
Resurrecting the Prodigal Son – Data Quality

“Rise from the Ashes: Battle of Data Quality Testing”

Authors:

Raj Kamal
Building -II,
Microsoft India (R&D) Private Limited,
Microsoft Campus,
Gachibowli, A.P., Hyderabad - 500032
India
Ext: 040-669 33118

rajkamal@microsoft.com

Bhoomika Goyal
Building -II,
Microsoft India (R&D) Private Limited,
Microsoft Campus,
Gachibowli, A.P., Hyderabad - 500032
India

Goyal.Bhoomika@microsoft.com

Abstract:

There is perhaps no other activity that is as much a bane of our existence in decision making as “reconciling” data. Data quality testing is a sea with a few large fish (data integrity, data consistency, data accuracy, redundancy-related issues etc.) and many minnows.

Every firm, if it looks hard enough, can uncover a host of costs and missed opportunities caused by inaccurate or incomplete data. This is where Data Quality Testing pitches in and can be instrumental in helping businesses achieve their true potential by reaping the benefits of the timely availability of high quality of data.

After all, who would like to buy yellow pages which contain outdated contact numbers or fly on an airplane that does not conduct preflight checks?

Our mission is to provide information about the costs that testing teams incur due to the lack of data quality testing versus the benefits of taking the actions we propose in this paper.

We would like to answer some important questions which might already be popping into your mind, such as “What is DQ Testing?” If it exists, “What’s wrong with the current approach?”, “What’s new in this paper?”, “What is there in it for me?” and “Where and how can I apply it?”

The objective is to share the key lessons regarding the importance of DQ Testing and present a step-by-step generic test strategy which will help the testers and test managers adopt and apply it when they go back to work, and enjoy the benefits.

As data quality is a strategic IT initiative which involves a strong commitment and a huge investment from management, including a steering committee, we will keep the scope of this paper restricted to the contribution the test team can make – an initiative to significantly improve the data quality by incorporating DQ in the testing process which will help your organization by detecting possible DQ issues earlier than your customer reports it.

The unique, yet simple, approach suggested here is to have an automated metadata-driven approach to continuously monitor the health of the application by automating DQ checks which will provide the test team and users with a DQ summary report containing DQ test metrics.

This will create a win-win situation for the testing staff and your enterprise where the testing team can earn the well-deserved credit for improving the DQ of the application by using an effective testing approach and helping the users of the application feel confident of the data health while making critical business decisions.

1 Introduction:

Data Quality: Data are of high quality "if they are fit for their intended uses in operations, decision making and planning" (J.M. Juran).

Poor data quality can seriously hinder or damage the efficiency and effectiveness of organizations and businesses. If not identified and corrected early on, defective data can contaminate all downstream systems and information assets. The growing awareness of such repercussions has led to major public initiatives like the “Data Quality Act” in the USA and the “European 2003/98” directive of the European Parliament. Enterprises must present data that meet very stringent data quality levels, especially in the light of recent compliance regulations standards.

2 Fable:

The task at hand is to test a large data warehouse and CRM implementation of a global bank which has a huge volume of transactional data (on the scale of terabytes) scattered among various disparate sources in different formats like flat files, DB2, Oracle, SQL server, MS Access, Excel etc. To make things even more complex, a medium-sized regional bank is acquired and merged which brings in additional gigabytes of data such as customer information, loan information, account details etc.

The test team faces the challenge of ensuring the quality of the data which is as complex as the data for a subject area, since a customer might be stored in more than one source in different formats and the underlying business rules vary among the systems. The integration and transformation of data performed by the development team has a higher probability of missing significant data, duplicating existing data and introducing latencies, inconsistencies and inaccuracies into the system.

The experienced test team manually verifies and validates the integrated data showing up in the reports coming out of data warehouse against the business rules. Few test cases are written to test the sample data flow from sources to data warehouse to reports and the test team certifies the application.

A couple of days after the production release, the customer comes back with a list of DQ issues, for example they can't find some important data or some of the data is repeated multiple times in the drop-downs and the KPI (key performance indicators) are showing incorrect results. Later it is discovered that there are flaws in the transformation and integration logic and data was missed, duplicated and incorrectly manipulated resulting in many DQ issues.

Finally the customer rejects the application and goes back to manual reporting based on the legacy application and the IT team starts working on fixing and testing the DQ issues which could have been avoided by a focused DQ test approach for testing various dimensions of data qualities.

3 DQ Testing Problems / Sore Points:

DQ Testing has always been given less importance than it deserves, and for more than one reason.

3.1 DQ Requirements are “missed” in the Functional Specifications

Typically, DQ Requirements (DQ Definitions, Scope, Acceptance Criteria, Measurement criteria) are not covered in the Functional Specs and are not explicitly articulated by the business.

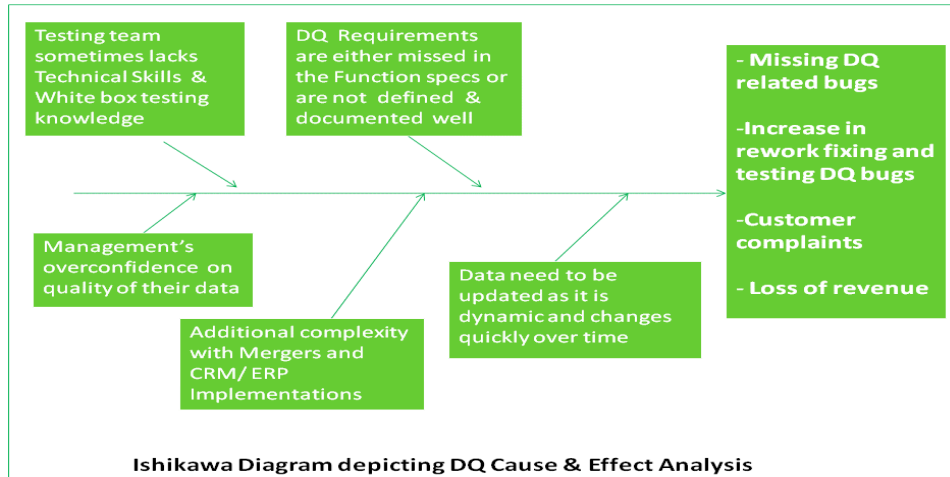


Fig. 1: Ishikawa Diagram

3.2 White box Testing / Technical knowledge is required

The testing team is often expected to perform black-box testing or functional testing based alone on the functional specifications given by the business. It was often assumed that the underlying architecture and the design (which also includes processing, transformation as well as flow of data) fall under the development team's responsibility and have to be tested by developers during unit testing.

3.3 Management overconfidence on quality of their data

It was often the case that management felt that their data couldn't have inconsistencies or inaccuracies and that activities like data integration can't impact the overall quality of data.

3.4 Complexity grows with CRM / ERP Implementations & Mergers / Acquisitions

In the last decade, as a result of globalization along with the worldwide CRM and ERP implementations trend and strategic decisions like mergers/acquisitions, data integrations has become a bigger challenge for testers due to the exponential increase in the complexity of data.

Testing Data warehousing, CRM and e-business projects often involves testing poor quality data because they require companies to extract and integrate data from multiple

operational systems. Data that is sufficient to run individual operations is often riddled with errors, missing values and integrity problems that do not show up until someone tries to summarize or aggregate the data.

3.5 Dynamic nature of the data which quickly becomes obsolete over time

Experts say that 2% of the records in a customer file become obsolete in a month because customers die or divorce, marry and move. In addition, data-entry errors, system migrations, system integrations and changes to source systems, among other things, generate numerous errors such as inconsistencies, inaccuracies, latencies etc.

4 DQ Adoption Guide

If your project falls under any of the categories shown in the diagram below then you enter the realm of DQ Testing. Basically whenever data is created, modified, integrated or processed from one form to another, there is a probability of introducing DQ issues and hence DQ testing is a must.



Project Categories for DQ Testing

Fig. 2: Project Categories

5 DQ Test Strategy:

Data Quality Testing is the process of verifying the reliability and effectiveness of data. Maintaining data quality requires monitoring the data periodically and reporting it as early as possible. DQ Test Strategy should be prepared during the requirement analysis phase as it is important to define and measure DQ requirements which are expected by the customer.

DQ Testing can be done in a “**Preventive**” as well as “**Reactive**” manner. Focusing on DQ from the envisioning phase ensures that DQ issues are prevented by means of verification/static testing. As explained in *Section 5* below, a Reactive approach has been developed which can identify the DQ issues that have crept into the system by monitoring the DQ health on a continuous basis.

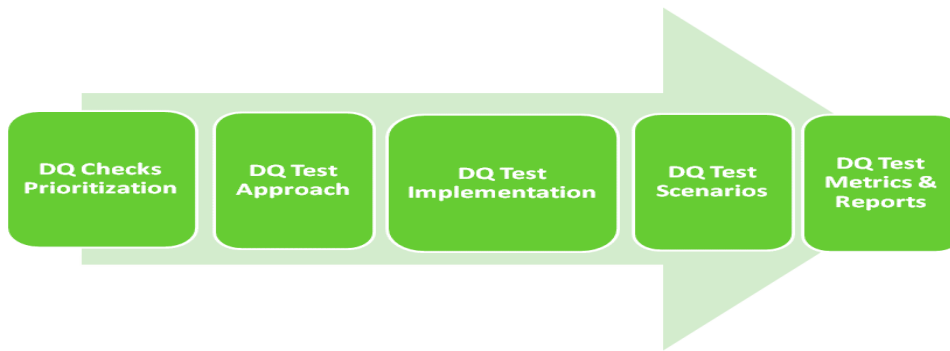


Fig. 3: DQ Strategy

5.1 Test Prioritization (Kano Analysis):

This approach suggests prioritizing the performance of DQ checks depending on their importance to the customer which results in increased performance and satisfaction.

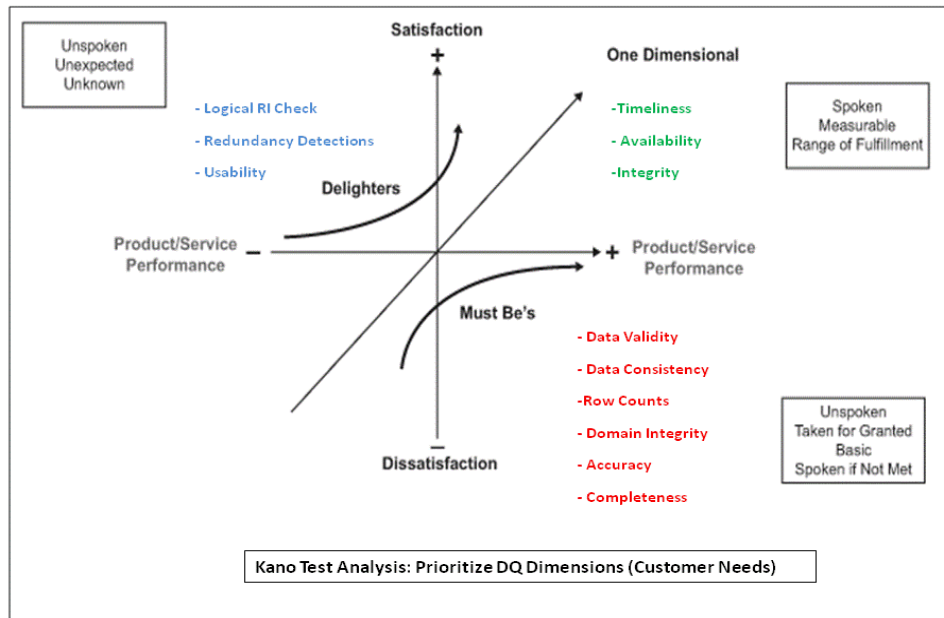


Fig. 4: Kano Analysis

As shown in the Kano analysis in the figure above, the DQ Dimensions/Audits are classified in terms of the following categories:

Must be's (Expected Quality):

The highest test priorities should be given to the typical DQ checks shown in the figure above as these are the basic explicit customer requirements

One Dimensional (Desired Quality):

The next important requirements are the DQ checks like Timeliness, Availability etc. as these are most often the promised requirements.

Delighters (Excited Quality):

DQ checks which are not requested by the customer but which can increase the satisfaction by meeting these implicit requirements.

5.2 DQ Testing Approach

The involvement of the testing team starts right from the requirements phase and continues throughout the life span of the project. The test team has to ensure that DQ is appropriately defined and the measurement criteria are included in the requirement document along with the threshold and acceptance criteria defined by the business for each DQ check.

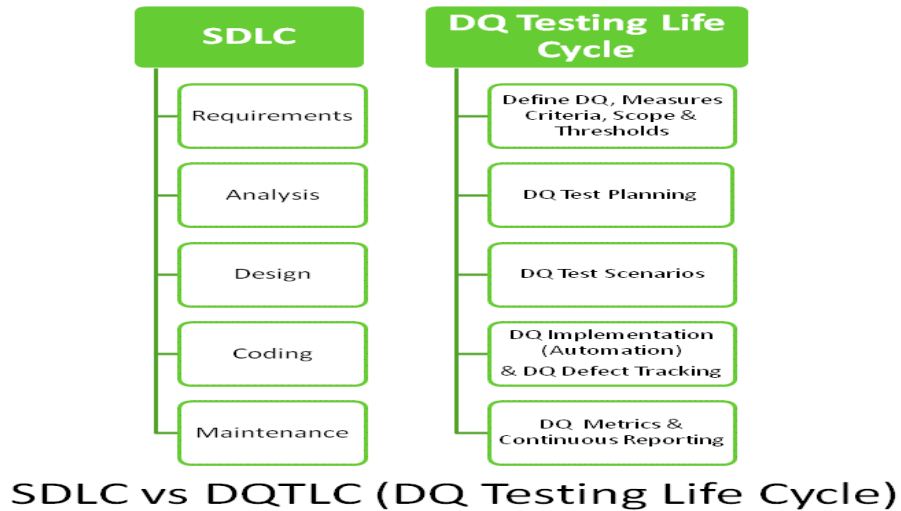


Fig. 5: DQ LifeCycle

DQ Test Planning:

- Define Data Quality test approach & the scope of testing by taking inputs from the business.
- Decide on the test metrics to monitor & measure DQ.

DQ Test Design:

- Create DQ Test Scenarios with the expected test results (*refer to Section 6 below*).
- Have it reviewed by the business and against the threshold values for each DQ check.
- Prepare DQ Metadata from Data dictionary, Business Rules from FS and Input from Business.

DQ Test Execution & Reporting: (*refer to Figure under Section 5*)

- Automate DQ test scenarios for checks which can be scheduled to run for the DQ metadata over a period of time.
- Log the DQ test scenarios result (manual/automation) and report DQ discrepancies to the users.

DQ Test Monitoring:

- Run DQ Test scenarios on a scheduled basis to continuously monitor the health of the applications.
- Send the automated reports/notifications to the users with the discrepancy summary of the DQ Test Metrics (*refer to Section 7*) which are violating the defined threshold during planning.

6 DQ Test Implementation / Solution:

To ensure information quality, the test team has to address it from the very beginning. The test team is required to define data quality in a broad sense by taking inputs from the business, establish metrics to monitor and measure it, and determine what should be done if the data fails to meet these metrics. Data quality issues need to be detected as early in the process as possible and dealt with as defined in the business requirements.

DQ Inputs:

- **Metadata:** contains list of server name, database name, table names, and column name against which DQ checks need to be performed.
- **Business Rules:** contain the functional logic and formulas which are used to process and transform the data.
- **Thresholds:** the values which define the valid range or accepted deviation for various DQ Metrics.

DQ Engine:

- **Automated Scripts for DQ Checks:** These can be your SQL code to validate consistency, accuracy, validity, row count etc. against the metadata which is keyed as input.
- **Discrepancy/Results Tables:** Automation scripts log the discrepancies in these tables by comparing the output against the discrepancies based on the threshold defined.
- **Scheduling for DQ Checks to run:** Automation scripts are run in an “unattended” mode on a continuous basis to track the DQ health of the application over a period of time.

DQ Output:

- **DQ Discrepancy Summary Reports with Metrics:** Based on the template defined, DQ metrics are calculated from the discrepancies logged by the DQ Engine and reported to customers and management in the form of DQ Notifications/e-mails.
- **DQ Issues/Bugs:** DQ issues like application bugs are triaged with the development team and the customers and pass through the typical defect lifecycle.

DQ Feedback & Continuous Process Improvement:-

- **DQ Automatic Feedback & Continuous Improvement:** The IT team along with the business decides the corrective actions and the priority.

Some of the corrective actions can be:

- Fixing the code to rectify the DQ issue
- Change in requirement or functional spec if it was a functional DQ issue
- Change in design, architecture if it was an environment or performance related issue

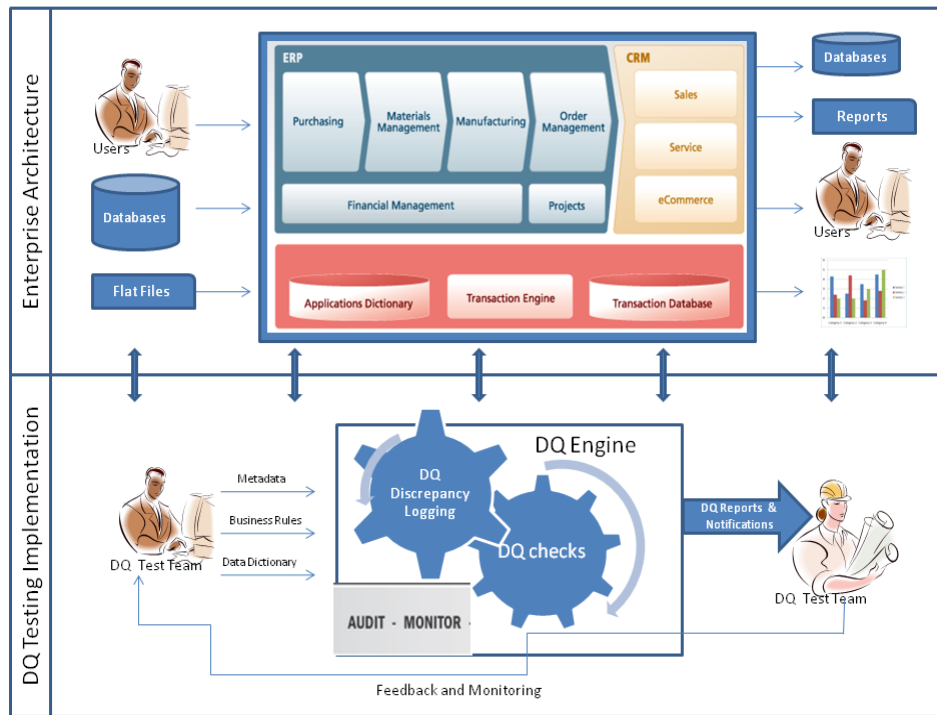


Fig. 6: DQ Implementation

7 DQ Test Scenarios:

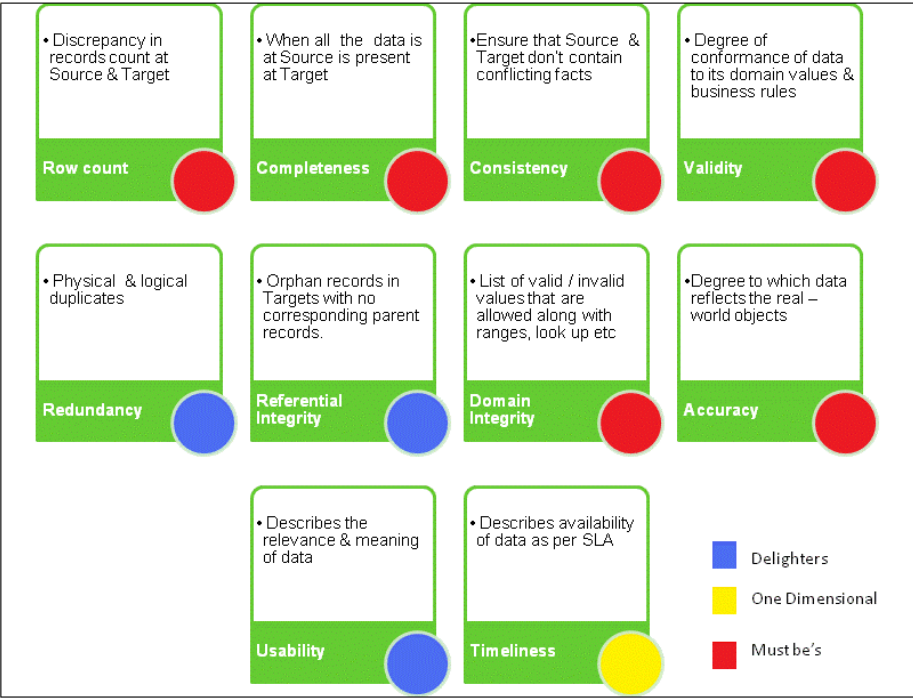


Fig. 7: DQ Checks

7.1 Row Counts: ■

Count of records at Source and Target should be the same at a given point of time.

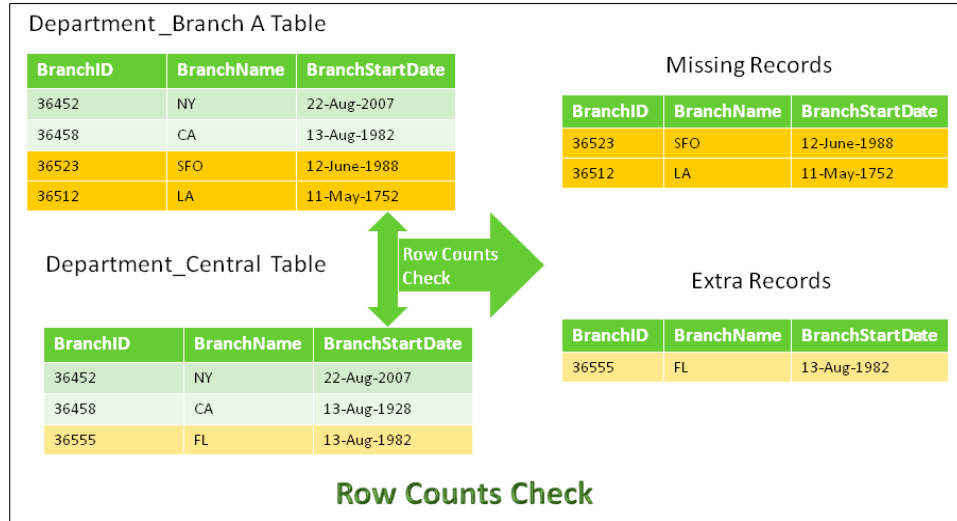


Fig. 8: Row Count Example

7.2 Completeness: ■

All the data under consideration at the Source and Target should be the same at a given point of time satisfying the business rules.

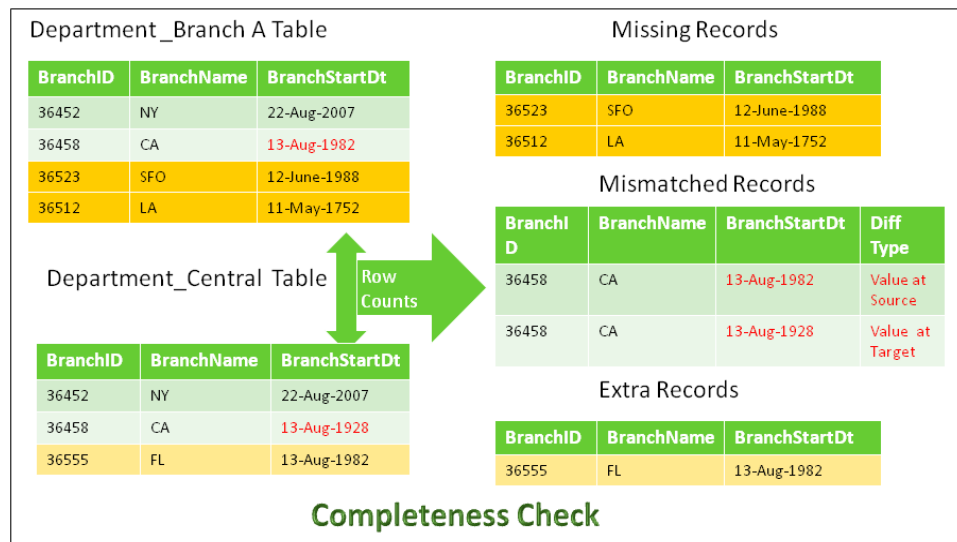


Fig 9: Completeness Check

7.3 Consistency: ■

This ensures that each user observes a consistent view of the data, including changes made by transactions. There is data inconsistency between the Source & Target if the same data is stored in different formats or contains different values in different places.

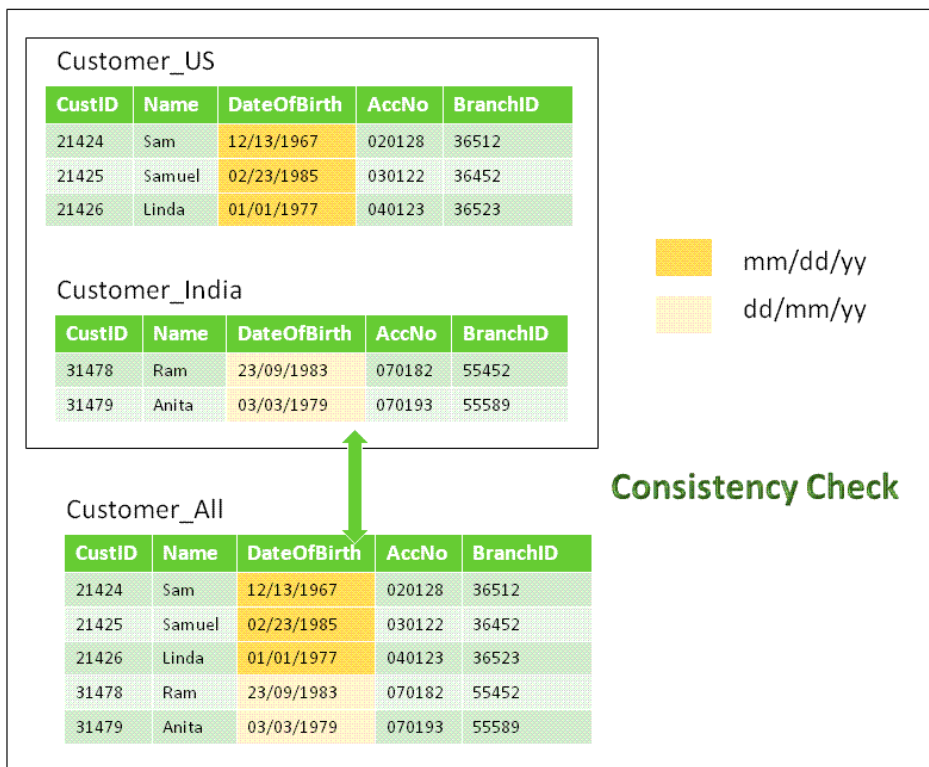


Fig 10: Consistency Check Example

There is a need for tools that can handle international data differences, that require such features as support for Unicode and rules engines that can deliver local address validation and other functions across multiple languages and formats.

Example: Regional Setting e.g. Language

BranchID	BranchName	Revenue (\$)	BranchStartDate	Difference Point
1	New York	100	22/08/2007	Same data, Inconsistent due to different language used
1	人的資源	100	22/08/2007	Same data, Inconsistent due to different language used

Example: Different values at different points

CustID	Name	AccountNumber	Balance (\$)	Difference Point
23456	John	0106723	125000	Same data, Inconsistent value for Account Balance at different locations/servers of the bank
23456	John	0106723	15000	Same data, Inconsistent value for Account Balance at different locations/servers of the bank

Consistency Check

Fig 11: Consistency Check Example #2

7.4 Validity: ■

Validity refers to the correctness and reasonableness of data. A valid measure must be reliable, but a reliable measure need not be valid.

Questions:

-> Is the information reliable?

-> How is the information measured?

Fact: A large bank discovered that 62% of its home-equity loans were being calculated incorrectly, with the principal getting larger each month.

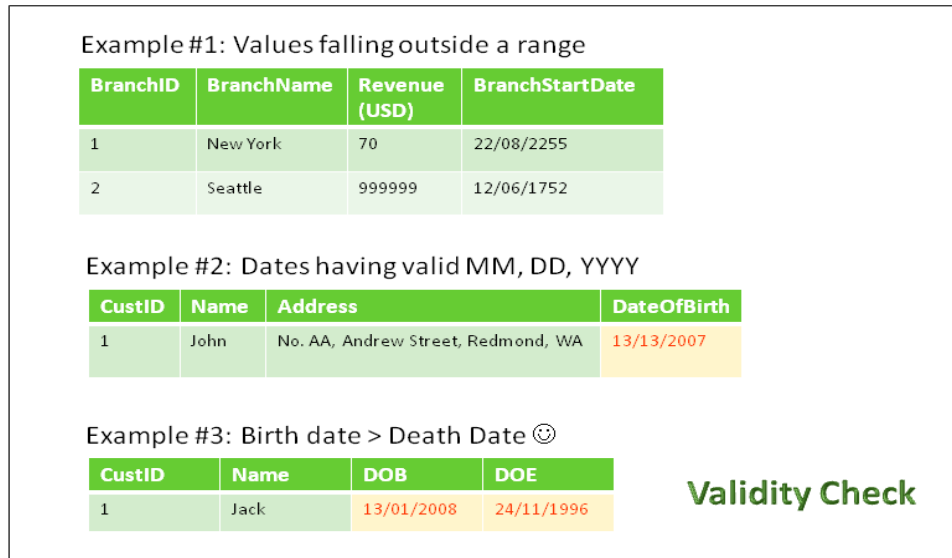


Fig 12: Validity Check

7.5 Redundancy/Duplicates Detection: ■

Consider what happens when a single customer is included in a company's database multiple times, each time with a different value for the customer identifier. In such a case, your company would be unable to determine the true volume of this customer's purchase. You could even be placed in the embarrassing situation of attempting to sell the customer an item that the customer has already purchased from you.

Physical Duplicates: All the column values repeating for at least 2 records in a table.

Logical Duplicates: Business Key (list of column) values are repeating for at least 2 records in a table.

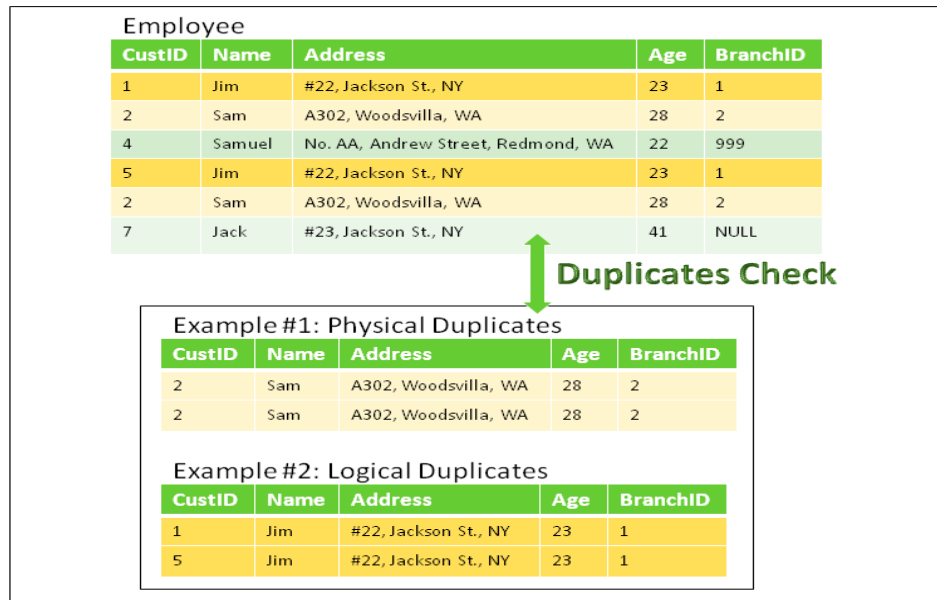


Fig 13: Duplicates Check

7.6 Referential Integrity: ■

If there are child records for which there are no corresponding parent records then they are called “Orphan Records”. Logical relationship rules between parent & child tables should be defined by the business.

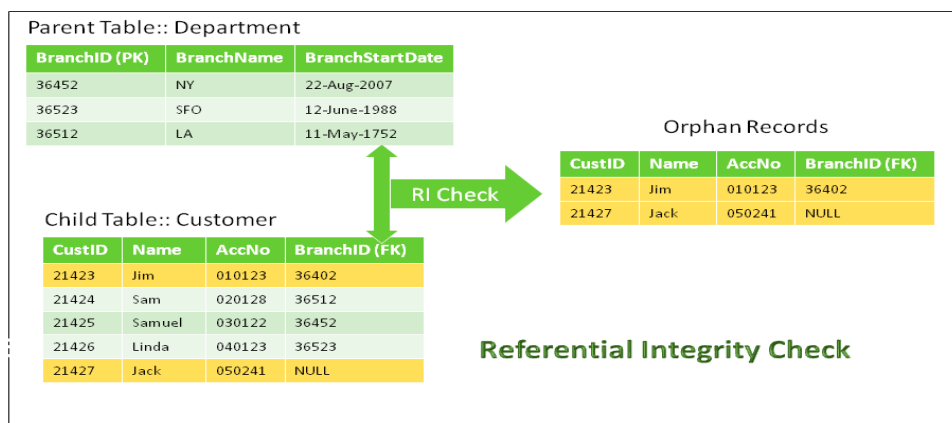


Fig 14: Referential Integrity Check

7.7 Domain Integrity: ■

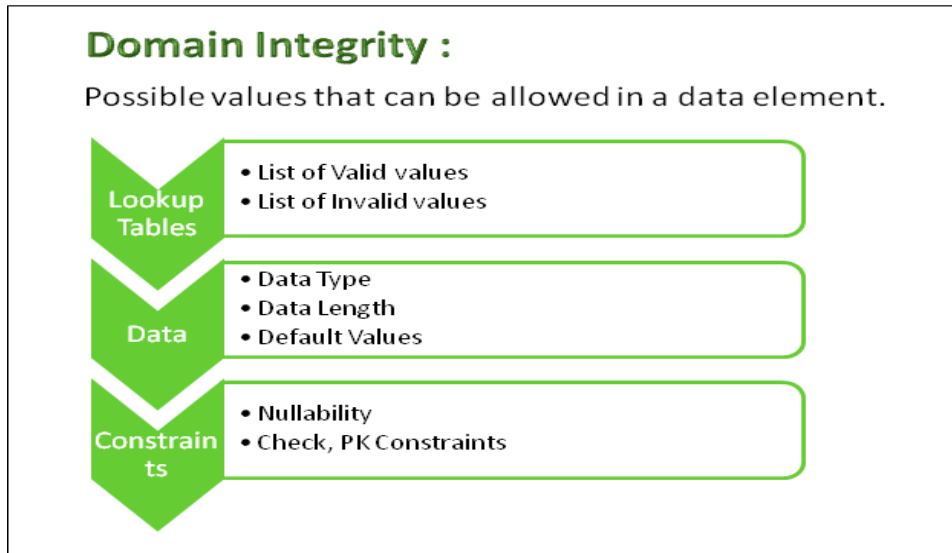


Fig 15: Domain Integrity Check

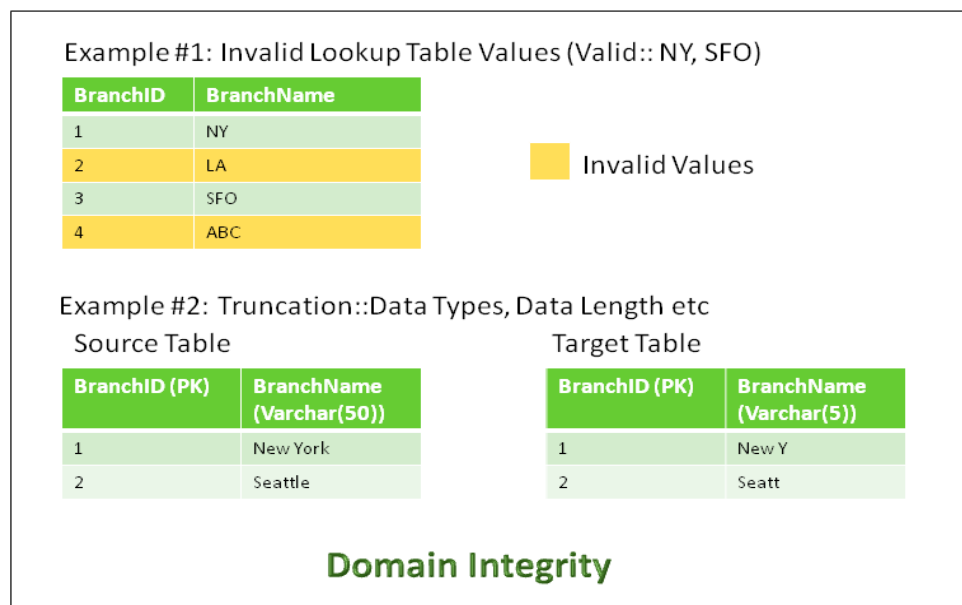


Fig 16: Domain Integrity Check

7.8 Accuracy: ■

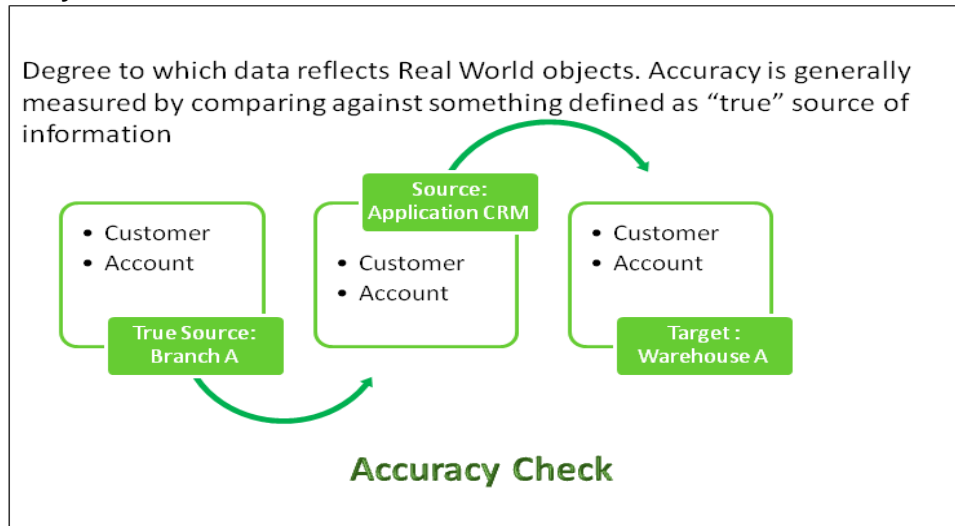


Fig 17: Accuracy Check

7.9 Usability: ■

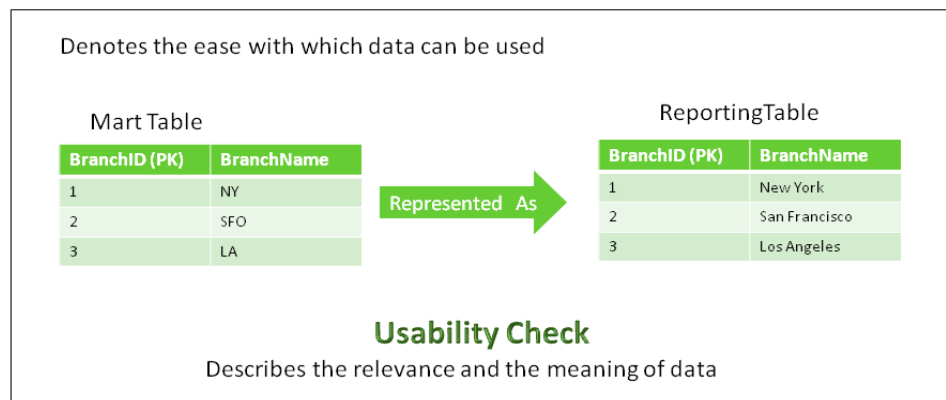


Fig 18: Usability Check

7.10 Timeliness ■

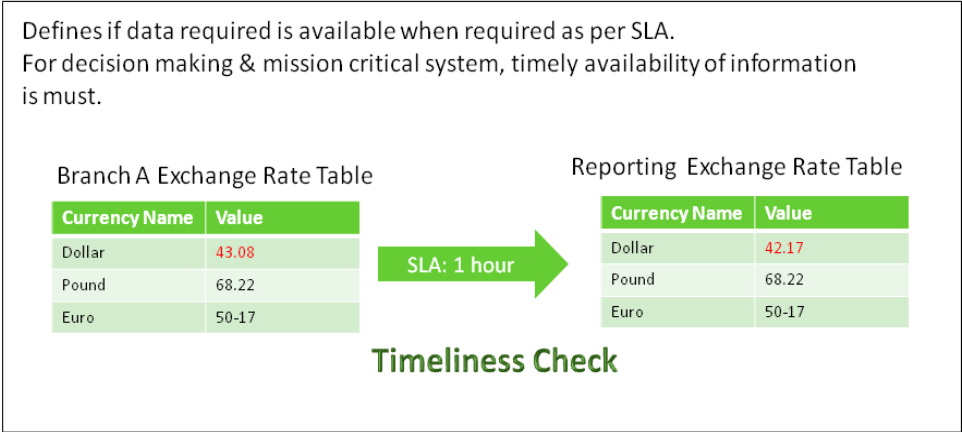


Fig 19: Timeliness Check

8 Key DQ Test Metrics:

Data Quality metrics, which helps in quantifying the quality of data over different dimensions, can be derived on a scheduled basis from the DQ Test Scenarios execution or result logs for various DQ checks manually/automatically by having a DQ test implementation in place (similar to the one proposed above)



Fig 20: DQ Test Metrics

These DQ metrics are critical so that management may determine the DQ health of the integrated system. Depending on the project needs and the criticality, the frequency of calculating and reporting these metrics can be determined.

9 Conclusion:

It is critical to measure the effectiveness of the test strategy as it is the key to “*continuous process improvement*”. We will talk about this in detail in our next white paper. Stay tuned.

Today we have introduced many DQ test problems and recommended a test strategy to tackle them. We can assure you that the complexity and the scope of data quality testing

in large enterprises are difficult for a layman to understand. Most test teams do not prepare or adopt a dedicated test strategy to build quality into their data in a proactive, systematic and sustained manner. Many potential DQ issues can be avoided by implementing the proposed DQ Test strategy as part of the overall test process. It is often not until you discover a major problem in production that could have been avoided through quality control of your data that you recognize the importance of data quality. As a consequence, the business may lose revenue, opportunities, even customers, and in the worst case can even get sued.

Don't end up like them. Be different. Always remember to put quality first.

9. References:

- [1] The Kano Analysis: Customer Needs Are Ever Changing by J. DeLayne Stroud
<http://finance.isixsigma.com/library/content/c071017a.asp>
- [2] Be Prepared to Duel with Data Quality – Rick Sherman
<http://www.athena-solutions.com/bi-brief/2006/jan06-issue24.html>
- [3] Trends in Data Quality – Lou Agosta
<http://www.dmreview.com/issues/20050201/1018111-1.html>
- [4] Gartner ranks data quality management software, reveals trends - Hannah Smalltree
http://searchdatamanagement.techtarget.com/news/article/0,289142,sid91_gci1263861,00.html
- [5] Data Warehousing Special Report: Data quality and the bottom line -Wayne W. Eckerson
<http://www.adtmag.com/article.aspx?id=6321&page=>
- [6] The Importance of Quality Control: How Good Is Your Data? - Andrew Greenyer
http://www.customerthink.com/article/importance_quality_control_how_good_data
- [7] Garbage In, Garbage out: The Importance of quality ideas – Jennifer Hanson
<http://www.fastcompany.com/blog-post/garbage-garbage-out-importance-quality-ideas>

10. Author:



Raj is a Test consultant specializing in different types of testing techniques, test automation and testability in different domains like Manufacturing, Healthcare and Higher Education. He holds an APICS certification in Supply Chain Management. Expertise with Rational and Mercury testing tools, he has helped teams develop test automation strategies and architectures for such companies as Cognizant Technology Solutions and Oracle Corporation. He also provides training in automated testing architectures and design. He is QAI (CSTE) & ISTQB Certified. He has a master's degree in Computer Applications. He is currently working at Microsoft, India, Business Intelligence COE. He has earlier represented Microsoft and Oracle at International test conferences as a Speaker.
His Blog (<http://www.itest.co.nr>)