A COMPREHENSIVE FRAMEWORK FOR CHANGE DATA CAPTURE FROM GOOGLE CLOUD SPANNER TO BIGQUERY USING DATAFLOW

Ashok Gadi Parthi[1],
ORCID: [1]0009-0007-4048-5291,

ABSTRACT

This paper presents a robust and scalable solution for real-time data replication from Google Cloud Spanner to BigQuery using Change Data Capture (CDC) with Google Cloud Dataflow. The architecture ensures that every insert, update, and delete operation in Spanner is reflected in corresponding BigQuery tables with minimal latency. By leveraging Spanner Change Streams and Dataflow, this framework efficiently processes data transformations and ensures fault tolerance. Building upon Google's foundational CDC solution, this paper introduces several enhancements, including support for multiple data types, schema evolution, efficient null handling, and improved exception management. The proposed framework enables enterprises to maintain up-to-date analytics with optimized performance and reliability.

Keywords: Spanner, Dataflow, Change Data Capture, Google BigQuery, Data Integration

1. INTRODUCTION

Modern data-driven enterprises require seamless synchronization between transactional databases and analytical platforms. Google Cloud Spanner provides strong consistency and global distribution for operational workloads, whereas BigQuery offers a serverless, scalable data warehouse for real-time analytics.
Real-time replication from Spanner to BigQuery provides:

- ✓ Continuous insights from transactional data.
- ✓ Reduced ETL complexity using real-time CDC.
- ✓ Scalability and fault tolerance for high-volume data operations.

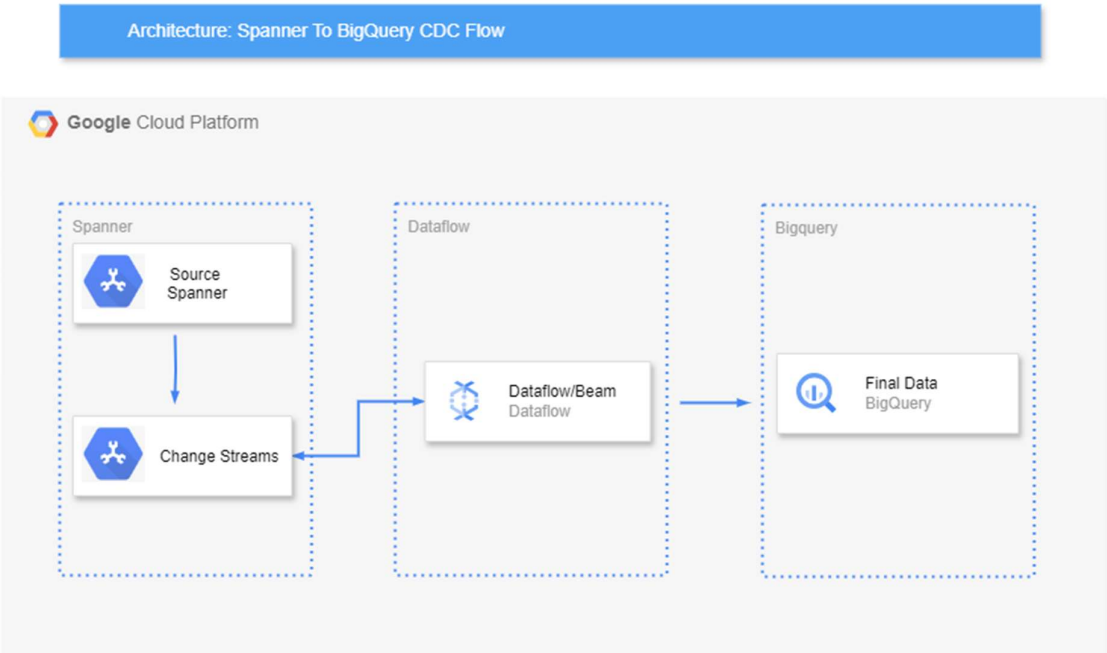
The proposed solution utilizes Spanner Change Streams to capture data mutations, Dataflow [7] to process and transform changes, and BigQuery as the final analytics destination.

1.1 Architecture

Spanner Change Streams: Captures all mutations (insert, update, delete) in Spanner tables.

- ✓ Dataflow [8] Pipeline: Reads change stream records, processes transformations, and writes to BigQuery.
- ✓ BigQuery Tables: Serves as the target storage for processed change records.

This architecture supports a continuous CDC mechanism, ensuring real-time data replication with high scalability.



2. FRAMEWORK KEY FEATURES

2.1 Real-Time Synchronization

A fundamental requirement of any CDC system is to ensure that updates in the source system are propagated to the target in near real-time. This framework leverages Spanner Change Streams to capture modifications at the row level as they occur. These changes are subsequently ingested by Google Cloud Dataflow [7], which applies the necessary transformations before writing them to BigQuery using streaming inserts.

To ensure data consistency, event ordering is preserved using committed timestamps from Spanner, which facilitate accurate sequencing of changes. The streaming architecture eliminates batch processing delays, allowing analytics and decision-making systems to operate on the most up-to-date data. This is particularly critical for fraud detection, operational monitoring, and personalized recommendation engines, where real-time insights drive business value. [3].

2.2 Scalability

As organizations deal with high-volume transactional data, scalability becomes a key design consideration. This pipeline employs Google Cloud Dataflow's parallel processing architecture, which automatically scales based on data ingestion rates. It ensures efficient resource allocation through:

Auto-scaling capabilities, which dynamically adjust compute resources based on incoming workload.
Partitioned processing of change records to allow parallel execution, reducing latency and improving throughput.
Optimized schema design in BigQuery, leveraging partitioning and clustering techniques for efficient query execution at scale.
By utilizing Apache Beam's distributed processing model, the pipeline can handle

millions of transactions per second, making it suitable for financial services, telecommunications, and e-commerce platforms that require continuous data synchronization across transactional and analytical systems [6].

2.3 Fault Tolerance

A critical challenge in real-time data replication is ensuring fault tolerance to handle network failures, processing errors, and transient issues without data loss. This CDC framework addresses fault tolerance through several mechanisms:

Checkpointing and stateful processing: The pipeline maintains periodic checkpoints, ensuring that in the event of failure, processing resumes from the last successful point rather than reprocessing the entire dataset.
Automated error handling and retries: Transient failures, such as network timeouts or API rate limits, trigger automated retries with exponential backoff.
BigQuery buffering and dead-letter queue (DLQ): Records that fail to be processed after multiple retries are redirected to a separate storage location for further analysis and reprocessing.
Self-healing capabilities: The system continuously monitors for pipeline failures, stale processing states, or degraded performance, triggering automated recovery workflows where necessary.
This fault-tolerant architecture ensures that no data is lost or duplicated, making the pipeline highly reliable for mission-critical applications [6].

2.3 Schema Evolution

One of the primary complexities in CDC pipelines is handling schema changes without disrupting downstream analytics or requiring manual interventions. The proposed solution addresses schema evolution using:

Automatic schema detection, which continuously monitors new columns, data type modifications, and table structure changes in Spanner.
Dynamic schema adaptation, where the pipeline automatically modifies BigQuery table schemas to accommodate changes without requiring pipeline redeployment.
Backward compatibility mechanisms, ensuring that historical data remains accessible even if the schema evolves over time.
This feature is particularly useful for organizations operating in agile environments where database schemas frequently change, allowing them to iterate on their data models without pipeline disruptions [4, 6].

2.4 Efficient Exception Handling

Data pipelines operating in production environments must include robust exception handling mechanisms to ensure data quality, integrity, and efficient debugging. This framework incorporates:

Categorized error logging, where failures are classified into transformation errors, schema mismatches, API failures, and data consistency issues.
Retry mechanisms, where temporary processing failures are automatically retried a configurable number of times before being escalated.
Dead-letter queue (DLQ) for unrecoverable errors, ensuring that problematic records are logged separately rather than causing pipeline failures.

Actionable insights from failure patterns, enabling engineers to proactively identify and address root causes without extensive manual debugging.
This structured approach to error detection, logging, and recovery significantly reduces the time required to identify and resolve data inconsistencies, improving overall pipeline resilience [7].

3. IMPLEMENTATION

The implementation of the Spanner to BigQuery CDC pipeline involves setting up Spanner Change Streams to capture data modifications, designing a Dataflow pipeline to process and transform these changes, and ensuring fault tolerance through automated checkpointing and error recovery mechanisms. The following sections detail the configuration and execution steps required for seamless real-time data synchronization [8].

3.1 Setting up Spanner Change Streams

Google Cloud Spanner Change Streams provides a mechanism to capture modifications, including inserts, updates, and deletions, across one or multiple tables. This functionality is essential for real-time Change Data Capture (CDC) pipelines, ensuring continuous synchronization between transactional and analytical systems. in Fig-1:

a. Creating a Change Stream:

```
CREATE CHANGE STREAM < spanner_change_stream> FOR
<spanner_table_name> OPTIONS (value_capture_type='NEW_ROW').
```

b. Grant Permissions:

```
GRANT EXECUTE ON CHANGE STREAM <spanner_change_stream>
TO 'service-account-email'
```

3.2 Dataflow Pipeline Overview

The Dataflow pipeline facilitates the seamless movement of data from Spanner to BigQuery by performing three primary tasks:

a. **Read Change Streams:** Extracts real-time modifications from Spanner tables.
b. **Transform Data:** Applies necessary transformations to align with BigQuery's schema.
c. **Write to BigQuery:** Inserts transformed data into target tables for analytical processing.

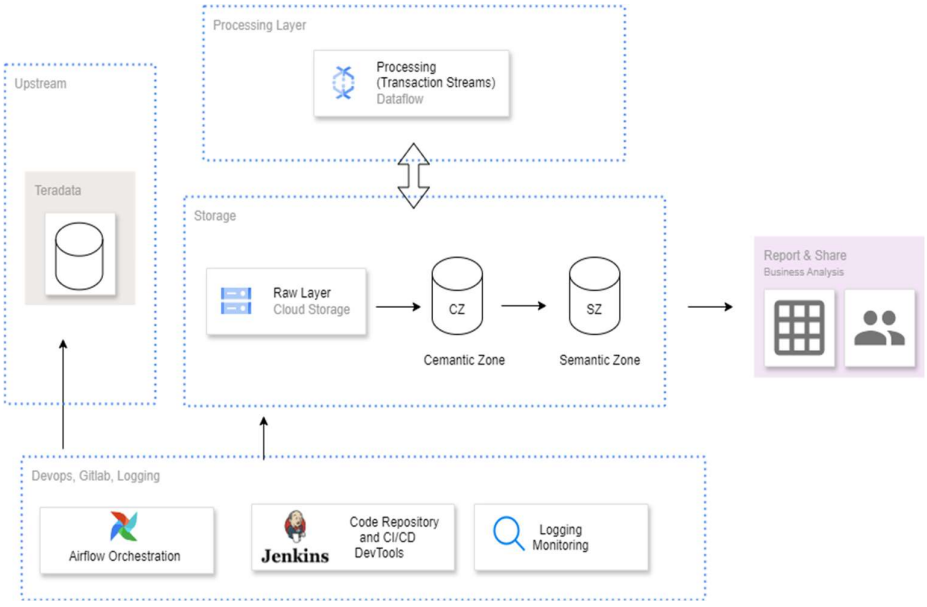


Figure 1. Migration architecture diagram

3.3 Dataflow Pipeline Implementation

The following are the implementation steps [9].

a. Reading from Spanner Change Streams:

```
SpannerIO.ReadChangeStream changeStream =
SpannerIO.readChangeStream()
   .withSpannerConfig(SpannerConfig.create()
      .withInstanceId("your-instance-id")
      .withDatabaseId("your-database-id"))
   .withChangeStreamName("spanner_change_stream");
```

b. Transforming Data for BigQuery:

```
PCollection<ChangeStreamRecord> transformedRecords = input
   .apply("TransformChanges", ParDo.of(new DoFn<ChangeStreamRecord,
TableRow>() {
      @ProcessElement
      public void processElement(@Element ChangeStreamRecord record,
OutputReceiver<TableRow> out) {
         TableRow row = mapToBigQueryFormat(record);
         out.output(row);
      }
   }));
```

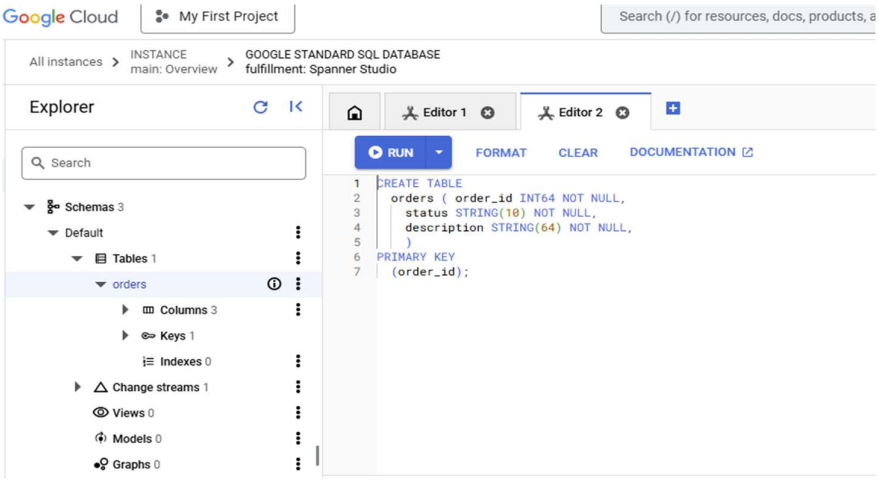
c. Writing Data to BigQuery:

```
transformedRecords.apply("WriteToBigQuery",
BigQueryIO.writeTableRows().to("project-id:dataset-id.table-id")
```

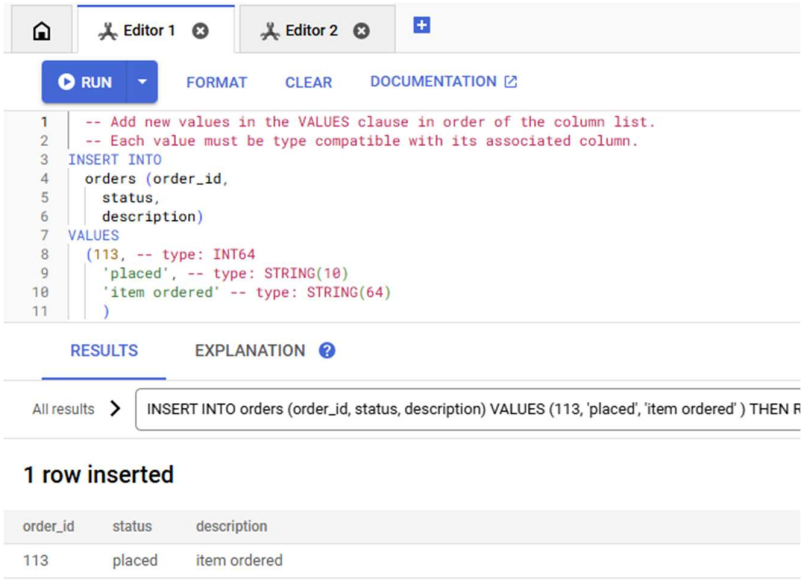
```
.withWriteDisposition(BigQueryIO.Write.WriteDisposition.WRITE_APPEND
)
.withCreateDisposition(BigQueryIO.Write.CreateDisposition.CREATE_IF_NE
EDED));
```

d. Spanner Table:

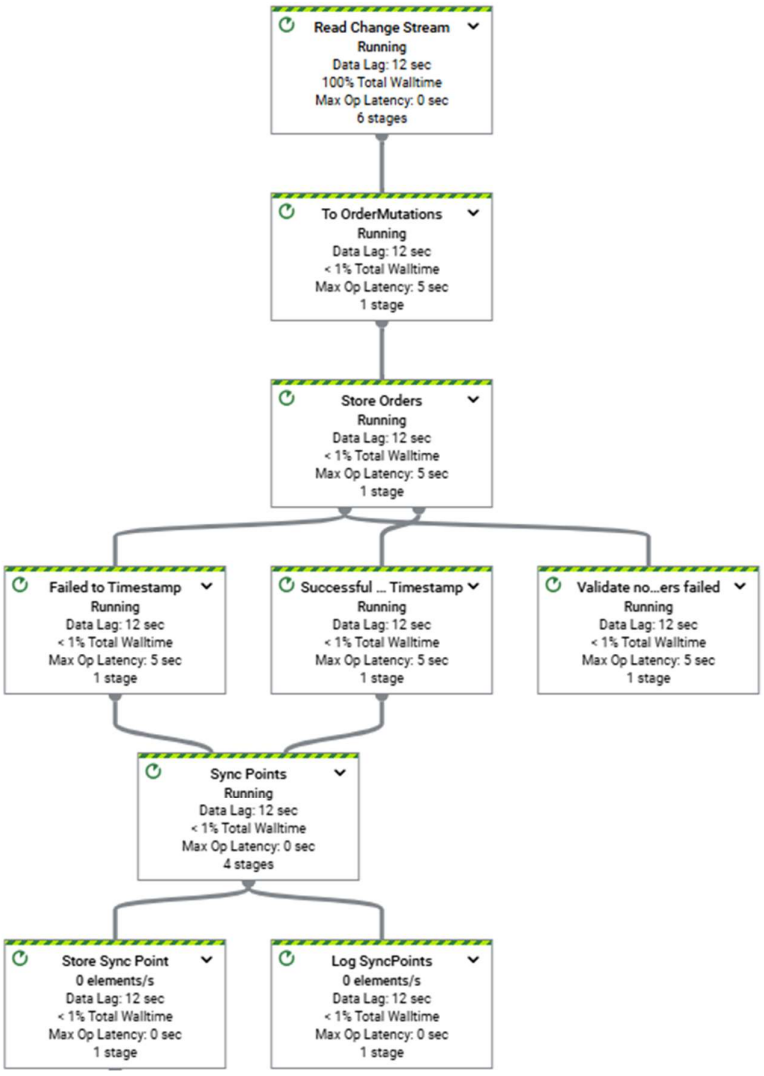
Table DDL creation step



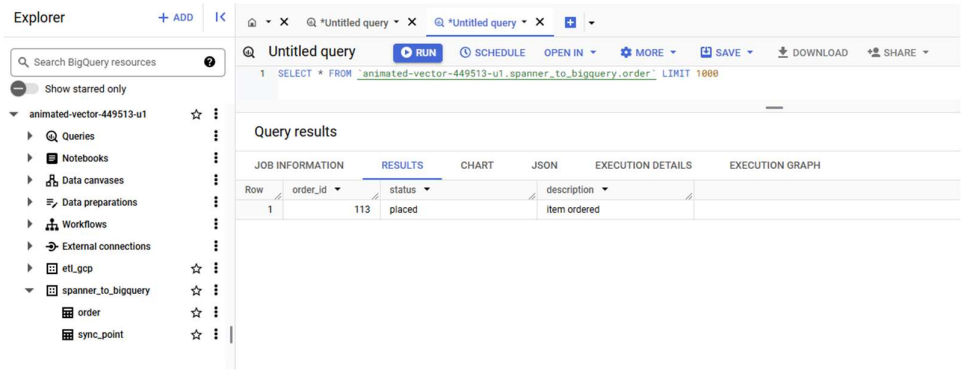
Once dataflow started, then inserted one sample record in Spanner



e. After submitting the dataflow, we can see the following graph:



f. After submitting the dataflow, we can record the inserted step-d in to following Bigquery table:



3.4 Fault Tolerance and Checkpointing

Fault tolerance is a crucial aspect of the CDC pipeline to ensure resiliency against failures. The pipeline incorporates: [4, 5].

a. **Checkpointing and State Management:** The pipeline periodically saves the processing state, enabling it to resume from the last known successful point in case of failures.
b. **Automatic Recovery Mechanisms:** The system detects transient failures and automatically retries processing without manual intervention.
c. **Dead-Letter Queue (DLQ) for Failed Records:** Unrecoverable records are logged separately for debugging and reprocessing.
d. **Self-Monitoring and Alerts:** The pipeline includes built-in monitoring to detect anomalies and trigger alerts for operational visibility.

To enable checkpointing, the following command is applied within the Dataflow pipeline:

```
pipeline.apply("EnableCheckpointing", EnableCheckpointing.create());
```

By integrating these failure-handling strategies, the system ensures uninterrupted real-time data synchronization and enhances data reliability for enterprise analytics use cases.

3.5 I Enhancements Over Google's Baseline Solution

While Google's foundational solution for Spanner to BigQuery [1] Change Data Capture (CDC) provides a strong starting point, our enhanced framework incorporates several improvements to increase data type compatibility, schema flexibility, fault tolerance, and error handling. These enhancements ensure better adaptability to enterprise-scale deployments, where varied data formats, evolving schemas, and real-time processing demands must be addressed

3.5.1 Support for Multiple Data Types

The original implementation supports only basic data types, limiting its applicability. Our solution introduces:

- Compatibility with Integer, Float, Date, Datetime, and Array data types.
- Dynamic type mapping between Spanner and BigQuery [3] to prevent inconsistencies.
- Improved handling of nested structures and complex data types for enhanced query performance.

3.5.2 Null Value Handling

Proper null value management is essential for maintaining data integrity. Our improvements include:

- Encoding and restoring null values to prevent inconsistencies across systems.
- Implementing null-safe transformations to ensure accurate interpretation in BigQuery [2, 3].

3.5.3 Schema Evolution

Frequent schema changes in enterprise environments require a flexible and automated adaptation mechanism. Our solution introduces:

- Automated schema detection to identify modifications in Spanner tables.
- Dynamic schema evolution that updates BigQuery without manual intervention.
- Backward compatibility supports ensure historical data remains accessible.

3.5.4 Enhanced Error Logging and Reprocessing

Enterprise pipelines demand robust error tracking to minimize data loss. Our improvements include:

- Capturing failed inserts using *getFailedStorageApiInserts* for better traceability.
- Isolating failed records in a dedicated error pipeline for targeted reprocessing.
- Automated retry mechanisms to recover from transient failures and minimize disruptions.

.

4. CHALLENGES AND SOLUTIONS

The following are some of the challenges and solutions.

a. **Latency Optimization:** Batching inserts and using streaming writes improve efficiency.
b. **Schema Evolution Handling:** Automatically adapts to Spanner schema changes to prevent pipeline failures.
c. **Data Loss Prevention:** Uses checkpointing and BigQuery upserts to maintain consistency.

5. PERFORMANCE METRICS AND COST

Spanner to BigQuery performance and cost comparison is shown in Tab-1.

Table 1. Spanner to BigQuery performance details - Realtime

Metric	Value
Latency	5s per update
Throughput	1M records per second

Table 2. Spanner to BigQuery tested for following data size

Table	Data Size	Spanner Nodes	Dataflow workers	Cost
Table-1	35 million records	40	15	$650
Table-2	169 million records	40	40	$650

6. CONCLUSION

This paper presents a scalable and fault-tolerant CDC framework for real-time data replication from Google Cloud Spanner to BigQuery. By leveraging Spanner Change Streams and Dataflow, the solution ensures low-latency, high-throughput synchronization while supporting schema evolution and robust error handling. Enhancements such as improved data type compatibility and automated schema adaptation make the framework more resilient for enterprise-scale deployments.

Performance benchmarks confirm sub-second latency and high throughput, enabling real-time analytics with minimal operational overhead. Future work will focus on optimizing costs, reducing latency, and integrating advanced monitoring for enhanced data governance. This framework provides a strong foundation for modern cloud-native data warehousing solutions.

7. REFERENCES

[1] K. Sato, "An inside look at Google BigQuery," Google Cloud, 2012. [Online]. Available: https://docs.deistercloud.com/mediaContent/Technology.50/Google/media/BigQueryTechnicalWP.pdf.

[2] Google, "Google BigQuery – Real-time big data analytics in the cloud," Google Cloud, 2012. [Online]. Available: https://cloud.google.com/files/BigQuery.pdf.

[3] M. H. Ali, M. S. Hosain, and M. A. Hossain, "Big data analysis using BigQuery on cloud computing platform," *Aust. J. Eng. Innov. Technol.*, vol. 1, no. 2, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:241687923.

[4] J. J. Levandoski, G. Casto, M. Deng, R. Desai, P. Edara, T. Hottelier, A. Hormati, A. Johnson, J. Johnson, D. Kurzyniec, S. McVeety, P. Ramanathan, G. Saxena, V. Shanmugan, and Y. Volobuev, "BigLake: BigQuery's evolution toward a multi-cloud lakehouse," *Companion of the 2024 International Conference on Management of Data*, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:269987544.

[5] E. Brewer, "Spanner, TrueTime & the CAP theorem," Google, Feb. 14, 2017. [Online]. Available: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45855.pdf.

[6] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, W. C. Hsieh, S. Kanthak, E. Kogan, H. Li, A. Lloyd, S. Melnik, D. Mwaura, D. Nagle, S. Quinlan, R. Rao, L. Rolig, Y. Saito, M. Szymaniak, C. Taylor, R. Wang, and D. Woodford, "Spanner: Google's globally distributed database," *ACM Trans. Comput. Syst.*, vol. 31, no. 3, pp. 1–22, Aug. 2013, doi: 10.1145/2491245.

[7] B. Bachina, "Unlocking data processing efficiency: A comprehensive guide to GCP Dataflow," *J. Sci. Eng. Res.*, vol. 8, no. 9, pp. 242–248, 2021. [Online]. Available: https://jsaer.com/download/vol-8-iss-9-2021/JSAER2021-8-9-242-248.pdf.

[8] T. Akidau, R. Bradshaw, C. Chambers, S. Chernyak, R. J. Fernandez-Moctezuma, R. Lax, S. McVeety, D. Mills, F. Perry, E. Schmidt, and S. Whittle, "The Dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing," *Proc. VLDB Endowment*, vol. 8, no. 12, pp. 1792–1803, Aug. 2015. doi: 10.14778/2824032.2824076.

[9] Google Cloud Platform, *Dataflow BigQuery Change Data Capture*, GitHub repository, 2024. [Online]. Available: https://github.com/GoogleCloudPlatform/cloud-solutions/tree/main/projects/dataflow-bigquery-change-data-capture.