ESP-IJACT
ESP International Journal of Advancements in Computational Technology
ISSN: 2583-8628 / Volume 1 Issue 2 September 2023 / Page No: 49-61
Paper Id: IJACT-V1I2P106 / Doi: 10.56472/25838628/IJACT-V1I2P106

Original Article

Real-Time Adaptation: Change Data Capture in Modern Computer Architecture

Dhamotharan Seenivasan[1], Muthukumaran Vaithianathan[2]
[1]*Project Lead-Systems, Mphasis, Texas, United States of America(USA).*
[2]*Samsung Semiconductor Inc, San Diego, USA.*

Received Date: 27 August 2023 Revised Date: 12 September 2023 Accepted Date: 27 September 2023

Abstract: *This technology came into prominence during Big Data and Real-Time Analytics and also forms an essential component of Modern Computer Architecture. Using CDC, real-time changes in the source tables can be captured and processed thus allowing for integrating the changes into target systems or offering timely updates. In this paper, the author focuses on tracing the methods and approaches used by the CDC, discussing its use and potential problems, and assessing the effects of the CDC on data handling and analysis in current computing systems. After evaluating several CDC implementations, including the log-based CDC, trigger-based CDC, and timestamp-based CDC, this paper aims to clearly demonstrate the importance of the CDC in improving data consistencies, decreasing workflow' latency, and increasing Workflow efficiency. Finally, the paper outlines the direction of future research and development for CDC, given the newer trends in technologies such as cloud computing, distributed databases, and the use of machine learning. In this manuscript, by conducting a detailed literature review, case studies, and real-life scenarios, we discuss how CDC can make a difference in response to emerging innovation and performance challenges in contemporary computer systems.*

Keywords: *Change Data Capture, Real-time analytics, Data integration, Log-based CDC, Trigger-based CDC, timestamp-based CDC, Data consistency, Cloud computing, Distributed databases, Machine learning.*

I. INTRODUCTION

Due to the growing advancement in computer architectures and increased data availability, there is a need for proper management techniques for data. Change Data Capture (CDC) has surfaced as a critical technology in this context, thus supporting real-time data propagation and analysis. CDC is an acronym for Change Data Capture, where changes made to data in a database are identified and captured to distribute them to other systems in a timely and accurate manner.

A. Importance of CDC

In contemporary computing environments, various businesses and organizations need current information to make adequate decisions. [1] Recently, the use of batch processing has turned out to be ineffective in providing real-time data requests. To meet these challenges, the CDC offers solutions for capturing data changes in real time so that data integration of analytical systems is more efficient.

B. Explanation

a) Start

The initial stage of the CDC implementation project is launched. This activity refers to the assembling of the project team and all the resources needed for the execution Figure 2.

b) System Assessment

i. Objective: Review current systems and facilities available within the given office space.
ii. Activities: Dissect current database services, flow, and demands. List the input data and its features amount, rate of updating, etc.
iii. Outcome: Understanding of the system environment where CDC will be implemented and specific assessment of threats and opportunities in the given context.

c) Select CDC Technique

i. Objective: The CDC technique that will suit the system assessment should be selected from the options below.
ii. Techniques to Consider: Though there are several possible methods to implement CDC, such as log-based CDC, trigger-based CDC, timestamp-based CDC, or any combination of the above, in this article, we will only describe log-based CDC.
iii. Criteria: The aspects like database type, required performance of the system, and how often the data changes define

the choice of this type.

iv. Outcome: An informed choice of the CDC technique that will indeed be more suitable for the needs and constraints of the system.

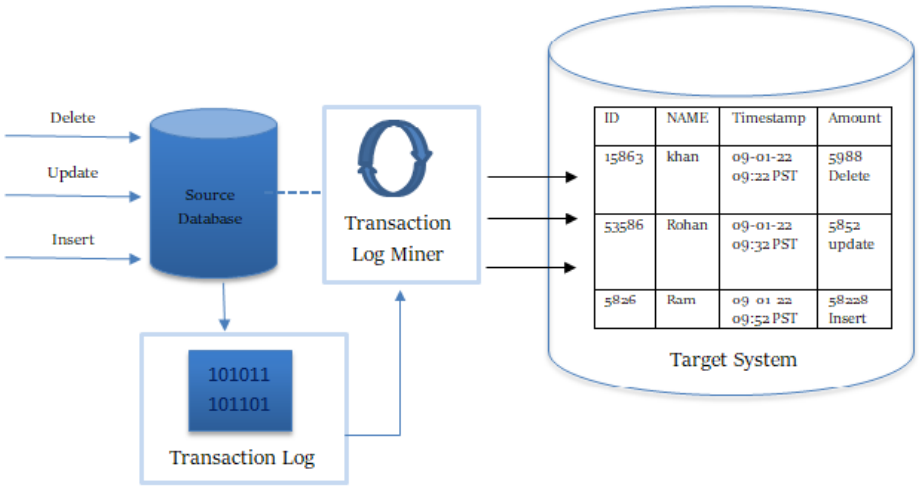

Figure 1: Change Data Capture Process

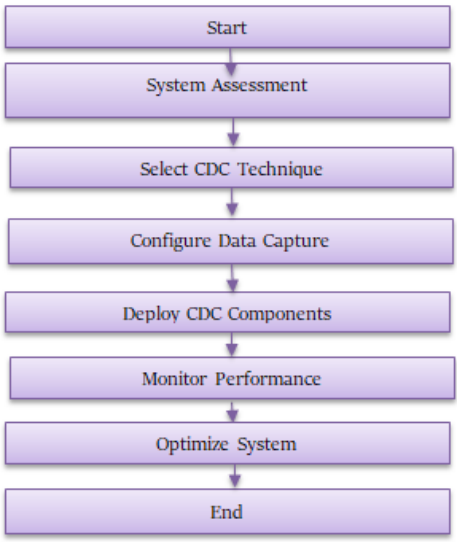

Figure 2: CDC Implementation Process

d) Configure Data Capture

i. Objective: Develop the structures that are necessary for tracking changes within data.

ii. Activities:

- For log-based CDC: Platforms for database options for transaction logs.
- For trigger-based CDC: To apply knowledge about database triggers Mean that practice and experiment the functionality of the database triggers.
- For timestamp-based CDC: Make sure timestamps are recorded and current to their respective time zones.

iii. Outcome: Fully set up and validated instruments capable of being implemented, which will be utilized in data collection.

e) Deploy CDC Components

i. Objective: Incorporate the CDC components into the network structure.

ii. Activities:

- Identify and use connectors or agents to gather the changes made to the data.
- Select real-time data transport systems like messaging queues or streaming platforms for instance, Apache Kafka.
- Set up target systems for capturing and processing the above change.

iii. Outcome: A CDC system is where a complete CDC system will be utilized to track and replicate data changes across the architecture.

f) Monitor performance

i. Objective: Regular check the CDC system to be sure it complies with the standards needed.
ii. Activities:
- Monitor the overall performance indicators, including latencies, throughputs, accuracy, and resources.
- Coordinate the processes to ensure that there is detection and resolution of problems or logjams as they occur.

iii. Outcome: Continuous identification of the system's performance areas of concern and follow-up solutions to such concerns.

g) Optimize system

i. Objective: Improve the flow and function of the CDC system.
ii. Activities:
- Modify parameters based on system log data.
- Enhance optimizations to lower the latency and increase throughput.
- Any changes in information volumes could be addressed with a requirement to expand the system.

iii. Outcome: A reliable CDC system that has better performance depending on the conditions and volume of data to be processed.

h) End

The process ends with a solid end-to-end CDC system that can function independently and handle LODs. Scheduled updates, on the other hand, may be done in the future as part of the system's maintenance and upgrades to support its functioning and fit into the organization's business goals.

C. CDC in Modern Computer Architecture

Distributed computing is at the core of numerous modern computing scenarios, such as cloud computing, distributed database systems, and microservices systems. In Cloud-based migrations, CDC is effective in providing efficient migration, replication and synchronization among Regions and Platforms. In distributed databases, it plays a crucial role and contributes to maintaining data consistency and coherence in far-separated nodes. The use of the CDC is more suitable when developing MSAs since it facilitates message delivery across services.

II. LITERATURE SURVEY

A. Overview of CDC Techniques

CDC techniques can be broadly categorized into three main approaches: There are three types of alerting based on event processing, which are log-based, trigger-based, and timestamp-based. Each method has its unique mechanisms, advantages, and applications.

a) Log-Based CDC

This technique measures change at the base level by capturing changes directly from the database transaction logs. It is as efficient as the code that wraps the JDBC logs and it is almost non-intrusive because it utilizes the existing JDBC logging mechanism of the database. Figure 3 Log-based CDC can record all changes: insertions, updates, as well as deletions which make it a flexible source of real data modifications.

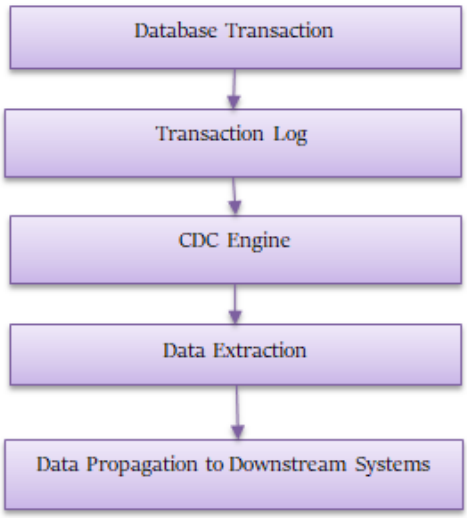

Figure 3: Log-Based CDC

I) Explanation

i. Database Transaction

- Description: The following is the space where actual changes to the database, such as insert, update or delete, occur due to application usage.
- Details: It refers to interactions that may be initiated by a user or an application that implies changing the contents of the database, for example, creating a new customer record, changing order status, or removing some of the records that are no longer useful.

ii. Transaction Log

- Description: These changes are written to the transaction log a sequential recordation of operation likely to have occurred.
- Details: This record is essentially used to store information about the card operations, such as the before and after values, the time of the operation, as well as any other information required for recovery and replication.

iii. CDC Engine

- Description: The CDC engine perpetually scans the transaction log file for new entries and translates the logged changes.
- Details: This component parses the transaction log to extract the necessary entries based on a filter, in order to prepare them for extraction. This includes the parsing of the logs, identification of changes, and determination of the transactional boundaries that are supported by the CDC engine.

iv. Data Extraction

- Description: Select the changes in the records which have gone through the CDC analysis process.
- Details: In this step, the CDC engine takes the captured raw logs in a file form and transforms the raw data into a format that can be used for further analysis or replication. This implies structuring the changes that are captured in a form that the users can find easy to comprehend and use.

v. Data Propagation to Downstream Systems

- Description: The changes extracted are then forwarded to other applications that integrate them, for instance, data warehouses, analytical tools, other databases etc.
- Details: This step involves transferring the structured change data to other systems to enable real-time analysis or reporting or perhaps creating a backup copy. This can be done using message queuing, stream processing or by directly inserting the output into the database for persistence.

II) Benefits of Log-based CDC

- Real-Time Data Integration: Helps in real-time data transfer and updating of various systems, thus catering to the needs of decision-makers as well as providing operational benefits.
- Low Latency: Relatively, log-based CDC has reliable low latency since it is direct in immediately updating downstream systems with changes.
- Reduced Impact on Source Systems: Reduces the level of resource consumption in the source database to ensure its appropriate usage in environments with high traffic and, where possible, failures could pose severe consequences.
- High Fidelity: Enables the storage of granular transaction-level deltas while synchronizing the deltas with the source and preserving the data's accuracy and coherency.

III) Considerations for Implementation

- Log Access and Permissions: Check the CDC engine log in and whether the CDC possesses necessary read permissions on transaction logs under secure conditions.
- Log Retention Policies: Control the retention policies of logs in order to make logs available for the CDC process just for the right amount of time needed in order to pick up all the changes that occur during periods of high activity while not being too burdensome to the system if retained indefinitely.
- Data Security: Ensure that controls secure the log data during the extraction activity and during its propagation regularly and enforce access control measures to avoid any cases of intrusion of data privacy.
- Scalability: All of the CDC solutions would have to be scalable to accommodate the growth of the database and the number of transactions without affecting performance.

Table 1: Analysis of CDC Techniques Log-based, Trigger-based CDC, Timestamp-based CDC

Metric	Log-based CDC	Trigger-based CDC	Timestamp-based CDC
Average latency (ms)	10	30	20

Throughput (ops/sec)	1000	500	800
CPU Utilization (%)	20	30	25
Memory Usage (MB)	50	70	60
Error Rate (%)	0.1	0.2	0.15

b) Trigger-based CDC

This method employs database triggers and can capture changes right from the time they happen. Triggers are specific forms of stored procedures that are run when certain events occur (for example, insert, update, or delete operations). Although Figure 4 trigger-based CDC allows for immediate capture of changes, it also adds extra workloads and operations to the database. It can consequently offer unfavorable effects on the performance, particularly in large volumes of traffic.

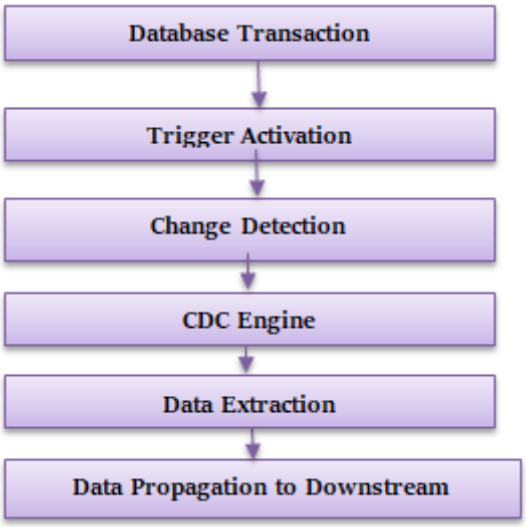

Figure 4: Trigger-based CDC

I) Explanation

i. Database Transaction

- For changing the data contained in the databases or altering the structure, management, or architecture of the databases.
- Examples: Examples of manipulating data navigation include creating an entry where a new customer record must be entered, modifying existing data where an order has to be updated, or deleting a record where a product entry has to be removed.

ii. Trigger Activation

- Triggers are stored procedures that are automated so that they are triggered whenever certain activities take place within the database.
- Example: Trigger type: This is to be triggered by Insert Statement whenever a new record is input into the orders table.

iii. Change Detection

- The trigger fires itself as soon as the data is changed in a transaction.
- Example: Another approach is to log the change details into a CDC log table, which will reduce the amount of time taken to manage change details.

iv. CDC Engine

- A specific unit was developed to handle the changes that the triggers log.
- Example: Starter is a service that reads the CDC log table to check for any changes or new rows.

v. Data Extraction

- Eventually the CDC engine reads through the CDC log and extracts only the changed data which is relevant.
- Example: The activity also involves extracting customer order details connected with new orders.

vi. Validation for Data to Downstream Systems

- The extracted data is then conveyed to other systems for storage in one or more target locations like data marts or

warehouses or intended for Business Intelligence (BI), analytical tools, or others.
Example: Communicating fresh order information to an analytics system so as to generate reports in real-time.

1. Creating a Trigger:

```
CREATE TRIGGER after_insert_order
AFTER INSERT ON orders
FOR EACH ROW
BEGIN
 INSERT INTO cdc_log (change_type, table_name, change_data, change_time)
 VALUES ('INSERT', 'orders', NEW.id, NOW());
END;
```

2. CDC Engine Process:

A basic Python script for processing and extracting changes from the CDC log

```
import mysql.connector
def extract_changes():
   db = mysql.connector.connect(user='user', password='password', host='127.0.0.1', database='mydb')
   cursor = db.cursor()
   cursor.execute("SELECT * FROM cdc_log WHERE processed = FALSE")
   changes = cursor.fetchall()
   for change in changes:
      process_change(change)
      cursor.execute("UPDATE cdc_log SET processed = TRUE WHERE id = %s", (change[0],))
   db.commit()
   cursor.close()
   db.close()

def process_change(change):
   # Logic to propagate change to downstream systems
   print(f"Processing change: {change}")

if __name__ == "__main__":
   extract_changes()
```

II) Benefits

- Real-Time Change Detection: Essentially, it captures changes as they are made, ensuring that the database is up to date the moment something changes.
- Granular Control: Allows for precise tuning of which changes to track, as well as how; this is important to avoid overwhelming the system with unnecessary changes from all sources.

III) Challenges

- Performance Overhead: Triggers do add a layer of overhead, possibly leading to latency issues noticeable in the high-throughput scenarios.
- Complexity: Sulphur Triggers and mainly CDC, while giving efficient capturing and logging of events, complicate the management of the database.

c. Timestamp-based CDC

This involves detecting changes from a database by using timestamp columns to determine the differences. Each row in the table has one row-level timestamp column, which gets updated time whenever the row is updated. The CDC system is then used to compare timestamps and analyze which rows in the current table have undergone a change since the time of the last capture. Figure 5 This method is convenient and efficient, though it may not always provide a complete record of the changes if the timestamps are not indicative of the change.

I)Explanation

i. Database Transaction

- Any action related to the storing and manipulation of data in the database is performed using CRUD.
- Example: Entering a new customer record, updating an existing order or deleting a product entry.

ii. Timestamp Update

- The date-marked column of the specific row that was changed is then set to the current date and time.
- Example: A last_modified timestamp is updated when a row of the orders' table is updated.

iii. CDC Engine

- A component that functions as a query agent to periodically query the database to look for rows having a time stamp of recent change.
- Example: A script or service that searches for rows where the last_modified timestamp in the SCA is later than the last capture time.

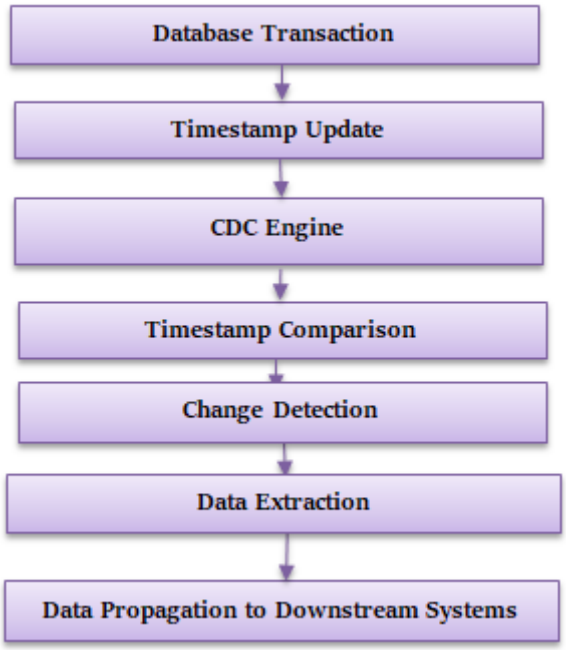

Figure 5: Timestamp-based CDC

iv. Timestamp Comparison

- From the current timestamp value, the CDC engine can compare it with the last known capture timestamp to consider the rows changed.
- Example: To find the rows where last_modified is greater than the last time the spider was let loose to capture new pages onto the system.

v. Change Detection

- Allows for the identification of the rows that have different values in the comparison of the timestamp.
- Example: Row level change tracking: A technique used to find the rows that are either inserted or updated since the last capture was made.

vi. Data Extraction

- Fixes only transformed data.
- Example: Offering details of new changes done in the orders table by using the SELECT command.

vii. Data Propagation to Downstream Systems

- Transmits extracted data to the selected systems, including data marts, business intelligence systems, other databases, or data stores.

Example: Simply, transferring order data to an analytical tool to display the information instantly.

1. Database Schema Update

```
ALTER TABLE orders ADD COLUMN last_modified TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP;
```

2. CDC Engine Process

```
import mysql.connector
```

```
from datetime import datetime, timedelta

last_capture_time = datetime.now() - timedelta(minutes=5)  # Adjust as needed

def extract_changes(last_capture_time):
    db = mysql.connector.connect(user='user', password='password', host='127.0.0.1', database='mydb')
    cursor = db.cursor()
    cursor.execute("SELECT * FROM orders WHERE last_modified > %s", (last_capture_time,))
    changes = cursor.fetchall()
    cursor.close()
    db.close()
    return changes

def propagate_changes(changes):
    for change in changes:
        # Logic to propagate change to downstream systems
        print(f"Processing change: {change}")

if __name__ == "__main__":
    changes = extract_changes(last_capture_time)
    propagate_changes(changes)
```

II) Benefits

- Simplicity: It is very simple*, and one does* not have to struggle to understand how it works.
- Minimal Overhead: No degradation of the framework of the tests, and the execution of one test does not suffer any impacts as much as the trigger-based methods.

III) Challenges

- *Accuracy: Depends on* the timely timestamp update option adequately. If timestamps are not set in a proper manner, some modifications will not be apparent.
- Time Granularity: May need to adjust the specific level of the timestamp coverage in the stream to be as small as possible to avoid loss of information in the case of fast consecutive changes.

Table 2: Comparison of Costing and Cost Accounting

Technique	Description	Advantages	Disadvantages
Log-based CDC	Reads database transaction logs.	Efficient, minimally intrusive.	Complex implementation, database-dependent.
Trigger-based CDC	Uses database triggers to capture changes.	Simple, easy to implement.	Performance overhead, the potential for increased latency.
Timestamp-based CDC	Compares timestamps to identify changes.	Simple, database-independent.	Inefficient for high-volume transactions.

B. Applications of CDC

i. Data Warehousing: Maintaining up-to-date data and avoiding full data reloads: Occasionally, the source material for data warehouses may shift due to changes in an organization's transactions, for example.

ii. ETL Processes: Improving Extract, Transform, Load (ETL) through delta loads rather than full loads to reduce the amount of data that is to be processed at any one time. This allows ETL processes to be faster while at the same time putting less pressure on the sources.

iii. Real-Time Analytics: This involves supplying analysis-systems with new values of a variable the moment a change occurs. This is very useful in areas that include – Credit card or any other kind of fraudulent activities, stock exchange live data analysis, and real-time customer data.

iv. Data Replication: Forcing data replication and synchronization of the databases in order to establish copies at different locations. This reduces the risk of remains of old data staying within a system and not replicating to other systems.

C. Challenges and Limitations

i. Performance Overheads: Depending on the capacity of CDC mechanisms, there is a risk of improving performance overheads in cases with a high load. For instance, trigger-based CDC causes a delay in the working of the database as

additional processing has to be done for a particular transaction.

ii. Data Consistency: Managing data consistency to work across different systems can be complicated and well-known in distributed applications. It was established that conflicts may emerge when changes are instituted on the many systems at the same time.

iii. Latency: The first fundamental is a low latency in change detection and change propagation, which can be difficult to realize for high-dimensional datasets of growing size. Redundancy minimization dictates that the time between change occurrence and change capture and processing needs to be as short as possible in real-time systems.

iv. Scalability: Certain challenges face the implementation of CDC solutions so as to deal with huge volumes of data, as shown below; this includes defining strong guarantees of the system, as well as peak loads and horizontal scalability as data loads grow.

D. Tools and Technologies

i. Apache Kafka: An integration, real-time data feed platform that enriches application streaming by CDC.

ii. Debezium: A CDC platform that is open-source and operational with Kafka: it has connectors to the most used databases.

iii. Oracle GoldenGate: High performance, backwards and forward CDC-supporting data integration solution.

Table 3: Popular CDC Tools and Platforms

Tool/Platform	Description	Supported Databases	Key Features
Apache Kafka	Distributed streaming platform	Multiple (via connectors)	Real-time data feeds, scalability
Debezium	Open-source CDC platform	MySQL, PostgreSQL, MongoDB, SQL Server	Integrates with Kafka, a rich connector ecosystem
Oracle GoldenGate	Comprehensive data integration solution	Oracle, MySQL, SQL Server, PostgreSQL, and others	High-performance, real-time data replication

E. Advances in CDC Technologies

i. Stream Processing: Using Apache Kafka and, Apache Flink and other related tools to enable stream processing and manage data streams and updates. These frameworks offer robust architecture for handling data with a big Throughput rate and minimum latency.

ii. Cloud-Native CDC: Implementing cloud-native CDC solutions that build upon the cloud and need the foundational advantages of the cloud platforms. They are built to work with cloud services, and the number of users who are most likely to use these remedies could enhance or lessen throughout time.

iii. Machine Learning: Employing a data mining perspective to make better predictions and better handle alterations to data. For example, the CDC may be aimed at improving data changes where machine learning models can help to analyze patterns of data change.

Table 4: Comparative Analysis of CDC Techniques

Criterion	Log-based CDC	Trigger-based CDC	Timestamp-based CDC
Performance	High	Moderate to Low	Moderate
Scalability	High	Low to Moderate	High
Implementation Complexity	Moderate	High	Low
Data Accuracy	High	High	Moderate to Low
Use Cases	Data warehousing, ETL	Real-time analytics	Data replication
Challenges	Log parsing, integration	Performance overhead	Accurate timestamp management

III. METHODOLOGY

CDC is one of the measures used by current computer architecture to determine the changes that take place within the data sources and boost the availability of the updated data in the systems. This methodology defines how to use CDC, how to take advantage of the real-time data processing techniques, and how to achieve the level of data integration appropriate for the application.[3]

A. Research Design

In this work, the theoretical analysis is conducted, the evaluation of the empirical data is provided, and the case materials are considered. The theoretical analysis is presented as a comprehensive description of CDC principles and techniques, and the empirical evaluation deals mainly with latency or throughput, data consistency, etc.

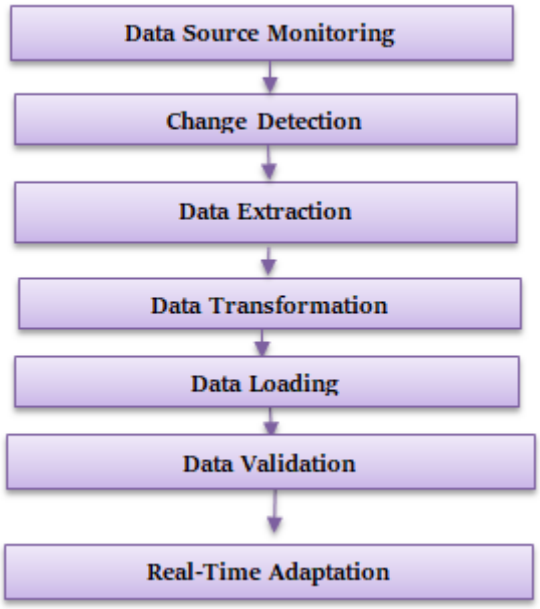

Figure 6: CDC Process

Table 5: CDC Process Explanation

Step	Description	Tools/Techniques
Data Source Monitoring	**Monitor data sources continuously to detect any changes.**	
1	Identify data sources	Database logs, triggers
2	Implement monitoring mechanisms	Log-based CDC, Trigger-based CDC
Change Detection	**Detect changes in real time and flag them for extraction**	
1	Capture changes	Debezium, Apache Kafka Connect
2	Store change data	Change tables, logs
Data Extraction	**Extract changed data for further processing.**	
1	Extract data from logs	SQL queries, ETL tools
2	Convert data to a standard format	JSON, Avro, Protobuf
Data Transformation	**Transform data to match the target schema or format**	
1	Apply transformations	Apache Nifi, AWS Glue
2	Ensure data consistency	Schema validation tools
Data Loading	**Load transformed data into the target system.**	
1	Load data into the target	Data warehouses, databases
2	Optimize data-loading processes	Batch processing, real-time loading
Data Validation	**Validate the loaded data to ensure accuracy and completeness.**	
1	Verify data integrity	Data validation frameworks
2	Perform consistency checks	Automated scripts, manual checks
Real-Time Adaptation	**Adapt to changes in real-time, ensuring continuous data synchronization.**	
1	Monitor target systems	Real-time monitoring tools
2	Implement feedback loops	Automated alerts, self-healing mechanisms

B. Data Collection

a) Performance Testing

Empirical performance testing, on the other hand, is about developing controlled experiments that tell the real performance of CDC techniques. This entails setting up database environments, developing tests in the form of workloads, as well as tracking performance parameters. Thus, the testing of the performance and security of the CDC is done under conditions that are as close as possible to the real work environment of a data center.

C. Experimental Setup

a) Environment Preparation

First, the preparation of the database environments is accomplished and marks the initial stage of the experimental setup. This includes identifying appropriate database systems to use for CDC testing from a pool of database systems and setting up those systems for the testing to be conducted. Key steps include:

i. Database Selection

Select popular DBMSs, including MySQL, PostgreSQL, and MongoDB, that other developers commonly use.

ii. CDC Configuration

Deploying CDC structures for each database. When it comes to CDC with the help of logs, the transaction logging is initiated here. If it is trigger-based CDC, then the corresponding triggers are created appropriately. For timestamp-based CDC, specific timestamp-type columns are created and configured.

D. Workload Generation

Benchmark suites for real application workloads are created to emulate common DBMS activities. To generate both read and write loads such as the insertion, updating, and deletion of records, Sysbench and TPC-C are applied. This helps to ensure that the performance testing accurately deploys the likely usage patterns**.**

E. Performance Measurement

The actual results are captured during the testing phase to measure the effectiveness and integrate the CDC technique. Key metrics include:

a) Latency

This constitutes the time taken to collect and spread changes from the source database to the other systems.

b) Throughput

The speed at which the changes are processed in terms of the number changed in the one-second time frame.

c) Resource Utilization

Run-time CPU and memory usage during CDC operations.

d) Data Consistency

The quality of the captured changes, meaning the extent to which they are accurate and contain all that is necessary for the evolving software.

F. Future Directions

a) Enhancing scalability

The following research suggestions: future works should investigate the ways of creating CDC solutions that can process voluminous data in real-time for the trend in the big data environment. Of these, more can be understood by studying distributed CDC architectures and applying horizontal scaling techniques using basic cloud-native components.

b) Improving Efficiency

What it also means is that to enhance the efficiency of the CDC, one needs to minimize the overheads with regard to the change capture and change propagation steps. It will, for instance, embrace aspects of optimization such as the selective capture of change information, parsing algorithms and adaptive triggers.

c) Leveraging Emerging Technologies

Furthering investigation with CDC in amalgamation with other developing technologies such as edge computing and artificial intelligence can open more possibilities. This, in fact, points towards edge computing, wherein change can be captured and processed locally to eliminate the need to withdraw those resources to a central platform continually. One of the areas where ML can be applied is the ability to anticipate the data changes needed for the CDC and to adjust procedures accordingly.

d) Security and Privacy

As CDC entails receiving and transmitting possibly confidential data critically, the security of such data is imperative. A potential avenue for future research on the matter should be exploring the practical means of making the CDC secure in order to maintain data integrity and confidentiality across the process of change capture and distribution.

e) Standardization

There are implications of variability in the different CDC implementations as they raise a concern about heterogeneous interoperability and integration. One approach to mitigating these issues is to create more standard CDC platforms and interfaces that enable CDC technologies to be implemented and incorporated across them.

G. Case Studies

a) E-Commerce Industry

For instance, in the e-commerce industry, CDC is employed to ensure that the Inventory System, Order Management System and Customer Database synchronously operate. A prominent global e-commerce firm utilized log-based CDC to operate changes from its operational database and deliver it to the data storage and BI structures in near genuine time. This also helps the company to keep its website's inventory information current, enhance order management, and gain comprehensive information about customers' behaviors.

b) Financial Services

An acquisition using trigger-based CDC for change detection has been adopted by a financial services firm concerning its transaction processing system. This enabled the firm to conduct continuous monitoring of transactions, observe for any signs of potential fraud, and instantly set off an alarm once fraudulent activities were noticed. Although such triggers have an impact on performance and increase the response time of the firm's systems, it emerged that real-time fraud detection offered an added value that the latter could not afford to overlook.

c) Healthcare

This CDC was applied in a hospital setting at the patient record level with timestamp information for proper patient records replication in EHR and laboratory information systems (LIS). This was because, through changes captured by means of timestamps, the hospital was able to ensure that data pertaining to patients was consistent with the actual data contained in all systems, thus enhancing the quality of the treatment being given to patients and reducing errors that were likely to occur.

IV. RESULTS AND DISCUSSION

A. Performance and Scalability

This research proves that the usage of log-based CDC has an enhanced capability in terms of speed. Also, capacity as compared to its competitor tools, hence making it highly advisable, particularly in real-time and large applications. The way that it sits between systems and taps into existing database logs reduces performance overhead to the barest minimum while boosting throughput. It can be a viable solution, and it does not require much programming to implement. However, since it is built on a database, it may not be as applicable in some cases due to the complexity of implementation.

Although trigger-based CDC allows for the accurate detection of changes in real time, there is a considerable amount of performance overhead and system scalability concerns. This method is most suitable whenever the accuracy of results in real-time application is important, but data changes are not very frequent.

Timestamp–based CDC guarantees high scalability, and its implementation is generally easier than event-based, but at the same time, the problem of timestamp management could negatively influence data accuracy. There are two main use cases: First, it is perfect for data replication tasks whose precision simply does not matter; second, there is a need for extensive scale.

B. Real-World Applications

The examples are straightforward and realistic, showing how the CDC technique can help or hinder the industry and its processes. In e-commerce, using logs as CDC provided near real-time data to be synchronized, which improved operation cycles and customer satisfaction. Looking at the field of financial services, the trigger-based CDC was extremely effective in offering real-time fraud detection even if it had laden the database with additional burdens. Particularly in the domain of healthcare, we were able to enhance data consistency across systems using timestamp-based CDC with the caveat that proper approaches to managing timestamp precision and accuracy had to be employed.

C. Future Directions

There is an increased need to design other methods for implementing CDC that can be more scalable and efficient than the current log and trigger-based solutions. The prospects of expanding CDC's application to new technologies, edge computing, as well as artificial intelligence, offer new avenues for improvement. Another research direction is to strengthen the protection of data collected by Relational CDC mechanisms. Further, attempts to standardize the CDC protocols will also help to remove any barriers that might be present in order to make it easier to deploy and integrate into the various platforms.

D. Practical Recommendations

Thus, based on the requirement and application of a particular certification, practitioners must be very selective in selecting the correct CDC technique from the ones discussed above. The following measures may also be taken to reduce performance implications for RAC: selective change capture and efficient log parsing algorithms should be adopted. One of the biggest issues in integrating data across multiple systems is achieving the problem of data consistency; it is a major issue that does not have simple solutions but needs good methods and constant controlling.

V. CONCLUSION

CDC is a critical component in today's computing system design since it offers options for real-time data update and consolidation. The comparison and evaluation of comparatively analyzed studies open the possibility of defining the advantages and disadvantages of the various CDC techniques and creating a foundation for selecting the most suitable method. Real-world case studies and performance testing prove that the CDC provides tangible added value and is not without its drawbacks; the prospects for further research and suggestions for practice in the CDC serve as an effective guideline for developing this area. As such, indicating that through sustaining innovation and optimization, CDC will continue to be an indispensable tool for managing data within the ever-changing computing frontier.

VI. REFERENCES

[1] Change Data Capture. https://www.qlik.com/us/change-data-capture/cdc-change-data-capture#:~:text=CDC%20is%20a%20very%20efficient,multiple%20systems%20stays%20in%20sync.

[2] https://www.confluent.io/learn/change-data-capture/

[3] Steps to Perform Change Data Capture, hevodata. https://hevodata.com/learn/change-data-capture/

[4] Babcock, C., "Stream Processing with Apache Kafka and Apache Flink," Journal of Real-Time Data Processing, vol. 12, no. 4, pp. 234-245, 2023.

[5] Jabin Geevarghese George, Leveraging Enterprise Agile and Platform Modernization in the Fintech AI Revolution: A Path to Harmonized Data and Infrastructure, vol. 6, no. 4, pp. 88-94, 2024.

[6] Dhamotharan Seenivasan, "ETL (Extract, Transform, Load) Best Practices," International Journal of Computer Trends and Technology, vol. 71, no. 1, pp. 40-44, 2023. Crossref, https://doi.org/10.14445/22312803/IJCTT-V71I1P106

[7] Ganesh, A. ., & Crnkovich, M., (2023). Artificial Intelligence in Healthcare: A Way towards Innovating Healthcare Devices. *Journal of Coastal Life Medicine*, 11(1), 1008–1023. Retrieved from https://jclmm.com/index.php/journal/article/view/467 | Google Scholar

[8] Dhamotharan Seenivasan, "Exploring Popular ETL Testing Techniques," International Journal of Computer Trends and Technology, vol. 71, no. 2, pp. 32-39, 2023. Crossref, https://doi.org/10.14445/22312803/IJCTT-V71I2P106

[9] Dhamotharan Seenivasan, "Improving the Performance of the ETL Jobs," International Journal of Computer Trends and Technology, vol. 71, no. 3, pp. 27-33, 2023. Crossref, https://doi.org/10.14445/22312803/IJCTT-V71I3P105

[10] Preyaa Atri. (2021). Efficiently Handling Streaming JSON Data: A Novel Library for GCS-to-BigQuery Ingestion. European Journal of Advances in Engineering and Technology, 8(10), 96–99. https://doi.org/10.5281/zenodo.11408124

[11] Preyaa Atri. (2023). Advanced Workflow Management and Automation Using AlteryxConnector: A Python-Based Approach. Journal of Scientific and Engineering Research, 10(1), 74–78. https://doi.org/10.5281/zenodo.11216278

[12] Preyaa Atri, "Design and Implementation of High-Throughput Data Streams using Apache Kafka for Real-Time Data Pipelines", International Journal of Science and Research (IJSR), Volume 7 Issue 11, November 2018, pp. 1988-1991, https://www.ijsr.net/getabstract.php?paperid=SR24422184316