International Journal of Computer Trends and Technology
ISSN: 2231-2803 / https://doi.org/10.14445/22312803/IJCTT-V68I1P102

Volume 68 Issue 1, 5-10, January, 2020
© 2020 Seventh Sense Research Group®

*Original Article*

# Big Data Architectural Pattern to Ingest Multiple Sources and Standardization to Immune Downstream Applications

Imran Quadri Syed

*Lead Systems Developer, Information Technology Prime Therapeutics, Eagan, Minnesota, USA*

Received Date: 04 November 2019
Revised Date: 29 December 2019
Accepted Date: 31 December 2019

***Abstract*** *- In today's era where organizations are handling a large volume of varying data to meet their business needs. Also, Organizations receive data from numerous sources for the same data domain but in different layouts and formats. In this article, we will go over a Big data architectural pattern that immunes traditional downstream systems of any change to the source system. This is achieved by Datahub (big data) by ingesting data from different sources, standardizing to denormalized canonical form, integrating with reference data, rejecting reprocess and publishing extract using big data technologies like hive, impala to traditional downstream systems. This article also discusses how key management service (KMS) is utilized to identify the latest iteration of a record and to achieve easier querying and then generating standard publications for downstream systems.*

***Keywords*** *- Big data, Data Ingestion, Data Integration, Standardization, Reject Reprocessing, Architecture, Key Management Service (KMS), Hive, Impala, Datahub, publisher-subscriber pattern.*

## I. INTRODUCTION

Big data refers to massive volumes of data being collected and parallelly processed that traditional databases/tools could not feasibly handle. In today's world, regular transactional data is only a subset of data that organizations use to make statistical predictions and business decisions. Today there is data being generated by sensors, mobile devices, online interactions, cars and many others. Large organizations like to use this data in real-time for their business needs. For example, to predict travel time, avoid traffic & closed roads under construction near real-time, data needs to be ingested, processed and results produced. All this is achieved by processing and analysing massive parallel datasets in real-time. Traditional batch ETL cycles that run once or a few times a day no longer meet the needs of organizations handling big data. In order to understand this massive amount of data and its characteristics are often explained by referring to 5 Vs. Volume: Organization's storage footprint is in Peta and exabytes as they collect, process and store a vast volume of data from different sources.

Variety: In today's world, the source is not limited to structured files or tables. Today data could be in various forms and shapes like audio, video, documents, logs, e-commerce transactions, Emails, communication signals from cell phones.

Velocity: Velocity is a very critical aspect of Big Data processing. Velocity refers to the rate at which data is being generated by sources like sensors, cell phones, e-commerce platforms and numerous other tools. For big data systems data, it is very imperative that the data is ingested in near real-time fashion as in many cases the data is relevant only for a short period of time, for example, data related to the weather, traffic patterns, stock market algorithmic predictions and any delay will make the data irrelevant for the purpose.

Veracity: Veracity refers to data quality and reliability. If the data is not reliable, then the data is not very useful for analytical and prediction purposes. Organizations should take steps to keep their data quality and integrity to the highest levels by having proper infrastructure, processes and controls to support the proper collection and processing of data.

Value: Organizations spend millions of dollars on setting up infrastructure for big data technologies. The core of this lies in generating value out of this enormous amount of data for their organization business needs and growth. The raw data ingested/collected from various sources is not going to be helpful to meet organizations goals unless this big data is processed/transformed for detailed statistical models to extract relevant information from this massive data. There are times were organizations would have to integrate this ingested data with their Mastered data from Master data management (MDM) systems to gather value from data.

## II. BIG DATA ARCHITECTURAL PATTERN TO INGEST MULTIPLE SOURCES AND STANDARDIZATION TO IMMUNE DOWNSTREAM APPLICATIONS

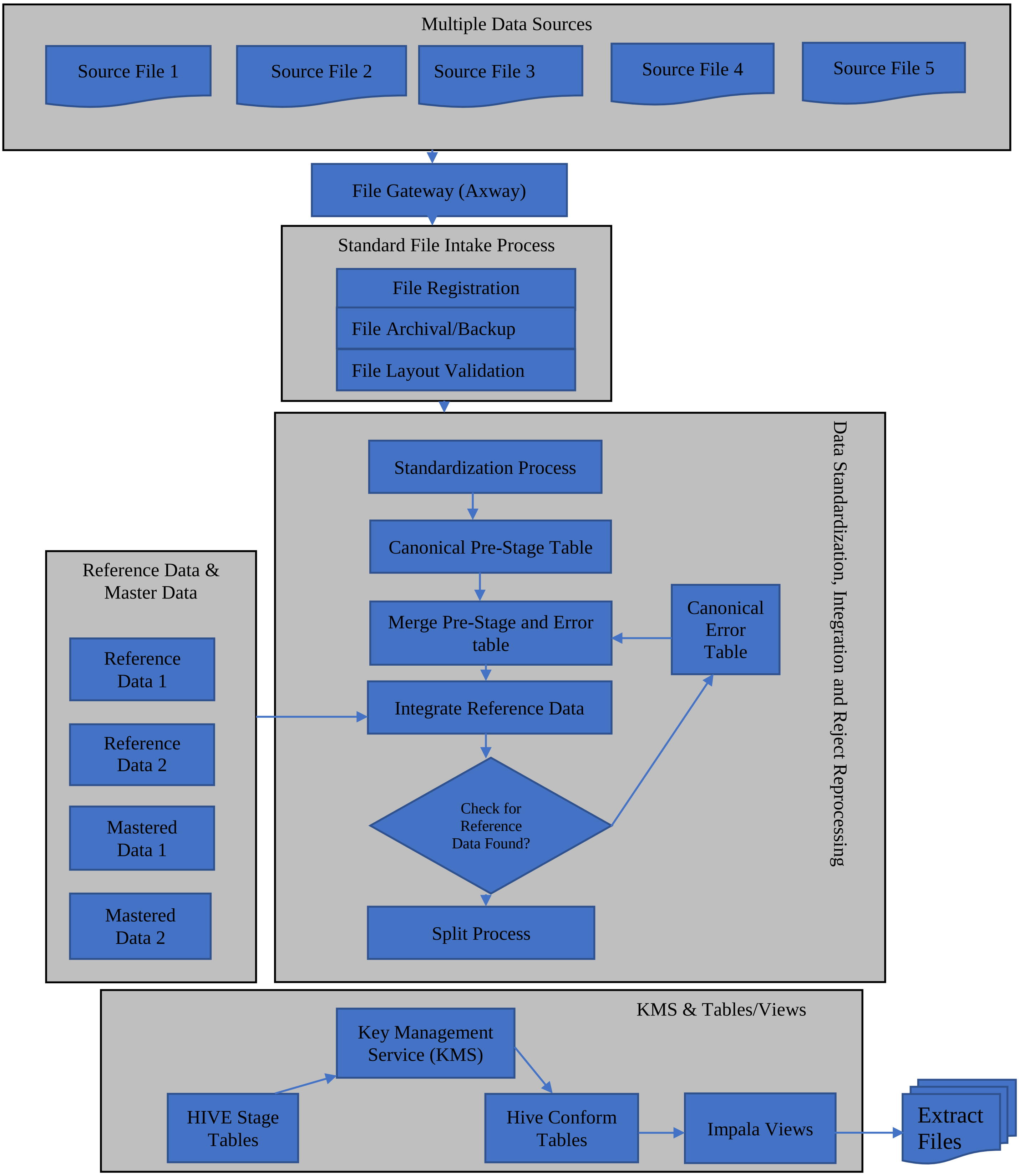

**Fig. 1 Overview of the Architectural Pattern**

### A. Data Ingestion from Multiple Data Sources

Organizations today receive data from multiple source systems, whether outside the organization or within an organization. For example, claims information from various adjudication systems for a health care company. In the diagram above, the sources are represented as source files, but this could be source data in any form.

### B. File Gateway

There needs to be a centralized File gateway server that handles all the files going out and coming into the organization. This is highly critical from a security perspective and also from a traceability perspective. There should be one system that caters to all incoming and outgoing files for an organization in order to mitigate security risk by opening only one server to the outside world past the firewall. File Gateway servers are responsible for routing incoming files to their appropriate internal destinations. Generally, when a new incoming file setup is being configured, the sender (vendor) will be provided with Gateway server details, location and credentials for the gateway server. Public and private key setup is done between the sending and the receiving systems. The configuration is set up in such a way that when a file arrives from a specific vendor, with a specific file name pattern and at a specific location on the Gateway server, this file is then routed based on a configuration to its corresponding destination folder on the organization's internal server. For example, a healthcare organization receives provider data from multiple vendors. When a specific provider file name pattern from a specific vendor is received at a specified location on the Gateway server, this file will be routed to internal Data Warehouse servers.

In the case of outgoing files, the team sending files outside the organization engages the File Gateway team. File Gateway team then works with the internal team as well as with the organization that is receiving a file. A dedicated landing location is created for the internal server to push the files on the Gateway server, and then the Gateway server pushes the files out to the external server using secure file transfer protocols.

A dedicated landing location is created for the internal server to push the files on the Gateway server, and then the Gateway server pushes the files out to the external server using secure file transfer protocols.

### C. Standard File Intake process

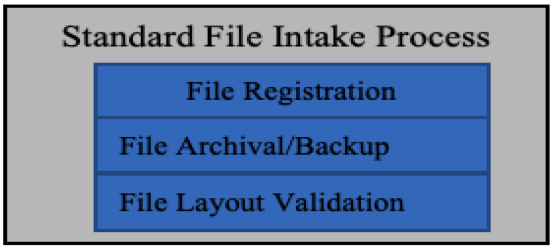

**Fig. 2 Standard File Intake Process**

When a Data Integration Team receives a file(s) from the gateway server, on the basis of file watchers, the Standard File Intake process would be invoked.

The standard File Intake process is a critical process that performs File Registration, File Archival and file layout verification for traceability and audit purposes. Often data received from outside clients are governed by retention policies of contractual agreements and are also required for auditing purposes.

Most of the data files transferred between organizations are accompanied by control files. These control files provide some important details related to data files such as the number of records in the data file, sum of amount fields that are critical from a financial as well as from data integrity perspective and date ranges for which this data belongs.

Some of the core capabilities of the Standard File Intake System are as follows

File Registration: File Registration is to register Incoming Files. As each file is landed on the Data Integration servers, the process takes these files to register them in Audit Balancing and Control Database where each file is catalogued with File Name, corresponding control file name, count of records in Datafile and count in the control file.

File Archival: The second part of the Standard File Intake process is to save the original file in the archival location for retention and recovery perspective. And a copy of the original input file is created and used for all Data Integration and Data Loading processes. The original file stays on the server in an archival location for a few months, and then it is archived to tape where it will stay for an extended period of time, governed by contractual agreements and organization retention policy for that data domain.

File Layout Verification: The file Layout verification process validates the layout of the received file against the expected layout for that file. The process validates each and each field in that record against the expected schema for that file. It checks if the integer field has only integer values in a data file, date fields have valid dates in the data file and also check if all fields in the data file are within the boundary limits for that fields data type as defined in the schema for that file. The process also validates the data file count against the control file, and if the data does not match as per expectations, the process aborts and provides details in the log files.

The standard file intake process would also validate the record counts in the data file against the count field in the control file to ensure a complete file data file was received and ensures there were no data lost during transmission.

### *D. Standardization (Standardize to Canonical Form)*

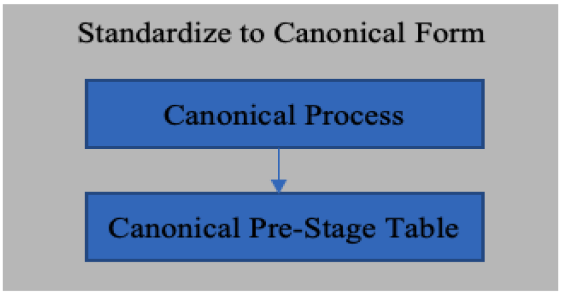

**Fig. 3 Standardize to Canonical Form**

As each source file might be coming in different systems with different file layouts. It is an ideal approach to standardize all different file layouts into one standard canonical layout so that all processes downstream to canonical layout are not impacted when a new source file is added to the ingestion layer.

Data Architects would have to do analysis to think about all the different attributes that could be received for that data domain from different sources, and that would be the Standard Canonical format so, that there are preferably no changes if absolutely required there should be minimal changes to the canonical model in future. Any changes to the canonical model will impact all the downstream to make those changes so. It is imperative to have a canonical model that is very versatile to handle future needs.

The Standardization process will format all files into a common standard layout. This layout would then be loaded into the Canonical Pre-Stage table. The common layout would have much more fields that not all fields would be present in all source files in for fields that are missing in files would be defaulted to null or a default value as per the use case.

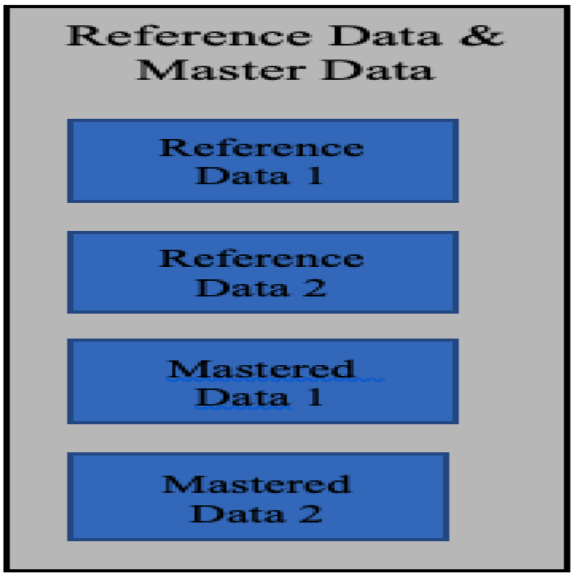

**Fig. 4 Reference and Mastered Data**

### *E. Reference and Mastered Data*

Reference Data refers to a standard set of codes, values that are used to categorize the data. This reference data could be standard set at the enterprise level or at the industry level or at country level or could be worldwide level. For example, in the medical industry, there are specific codes that refer to Diagnosis, Procedures, Diagnostic tests, Treatments and Equipment and supplies.

Some reference data like standard codes or abbreviations for states in the United States are
pretty standard and are consistent across the organization or across organizations.

But there are times where data is not consistent within the organization itself. For example, a health care organization has various departments that manage different aspects of membership like eligibility, pharmacy claims and medical claims as each department is getting data from different sources like vendors, point of sale systems (POS) where users of POS are using shorthand for names and address. There are many times where demographic information like Name, Address are spelt differently in different systems. This causes problems in linking data across data domains and could result in data integrity issues and incorrect data aggregations and reporting. This inconsistent reference data across data domains causes various major challenges to organizations. To resolve this issue, reference data is mastered at the enterprise level using Master Data Management tools. Then this data is used to link with transactional data to create more reliable integrated data increasing the trust of business users across the organization.

### *F. Data Integration and Reject Reprocessing*

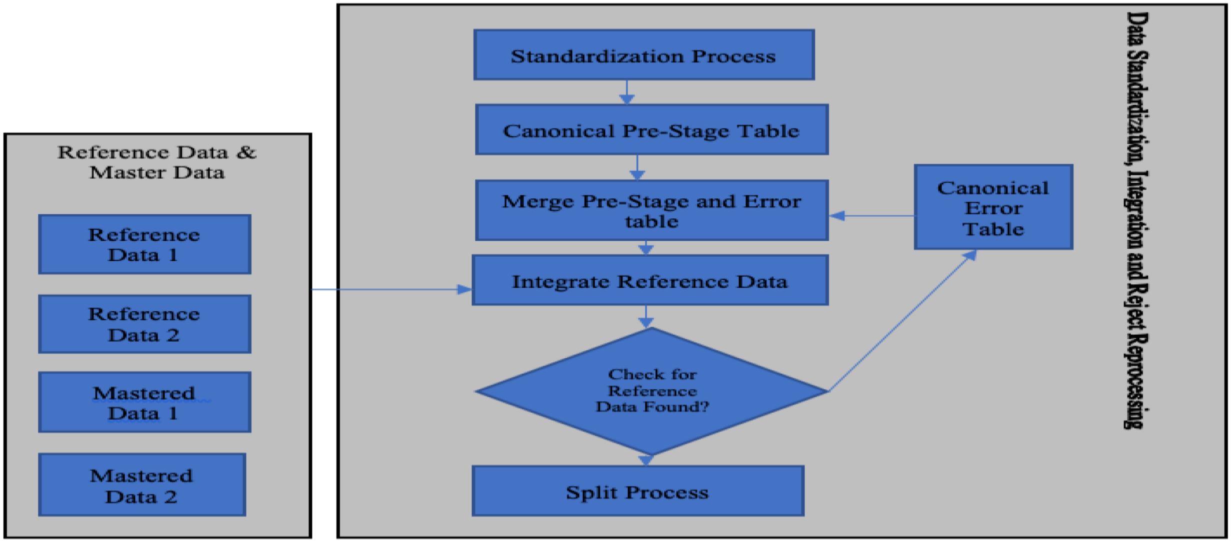
Reference Data & Master Data
Reference Data 1
Reference Data 2
Mastered Data 1
Mastered Data 2
Standardization Process
Canonical Pre-Stage Table
Merge Pre-Stage and Error table
Canonical Error Table
Integrate Reference Data
Check for Reference Data Found?
Split Process
Data Standardization, Integration and Reject Reprocessing

**Fig. 5 Data Integration and Reject Reprocessing**

Once all the source files are converted to a common standard canonical layout and looked up against reference and master data. Not all transaction data would tie with the reference/master data. Enterprises need the best quality of data to be available for their reporting and analytical needs. There are times where reference data attributes are good to have but not critically required, and there are times when specific data attributes are absolutely required for analytical and reporting needs. These data attributes might be so critical that without those attributes reporting needs might be not met, and it would be a good idea to stop this kind of data from even going to reporting layer. These are the cases where reject reprocessing comes into the picture.

After the standardized data is integrated with the reference data, the process should check if any of the required reference data is missing. If it is missing, then that data should be rejected and should be loaded into the Error Canonical Table. This rejected data would remain in the error table for the next processing cycle. When the next processing cycle starts at that time, the new data in PreStage Canonical table will be merged with the error table data and will be looked up against the reference/mastered data. Any data that finds a match in reference would be passed to the next step, whereas any data rejected will be constantly reprocessed until a match is found. There are instances where organizations would like to reprocess rejected data a specified number of times or a specified number of days before they purge the data out of the error table and send them to Data Stewards for further analysis.

### *G. File Splitting/Normalization Process:*

As part of this process, the canonical denormalized data is split into multiple files/data frames to be loaded into HDFS and then eventually to Hive stage tables as per the data model.

### *H. KMS &Stage/Confirm Tables*

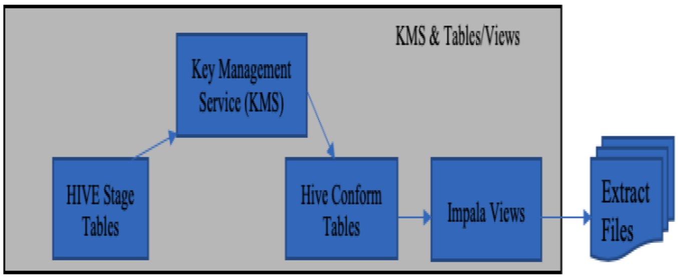

**Fig. 6 KMS and Tables/Views**

#### *a) Stage Tables*

Stage tables are typically truncated and load tables. There are instances where the stage tables are loaded in append mode due to specific scenarios.

#### *b) Key Management Service*

Unlike traditional RDMS, where data could be inserted/updated and deleted on the basis of a primary/business key, in Hive tables, data could only be inserted. In order to better manage data in a big data platform, Key Management Service could be used to generate the same surrogate key for the same business key combination. This capability is achieved by creating a key management table that stores a combination of business keys and corresponding surrogate keys generated. When transaction data is received, it is checked in the Key Management table; if the business key does not exist, then a record for that business key is inserted in the Key Management table, and a surrogate key is generated. Next time, when transactional data comes for the same business key, the business key will be looked up in the KMS table, and if a match is found, then the same KMS key will be utilized. If a match is not found, then a new surrogate key would be generated for that business key.

Conform Tables Loads with KMS Integration**:** Data from stage table while being loaded to conform tables needs to do a lookup with KMS table if the business key does not exist in the KMS table then a new key needs to be generated for that business key and if the business key already exists in the KMS table, in that case, the surrogate key available should be used while loading to conform table. As the updated transactional data with the same business key is received multiple times, then all records will be loaded to conform table with the same surrogate key. In this way, the surrogate key with the latest insert record timestamp will be the most current record.

### *J. Views*

Denodo Views: Denodo views are used on top of the conform tables to grant Role-Based Access Control (RBAC) to reporting layer.

#### *a) Impala views:*

In a traditional database, we generally use the temporal table concept or update in-place records on the basis of keys to have only the latest current active record.

In the case of Hive, Impala views could be leveraged to rank the records on the basis of surrogate and timestamp and then filter records with the latest timestamp for that surrogate key. This way, the impala views would be displaying the most current state of transactional data.

### *K. Extracts*

In an organization, there would be multiple downstream systems that would need data from Enterprise Datahub. In order to avoid various downstream systems querying Enterprise Datahub and impacting its performance, one of the most common patterns used today to expose data to the downstream system is the Publisher and Subscriber pattern (Pub-Sub) pattern. In this pattern, the source system publishes the data files along with control files at a specified location and time, and all the downstream systems consume these files for their processing. The advantage of using this pattern is the publisher guarantees an agreed-upon file layout for downstream systems (subscriber) irrespective of any database/layout changes in the Publisher system, and the publisher is immune to a number of downstream systems as all downstream systems will consume published extracts. The other advantages of this pattern are that the onus of guarantying the data integrity and extraction type (full or delta files) lies on the publisher side, and the subscriber does not need to understand the source (Datahub) system. The subscriber would just consume the extracts in a timely fashion. There are times where the publishing system could be down for maintenance, and subscribers would have to plan accordingly.

## V. CONCLUSION

By Implementing the above architectural pattern, an organization could ingest data from multiple source systems and can add new source systems as required, but all the impact will be absorbed by the standardization to canonical form so, all the downstream processes/systems are immune to source file layout changes or a number of sources added or removed. Following this pattern, an organization can reduce the cost of implementing new sources and also downstream immune systems to changes that come with modifying the layout of source files or adding/removing new source systems for a data domain. It also explains how the rejected data could be reprocessed and how easily data could be published, providing immunity to the source system by not overwhelming the source system with multiple downstream systems querying it.

## REFERENCES

[1] John Russell., Getting Started with Impala. Publisher O'Reilly Media, Inc., (2014) ISBN: 9781491905777

[2] Li, N., &Mahalik, N., A big data and cloud computing specification, standards and architecture: agricultural and food informatics. International Journal of Information and Communication Technology, 14(2) (2019) 159-174.

[3] James Le, An Introduction to Big Data: Data Integration: Published at Medium.com

[4] Ruojing Zhang, Marta Indulska, Shazia Sadiq., Discover Data Quality Problem spublished in Business and Information Systems Engineering journal in July (2019).

[5] Atif Mohammad, Hamid Mcheick, Emanuel Grant., Big Data Architecture Evolution: 2014 and Beyond., published in Association for Computing Machinery in September (2014).

[6] Mohammed M.A, Bartholomew E., Big Data Performance Analysis In Apache And Internet Information Services., published in International Journal of Computer Trends and Technology in November (2019).

[7] Eric Huey., Cloud Computing-Challenges and Benefits published in the International Journal of Computer Trends and Technology in September (2019).