Data Quality Practices and Measurements

Nick Bonnichsen

nick.bonnichsen@hmhco.com

Abstract

If your business and customers depend on data you need data you can depend on. Storage is cheap, processing is flexible, and data is a hidden trove of value increasingly used to drive business decisions and priorities. We generate it by the terabyte. We copy it from database to database, transform it from NoSQL to Relational data to graphs and charts. We ingest it from customers and reflect it back to them with augmentations and additions our Sales departments have promised are just what is needed to propel customers to the next level. More than ever data drives our customers' business decisions as well as our own. Poor quality data can mean poor decision making, or worse, customer outrage.

After some hard lessons in what poor Data Quality can cost, we determined to find ways to measure and assess Data Quality outside of functional testing requirements. For Data Quality to be measured we needed methods to compare datasets together, either by comparing a dataset to an idealized dataset or comparing two datasets against each other. Independent Data Quality dimensions could be defined, agreed upon, and measured. Further we extended quality analysis work to practices and cultures that encourage Data Quality or that enable poor quality to sneak in when absent. Data Stewardship was identified as a starting point for data ownership and continual improvement work. Data standards enable development teams to build in Data Quality from the get-go with practices like rigorous input validation and periodic data surveys.
.

Biography

Nick Bonnichsen is a Staff Quality Assurance Engineer with 20 years of experience in testing and quality initiatives. Currently he is working on adjusting company culture and processes in pursuit of higher quality development and testing practices, like Agile and DevSecOps. Outside of work he enjoys glamping, backpacking, travel, role playing games, video games, and board games. Occasionally he mauls wood in a futile attempt to create furniture. He is married to an Australian, has an adopted daughter, and a one-year-old grandson.

1 Introduction

On a completely normal cold wet miserable day in February the CTO called a meeting with me and some teammates to discuss some recent problems with our data. He was justifiably nervous. An update in the recent past caused many of the reports that we supply to our customers to have faulty information. Other problems had been reported with customer-facing APIs where data was missing even after a year-long project to overhaul the data pipelines that fed the service. More pressing some recently signed contracts with high value customers included punitive payments should certain Data Quality issues arise. His ask from me and my team of Senior QA fellows was to “improve Data Quality” and ensure that we didn’t continue to have these issues.

As I had the most experience in testing our data centric applications I was quickly nominated to lead this effort. Our CTO hinted that what we wanted was a silver bullet of QA mastery that would settle the matter quickly and with minimal extra work or disruption to the yearly schedule. Having neither a tricorder or a wand of elm and unicorn hair to measure our datas’ output of qualitons, the fundamental quantum quality particle, I was forced to resort to existing practices, the vast quantity of Data Quality(DQ) related information on the internet, and my own experience to craft a set of Data Quality recommendations.

2 Definitions for Data Quality

The first step was aligning on a solid definition for Data Quality that we could work from. As we began researching the problem in depth we found that Data Quality was a widespread term used by different industries with differing focuses. Search engine results quickly informed me of the top selling Data Quality tools that purported to do the job for me at exorbitant price and without a lot of detail on how they performed the task.Other sites wrote intelligently about the business challenges of Data Quality without offering a real definition of what that meant, as if the audience just knew what quality was or was not. Quality data is “fit for intended uses in operations, decision making and planning” (Redman, 2008) or “fit for purpose” were common phrases. “No one complaining about your data" or data that “exceeds customer expectations” as an indicator of quality was also mentioned frequently.

While those sentiments captured the important essence and expectations of high-quality data, they did not directly lend themselves to an easy method of measurement. Like many descriptions of quality, they fall back to a “you know it when you see it” sort of description which I felt was lacking specificity. I was looking for something more proactive and concrete that could be reduced to actual measurements that can be regularly taken, tracked, and reported on. I needed data on Data Quality.

In desperation I turned to the Wikipedia article on Data Quality which listed similar statements as those above and concluded that what they all had in common was that "Data Quality is a comparison of the actual state of a particular set of data to a desired state” (Data Quality, 2021). Therein lay the foundation for a practical set of metrics that I had been looking for. To measure Data Quality, you must *compare* a set of data to some other set of data. How nearly it approaches the idealized or actual set used for comparison in various dimensions can, overall, be taken as a measure of general quality.

This definition fit well with the other nearly universally described practice of Data Profiling. Data Profiling is the process of measuring different aspects or Dimensions of Data Quality. Data Profiling in action involves the periodic scanning and aggregations of data to understand its structure, content, and adherence to business rules. It’s the most frequently cited technique used to determine Data Quality. Many of the tools that I initially found when searching for a Data Quality definition were focused on setting up regular data profiling jobs and displaying the results of those scans. Such a tool is nice but not required as data profiling can also be achieved with simple data scripts to count records or fields within a dataset that meet particular criteria and collect that data.

Now that I defined Data Quality to be measurable by comparison to some other data I needed some other data to compare to. Our frequent data related defects immediately suggested that comparing a source dataset to one that had been copied from it, a target dataset, as a great place to start. This is applicable whether your target dataset is immediately downstream of your source dataset or if you want to compare a target dataset that had been copied several times through different transitions from some blessed and trusted source dataset, a source of truth. If all software were operating correctly your target dataset should exactly match your source dataset. I referred to this kind of comparison as Data Fidelity.

A second less obvious comparison can be made between a given dataset and an idealized version of that dataset that perfectly meets some business or real-world thing which the data is meant to describe.A dataset of Persons, for instance, could be idealized as having a surname, firstname, date of birth, and identifier for each record within the dataset. That can be compared to a real dataset that may be missing one or more of those attributes. I call this Intrinsic Data Quality.

Someone looking to understand their Data Quality by looking at Data Fidelity or Intrinsic Quality will find that there are a number of different comparisons and measures that can be made. Any given aspect of comparison between two datasets can be thought of as a Data Dimension, each of which should have a commensurate measurement, though some resist easy measuring. The key idea behind focusing on measurements was that we wanted some way to judge whether any changes we made to processes or programs had a positive, negative, or neutral impact on Data Quality. By taking actual measurements periodically, by gathering data on the data itself, we had a metaphorical yardstick we could use to judge whether our Data Quality improvement efforts were successful.

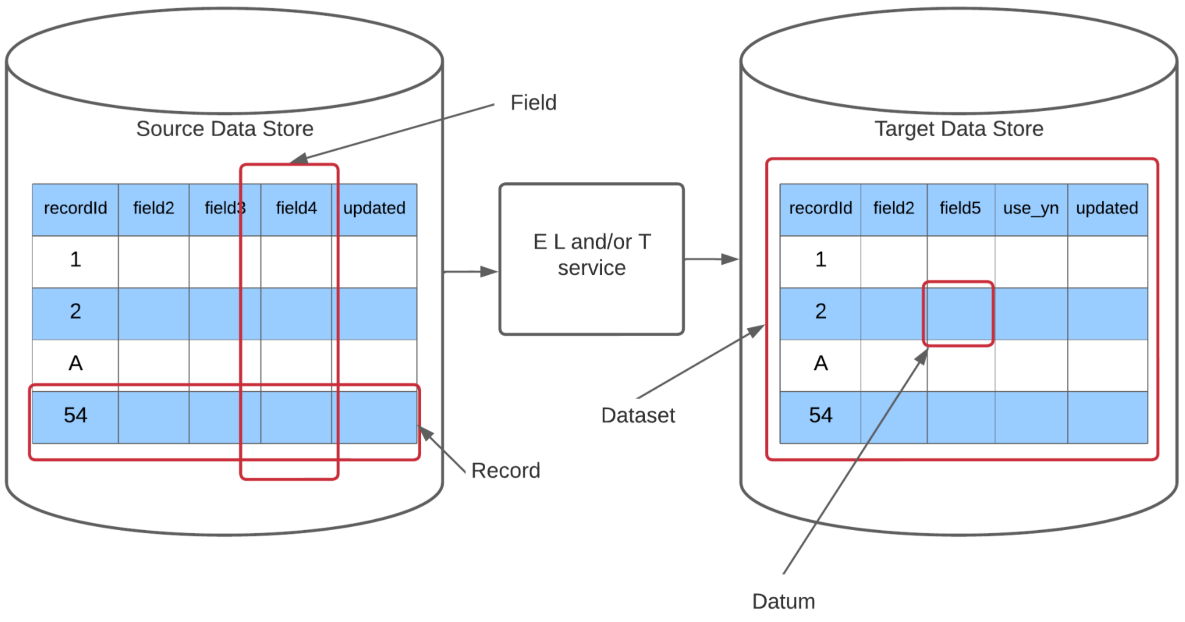


Figure 1: Data Quality terms

As we set up out new processes to address Data Quality on a select few teams where troubles were prevalent we considered how to use data profiling and data comparisons to get to actual measurements, real data about our data, As we considered the best dimensions to explore we began by aligning on some basic terminology for the subject. For the purpose of this paper, and to avoid confusion or database specific terminology, I'll use the following terms

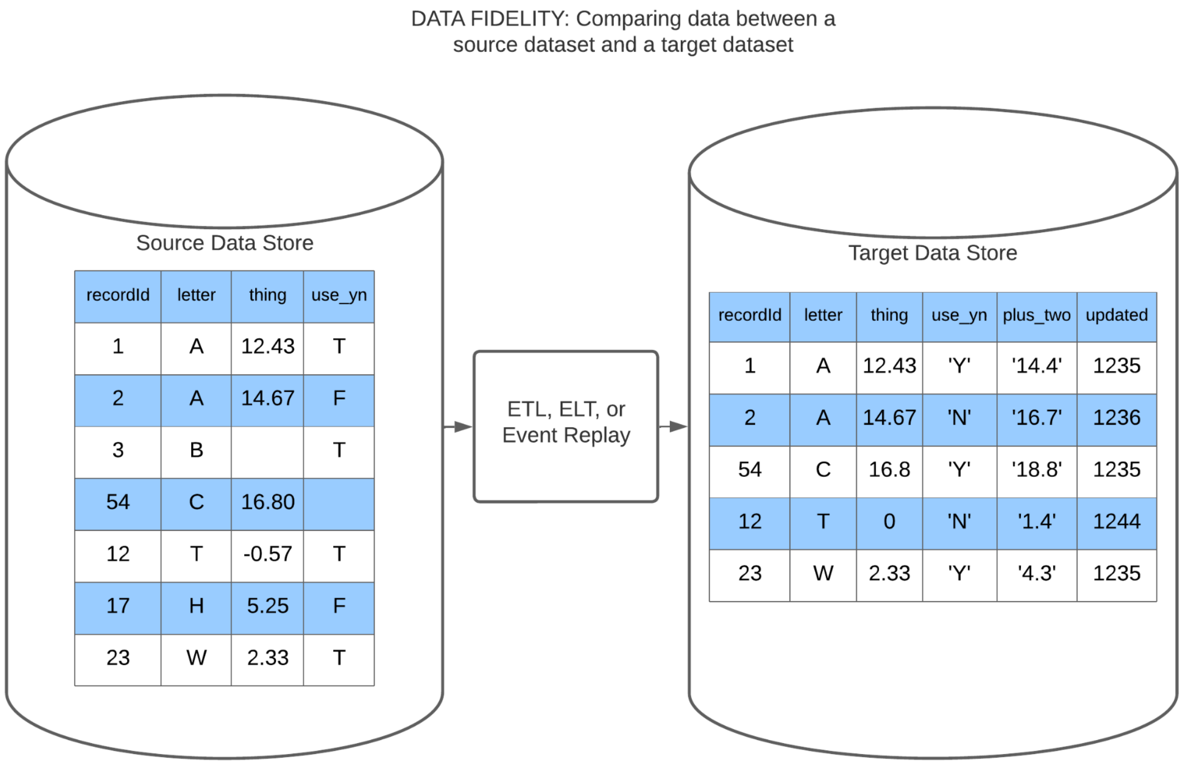
- Datum: a single piece of information that is necessary, useful, interesting, or important for operations, decision making, planning, tracking, or any other purpose as defined by business or technical needs
- Data: Multiple datum usually found collected together in some structured way.
- Dataset: A discrete set of data that is organized for a particular purpose or around a specific real-world concept that the data attempts to describe. Most often when measuring DQ a dataset will

be considered equivalent to a single table of data from a relational database or a set of documents from a NoSQL database.
- Data store: A collection of datasets and the physical system that holds them. Equivalent to a database in most cases with some flavor of schema. Basically, a set of datasets.
- Field: A named category of data that describes what the data represents and ties data to requirements regarding the data type, precision, format, and any applicable business rules about the data. In a relational database a field is equivalent to a column. In NoSQL databases it's usually equivalent to a single key-value pair wherein the key is the defined name for the field and the value is the datum. Fields include all the data in the dataset with that same field, in other words all the datum within that column of all of the values with the same key name.
- Record: The set of data that contains all the information for one particular real-world or business object, entity, relationship, concept, measurement, or thing within one dataset. In relational databases a record would be one row of a particular table. In NoSQL data stores a record would be a single NoSQL document. Records should have either a unique identifier or a set of identifiers that collectively are unique which is usually called the primary key. Primary keys are not required but it's a real pain if they aren't there.
- Source dataset: When data is copied between a dataset in data store A to a dataset in data store B the starting dataset in data store A is the source dataset
- Target dataset: When data is copied between a dataset in data store A to a dataset in data store B the resulting dataset in data store B is the target dataset.

3 Data Fidelity

Among the most glaring problems that triggered our CTO to reach out to my team to build a broader foundation of Data Quality was a host of issues with missing data. Having adopted a microservice approach to architecture we had a number of stand-alone APIs and single page web applications that were engineered to rely on a local copy of data in order to achieve the performance desired by our customers and Product Owners. During functional and automated testing these applications performed very well and consistently passed QA checks. In Production however these services generated a continual stream of reports from customers of missing data. Sometimes as small as a single record, sometimes thousands. Root cause remained hard to identify or outside of the resources of the responsible teams to address. Teams were reduced to solving problems with direct data editing or some brute force tools to reload data from a trusted source dataset when customers reported problems.

To address this problem we needed to measure the fidelity of the various target datasets that resulted from a trusted source of dataset. We began doing so with some custom made tools that would query both the trusted source data store, a postgres database, and target stores that could be NoSQL or relational databases. The tool relied on universal unique identifier fields which were consistent across all data stores that some wise designer had previously established in our data. Thus we were able to perform record to record and field to field comparisons on data that we would otherwise have not been able to match due to data store dependent identifiers. When measuring the datasets we also needed to keep in mind any and all rules that may have intentionally omitted data from the copy, transformed the data during the move, or other factors that would legitimately change the data during the process.


DATA FIDELITY: Comparing data between a source dataset and a target dataset
Source Data Store
ETL, ELT, or Event Replay
Target Data Store

recordId	letter	thing	use_yn
1	A	12.43	T
2	A	14.67	F
3	B		T
54	C	16.80	
12	T	-0.57	T
17	H	5.25	F
23	W	2.33	T

recordId	letter	thing	use_yn	plus_two	updated
1	A	12.43	'Y'	'14.4'	1235
2	A	14.67	'N'	'16.7'	1236
54	C	16.8	'Y'	'18.8'	1235
12	T	0	'N'	'1.4'	1244
23	W	2.33	'Y'	'4.3'	1235

Figure 2: Dimensions of Data Fidelity

3.1 Measures of Data Fidelity Dimensions

3.1.1 Set-Completeness

Set-Completeness, also just called completeness, is a measure of the number of records in the source dataset that should have been copied to the target dataset compared to the count of records in the target dataset. In *Figure 2* above the Source dataset has 7 records while the target dataset has 5. Compare them together for each dataset to understand how many records have been transferred successfully. If the business rules of the copy are that all records in the source dataset should be in the target dataset then we have detected a quality issue.

This measure can be made difficult if there are business rules involved in the data transfer operation that omit some records from transfer. In such a situation you are required to understand the business rules and incorporate that logic into the programming doing the counting. In our *Figure 2* example if a business rules is that no records where use-yn is False (F) should be transferred then the record count on the Source dataset would be 5, the same as in the target dataset. No quality issue detected, even though the eagle-eyed will notice that it is not the records with use-yn set to False that have missed being transferred. Thus Set-Consistency is not the sole measure of Data Quality you should use but it is often the easiest and quickest measurement to make frequently to determine if quality is slipping enough to warrant more extensive measurement. .

3.1.2 Consistency

Data between two datasets is "consistent" if the data for each record in the source dataset, or a selection of fields in each record, matches the record and field in the target dataset exactly. This is a high value measurement but also an expensive one to execute as it requires a record to record and field to field comparison of the data between the two datasets. When the datasets are in two different data stores, in possibly wildly different styles, the comparison is even more onerous to perform. In *Figure 2* a consistency comparison would compare each record in the source dataset with each record in the target dataset, matching on the *recordId*. Assuming the *plus_two* and *updated* fields are not directly copied we can omit those and just compare the *letter, thing,* and *use_yn* fields. We can see that the *letter* field is transferred with high consistency, having a 1:1 match with all records in the target dataset. The Consistency on *thing* is less ideal as the value in record 12 is 0, not -0.57. So for that field only 4 out of 5 records are consistent, maybe even 3 of 5 if you notice the 16.80 became 16.8. Further a deep comparison would also pick up either that records 3 and 17 are missing in the target dataset or that record 2 was included when it maybe shouldn't be.

Consistency is probably the most thorough measurement that can be made to assess Data Fidelity. However with large complicated data sets compounded by complicated business rules it can become time consuming and expensive to run frequently.

3.1.3 Validity/Precision

Data between two sets is "valid" if the data in each field matches the requirements for numerical precision (number of decimal places as set by business rules), data type, and format. For calculated values it matches the expected result from independent calculation given the same inputs and determined by business rules. In our example datasets in *Figure 2* we can assess the rounding rules and data type for the *thing* column and the calculated *plus_two* column. The *thing* column in the target seems to match the values and precision of the source dataset exactly, except for the 0 for reportId 12. The *plus_two* field however may have some issues.Despite being a numerical calculation it is being stored as text. Secondly the resulting value is rounded to one decimal place, not two like that addition of 2 to the source *thing* data would suggest. Depending on your business rules all of those values may be invalid. A Validation/Precision measurement can be expressed as the number of invalid records over the number of records measured.

3.1.4 Transformational Integrity

Data that has been transformed during movement has been transformed correctly. Data is transformed for a huge number of reasons and those transformations should be checked to make sure they were executed correctly. In our example data there has been a clear transformation of the values in the source dataset *use_yn* field from boolean True and False values to the odious 'Y' and 'N' text values. Comparing the source and target datasets will involve performing the transformation being done by code again in your profiling tools and validating that the transformation of the profiling tool does based on the source data matches what is stored in the target data. The overall transformational integrity can be expressed as the number of records correctly transformed over the total number of records that should have been transformed. We could score our example in *Figure 2* as having 4 out of 7 records transformed correctly

3.1.5 Exactness

The data in the target dataset has the same meaning and names as the data in the source dataset. This is a more manual measure to make than the others as it requires a human (for now) understanding of the meaning of the fields and the data within them between the source and the target data. Our example data may have high Exactness as the field named between the two datasets closely match and seem to describe the same data. However we may have a Data Quality issue if the *use_yn* field in the source dataset is meant to express whether that record is useful in the target dataset and the *use_yn* field in the target is expressing that the data for that record is useful in a particular report or application. In that case those fields would fail an Exactness test and you could say that only 3 of 4 fields in the target data set were Exact.

Exactness is a lesser used dimension that might come up as data proliferates to independent data stores, each of which can have their own business rules and naming conventions. It is important to keep all data consistently named with what it means. Duplicate names that mean different things within the same are a fantastic way to make poor Quality Data when one developer or team makes the wrong assumption.

3.2 Other Data Fidelity Dimensions

There are a couple of Data Fidelity Dimensions worth mentioning that are difficult to apply direct measurement to but are worth keeping in mind.

3.2.1 Cross data connections

Data that is related between datasets. For instance, if one field is dependent on the values of other fields, or even on an aggregation of data in other datasets, you'll need to develop bespoke profiling rules to validate the data. Many of the measurements you can make in pursuit of Data Quality fall into this category.

3.2.2 Timeliness

Timeliness is a measure of the time it took to update data in a target source dataset from when it was updated or created in a source dataset. This usually requires a comparison of explicitly added fields from the source dataset that record when it was last updated and similar fields in the target dataset for the same record. The gap between the two is your timeliness for that record. An overall accounting of the timeliness of each record in the target dataset should identify situations when data is taking longer than normal to propagate or even when the copy has failed if the time the source was updated is later than the time the target was updated by more than an acceptable amount..

3.2.3 Data Legacy or Data Lineage

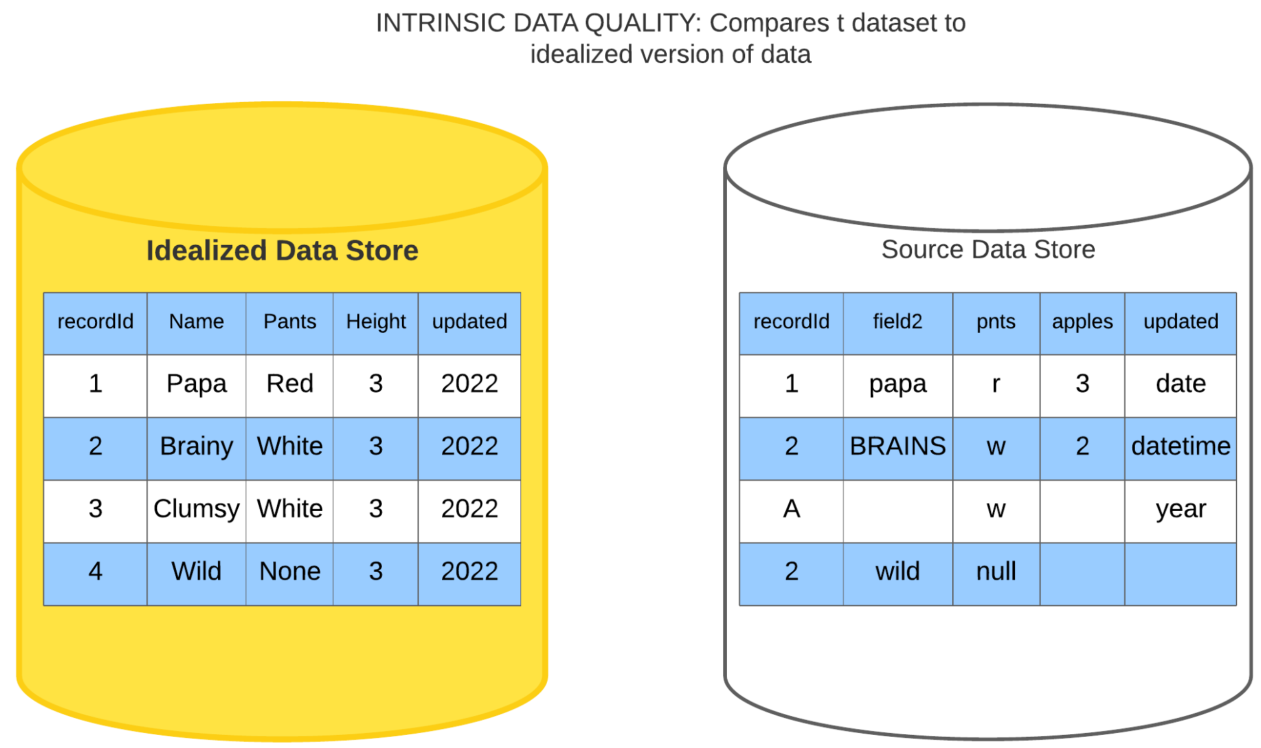
A record of how the data got from the source dataset to the target dataset that you are testing from the first time that it was entered or generated by your software and through any and all transformation steps. A data legacy or lineage can be generated from a variety of tools and can be extremely useful in troubleshooting data problems. Data Lineage is difficult to measure as it usually requires programming the collection of data on how records moved through the system which creates yet more data that needs QAed. The updated field in our example is a rudimentary kind of data lineage that tracks when a record was last updated in the target dataset. More complicated ones could include timestamps for when data was modified, what process modified it, and almost anything else that is deemed relevant.

4 Intrinsic Data Quality

Once progress was underway to measure our Data Fidelity issues we turned our attention to finding ways to validate a dataset when we could not simply compare it to some trusted source data. As with our Data Fidelity problems there were a number of high profile defects that pointed to weaknesses in our understanding of our data and its consumption. In one instance a number of frequently used reports were displaying data that did not agree with each other even though they were using the same source data when generated. In other cases we had reports from data analysts that they were spending up to “80%” of their time scrubbing and cleaning data from our production data stores before beginning analysis. For these we needed to examine qualities within our dataset not simply compare them to data from which they were copied and assume our sources were flawless.

In the examined literature the most frequent dataset for which quality was a concern was a defined dataset that someone needed for a technical or business operation, a purpose. In the case of our company, that was data gathered from school districts detailing student, school, teacher, and class information, relationship data about which students were attending which schools, taught by which teachers, and other details of school district operations that needed to be reflected back to the customers and augmented with our own data. Whether it is school data, or online order details, or bank transactions the data overall needs to be of high quality to allow for operations to run smoothly and for later data analysis to aggregate and sift in the hopes of finding patterns for further business opportunities. According to our definition of Data Quality in order to determine the quality of an independent dataset, even indirectly, we must compare its actual state with a desired state. In other words some other dataset. a comparison of the actual state of a particular set of data to a desired stateWhen you want to measure the quality of an independent data set you compare it to an idealized version of that same data.

The Idealized data need not be actual. It's often sufficient to have a description of what the idealized data will look like and go from there. For instance, identifying and putting limits on which fields in a dataset could be NULL or empty and identifying what percentage of empty datum in a given field across the dataset gives a good indication of whether or not you are missing data. A common approach to defining what goes into an idealized dataset is to establish a data dictionary that explicitly lays out, in technical, business, and common language, what the meaning and limits for all fields in the dataset should be. Each method of idealization is a Dimension which could potentially be measured by doing full scans on a dataset and collecting aggregate values for which records or fields fall outside of the ideal state..


INTRINSIC DATA QUALITY: Compares t dataset to idealized version of data
Idealized Data Store
Source Data Store

recordId	Name	Pants	Height	updated
1	Papa	Red	3	2022
2	Brainy	White	3	2022
3	Clumsy	White	3	2022
4	Wild	None	3	2022

recordId	field2	pnts	apples	updated
1	papa	r	3	date
2	BRAINS	w	2	datetime
A		w		year
2	wild	null		

Figure 3: Dimensions of Intrinsic Data Quality

4.1 Measures of Intrinsic Data Quality Dimensions

As mentioned there are many data dimensions. In fact some of the more useful ones are probably defined more by the business specific nature of the data that is being analyzed than by any standard set. I wanted to focus on specific and somewhat generic Dimensions that can be easily measured with automated software tools. Note that not all of these dimensions apply to all datasets.

4.1.1 Completeness

A "complete" record would have values in every field. Incomplete records are missing data. Completeness overall is a measure for each field of how many records are missing data in that field. A usual example would be US addresses. Each record is presumed to have a street number, street name, city, state or province, zip code and some optional fields like apartment number. A dataset of addresses could be measured for completeness in every field and compared to expected results. All records should have a zip code, city, and state or province record, so any missing those values would be considered incomplete. Other fields are usually, but not always, required (yes, even street names and numbers can be omitted in strange situations) so your dataset may have 99% of records with street names and still be considered within quality bounds. Apartment numbers would be lower still. If 30% of your records have apartment numbers that might be high quality or low quality depending on the nature of the data.

In the example shown in *Figure 3* a complete record would be expected to have data in every field. Comparing that to the dataset that we're testing we can see that a number of the fields are missing. Under field2 we find 1 out of 4 records are missing. Thus we can say that that field for this dataset is only 75% complete.

4.1.2 Uniqueness

Records in a table are unique if they have identifiers and values which are not duplicated anywhere else in the dataset. Identifiers generated and used by data stores often enforce uniqueness and will not allow the storage of non-unique values in a field that is an identifier. Other fields are not so lucky. In either case it is still recommended to count unique values from your dataset and report a number of unique values over the total number of records.If those values don't match then some values are not unique. Most fields in a dataset will not be expected to have unique values, but those that are defined as unique usually MUST be so and having duplicates can cause a number of high profile data defects.

4.1.3 Validity

We can say a datum is valid if it is of the correct data type, format, value range, and precision as would be expected for what the datum defines. A price, for instance, would be valid if it was expressed in a legitimate currency (US Dollars), was a number data type and had two decimal points of precision. A negative price would also be invalid, as would a price with two decimal points, or a non-numeric symbol. In *Figure 3* the data in the updated column would all be invalid as they are the wrong type, being descriptions of dates, not years. Measure the validity field by field in a dataset by seeing if it compares with the established pattern for values of that field as established by convention and business rules and documented in a data dictionary. The number of valid fields divided by the number of fields tested is a validity value for that field.

4.1.4 Orderliness

Data is "orderly" if it conforms to the required structure defined for the data.This is a great dimension to use for higher order structured data like JSON or XML fields. Similar to Validity, high quality data that is Orderly would match a pattern of how the data is supposed to be structured according to a data dictionary or other business rules. A simple example would be a 16 digit credit card number that should appear as 4 sets of 4 numbers. Orderly data would match that pattern while less orderly data might omit the spaces between the data sets at times, string the numbers together, or have less or fewer numbers.

4.2 Other Dimensions not as easily measured

There are a number of important Dimensions of a dataset that are useful to consider but hard to measure in an automated or manual fashion. Nevertheless the presence of these dimensions in a dataset can enhance its quality.

4.2.1 Auditability and Legacy

Data that is "auditable" contains fields and data in each record that track when the record was last changed and by what user or service changed it. More sophisticated systems can even keep and continually update a list of all changes to a record from its initial creation until the current time, which is referred to as a legacy. While it is more in the realm of functional testing to validate that the audit fields are updated correctly when records change, their presence in a dataset enhances the ability to troubleshoot data issues and determine how poor quality data entered the system and what user or service is responsible.

4.2.2 Privacy

Data is private if it is only accessible by authorized users and services and is otherwise protected. One could measure the effectiveness of Privacy conditions on fields by attempting access during Profiling with both authorized and unauthorized users.

4.2.3 Accuracy

Accuracy has a variety of definitions. A common one is to say that data is Accurate if it exactly describes the real world object or process which it represents. That is extremely difficult to measure with any sort of automated process, or manual ones for that matter. In some cases Accuracy can be measured if an established source of absolute truth can be used instead of an idealized dataset, for instance postal addresses can now easily be validated against the list of all valid addresses kept by the USPS. That does not tell you if that's the correct address for a particular person though.

4.2.4 Name Correctness (Aptronymity)

Names of fields describe the data the fields contain. Names of datasets should describe what the individual records within the dataset collectively describe. When either the data in the field changes to reflect something other than the field or dataset name describes or when business changes the names of terms and the datasets don't keep up with those changes then the names in your data become misleading. At best they are quirks of your data model that must be consistently kept in mind when working with it or when consuming that data. At worst they can become active points of confusion that consistently cause defects.

5 Meta-Data Quality thinking

Once we had build some data profiling tools and set teams loose on making data quality measurements we needed to start thinking of ways we could improve our development and data handling practices to improve quality. The first method, to make actual measurements so we can make quality more visible, was finding more defects and resolving more issues than years of customer complaints had. Having a set of measurements taken over time created a baseline of Data Quality that we could use to determine if any other initiatives to improve Data Quality were actually working. That, after all, was the main reason to focus on measurements.

So we began looking more closely at the list of defects that had caused issues and tried to get a deeper understanding of causes. In many cases we found that development teams had made mistakes in programming based on faulty assumptions of what data within the system meant or was used for. In others we found situations where business rules restricting data movement were established some time in the past, poorly documented, and then forgotten completely by which time the rule looked more like a defect than an intentional omission. Specific defects were caused by incorrect assumptions made about the data they were ingesting which then became bad test cases based on those assumptions. Our customers knew better.

I began putting together a list of better practices that would hopefully lead to better Data Quality, a kind of meta-Data Quality practice. We are still refining the practices and in the end these better practices may not be better. We can only tell by leaning into the measures that we're gathering from the various dimensions as well as the ultimate lagging indicators of quality, Production defects and customer complaints, to tell if any particular practice is providing higher quality data or not.

5.1 Hopefully Better Data Practices

5.1.1 Data Stewardship

Data Stewardship is the practice of assigning a subject matter expert, called the Data Steward, to be responsible for the Data Quality and lowest level Data Governance of particular datasets. The Data Steward needs to be knowledgeable about the data they are responsible for, understand what it represents in the real world, understand what it means for the business, how it's been transformed, and how other services are using the data.A Data Steward can help a development team understand their data and what it means which I hope will prevent the kinds of defects we saw that were caused by developers blindly processing stories to transform data without really understanding why. A Data Steward can also help run periodic profiles on data they are responsible for and process the results for broader consumption..

5.1.2 Data Dictionary

Build and maintain a Data Dictionary which details the names, meaning, and business and technical rules that apply to each data field and dataset. Keep it up to date. This is a great way for the Data Steward to communicate to teams about what the data means and how it's being used. It is also a key component in several Data Quality dimensions like Validity, Orderliness, or Uniqueness.

5.1.3 Track your data defects

Prior to our Data Quality initiative we did not categorize our defects based on anything other than the service that caused the problem and the team responsible for fixing it. Now we have implemented a field on our defects which allows us to track whether a defect is related to data issues, software functionality, UI, or other categories. This allows us to better analyze our data periodically to see if our Data Quality efforts are paying off in a reduction in data related defects in total or as a percentage of overall defects.

Since we have started tracking our data defects, which coincided with our other efforts around data profiling and measurements we have seen a marked reduction in the number of data defects being reported and have had no high profile defects..

5.1.4 Implement Data Quality checks within your data pipelines

One of our teams found great success in building their data profiling into the build pipelines they have to deploy new code to production. Running each time they deploy new code and on a nightly basis their tests measure for Completeness, Validity, Uniqueness, and Consistency of a presentation layer of data that one of our reports depends on. They have then taken that data on their Data Quality (talk about meta-data) and created a dashboard compiling it and showing recent and past results, trends, and other automated tests to get an overall view of the project and data pipeline health.

5.1.5 External profiling

Run external profiling on your data. Do not just depend on the QA testing that went into testing the services that move or manipulate the data. Especially with microservice architectures weird things can happen that cannot be tested or predicted from the QA of individual services alone. You need independent testing of your data itself to validate its quality.

Once you have set up profiling to run on a regular basis, capture the results and routinely analyze them. Compare profiles across time as the data changes to get an idea of what your stable baseline is in measurements like Completeness or Timeliness. Further your Data Quality journey by combining your profiling data with the release dates of data impacting services, process changes meant to improve Data Quality, or anything that may alter Data Quality to try and quantify its impact, positive or negative.Combined with an easy reporting dashboard it's a great way to understand where efforts to improve Data Quality are working and where they are not.

5.1.6 Build and ship manifests

When moving a lot of data in bulk include additional meta-data like the total number of records being sent and aggregate values for that set such as a MD5 hash of all values in a particular field. Receiving services in data pipelines can then check that all data was received and appropriately processed by comparing what data they received against the meta-data..

5.2 Practices to avoid

In addition to practices we hope will avoid some of the problems we saw that caused issues with our Data Quality we found a number of practices that seemed to directly contribute to reducing Data Quality. These are to be avoided, if possible.

5.2.1 Multiple tools to ETL/ELT the same data

A practice that we've seen repeatedly was for a team to develop a data pipeline to ETL/ELT data from a source data store, the parent system where data originated, to a target data store, a new system that needed data to operate, on a periodic basis varying from 5 minutes to daily. The data pipeline was rigorously tested and exercised to validate that it was capturing all data additions and updates in a source data store that occurred in the data since the last time the pipeline had run and that the pipeline was able to build up the data in the target data store to be an accurate copy of the source.

Teams then determined that the data pipeline, as designed, was insufficient to move the existing mass of data in the production source data store to the target. So they proceeded to develop a separate tool to load all historic data from the source into the target as an "one time" initial data load.

Inevitably, when Data Quality issue arose this one-time tool was called into action to realign data that was missing or flawed by forcing all data for a particular customer from the source to be refreshed in the target data store. Now any changes to the business logic had to be applied to both tools, both tools needed testing routinely, and any time a Data Quality problem arose with the data in the target data stores we would have to determine which tool was causing the problem.

We recommend that this is a practice to avoid. Rather than have multiple tools to copy the same data it would be preferable to have one tool that applies the same transformations to all the data that it is moving and can be configured to switch between updating with newly entered data or historic datasets.

5.2.2 Copying partial datasets between data stores

On several projects over the years we made decisions when developing ETL/ELT data pipelines to only move part of a dataset between two data stores. The usual logic at the time was that we wouldn't ever need the data left behind in the new data stores.Inevitably as use cases changed for what the new service was doing and people forgot the original constraints the data that was being left behind was wanted, or worse, was just assumed to be there and defects filed when it was found to be missing. Development that could have easily handled the missing records when they were originally omitted now had to be forced into newer logic that never anticipated the ways the left behind data was different. .

For instance if you have a source dataset of Orders which are in a variety of states or possibly even Invalid do not build an ETL service that only pulls the Valid orders for reporting purposes into a new reporting focused data store. Bring all the orders, similarly transformed, and filter them at the source when generating reports or build a more ELT style service that loads all the Orders into a raw staging data store that is then used to generate more useful data from which Invalid Orders can be removed.

5.2.3 Records lacking a cross data store unique identifier

If you do any ETL/ELT operations between data stores, make a unique identifier for all records of data that is agnostic to the data store. This enables you to very easily and concretely track that data as it moves through different data stored by unique id. This is in addition to any data store specific identifiers that you might need.We call this unique identifier the “business identifier”, or BID, other places refer to this as a master ID. Not having a business identifier in some of our data has resulted in confusion and time wasted on troubleshooting that could have easily been avoided if a BID were present. Further we recommend that if you have a BID make it UUID type as it is less prone to be duplicated accidentally.

5.2.4 Keeping existing database field names when business changes the name

Time rolls on and businesses love to change their nomenclature. A company may rebrand or may change the names for the data that it provides to customers for better clarity. For whatever reason if you have a data field, a dataset, or even a whole data store that is named for something that made sense years ago but no longer reflects the business operations that the object is used for or describes- change the name, and track the old names in the data dictionary. To not do so builds a continual debt of historical understanding that all new and existing employees must be taught and keep in mind when working with the data.

6 Conclusion

Data is an essential component of business operations. To ensure that data is of high quality, reliable, and fit for purpose it is necessary to test your data apart from the services that create and manipulate it. Testing data to determine its overall quality requires techniques unique to the task by comparing your data to known or idealized datasets and making specific and regular measurements of Data Quality dimensions independent of other QA efforts to ensure that high quality software is developed. Further, you can support good measurements with good governance of data by Data Stewards and Data Dictionaries. Once a baseline of Data Quality is established you are well positioned to try better data handling practices or retire poor practices and see how your Data Quality is impacted.

Data Quality, all the cool kids are doing it. We implemented data profiling and measurements in a number of teams with problematic data and saw a marked improvement in Data Quality as measured by a decline in Data related defects and improvements in the measurements that we were taking over time. Meta-data practices to help team members know where their data comes from, to know where it’s going, and to know what it means have helped them build in better quality during development and avoid common pitfalls.

Data Quality issues are an overall detractor from business efficiency. Conversely high Data Quality, proven by routine measurement, gives leadership confidence in their data and data driven decisions, customers confidence in your product, and developers confidence in their software’s data handling abilities.

References

Data Quality. (2021). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Data_quality

Redman, T. C. (2008). *Data Driven: Profiting from Your Most Important Business Asset.* Cambridge, MA: Harvard Business Review.