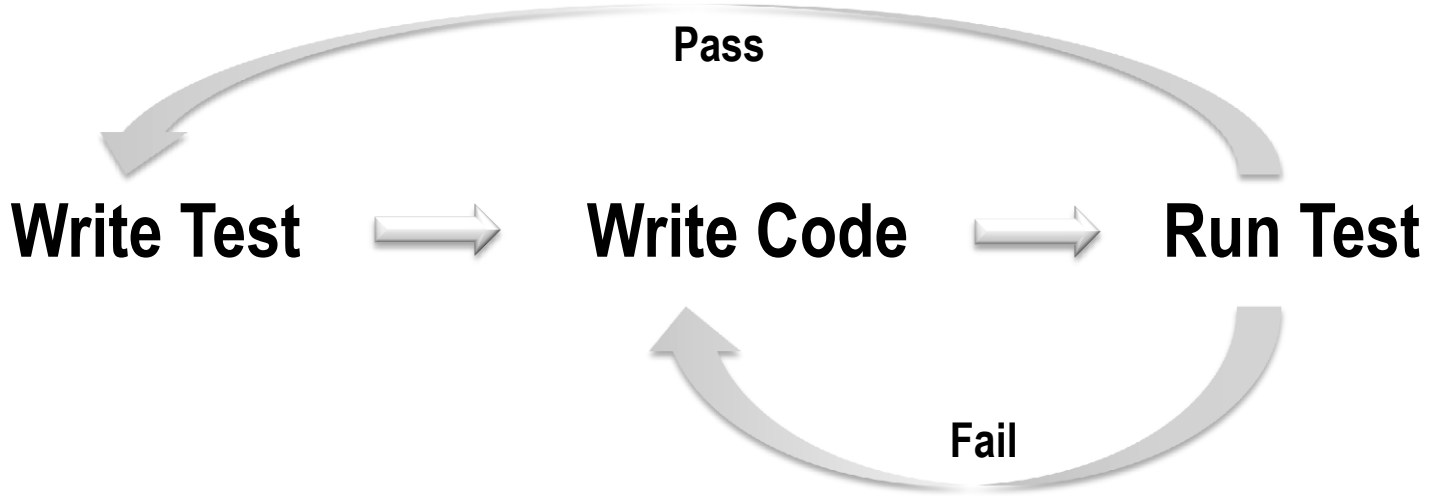
When should we Test?

- **The process of Test First Development (TFD)**
  - Add a test knowing it will fail
  - Run your tests and surely it will fail :)
  - Update your code
  - Run the test again
  - If fail again, update and re-test
  - One the test passes, start over again



What exactly is TDD?

- Is it a testing methodology?
- TDD is a software development methodology



What exactly is TDD?

```
CREATE TABLE dbo.StockPrice
(
StockId INT IDENTITY(1,1) PRIMARY KEY,
Symbol VARCHAR(10) NOT NULL,
ClosePrice MONEY NOT NULL,
CloseDate DATETIME NOT NULL
)
```

What exactly is TDD?

```
CREATE PROC dbo.ut_TestGetAveragePriceBySymbol
  AS
  SET NOCOUNT ON -- Setup the test conditions by inserting
  test data
  INSERT INTO dbo.StockPrice VALUES ('XYZ', 10, GETDATE() - 2)
  INSERT INTO dbo.StockPrice VALUES ('XYZ', 15, GETDATE() - 1)
  INSERT INTO dbo.StockPrice VALUES ('XYZ', 5, GETDATE())
  INSERT INTO dbo.StockPrice VALUES ('PDQ', 100.00, GETDATE())
  -- Exercise the test
  DECLARE @ActualAvgClosePrice MONEY
  EXEC dbo.GetAveragePriceBySymbol 'XYZ', @ActualAvgClosePrice
  OUT -- Assert expectations
  DECLARE @ExpectedAvgClosePrice MONEY
  SET @ExpectedAvgClosePrice = 10 --(10 + 15 + 5) / 3 = 10
  IF (@ExpectedAvgClosePrice != @ActualAvgClosePrice)
  EXEC dbo.tsu_Failure 'GetAveragePriceBySymbol failed.' --
  Teardown
  -- Implicitly done via ROLLBACK TRAN
  GO
```

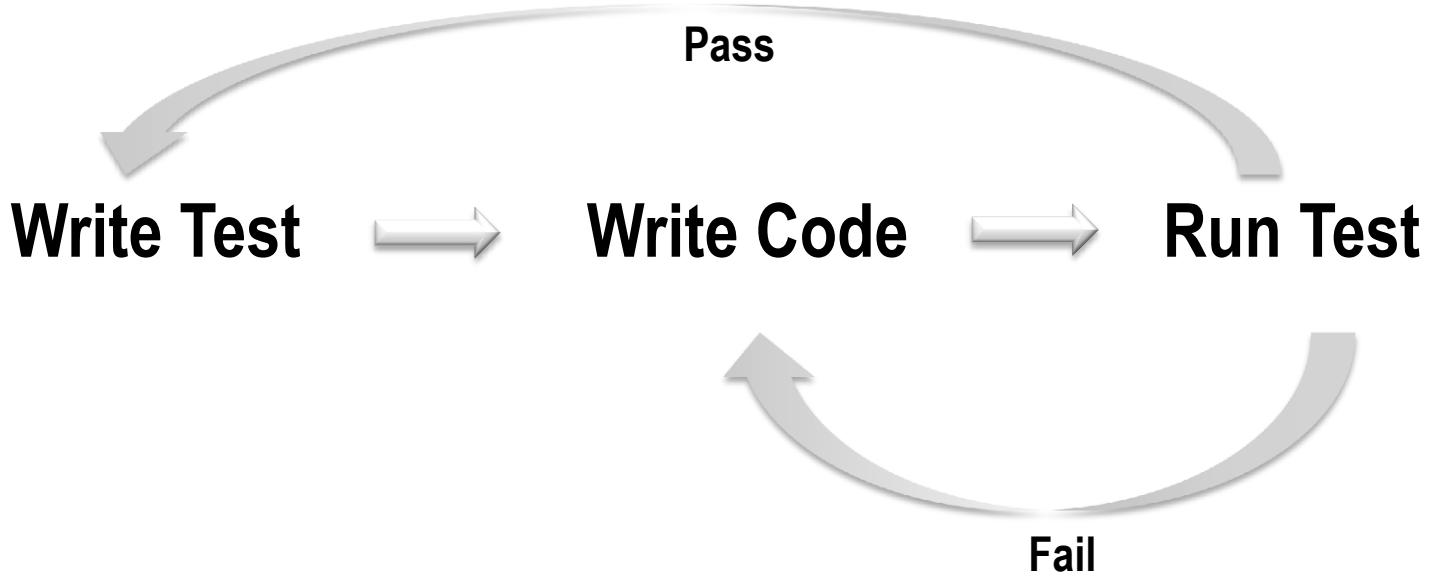
What exactly is TDD?

```
CREATE PROCEDURE
 dbo.GetAveragePriceBySymbol
 @Symbol VARCHAR(10),
 @AvgClosePrice MONEY OUT
 AS
 SET NOCOUNT ON

 SELECT @AvgClosePrice =
 AVG(ClosePrice)
 FROM dbo.StockPrice
 WHERE Symbol = @Symbol
```

What exactly is TDD?

- Is it a testing methodology?
- TDD is a software development methodology



Test Driven Database Development

- If TDD works for application development, shouldn't it also work for database development?

Test Driven Database Development

- **TFD approach**
  - 1. Quickly add a test. You basically need just enough code to fail, typically a single test.
  - 2. Run your tests. You will often need the complete test suite although for sake of speed you may decide to run only a subset, to ensure that the new test does in fact fail.
  - 3. Update your production code. Do just enough work to ensure that your production code passes the new test(s).
  - 4. Run your tests again. If they fail you need to update your production code and retest. Otherwise go back to Step #1.

Test Driven Database Development

- Just as agile application developers take a quality-driven, TDD approach to software development so can agile database developers.
- This requires the adoption of new techniques, in particular database refactoring and database regression testing.

Test Driven Database Development

DEMO

Summary

- Start embracing testing framework in Databases !!

Continue your learning:

- www.SQLServerGeeks.com is organizing SQL Server virtual conference soon- Register now to be notified !
- http://www.amitbansal.net
- Twitter – http://www.twitter.com/A_Bansal
- Twitter – http://www.twitter.com/SQLServerGeeks

Q&A

Thank you ☺

for suggestions, please email at community@peoplewareindia.com