International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677● Website: www.ijsat.org● Email: editor@ijsat.org

Data Quality using WAP Pattern for Data Pipelines

Arjun Reddy Lingala

arjunreddy.lingala@gmail.com

Abstract

One of the major problems in batch and real- time data pipelines in making sure data is accurate. Without data quality at every step of batch pipeline, it is tough to build reliable analytics and decision-making platforms. In this paper, we discuss an approach based on WAP (Write, Audit and Publish) pattern to improve data quality across each step of data pipeline processing. Many organizations use different batch processing approaches and WAP pattern facilitates all approaches systematic data transformation, validation and storage to mitigate quality issues such as inconsistencies, missing values, and outliers. WAP pattern eliminates downstream processing pipelines from consuming incorrect data and also eliminates re-processing of data to fix the incorrectness. The WAP pattern structures the data flow into three distinct phases: Write, where data is ingested and processed; Audit, where quality checks and validations are conducted; and Publish, where verified data is made available for downstream pipelines or frameworks. Usage of WAP pattern in real-world scenarios have enhanced traceability, accountability, and consistency through batch pipelines and also saves the compute by eliminating the need of re-running pipelines to address data incorrectness. This paper provides a modular approach to data quality that is adaptable to various pipeline structures, highlighting its practical relevance in data engineering workflows.

Keywords: Data Quality, Batch Pipelines, Data Anamolies, Audit- Ing, Fault Tolerance, Monitoring, Observability, Streaming Pipelines

I. INTRODUCTION

The rapid growth in data volume and variety has increased the challenges associated with maintaining data quality. En- suring quality of data flowing through pipelines is critical. Organizations rely on batch data processing pipelines to ingest, process, and deliver vast amounts of data for analytics use- cases. Data pipelines often face challenges such as schema changes, inconsistencies, incomplete records which compro- mise the reliability of downstream insights. The Write(W), Audit(A) and Publish(P) pattern is systematic and scalable approach to address these challenges in batch data process- ing for analytics. WAP pattern divides the batch processing pipelines into three phases: Write – data ingestion and initial processing occur in this phase. Raw, semi-processed or fully transformed data is stored in this stage, preserving its original state for traceability; Audit – data quality checks are performed in this phase which include schema validation, completeness checks, anomaly detection, and consistency verification. This phase ensures that data meets predefined quality thresholds before proceeding; Publish – once validated, data is formatted as needed and made available to downstream users. This phase emphasizes making only high quality data is accessible to ensure analytics and decision making. WAP enables structured

approach to monitoring and maintaining data quality and separation of each step in the pipeline has clear responsibilities and well-defined quality controls. By isolating the Audit phase, the WAP pattern enhances traceability, accountability, and maintainability of data pipelines.

II. BACKGROUND

Data pipelines are engineered to process data from var- ious sources such as streaming systems, database changes, transactional databases batch workloads, APIs, external third- party systems. These pipelines process and transform data into various formats that are suitable for analytics and decision making use cases. However, without proper data quality mod- ern data pipelines introduce challenges due to dynamic nature and complexity of data. Few examples of issues that may arise due to lack of data quality are:

1) *Schema Changes:* Data pipelines usually process data from multiple sources or applications which support end users and are tend to many changes based on the application needs. Frequent changes in the structure of the source data can disrupt downstream processes. Few examples include changes to column names, data types or new/removal of fields leading to data pipeline failures.
2) *Missing Data:* Upstream system failures or data extrac- tion errors can cause missing records or fields during ingestion and these errors can be propagated through the pipeline, reducing the accuracy of data pipelines.
3) *Data Anomalies:* Datasets sometimes have anomalous data and can cause outliers, duplicate records which can degrade the quality of insights. When anomalous data is processed further downstream it will cause incorrect insights and requires re-processing the data pipelines which will incur cost.

III. WRITE-AUDIT-PUBLISH (WAP) PATTERN

The Write-Audit-Publish (WAP) pattern provides a struc- tured approach to mitigate data quality challenges in modern batch processing pipelines by dividing the pipeline lifecycle into three distinct phases which emphasizes modularity and accountability.

A. *Write Phase*

In this phase, data is ingested or stored into staging layer in its raw form or as transformed data. This stage provides traceability and a fallback mechanism for reprocessing in the event of downstream or upstream issues. Without WAP pattern, data is write phase is usually used for downstream processing as soon as the data is landed.

B. *Audit Phase*

Audit phase is the main phase which validates whether the data stored in write phase can be consumed by downstream pipelines. This phase performs a series of quality checks and validations. Few example include:

- Schema validation to check if there is any schema changes from the previous version of the data and vali- date if there are any changes, they are compatible with previous version and doesn't break downstream pipelines.

- Checks to validate if there are any missing fields that are required for downstream
- In modern data pipelines using distributed systems, data in write phase is written as multiple partitions and few partitions might fail validation checks. Audit phase checks and validates data for each partition based on rules and thresholds predefined. Failed partitions are marked for remediation, marked as not to use for further investigation and making sure only high quality data is processed by next steps

C. *Publish Phase*

In publish phase, data that passes audit phase is published to a signal table and is made available for downstream use. This phase ensures only high quality data is available to downstream consumers.

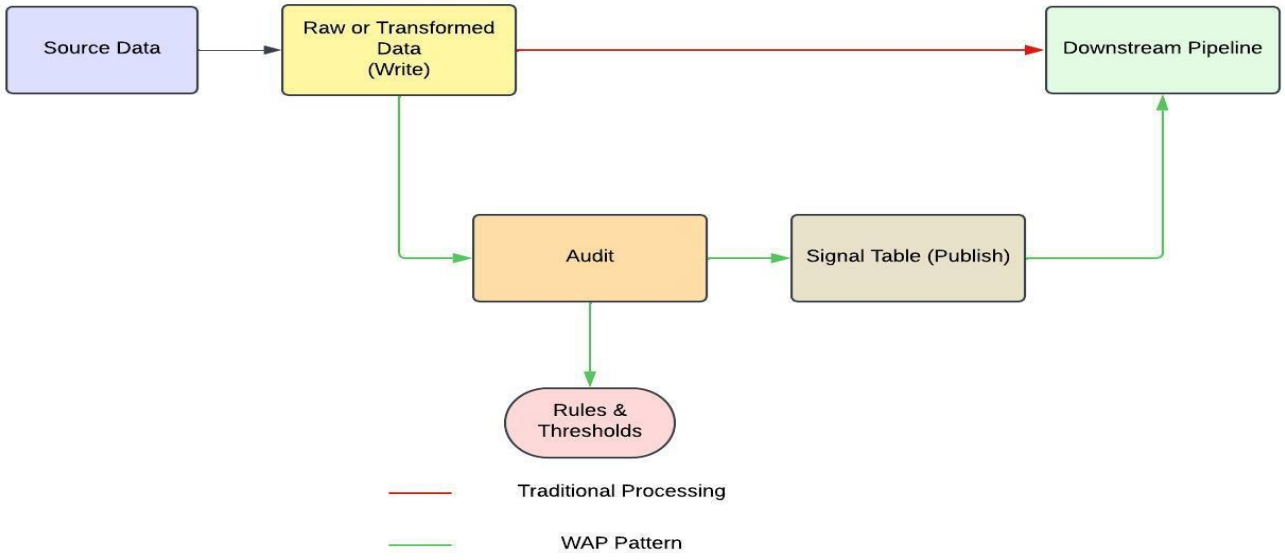

Fig. 1. Data Quality using WAP Pattern

IV. IMPLEMENTATION DETAILS

The implementation of the WAP pattern can leverage a combination of open-source and service based tools based on data processing pipeline requirements. The core principles of WAP pattern are *a) Ownership:* Each phase operates indepen- dently, isolating concerns and responsibilities. *b) Scalability:* Horizontal scaling in various phases of the pipeline enables the pipeline to handle growing data volumes. *c) Resilience:* Fault tolerance and data integrity are preserved throughout the pipeline life cycle *d) Cost:* Saving cost by eliminating the need to reprocess data pipelines in case of failures. WAP pattern includes building an additional table called signal table which captures the partitions of high quality data. Downstream consumers of data only consume from signal tables instead of

main or staging table. Signal tables doesn't consume lot of space or cost as it only saves metadata in the table. In this section, we deep dive into implementation of WAP pattern, covering architecture, workflows, and operational considera- tions.

A. *Write Phase*

Write phase is responsible for ingesting raw data from upstream systems and storing it in staging area in raw or transformed format. Key objectives of write phase are to ensure data is stored in its original format for traceability, preserve metadata for lineage and audit purposes, and pro- viding a reprocessing mechanism in case of errors. Write phase includes ingestion, storage, metadata collection, and fault tolerance.

1) *Ingestion:* Ingestion of data into data processing pipelines could be done in multiple ways. For streaming systems, real time ingestion is performed using tools like Apache Kafka [1], Google Pub/Sub, Amazon Kinesis [5]. For batch processing systems, ingestion pipelines run as sched- uled per hour or day using tools like Apache Spark [2] for data transformations, Apache Sqoop for extracting data from databases or custom ETL scripts which either pull data from databases or perform transformations.
2) *Storage:* Cloud based object stores like Amazon S3, Google cloud storage are used for storing huge amount of data. In addition to that, On-premise storage solutions like HDFS [3] can also be used for high volume data processing. Input data is usually organized by source, schema version, and partitioning is used by time of the arrival of the data.
3) *Metadata & Fault Tolerance:* Incoming data should be tagged with new attributes like time, source system informa- tion, schema version using metadata based tools like Apache Atlas. In some cases schema versioning can be handled by file formats [10] like Avro which send the schema with data and different versions are validated during reading or writing the data. All new fields added to file formats like Avro should be made optional as old versions might not have systems updated and may not these fields. Write phase pipelines should be idempotent and enable retries for failed jobs or re-processing the data.

B. *Audit Phase*

Audit phase is responsible for validating the data written in write phase with predefined rules and thresholds and make only high-quality data is available to downstream pipelines. Key objectives of audit phase include: identify and flag invalid, incomplete or inconsistent data, isolate problematic records for re-processing and generate reports for auditing purposes.

1) *Validation Rules:* Validation rules for each pipeline should be predefined based on the expectations of the input data and processing pipeline. Tools like Great Expectations [6] can be used to check schema compatibility for schema validation. Few validation rules include completeness checks to make sure mandatory fields are not missing, verify column values are within acceptable limits, and checking foreign key relationships.

2) *Anomaly Detection:* Implementing anomaly detection is one of the key areas of auditing phase. Due to upstream issues, there is very high chance of either data being dropped completely or more data being sent to downstream pipelines. This will be impact pipeline processing and misrepresentation of analytics. Using statistical methods like Z-scores for outlier detection or quantile based thresholds is best approach in this case.
3) *Auditing:* Traceability and lineage are key areas of WAP pattern. Logs should be generated for each validation step, pro- viding traceability to failures to upstream pipeline owners and help in avoiding similar errors in future. Structured logging formats like JSON, Avro and these logs will be integrated with monitoring systems like Splunk, Datadog which will be easier to analyze the logs.

C. *Publish Phase*

Publish phase publishes signals for validated and high qual- ity data to be consumed by downstream systems for processing in desired formats. Key objectives of publish phase include ensuring only high quality data is consumed by downstream consumers, maintain a signal table and ensure that information is populated to signal table properly. All downstream con- sumers pull data from the signal table partitions published by publish phase. Published partitions define high quality data and is safe to consume by downstream users.

V. METRICS

Capturing metrics for various steps of WAP pattern helps understand how many fault records are flowing through pipeline and helps in properly tracking them. Below are few areas of metrics that are best suitable to monitor and scale processing

1) *Fault Tolerance:* Using WAP pattern to store interme- diate results into staging tables enable recovery in case of failures. Distributed check-pointing can be used in streaming pipelines to recover from transient failures.
2) *Monitoring & Observability:* Tools like CloudWatch [4] help collect pipeline throughput, latency, and error metrics Tools like Grafana and Tableau can be used to visualize metrics and fault or anamolies rates. Real-time alerts can be integrated with tools like PagerDuty, Slack when validation of data or thresholds are breached.
3) *Batch & Stream pipelines:* For batch pipelines, ingestion is scheduled using cron jobs or orchestration tools like Apache Airflow, validation or Auditing can be applied using Spark jobs, and publishing results as signal tables can be done as data warehouse tables.

CONCLUSION

Ensuring data quality is critical to the success of data engineering projects, as it directly impacts analytics, decision- making initiatives. The Write-Audit-Publish (WAP) pattern provides a structured framework to tackle the complexities of maintaining data quality in modern data pipelines. By dividing the pipeline into distinct phases—Write, Audit, and

Publish—this approach ensures clear roles for ingestion, val- idation, and delivery, enhancing both pipeline robustness and maintainability. The WAP pattern emphasizes strict quality control during the Audit phase, where data undergoes com- prehensive validation and anomaly detection. This ensures that only accurate, consistent, and complete data is passed on to the Publish phase for downstream consumption. The separation of raw or transformed, validated, and processed data in distinct layers also enhances traceability, allowing for efficient debug- ging and compliance adherence. Adaptability is a key strength of the WAP pattern. Whether implemented in batch workflows or real-time streaming pipelines, it offers scalability, fault tolerance, and flexibility. Coupled with modern tools such as Apache Kafka [1], Great Expectations [6], and cloud-native storage systems, this pattern addresses challenges like data volume growth, schema changes, and the need for operational efficiency.

REFERENCES

[1] Kreps, J., Narkhede, N., Rao, J. (2011). ”Kafka: A Distributed Messag- ing System for Log Processing.” In Proceedings of the 6th International Workshop on Networking Meets Databases (NetDB), Athens, Greece, 2011, pp. 1-7.

[2] Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M. J., Shenker, S., Stoica, I. (2012). ”Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Com- puting.” In Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation (NSDI), San Jose, CA, 2012, pp. 1-14.

[3] Shvachko, K., Kuang, H., Radia, S., Chansler, R. (2010). ”The Hadoop Distributed File System.” In 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), Incline Village, NV, USA, 2010, pp. 1-10. doi: 10.1109/MSST.2010.5496972.

[4] Amazon Web Services, Inc. ”Amazon CloudWatch – Observe and monitor resources and applications on AWS, on premises, and on other clouds.” Available: https://aws.amazon.com/cloudwatch/.

[5] Amazon Web Services, Inc. ”Amazon Kinesis – Collect, pro- cess, and analyze real-time video and data streams.” Available: https://aws.amazon.com/kinesis/.

[6] Great Expecations - A Data Quality tool. Available: https://greatexpectations.io/.

[7] Dbt Labs. ”Dbt - Power everything in your business with high-quality, trusted, and reliable data.” Available: https://www.getdbt.com/

[8] S. Challita, F. Zalila, C. Gourdin and P. Merle, ”A Precise Model for Google Cloud Platform,” 2018 IEEE International Conference on Cloud Engineering (IC2E), Orlando, FL, USA, 2018, pp. 177-183, doi: 10.1109/IC2E.2018.00041.

[9] Balamurugan Balusamy; Nandhini Abirami R; Seifedine Kadry; Amir H. Gandomi, ”Driving Big Data with Hadoop Tools and Technologies,” in Big Data: Concepts, Technology, and Architecture , Wiley, 2021, pp.111-160, doi: 10.1002/9781119701859.ch5.

[10] A. Gohil, A. Shroff, A. Garg and S. Kumar, ”A Compendious Research on Big Data File Formats,” 2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2022, pp. 905-913, doi: 10.1109/ICICCS53718.2022.9788141.