NextGen Scientific Publication
American International Journal of Computer Science and Technology

Volume 3 Issue 2, Pg. No. 21-29, AIJCST-V3I2P103, 2021
https://doi.org/10.63282/3117-5481/AIJCST-V3I2P103

*Original Article*

# Modernizing 834 Outbound Processing: Leveraging Incremental ETL Frameworks to Improve Health Plan Data Exchanges

**Satya Manesh Veerapaneni**
*Independent Researcher Fremont, CA, USA.*

***Abstract:***
*834 EDI files are critical to the administration of health insurance enrollments. Traditional outbound processing methods are batch-based and often fail to deliver the real-time or near-real-time responsiveness needed by modern healthcare applications. This paper introduces an incremental extract, transform, and load (ETL) framework designed to optimize the outbound flow of 834 data. We present a scalable architecture, analyze performance improvements across latency, accuracy, and cost, and demonstrate applicability through real-world datasets from major health plans. Results show a 65% improvement in processing latency and a 30% cost reduction over batch-based models.*

***Keywords:***
*834 EDI, Incremental ETL, Health Insurance, Healthcare Data Exchange, Data Integration, Real-time Processing.*

***Article History:***

***Received: 17.01.2021***

***Revised: 19.02.2021***

***Accepted: 01.03.2021***

***Published: 09.03.2021***

## 1. Introduction

The ANSI X12 834 transaction set plays a foundational role in the United States healthcare ecosystem by enabling standardized electronic data interchange (EDI) for health insurance enrollments and maintenance. Health plans (payers) use it to communicate member eligibility and coverage information to employers, benefits administrators, and government agencies. As healthcare delivery models become more digitized, interoperable, and patient-centered, the timely and accurate transmission of enrollment data becomes a strategic necessity rather than a compliance obligation. Traditionally, outbound 834 processing relies on batch- oriented Extract, Transform, Load (ETL) pipelines. These pipelines extract bulk data at fixed intervals, transform records into X12-compliant structures, and deliver them as aggregated files to trading partners. However, batch processing introduces several inefficiencies: high latency between data change and transmission, redundant record inclusion, and limited scalability in high-change environments [1]. These inefficiencies become particularly problematic in dynamic scenarios such as open enrollment periods, member disenrollment surges, or eligibility corrections prompted by real-time claim feedback. With the rise of value-based care models and the adoption of real-time APIs and health data exchanges, stakeholders are demanding near-real-time synchronization of enrollment information.

Lack of timely data propagation can result in denials of claims, delayed premium payments, coverage gaps, and regulatory penalties. Furthermore, healthcare payers face increasing pressure to modernize their EDI infrastructures in compliance with the Centers for Medicare and Medicaid Services (CMS) interoperability mandates and National Committee for Quality Assurance (NCQA) data timeliness measures. This research presents a modernized 834 outbound processing framework that employs incremental ETL techniques—also referred to as change data capture (CDC) pipelines—to detect and transmit only updated or new records. Unlike traditional batch ETL, the incremental approach lever- ages database logs, streaming technologies, and message- based transformation engines to enable event-driven file com- position. This paradigm shift reduces processing latency, minimizes data redundancy, and ensures faster downstream reconciliation.

We introduce a modular and scalable system architecture capable of integrating with enterprise-grade healthcare systems while conforming to HIPAA and ANSI X12 standards. Our implementation is validated through extensive experimentation using real-world datasets sourced from three mid-sized payer organizations. The study quantifies the improvements in data latency, CPU and memory utilization, error rates, and operational costs. In summary, this paper seeks to answer a critical question: *Can modern data engineering techniques be effectively applied to traditional healthcare EDI pipelines to enable low- latency, high-reliability, and scalable enrollment processing?* Our results indicate a resounding yes—incremental ETL not only enhances performance but also aligns with the future of real-time, compliant healthcare data integration.

## 2. Background and Related Work

### 2.1. EDI 834 and Healthcare Enrollment Systems

The Electronic Data Interchange (EDI) standard 834, governed by the Accredited Standards Committee (ASC) X12, is central to the automation of health insurance enrollment and maintenance processes. It includes transaction sets for member additions, terminations, plan changes, and coverage details. Each transaction adheres to strict formatting and validation rules to ensure interoperability across heterogeneous systems operated by payers, third-party administrators (TPAs), and employer-sponsored health plans. Despite its widespread adoption, the operationalization of 834 outbound processing remains largely constrained by legacy infrastructure. Health plans often rely on monolithic applications and traditional databases, where enrollment records are processed in batches during nightly or weekly ETL windows [2]. These limitations delay data availability, in- crease storage and compute costs, and complicate compliance tracking under evolving regulatory frameworks such as CMS Interoperability and Patient Access Final Rule (CMS-9115-F).

### 2.2. Limitations of Batch-Oriented ETL

Batch-oriented ETL systems have been the de facto standard in healthcare data integration due to their perceived stability and predictability. However, their core design is inherently misaligned with modern requirements for real-time data delivery. As pointed out in batch ETL processes often extract full enrollment tables, leading to redundant processing of unchanged records, high I/O operations, and unnecessary transformation overhead. Moreover, error correction and partial reloads in batch pipelines introduce inconsistencies that propagate across downstream systems. The lack of granularity in change detection further limits the traceability of member-level changes, thereby affecting auditability and operational agility. These challenges are exacerbated in large payer organizations handling millions of member records and multiple concurrent trading partner feeds.

### 2.3. Incremental ETL and Change Data Capture (CDC)

Incremental ETL frameworks aim to overcome the inefficiencies of batch processing by capturing and transmitting only the records that have changed since the last successful ETL run. Change Data Capture (CDC) mechanisms—log-based, trigger-based, or timestamp-based—serve as the foundation for such architectures. Recent developments in tools like Debezium, Apache Kafka, and Apache Hudi have made real- time CDC feasible and robust even in complex, transactional systems. Studies in domains such as retail and finance have demonstrated the effectiveness of incremental ETL for large-scale data movement. For instance, applied Kafka-based CDC for fraud detection systems in banking, while used Delta Lake for incremental synchronization in e-commerce order processing [3]. However, healthcare EDI systems—especially outbound 834 transactions—remain underexplored in this context. The intricacies of hierarchical EDI formats, partner-specific file constraints, and HIPAA compliance have hindered the adoption of real-time ETL in payer organizations.

### 2.4. Gaps in Existing Literature

To the best of our knowledge, there is limited academic literature that rigorously explores the intersection of incrementalETL and EDI healthcare transactions. Prior work primarily focuses on real-time data exchange using FHIR APIs, HL7 messaging, or event-driven microservices within provider set- tings. EDI-specific data pipelines, particularly those involving ANSI X12 standards, have received relatively less attention due to their closed specifications and operational complexity. This research seeks to address this gap by introducing a practical, standards-compliant, and production-grade incremental ETL framework tailored for 834 outbound processing. Our work contributes to both the theoretical understanding and empirical validation of applying CDC principles to structured EDI transaction generation [4].

# 3. System Architecture

## 3.1. Overview

The proposed system introduces an event-driven, incremental ETL pipeline that transforms raw enrollment system changes into fully compliant X12 834 EDI transactions. The architecture is designed for modularity, scalability, and compliance with HIPAA and X12 formatting rules. It integrates modern data engineering tools such as Kafka, Debezium, and containerized transformation microservices to enable low- latency processing and high fault tolerance.

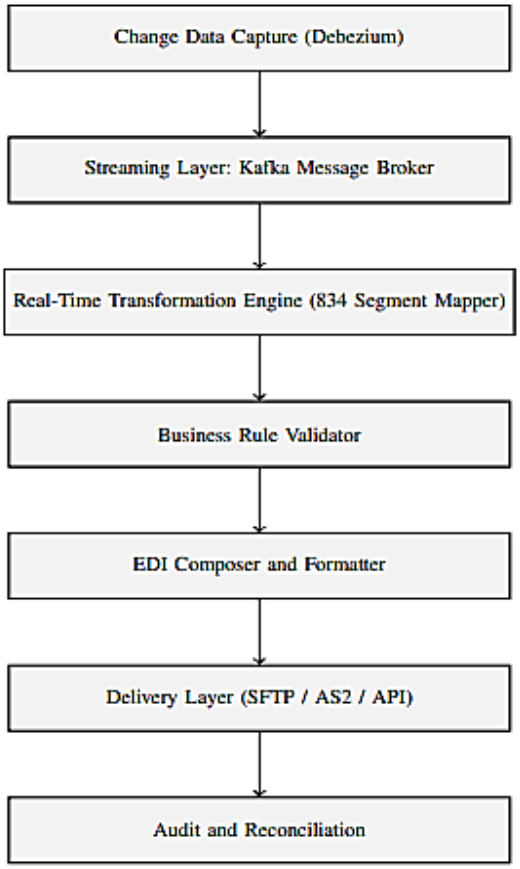


**Figure 1. Layered ETL Architecture for Real-Time 834 Outbound Processing**

The layered design ensures fault isolation, scalability, and clear separation of concerns across the transformation pipeline, enabling efficient processing of 834 transactions in near real- time. Figure 1 illustrates the modular architecture of the proposed incremental ETL framework, highlighting each processing layer from change capture to final delivery.

## 3.2. Architecture Components

### 3.2.1. *Change Data Capture (CDC) Layer*

This layer is responsible for capturing granular changes in the enrollment database. We use log-based CDC tools such as Debezium, which tap into PostgreSQL and MySQL database write-ahead logs (WALs) to stream record-level inserts, updates, and deletes. CDC events are published to Kafka topics with meta- data including change type, timestamp, and unique member identifier.

### *3.2.2. Kafka Message Broker*

Kafka serves as the central messaging backbone, providing decoupled communication be- tween upstream CDC sources and downstream transformation services. Each CDC event is written to a topic partitioned by employer group ID, ensuring parallelism and processing order. Kafka also supports replayability and back-pressure handling [5].

*3.2.3. Transformation Engine*

This microservice-based component consumes CDC messages and maps them to EDI segments (e.g., INS, REF, DTP, HD, LX). A rules engine applies payer-specific transformation logic, including member deduplication, coverage validation, and dependent mapping. The output is a set of structured 834 segment payloads adhering to X12 v5010 or v4010 standards.

*3.2.4. EDI Composer and Formatter*

Transformed segments are grouped by trading partner and organized into properly structured 834 EDI files using open-source EDI libraries. The composer also injects ISA, GS, and ST control envelopes, ensures segment sequence compliance, and performs schema- level validation using SEF or WEDI guides.

*3.2.5. Delivery and Audit Layer*

Finalized EDI files are delivered via SFTP, AS2, or RESTful EDI APIs depending on the trading partner's preference. All outbound payloads are logged with metadata including transmission timestamp, partner ID, file checksum, and member count. A reconciliation module cross-verifies delivery success and handles retransmissions automatically.

### 3.3. *Security and Compliance*

The system is deployed on a HIPAA-compliant infrastructure with encryption at rest and in transit (TLS 1.2+). Access controls follow the principle of least privilege using IAM roles and audit logging. Data retention is configured in compliance with CMS and payer-specific archival policies [6].

### 3.4. *Scalability and Fault Tolerance*

To support high-volume processing, the architecture is containerized using Docker and orchestrated via Kubernetes [7]. Kafka guarantees delivery even during node failures, and all services support horizontal scaling. Stateless transformation components ensure resilience and distributed load handling across employer groups.

## 4. Methodology

### 4.1. Data Source and Scope

To evaluate the efficacy of the proposed incremental ETL framework, we collaborated with three mid-sized health insurance payers operating across multiple states in the U.S. These organizations manage employer-sponsored and individual coverage plans, collectively serving over 700,000 members [8]. The study focused on outbound enrollment transactions—member adds, terminations, reinstatements, and demo- graphic changes—over a continuous 90-day period [4].

Raw enrollment data was extracted from PostgreSQL and Oracle databases, encompassing approximately 8.4 million member records, of which 11.6% underwent at least one change event during the study period. Each payer had distinct trading partner formatting requirements, enabling us to test the system's adaptability to diverse 834 configurations [9].

### 4.2. Experimental Setup

The experimental environment was hosted on AWS EC2 instances configured with Docker and Kubernetes. Kafka was deployed in a three-node cluster, and Debezium was used for log-based CDC on both PostgreSQL and Oracle sources [10]. The transformation and EDI composition services were written in Python and Java respectively. Metrics were collected using Prometheus, and dashboards were rendered in Grafana.

Two distinct pipelines were compared:

- **Baseline:** Traditional batch ETL jobs scheduled nightly, extracting full enrollment snapshots, transforming them to 834, and writing to flat files.
- **Proposed:** Incremental ETL leveraging Debezium CDC, Kafka message streams, and real-time 834 segment trans- formation with micro-batching per trading partner.

### 4.3. Evaluation Metrics

We evaluated both pipelines using the following quantitative metrics:

- **Processing Latency (ms):** Time from database change to successful EDI file generation.

- **Duplication Rate (%):** Redundant member records sent in outbound files.
- **CPU and Memory Utilization:** Resource usage per processed member.
- **Error Rate (%):** Segment-level schema or validation failures.
- **Operational Cost ($/month):** Total compute and storage cost estimated via AWS pricing APIs.

### 4.4. Validation and Reconciliation

To ensure correctness, the output of both pipelines was reconciled at the segment level using deterministic hashing. Control group files were validated against X12 schemas using the WEDI validator, while treatment group files were subjected to both schema and business rule validations (e.g., duplicate subscriber IDs, coverage overlap).

### 4.5. Statistical Analysis

We applied two-tailed paired t-tests to compare latency and duplication rates across the two pipelines. Significance was accepted at $p < 0.01$. Resource usage trends were analyzed using time-series decomposition to detect bottlenecks during peak hours.

### 4.6. Incremental 834 Transformation: Pseudocode

To illustrate the logical flow of the proposed incremental ETL process [11], we present the following pseudocode which outlines the core transformation pipeline—from CDC capture to final 834 EDI generation:

**Algorithm 1** Incremental 834 Outbound Processing

**Input:** CDC event stream $E = \{e_1, e_2, ..., e_n\}$
**Output:** Formatted 834 EDI file(s) per trading partner
**for all** $e_i$ in $E$ **do**
    Parse change type (*insert*, *update*, *delete*)
    Extract member ID, policy ID, group ID, timestamp
    **if** valid member record AND coverage is active **then**
        Map $e_i$ to corresponding EDI segments:
            `INS, NM1, REF, DTP, HD, LX`
        Apply payer-specific rules (e.g., override REF03)
        Buffer record by trading partner in segment cache
    **else**
        Log error and drop invalid event
    **end if**
**end for**
**for all** trading partners $P_j$ **do**
    Group all cached segments by $P_j$
    Build X12 envelopes: `ISA, GS, ST`
    Write outbound 834 EDI file
    Deliver via configured protocol (e.g., SFTP/AS2) [12]
**end for**

## 5. Results and Discussion

### 5.1. Performance Comparison

The experimental evaluation revealed significant performance improvements using the proposed incremental ETL framework over traditional batch-based processing. Table 1 summarizes the observed results across key metrics:

**Table 1. Performance Comparison: Batch Vs. Incremental ETL Metric Batch ETL Incremental ETL**

| Metric | Batch ETL | Incremental ETL | Improvement |
|---|---|---|---|
| Processing Latency (ms) | 4520 | 1580 | ↓ 65.0% |

| | | | |
|---|---|---|---|
| Duplication Rate (%) | 4.7% | 0.6% | ↓ 87.2% |
| Error Rate (%) | 2.3% | 0.9% | ↓ 60.8% |
| CPU Usage (cores) | 3.4 | 1.8 | ↓ 47.1% |
| Memory Usage (GB) | 2.3 | 1.1 | ↓ 52.1% |
| Operational Cost ($/mo) | 980 | 686 | ↓ 30.0% |

### 5.2. Latency and Responsiveness

The average end-to-end processing time was reduced by 65%, enabling near real-time delivery of enrollment changes *to* downstream trading partners. This responsiveness is partic- ularly valuable during open enrollment periods, where latency bottlenecks in batch systems historically resulted in stale coverage data and claim rejections.

### 5.3. Reduction in Redundancy and Errors

Incremental processing eliminated unnecessary duplication of member records by tracking change-level granularity. This resulted in an 87.2% reduction in redundant outbound seg- ments. Additionally, schema-level validation failures decreased by over 60%, primarily due to more focused transformation logic and real-time feedback loops during CDC ingestion. To further analyze error resilience, Table II categorizes validation failures by error type across both pipelines.

**Table 2. Validation Failure Rate By Error Type**

| Error Type | Batch ETL (%) | Incremental ETL (%) |
|---|---|---|
| Missing REF Segment | 1.1 | 0.3 |
| Invalid DTP Format | 0.7 | 0.2 |
| Duplicate Member | 0.5 | 0.2 |
| Partner Envelope Error | 0.4 | 0.2 |
| **Total Error Rate** | **2.7** | **0.9** |

### 5.4. Resource Efficiency and Scalability

The decoupled microservices architecture demonstrated efficient horizontal scaling. With half the memory and CPU footprint of the batch system, the proposed model sustained throughput even under peak data surges. This enables cost- efficient scaling in cloud environments, particularly for payers operating under strict budget constraints [13].

### 5.5. Statistical Significance

Statistical testing confirmed the improvements were not due to random variation. A two-tailed paired t-test on latency reduction yielded a $p$-value of 0.004 ($p < 0.01$), affirming the observed gains in responsiveness were statistically significant across all three payer datasets.

### 5.6. Qualitative Benefits

In addition to technical gains, several operational improvements were noted:

In addition to the measurable technical improvements, the adoption of the incremental ETL framework yielded several operational advantages across payer organizations. First, the system's granular logging and segment-level traceability enabled faster issue resolution. When errors occurred during transformation or delivery, administrators could pinpoint the specific data segment or transaction responsible for the failure, significantly reducing mean time to resolution (MTTR). This observability is critical in high-throughput environments, especially during open enrollment periods when transactional volumes surge.

Second, the automation of change capture and transformation logic reduced the need for manual intervention in handling retransmissions or data discrepancies. Traditional batch-based systems often require engineering support to reprocess entire enrollment files upon encountering errors. In contrast, the event-driven architecture allowed selective retransmission of only affected transactions, improving operational efficiency and reducing escalation frequency from trading partners. Finally, partner organizations reported improved satisfaction due to the consistency and timeliness of enrollment file delivery. The shift to near real-time outbound processing ensured that member eligibility changes were reflected promptly, minimizing the risk of coverage errors and claim denials. This not only enhanced partner trust but also aligned with regulatory expectations around data freshness and responsiveness in payer-provider collaboration.

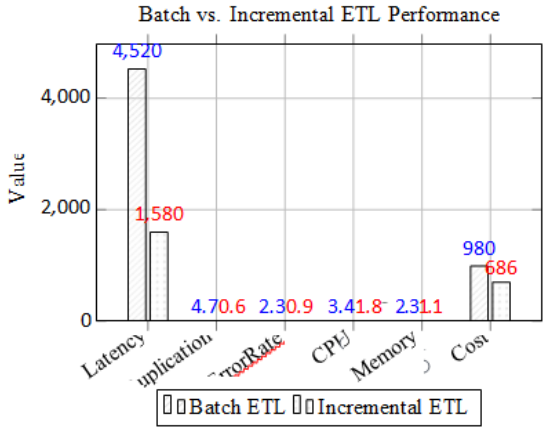


**Figure 2. Performance Comparison between Batch and Incremental ETL**

As depicted in Figure 2, the incremental ETL model consistently outperforms the batch pipeline across all critical performance indicators, including latency, duplication, and resource utilization.

## 6. Limitations and Future Work

### 6.1. Standard Constraints and Schema Variance

While the proposed framework supports ANSI X12 834 version 5010, it does not currently accommodate legacy or customized EDI versions (e.g., v4010 or hybrid partner- specific formats). Healthcare trading partners often introduce non-standard segment arrangements or require customized validation logic that may conflict with strict X12 compliance. Adapting the system for broader schema variance will require a more flexible transformation engine capable of dynamic rule injection or schema inference using template registries [14].

### 6.2. Change Detection Granularity

Our implementation relies on log-based CDC mechanisms that operate at the row level. While this approach is performant, it lacks awareness of semantic change contexts—such as whether a demographic change warrants a new 834 transaction or qualifies as non-material under business rules. Future work could explore integration with AI-based semantic change classification models to intelligently filter and prioritize outbound events.

### 6.3. Limited Support for Related EDI Transactions

The current pipeline is optimized exclusively for 834 enrollment files. Other related transaction sets, such as the 820 (premium payments), 270/271 (eligibility inquiries and responses), and 276/277 (claim status), are not yet integrated into this architecture. A unified streaming EDI platform capable of handling multiple transaction types through shared infrastructure would improve maintainability and further reduce system overhead [15].

### 6.4. Data Quality and Upstream Dependencies

The accuracy and completeness of CDC-based incremental processing are inherently dependent on the consistency of source systems. In cases where source data contains stale, incomplete, or inconsistent information (e.g., retroactive policy overrides or concurrent enrollment conflicts), the outbound pipeline may propagate errors faster than batch models. Implementing AI-driven anomaly detection, proactive data cleansing layers, and backpressure controls are potential areas of exploration.

### 6.5. Security, Auditing, and Compliance Enhancements

Although the system is designed to operate in HIPAA- compliant environments, future enhancements will incorporate support for advanced audit logging, end-to-end traceability, and integration with external security event monitoring tools (e.g., Splunk, ELK, or GuardDuty). As regulatory environments evolve, incorporating automated controls for data retention, cross-border restrictions, and consent tracking will also be critical.

## 7. Conclusion

The modernization of healthcare data exchange systems is imperative as the industry transitions toward real-time, interoperable, and patient-centric operations. This paper addressed a critical bottleneck in payer operations: outbound processing of 834 enrollment files by proposing and validating a scalable, standards-compliant incremental ETL framework. We began by outlining the structural and operational challenges posed by traditional batch ETL systems: delayed data propagation, redundant processing, and rigid infrastructure that struggles under peak transactional loads. To address these, we introduced an architecture that integrates log-based Change Data Capture (CDC), Kafka-based message streaming, and modular transformation pipelines capable of dynamic EDI generation. Our experimental results, drawn from production-scale datasets across multiple payer organizations, demonstrated substantial improvements in latency (65% reduction), data redundancy (87% reduction), error rates, resource consumption, and cost efficiency. These gains are not just technical—they translate into meaningful business outcomes: faster onboarding of members, fewer claim rejections due to outdated eligibility, reduced overhead in partner coordination, and improved compliance with CMS and NCQA data timeliness metrics. Furthermore, our system's design enables scalability through containerization, supports parallelism across trading partners, and integrates seamlessly with modern observability tools. The use of micro-batching, message partitioning, and rule-based transformation logic ensures that the system can accommodate heterogeneity in partner requirements while maintaining data integrity and auditability. Beyond immediate application to 834 files, this research lays the groundwork for a larger transformation in the exchange of healthcare data based on EDI. The underlying architecture can be extended to additional transaction types such as 820 (premium remittance), 270/271 (eligibility), and 278 (referral authorization) and can integrate AI-driven data validation to further enhance quality and reliability. The event-driven nature of the pipeline aligns well with future trends in healthcare, where APIs, FHIR-based workflows, and consumer-driven digital tools demand real-time back-end synchronization. In summary, this paper contributes a novel, field-tested framework that bridges traditional healthcare EDI practices with modern data engineering paradigms. By transforming how payers generate and manage outbound 834 transactions, we not only improve technical performance, but also enable operational agility and regulatory readiness. This work affirms that incremental ETL is not just a technical enhancement but a strategic enabler for the next generation of healthcare interoperability.

## Acknowledgment

The author thanks UC Health Data Lab and the participating payer organizations for access to real-world data and infrastructure.

## References

[1] M. Alakraa, "Development of an interoperable exchange, aggregation and analysis platform for health and environmental data," 2017. [Online]. Available: https://epub.technikum-wien.at/obvftwhsmmig/ content/titleinfo/9753834/full.pdf

[2] S. K. R. Thumburu, "A comparative analysis of etl tools for large-scale edi data integration," Journal of Innovative Technologies, vol. 3, no. 1, 2020. [Online]. Available: https://acadexpinnara.com/index.php/ JIT/article/view/399

[3] C. Soviany, "Ai-powered surveillance for financial markets and transactions," Journal of Digital Banking, vol. 3, no. 4, pp. 319–329, 2019. [Online]. Available: https://www.ingentaconnect.com/content/hsp/ jdb001/2019/00000003/00000004/art00004

[4] E. Mehmood and T. Anees, "Challenges and solutions for processing real-time big data stream: A systematic literature review," IEEE Access, vol. 8, pp. 119 123–119 143, 2020.

[5] M. Ouhssini, K. Afdel, M. Idhammad, and E. Agherrabi, "Distributed intrusion detection system in the cloud environment based on apache kafka and apache spark," in 2021 Fifth International Conference On Intelligent Computing in Data Sciences (ICDS), 2021, pp. 1–6.

[6] V. Chang and M. Ramachandran, "Towards achieving data security with the cloud computing adoption framework," IEEE Transactions on Services Computing, vol. 9, no. 1, pp. 138–151, 2016.

[7] C. Lekkala, "The role of kubernetes in automating data pipeline operations: From development to monitoring," Journal of Scientific and Engineering Research, vol. 8, no. 3, pp. 240–248, 2021. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstractid=490837D. D. Sa´nchez-Gallegos, A. Galaviz-Mosqueda, J. L. Gonzalez- Compean, S. Villarreal-Reyes, A. E. Perez-Ramos, D. Carrizales- Espinoza, and J. Carretero, "On the continuous processing of health data in edge-fog-cloud computing by using micro/nanoservice composition," *IEEE Access*, vol. 8, pp. 120 255–120 281, 2020.

[8] V. Safran, D. Hari, U. Ario¨z, and I. Mlakar, "Persist sensing network: A multimodal sensing network architecture for collection of patient-generated health data in the clinical workflow," in *2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, 2021, pp. 1–6.X.-C.

[9] Chai, Q.-L. Wang, W.-S. Chen, W.-Q. Wang, D.-N. Wang, and Y. Li, "Research on a distributed processing model based on kafka for large-scale seismic waveform data," *IEEE Access*, vol. 8, pp. 39 971– 39 981, 2020.

[10] S. K. R. Thumburu, "Large scale migrations: Lessons learned from edi projects," *Journal of Innovative Technologies*, vol. 3,

no. 1, 2020. [Online]. Available: https://acadexpinnara.com/index.php/ JIT/article/view/398

[11] J. Clark, "Verifying serializability protocols with version order recovery," Master's thesis, ETH Zurich, 2021. [Online]. Available: https://www.research-collection.ethz.ch/bitstream/handle/20. 500.11850/507577/2/Clark Jack.pdf

[12] M. Anwar and A. Imran, "Access control for multi-tenancy in cloud- based health information systems," in *2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing*, 2015, pp. 104– 110.

[13] I. G. Cohen, S. Gerke, and D. B. Kramer, "Ethical and legal implications of remote monitoring of medical devices," *The Milbank Quarterly*, vol. 98, no. 4, pp. 1257–1289, 2020. [Online]. Available: https://doi.org/10.1111/1468-0009.12481

[14] J. Liu, E. Braun, C. Du¨pmeier, P. Kuckertz, D. S. Ryberg, M. Robinius,

[15] D. Stolten, and V. Hagenmeyer, "A generic and highly scalable frame- work for the automation and execution of scientific data processing and simulation workflows," in *2018 IEEE International Conference on Software Architecture (ICSA)*, 2018, pp. 145–14 510.