Cloud Providers’ Services

Data Warehouse	Visualization and BI Service	Cloud Provider	Parent Company
Redshift	**QuickSight**	Amazon Web Services (AWS)	Amazon
Synapse Analytics	**Power BI**	Microsoft Azure	Microsoft
BigQuery	**Looker**	Google Cloud	Alphabet Inc.

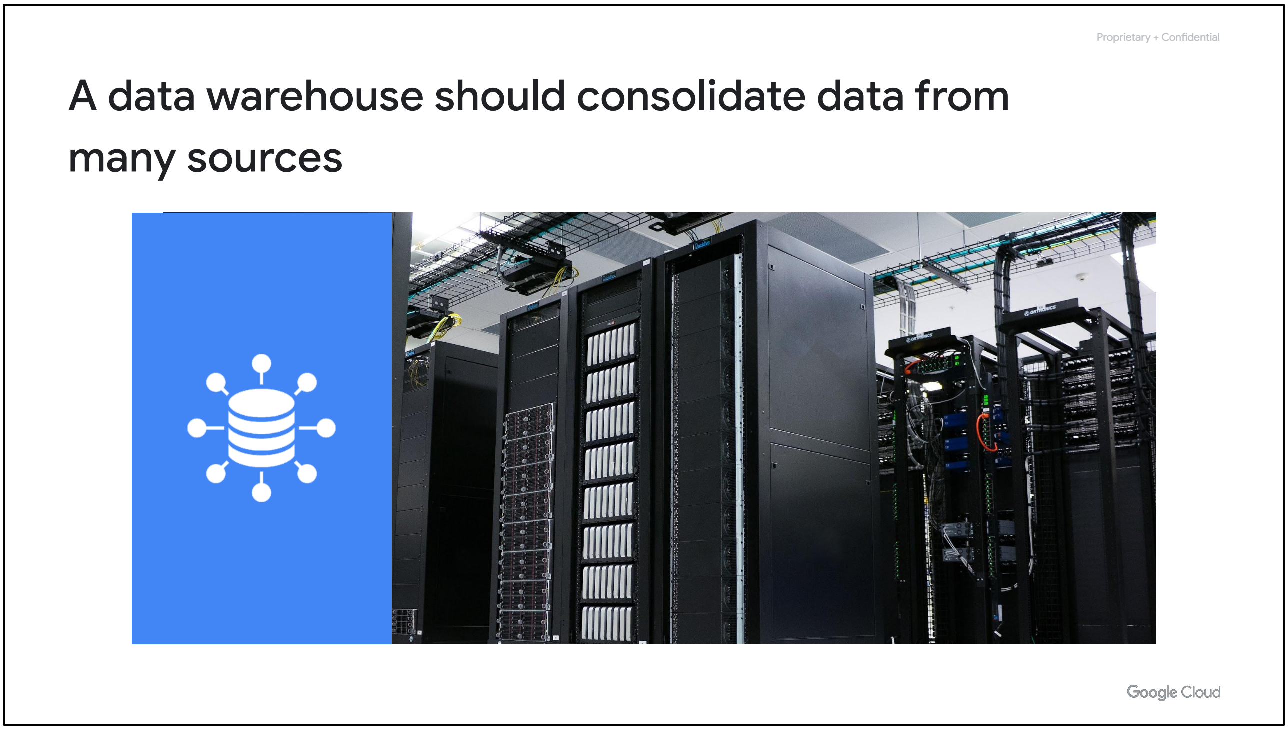
Proprietary + Confidential

01

The Modern Data Warehouse

Google Cloud

Let’s start by describing what makes a modern data warehouse.



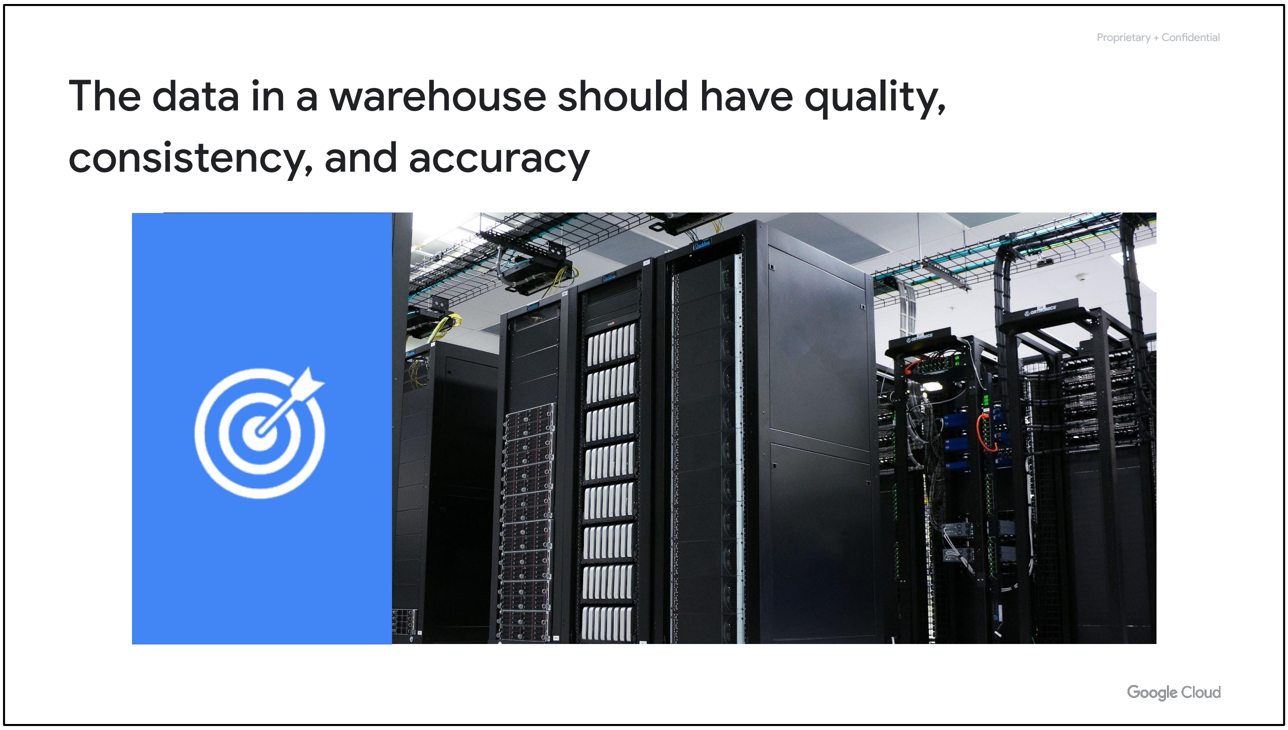
Proprietary + Confidential

A data warehouse should consolidate data from many sources

Google Cloud

An enterprise data warehouse should consolidate data from many sources. If you recall from the previous module, a data lake does something very similar. The key difference between the two is the word “consolidate.”

A data warehouse imposes a schema. A data lake is just raw data, but an enterprise data warehouse brings the data together and makes it available for querying and data processing. To use a data warehouse, an analyst needs to know the schema of the data. However, unlike for a data lake, the analyst doesn’t have to write code to read and parse the data.



The data in a warehouse should have quality, consistency, and accuracy

Another reason to consolidate all your data, besides standardizing the format and making it available for querying, is making sure the query results are meaningful. You want to make sure the data is clean, accurate, and consistent.

Proprietary + Confidential

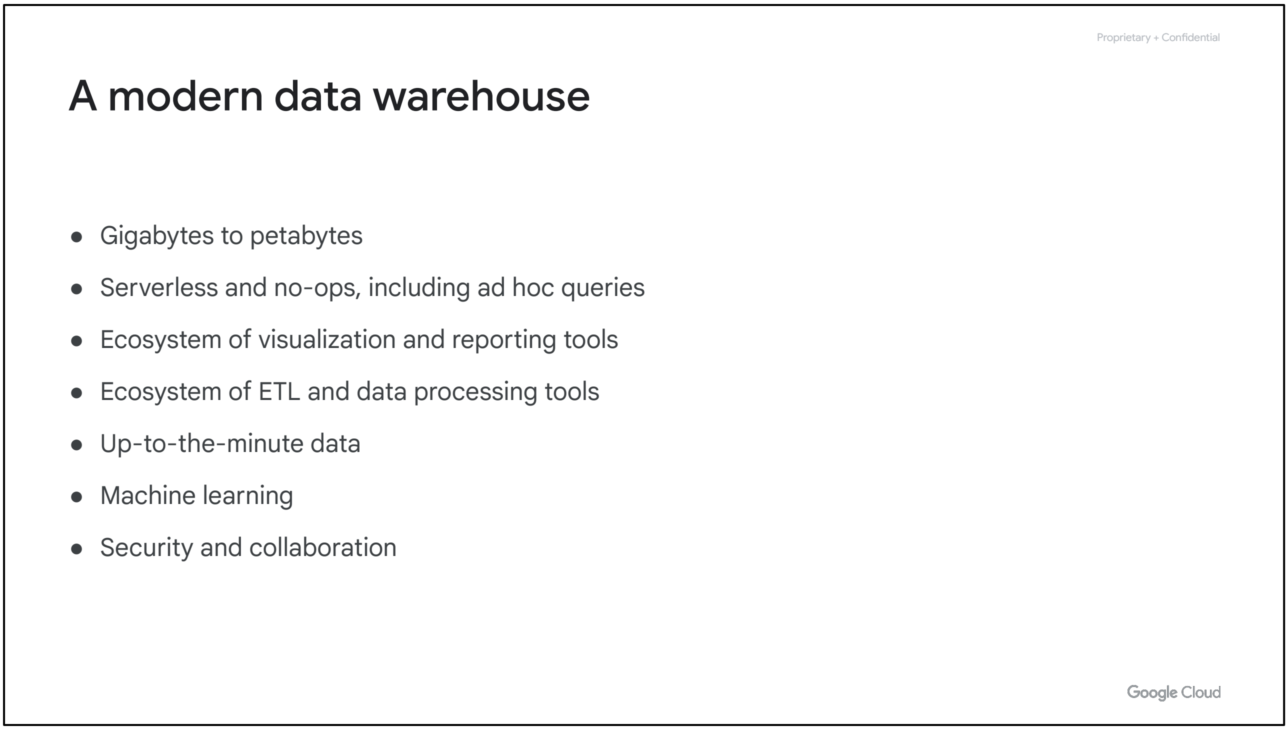
A data warehouse should be optimized for simplicity of access and high-speed query performance

Google Cloud

The purpose of a data warehouse is not to store data. That’s the purpose of a data lake.

If you have raw data that you want to keep around but not necessarily query, don’t bother with cleaning and streamlining it. Leave it in a data lake.

All data in a data warehouse should be available for querying. It’s important to ensure that those queries are quick: you don’t want people waiting hours or days for results.



Proprietary + Confidential

A modern data warehouse

- Gigabytes to petabytes
- Serverless and no-ops, including ad hoc queries
- Ecosystem of visualization and reporting tools
- Ecosystem of ETL and data processing tools
- Up-to-the-minute data
- Machine learning
- Security and collaboration

Google Cloud

We described an enterprise data warehouse and how it’s different from a data lake. What makes a data warehouse modern?

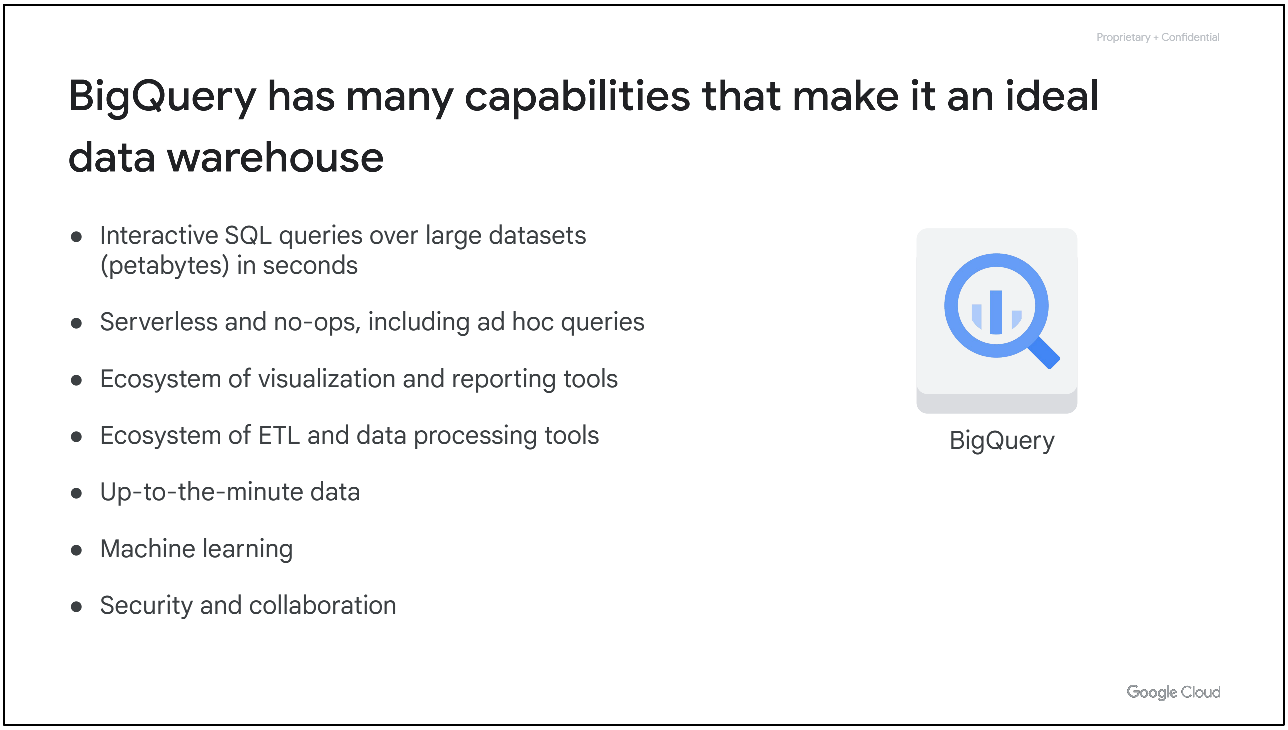
- Businesses’ data requirements continue to grow. You want to make sure the data warehouse can deal with datasets that don’t fit into memory. Typically, this is gigabytes to terabytes of data but occasionally can be petabytes. You don’t want separate warehouses for different datasets. Instead, you want a single data warehouse that can scale from gigabytes to petabytes of data.
- Second, you want the data warehouse to be serverless and fully no-ops. You don’t want to be limited to clusters that you need to maintain, or indexes that you need to fine-tune. Removing these responsibilities will allow data analysts to carry out ad hoc queries faster, which is important because you want the data warehouse to increase the speed at which your business makes decisions.
- Next, your data warehouse is not productive if it allows you to do queries but doesn’t support rich visualization and reporting. Ideally, your data warehouse can seamlessly plug into whichever visualization or reporting tool your business is most familiar with.
- Similarly, because the data warehouse requires clean and consistent data, you will often have to build data pipelines to bring data into the warehouse. The modern data warehouse should be able to integrate with an ecosystem of processing tools for building ETL pipelines.
- Your data pipeline should be capable of constantly refreshing data in the

- warehouse in order to keep it up to date. You need to be able to stream data into the warehouse and not rely on batch updates.
- Also, predictive analytics is becoming increasingly important for data analysts. As a result, a modern data warehouse has to support machine learning without moving the data out of the warehouse.
- Last but not least, in a modern data warehouse it should be possible to impose enterprise-grade security like data exfiltration constraints. It should also be possible to share data and queries with collaborators.

02

Introduction to BigQuery

In this lesson, we’re going to introduce BigQuery, a data warehouse solution on Google Cloud.



Proprietary + Confidential

BigQuery has many capabilities that make it an ideal data warehouse

- Interactive SQL queries over large datasets (petabytes) in seconds
- Serverless and no-ops, including ad hoc queries
- Ecosystem of visualization and reporting tools
- Ecosystem of ETL and data processing tools
- Up-to-the-minute data
- Machine learning
- Security and collaboration

BigQuery

Google Cloud

BigQuery has many capabilities that make it an ideal data warehouse.

When we talked about a modern data warehouse, we talked about having the warehouse be able to scale from gigabytes to petabytes seamlessly.

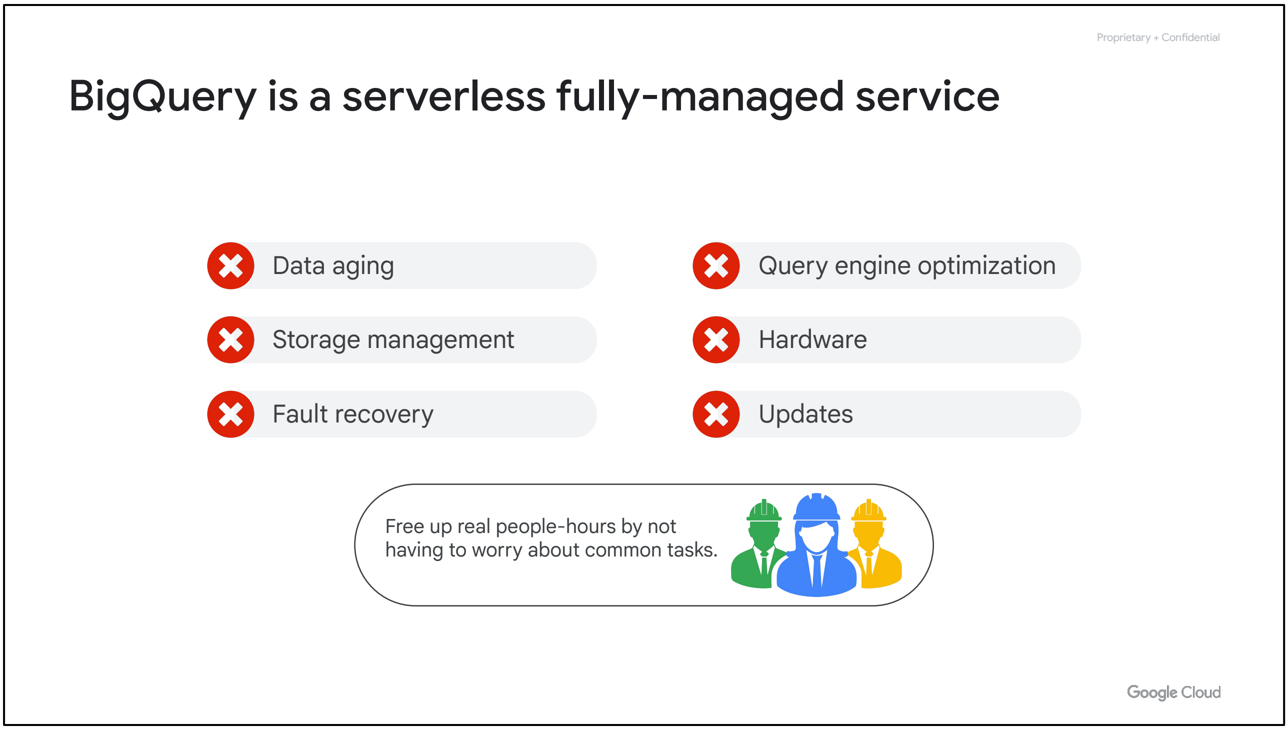
We talked about being able to do ad hoc queries and no-ops.

BigQuery cost-effectively handles large, petabyte-scale datasets for storage and querying. In fact it’s similar to the cost of Cloud Storage. This enables you to store your data without having to worry about archiving off older data to save on storage.

Unlike traditional data warehouses, BigQuery has features like GIS and machine learning built in.

It also provides capabilities to stream data in, so you can analyze your data in near real time.

Because it’s part of Google Cloud, you get all of the security benefits the cloud provides while also being able to share datasets and queries. BigQuery supports standard SQL queries and is compatible with ANSI SQL 2011.



Proprietary + Confidential

BigQuery is a serverless fully-managed service

- ✖ Data aging
- ✖ Storage management
- ✖ Fault recovery
- ✖ Query engine optimization
- ✖ Hardware
- ✖ Updates

Free up real people-hours by not having to worry about common tasks.

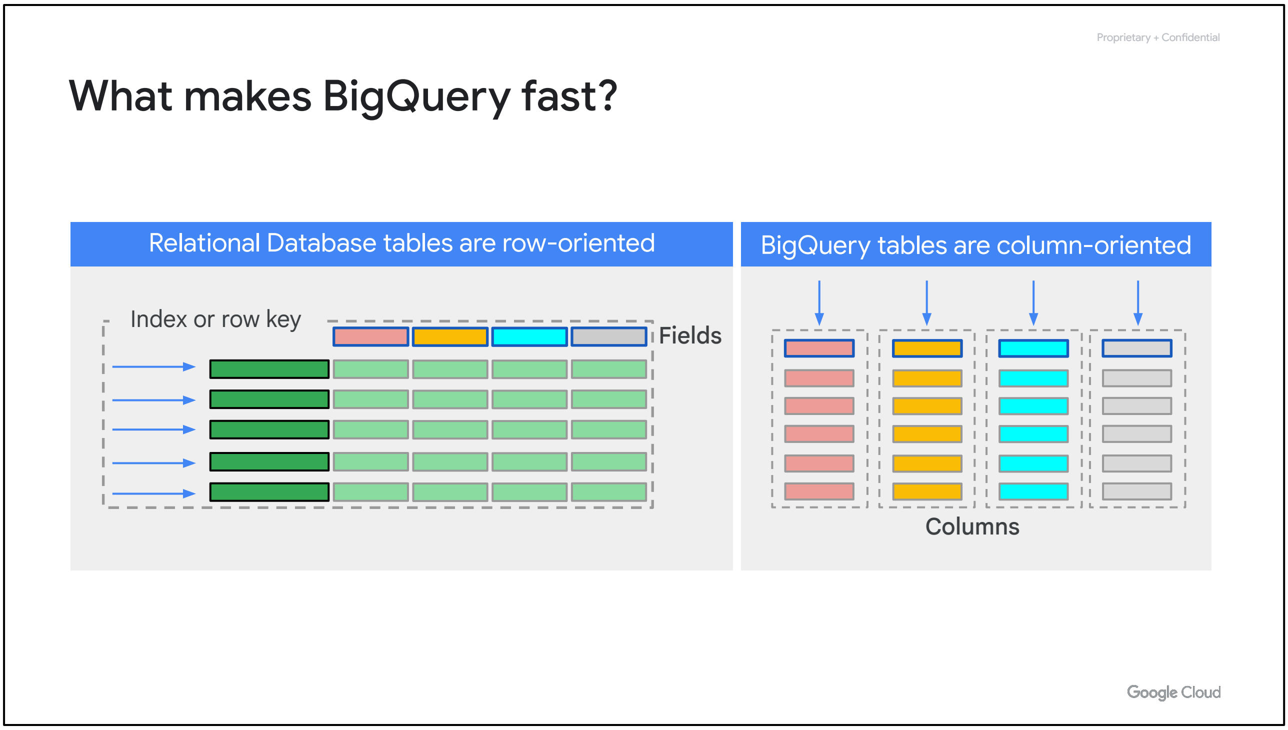
Google Cloud

BigQuery is a serverless fully-managed service, which means that the BigQuery engineering team takes care of updates and maintenance for you. Upgrades don’t require downtime or hinder system performance.

For example, data aging and expiration can be a cumbersome operation in traditional data warehouses. In BigQuery, you just supply a table expiration flag at the time of table creation or update a table to add this feature. The table will automatically expire when it reaches that age or duration.

Many traditional systems require resource-intensive vacuum processes to run at various intervals to reshuffle and sort data blocks and recover space. BigQuery has no equivalent of the vacuum process, because the storage engine continuously manages and optimizes how data is stored and replicated. Also, because BigQuery doesn't use indexes on tables, you don't need to rebuild these.

The bottom line is that you can free up real work hours by not having to worry about common database management tasks.



Proprietary + Confidential

What makes BigQuery fast?

Relational Database tables are row-oriented

Index or row key

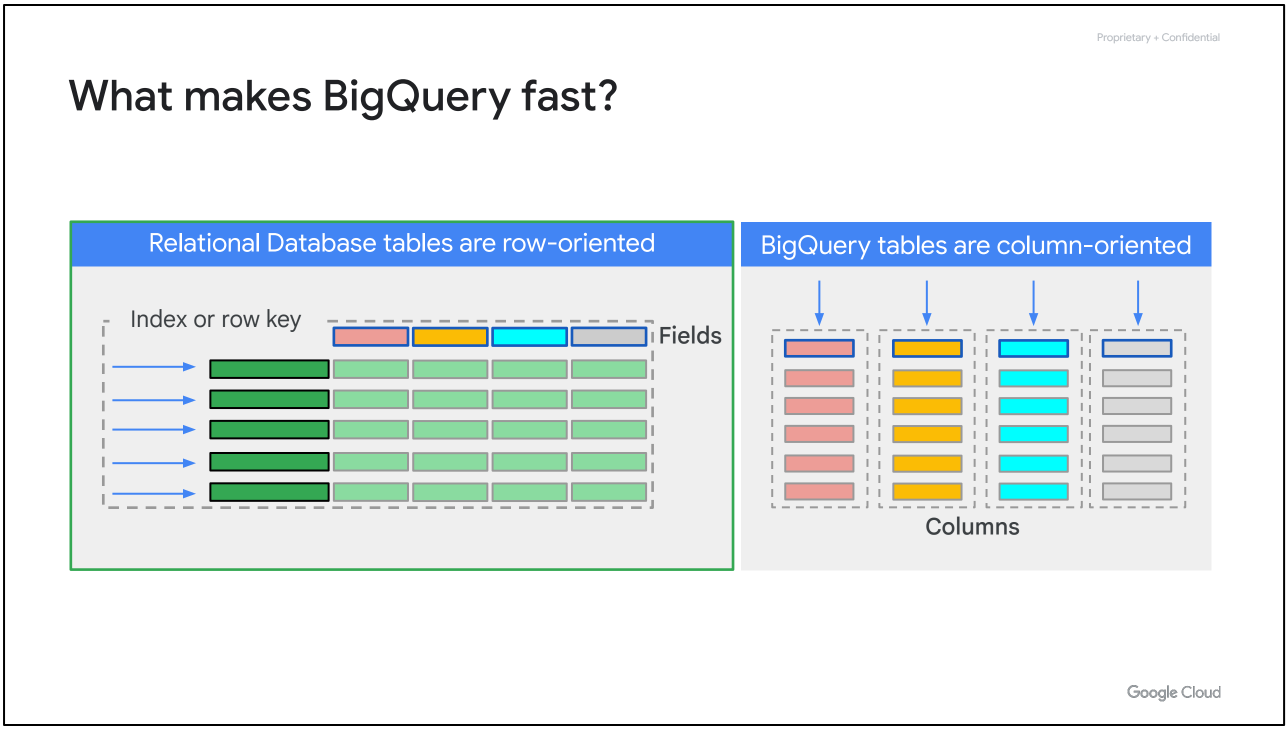
Fields

BigQuery tables are column-oriented

Columns

Google Cloud

So what makes BigQuery fast? BigQuery tables are column-oriented, compared to traditional RDBM tables which are row-oriented.



Proprietary + Confidential

What makes BigQuery fast?

Relational Database tables are row-oriented

Index or row key

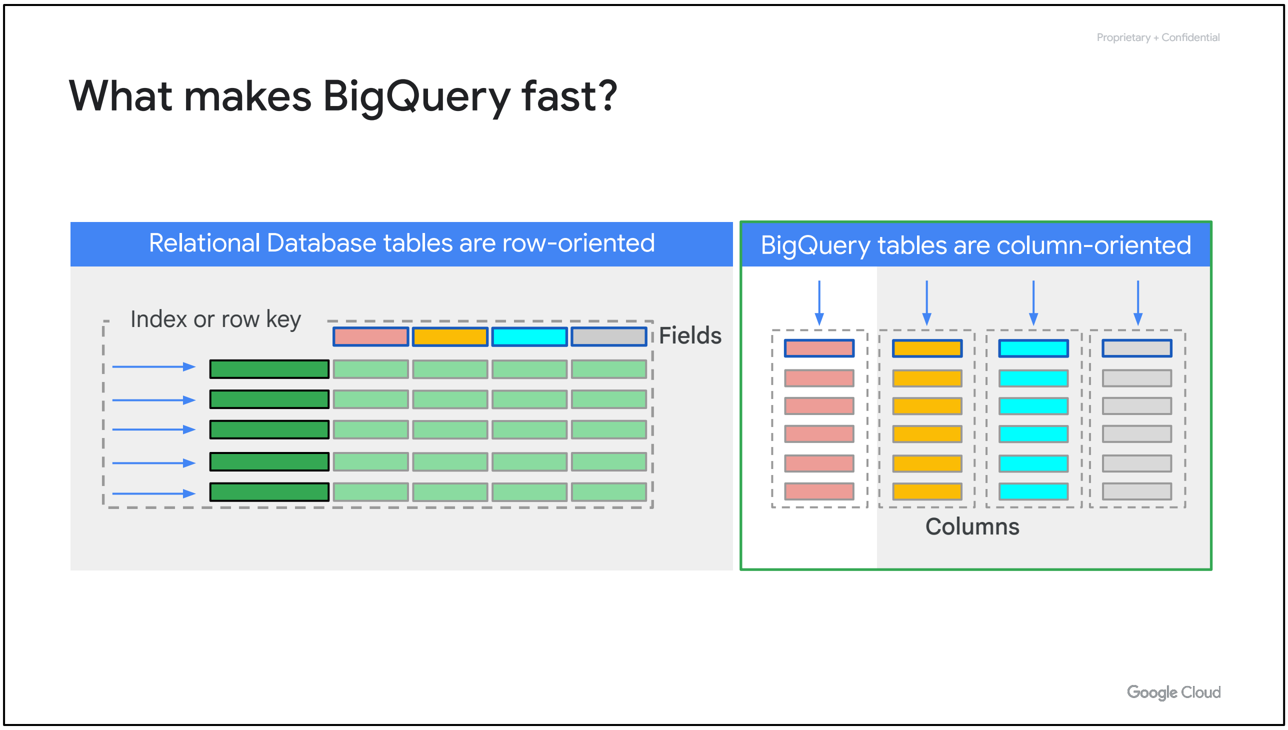
Fields

BigQuery tables are column-oriented

Columns

Google Cloud

Row-oriented tables are efficient for making updates to data contained in fields. For OLTP systems, row-oriented tables are necessary because OLTP systems have frequent updates. Analytics is slow on row-oriented tables because they have to read all the fields in a row and, depending on the kind of indexing or key, they may have to read extra rows and fields to find the information that is requested in a query.



Proprietary + Confidential

What makes BigQuery fast?

Relational Database tables are row-oriented

Index or row key

Fields

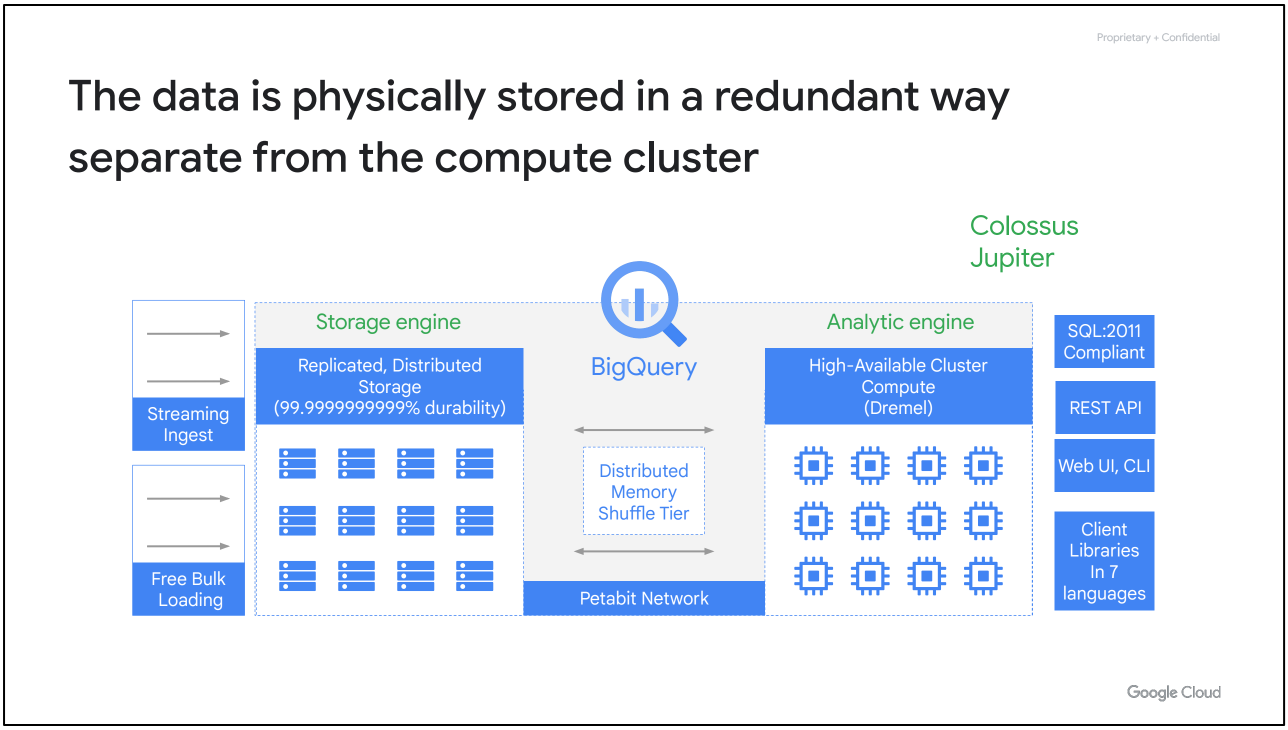
BigQuery tables are column-oriented

Columns

Google Cloud

BigQuery, however, is an O-LAP system. It’s meant for analytics. BigQuery tables are immutable and are optimized for reading and appending data. BigQuery tables are not optimized for updating.

BigQuery leverages the fact that most queries involve few columns, and so it only reads the columns required for the query. BigQuery is very efficient in this sense and is the reason tables are column-oriented.



The data is physically stored in a redundant way separate from the compute cluster

Colossus
Jupiter

Streaming Ingest

Free Bulk Loading

Storage engine

Replicated, Distributed Storage (99.9999999999% durability)

BigQuery

Distributed Memory Shuffle Tier

Petabit Network

Analytic engine

High-Available Cluster Compute (Dremel)

SQL:2011 Compliant

REST API

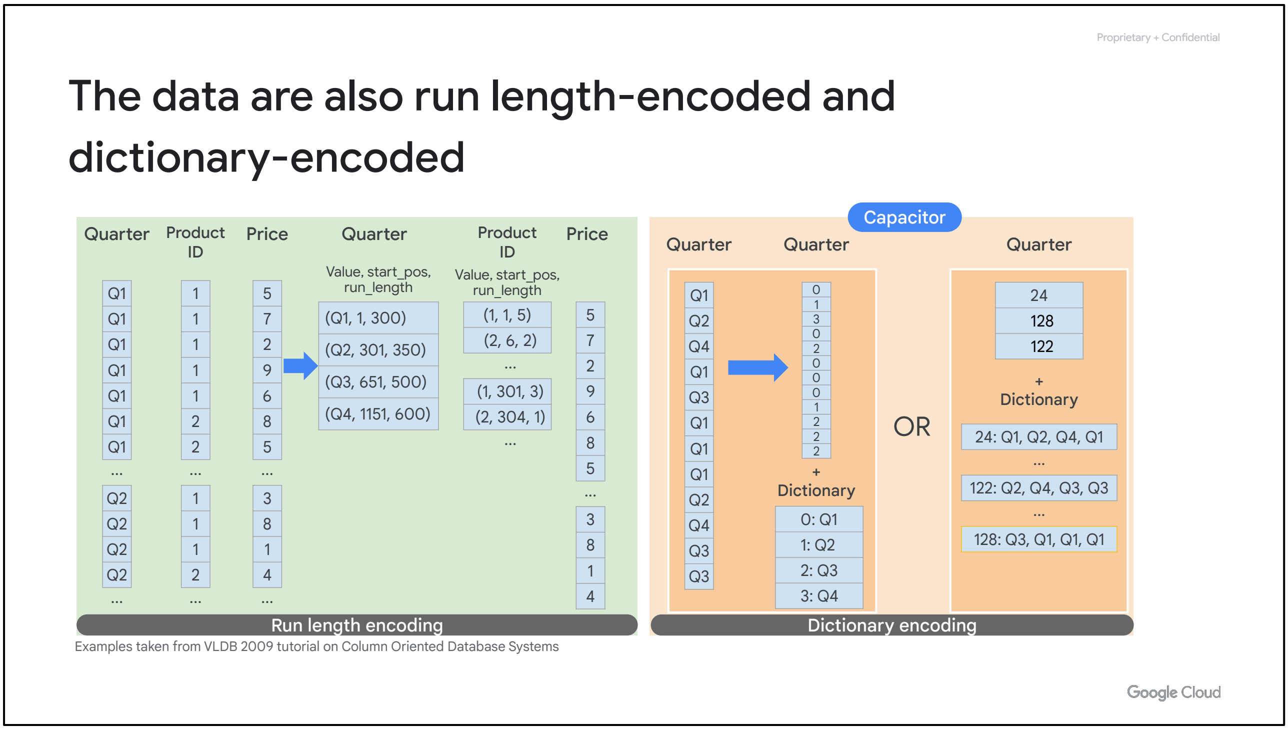
Web UI, CLI

Client Libraries In 7 languages

Google Cloud

BigQuery is implemented in two parts: a storage engine and an analytic engine as illustrated. The separation of compute and storage is a common theme in Google Cloud and works effectively because of Google’s petabit network called Jupiter. Jupiter allows blazing fast communication between compute and storage.

BigQuery data is physically stored on Google's distributed file system, called Colossus, which ensures durability by using erasure encoding to store redundant chunks of the data on multiple physical disks. Moreover, the data is replicated to multiple data centers.



The data are also run length-encoded and dictionary-encoded

Quarter	Product ID	Price
Q1	1	5
Q1	1	7
Q1	1	2
Q1	1	9
Q1	1	6
Q1	2	8
Q1	2	5
…	…	…
Q2	1	3
Q2	1	8
Q2	1	1
Q2	2	4
…	…	…

Quarter Value, start_pos, run_length	Product ID Value, start_pos, run_length	Price
(Q1, 1, 300)	(1, 1, 5)	5
(Q2, 301, 350)	(2, 6, 2)	7
(Q3, 651, 500)	…	2
(Q4, 1151, 600)	(1, 301, 3)	9
	(2, 304, 1)	6
	…	8
		5
		…
		3
		8
		1
		4

Run length encoding

Capacitor

Quarter	Quarter
Q1	0
Q2	1
Q4	3
Q1	0
Q3	2
Q1	0
Q1	0
Q1	0
Q2	1
Q4	2
Q3	2
Q3	2

+
Dictionary

0: Q1
1: Q2
2: Q3
3: Q4

OR

Quarter

24
128
122

+
Dictionary

24: Q1, Q2, Q4, Q1
…
122: Q2, Q4, Q3, Q3
…
128: Q3, Q1, Q1, Q1

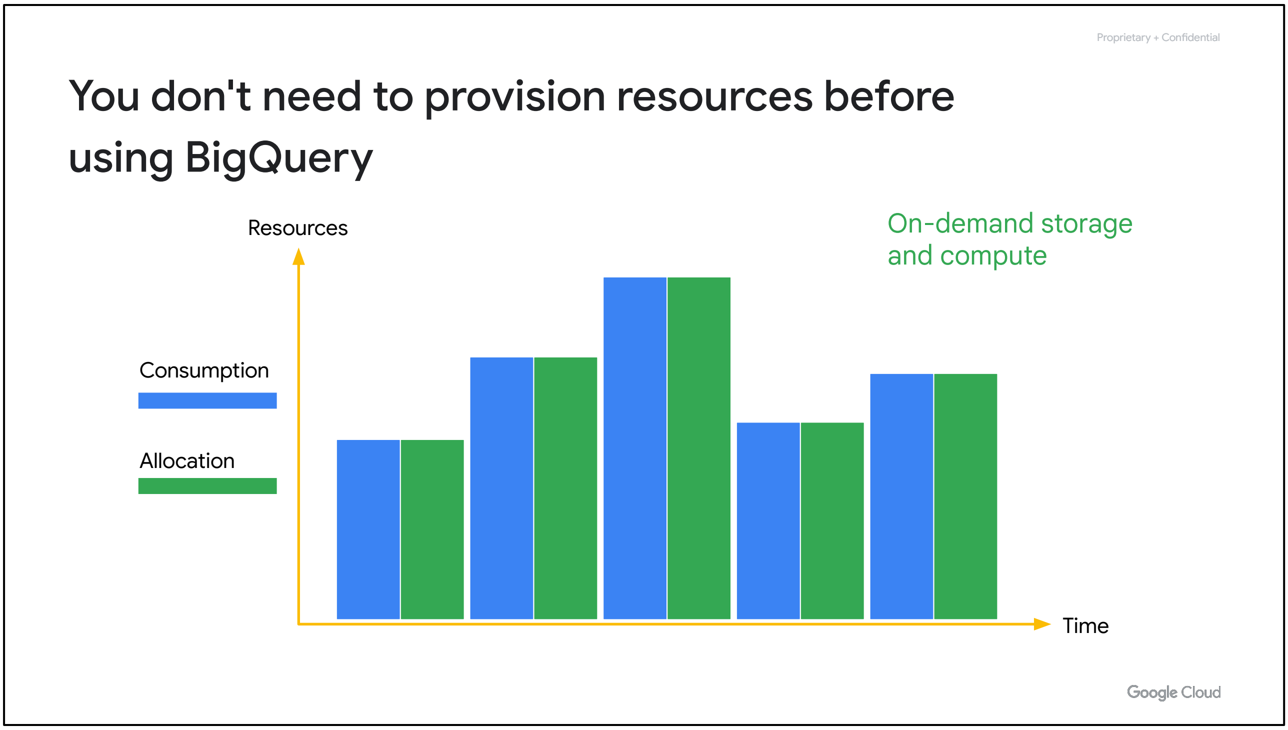
Dictionary encoding

Examples taken from VLDB 2009 tutorial on Column Oriented Database Systems

Google Cloud

Here are a couple of the optimizations that Capacitor does. Capacitor runs length-encodes on the data so that it can reduce the amount of data needed to be read. It also reorders the data to make it more conducive for run-length-encoding. Reordering the data is also called dictionary encoding.

All this “beneath the covers” happens in BigQuery native storage. It doesn’t affect you in any way. That’s the whole point of serverless and fully managed.



You don't need to provision resources before using BigQuery

On-demand storage and compute

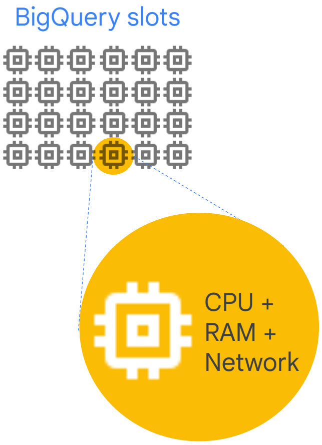
You don't need to provision resources before using BigQuery, unlike many RDBMS systems. BigQuery allocates storage and query resources dynamically based on your usage patterns.

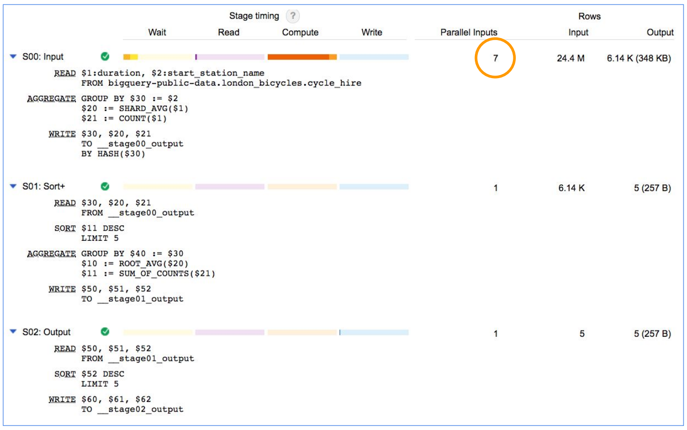
Storage resources are allocated as you consume them and deallocated as you remove data or drop tables.

Query resources are allocated according to query type and complexity. Each query uses some number of slots, which are units of computation that comprise a certain amount of CPU and RAM.

You don't have to make a minimum usage commitment to use BigQuery. The service allocates and charges for resources based on your actual usage. By default, all BigQuery customers have access to 2,000 slots for query operations. You can also reserve a fixed number of slots for your project.

A BigQuery slot is a combination of CPU, memory, and networking resources





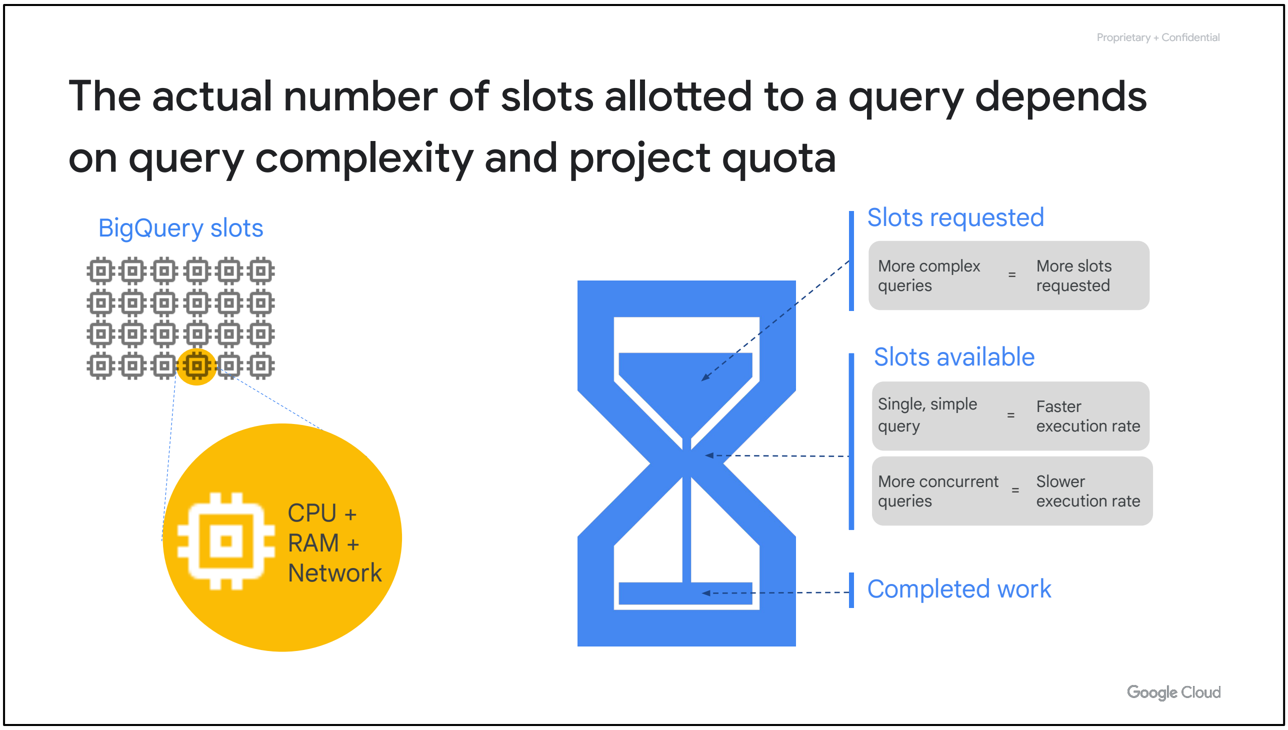
```
                         Stage timing ?                                      Rows
              Wait      Read      Compute      Write     Parallel Inputs    Input         Output
▼ S00: Input   ✓                                               7           24.4 M   6.14 K (348 KB)
       READ $1:duration, $2:start_station_name
            FROM bigquery-public-data.london_bicycles.cycle_hire
  AGGREGATE GROUP BY $30 := $2
            $20 := SHARD_AVG($1)
            $21 := COUNT($1)
      WRITE $30, $20, $21
            TO __stage00_output
            BY HASH($30)

▼ S01: Sort+   ✓                                               1           6.14 K        5 (257 B)
       READ $30, $20, $21
            FROM __stage00_output
       SORT $11 DESC
            LIMIT 5
  AGGREGATE GROUP BY $40 := $30
            $10 := ROOT_AVG($20)
            $11 := SUM_OF_COUNTS($21)
      WRITE $50, $51, $52
            TO __stage01_output

▼ S02: Output  ✓                                               1                5        5 (257 B)
       READ $50, $51, $52
            FROM __stage01_output
       SORT $52 DESC
            LIMIT 5
      WRITE $60, $61, $62
            TO __stage02_output
```

BigQuery is implemented using a microservice architecture, so there are no virtual machines to configure and maintain. Under the hood, analytics throughput is measured in BigQuery slots. A BigQuery slot is a unit of computational capacity required to execute SQL queries. BigQuery automatically calculates how many slots are required at each stage in a query, depending on size and complexity.

A BigQuery Slot is a combination of CPU, memory, and networking resources. It also includes a number of supporting technologies and sub-services. Note that each slot doesn’t necessarily have the same specification during query execution. Some slots may have more memory than others, or more CPU or more I/O.



The actual number of slots allotted to a query depends on query complexity and project quota

BigQuery slots

CPU + RAM + Network

Slots requested

More complex queries = More slots requested

Slots available

Single, simple query = Faster execution rate

More concurrent queries = Slower execution rate

Completed work

By default, each account has a quota limit of 2000 BigQuery slots for on-demand querying. A flat-rate pricing model is available that provides reserved slots for customers who want more predictable pricing.

If a single, simple query is submitted that needs fewer slots than are available, the query will generally execute faster.

If you have reserved 10,000 slots, but you have 30 concurrent queries that together ask for 15,000 slots, the queries will not get all the slots they require. Instead, the slots are divided fairly among all the projects in the reservation and all the queries in the project. This will generally result in each query executing more slowly.