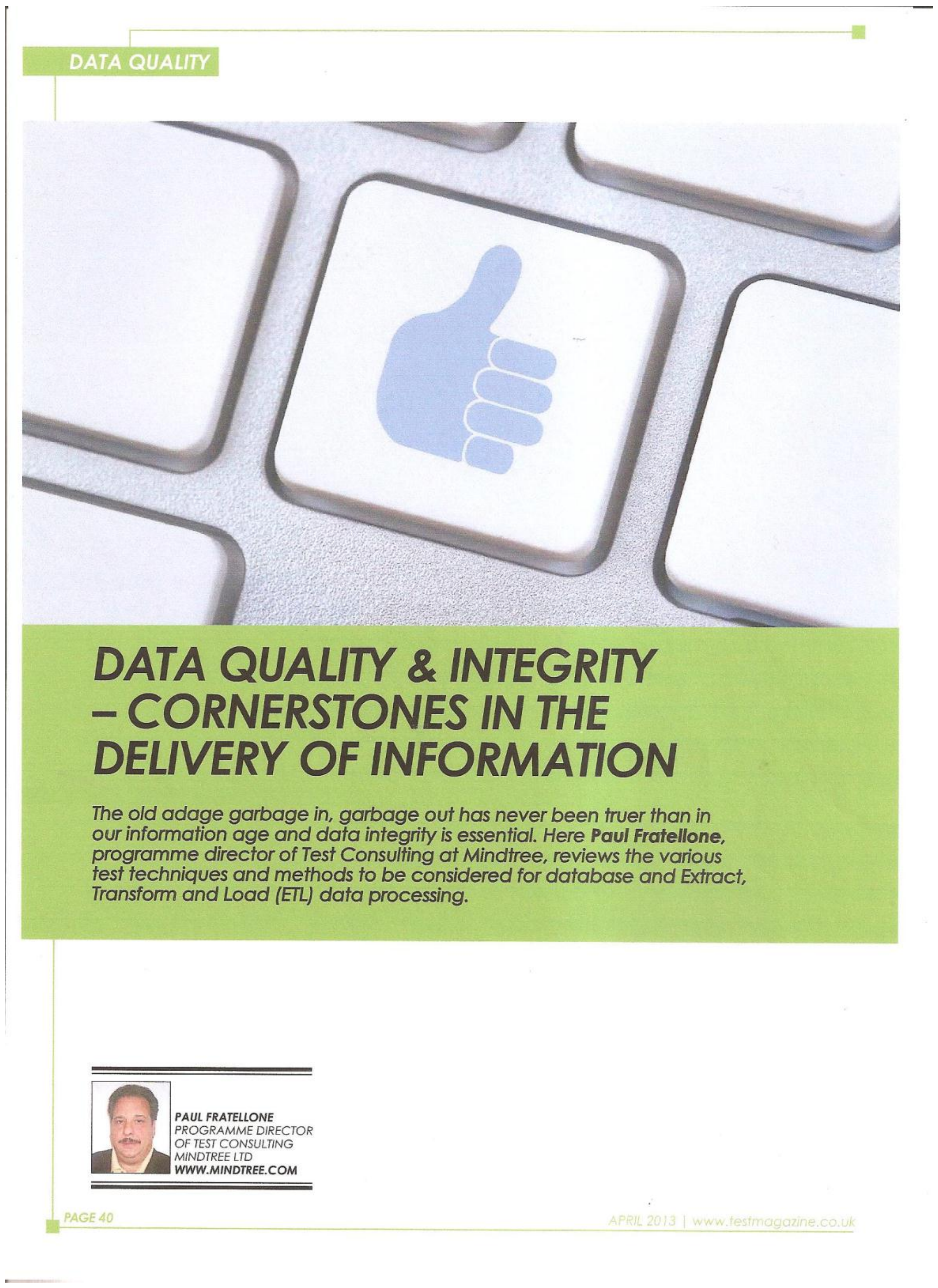
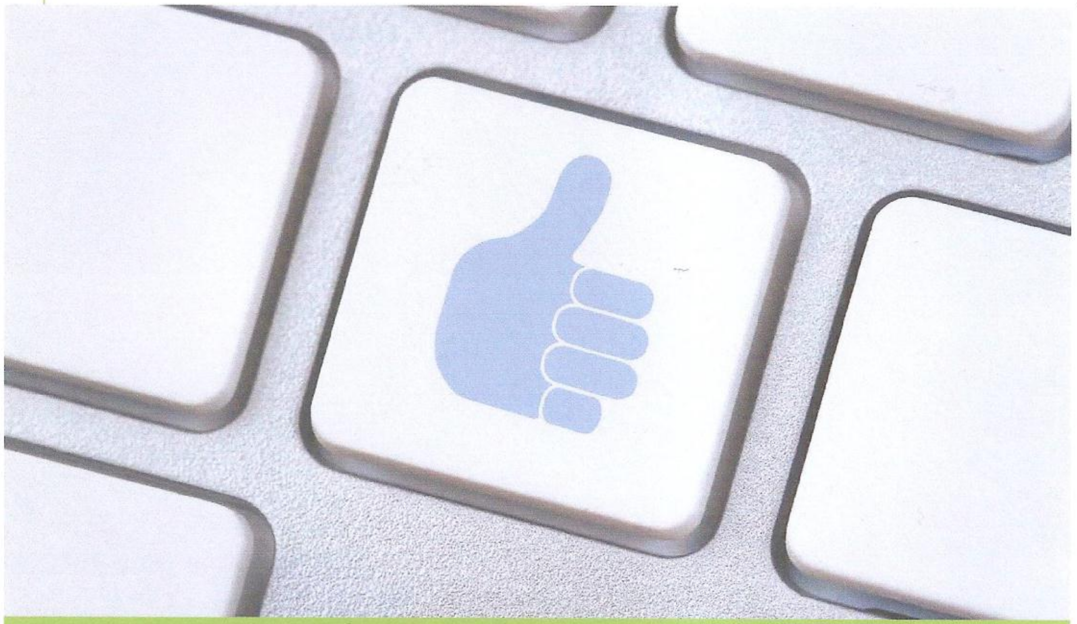
Publication: Data quality
Date: April, 2013





DATA QUALITY & INTEGRITY – CORNERSTONES IN THE DELIVERY OF INFORMATION

*The old adage garbage in, garbage out has never been truer than in our information age and data integrity is essential. Here **Paul Fratellone,** programme director of Test Consulting at Mindtree, reviews the various test techniques and methods to be considered for database and Extract, Transform and Load (ETL) data processing.*

PAUL FRATELLONE
PROGRAMME DIRECTOR
OF TEST CONSULTING
MINDTREE LTD
WWW.MINDTREE.COM

In our age of information and intense technology marketing it is sometimes helpful to take a step back from the hype and review the basics. I was recently engaged with a client about the cost of testing and before I answered him I asked what the cost of failure is? How much (testing effort/cost) should we really invest to ensure quality or to meet user requirements and is it really important to have accurate and reliable data?

As we discussed the topic, I introduced the reality of risk. What is the risk to your investment/company in the event of failure, or some degree of failure? How much of an appetite for risk do you have and we could always test less and reduce your cost/effort. In this particular case, the client was making an investment in business information and analytics along with major database changes. ETL (Extract, Transform and Load) was going to be a major part of the initiative. In this article I will review the various test techniques and methods to be considered for database and ETL data processing.

APPLICATIONS, FILES AND DATABASES

I have never worked on an application or system that had a tolerance for inaccurate data. Consistency, reliability and accuracy are always mandated. Figure 1 identifies the major testing control points, (stars) that require validation of the data. Making sure the data is correct (quality & integrity) needs to be validated from any transactions initiated from the GUI or by means of some ETL process from an existing database or external file.

Needless to say query capabilities should be part of every tester's toolbox and those of you who do not have this skill, address it as standard good testing practice is to validate the database impact of any GUI/application transaction/ETL. We perform this via querying the database for the test condition we created. Adding, deleting modifying records are recorded properly in the database and that is in addition to any referential integrity that needs to be maintained for those actions.

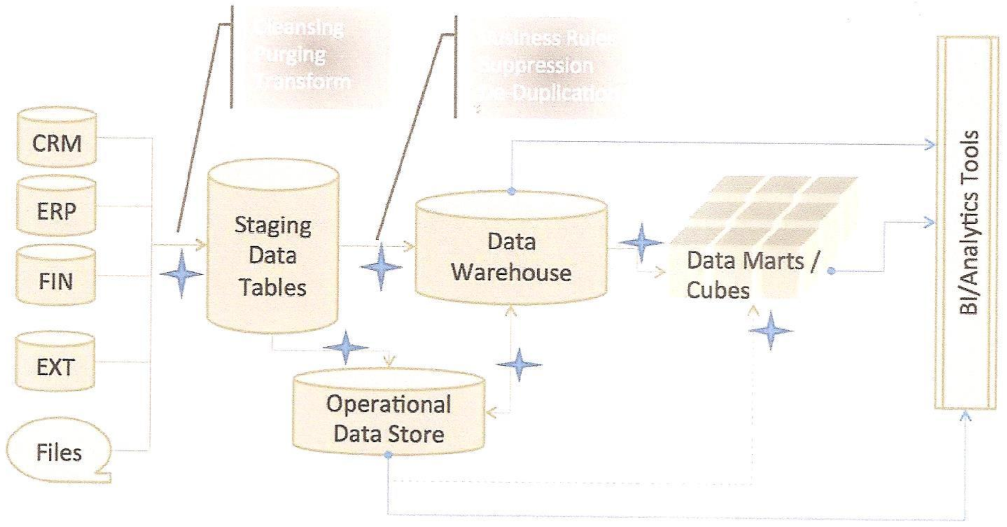
I once worked for a company that had three channels of delivery for their software platform. They had a desktop, a web-based version and EDI (electronic data interface). Each channel had its own code base and different validation rules so many of the tests did not apply across the platform. Even though there were these technical differences, our standard test approach for validating data quality and integrity were applicable to all delivery channels and specific challenges and additional effort were addressed for the EDI channel. These were formatted message files that were very cumbersome to create and then test.

Good testing practice addresses quality dimensions and objectives regardless of the delivery mechanism and indeed, when I speak of quality dimensions we group them into some of these examples such as reliability, stability, accuracy, performance, scalability, etc. In our discussion of data we need to include completeness.

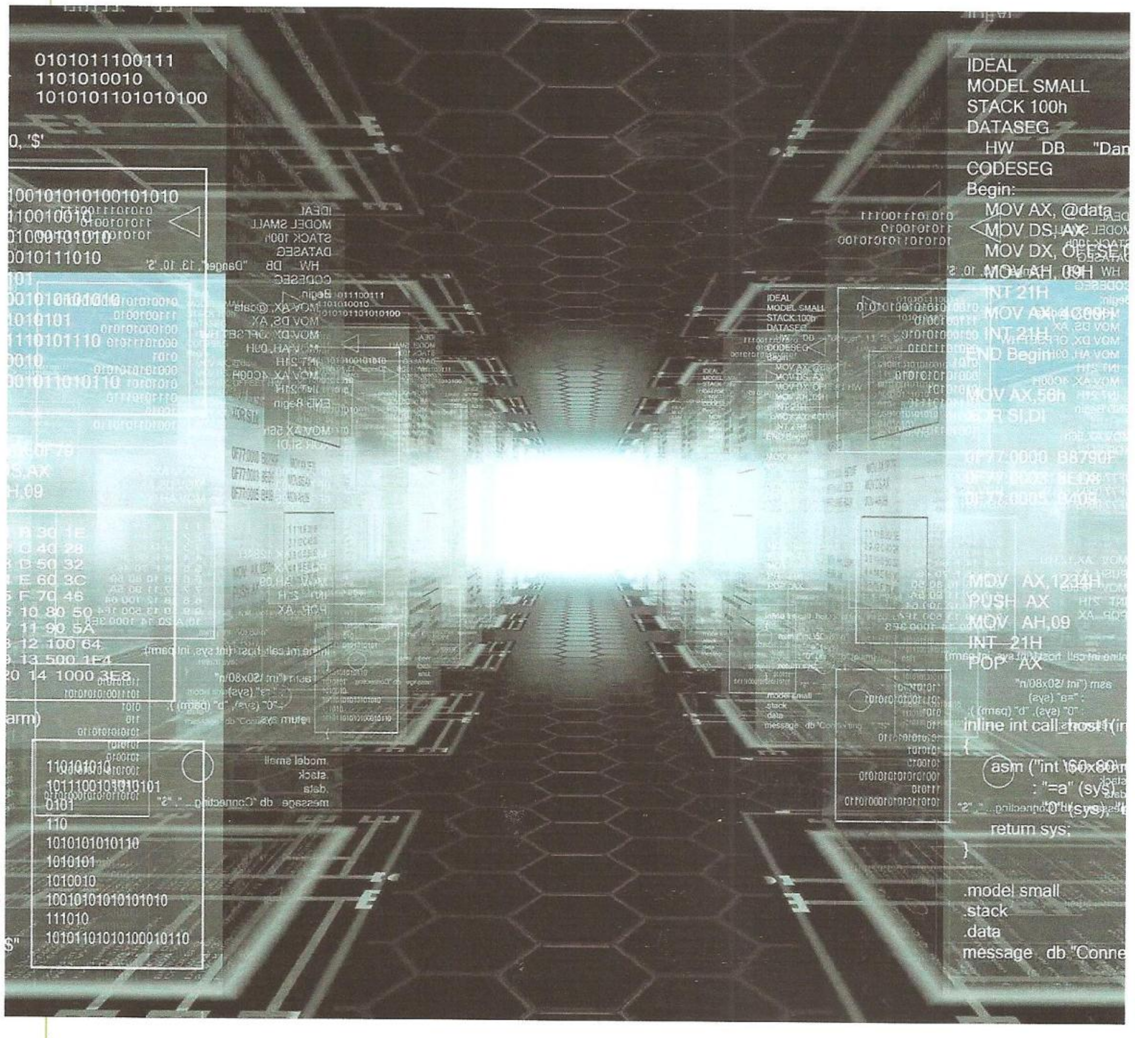
DATABASE POPULATION

There are three main ways in which to populate a database. You can copy records from one database to another, you can process inbound files and populate the database or you can directly add records via a front end GUI/application. There also could be a business rules engine applied during the processing that will change the data from the source system prior to loading to the target. The validation of these business rules can be most complex and, as always, talk to your developer/dba and understand the code paths, the list of values and the data conditions that trigger the rules engine and those that take different branches/code paths.

GOOD TESTING PRACTICE ADDRESSES QUALITY DIMENSIONS AND OBJECTIVES REGARDLESS OF THE DELIVERY MECHANISM.



DATA QUALITY



KINDS OF DATA

When you copy data from one database to another, it is usually structured data. This kind of data is discrete units of information; customer number, transaction ID etc and will be uniquely and separately identified in the column of the table within the database.

Indeed, most business processes and their corresponding data fall very nicely into a structured database. However, in our social media world images, videos, tweets, 'likes', etc do not fall nicely into a structured data world. When this information is being captured, you need to expand the number of textual combination tests to ensure these situations are addressed and certified.

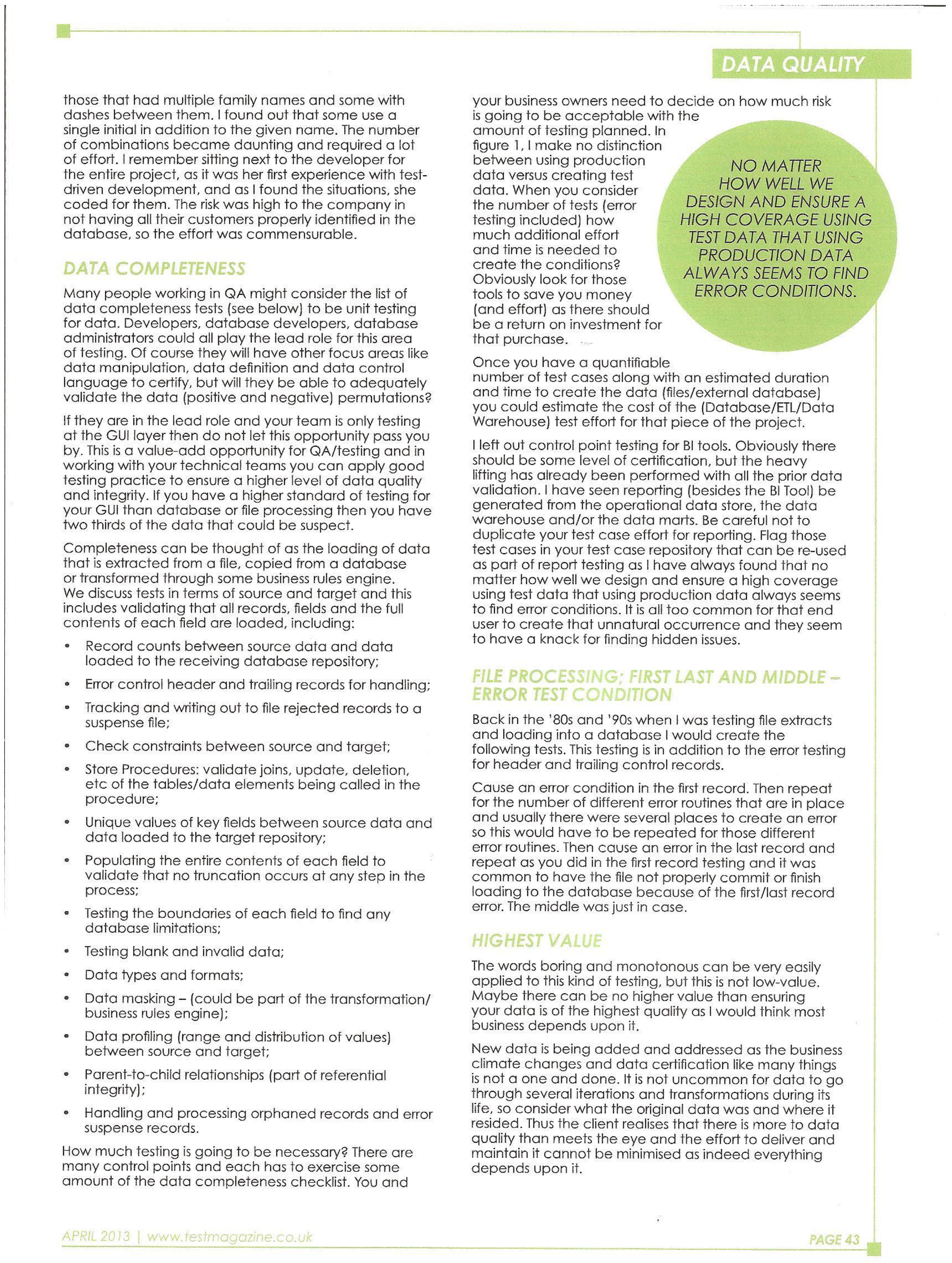
Fast growing use of unstructured data can be seen in the medical field. Besides the doctor's notes and comments that are in text, you have images of x-rays and MRIs/ scans of all sorts that are being captured digitally. Referential integrity and data completeness in the medical field is paramount. Not only are there regulatory compliance issues but the impact to a patient could be life or death.

Thankfully, all my patients are applications and by the push of a button we bring the patient back to life. In the mid 90's, I was testing an application that was parsing a name (salutation, given, family and suffix) from a single field into separate data columns within their customer table.

Besides the Mr, Ms, Mrs salutations, we needed to identify those that used "Dr" before their name and those that had 'MD' at the end; all with and without the periods. There were family names that had an apostrophe and

PAGE 42

APRIL 2013 | www.testmagazine.co.uk



those that had multiple family names and some with dashes between them. I found out that some use a single initial in addition to the given name. The number of combinations became daunting and required a lot of effort. I remember sitting next to the developer for the entire project, as it was her first experience with test-driven development, and as I found the situations, she coded for them. The risk was high to the company in not having all their customers properly identified in the database, so the effort was commensurable.

DATA COMPLETENESS

Many people working in QA might consider the list of data completeness tests (see below) to be unit testing for data. Developers, database developers, database administrators could all play the lead role for this area of testing. Of course they will have other focus areas like data manipulation, data definition and data control language to certify, but will they be able to adequately validate the data (positive and negative) permutations?

If they are in the lead role and your team is only testing at the GUI layer then do not let this opportunity pass you by. This is a value-add opportunity for QA/testing and in working with your technical teams you can apply good testing practice to ensure a higher level of data quality and integrity. If you have a higher standard of testing for your GUI than database or file processing then you have two thirds of the data that could be suspect.

Completeness can be thought of as the loading of data that is extracted from a file, copied from a database or transformed through some business rules engine. We discuss tests in terms of source and target and this includes validating that all records, fields and the full contents of each field are loaded, including:

- Record counts between source data and data loaded to the receiving database repository;
- Error control header and trailing records for handling;
- Tracking and writing out to file rejected records to a suspense file;
- Check constraints between source and target;
- Store Procedures: validate joins, update, deletion, etc of the tables/data elements being called in the procedure;
- Unique values of key fields between source data and data loaded to the target repository;
- Populating the entire contents of each field to validate that no truncation occurs at any step in the process;
- Testing the boundaries of each field to find any database limitations;
- Testing blank and invalid data;
- Data types and formats;
- Data masking – (could be part of the transformation/ business rules engine);
- Data profiling (range and distribution of values) between source and target;
- Parent-to-child relationships (part of referential integrity);
- Handling and processing orphaned records and error suspense records.

How much testing is going to be necessary? There are many control points and each has to exercise some amount of the data completeness checklist. You and your business owners need to decide on how much risk is going to be acceptable with the amount of testing planned. In figure 1, I make no distinction between using production data versus creating test data. When you consider the number of tests (error testing included) how much additional effort and time is needed to create the conditions? Obviously look for those tools to save you money (and effort) as there should be a return on investment for that purchase.

NO MATTER HOW WELL WE DESIGN AND ENSURE A HIGH COVERAGE USING TEST DATA THAT USING PRODUCTION DATA ALWAYS SEEMS TO FIND ERROR CONDITIONS.

Once you have a quantifiable number of test cases along with an estimated duration and time to create the data (files/external database) you could estimate the cost of the (Database/ETL/Data Warehouse) test effort for that piece of the project.

I left out control point testing for BI tools. Obviously there should be some level of certification, but the heavy lifting has already been performed with all the prior data validation. I have seen reporting (besides the BI Tool) be generated from the operational data store, the data warehouse and/or the data marts. Be careful not to duplicate your test case effort for reporting. Flag those test cases in your test case repository that can be re-used as part of report testing as I have always found that no matter how well we design and ensure a high coverage using test data that using production data always seems to find error conditions. It is all too common for that end user to create that unnatural occurrence and they seem to have a knack for finding hidden issues.

FILE PROCESSING; FIRST LAST AND MIDDLE – ERROR TEST CONDITION

Back in the '80s and '90s when I was testing file extracts and loading into a database I would create the following tests. This testing is in addition to the error testing for header and trailing control records.

Cause an error condition in the first record. Then repeat for the number of different error routines that are in place and usually there were several places to create an error so this would have to be repeated for those different error routines. Then cause an error in the last record and repeat as you did in the first record testing and it was common to have the file not properly commit or finish loading to the database because of the first/last record error. The middle was just in case.

HIGHEST VALUE

The words boring and monotonous can be very easily applied to this kind of testing, but this is not low-value. Maybe there can be no higher value than ensuring your data is of the highest quality as I would think most business depends upon it.

New data is being added and addressed as the business climate changes and data certification like many things is not a one and done. It is not uncommon for data to go through several iterations and transformations during its life, so consider what the original data was and where it resided. Thus the client realises that there is more to data quality than meets the eye and the effort to deliver and maintain it cannot be minimised as indeed everything depends upon it.