www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 10, October 2025, pp : 303-305

e-ISSN : 2583-1062
Impact Factor : 7.001

IMPLEMENTING CHANGE DATA CAPTURE WITH AZURE DATA FACTORY AND DATABRICKS

Dr. Dinesh D. Puri[1], Mr. Bhairav D. Attarde[2]

[1]Head Of Department, Department Of Computer Applications, SSBT COET, Jalgaon Maharashtra, India.

[2]Research Scholar, Department Of Computer Applications, SSBT COET, Jalgaon Maharashtra, India.

DOI: https://www.doi.org/10.58257/IJPREMS44114

ABSTRACT

Change Data Capture (CDC), a contemporary data engineering technique, effectively tracks changes made to Online Transaction Processing (OLTP) systems and sends them to analytical platforms downstream.

Instead of reloading entire datasets, which can be resource-intensive and slow in standard ETL procedures, CDC focuses on recording only the inserts, updates, and deletions. This leads to better performance and almost instantaneous analytics.

In this survey, we look at how to implement CDC using SQL Server, Azure Data Factory (ADF), and Databricks. We will discuss current methods such as log-based CDC in SQL Server, orchestration patterns using ADF, and the advanced processing capabilities offered by Databricks Change Data Feed (CDF) and Delta Lake.

This study discusses the rationale behind this approach, the techniques employed, possible roadblocks, and best practices for creating scalable CDC pipelines.

1. INTRODUCTION

These days, organizations need quick analytics to make informed decisions. Despite being widely used, batch ETL tasks can be inefficient, especially as datasets get larger.

Reloading entire tables every few hours can be costly and often unnecessary when only a few rows may change. The CDC was established to get around these limitations, but it only keeps track of minor data changes.

Transaction logs are read by SQL Server's built-in CDC feature, which then uses system tables to show the modifications [1]. In contrast, the cloud-based ETL and orchestration tool Azure Data Factory (ADF) effectively extracts and transfers CDC-enabled data by combining connectors and runtimes [4]. Based on Apache Spark, Databricks enhances CDC pipelines by integrating incremental data into Delta Lake tables, ensuring high speed, transactional guarantees, and support for Change Data Feed (CDF) [2][3].

In order to develop an automated, scalable, and efficient CDC pipeline, this study investigates the integration of SQL Server CDC with Databricks Delta Lake and ADF orchestration. As we proceed, we'll focus on pertinent research, technical operations, and real-world challenges.

2. LITERATURE SURVEY

2.1 Change Data Capture Techniques

There are several approaches to implementing Change Data Capture (CDC). The log-based CDC in SQL Server is one of the most widely used techniques because it allows for reliable and low-latency data change recording without requiring modifications to the source schema [1]. Other options include trigger-based and timestamp-based CDC. Trigger-based methods [5] may result in extra overhead and longer transaction times, even though they depend on database triggers to record modifications. On the simpler side, timestamp-based methods may miss deletes and require careful timestamp management [6]. The most widely used method for enterprise-scale workloads is log-based.

2.2 SQL Server CDC

SQL Server CDC uses the transaction log to record changes in system tables. It offers metadata about the changes, including the type of operation, and supports functions for querying changes between log sequence numbers (LSNs) [1]. This system has extensive documentation in Microsoft's technical references and is commonly utilized in real-time data integration projects.

2.3 Azure Data Factory (ADF) for CDC Orchestration

ADF is necessary to coordinate CDC pipelines. It provides linked services, datasets, and copy activities to facilitate data transfer between SQL Server, Azure Blob Storage, and Azure Data Lake Storage (ADLS) [4]. When CDC is enabled in SQL Server, ADF can only record changes rather than complete loads. On-premises SQL Server requires a

www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 10, October 2025, pp : 303-305

e-ISSN : 2583-1062
Impact Factor : 7.001

Selfhosted Integration Runtime (SHIR) [4]. Numerous studies and blogs highlight ADF's flexibility in handling batch and near-real-time CDC pipelines, integrating with Logic Apps and Azure Monitor for notifications [7].

2.4 Databricks and Delta Lake CDC

Databricks improves ADF by providing scalable transformation and storage options. Delta Lake guarantees ACID transactions, facilitates schema development, and offers time travel capabilities. Row-level changes across table versions can be recorded using the Change Data Feed (CDF) feature to make it easier for downstream applications to consume incremental changes [2]. The Lakeflow and AUTO CDC APIs from Databricks may make handling Slowly Changing Dimensions (SCD) even simpler [3]. Literature and community case studies show that Delta Lake significantly enhances CDC pipelines by simplifying code and enabling efficient incremental processing.

2.5 Comparative Studies & Related Work

Previous studies on incremental ETL have discussed the trade-offs between full-load and CDC pipelines. Although full-load pipelines are typically slower and more costly, they ensure completeness [6]. On the other hand, CDC pipelines are quick and effective, but they need to handle errors, schema changes, and idempotency carefully [5]. For real CDC implementations, both the Microsoft and Databricks documentation [1][2][3][4] offer useful guidance on how to schedule using Databricks Jobs, export results in specific formats, and set up automated alerts.

2.6 Challenges in CDC Pipelines

Some key challenges include:

- Using timestamps to handle pipeline updates and deletions accurately [6].
- Schema evolution in long-running systems may result in pipeline failures if Delta Lake's schema evolution features are not utilized [2].
- Issues with data migration, credentials security and compliance, and access controls [7].
- Z-Ordering, file compaction, and partitioning are used to optimize delta tables in order to preserve performance at scale [3].

3. METHODOLOGY

To ensure automation, scalability, and efficiency, the CDC implementation uses a structured methodology with SQL Server, Azure Data Factory (ADF), and Databricks. Once CDC is enabled in SQL Server, transaction logs are read to document all inserts, updates, and deletions. Functions are used to query incremental changes between log sequence numbers (LSNs), and these changes are kept in system-managed change tables. After the data is gathered, ADF, the orchestration tool, configures linked services for SQL Server and Azure Data Lake Storage (ADLS). Subsequently, pipelines are configured to pull only the altered records instead of full table loads. On-premises setups require a Self-hosted Integration Runtime (SHIR). The staged incremental data is saved in ADLS in either Parquet or CSV format for efficient processing downstream, even though ADF's scheduling and triggers allow for near real-time orchestration and interaction with Azure Monitor or Logic Apps for notifications.

The next step requires that the data be converted and saved in Databricks. Here, notebooks or scheduled procedures will handle the incremental records that were taken out of ADLS. These entries are merged into Delta Lake tables using the MERGE INTO operation. We ensure ACID transactions, facilitate schema evolution, and optimize query performance through file compaction and partitioning with Delta Lake. Additionally, the use of Databricks Change Data Feed (CDF) streamlines the consumption of incremental changes between table versions. The entire process is automated by ADF pipelines and Databricks Jobs scheduling, while activity runs in ADF and detailed logging in Databricks are used for monitoring. We also use email or Logic Apps notifications to keep operational visibility and alerting up to date.

4. RESULTS

There are several benefits to integrating SQL Server CDC, Databricks, and ADF. The pipeline greatly reduces computing costs and boosts overall performance by focusing on detecting and processing only minor changes rather than reloading large datasets. Data extraction and transformation become more effective as a result. Additionally, Delta Lake enables scalable management of large datasets through the use of Z-ordering, compaction, and partitioning, ensuring efficiency as data volume rises. Near real-time analytics are made possible by this pipeline's low latency, which also reduces delays in business decision-making.

Delta Lake provides transactional guarantees to maintain data consistency even during concurrent updates or possible failures, which further boosts reliability. ADF's adaptability is another benefit; it supports a wide range of integration runtimes and connectors, which makes it appropriate for hybrid architectures that utilize both cloud services and on-

premises SQL Server. These results demonstrate the high levels of efficiency, scalability, and resilience that can be provided by a CDC pipeline constructed on SQL Server, ADF, and Databricks.

5. DISCUSSION

The findings demonstrate the advantages and difficulties of building CDC pipelines with SQL Server, ADF, and Databricks. The log-based CDC mechanism in SQL Server is one important advantage; it continuously records changes while significantly reducing the performance impacts on OLTP applications. Other Azure services are easily integrated with ADF's scheduling, triggering, and monitoring tools. Incremental data processing is made much simpler by Databricks' and Delta Lake's scalability and advanced features, which include schema evolution, ACID guarantees, and Change Data Feed. These benefits make this combined approach a powerful alternative to traditional ETL pipelines.

However, challenges remain. The difficulties in handling updates and deletes in timestamp-based pipelines are lessened by the log-based CDC. Although many of these problems are lessened by Delta Lake's schema management features, schema evolution poses a risk to long-term systems. Operational complexity is another consideration because the setup necessitates proper configuration and upkeep across several services, such as SQL Server, ADF, Databricks, and ADLS. Costs could also be a problem because ongoing use of Databricks clusters and CDC ingestion could raise expenses. Organizations must also exercise extreme caution when implementing security and compliance measures, such as credential maintenance and access control, to protect their data pipelines.

When compared to other methods discussed in the literature, this hybrid approach strikes a good balance. Even though full-load ETL pipelines provide complete data, as scale increases, they may become slow and resource-intensive. Trigger-based CDC methods are simpler, but they have more delay and system overhead. Although they complicate the infrastructure, streaming alternatives based on Kafka can handle updates in real time. However, because it makes use of existing Azure investments, lowers latency, and ensures efficient incremental processing, the SQL Server–ADF–Databricks pipeline is an excellent choice for enterprise adoption.

6. CONCLUSION

In conclusion, a creative, automated, and scalable solution to modern data engineering issues is offered by the creation of Change Data Capture using SQL Server, Azure Data Factory, and Databricks. Combining log-based CDC in SQL Server with orchestration via ADF and transformation via Delta Lake in Databricks allows businesses to capture and manage changes in almost real-time. This permits schema change without disrupting ongoing operations, while also decreasing latency and improving reliability with ACID transactions. The approach we described enhances performance and enables companies to gain rapid insights while removing unnecessary resource usage when compared to traditional ETL processes.

However, this strategy has shortcomings in areas like cost management, operational complexity, and data governance, which call for careful consideration through automation, best practices, and monitoring. All things considered, this integration provides a workable and dependable solution for companies looking to update their data pipelines. Additional research could examine the use of machine learning techniques and real-time streaming platforms like Kafka or Azure Event Hubs to detect anomalies and improve the quality of the data in CDC streams.

7. REFERENCES

[1] Stacey, W., and F. Waas (2008). Adjust Data Capture for Real-Time Data Warehousing. Microsoft's SQL Server Group Technical Report.

[2] Das, T., Armbrust, M., Torres, J., et al. (2021). Delta Lake: High-Performance Acid Table Storage with Cloud Object Stores. Proceedings of the VLDB Endowment, 13(12), 3411–3424.

[3] Databricks Engineering Team (2021). Lakehouse CDC Patterns and Data Feeds are modified by Databricks. The Databricks Whitepaper.

[4] Dayal, U., Narasayya, V., and Chaudhuri, S. (2011). Overview of Business Intelligence Technology. ACM Communications, 54(8), 88-98.

[5] Stonebraker, M., and Çetintemel, U. (2005). "One Size Fits All: An Idea Whose Time Has Come and Gone." 21st IEEE International Data Engineering Conference (ICDE).

[6] Hamilton, J., Hellerstein, J. M., and Stonebraker, M. (2007). database system architecture. 141–259 in Databases: Foundations and Trends®, 1(2).

[7] Kreps, J., Rao, J., & Narkhede, N. (2011). Kafka: A Log Processing Distributed Messaging System. Workshop on NetDB.