**FinOps: Four Ways to Reduce Your BigQuery Storage Cost** fostering

**https://towardsdatascience.com/finops-four-ways-to-reduce-your-bigquery-storage-cost-82d99c47f139**

Don’t overlook the cloud storage cost

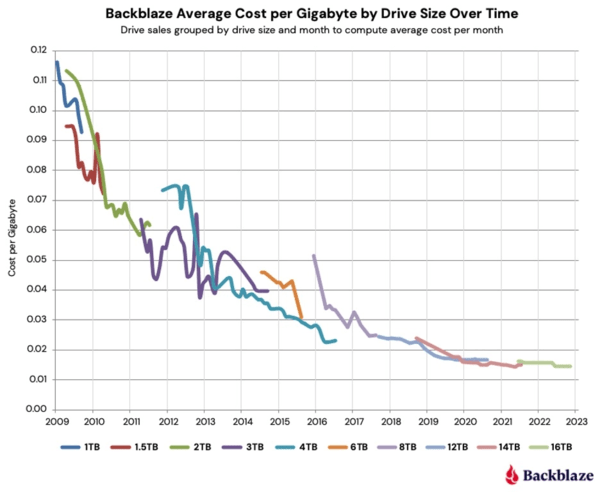


Photo by [Nathan Dumlao](https://unsplash.com/@nate_dumlao) on [Unsplash](https://unsplash.com/" \t "_blank)

With the current state of the economic situation, it’s more important than ever to maximize our cash on hand and develop a series of cost optimization strategies. The growing use of cloud services has brought not only many opportunities for the business but also the potential for management challenges that can lead to cost overruns and other issues.

FinOps, a newly introduced concept, is an evolving operational framework and cultural shift that allows organizations to get maximum business value by bringing technology, finance, and business together through cloud transformation. One of its key pillars is spend control. It’s not about cutting business but being more aware of the possibilities on the cloud and optimizing resources to achieve the same goal with less spending.

The area we are looking at today is BigQuery storage cost. Many people believe that storage is cheap, which is not entirely wrong. According to Backblaze, a cloud storage and data backup company, the cost per Gigabyte has dropped by 90% since 2009.



Source: [Backblaze](https://www.backblaze.com/blog/hard-drive-cost-per-gigabyte/" \t "_blank)

Does it mean that we will spend less and less money on storage? No, the truth is data volume growth has gone through the roof in 10 years with an [increase of 1900%](https://www.statista.com/statistics/871513/worldwide-data-created/), and the [cloud storage price has stagnated for the past five years](https://www.storj.io/blog/the-high-price-of-traditional-cloud-storage). With the inflation going on, many providers have even increased their storage prices for 2023. So, make sure you have a (near) real-time billing dashboard at all layers in your organization to have this insight.

In this article, I want to introduce four ways to help your organization reduce BigQuery storage costs. You will be surprised at the result!

*I’ve a few other articles about cost optimization:*[*7 Cost Optimization Practices for BigQuery*](https://medium.com/towards-data-science/7-cost-optimization-practices-for-bigquery-6f776582e62d?source=your_stories_page-------------------------------------)*,*[*How I build a Real-time BigQuery Pipeline for Cost Saving and Capacity Planning*](https://medium.com/towards-data-science/how-i-build-a-real-time-bigquery-pipeline-for-cost-saving-and-capacity-planning-15712c97f058?source=your_stories_page-------------------------------------)*. Feel free to checkout them out as well.*

**BigQuery storage pricing model**

Let’s first see the pricing model of BigQuery storage (price in Jan 2023). BigQuery offers two pricing models: Logical and Physical. It is a property on the dataset level called [storage\_billing\_model](https://cloud.google.com/bigquery/docs/updating-datasets" \l "update_storage_billing_models" \t "_blank) :

* Logical: This is the default billing model of a dataset. The data size is calculated based on the data types of the individual columns. For example, INT64 type takes 8 logical bytes.
* Physical: The data size is calculated based on the data stored on the disk after compression. It’s worth noting that it includes the bytes used for [time travel](https://cloud.google.com/bigquery/docs/time-travel) storage (7 days by default).

Table

Description automatically generated

BigQuery storage pricing model (Created by author)

In both pricing models, we pay for active storage and long-term storage with different price tags. BigQuery automatically labels data as active or long-term based on modifications.

* Active: Table or table partition that has been modified in the last 90 days. Actions include loading, copying, and streaming data into a table and using DML, DDL, etc. Simply querying the table won’t make it active.
* Long-term: Table or table partition that hasn’t been modified for 90 consecutive days. There is no difference in performance, durability, or availability between active storage and long-term storage.

**Switch storage\_billing\_model to physical**

BigQuery stores data in a columnar format — [Capacitor](https://cloud.google.com/blog/products/bigquery/inside-capacitor-bigquerys-next-generation-columnar-storage-format), which can achieve a very high compression ratio of up to 1:10 and high scan throughput. Physical model charges the compressed bytes, but is it always cheaper than logical? Let’s see a few examples from [BigQuery public datasets](https://cloud.google.com/bigquery/public-data).

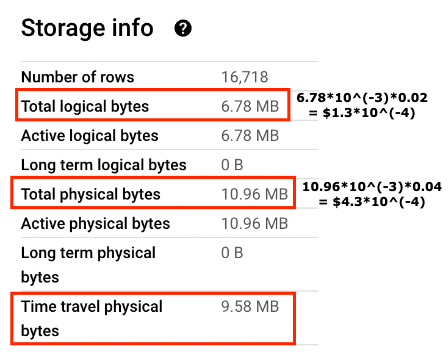
* bigquery-public-data.cloud\_storage\_geo\_index.landsat\_index — This table is recreated every day, so every byte is an active byte, and it doesn’t have time-travel data. The table has ~3GB logical bytes and ~480MB physical bytes. The compression ratio is almost 86%. Pretty impressive! In terms of the price, physical model is much cheaper because it doesn’t have time-travel bytes, and the compression ratio is more than 50%.

Table

Description automatically generated

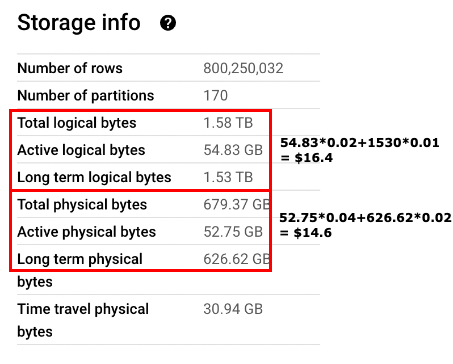
Storage info of `landsat\_index` table

* bigquery-public-data.google\_cloud\_release\_notes.release\_note — This table is updated daily. The total physical bytes are more than logical bytes because it includes time-travel data. As every byte is active, switching to physical pricing model actually costs more.



Storage info of `release\_note` table

* bigquery-public-data.crypto\_bitcoin.transactions — This table is updated daily, and more than 90% of its storage is labeled as long-term. For the price, the physical model is slightly cheaper.



Storage info of `transactions` table

We can’t make the conclusion now that one billing model is more cost-effective than the other because it depends on how tables in the dataset are modified. But here are a few rules to help you decide: (assume compression rate is more than 50%).

1. If the table has no (or very few) time-travel bytes, then choose physical.
2. If the table only has active bytes and it has the default time-travel setting, then think about keeping it logical.
3. If the table has a high percentage of long-term bytes and default time-travel setting, physical might be cheaper but not that significant.

An important note is that if you change a dataset’s storage billing model to use physical bytes, you can’t change it back to using logical bytes. So, make the switch with caution.

If you are not sure about the switch, you can use BigQuery information\_schema.table\_storage\_timeline\_by\_project table to monitor the storage metadata daily or monthly and make the switch when the trend is more stable.

**Use table clone/snapshot instead of table copy**

This tip is helpful if you often need to copy tables. You may copy the table daily to keep the history or copy the table from prod to test for testing purposes. There are four ways to copy tables:

* Data transfer service
* Table copy
* Table clone
* Table snapshot

[Data transfer service](https://cloud.google.com/bigquery/docs/dts-introduction) automates data movement into BigQuery on a scheduled, managed basis. It can copy tables and datasets and import data from external sources into BigQuery. [Table copy](https://cloud.google.com/bigquery/docs/managing-tables#copy-table) is a table-level operation that creates a full copy of the table. In both approaches, the copied tables have type “BASE TABLE”. It means that BigQuery will charge the full storage amount of the copied table.

*We can find out table type from information\_schema.tables.*

[**Table clone**](https://cloud.google.com/bigquery/docs/table-clones-intro)

Table clone is a lightweight, writeable copy of the base table. The best part is that you are only charged for the storage of data in the table clone that differs from the base table. So, initially, there is no storage cost for a table clone! The following graph illustrates the costs.

Diagram

Description automatically generated

Storage difference between base table and table clone (Created by author)

Initially, when you clone the table, there is no storage cost for the new table because it’s the same as the base table. It only charges when there is a difference, and here is the formula.

New base table: original table + new data - deleted data  
 [0-9] + [A] + [B] + [C] - [0] - [1]  
New table clone: changed data + new data  
 [0]+[1]+[4]+[7]+[8]+[9]+[D]  
[0] + [1] : deleted data in base table which exists in table clone  
[4] + [9] : modified data in base table which exists in table clone  
[7] + [8] : modified data in table clone which exists in base table  
[D]: new data

It’s worth noting that some changes to a base table can result in you being charged the full storage amount for a table clone. For example, if you modify a base table with clustering, you are charged the full storage amount of the updated partition in any cloned tables of that base table. In the following graph, one value change in P2 updates the entire P2, so it charges the entire partition.

Diagram, table

Description automatically generated

Storage difference when base table is partitioned and clustered (Created by author)

Having partitions can always help reduce storage costs for table clones because BigQuery only charges the partitions with modified data instead of the entire table.

Table clone is recommended when you copy the table to a test project for testing purposes, and you only update part of the table (e.g., 20%). In that case, you can save 80% of the original storage cost. In the worst case, when you update the entire table, the storage cost of the table clone will be the same as if you use the normal table copy. Thus, it’s definitely worth a try!

But there are a few limitations. For example, you can’t create a clone of a view, materialized view, or an external table. The table clone must be in the same region as the base table. This feature is currently in the Preview stage (Jan 2023), which might have limited support.

[**Table snapshot**](https://cloud.google.com/bigquery/docs/table-snapshots-intro)

Another variation of table clone is called table snapshot. The difference between table clone and table snapshot is that table snapshot is read-only. It preserves the content of a base table at a particular time. For the storage cost, BigQuery only charges for data in a table snapshot that no longer exists in its base table or that has changed in its base table. For example:

Diagram

Description automatically generated

Storage difference between base table and table snapshot (Created by author)

Similar to table clone, there is no initial storage cost. Here is the cost when there is an update in the base table.

New table snapshot: deleted from or changed in base table  
 [0]+[1]+[4]+[9]  
[0] + [1] : deleted data in base table which exists in table snapshot  
[4] + [9] : modified data in base table which exists in table snapshot

Having partitions in the base table also helps reduce storage costs for table snapshots. In terms of the limitations, they are similar to table clone. But table snapshot is a GA feature, so it has more documentation and receives more support from GCP.

Table snapshot is beneficial if you want to preserve the table history for longer than 7 days. With BigQuery time travel, you can only access table’s data from 7 days ago. Table snapshot can preserve read-only data for as long as you want.

**Set table expiration**

One of the easiest cost-saving tips is to delete unused tables or table partitions. We often have such tables in a test environment. BigQuery allows us to set expiration on partition-level, table-level, and dataset-level.

-- partition level  
ALTER TABLE mydataset.mytable SET OPTIONS (partition\_expiration\_days = 5);  
-- table level  
ALTER TABLE mydataset.mytable SET OPTIONS (expiration\_timestamp = TIMESTAMP '2025-02-03 12:34:56');  
-- dataset level  
ALTER SCHEMA mydataset SET OPTIONS( default\_table\_expiration\_days = 3.75);

After the expiration date, tables or partitions will be automatically deleted. Note that if you update the default table expiration for a dataset, it will only apply to the new tables created.

**Keep old data in BigQuery rather than exporting it**

BigQuery is not a traditional data warehouse. It’s grown into a data lakehouse which is a new architecture that combines the best elements of a data warehouse and data lake.

We can sense this by looking at their pricing model. The storage costs of active and long-term bytes in the logical model are the same as standard and nearline storage types in Cloud Storage. However, cloud storage has operation charges, as shown in the following table.

Table

Description automatically generated

GCP Cloud storage pricing model (Created by author)

BigQuery doesn’t have such operation charges, and simply querying a long-term table won’t change it to active. So, from the cost and operation perspective, keeping old data in BigQuery is a preferable option.

**Conclusion**

As always, I hope you find this article inspiring and useful. 2022 has been one of the hardest years ever to run a business. All sorts of challenges pushed engineers to look at their technical stacks from different perspectives, thinking from how to scale the system to how to control the cost to make the business more resilient.

If you use BigQuery, share these four tips with your colleagues. I’m sure that they can bring a huge impact on your business and allow the company to allocate the money to more critical domains. If you have any thoughts, please let me know in the comment. Cheers!