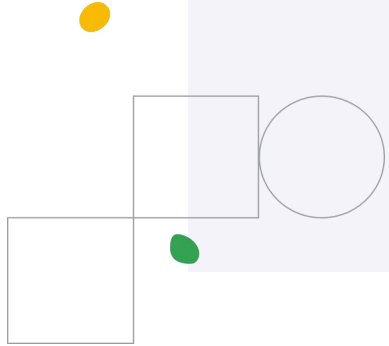
BigQuery Architecture Fundamentals

01



Google Cloud

Topics

01	Introduction to BigQuery
02	BigQuery Core Infrastructure
03	BigQuery Storage
04	BigQuery Query Processing
05	BigQuery Data Shuffling

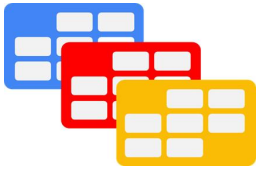
Google Cloud

Introduction to BigQuery

BigQuery is Google's data warehouse solution

Data warehouse

BigQuery replaces a typical data warehouse hardware setup.

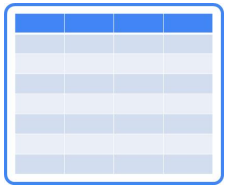


Data mart

BigQuery organizes data tables into units called datasets.

Data lake

BigQuery defines schemas and issues queries directly on external data sources.



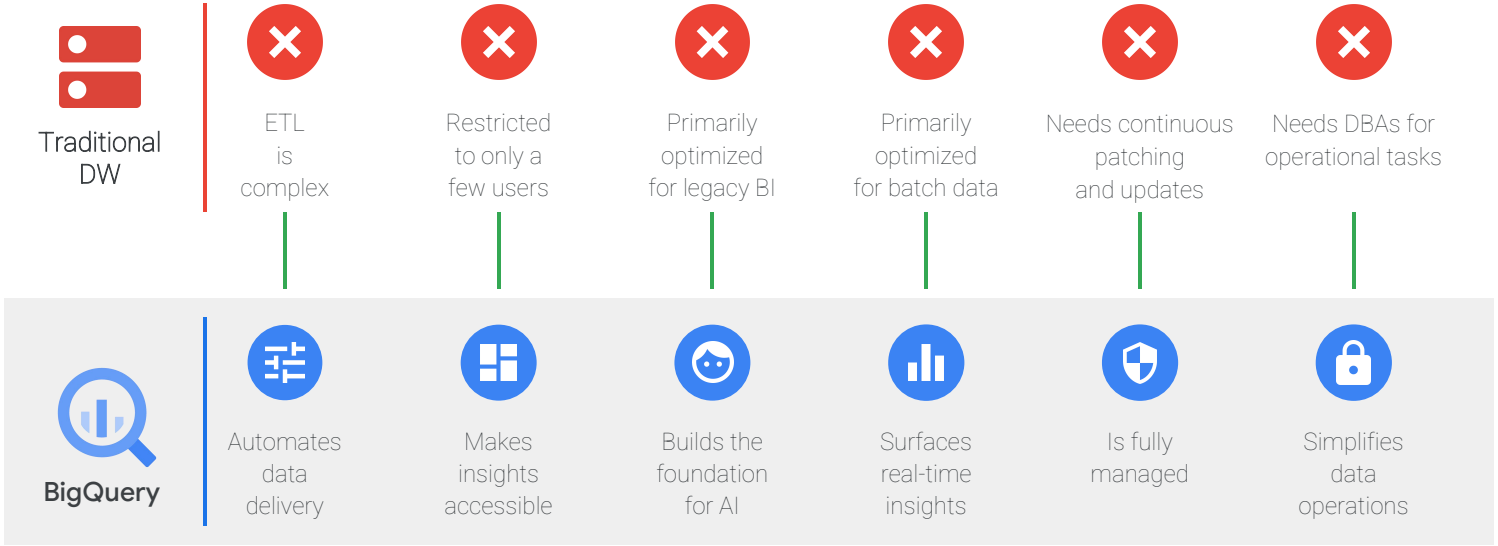
Tables and views

Tables and views function like a traditional data warehouse.

Grants

Cloud IAM grants permission to perform specific actions.

BigQuery is a modern data warehouse that changes the conventional mode of data warehousing



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

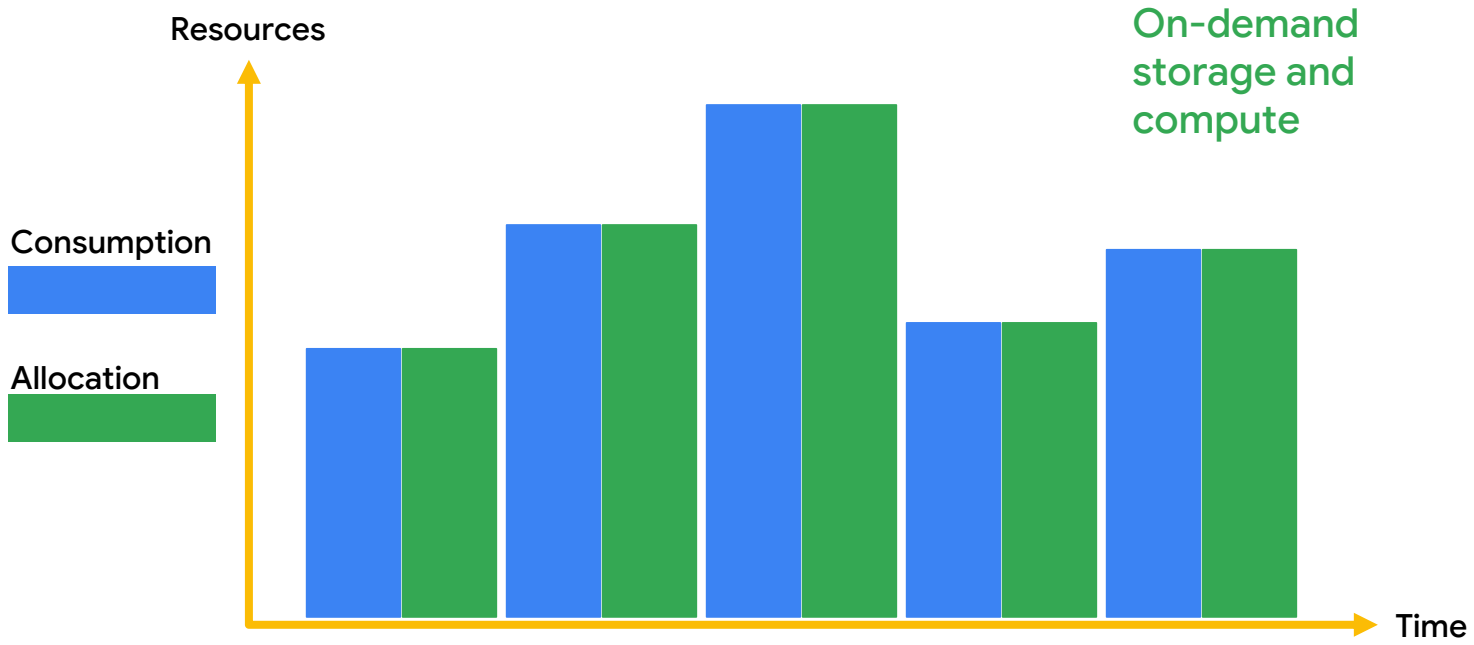
BigQuery

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 manage common tasks.

You don't need to provision resources before using BigQuery



Rule #1

Don’t optimize prematurely:

- Ignore all other best practices.
- Try out your query.
- If your query runs fast/inexpensive enough, leave it alone.

Fun with numbers

Analyze highly complex data at any scale

350 PB of data

Stored by one customer

100,000,000,000,000 rows
(one hundred trillion)

Queried by multiple customers

10,000 concurrent
queries

Run by another customer

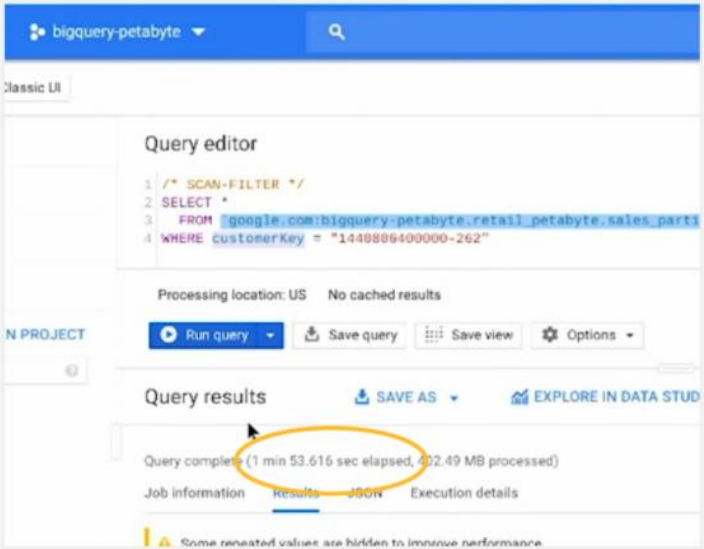
Rule #2

BigQuery is always getting better/faster. Read Rule #1.

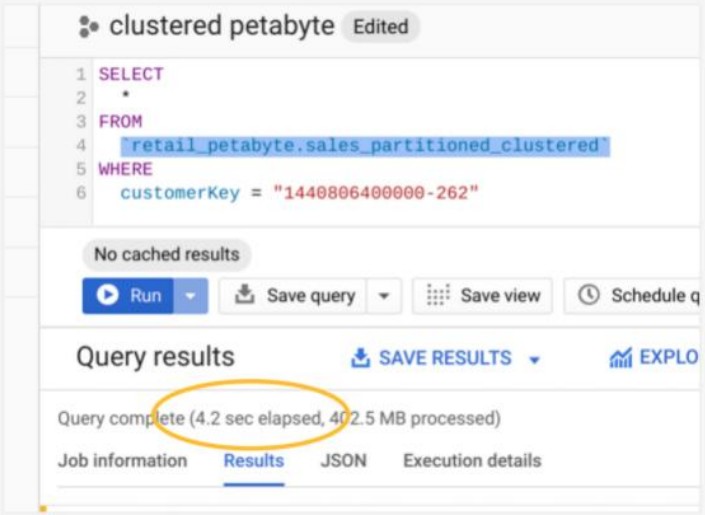
Continuous performance improvement

Performance

2018



2020



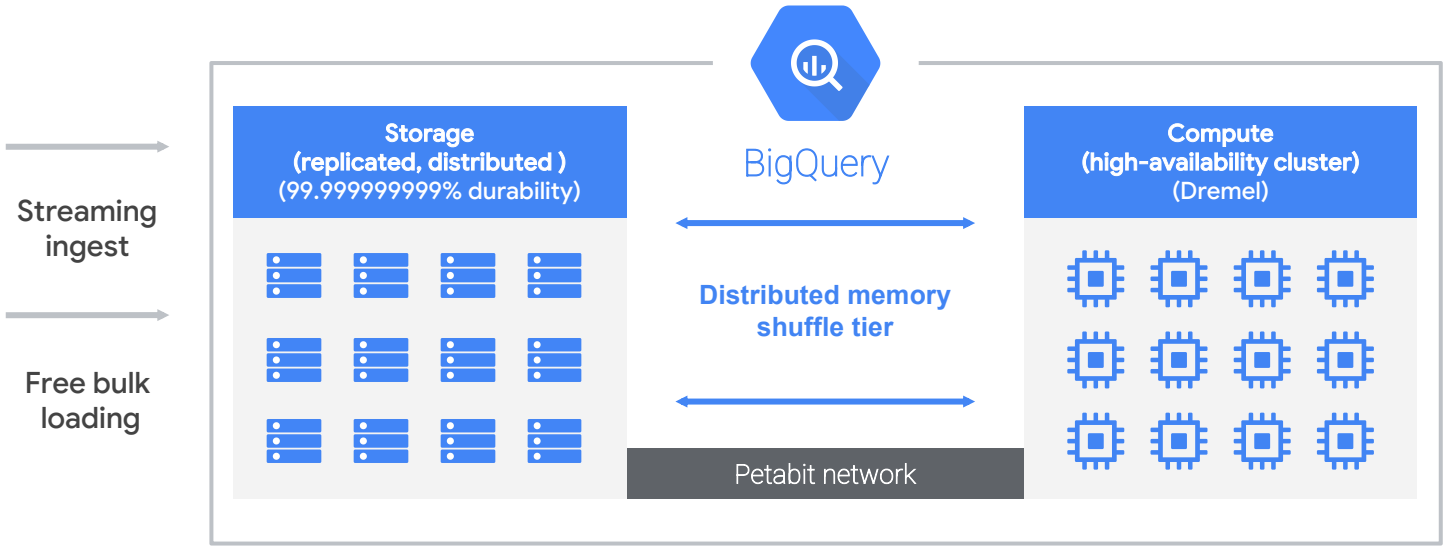
Rule #3

You're taking this course because Rules #1 and #2 are occasionally not enough.

BigQuery Core Infrastructure

Fully managed and serverless

Decoupled storage and compute for maximum flexibility



Key architecture design principles

Storage and compute separation

- Petabyte-scale
- High availability
- Serverless and multi-tenant

Integrated hardware/software stack

- Benefit from hardware primitives
- High performance at low cost

Colocation and caching

- High performance at low cost

Integration with Google Cloud

- Common security and privacy policies across products
- Seamless Google Cloud experience

BigQuery service locations

BigQuery is a **regional** service.

- **Regions** - (us-east4, europe-west4)
  - Multiple zones, one or more campuses, single metropolitan area, single jurisdiction
  - Data residence and colocation guarantees

- **Multi-Regions** - US, EU
  - Multiple zones, multiple campuses, multiple metropolitan areas
  - Flexible capacity planning
  - Generally less expensive
  - Improved durability due to off-region backups

Google Cloud zone: *A zone is a deployment area for Cloud Platform resources within a region. Zones should be considered a single failure domain within a region.*

BigQuery service deployment

Global layer

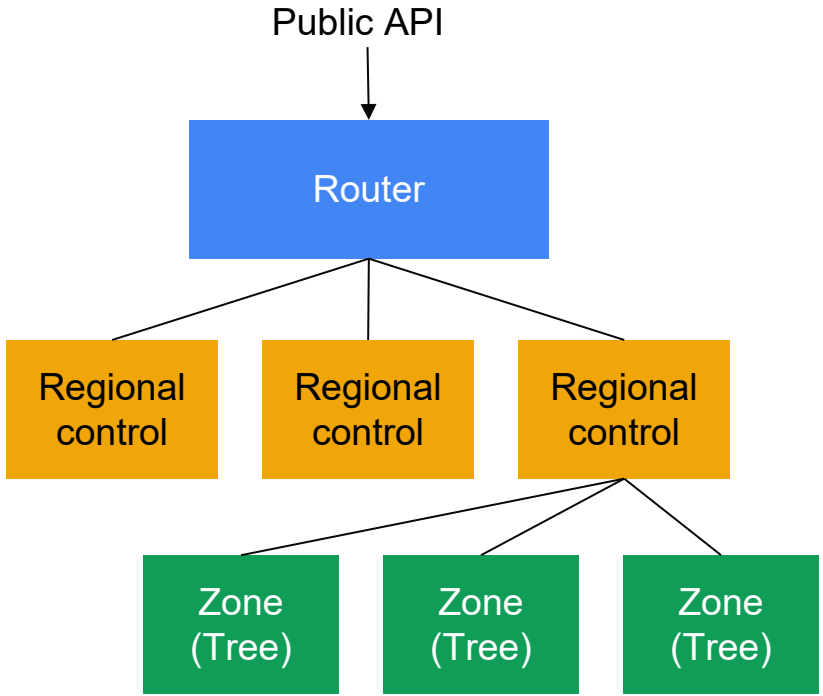
- Components needed to route to the right region
- Spread across zones globally

Regional layer

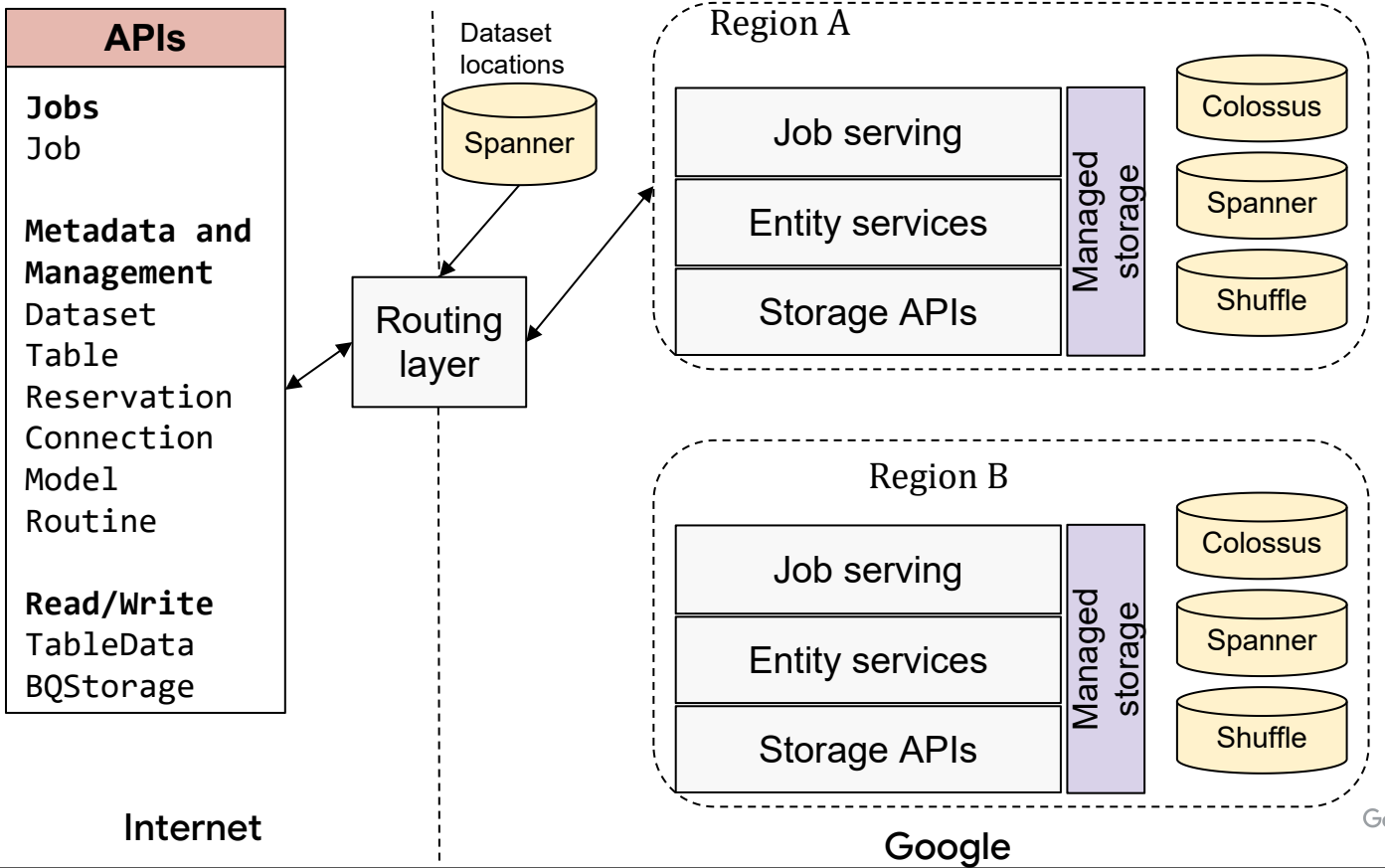
- Manage capacity, data, and metadata redundancy
- Spread across zones within the region

Zonal layer

- Compute and storage backend



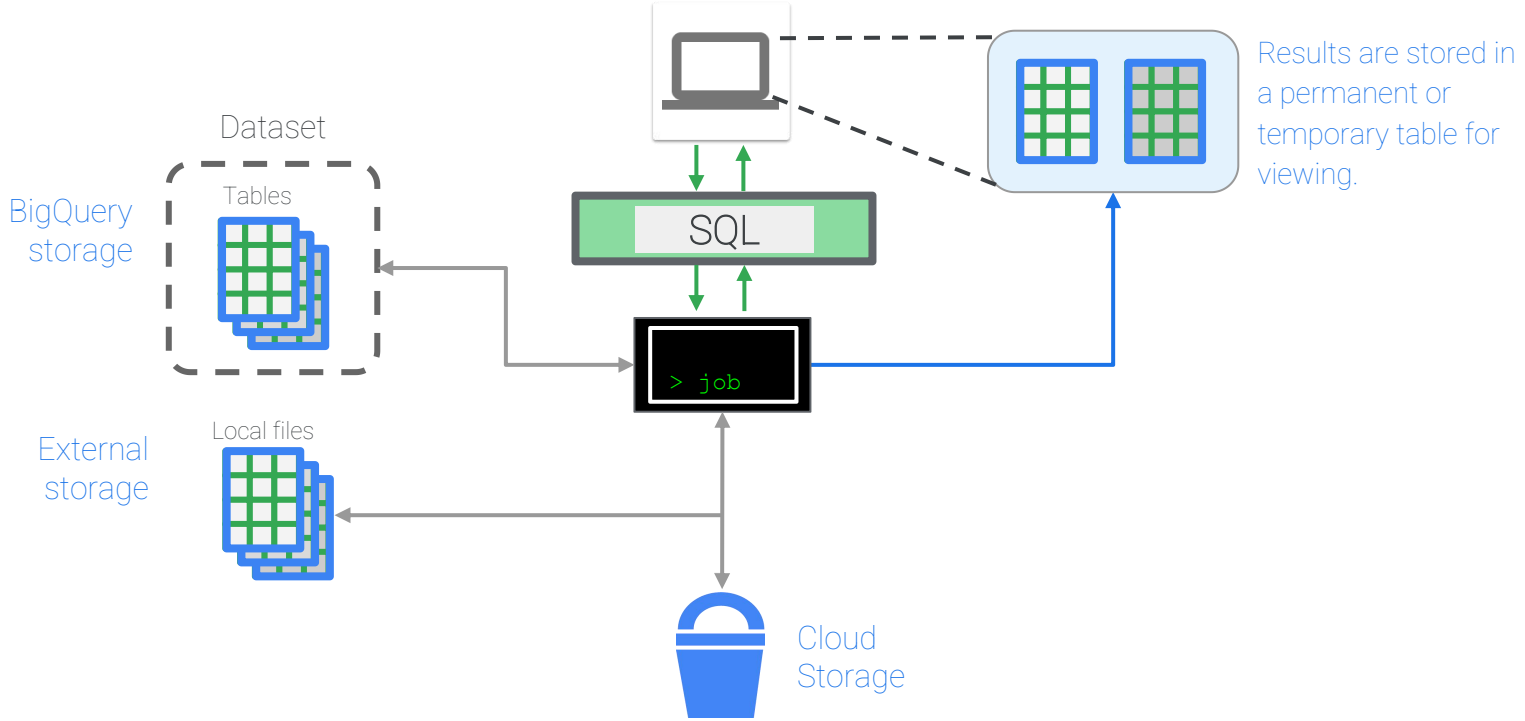
Service layers and components



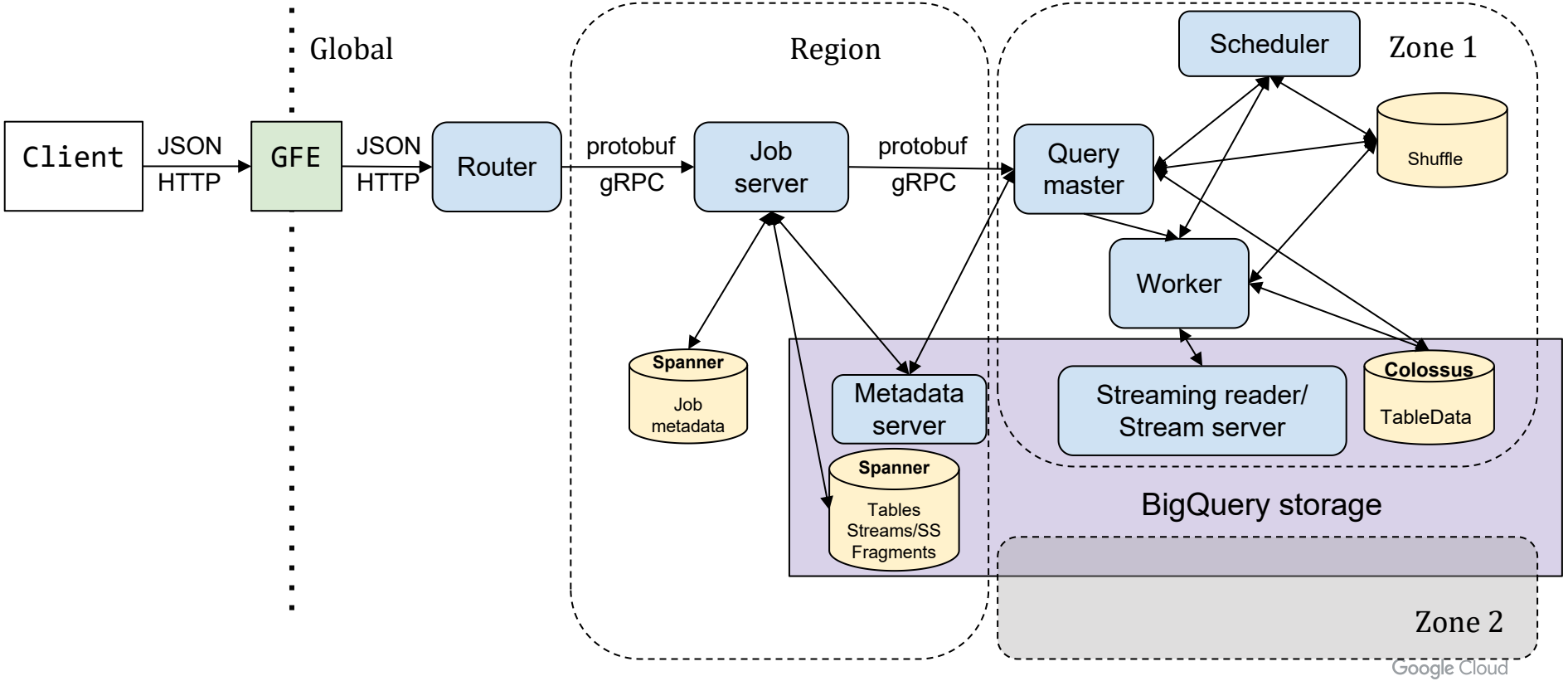
APIs

- **Job/Query** - Run a single SQL query or a script. Load or export data.
  - UI, ODBC/JDBC, Command line client, Looker
  - Example: jobs.insert(), jobs.query(), jobs.getQueryResults()
- **Storage** - Read from and write to BigQuery tables.
  - UI, Dataflow and Dataproc, custom code
  - Example: tabledata.insertAll().
- **Metadata** - Create dataset. Add a routine or script.
  - UI, SQL, CLI
  - Example: datasets.insert(), tables.list()
- **Management** - Create and modify reservations.
  - UI, CLI

The life of a BigQuery SQL query

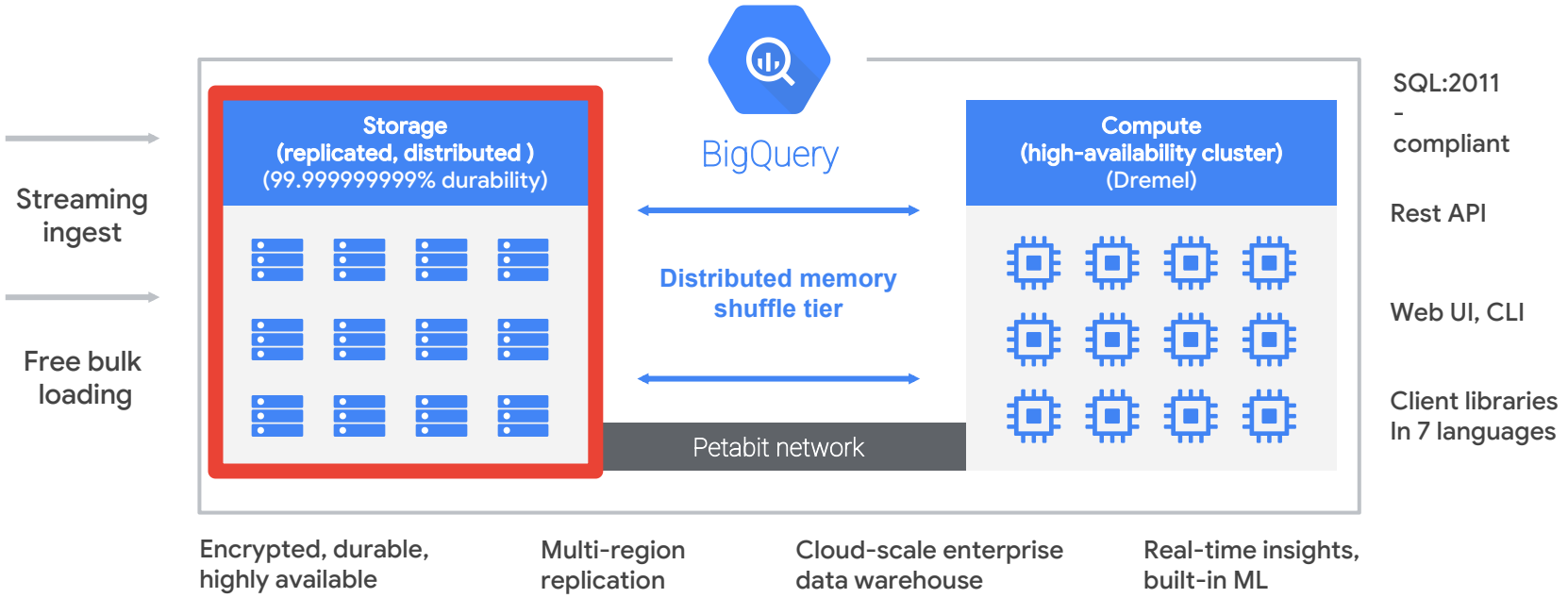


The internal life of a BigQuery SQL query

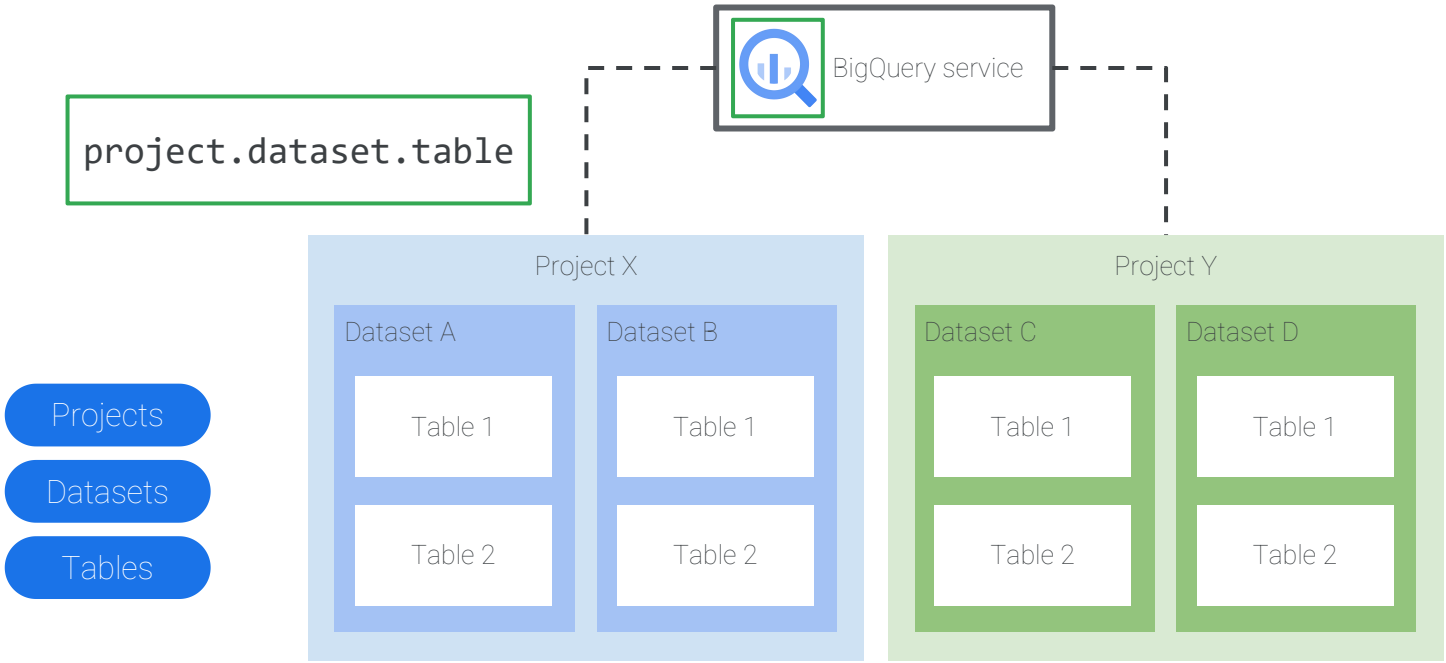


BigQuery Storage

BigQuery architecture



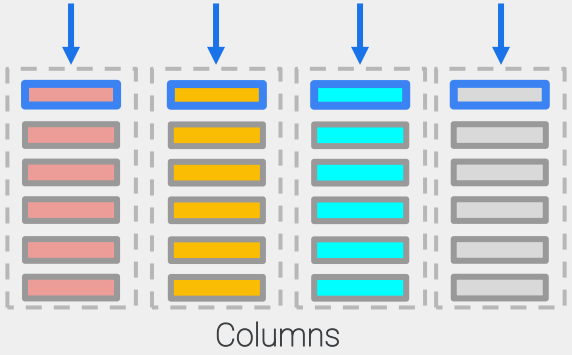
BigQuery organizes data tables into units called datasets



What makes BigQuery fast?

Relational database tables are row-oriented

BigQuery tables are column-oriented



Column-oriented versus row-oriented storage

- Read less data faster.
- Skip unused columns.
- Column compression versus row compression.
- Supports vectorized columnar processing.

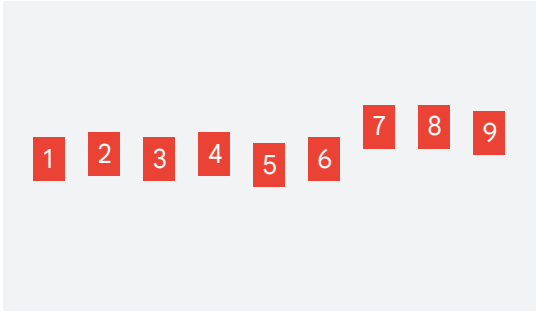
BigQuery divides all tables into smaller shards of data to enable massively parallel reads and operations

One large table

Company ID	Company Name
161218560	NY Association Inc.
...	...
...	...
...	...

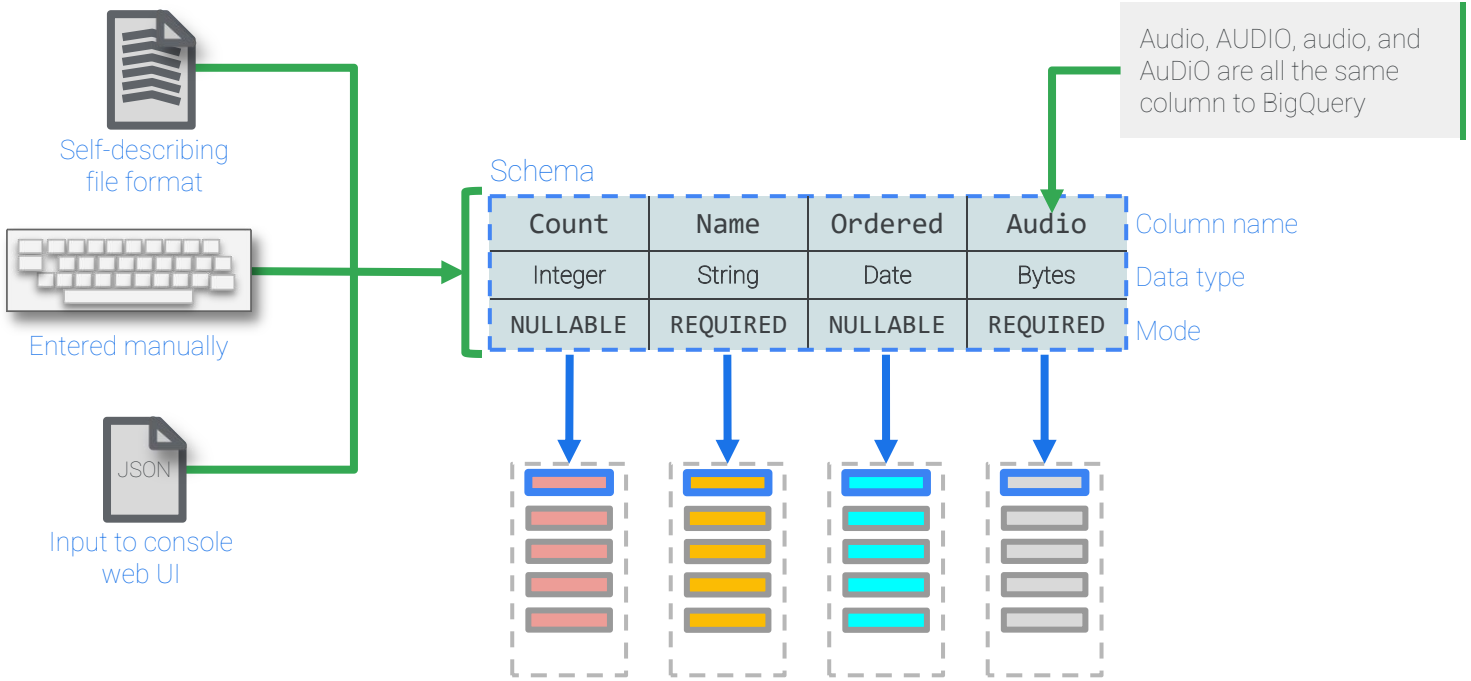
One table with 10 billion rows

BigQuery divides the table (not seen to you) into many small shards.



10 billion rows spread across many shards

The table schema provides structure to the data



Physical layout

Capacitor: Our proprietary columnar format

- Maintain the optimum sharding structure.
- Implement the logical metadata hints: partitioning/clustering.

Why create a new proprietary format?

- Google can apply what it learned over the last 10 years.
- The format is deeply tied to the execution engine.
- Google can improve it easily.

Capacitor features

- Dictionary encoding (low cardinality)
- Constraints and Bloom filters (high cardinality)
- Run-length encoding
- Compression
- Row reordering

[see this article]

Physical metadata

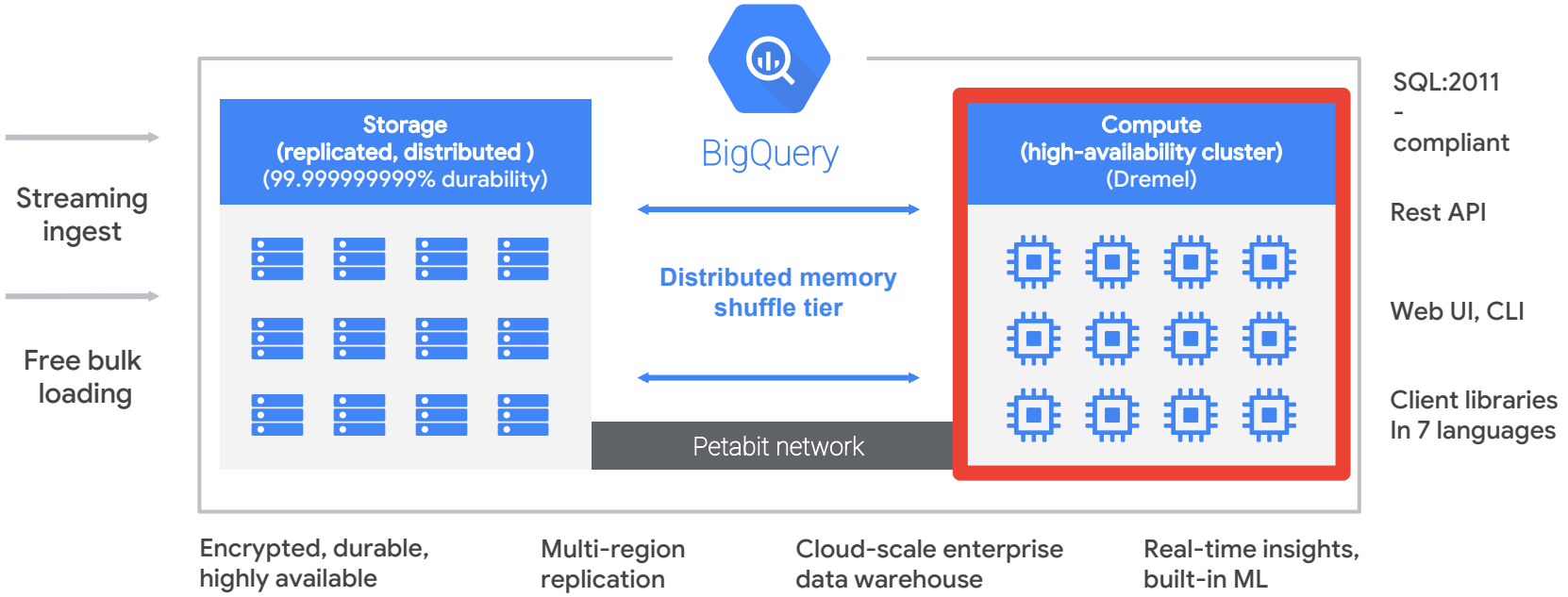
Critical part of BigQuery storage that is designed to support:

- Streaming
- ACID commits
- Time travel (7 days)
- Backups
- Active storage management
- Storage optimization
- Partitioning and clustering
- DML

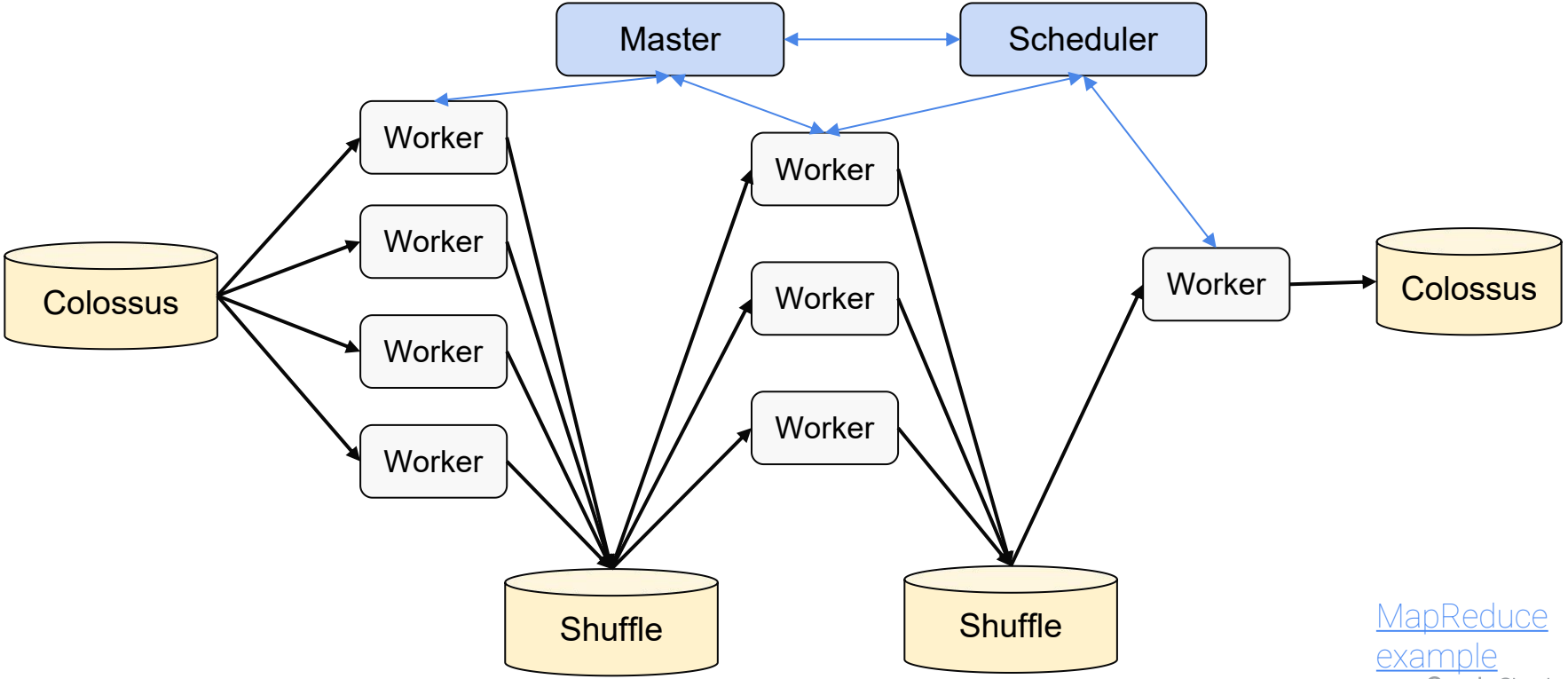
```
CREATE OR REPLACE TABLE restored_table AS
SELECT
  *
FROM current_table
  FOR SYSTEM_TIME AS OF
  TIMESTAMP_SUB(CURRENT_TIMESTAMP(),
  INTERVAL 24 HOUR)
```

BigQuery Query Processing

BigQuery architecture

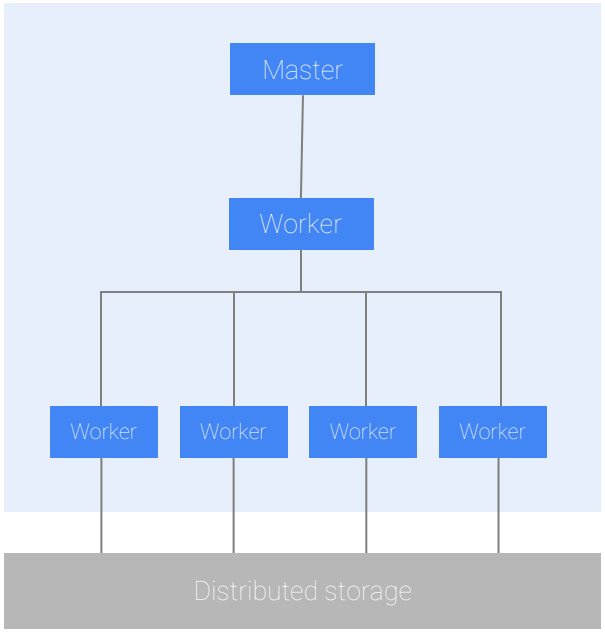


BigQuery processes data in a distributed way



MapReduce example

Simple query execution

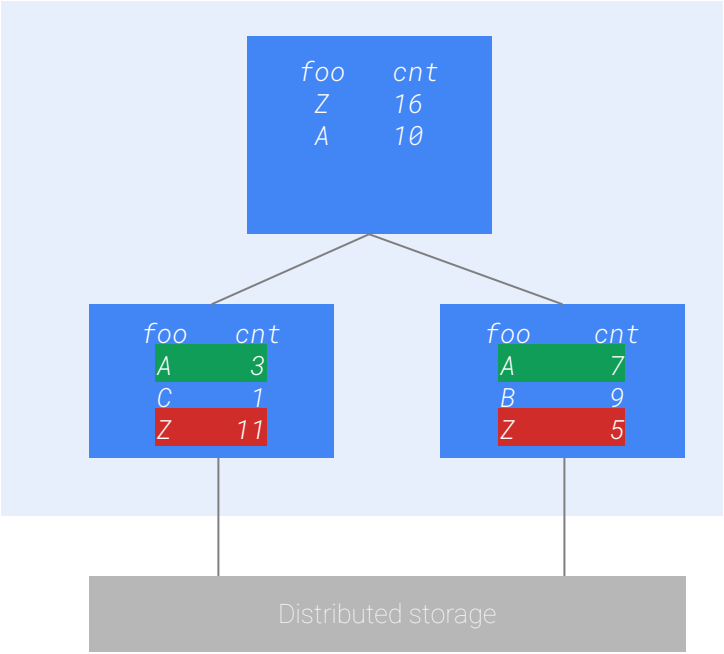



```
SELECT COUNT(*) FROM
wikipedia_benchmark.Wiki1B
WHERE title LIKE "G%o%o"
```

Stage 2: Sum

Stage 1: Filter, Count

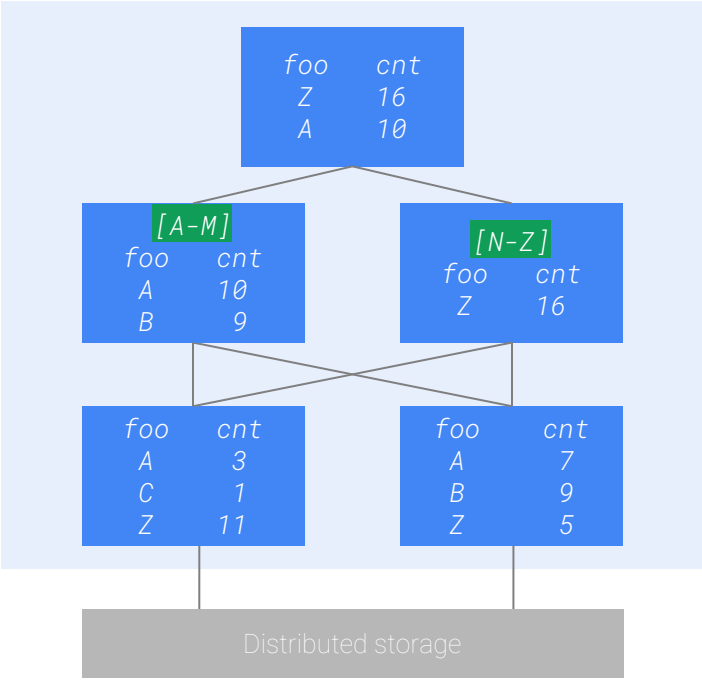
Aggregation with high cardinality




```
SELECT foo, COUNT(*) as cnt
FROM `...`
GROUP BY 1
ORDER BY 2 DESC
LIMIT 2
```

- Can't discard 'B' or 'C' until after all previous stages are complete.
- High cardinality 'foo' will overwhelm the root node.

Aggregation with shuffle

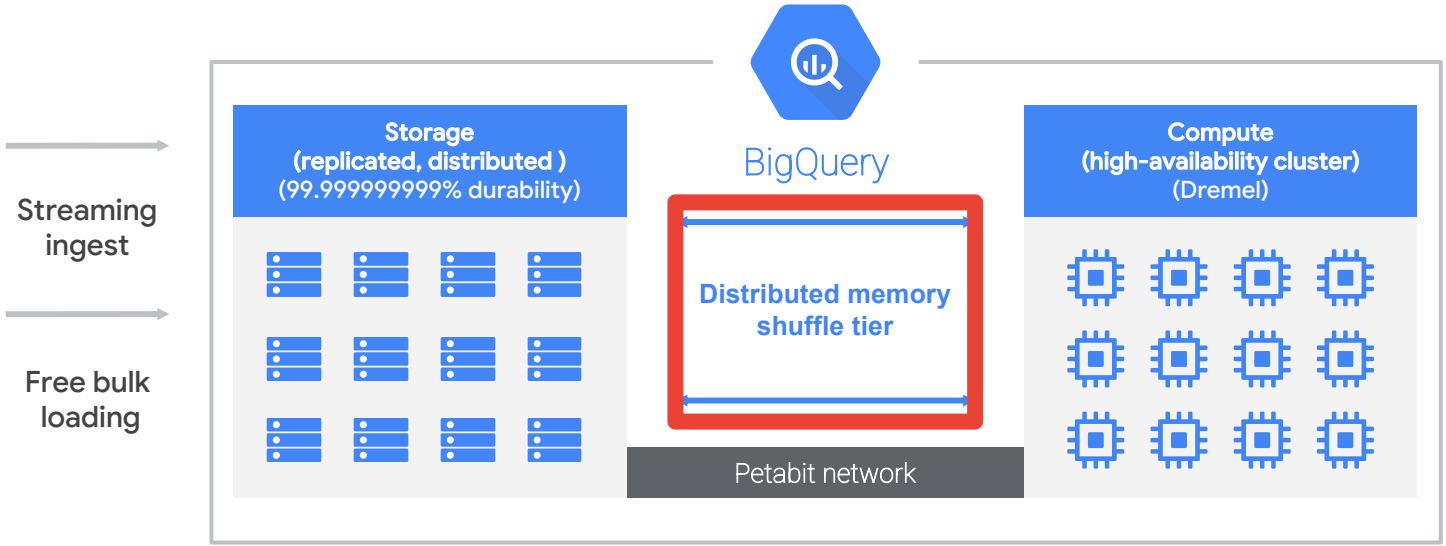



```
SELECT foo, COUNT(*) as cnt
FROM `...`
GROUP BY 1
ORDER BY 2 DESC
LIMIT 2
```

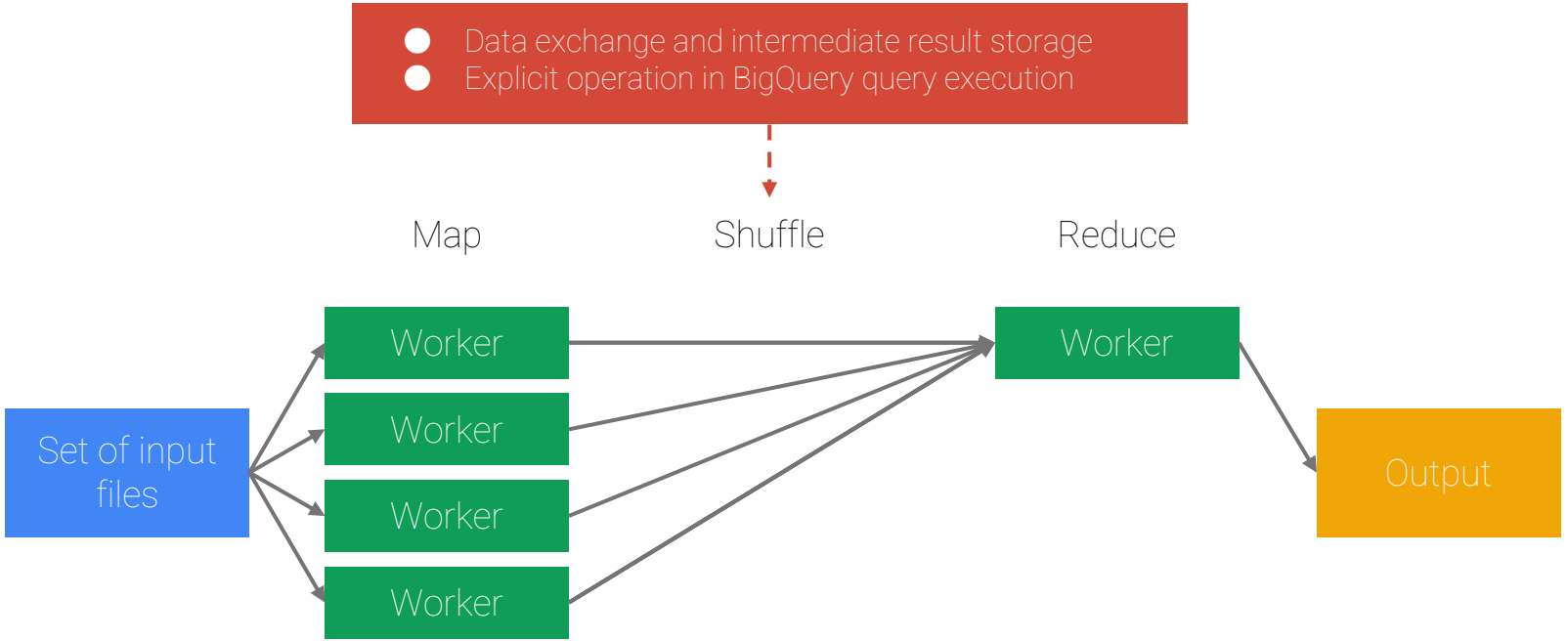
- Shuffle puts like values in the same node.
- This is scalable because you never have to return more than the LIMIT value from each node in the middle tier.

BigQuery Data Shuffling

BigQuery architecture

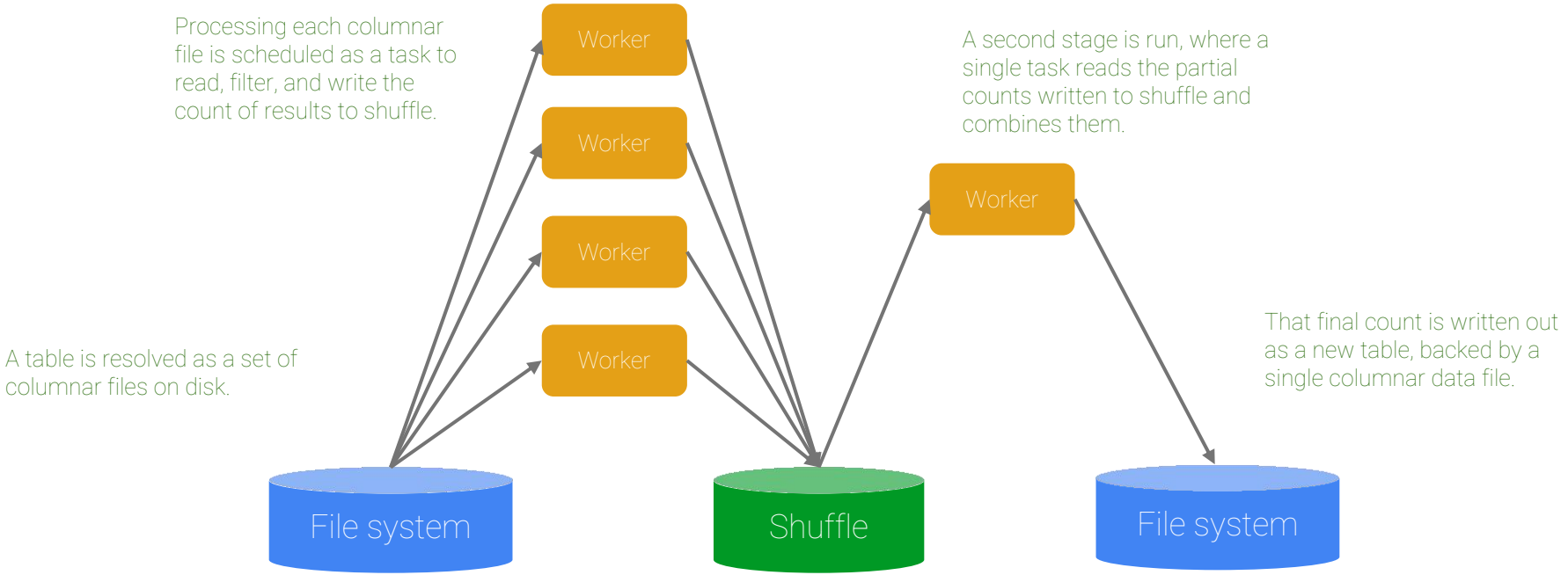


Parallel to Hadoop/MapReduce



MapReduce example

With more detail



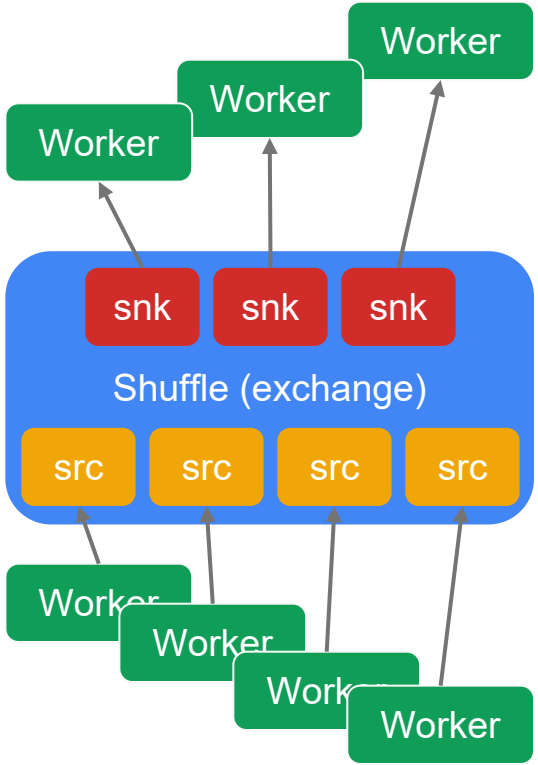
Parallel to traditional parallel query execution

Shuffle is similar to Exchange operator

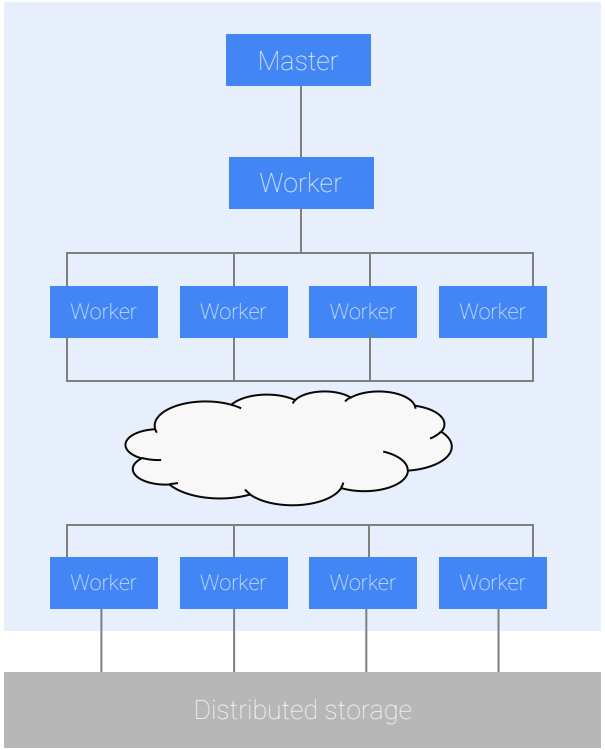
Pluggable way of changing degree of parallelism in query execution

- M inputs, N (disjoint) outputs
  - Read data from multiple (M) inputs.
  - Determine receiving output through some partitioning scheme (e.g., hash, range).
  - Write data to N outputs.

- The shuffle is the BigQuery-specific implementation of an exchange.
  - *Sources* model the exchange inputs.
  - *Sinks* model the exchange outputs.
  - Data reads and writes are orchestrated through Mindmeld, a distributed in-memory file system.



Shuffle aggregation execution




```
SELECT language, MAX(views) as views
FROM `wikipedia_benchmark.Wiki1B`
WHERE title LIKE "G%o%"
GROUP BY 1 ORDER BY 2 DESC LIMIT 100
```

Stage 3: SORT, LIMIT (1 worker)

Stage 2: GROUP BY, SORT, LIMIT (289 workers)

Shuffle

Stage 1: Partial GROUP BY (40,859 inputs)

Shuffle aggregation execution

Stage 1

Row	language	views	
1	hr	131	
2	tl	160	
3	id	257	max
4	id	114	
5	da	101	
6	da	124	max
7	meta	166	

Stage 2

Row	language	views	
1	en	89322	max
2	en	47900	
3	en	43067	
4	en	38988	
5	en	37628	
6	en	37089	
7	en	35870	

Stage 3

Row	language	views
1	en	89322
2	it	38611
3	de	27715
4	pt	22974
5	tr	13552
6	fr	12447
7	meta	11117

Query execution design choices

Shuffle is the **data transfer mechanism** between workers:

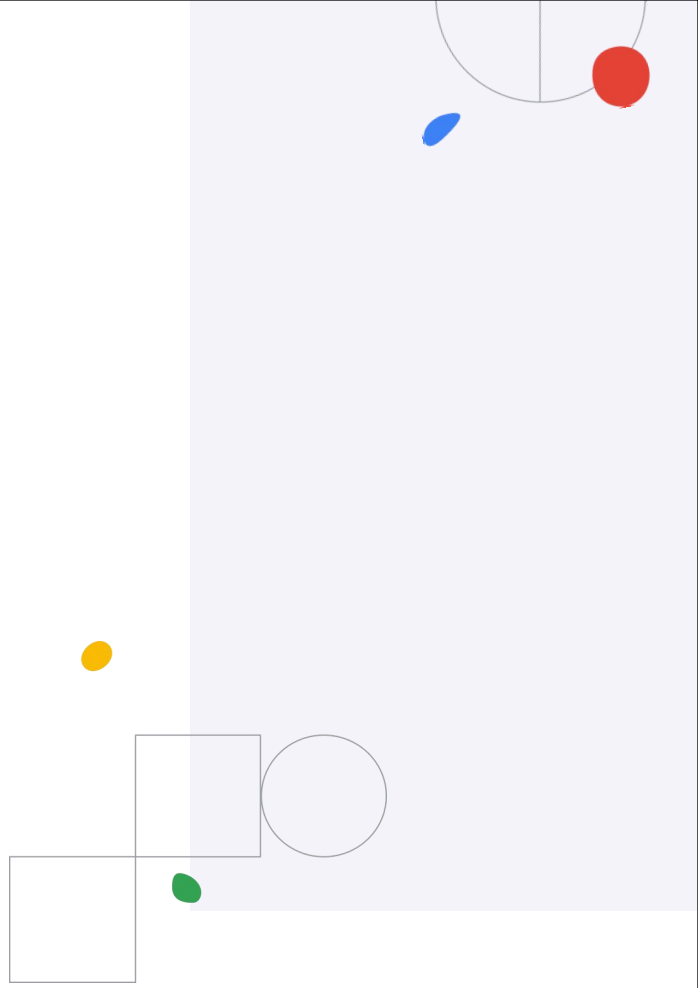
- Allows flexible query planning and execution.
- Can act as a staging area or partitioning mechanism.

Query optimization using **dynamic query execution**:

- Observe execution and quickly react.
- Is more robust than static (cost-based) query optimization.

Decouple **scheduling** from query **planning**.

Questions?



Google Cloud

Lab Intro

Using BigQuery to Do Analysis

Objectives

- Execute interactive queries in the BigQuery console.
- Combine and run analytics on multiple datasets.