Cost Optimization in Google Big Query: Best Practices for Query Efficiency and Storage Management

Tushar Gohil
Assistant Professor
Information Technology Department,
Sarvajanik College of Engineering and Technology, Surat, India
tushar.gohil@scet.ac.in

Abstract—Google Big Query's serverless architecture delivers high-speed analytics at scale, yet its pay-as-you-go pricing requires diligent management to prevent escalating costs. This paper investigates strategies to optimize query execution and storage, focusing on efficient design principles like selective column retrieval and early filtering. By implementing advanced data organization techniques such as partitioning and clustering, organizations can significantly minimize data processed.

Index Terms—Google BigQuery, Cost Optimization, Query Efficiency, Storage Management, Cloud Computing, Data Warehousing, Big Data, Partitioning and Clustering, Cloud Cost Management, Data Analytics

I. INTRODUCTION

As organizations increasingly rely on Google Big Query for petabyte-scale analytics, managing its performance-linked costs becomes a critical operational challenge [1], [3]. This paper provides a comprehensive framework for optimizing BigQuery expenses through a dual focus on query efficiency and storage management. By implementing technical best practices—such as table partitioning, clustering, selective schema design, and approximate aggregation—organizations can significantly reduce data scan volumes and computational overhead. Furthermore, we examine the strategic use of Big Query's tiered storage, automated lifecycle data policies, and caching mechanisms to ensure cost-proportionality. Beyond technical execution, the study emphasizes the necessity of proactive governance, utilizing real-time monitoring tools and cost-analysis dashboards to identify inefficiencies [3], [8]. By integrating these optimization strategies, data

professionals and decision-makers can leverage Big Query's high-speed analytical power while maintaining sustainable, cost-effective data architecture in a cloud-centric business environment.

II. BASELINE DATA STRUCTURE

To demonstrate optimization, this study utilizes the GA Source Table, derived from Google Analytics 4 (GA4) public sample dataset (bq-public-data.ga4_obfuscated_sample_ecommerce. events_*) [1], [4]. This dataset contains approximately 3.4 GB of obfuscated e-commerce event data, including timestamps, event names, and user IDs. The table was initially created in an unoptimized state—lacking partitioning or clustering—using a standard CREATE TABLE AS SELECT * statement. In this configuration, any query (even those filtering for specific dates or events) triggers a full table scan. This baseline represents a common but inefficient data structure, serving as a clear benchmark to measure the cost-saving impact of the partitioning and clustering techniques discussed in subsequent sections.

III. UNDERSTANDING QUERY COSTS IN BIGQUERY

Big Query's cost-effectiveness depends on understanding its pay-as-you-go pricing model, where expenses are primarily driven by the volume of data processed during query execution. Costs are determined by the total bytes scanned. Because BigQuery utilizes columnar storage, it only bills for the specific columns accessed by a query. Typically, pricing is calculated per terabyte (TB) processed [1], [3] —for instance, a query scanning 1 TB at a $5/TB rate results in a $5 charge.

IV. COMMON DRIVERS OF HIGH EXPENSES

Inefficient query practices lead to excessive data processing and avoidable costs. Based on the baseline GASourceTable (3.34 GB), common pitfalls include [4],[6]:

- Full Table Scans: Queries that scan the entire table instead of utilizing partitions. For example, filtering by a specific date on an unpartitioned table still results in the full 3.34 GB billed.
- Overuse of SELECT *: Retrieving every column when only a subset is needed. Executing SELECT * to find a specific event name forces a scan of all 3.34 GB.
- Unoptimized JOINs: Operations that process large datasets unnecessarily. An unoptimized JOIN on the baseline table can process the entire 3.34 GB volume.
- Lack of Filtering: Queries that fail to use effective conditions to limit the data scan.

V. PARTITIONING IN BIGQUERY

BigQuery supports various partitioning methods tailored to specific data structures [1],[4]:

- Time-Based Partitioning: Ideal for time-series data like logs or events, this method divides tables by DATE, TIMESTAMP, or DATETIME columns.
  - DATE Partitioning: Tables are partitioned by a specific date column (e.g., event_date).

- TIMESTAMP/DATETIME Partitioning: Tables are segmented by timestamp or datetime values.
- Integer-Range Partitioning: This divides a table based on a numeric column, such as user_id or order_id, and is useful for datasets frequently filtered by numeric ranges.

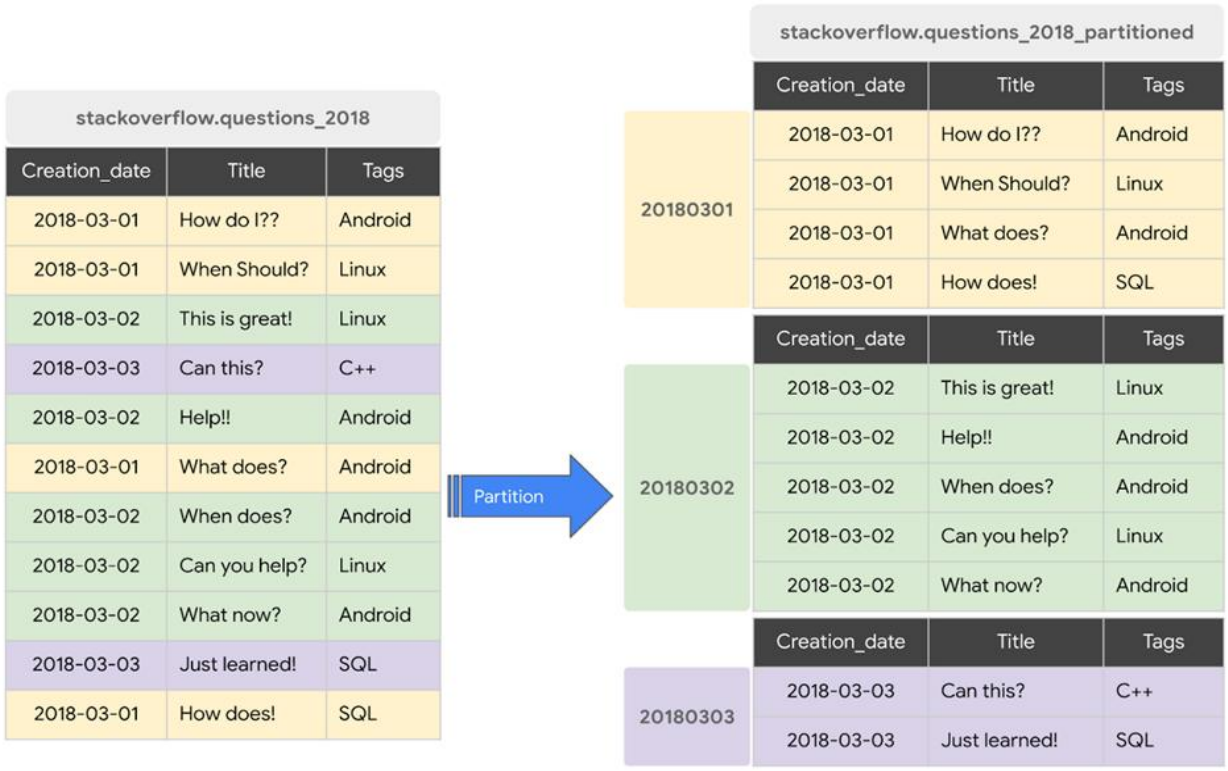

FIGURE 1 *PARTITIONING BEST PRACTICES*

To maximize the benefits of partitioning, organizations should adhere to the following guidelines:

- Select the Right Column:

Choose a column frequently used in query filters, such as event date for time-series data.

- Avoid Over-Partitioning:

Aim for partitions that are at least 1 GB in size to maintain optimal performance.

- Combine with Clustering:

Use partitioning alongside clustering on additional columns for even greater efficiency.

- Use Partition Expiration:

Automatically delete old partitions to reduce long-term storage costs for temporary or log data [1],[8].

VI. CLUSTERING IN BIGQUERY

Clustering optimizes BigQuery efficiency by strategically organizing table data to group related rows based on specific columns, known as clustering keys. This organization allows BigQuery to perform data skipping, bypassing irrelevant data blocks during query execution [4],[5].

Clustered Tables

Table 1
Not clustered

Order_Date	Country	Status
2022-08-02	US	Shipped
2022-08-04	JP	Shipped
2022-08-05	UK	Canceled
2022-08-06	KE	Shipped
2022-08-02	KE	Canceled
2022-08-05	US	Processing
2022-08-04	JP	Processing
2022-08-04	KE	Shipped
2022-08-06	UK	Canceled
2022-08-02	UK	Processing
2022-08-05	JP	Canceled
2022-08-06	UK	Processing
2022-08-05	US	Shipped
2022-08-06	JP	Processing
2022-08-02	KE	Shipped
2022-08-04	US	Shipped

Table 2
Clustered by country

Order_Date	Country	Status
2022-08-04	JP	Shipped
2022-08-04	JP	Processing
2022-08-05	JP	Canceled
2022-08-06	JP	Processing
2022-08-06	KE	Shipped
2022-08-02	KE	Canceled
2022-08-04	KE	Shipped
2022-08-02	KE	Shipped
2022-08-05	UK	Canceled
2022-08-06	UK	Canceled
2022-08-02	UK	Processing
2022-08-06	UK	Processing
2022-08-02	US	Shipped
2022-08-05	US	Processing
2022-08-05	US	Shipped
2022-08-04	US	Shipped

Table 3
Clustered by country and status

Order_Date	Country	Status
2022-08-05	JP	Canceled
2022-08-04	JP	Processing
2022-08-06	JP	Processing
2022-08-04	JP	Shipped
2022-08-02	KE	Canceled
2022-08-06	KE	Shipped
2022-08-04	KE	Shipped
2022-08-02	KE	Shipped
2022-08-05	UK	Canceled
2022-08-06	UK	Canceled
2022-08-02	UK	Processing
2022-08-06	UK	Processing
2022-08-05	US	Processing
2022-08-02	US	Shipped
2022-08-05	US	Shipped
2022-08-04	US	Shipped

FIGURE 2

Table clustering sorts data within a table or partition according to the specified keys. When a query filters on these keys, BigQuery narrows the search to specific blocks, leading to:

- Cost Savings: Expenses are lowered by processing fewer bytes.
- Faster Execution: Performance is accelerated through reduced data scans.

Best Practices in Clustering

- Prioritize Frequent Filters:

Use columns often found in WHERE, GROUP BY, or JOIN clauses, such as user id.

- Limit Key Count:

While BigQuery supports up to four keys, 1–2 is typically sufficient; over-clustering can dilute efficiency.

- Pairing for Impact:

Maximum efficiency is achieved by combining partitioning with clustering.

- Target Large Tables:

Clustering is most effective on sizable datasets where data skipping results in significant scan reductions [5], [7].

VII. CONCLUSION

Cost optimization in Google BigQuery is essential for organizations leveraging cloud-based analytics at scale. By balancing high-speed performance with budget-conscious operations, businesses can maintain a sustainable data ecosystem. This study has demonstrated that a multifaceted approach—blending technical efficiency with strategic oversight—is necessary to prevent spiraling expenses.

ACKNOWLEDGMENT

I express my gratitude to the Google Cloud Platform for providing robust infrastructure and tools that facilitated the efficient processing and analysis of large datasets for this research. Special thanks are also extended to Data Talks Club's Data Engineering Zoom camp. Their comprehensive curriculum, hands-on projects, and vibrant community support were instrumental in sharpening the data engineering principles and methodologies employed throughout this study.

REFERENCES

[1] BigQuery Documentation, "Introduction to partitioned tables," Google Cloud, 2025.
[2] "Adaptive performance and cost optimization strategies in cloud data warehousing: A comprehensive theoretical and applied synthesis," Int. J. Data Sci. Mach. Learn., vol. 5, no. 2, pp. 324–333, 2025.
[3] OWOX BI, "BigQuery pricing 2025: Forecast and manage costs," OWOX Blog, Aug. 5, 2024.
[4] BigQuery Team, "Optimizing BigQuery for cost and performance," Google Cloud Whitepaper, 2023.
[5] R. Adhikari and C. Kambhampati, "Cloud data warehousing: Architecture, techniques, and challenges," J. Cloud Comput., vol. 12, no. 1, pp. 45–68, 2023.
[6] P. Kodakandla, "Refactoring petabyte-scale data warehouses for performance and cloud optimization," Int. Res. J. Modernization Eng. Technol. Sci., vol. 5, no. 3, 2023.
[7] L. Garcia and M. Soto, "Storage optimization and compression modelling in cloud analytics platforms," Int. J. Big Data Syst., vol. 7, no. 1, pp. 55–70, 2023.
[8] Google Cloud, "An insider's guide to BigQuery cost optimization," Google Cloud Whitepapers, 2024.