Snowflake vs Google BigQuery: A Comprehensive Guide

Understanding the key differences and how to migrate from Snowflake to BigQuery

Table of Contents

Executive summary

In 2025 and beyond, more enterprises are adopting digital transformation in a bid to gain more market competitiveness and innovation. In this context, data modernization is a "core" business necessity - and not just another "to-do" process.

The recent numbers in data modernization look encouraging:

- 52% of companies have migrated their IT infrastructure to the cloud.
- 57% of technology decision-makers are accelerating their cloud migration.
- The public cloud market is projected to grow to $825 billion in 2025.

With the growing demand for data-driven decision-making, industry leaders are recognizing the need to integrate data management. By streamlining data integration with AI and data warehousing, modern data management platforms are creating the foundation for their modernization strategy.

Among the latest trends in data modernization, enterprises are choosing operational data warehouses, which provide them real-time data access for faster decision-making. As compared to "traditional" warehouses, operational warehouses can handle massive data volumes and high-speed data processing.

Cloud-based operational warehouses like Google BigQuery offer a host of business benefits including:

1. Scalability

With Google BigQuery, enterprises in their modernization efforts can overcome scalability-related limitations - with its cloud infrastructure that automatically scales according to growth in data volumes.

2. Centralized data storage

Enterprises can also centralize and consolidate their data on BigQuery's unified environment, thus making it easier to perform data analytics and develop AI models.

3. Integration with other Google Cloud services

Google BigQuery is also part of the Google Cloud ecosystem and provides built-in integration with:

- Google Cloud AI and machine learning tools
- Google Data Studio for data visualization
- Google Cloud Functions for serverless computing
- Google Cloud Pub/Sub for real-time data streaming

4. Data security

BigQuery also elevates data security and governance with its robust security framework comprising the following features:

- Data encryption (during storage and in transit)
- Customer-managed encryption keys
- Finely-tuned access control
- Cloud-powered data loss prevention (DLP)

Choosing Onix as your trusted cloud migration partner

When it comes to large-scale migration projects, Onix has a successful track record and expertise in cloud migration and modernization. Here's a case study of how Onix enabled a national retailer to migrate their Snowflake and MS Azure workloads to Google Cloud and BigQuery for real-time analytics.

Evaluating the importance of data migration and modernization

Cloud platforms like Snowflake and Google BigQuery both offer a reliable platform for data migration and modernization. With these cloud platforms, organizations can avail benefits in the form of improved scalability, lower costs, and better performance.

By separating data storage from their computing resources, Snowflake and BigQuery are built for scalability. With its centralized data storage, Snowflake can store data from external platforms like Google Cloud and MS Azure. On the other hand, Google BigQuery features a serverless architecture, which can automatically scale its resources depending on its computing demands. Here are some other critical aspects of both Snowflake and BigQuery that matter for data modernization:

Query performance

For improved performance, enterprises need to optimize their query processing speed, which can lead to faster decision-making. Snowflake provides optimum performance by running multiple workloads at the same time - while keeping them isolated from each other.

On the other hand, Google BigQuery has an in-built caching mechanism, which reduces the querying time. Additionally, its serverless architecture can allocate more computing resources during peak querying demands.

Data security

On the cloud, data security is of paramount importance to any enterprise implementing cloud modernization. Both Snowflake and BigQuery (BG) offer strong data encryption in compliance with AES-256 standard. On its virtual private version, Snowflake limits the number of subscribers to virtual private networks (VPNs) to ensure network security.

With Google BigQuery, enterprises can implement comprehensive data governance for managing data security and ensuring that data access is aligned with the organization's data policies and regulations. BigQuery implements data governance through the following 3 categories:

1. Access control

BigQuery provides access control with the following features:

- Identity and access management (IAM) defines access control to valuable BigQuery resources like projects, datasets, and database tables.
- Column-level (and row-level) access control restricts user access to specific table columns and rows respectively, based on the user's attributes or data values.
- Data transfer management enables the creation of sensitive perimeters around Google Cloud's resources, which can define the access control to these resources.

2. Data stewardship

Data stewardship can protect sensitive data through proven techniques like data masking and encrypting during data querying or transit. For instance, BigQuery can encrypt all stationary and in-transit data.

3. Data quality

With data quality management, BigQuery ensures that the data is accurate, complete, and consistent. Using data lineage, BigQuery can provide insights into the data source and how it has changed over time.

Besides, BigQuery's serverless architecture decouples its data storage from computing, thus empowering enterprises to have more control over their data. This decoupled framework enables BigQuery users to scale their computing resources whenever required.

In addition to BigQuery's data governance and security, it can seamlessly manage common data-related challenges through:

- Faster data ingestion and transfer from SaaS applications like Google Ads and data warehouses like Teradata.
- Support for real-time data analytics by running analytical queries on massive datasets (ranging from terabytes to petabytes) in a few seconds or minutes.

As an integral part of the Google Cloud Platform (GCP) environment, BigQuery is deeply integrated with GCP's data processing and analytics capabilities. This enables customers to implement a modern data warehouse on the cloud. Depending on the stage of the data lifecycle, GCP delivers a variety of data-related services, including:

- Data ingestion
  - Google Kubernetes Engine (GKE)
  - Compute Engine
  - Google Cloud Pub/Sub messaging
- Data storage on the cloud
  - BigQuery
  - Cloud SQL
- Data processing and analysis
  - Dataflow
  - BigQuery
  - Dataproc
- Data exploration and visualization
  - Looker
  - BigQuery BI Engine
  - Datalab

Next, let's understand how the structural differences of these 2 platforms pose a challenge in Snowflake to BigQuery migration.

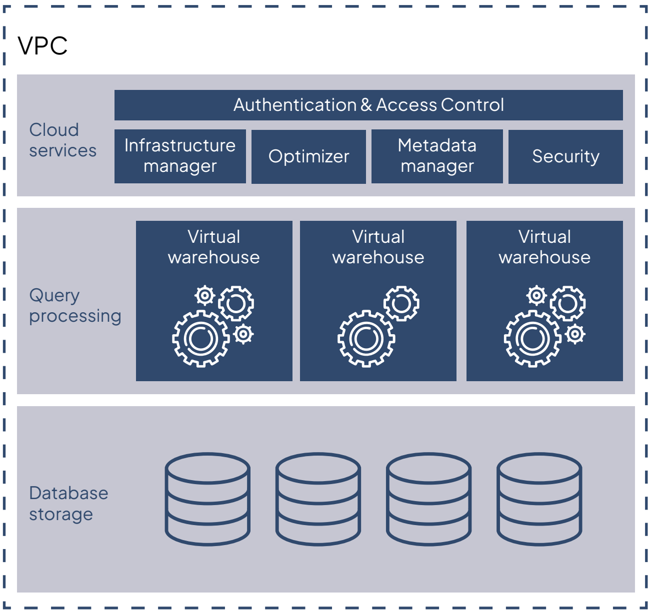
Areas to consider when migrating from Snowflake to BigQuery

Enterprises using Snowflake must understand the challenges of migrating this data warehouse to BigQuery – and how to address each of these challenges. This section highlights the differences in the following technical areas:

- Architecture
- Object mapping
- Data types

Snowflake & BigQuery – Architecture

While both Snowflake and BigQuery are essentially data warehouses, their architectures do have some key differences.



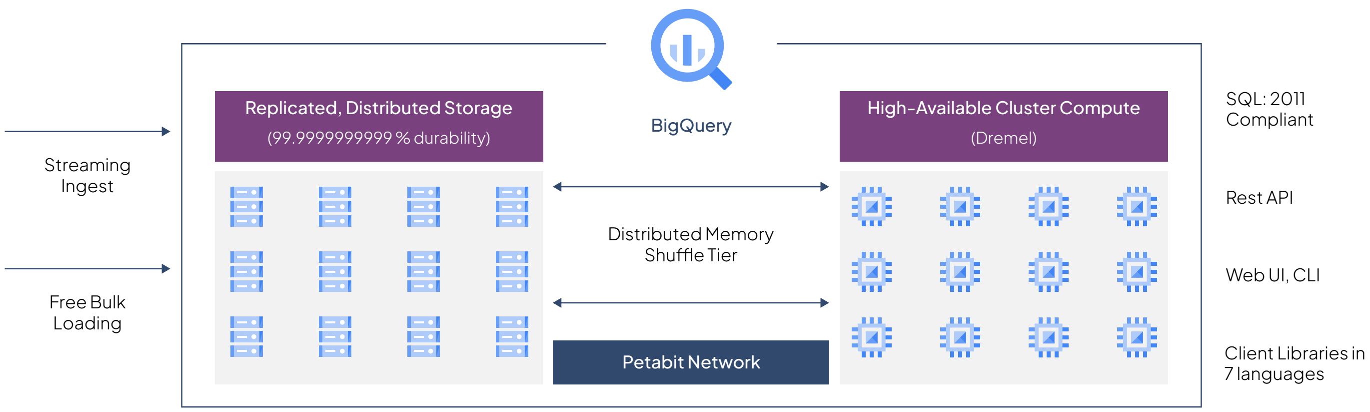
Source

The Snowflake architecture is a hybrid combination of the following database architectures:

- Shared disk architecture manages the database queries with dedicated computing clusters.
- Shared-nothing architecture manages the data using a separate cloud object storage service.

In Snowflake, each cluster in the network manages the cached portion of the dataset to improve query performance. On the other hand, Google BigQuery's architecture does not use a traditional node-based cloud warehouse system – or an on-premises massively parallel processing (MPP) system. It completely decouples data storage from its computing resources.

With its decoupled architecture, BigQuery can dedicate more computing power to database queries, thus enabling faster processing. Overall, BigQuery is powered by many data centers with massive storage capacity and networking bandwidth, which can meet the requirements of concurrent users.



Source

Snowflake & BigQuery – Object hierarchy and mapping

Snowflake and BigQuery use the same hierarchical object structure but with different names. With Snowflake, the top of the object hierarchy is referred to as "database," while BigQuery refers to it as a "project."

Here's how each of these data warehouses terms their object hierarchical structure:

- Snowflake - <database>/<schema>/<tables/views>
- BigQuery - <project>/<dataset>/<tables/views>

The following tables list the entire object hierarchy and mapping in Snowflake and BigQuery:

Snowflake	BigQuery
• Database	• Project
• Schema	• Dataset
• Session-based temporary table	• Anonymous or temporary table
• Secure views	• Authorized views
• Virtual warehouse	• Reservation
• Micro-partitioning	• Partitioning
• Materialized view	• Materialized view
• Clustering	• Clustering
• Primary keys – not enforced in standard tables	• Primary keys – not enforced

Snowflake & BigQuery – Data types

Snowflake and BigQuery have equivalent data types that serve the same purpose. However, some data type mismatches need to be considered for a smoother migration:

1. BigQuery supports NUMERIC and BIG NUMERIC data types, which are not compatible with the NUMBER/ NUMERIC data type in Snowflake.
2. BigQuery's BIG NUMERIC data type has a precision count of 76.76 and a scale of 38. In Snowflake, the NUMBER data type has precision count of 38 digits and 37 digits of scale (which can be more than 0.)
3. Snowflake defines the FLOAT data type as "NaN > X" - where X is any FLOAT value apart from NaN. Alternatively, Google BigQuery defines the FLOAT data type as "NaN < X" - where X is any FLOAT value apart from NaN.
4. In Snowflake, the DOUBLE data type is synonymous with the FLOAT data type - but is incorrectly displayed as FLOAT. Instead, it must be represented as DOUBLE.
5. The VARCHAR data type in Snowflake has a maximum data length of 16MB, which is the default length (unless otherwise specified). In BigQuery, the corresponding data type is STRING - with a variable length of UTF-8 encoded Unicode. It has a maximum length of 16,000 characters.

1. The CHAR data type in Snowflake has a maximum length of 1.
2. BigQuery does not have an equivalent data type of VARIANT in Snowflake.
3. The BOOL data type in BigQuery can accept values of TRUE and FALSE. The corresponding BOOL data type in Snowflake can accept values of TRUE, FALSE, and NULL.
4. The DATE data type in Snowflake accepts most common date formats. The DATE type in BigQuery only accepts dates in the 'YYYY-[M]M-[D]D' format.
5. The TIME data type in Snowflake supports up to 9 nanoseconds of precision, whereas the TIME data type in BigQuery supports up to 6 nanoseconds of precision.
6. The TIMESTAMP_LTZ data type in Snowflake defines local-time-zone along with the date and time. Besides, this data type can define multiple-time-zone along with the date and time. This means that a single TIMESTAMP_TZ column can hold timestamp values with different time zones. Alternatively, BigQuery doesn't have a datatype that supports a non-utc timezone.
7. The OBJECT data type in Snowflake does not support explicitly-typed values. Values are of the VARIANT type.
8. Snowflake supports the XML data format with VARIANT data types. BigQuery does not have any equivalent data type.
9. Google BigQuery has the "DATETIME" and "GEOGRAPHY" data types - with no equivalent type in Snowflake

Here's how BigQuery users can overcome these technical challenges:

Problem	Solution
Mismatch between the NUMERIC/ NUMBER data type in Snowflake and the NUMERIC data type in BigQuery	In BigQuery, use BIG NUMERIC as the equivalent data type if the precision and scale don't align with that of Snowflake. For the Snowflake's NUMERIC columns (without any specified precision and scale), use BIG NUMERIC as the default type in BigQuery - or scan the column data for the maximum length.
No equivalent of Snowflake's VARIANT data type	If the VARIANT uses a single data type, define the data types in BigQuery as JSON, STRING, or INT (or NUMERIC). If the VARIANT uses multiple data types, define the data type in BigQuery as STRING.
No equivalent of Snowflake's VARIANT data type with XMLs.	BigQuery does not support XML data type. For every "Variant XML" column in Snowflake, create a "STRING" column in BigQuery with XML data and an equivalent JSON data type.

Next, let's understand the best strategy to migrate the Snowflake data warehouse to Google BigQuery.

How to migrate from Snowflake to BigQuery

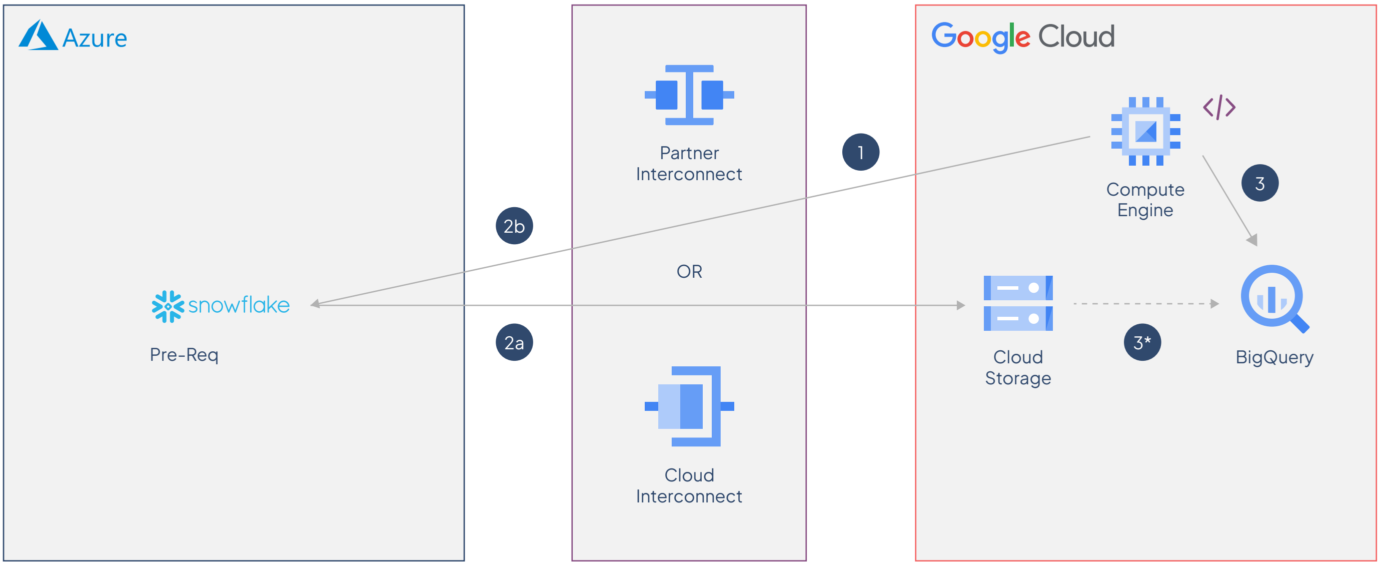
Enterprises can migrate incremental data from Snowflake to BigQuery by using the “ETL pipeline conversion to BigQuery” tool (or equivalent). However, for a comprehensive migration, enterprises must also migrate historical data accumulated in the ETL pipelines of the existing system. This can be performed using any of the following methods:

1. **Unloading data into the Google cloud storage.**
   If you have a Google cloud account, you can use the “COPY INTO GCS" to copy data from the Snowflake database table to Google's cloud storage (GCS) bucket.

2. **Using the Snowflake JDBC driver.**
   With this method, you can pull Snowflake data into GCP-enabled systems like virtual machines and Dataflow.

3. **Using Apache Sqoop.**
   This method essentially pulls Snowflake data from Dataflow or any Hadoop installations.

Among these methods, the "COPY INTO GCS" method is the fastest and most efficient mode of extracting historical data from Snowflake into GCS. Additionally, you can automate the process of migrating historical backfills from Snowflake to BigQuery with a data migration framework. Here’s an example:



Here are the necessary steps:

1. Running in the Compute Engine, the automated framework initiates the "Copy into GCS" command for the selected data from the database table.
2. The command copies the selected data into the GCS location – and sends an acknowledgment back to the migration framework.
3. Next, the framework pushes the data to BigQuery – and performs the data cleanup in GCS after successfully copying the data to the table.

Data migration challenges & considerations

Here are some challenges – and solutions to consider - during data migration from Snowflake to BigQuery:

1. **Challenge#1 –**
   Maintaining referential integrity between master and child tables – when the child table is migrated after the master table is already migrated and live on the target warehouse.

   Why does this happen? Parent-child tables with surrogate keys (which are system-generated keys) can pose a challenge during migration. This is because the surrogate key is not directly mapped to the source keys – thus breaking down the parent-child table relationship.

   Here's the solution:

   - Extract the natural keys from the Snowflake master table along with the child table data.
   - Load these keys in a temporary BigQuery database table.
   - Update the surrogate keys in the child table.

2. **Challenge#2 –**
   While loading large datasets into BigQuery , some database queries may exceed the 6–hour execution limit, thus leading to load failures and incomplete data migrations.
   Why does this happen? The 6–hour query execution limit is applicable only to individual queries – including those used for data loading and transformation. Data loading and processing are both time and resource-intensive for exceptionally massive tables and high-volume data sources.

   Here's the solution:

   Divide the data extraction into smaller chunks of data before loading it in BigQuery. Load segmentation can also help in completing data loading within the permitted time

Data testing and validation

In addition to migration challenges, enterprises can face challenges in data accuracy and referential integrity. Here are some of the reasons for this:

1. **Multi-row updates**
   By using UPDATE/ MERGE commands, BigQuery can match one source row for each target row. Snowflake supports non-deterministic updates that can perform multi-row updates.

2. **Un-nesting approach for JSON**
   This is applicable for JSON feeds during un-nesting in BigQuery. Snowflake can un-nest JSON columns internally by converting them into a JSON array. This is not possible in BigQuery.

3. **Implicit String to Boolean casting**
   Snowflake does not support the implicit casting of a String variable to Boolean. BigQuery only supports Integer and String (with value "true") to be cast to Boolean.

4. **Concurrent transactions**
   Concurrent transactions fail in BigQuery, while Snowflake adds them to a queue (with a lock).

5. **Time travel**
   Snowflake provides access to historical data at any point within a specified period (from 0 to 90 days). BigQuery uses time travel to access data within the previous 7 days. For historical data beyond 7 days, BigQuery supports exporting regularly scheduled snapshots.

Post-migration, enterprises can check for data fidelity with the right testing framework and validation tool. As an automated data validation tool, Onix's Pelican is recommended to perform this task.

Conclusion

Despite all the workarounds, it's challenging for enterprises to migrate from Snowflake to BigQuery without the support of a cloud migration partner. Here are some additional feature gaps in Snowflake and BigQuery that can delay the migration process:

1. BigQuery does not support "change data capture" (CDC) for its database tables using a stream object.
   Onix's solution: Develop a custom framework to fetch the incremental data. This can later be replaced with the "Time series functions" when it's made available.
2. For capturing table streams for continuous ELT workflows, Snowflake users can rely on orchestration and scheduling. BigQuery does not support any function for continuous workflows. Onix's solution: Use Airflow for scheduling and orchestration – or opt for continuous queries.
3. For stored procedures, Snowflake supports programming languages like Java, Python, and Javascript. BigQuery mainly supports SQL and PySpark.
   Onix's solution: Stored procedures in Snowflake (without SQL code) need to be rewritten to SQL code in BigQuery.
4. Snowflake supports XML handling for loading and querying database tables, while BigQuery does not support XML as a data type – nor provide any way to query XML data in its database tables.
   Onix's solution: Convert XML to JSON by:
   - Converting in-flight data using Dataflow or any computing tool.
   - Converting data (post ingestion) by using UDFs.

As an award-winning Google Cloud partner, Onix can address your specific migration-related challenges and minimize business disruption during this transition.

With our expertise, you can streamline your cloud migration to Google Cloud and BigQuery. Book a free product demo with our cloud migration experts.

onix

onixnet.com

connect@onixnet.com

800.664.9638

Get in touch

Follow us: