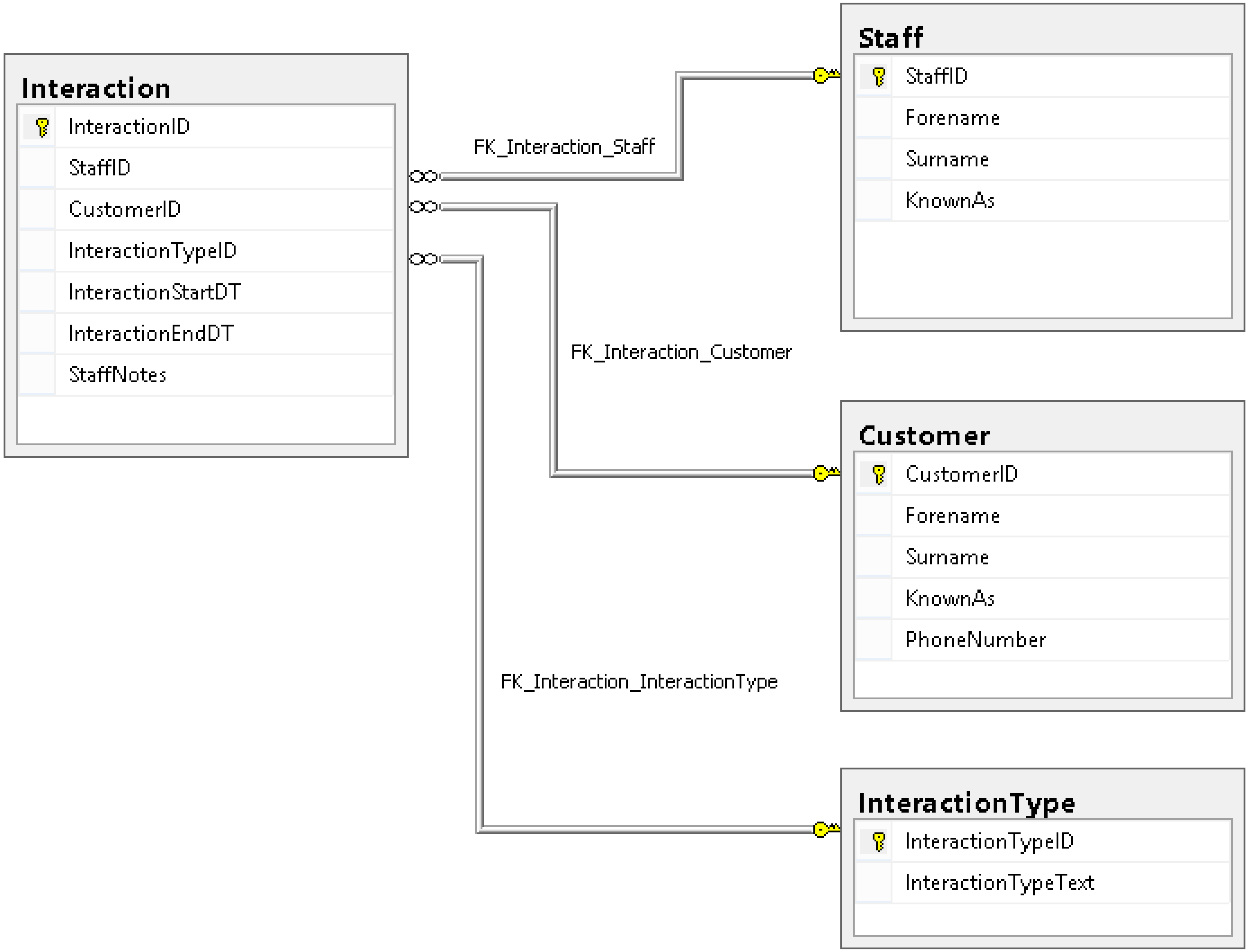
Unit Testing T-SQL code with tSQLt

How to write your first unit test

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

hardcore developer training

CustomerManagement Database



Interaction

- InteractionID
- StaffID
- CustomerID
- InteractionTypeID
- InteractionStartDT
- InteractionEndDT
- StaffNotes

FK_Interaction_Staff

FK_Interaction_Customer

FK_Interaction_InteractionType

Staff

- StaffID
- Forename
- Surname
- KnownAs

Customer

- CustomerID
- Forename
- Surname
- KnownAs
- PhoneNumber

InteractionType

- InteractionTypeID
- InteractionTypeText

Incomplete tests

- **Test procedure created before test is completely working**

- **Test written in advance of procedure**

- **tSQLt *assumes* the test passes**

  … Until it is explicitly told the test has failed

  … Or an assertion is called which is not proved to be correct

- **If no assertions are called, this can create a false impression of success**

Report contacts and average duration

- **For Each Contact Type**
  - How many contacts have taken place
  - How much time in total was used?

- **From this the presentation layer can generate average (mean) contact time, and order appropriately.**
- **Example output:**

InteractionTypeText	Occurrences	TotalTimeMins
Meeting	150	50000
Introduction	200	20450
Phone Call	230	34572

Possible tests

- **Does the solution exist?**
- **Does it have the right output format?**
- **Is the output what you would expect?**
  - … if there is no data to display?
  - … if there is only a small amount of data to display?
  - … if there is a large quantity of data to summarize?

Testing our output

- Data table to be returned
- Expected set of data
- Capture output of object under test
- Assert they are the same
- See if our assertion is met

Summary

- Introduction to the database and tables
- Implemented our first test
- Implemented a test of output data
- False test passes
- Our tests initially failed, and we made them pass