Paper DH03

Educating for the Future: Data Engineering Techniques

Guy Garrett, Achieve Intelligence Ltd., Brighton, United Kingdom

ABSTRACT

Clinical development has made huge efficiencies over recent decades (e-CRF and CDISC standards to name a few). However, just as we are reaching the summit of traditional techniques we are faced with another step-change in our journey – the big data mountain.

The methodologies we are currently using cannot cater for these vast and varied data sources without a huge increase in resources, something which most biopharma companies have not provisioned for.

Common data management and statistical programming techniques need to give way to a centralised automated data engineering mindset – including transformative data consolidation, integration and access – which goes beyond cost reductions, to enable a new phase of data-driven clinical discovery.

This presentation will discuss a step-by-step approach data managers and statistical programmers to move from traditional methods towards the new, preparing themselves for more innovative and varied data sources whilst supporting business-as-usual submission work.

INTRODUCTION

Data Engineering has been growing in popularity in the BioPharma industry ever since the introduction of Real World Evidence to support clinical development. Coupled with the step-change in organizational strategy, with data becoming a central asset, as opposed as a single means to an end with regards the regulatory submission, data engineering is becoming the must have skill to support holistic data science practices.

Much has been written on the subject, and other industries have successfully adopted data engineering techniques, however BioPharma has some way to go to make data pipelines and centralized data stores the norm. In 2018 the **PHUSE Data Engineering Project** was created under the **Educating For the Future Working Group** to research the topic area and understand what benefits can be realized within our industry, based on successful use cases in other industries.

Following an introductory paper presented at EU Connect 2018 and US Connect 2019 the project team have been investigating specific techniques from within the area with the objective to apply them to the clinical data landscape.

This paper is the result of those efforts, where we look into the following sub-topics:

- Master Data Management
- Extract, Transform, Load Techniques
- Data Trawling
- Data Targeting

Each section will provide definition, explore the practicalities of the methodology and provide a use case of how this technique could be adopted within the clinical data landscape. This is not meant to be a prescriptive paper of how the industry should adopt these techniques, but rather an example of how they could be adopted with an indication of the pros and cons associated with adopting these approaches.

MASTER DATA MANAGEMENT

Per Gartner "**Master data management (MDM)** is a technology-enabled discipline in which business and IT work together to ensure the uniformity, accuracy, stewardship, semantic consistency and accountability of the enterprise's official shared master data assets. Master data is the consistent and uniform set of identifiers and extended attributes that describes the core entities of the enterprise including customers, prospects, citizens, suppliers, sites, hierarchies and chart of accounts".[1] Master data can include relatively static reference data, transactional data, metadata, unstructured and analytical data.

To successfully implement an MDM effort, the engagement and 'buy in' of key stakeholders in the organization who either own master data or are consumers of master data is critical. Managers of the MDM effort should have a clear understanding of the expectations and needs of stakeholders so these can be incorporated into specifications. Without a strong governance model with clear roles and responsibilities, the integrity of master data may be jeopardized. Experts recommend that organizations develop a common view of the metadata "harmonization of the definitions and contextual semantics to ensure that data users are comfortable relying on the shared data". Management of data quality and alignment with various functions to consume the master data as single version of truth can ensure an organization achieves a return on it's MDM investment. Establishing a robust feedback mechanism, that monitors the process and facilitates process improvement, should be considered as well.

PROS & CONS OF MASTER DATA MANAGEMENT

The table below highlights some key pros and cons for an organization considering MDM:

PROS	CONS
Data entered in one system, used across organization	'Buy in' across functions and departments across the organization to realize return on investment
Clear ownership of each 'master' data element	Effort to maintain robust governance of data that is sustainable over lifecycle of data
Can be integrated with other systems over time	
Ongoing 'cleaning' of master data to ensure it is current	
MDM techniques can offer efficiencies to larger organizations where data are distributed and growing due to integration post mergers and acquisitions	Organizations as they evolve will reach a stage where multiple tools or systems are deployed. Frequently, there is redundancy in systems, conflicting data may be captured which can lead to erroneous assumptions and decision making.
Reduction in errors and redundancy, in business processes supported by accurate and consistent data that can be achieved within organizations by following MDM.	

Many large pharmaceutical companies are already implementing MDM in the commercial area to manage customer and sales data. Within Research & Development, potential use cases for master data may include data about clinical trials, sites and products. Large sponsors conduct a few hundred trials annually. A single 'master' repository to capture key data points about the trials, sites used for various trials, therapies that were tested in the trials can facilitate responses to regulatory bodies, trial design of new trials and identification of sites for new trials.

ETL OVERVIEW – "TRANSFORMING"

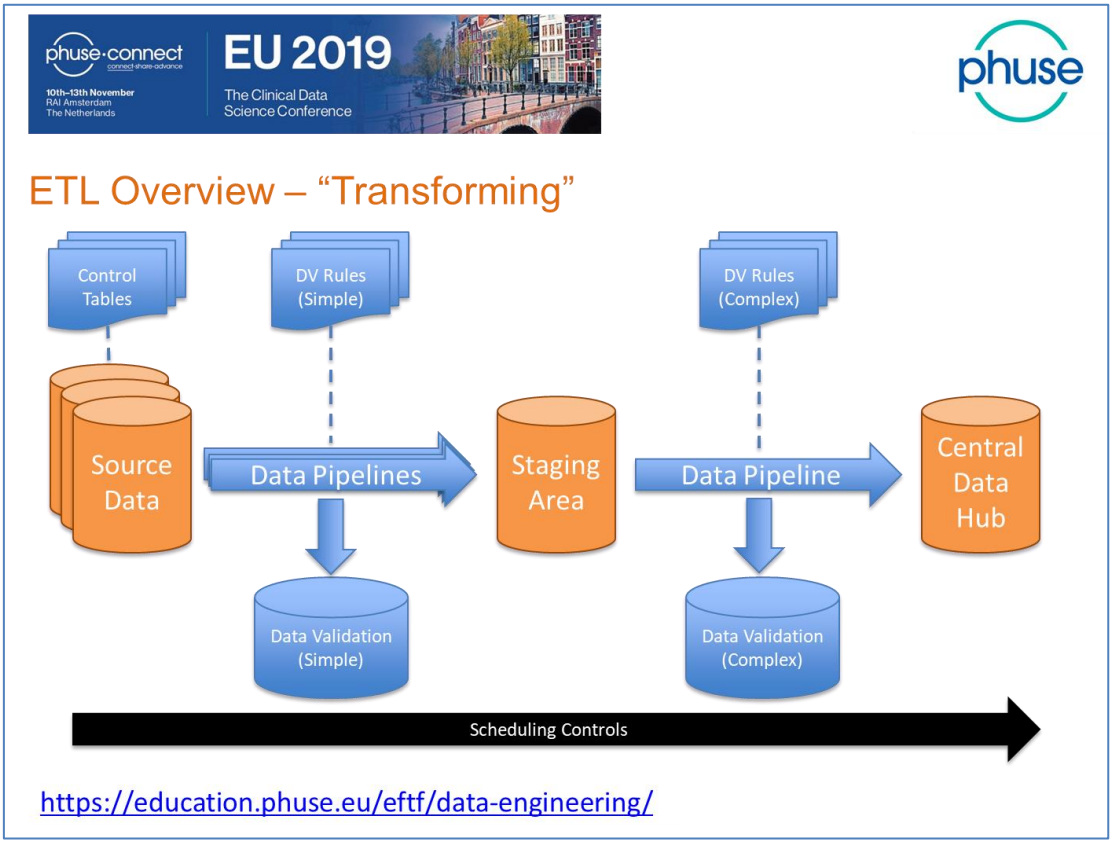
ETL DEFINITION

The concept of Extract, Transform & Load (ETL) was developed early in the history of data warehousing. Following the conversion of manual paper-based systems to computer-based systems in the 60s & 70s the focus moved towards measuring the information collected by these operational systems. Statistical Analysis became a sought-after skill for organizations wanting to produce metrics to determine a) how well the company had performed in the past and b) how well the company is likely to perform in the future. In the BioPharma industry the focus was obviously to determine the efficacy and safety of a potential new drug based on a sample of study cases.

However, much of the statistician's time was taken up collecting information from all the disparate operational systems in order to measure and make predictions from siloed, disparate data. Thus, during the 80s several approaches to automatically collate the data into a centralized data store, modelled in such a way that lent itself to easy analysis. The data warehousing revolution was born.

Several proponents of data warehousing put forward their view on the most appropriate method for collating data centrally. Ralph Kimball developed the bus architecture approach, to create views on operational systems which extracted information when queried, Bill Inmon formed the approach to extract copies of data from the operational systems and manage any changes to the underlying operational data via change data capture methods. Most organizations took a hybrid approach to this and combined bus architecture with the extract method and collectively referred to ETL as the methodology that populates the data warehouse.

FIG 1. HIGH-LEVEL DIAGRAM OF ETL COMPONENTS



STAGING

Through the 90s and 00s more hybrid approaches were developed such as ELT (the concept of extracting raw data and loading (or "staging") it to the data warehouse, with transformations happening after loading and creating data marts for most popular analysis. Staging was used normally overnight or monthly to ensure that a specific time-slot extract is taken from the operational systems. The data transformation routines could then continue working off a static set of data, rather than worrying about operational data changing and bringing in inconsistent figures between one data mart and another.

The term ETqL also crept into the popular terminology to highlight the importance of a robust, measured and governed data quality framework in the process.

Along with the plethora of three-(or-four)-letter acronyms new methods of collating data, such as data streaming & data targeting (and even hosting data in physical different locations, yet with a logical centralized view) it was clear that a new term was required for the method of gathering operational data in order to perform analysis effectively. As the term Data Science was gathering momentum across industry the term Data Engineering took form – which reflects the level of automated, structured processes required to provision data for the Data Scientist to perform their analysis.

CHANGE DATA CAPTURE

Change Data Capture is the technique to recognize when change has occurred in the source data and whether to import that information into the centralized data store.

A "change" in this instance could be:

a) The addition of a new record
b) An update to a record
c) The removal of a record

Traditionally pharma have worked in silo's i.e. when a new iteration of data is available from a site it "passes" it along with any previous records to the data management team, who in turn run their programs to perform data validation/cleansing and then "pass" the next iteration of all the records collected so far to the stats programming team.

This method incurs re-testing, query management "what happened to this record?", and re-work if field names change, or ad-hoc fields appear.

The data engineering approach is to create a pipeline where only updates flow along to a centralized data source, rather than a whole new cut of the entire database. Identifying which records have changed, or are new, or indeed have been removed, is this concept of change data capture. Or in other words "How has the data changed since last time we took an extract?".

There are several ways to perform change data capture, the most common being:

- Database Timestamping
- Delta Identification

Database timestamping refers to the source system containing a timestamp of some form on each record in the database. Often this is the datetime which the record was loaded into the database. This field can be used, along with a control table which records the last time an extract was taken to determine which records are new in the database since the last extract. The risks to this approach are minimal but could involve loading duplicates records or indeed missing new records if the control table gets corrupted in some way. To mitigate against this risk companies also perform record count validations separately – to ensure the correct number of records are generated for each extract.

Delta Identification is a method where the data source has no timestamp upon which to create an extract. Therefore, an additional step needs to be performed to match the records you've previously extracted against the source system and determine if there are any new records or updates on existing records in the source system. This is a less efficient method of generating deltas, and therefore more appropriate for smaller databases in the region of tens of thousands of records rather than millions.

BioPharma faces specific challenges, regarding the updating or removal of records, due to regulations. Therefore it is essential that audit logs are recorded to record any changes captured from the source system. However, if this is performed by centralized, batch activities, sequestered from normal user activity, then authorities may be more likely to look favorably on this approach.

DATA VALIDATION

Data validation is an essential element to the data pipeline approach. Rather than having an individual (or indeed teams of individuals) manually checking data items as they arrive in the organization it is more efficient to configure a set of validation checks at appropriate points through-out the pipeline.

Data Validation at Source: Many EDC systems have the capability to set data-entry validation checks, for instance, only allowing a valid date to be entered in a date field, and these should be set as standard on data entry screens. A balance needs to be made here, between the user experience and the quest for clean data. If a data entry screen is too "heavily" validated then there is a risk that users may bypass control for expediency's sake, for instance leaving a field blank rather than scrolling through long drop-down lists for a specific data item.

Data Validation during Transformation: Another form of validation is checking that any transformation performed during the pipeline process is valid. These can be a variety of checks, such as ensuring records have not been dropped, calculations are correct, and standardization is valid (for instance text addresses conformed into a recognized standard).

Cross-Validation prior-to load: Another type of data validation is ensuring reasonability once the data has been extracted and transformed. This often considers checking between data sources and is the "belt and braces" for data validation. Examples of this could be ensuring pregnancy trials have no records relating to male patients. The correct level of effort needs to be applied to this type of data validation, as it's often missed out due to time-pressures. However, issues found at this stage can lead to identifying problems with the data pipeline, which could go undetected otherwise.

Finally, on the topic of Data Validation, here are few practical techniques recommended to establish your data validation strategy:

- Conduct a spot check on a snapshot of data (e.g. for a day) at an aggregate level between the source and data warehouse. e.g. check total number of records, are the counts consistent?
- Data that is similar but coming from different sources can be cross checked for consistency. e.g. if multiple systems have data on clinical sites (address/staff names etc.), these can be cross checked against each other and as part of data cleaning also combined to create a 'golden' record
- Establish a central mechanism to track and store issues so data managers/users/stakeholders have a single place to refer to
- Conduct the validation steps prior to loading data into the warehouse, to proactively avoid loading erroneous data which are then potentially used in reporting, thus leading to bad decisions downstream
- Build some automated metrics that run on a periodic basis (daily, weekly, monthly…) that allow users to monitor the process. e.g., number of records loaded, % of records stopped from loading due to checks firing; which data points have higher frequency of checks firing etc.
- Think critically about the workflow to load data from source systems into a warehouse, not just about the data per se, so the workflow facilitates and supports a robust data warehouse.

SCHEDULING & BATCH PROCESSING

Most of the techniques discussed so far bring benefits to creating data pipelines for populating and updating an organization's conformed data into a central store of data. Benefits can be further realized by automating these processes and methods, thus releasing the data engineer to consider how to deal with non-conformed, ad-hoc and more challenging data items.

Data update programs are configured into a schedule with one program (often referred to as a "job") being dependent on the successful completion of a previous job. If there are many data sources and transformations required then certain schedules can run in parallel, thus reducing the time required to execute the entire suite of jobs. For instance if several datasets need to be extracted from one database and transformed before merging with another set of datasets from another database, then two schedules could be created (one for the extract from each database) and then a third schedule only triggered to run once both of the previous schedules have completed successfully.

Triggering jobs & schedules is fundamental to the organization of the data pipeline update.

There are several types of trigger:

- **Time triggers** are the most common form of schedule trigger to initiate the execution of the schedule. Often suites are configured to run monthly, weekly or daily, however some cases hourly or even quarterly schedules make sense – depending on the application.
- **File triggers** are another common type, which will begin the running of a schedule based on the arrival of a file in a pre-determined location. In order for this to be managed going forward the schedule need to include a final job to remove/archive the file, ready for the next time the file arrives.
- **On Successful Completion** is another type of trigger, which is commonly used for jobs within a schedule. Jobs only execute if the previous job completed successfully (based on a return code of some sort). Normally suites are configured that if a job fails the next job doesn't execute – to ensure the central datastore is not populated with corrupted data. Often alerts (possibly automatic emails being sent, or problem records being written to an audit log) are activated on a failure also – so that the data engineer is immediately made aware of a problem with the suite.

There are many scheduler software applications available in order to configure and run automated program suites, which have capabilities to set triggers, organize workflows and send alerts.

Configuring automatic and regular updates of data, ensures that up-to-date data is available for downstream processing such as safety reporting and analytics. Update frequency can depend on several different types of trigger, depending on the appropriate application.

DATA LAKE – "TRAWLING"

DATA LAKE DEFINITION

A data warehouse holds 'structured' data which are based on a pre-defined schema (usually star or snowflake). Data warehouses can accept incremental or cumulative loads of data.

A data lake is a repository that can accept data in any format, hence data can be stored in their native format with no transformation/mapping required. There is no pre-defined schema, however, metadata associated with the data 'flowing' into the data lake are essential for users of these data. Each data point ingested into the data lake should be tagged with metadata that allows users the ability to search effectively. Advanced users who are familiar with the data can design good search strategies, however, if the metadata are not robust, lay users may struggle to fully leverage the data. Users should also realize that there are no checks in place as data are loaded to the lake so data integrity is assumed at the source. Furthermore, data lakes do not support incremental loads of data.

As organizations start collecting large amounts of different data sources, they need to store that data before it's analyzed. Traditionally, organizations store the data in a SQL data warehouse. However, in the context of "big data", when the organizations initially don't know how the data will be used, data warehouses become a burden.

Data lakes provide the following capabilities:

- Storage of different types of data for instance – SQL, Text, Binary (images/pdfs), or any other type of data
- Metadata Discovery – A centralized Metadata repository that can be used to identify/search data across the lake
- Analytics: Data Lakes provide capabilities to run analytics (such as data visualization, or machine learning) on the underlying data
- Fine grained security model – for managing access control

With cheap storage, and availability of large amounts of structured/unstructured data, data lakes became prominent in later part of 2000-2010 decade. Hadoop – which provides a distributed platform, along with HDFS (Hadoop Distributed File System) became the prominent technology for data lakes. With the advent of cloud, cloud-based data lakes provide easy entry point for organizations to start using a data lake.

Within the clinical trial data landscape, data lakes could be explored to store unstructured supporting data such as images from vendors capturing ECG's, venograms, EEG's, X-rays etc. Currently, these types of supporting materials are not available within a database or easily accessible to the sponsor. Capturing these data and being able to link these to clinical trial patient data could allow clinical teams to understand more about their patient data.

STREAMING

Streaming data architecture enables collection, analysis, and storage of streaming data in real time. Streaming data architecture provides following capabilities:

1. Ability to ingest streaming data in real time
2. Ability to provide fault tolerance – in case of failure
3. Ability to analyse data, and take actions in real time

At a high level, streaming consists of following components

1. Producers -> Producers refers to applications, systems that generate data in real time (streaming). Producers write data to a message broker
2. Message Brokers-> Message Brokers store messages and make them available for consumers
3. Consumers: Consumers are downstream applications that access the data
   For instance, consider a real time streaming application that analyses heart rate data in real time from patient mobile device, and sends notifications in case of an abnormal heartbeat. In this case, the producer would be the mobile devices, the consumer would be a real time analytics engine (such as Apache Spark), and message broker (such as Apache Kafka) can manage the communication between the producer and the consumer

There are 2 types of streaming architecture:

- Publish / Subscribe
- Queue

DATA LAKE USE CASE

Consider a future scenario where a pharmaceutical company is running a clinical trial, and collecting the following types of data:

1) Clinical Trial Data
2) EHR Data
3) Real time streaming data (from Mobile Applications that patients use)

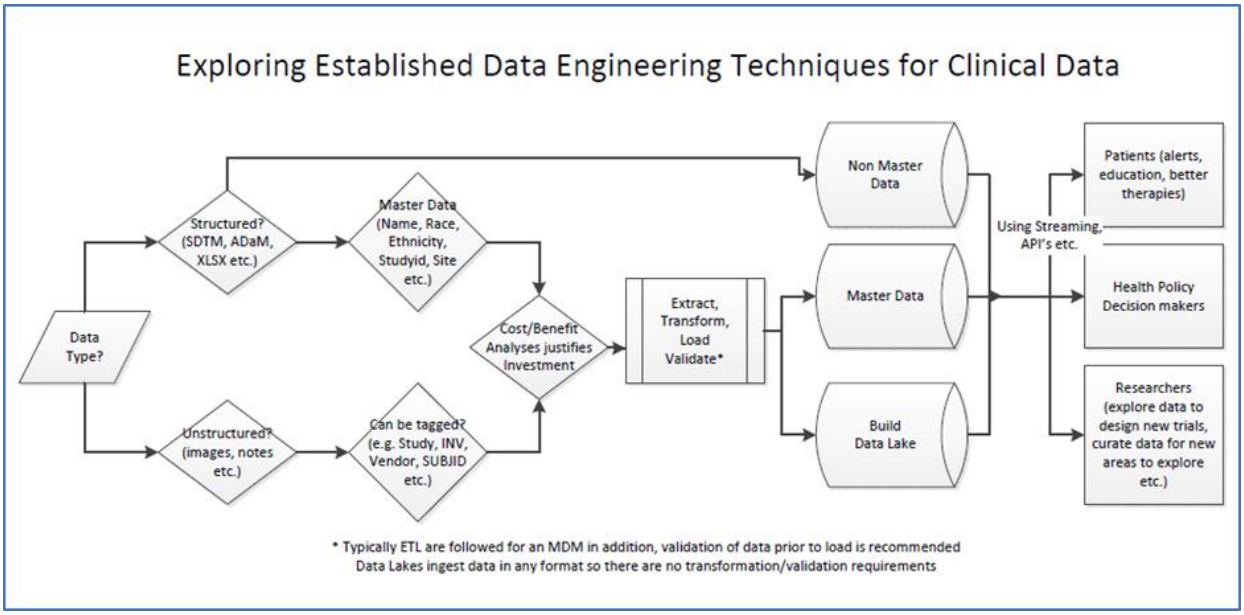
The business objective is to analyze patient information across the different systems.

The architecture would include the following components:

1) Clinical Trial Data:
   a. Every night data is pushed from the EDC system to a folder (data lake file system such as HDFS) in data lake through a batch ETL job
   b. Clinical Trial Data is in the form of SAS, CSV, SQL, or another tabular format
2) EHR Data
   a. Similar to clinical trial data, EHR data is pushed from source systems to a folder, on data lake
   b. EHR data in a tabular format is stored in another folder
3) Data from mobile app:
   a. Real time data from mobile app (producer) is pushed to a message broker (such as Kafka), where a downstream application that has "subscribed" to it reads the data and pushes it to another folder in the data lake

Once the data is in the data lake, the metadata repository for the data is created. Metadata repository stores the metadata for different datasets across the data lake. An analyst can execute SQL, Python, or code in any language to query the underlying data in the data lake based on the metadata repository that was created for the data.

FIG 2. HIGH LEVEL WORKFLOW AND POTENTIAL USE CASES FOR CLINICAL TRIAL DATA



DATA LAKE STORAGE APPROPRIATENESS

When data lakes first became popular there was a tendency to try to collect "absolutely everything" in case it may be of value someday. The reasoning behind this was that disk space is now relatively cheap, especially with cloud offerings.

However, with the revelation of certain "mis-conducts" in the area of data privacy and the introduction of stricter regulations (such as GDPR) this approach has been questioned in more recent years. It also raises a Green IT sustainability issue, with energy being wasted to store data in data centers which may never be utilized.

DATA MARKETPLACE – "TARGETTING"

A more recent development in the area of data engineering is that of the Data Marketplace. The concept here is that providers collect appropriate data for certain usage and allow users and researchers access to this data for various purposes. This data is quite often available for free, though some more "valuable" data commands a fee.

The method for extracting this data is often via RESTful Application Programming Interfaces (APIs). APIs allow developers to call methods & functions (packages of code) to return specific information. APIs vary from provider to provider often based on the expected desire of the data consumer.

Some APIs allow you to return multiple elements of data in one command. For instance, certain EDC systems have APIs that allow you to return all the Study Sites allocated to a particular Study. This type of request normally returns information in an XML or JSoN file format, which the data engineering program can then query, via interrogating tags associated with the data values.

Other APIs return a single string for a specific data element. This is more commonly used in apps, which are dealing with single item search queries as opposed to full extract analytics.

In general, this approach for targeting specific data items is gathering momentum, especially for sparsely populated specific information. When querying and entire database it is not generally as time-efficient as pulling a full extract of data, but if that option isn't available, or indeed if specific information is required for a smaller number of records then individual API calls are another approach which can be considered for data engineering.

CONCLUSION

In this paper, we provided an overview of selected established data engineering techniques and potential use cases that can be explored to apply these in the Biopharma industry. For data scientists working in clinical research, these techniques offer powerful approaches to storing, searching and using data for decision making. However, we recommend that before investing in any techniques, teams conduct a thorough cost benefit analyses and evaluate anticipated return on investment. For example, potential adopters of master data should consider governance and buy-in by key stakeholders. Teams should evaluate both short-term and long-term sustainability and applicability of these techniques along with flexibility to integrate with newer techniques in the future.

REFERENCES

1 https://www.gartner.com/it-glossary/master-data-management-mdm

ACKNOWLEDGMENTS

This paper is the product of volunteer effort from the Data Engineering Project, part of the PHUSE Educating for the Future Working Group, and not just the sole effort of the nominated author.

Special Acknowledgement goes to go to **Renu Shukla** and **Mohit Juneja** for much of the research and written content of this paper.

RECOMMENDED READING

https://www.informatica.com/services-and-training/glossary-of-terms/master-data-management-definition.html#fbid=9X0aV1w73Ob

https://www.sas.com/content/dam/SAS/en_us/doc/whitepaper1/master-data-mgmt-techniques-106161.pdf

https://www.sas.com/content/dam/SAS/en_us/doc/whitepaper1/sas-data-governance-framework-107325.pdf

https://whitepapers.em360tech.com/wp-content/files_mf/white_paper/wp_iway_7steps.pdf

http://www.tcdii.com/PDF/Data_Governance_white_paper_-_April_2006.pdf

https://www.sas.com/en_us/insights/articles/data-management/data-lake-and-data-warehouse-know-the-difference.html

https://www.forbes.com/sites/bernardmarr/2018/08/27/what-is-a-data-lake-a-super-simple-explanation-for-anyone/#63364cb176e0

https://tdwi.org/articles/2016/02/19/six-validation-techniques-data-quality.aspx

https://ec.europa.eu/eurostat/cros/system/files/methodology_for_data_validation_v1.0_rev-2016-06_final.pdf

https://resources.zaloni.com/blog/validating-data-in-the-data-lake-best-practices

https://www3.epa.gov/ttnamti1/files/ambient/pm25/qa/vol2sec17.pdf

https://www.fws.gov/Planning/Documents/Validation/DOI_Performance_Data_V&V_Guide_Jan03.pdf

https://aws.amazon.com/big-data/datalakes-and-analytics/what-is-a-data-lake/

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Guy Garrett
Achieve Intelligence Ltd
90-92 High Street
Evesham WR11 4EU
Work Phone: +44 (0) 7887 954496
Fax:N/A
Email: guy.garrett@achieveintelligence.com
Web: www.achieveintelligence.com