Google BigQuery & Tableau: Best Practices

Google BigQuery & Tableau: Best Practices

Tableau and Google BigQuery allows people to analyze massive amounts of data and get answers fast using an easy-to-use, visual interface. Using the tools together, you can:

- Put the power of Google BigQuery into the hands of everyday users for fast, interactive analysis.
- Analyze billions of rows in seconds using visual analysis tools without writing a single line of code and with zero server-side management.
- Create stunning dashboards in minutes that connect to your Google BigQuery data and keep your organization up to speed.
- Share reports and insights on the web using Tableau Server and Tableau Online to allow anyone to connect from any device.
- Combine the cloud agility of Google BigQuery with the blazing speed of Tableau to recognize project value faster.

Optimizing the two technologies together will yield significant performance gains, shorten design cycles, and help users and organizations become more successful. In this paper, we will discuss techniques to optimize data modeling and query formation to maximize the responsiveness of visualizations. We will also discuss techniques to get the best cost efficiency when using Tableau and BigQuery together.

Martin Sleeman, Product Manager, Tableau
Marc Lobree, Product Consultant, Tableau
Vaidy Krishnan, Senior Product Marketing Manager, Tableau
Babu Prasad Elumala, Solutions Engineer, Google
Seth Hollyman Technical Program Manager, Google
Tino Tereshko, Enterprise Solutions Engineer, Google
Mike Graboski, Solutions Engineer, Google

Contents

Technology Overview

Google BigQuery

BigQuery can process petabytes of data in seconds in plain SQL with no fine-tuning or special skill set required. Powered by Dremel, Google's revolutionary technology for analyzing massive data sets, BigQuery provides a level of performance that large businesses previously had to pay millions to obtain—at a cost of pennies per gigabyte.

BigQuery is a data warehouse best suited for running SQL queries against massive, structured, and semi-structured data sets. Example use cases and data sets include:

- Ad hoc analytics
- Web logs
- Machine/server logs
- Internet of Things data sets
- E-commerce customer behavior
- Mobile app data
- Retail analytics
- Gaming telemetry
- Google Analytics Premium data
- Any data set for which a traditional RDBMS is taking minutes (or hours) to run a batch query

BigQuery is completely NoOps and maintenance-free, and is integrated with the Google Cloud Platform. Unlike other cloud-based analytics solutions, BigQuery does not require you to provision a cluster of servers in advance. Processing clusters are sized and provisioned by BigQuery at runtime.

As your data size increases, BigQuery will automatically add processing power—but you pay the same price per gigabyte.

Legacy SQL vs. Standard SQL

Google BigQuery upgraded its APIs to use standard SQL in addition to BigQuery SQL (now called legacy SQL), and Tableau upgraded the Google BigQuery connector to support this change to standard SQL. With standard SQL comes benefits for BigQuery users including Level of Detail Expressions, faster metadata validation, and the ability to select a billing project with your connection.

This guide is written assuming standard SQL. For more information about migrating from legacy SQL to standard SQL, see our Online Help guide on migrating from legacy SQL on the Google Cloud Platform website.

Tableau

Tableau helps people to see and understand data. Our software products put the power of data into the hands of everyday people. This allows a broad population of users to engage with their data, ask questions, solve problems, and create value. Based on technology developed at Stanford University, our product reduces the complexity, inflexibility, and expense associated with traditional business intelligence applications. Anyone who is comfortable with Excel can leverage Tableau Desktop to create rich, interactive visualizations and powerful dashboards using a drag-and-drop user interface as well as share them securely across organizations using Tableau Server or Tableau Online.

Tableau has a native, optimized connector to Google BigQuery that supports both live data connectivity and in-memory extracts. Tableau's data blending allows users to mash up data from BigQuery with data from any of our 60 other supported data sources. For visualizations published to the cloud using Tableau Server or Tableau Online, direct connectivity to Google BigQuery can be maintained.

Best Practices for Performance: Tableau

Leverage Tableau 10

One of the easiest ways to accelerate performance is to ensure you are using Tableau 10. Keeping your deployment up to date, allows you to benefit from all the performance goodness that we keep adding to the product on a regular cadence.

Tableau 9 was a watershed release for us. It represented a giant leap in which we introduced an astonishing number of groundbreaking performance improvements to ensure your visualizations are responsive.

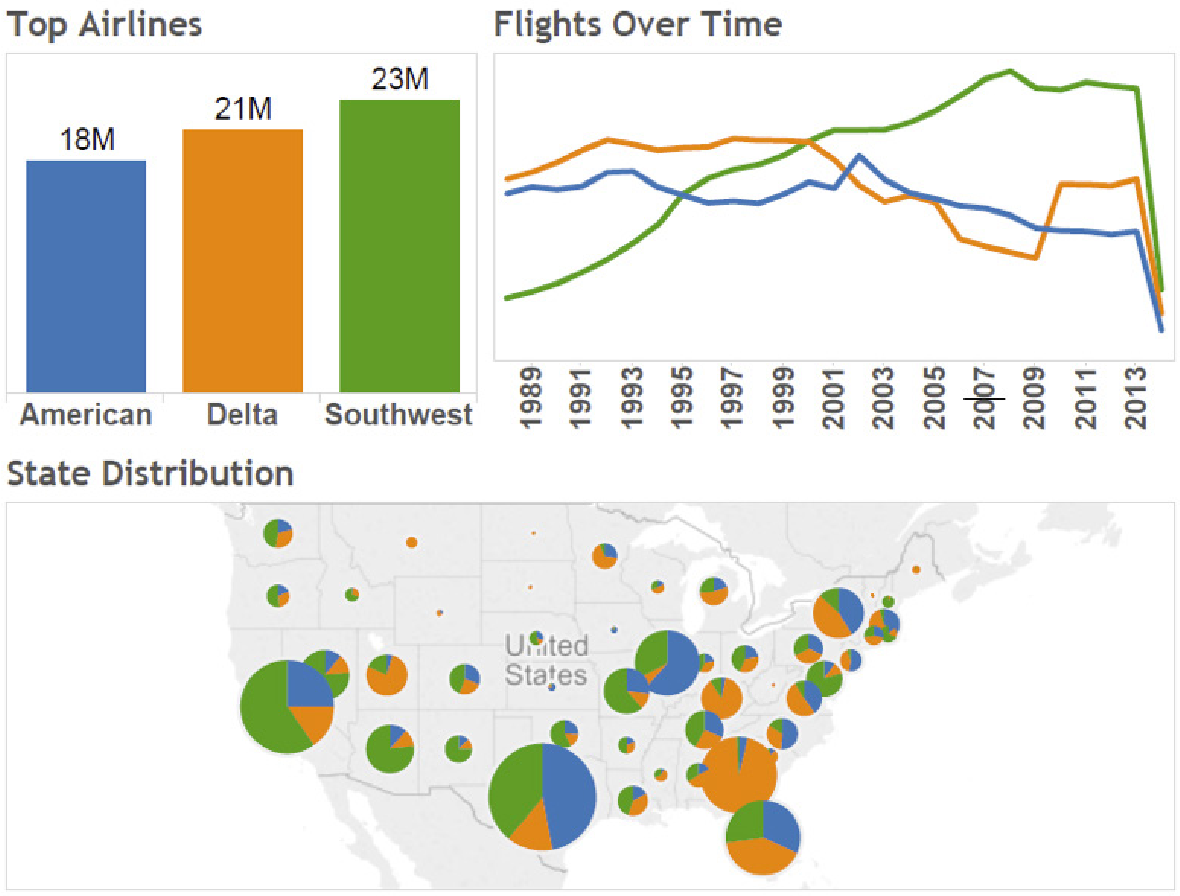
These enhancements included:

- **Level of Detail calculations:** LOD Expressions allow us to look beyond the visualization level of detail. The data in the visualization is often a result of filtering the data from the data source. LOD Expressions can look at the data before it is filtered and thus enables more powerful analyses.

- **Parallel queries:** Tableau will take advantage of the capability of Google BigQuery and other data sources to execute multiple queries at the same time for a total of up to 16 concurrent queries. Batches of independent and de-duplicated queries are grouped together and sent to BigQuery if the result is not already cached. Users should expect to see large performance gains due to parallel queries because of BigQuery's scale-out architecture.

- **Query fusion:** Tableau will take multiple queries from workbooks and dashboards and fuse them together when possible, reducing the number of queries sent to BigQuery. First, Tableau identifies similar queries, excluding differences in the columns that are returned. Then, it combines queries where the differences are only the level of aggregation or a user calculation.

- **External query cache:** If the underlying data source hasn't changed since the last time you ran the same query, Tableau will automatically read from the previously saved query cache, providing nearly instantaneous load times. For example, a workbook with a 157 million-row Tableau Data Extract file opens 50 times faster when cached in Tableau 9 than it does without a cache in Tableau 8.3.



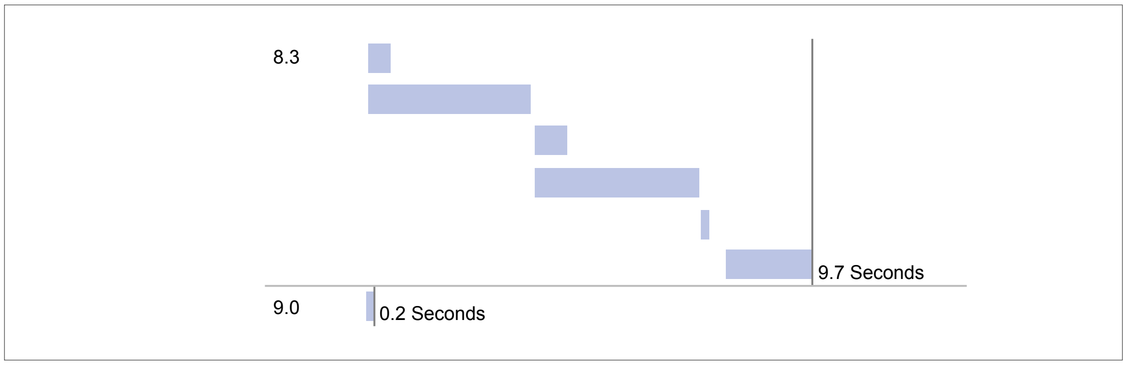


Figure 1: Performance improvement for a 157 million-row data set opened in Tableau 9.0 versus Tableau 8.3.

In Tableau 10 we've continued on that innovation path to ensure performance for scalable enterprise deployments. In particular, we've introduced:

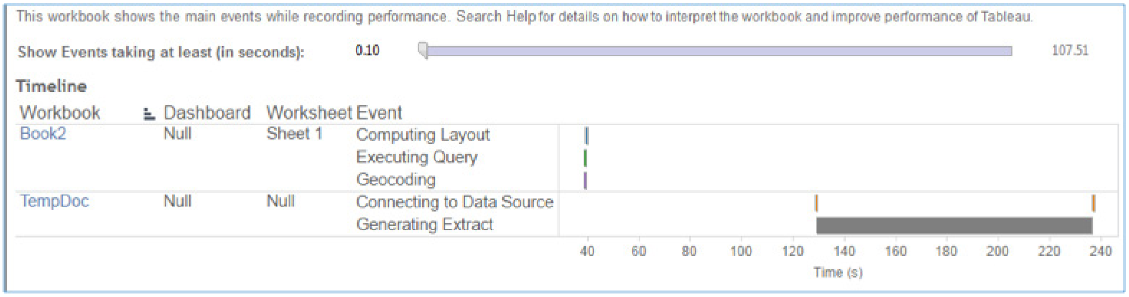
- **Improved browser performance:** In Tableau 10, we are caching the initial workbook load to give you faster workbook performance. Enjoy speedier load times, instant updates based on your interactions, and more.

- **On-demand connections in Tableau Desktop:** When you open a published workbook, Tableau Desktop only connects to the data sources required to display the current sheet's data. In other words, see your data a whole lot faster.

- **Tableau Server stability improvements:** Tableau 10 includes many stability improvements for Tableau Server. For single-server installations, we've made Tableau Server more robust to issues during periods of high-disk IO latency. Tableau Server requires three nodes for HA, yet we previously allowed customers to configure failover and replication with two nodes, causing customer confusion. For two-node installations, you'll now be limited to a single instance of the repository. If you need failover or high availability with a second instance of the repository, install Tableau Server on a cluster of at least three nodes. This way, you can configure two instances of the repository and gain the benefit of an automatic failover. We've also reduced the overall memory usage of the Tableau Server processes to improve performance.

Performance Recorder

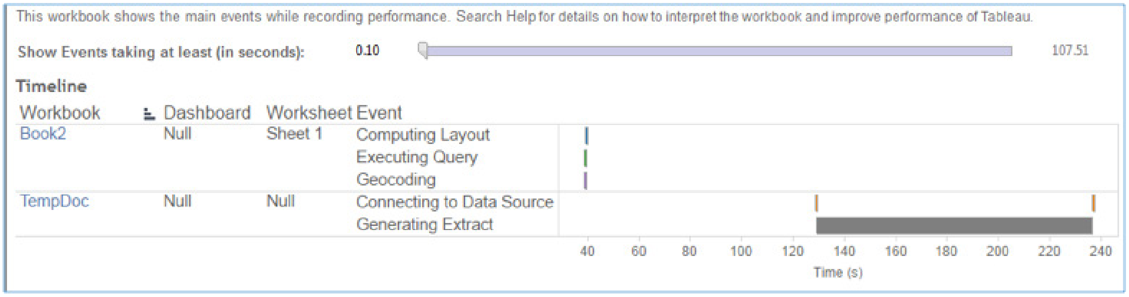
Performance Recorder is a powerful built-in tool that allows you to pinpoint slow queries and optimize your workbooks for maximum performance. It does this by tracking the elapsed time for an individual workbook to execute a query and compute the layout. Hovering over one of the green bars below will show the user the query that's being generated against BigQuery. After identifying a slow query, you can often resolve the performance issue by revisiting your data model.


This workbook shows the main events while recording performance. Search Help for details on how to interpret the workbook and improve performance of Tableau.

Show Events taking at least (in seconds): 0.10 — 107.51

Timeline

Workbook	Dashboard	Worksheet	Event
Book2	Null	Sheet 1	Computing Layout
			Executing Query
			Geocoding
TempDoc	Null	Null	Connecting to Data Source
			Generating Extract


This workbook shows the main events while recording performance. Search Help for details on how to interpret the workbook and improve performance of Tableau.

Show Events taking at least (in seconds): 0.10 — 107.51

Timeline

Workbook	Dashboard	Worksheet	Event
Book2	Null	Sheet 1	Computing Layout
			Executing Query
			Geocoding
TempDoc	Null	Null	Connecting to Data Source
			Generating Extract

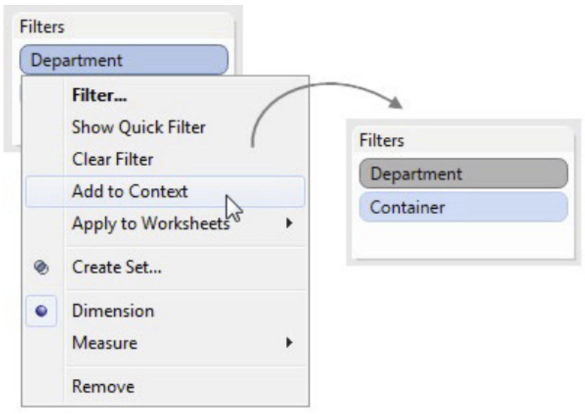
For instructions on how to create or interpret a performance recording, please follow one of these links:

- *Performance Recorder on Tableau Desktop*
- *Performance Recorder on Tableau Server*

Context Filters

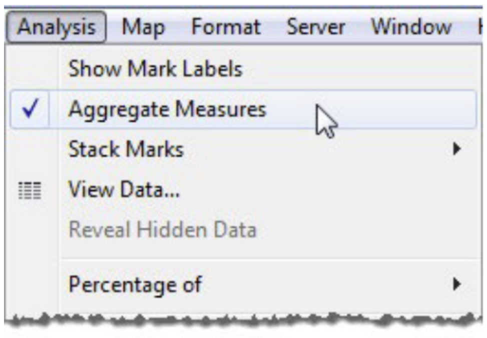
If you are applying filters to a large data source, you can improve performance by setting up context filters. A context filter is applied to the data source first, so that additional filters are applied only to the resulting records. This sequence avoids applying each filter to each record in the data source.

If you are setting filters that significantly reduce the size of the data set and will use those filters for more than several data views, then you should set those filters as context filters. Refer to our Online Help guide on improving view performance with context filters to learn how to create context filters.



Aggregate Measures

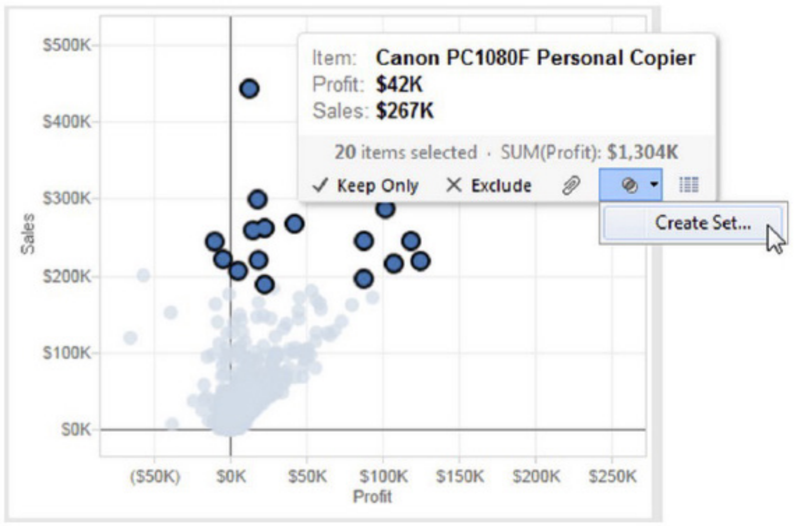
If the views you create are slow, make sure you are working with aggregated measures rather than disaggregated measures. When views are slow, it usually means you are trying to view many rows of data at once. You can reduce the number of rows by aggregating the data. In other words, make sure the Aggregate Measures option on the Analysis menu is selected. For more information, refer to our Online Help guide on aggregating data.



Sets

If you want to filter a dimension to remove members based on a range of measure values, you should create a set rather than using a quantitative filter. For instance, you can create a set that only returns the top 50 items in a dimension, rather than all of the items in a dimension. For more information, refer to our Online Help guide on creating sets.

When creating a group from a selection as described in our Online Help guide on creating groups, make sure you've included only the columns of interest. Each additional column included in the set will result in decreased performance.



Add Filters First

If you are working with a large data source and have automatic updates turned off, it is possible to create a really slow query when adding filters to the view. Rather than build the view and then specify filters, you should first specify the filters and then drag fields to the view. That way, when you run the update or turn on automatic updates, the filters will be evaluated first.

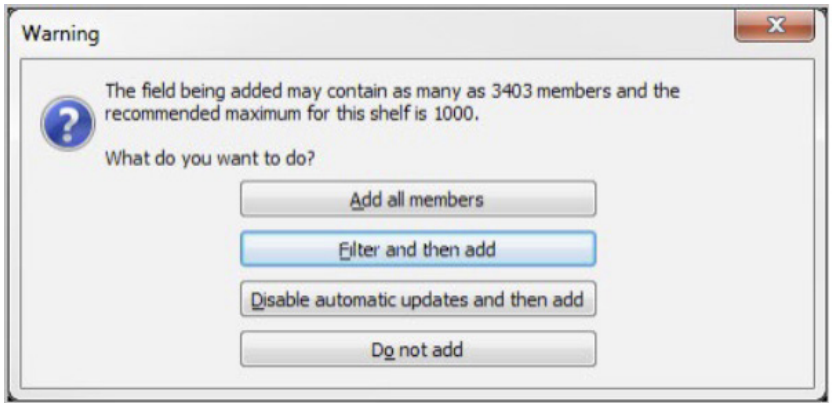
Turn Off Automatic Updates

When you place a field on a shelf, Tableau generates the view by automatically querying the data source. If you are creating a dense data view, the queries might be time consuming and significantly degrade system performance. In this case, you can instruct Tableau to turn off queries while you build the view. You can then turn queries back on when you are ready to see the result. Refer to our Online Help guide on managing queries for more information.

Look for Warnings

Tableau displays a performance warning dialog box when you attempt to place a large dimension (with many members) on any shelf. The dialog box provides four choices as shown in the figure below.

If you choose to add all members, you might experience a significant degradation in performance.


Warning

The field being added may contain as many as 3403 members and the recommended maximum for this shelf is 1000.

What do you want to do?

Best Practices for Cost and Performance: Google BigQuery

For high performance querying and reduced costs, it is generally best to avoid using federated tables where the data is in an external data source like Google Cloud Storage. In such cases, if you are looking to perform iterative querying on the data set, you should use the Query API to materialize the data in BigQuery (independent of Tableau) to enable high performance querying on the dataset with Tableau.

Denormalize and Pre-JOIN

BigQuery supports very large JOINs, and JOIN performance is very good. That being said, BigQuery is a Columnar Datastore, and maximum performance is achieved on denormalized data sets.

Because BigQuery storage is very inexpensive and scalable, it is often prudent to denormalize and pre-JOIN data sets into homogeneous tables. In essence, you are exchanging compute resources for storage resources (the latter being more performance- and cost-effective). Since BigQuery is a columnar store, exchanging compute for storage is not a bad choice since it can compress data better.

BigQuery is an excellent ETL tool, allowing you to execute massive transforms and pipelines quickly and efficiently. Be sure to enable "allow large results" when materializing data sets larger than 128 MB.

For more information on how to prepare data for loading and how to query data using BigQuery's SQL dialect visit the online documents below.

- cloud.google.com/bigquery/preparing-data-for-bigquery#denormalizingdata
- cloud.google.com/bigquery/querying-data#largequeryresults

Shard Tables by Date

Some data naturally lends itself to being partitioned by date: for example, log data, or any data for which the records include a monotonically increasing timestamp. In this case, shard your BigQuery tables by date and include the date in the table name. To take advantage of this, you would need to leverage custom SQL in Tableau. For more information see our Online Help guide on connecting to a custom SQL query. For example, name your tables something like:

mytable_20170501, mytable_20170502, etc.

Then, when you want to run a query that filters by date, use BigQuery's Wildcard Table's function:

```
SELECT
       name
FROM
       `myProject.myDataSet.mytable_*'
WHERE
       age >= 35
```

The example above will automatically include all tables with the prefix mytable_.

To use wildcard , your tables must be named according to the pattern:

[arbitrary prefix]YYYYMMDD.

Other database systems rely on sharding to improve performance. Sharding by date actually has a negligible performance difference on BigQuery, but the main driver here is cost. Because you're processing less data, you're paying less money per query.

A caveat here is, if you decide to shard by minute level, then you may have too many shards which directly impacts performance. Care must be taken so that you don't shard too much at the same time. Anything upwards of daily is generally acceptable.

For a deeper understanding of how to work with wildcards click here

Specify a Destination Table If Running Many Similar Queries

While query caching is useful if you're running many identical queries, it won't help if you're running similar, but slightly different, queries (e.g., changing only the values in a WHERE clause between query runs). In this case, run a query on your source table and write the records you will repeatedly query to a new destination table. Then, run queries against the new destination table that you created.

For example, let's say you plan to run three queries with three different WHERE clauses:

```
WHERE col1 = "a"
WHERE col1 = "b"
WHERE col1 = "c"
```

Run a query against your source table, and write the output records into a destination table:

```
SELECT col1
FROM source
WHERE col1 = "a" OR col1 = "b" OR col1 = "c"
```

By "OR"ing the WHERE clauses together, we capture all relevant records. Our new destination table is potentially much smaller than the original source table. Since BigQuery charges based on the amount of data processed in a query, running subsequent queries against the new destination table will save money instead of running the queries directly against the source table. Care must be taken to clean up these tables in the future so as to prevent storage cost accumulating for these tables.

Conclusion

By applying best practices, business users and data analysts alike will be able to maximize the performance and responsiveness of Tableau visualizations built against Google BigQuery. When these technologies are combined, users can truly visualize billions of rows of data at the speed of thought.

About Tableau

Tableau helps people see and understand data. Tableau helps anyone quickly analyze, visualize and share information. More than 29,000 customer accounts get rapid results with Tableau in the office and on-the-go. And tens of thousands of people use Tableau Public to share data in their blogs and websites. See how Tableau can help you by downloading the free trial at tableau.com/trial.

Additional Resources

Download Free Trial

Related Whitepapers

Why Business Analytics in the Cloud?

Top 10 Cloud Trends for 2017

Tableau Server and Google Cloud Platform: Rapid Fire Business Intelligence in the Cloud

See All Whitepapers

Explore Other Resources

Product Demo

Training & Tutorials

Community & Support

Customer Stories

Solutions