Unit Testing T-SQL code with tSQLt

Advanced Topics

Dave Green
Twitter: @d_a_green
Email: dave@dgta.co.uk

pluralsight
hardcore developer training

Testing Multiple Result Sets

- **One result set:**
  - tSQLt.AssertEqualsTable
  - tSQLt.AssertEmptyTable
  - tSQLt.AssertResultSetsHaveSameMetaData

What if my result set has more than one result set?

Getting result sets

```
INSERT INTO MyTable
EXEC dbo.MyProc
```

Puts all results into the table (UNION ALL)

Only works if all result sets are the same format

Get a specific result set

tSQLt.ResultSetFilter

Returns ONE (specified) result set

Multiple result sets = multiple execution of code under test

Insert…Exec

- **Not best practice to use inside stored procedures**
- **Can't be nested**
- **If a SP uses it, we cannot use either of the previous methods to get the result set inside tSQLt**
- **Workaround possible with OPENQUERY**
- **BUT:**
  - Syntax different between SQL versions (SET FMTONLY OFF)
  - Can be tricky to get right
  - May need setup on development/test servers.

What are we missing?

By isolating tables we remove

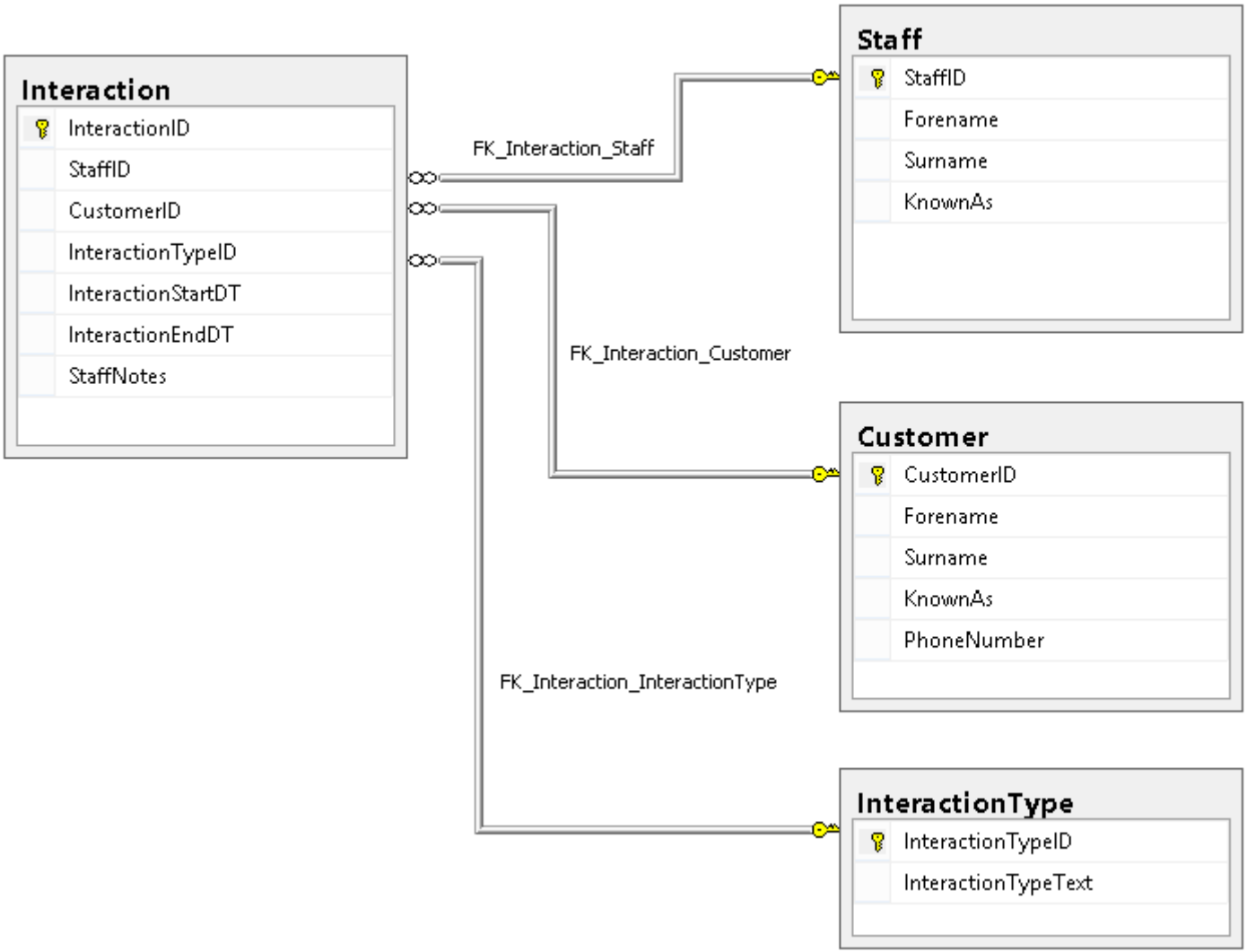
- **Data**
- **NULL / NOT NULL**
- **Optionally – computed columns, identity columns, default values**

- **Check constraints**
- **Foreign Key constraints**
- **Triggers**

Testing constraints

- **Foreign keys and check constraints need to be tested**
- **Not part of table definition but relate to data integrity**
- **Test one by one (so you know what's wrong)**
  - Remove all
  - Add back in

CustomerManagement Database



tSQLt.ApplyTrigger

- Works in a similar way to tSQLt.ApplyConstraint
- Fake the table, removing all constraints / triggers
- Add the trigger back in (so you are only testing one at a time)

Complicated logic

Sometimes a requirement is more complex:

“x AND y must be true”

“ x OR y must be true”

Complicated logic

tSQLt can help you to assert custom logic

Complicated logic

A test is a stored procedure

Plan your logic and call tSQLt.Fail unless your conditions are met.

Note – this sort of logic is more susceptible to bugs – be sure to peer review it.

Development standards

- **We can use tSQLt to enforce development standards**
- **Test naming convention – test class name must exist as object**
- **What else could you enforce?**
  - Procedures named sp_?
  - Tables without a primary key?
  - Unnamed constraints?
  - Procedures without tests?
  - Development environment configured to desired specs?

The tSQLt transaction

tSQLt runs each test in a transaction

This transaction is rolled back after the test

How can you perform an action outside of this transaction?

tSQLt.NewConnection

What could you use it for?

- **Testing Readpast hints**
- **Logging additional data from tests into a table**

Summary

- Stored procedures with multiple data sets
- Testing Constraints and Foreign Keys
- Enforcing development standards
- Performing work outside of the test transaction

- tSQLt can help you to gain confidence that your code is robust and meets your requirements