



HEVO

Best Practices for High- Performance ETL To BigQuery



Table of Contents

BigQuery Introduction	1
GCS - Staging Area for BigQuery Upload	2
Nested & Repeated Data	3
Data Compression	4
Time Series Data & Table Partitioning	5
Streaming Insert	7
Bulk Updates	8
Transforming Data After Load (ELT)	9
Federated Tables for Adhoc Analysis	10
Access Control & Data Encryption	11
Character Encoding	12
Backup & Restore	12
Bonus - There is An Easier Way to Perform ETL!	13



BigQuery Introduction

In today's data-intensive ecosystem there is always a need for a reliable data warehouse. BigQuery - a fully managed cloud data warehouse for analytics from Google Cloud Platform (GCP), is one of the most popular cloud-based analytics solutions. Due to its unique architecture and seamless integration with other services from Google Cloud Platform, there are certain elements to be considered as best practices while migrating data to BigQuery. All the important points are discussed in the following sections.

- **GCS - Staging Area for BigQuery Upload**
- **Nested & Repeated data**
- **Data Compression**
- **Time Series Data & Table Partitioning**
- **Streaming Insert**
- **Bulk Updates**
- **Transforming Data After Load (ELT)**
- **Federated Tables for Adhoc Analysis**
- **Access Control & Data Encryption**
- **Character Encoding**
- **Backup & restore**

2 GCS - Staging Area for BigQuery Upload

Unless you are directly loading data from your local machine, before loading the data into BigQuery you have to upload data to GCS. To move data to GCS you have multiple options:

- [Gsutil](#) is a command line tool which can be used to upload data to GCS from different servers.
- If your data is present in any online data sources like AWS S3 you can use Storage Transfer Service from Google cloud. This service has options to schedule transfer jobs.

Other things to be noted while loading data to GCS:

- GCS bucket and BigQuery dataset should be in the same location with one exception - If the dataset is in the US multi-regional location, data can be loaded from GCS bucket in any regional or multi-regional location.
- Format supported to upload from GCS to BigQuery are - Comma-separated values (CSV), JSON (newline-delimited), Avro, Parquet, ORC, Cloud Datastore exports, Cloud Firestore exports.

3 Nested & Repeated Data

BigQuery performs best when the data is denormalized. Instead of keeping relations, denormalize the data and take advantage of nested and repeated fields. Nested and repeated fields are supported in Avro, Parquet, ORC, JSON (newline delimited) formats. STRUCT is the type that can be used to represent an object, which can be nested and ARRAY is the type to be used for repeated value. For example, the following row from a BigQuery table and "address" field is an array of a struct:

```
{
  "id": "1",
  "first_name": "John",
  "last_name": "Doe",
  "dob": "1968-01-22",
  "addresses": [
    {
      "status": "current",
      "address": "123 First Avenue",
      "city": "Seattle",
      "state": "WA",
      "zip": "11111",
      "numberOfYears": "1"
    },
    {
      "status": "previous",
      "address": "456 Main Street",
      "city": "Portland",
      "state": "OR",
      "zip": "22222",
      "numberOfYears": "5"
    }
  ]
}
```

Data Compression

Most of the time the data will be compressed before transfer. You should consider the below points while compressing data.

- The binary Avro is the most efficient format for loading compressed data.
- Parquet and ORC format are also good as they can be loaded in parallel.
- For CSV and JSON, BigQuery can load uncompressed files significantly faster than compressed files because uncompressed files can be read in parallel.

5

Time Series & Table Partitioning

Time series data is a generic term used to indicate a sequence of data points paired with timestamps. Common examples are clickstream events from a website or transactions from Point Of Sale machine. The velocity of this kind of data is much higher and volume increases over time. Partitioning is a common technique used to efficiently analyze time series data and BigQuery has good support for this with partitioned tables.

A partitioned table is a special BigQuery table that is divided into segments often called as partitions. It is important to partition bigger table for better maintainability and query performance. It also helps to control costs by reducing the amount of data read by a query.

BigQuery has mainly three options to partition a table:

- **Ingestion-time partitioned tables** - For these type of table BigQuery automatically loads data into daily, date-based partitions that reflect the data ingestion date. A pseudo column named `_PARTITIONTIME` will have this date information and can be used in queries.

- **Partitioned tables** - Most common type of partitioning which is based on TIMESTAMP or DATE column. Data is written to a partition based on the date value in that column. Queries can specify predicate filters based on this partitioning column to reduce the amount of data scanned.
 - ❖ You should use the date or timestamp column which is most frequently used in queries as partition column.
 - ❖ Partition column should also distribute data evenly across each partition. Make sure it has enough cardinality.
 - ❖ Also, note that the Maximum number of partitions per partitioned table is 4,000.
 - ❖ Legacy SQL is not supported for querying or for writing query results to partitioned tables.
- **Sharded Tables** - You can also think of shard tables using a time-based naming approach such as [PREFIX]_YYYYMMDD and use a UNION while selecting data.

Generally, Partitioned tables perform better than tables sharded by date. However, if you have any specific use-case to have multiple tables you can use sharded tables. Ingestion-time partitioned tables can be tricky if you are inserting data again as part of some bug fix. You can read a detailed comparison [here](#).



Streaming Insert

For inserting data into a BigQuery table in batch mode a Load Job will be created which will read data from the source and insert into the table (read more on the [Load Jobs](#)). Streaming data will enable us to query data without any delay of load job. Stream insert can be done to any BigQuery table using Cloud SDKs or other GCP services like Dataflow (Dataflow is an auto-scalable stream and batch data processing service from GCP - read [more](#)).

Following things to be noted while stream insert:

- Streaming data is available for the query after a few seconds of first stream insert in the table.
- It takes up to 90 minutes to become data available for copy and export.
- While streaming to a partitioned table, the value of `_PARTITIONTIME` pseudo column will be NULL while data is in the streaming buffer.
- While streaming to a table partitioned on a DATE or TIMESTAMP column, the value in that column should be between 1 year in the past and 6 months in the future. Data outside this range will be rejected.

7

Bulk Updates

BigQuery has quotas and limits for DML statements which is getting increased over time. As of now the limit of combined INSERT, UPDATE, DELETE, and MERGE statements per day per table is 1,000. Note that this is not the number of rows. This is the number of the statement and as you know, one single DML statement can affect millions of rows.

Now within this limit, you can run updates or merge statements affecting any number of rows. It will not affect any query performance, unlike many other analytical solutions.

8

Transforming Data After Load (ELT)

Sometimes it is really handy to transform data within BigQuery using SQL, which is often referred to as Extract Load Transfer (ELT). BigQuery supports both *INSERT INTO SELECT* and *CREATE TABLE AS SELECT* methods to data transfer across tables.

Example:

```
INSERT das.DetailedInve (product, quantity)
VALUES('countertop microwave',
(SELECT quantity FROM ds.DetailedInv
WHERE product = 'microwave'))

CREATE TABLE mydataset.top_words
AS SELECT corpus,ARRAY_AGG(STRUCT(word, word_count)) AS top_words
FROM bigquery-public-data.samples.shakespeare GROUP BY corpus;
```



Federated Table For Adhoc Analysis

You can directly query data stored in the location below from BigQuery which is called federated data sources or tables.

- Cloud BigTable
- GCS
- Google Drive

Read more on [Federated tables](#)

Things to be noted while using this option:

- Query performance might not be good as native BigQuery table.
- No consistency guaranteed in case of external data is changed while querying.
- Can't exports data from an external data source using BigQuery Job.
- Currently, Parquet or ORC format is not supported.
- The query result is not cached, unlike native BigQuery tables.

10

Access Control & Data Encryption


Data stored in BigQuery is encrypted by default and keys are managed by GCP. Alternatively, customer can manage key using Google KMS service.

To grant access to resources, BigQuery uses IAM (Identity and Access Management) to dataset level. Tables and views are child resources of datasets and inherit permission from the dataset. There are predefined roles like *bigquery.dataViewer* and *bigquery.dataEditor* or user can create custom roles. Check out the [documentation](#) to know more.

11

Character Encoding

Sometimes it will take some time to get the correct character encoding scheme while transferring data. Take notes of the points mentioned below as it will help you to get it correct in the first place.

- BigQuery expects all source data to be UTF-8 encoded with below exception
- If a CSV file with data encoded in ISO-8859-1 format, it should be specified and BigQuery will properly convert the data to UTF-8
- Delimiters should be encoded as ISO-8859-1
- Non-convertible characters will be replaced with Unicode replacement character: 

12

Backup & Restore

BigQuery addresses backup and disaster recovery at the service level. The user does not need to worry about it. Still, BigQuery is maintaining a complete 7-day history of changes against tables and allows to query a point-in-time snapshot of the table. Read more about the syntax of [querying snapshot](#).

There is An Easier Way To Perform ETL!

The detailed steps of ETL processing using Google BigQuery mentioned involves multiple complex stages and can be a cumbersome experience. If you want to load any data easily into Google BigQuery without any hassle, you can try out **Hevo**. Hevo automates the flow of data from various sources to Google BigQuery in real time and at zero data loss. In addition to migrating data, you can also build aggregates and joins on Google BigQuery to create materialized views that enable faster query processing.

Hevo integrates with a variety of data sources ranging from SQL, NoSQL, SaaS, File Storage Base, Webhooks, etc. with the click of a button.

[Sign up for a free trial](#) or [view a quick video](#) on how Hevo can make ETL easy.

About Author:

Faisal is currently working as a Technical Specialist and has worked on numerous Big Data analytics services from AWS like Redshift, Athena, EMR. Kinesis, QuickSight and Google Cloud like PubSub, Dataflow, and BigQuery. He has solid experience in open source Big Data frameworks like Hive, Presto and Spark for handling both batch and stream pipelines.

Published on: 21st February 2019

Looking for a simple and reliable way to bring
Data from Any Source to BigQuery?

TRY HEVO

SIGN UP FOR FREE TRIAL