BigQuery Best Practices

Google Cloud Platform

Agenda

Architecture Recap

Data Ingestion

Schema Design

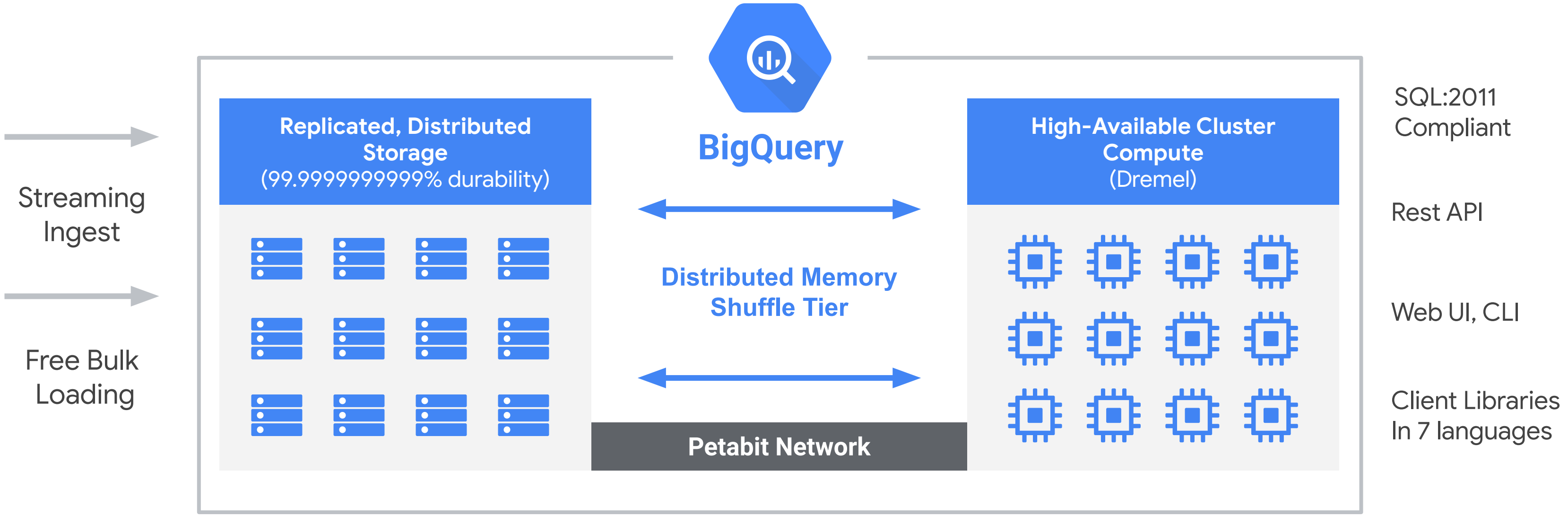
Workload Management

Auditing

Architecture Recap

BigQuery | Fully managed and Serverless

Decoupled storage and compute for maximum flexibility



Streaming Ingest
Free Bulk Loading
Replicated, Distributed Storage (99.9999999999% durability)
BigQuery
Distributed Memory Shuffle Tier
Petabit Network
High-Available Cluster Compute (Dremel)
SQL:2011 Compliant
Rest API
Web UI, CLI
Client Libraries In 7 languages

Encrypted, durable, highly available

Multi-region Replication

Cloud-scale enterprise data warehouse

Real-time insights, built-in ML

Google Cloud

Data Ingestion

Data Ingestion | Options

Batch ingestion

Data from GCS or via HTTP POST

Multiple File Formats Supported

Snapshot-based arrival - All Data arrives at once, or not at all

Streaming ingestion

Continuous ingestion from many sources (web/mobile apps, point of sale, supply chain)

Immediate query availability from buffer

Deferred creation of managed storage

Query materialization

SELECT results yield data in the form of tables, either anonymous (cached) or named destinations

ETL/ELT Ingest + Transform via Federated Query

Data Transfer Service (DTS)

Managed ingestion of other sources (doubleclick, adwords, youtube)

Scheduled Queries, Scheduled GCS Ingestion

Options for third-party integration

Data Ingestion | Load Jobs

Batch Ingest is free

Doesn't consume query capacity

ACID semantics

Load Petabytes per day

Streaming API for real-time

Google Cloud

Best Practice | Data Format

Faster

Avro (Compressed)

Avro (Uncompressed)

Parquet / ORC

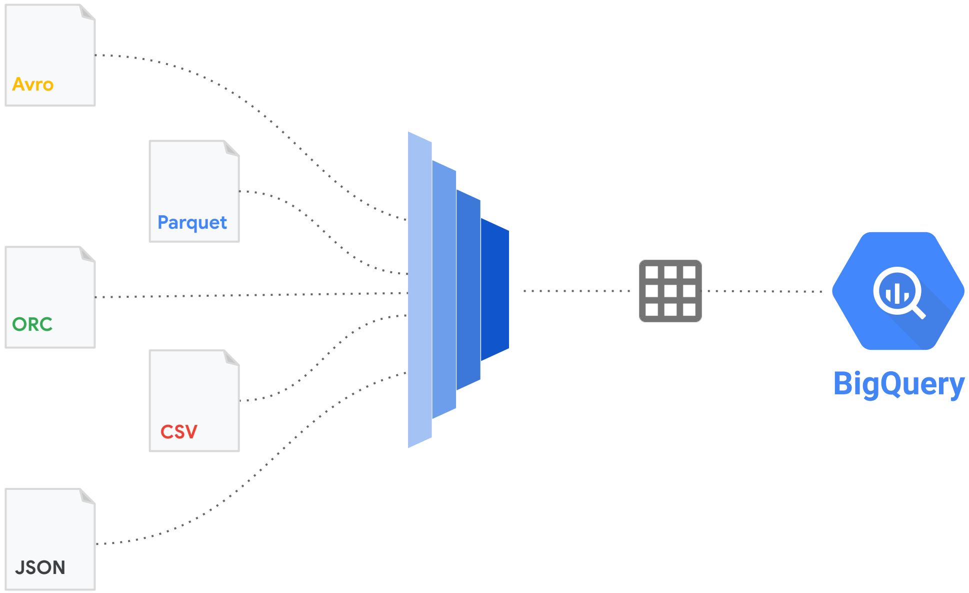
CSV

JSON

CSV (Compressed)

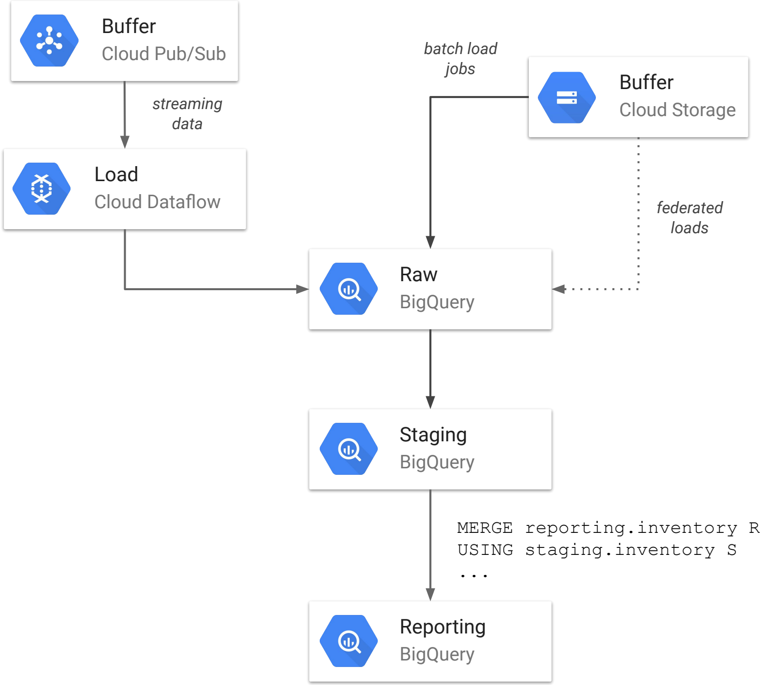
JSON (Compressed)

Slower



Google Cloud

Best Practice | ELT / ETL



```
MERGE reporting.inventory R
USING staging.inventory S
...
```

Prefer ELT into BigQuery over ETL where possible

Leverage federated queries to GCS to load and transform data in a single-step

Load data into raw and staging tables before publishing to reporting tables

Utilize Dataflow for streaming pipelines and to speed up large complex batch jobs

Get started streaming using Google-Provided Dataflow Templates and modify the open-source pipeline for more complex needs

Schema Design

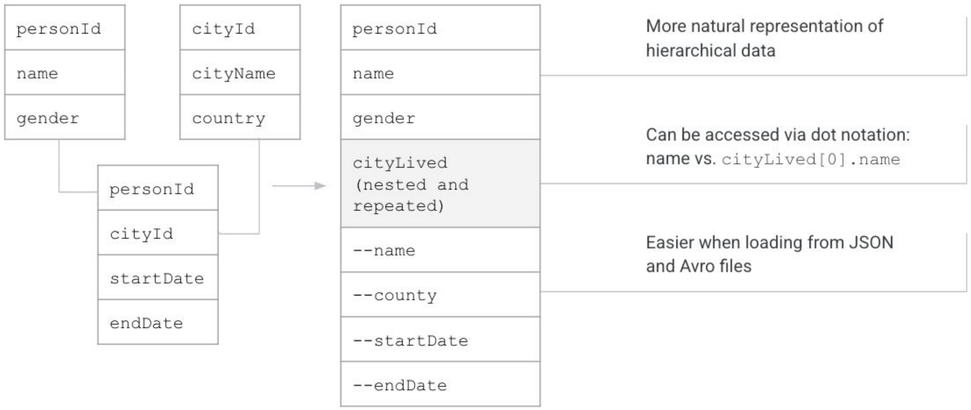
Schema Design | Overview

Migrate your Star or Snowflake schemas **as-is** to BigQuery and optimize as necessary

Denormalization is **NOT** a requirement but can speed up slow analytical queries by reducing the amount of data to shuffle

Leverage Nested and Repeated fields for:

- Tightly-coupled or Immutable relationships
  - Session w/ Events
  - Order w/ Line Items
- Infrequently changing data
  - Country, Region, Date, etc
- Simplifying queries



personId
name
gender

cityId
cityName
country

personId
cityId
startDate
endDate

personId
name
gender
cityLived (nested and repeated)
--name
--county
--startDate
--endDate

More natural representation of hierarchical data

Can be accessed via dot notation: name vs. `cityLived[0].name`

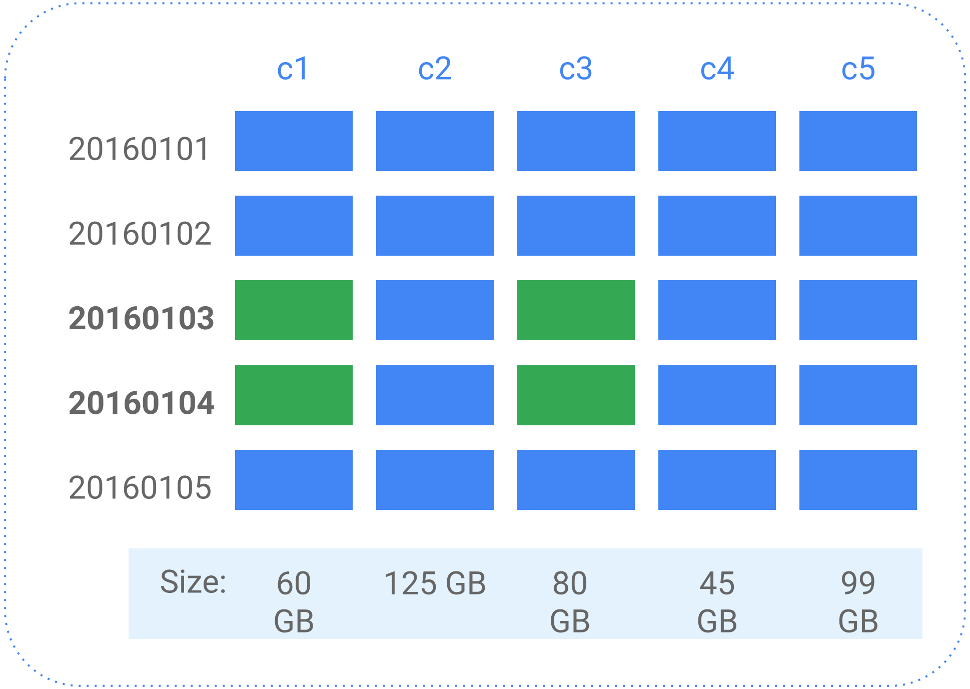
Easier when loading from JSON and Avro files

Best Practice | Partitioning

Efficiently query over the parts of the table you want and cut costs on the data read.

Use the “Require Partition Filter” option for **large** Fact tables

```
SELECT
  c1, c3
FROM
  dataset.table
WHERE
  eventDate IN
     (‘20160103’, ‘20160104’)
```



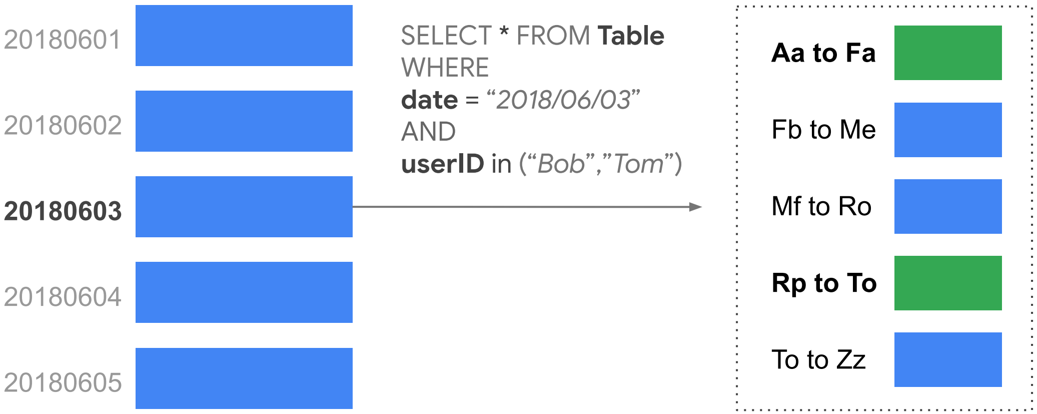
	c1	c2	c3	c4	c5
20160101	🟦	🟦	🟦	🟦	🟦
20160102	🟦	🟦	🟦	🟦	🟦
20160103	🟩	🟦	🟩	🟦	🟦
20160104	🟩	🟦	🟩	🟦	🟦
20160105	🟦	🟦	🟦	🟦	🟦
Size:	60 GB	125 GB	80 GB	45 GB	99 GB

Best Practice | Cluster Frequently Accessed Fields

Use on high-cardinality fields for up to 10x+ performance

Filter clustered columns in the order they're specified

Use natural partitioning or a fake date column to cluster non-date partitioned tables



SELECT * FROM **Table**
WHERE
date = *"2018/06/03"*
AND
userID in (*"Bob"*,*"Tom"*)

Google Cloud

Best Practice | Surrogate Keys

via UUID

```
SELECT
  GENERATE_UUID() AS SurrogateKey,
  *
 FROM
  `project.dataset.table`
```

via Hashing

```
SELECT
  (SHA256(bizKey)) AS SurrogateKey,
  *
FROM
  `project.dataset.table`
```

Surrogate keys substitute for natural keys and have no business meaning

Avoid using ROW_NUMBER() to generate surrogate keys

Prefer UUIDs in place of sequenced surrogate keys

Prefer hashing for deterministic surrogate keys derived from the business key

Design Pattern | Partitioning Types

- **Ingestion date/time partitioning**
  - Based on date/time that data is loaded
  - Filter using pseudo-columns: **_PARTITIONDATE**, **_PARTITIONTIME**
    - SELECT col FROM d.t WHERE **_PARTITIONDATE** > “2018-05-01”
    - Note: Streaming buffer has NULL values in pseudo-columns

- **Column partitioning**
  - Supported column types
    - TIMESTAMP, DATE
    - INT64
  - Filter using column name
    - SELECT COUNT(*) FROM d.t WHERE datecol > “2018-05-01”

Note: 4000 partition limit per table
Can be increased to 10,000 on request.

Design Pattern | Authorized Views

Authorized views permit users to query a view without having read access to the underlying tables

Create logical views into your dataset that filter based on user roles for more **fine-grained** access control

Authorized views must be placed in a dataset separate from the restricted dataset

Use SESSION_USER() to add your own custom authorization logic. For example, JOIN against user lookup table to manage access by user

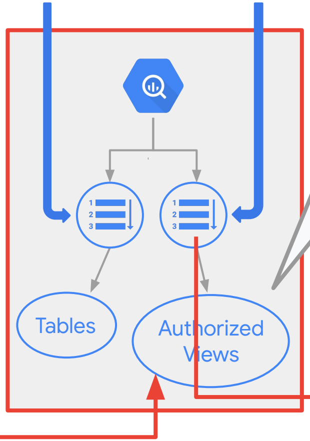
```
#standardSQL
SELECT column1, column2
FROM `dataset.table`
WHERE allowed_viewer = SESSION_USER()
```

Best Practice | Authorized Views

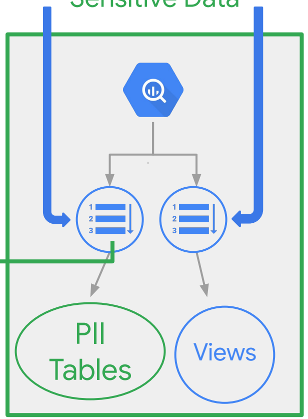
```
# Returns only
# non-sensitive data
SELECT *
FROM non_pii_view
```

```
# Authorized View CREATE SQL
CREATE VIEW non_pii_view AS(
  SELECT * EXCEPT(
    pii_col1,
    pii_col2
  )
  FROM pii_table
)
```

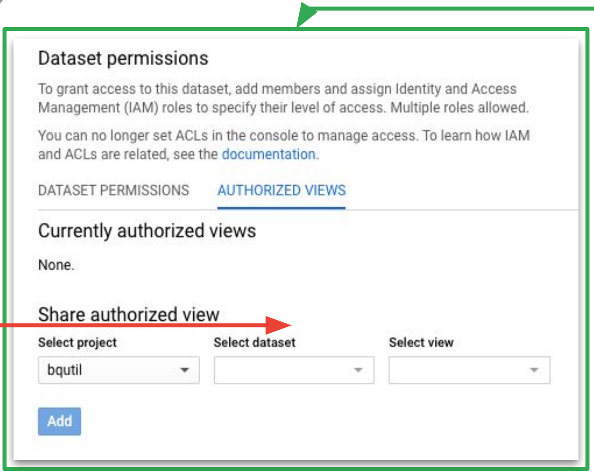
BigQuery Datasets:
Non-sensitive Data



BigQuery Datasets:
Sensitive Data



SHARE DATASET



Dataset permissions

To grant access to this dataset, add members and assign Identity and Access Management (IAM) roles to specify their level of access. Multiple roles allowed.

You can no longer set ACLs in the console to manage access. To learn how IAM and ACLs are related, see the documentation.

Currently authorized views

None.

Share authorized view

Select project	Select dataset	Select view
bqutil		

Google Cloud

Best Practice | Materialized Views for Reporting

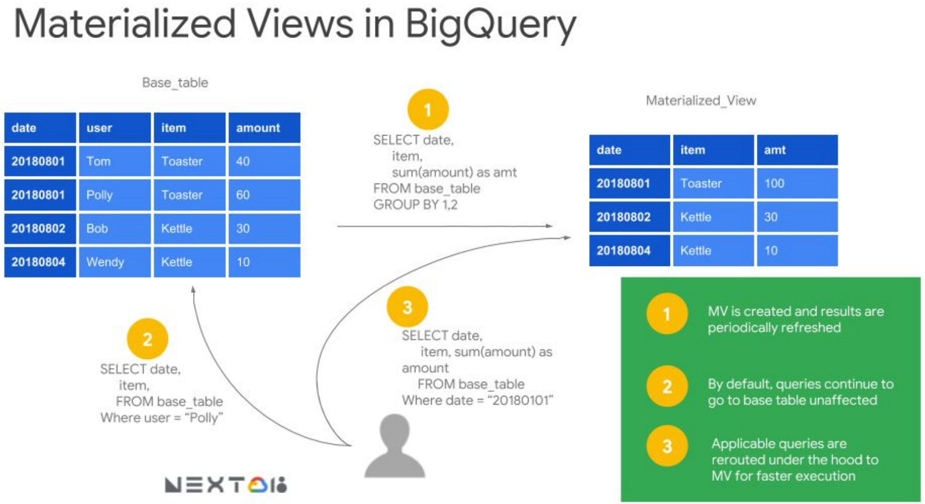
Improve query efficiency (slots usage and bytes processed) and reduce the execution time for complex queries with aggregate functions.

Automatically improve query execution plan.

Inherits the same benefits of resilience and high availability of BigQuery tables.

Real-time aggregation

Analytical queries against large data in size



Materialized Views in BigQuery

Base_table

date	user	item	amount
20180801	Tom	Toaster	40
20180801	Polly	Toaster	60
20180802	Bob	Kettle	30
20180804	Wendy	Kettle	10

1

```
SELECT date,
   item,
   sum(amount) as amt
FROM base_table
GROUP BY 1,2
```

Materialized_View

date	item	amt
20180801	Toaster	100
20180802	Kettle	30
20180804	Kettle	10

2

```
SELECT date,
   item,
   FROM base_table
Where user = "Polly"
```

3

```
SELECT date,
   item, sum(amount) as
amount
   FROM base_table
Where date = "20180101"
```

1 MV is created and results are periodically refreshed

2 By default, queries continue to go to base table unaffected

3 Applicable queries are rerouted under the hood to MV for faster execution

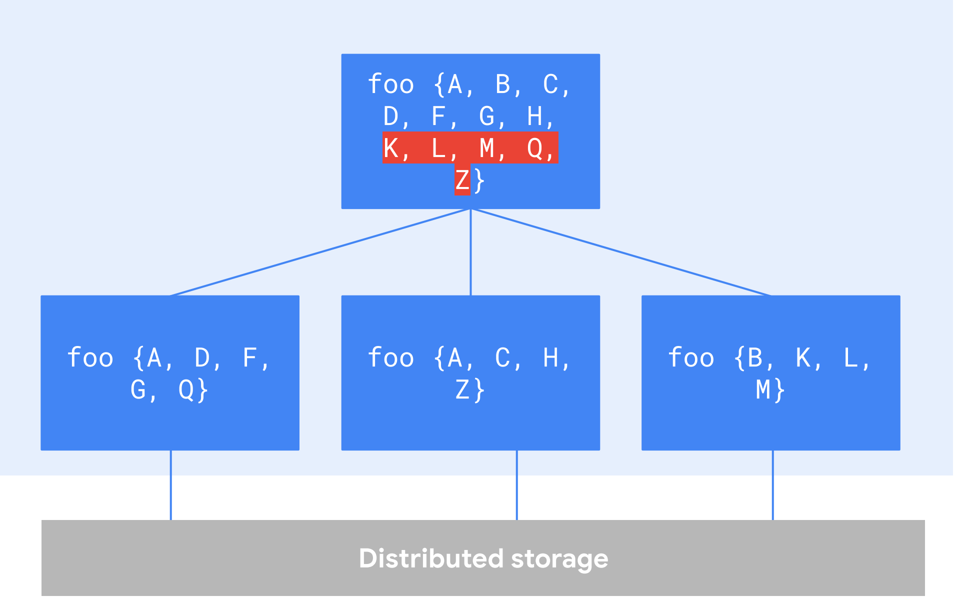
Google Cloud

Best Practice | Query Rewrites

Order WHERE predicates by most selective first.

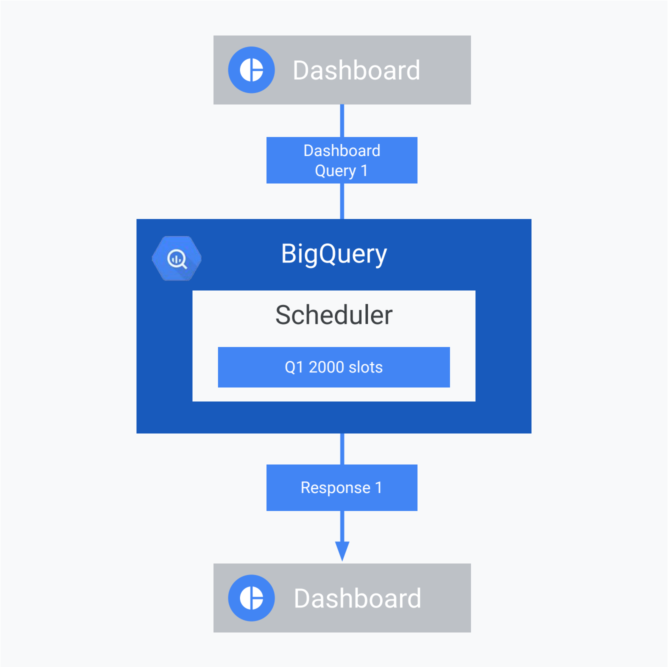
Use ARRAY_AGG with LIMIT 1 instead of ROW_NUMBER for finding the latest record.

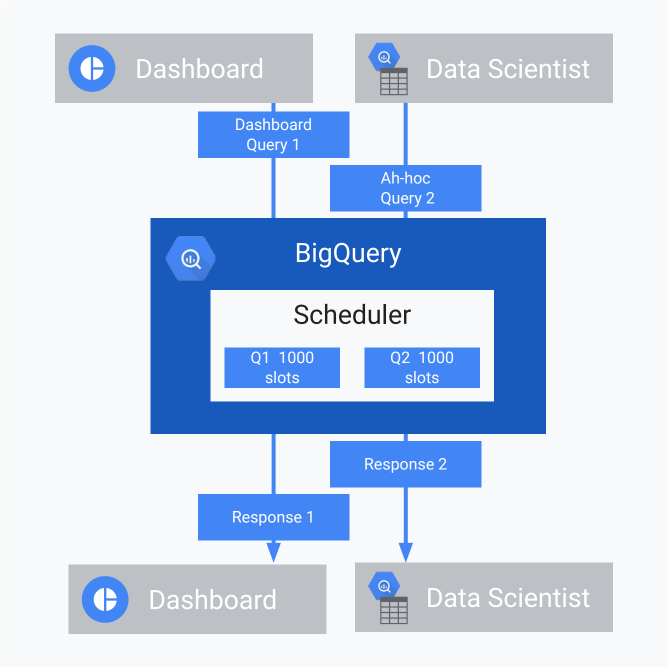
Use analytical functions such as LAG/LEAD instead of self-joins



Workload Management

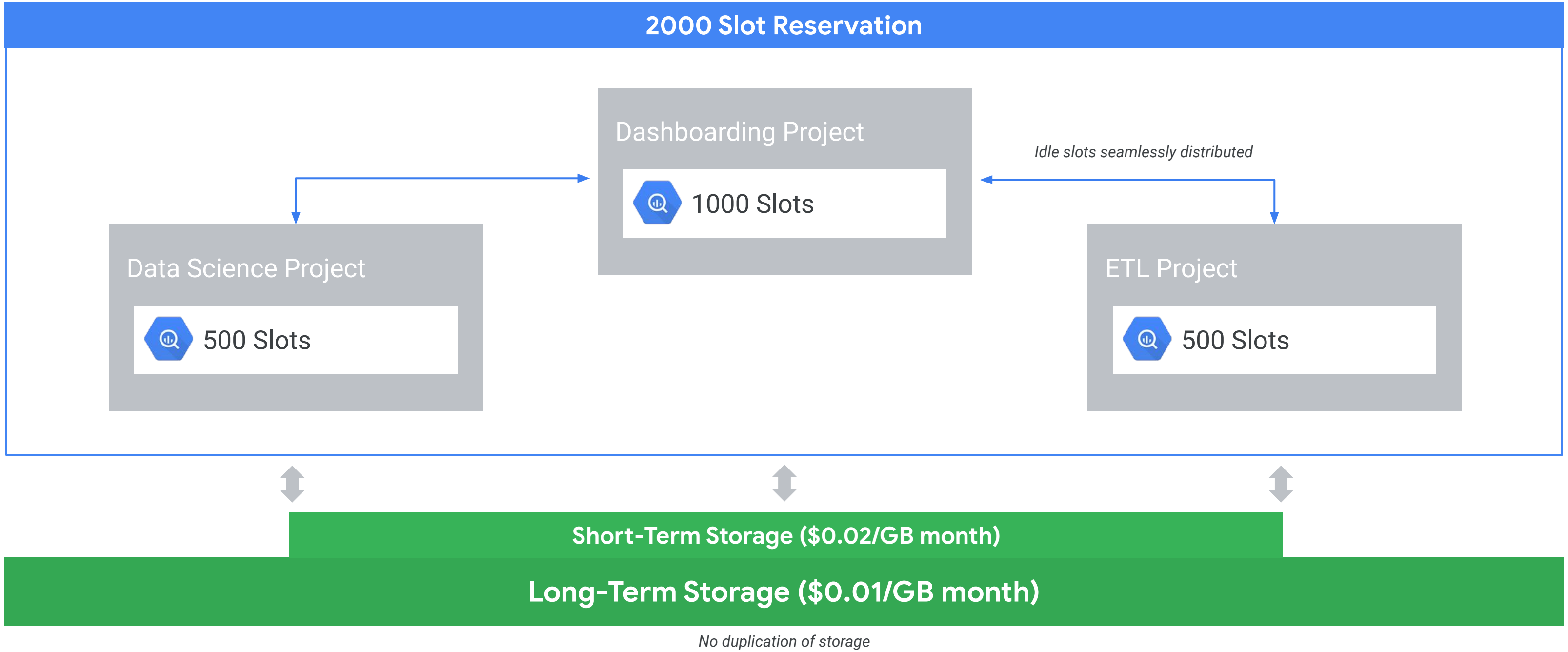
Workload Management | Fair Scheduler





Google Cloud

Best Practice | BigQuery Reservations



2000 Slot Reservation

Dashboarding Project

1000 Slots

Idle slots seamlessly distributed

Data Science Project

500 Slots

ETL Project

500 Slots

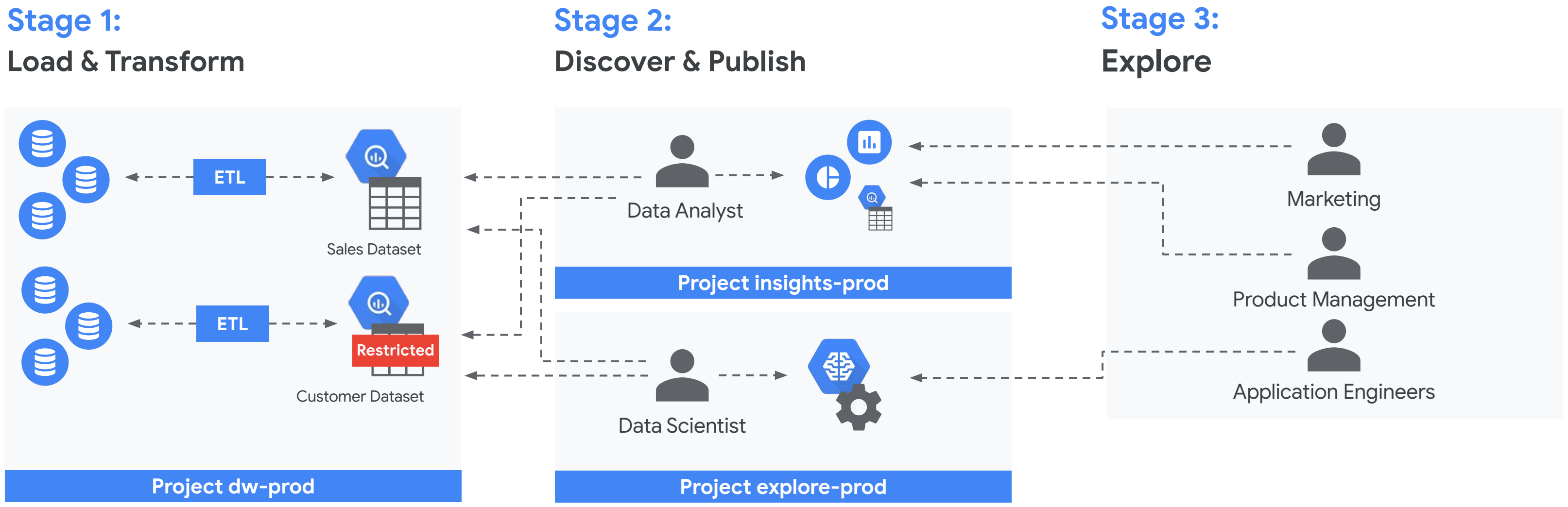
Short-Term Storage ($0.02/GB month)

Long-Term Storage ($0.01/GB month)

No duplication of storage

Google Cloud

Best Practice | Segment Users by Roles



Data Engineers manage a project per environment which stores data.

Data Analysts use Stage 1 data to publish reports, visualizations, and derivative datasets for broader consumption

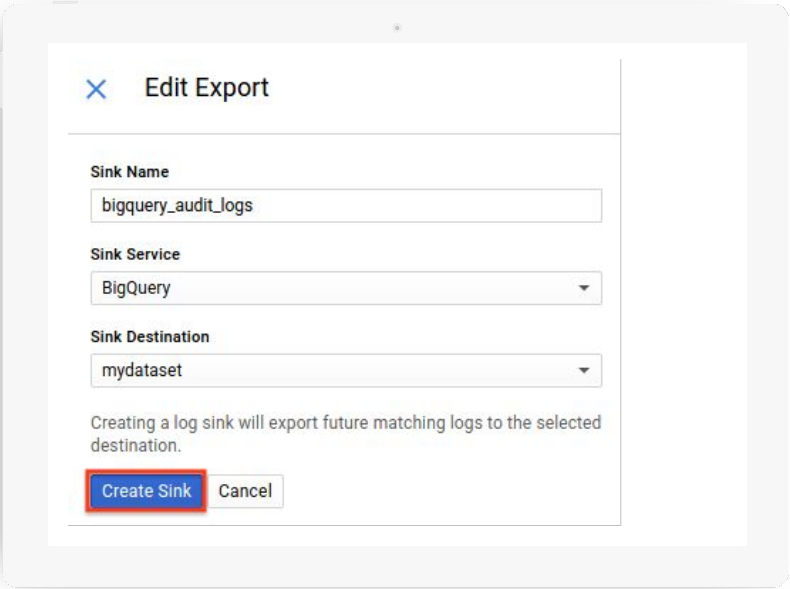
Data Scientists use Stage 1 data to develop and train new Machine Learning models

Functional users explore Stage 2 reports and data, and may publish additional derivative reports

Google Cloud

Auditing

Best Practice | Export Data Access Logs to BigQuery



Edit Export

Sink Name: bigquery_audit_logs

Sink Service: BigQuery

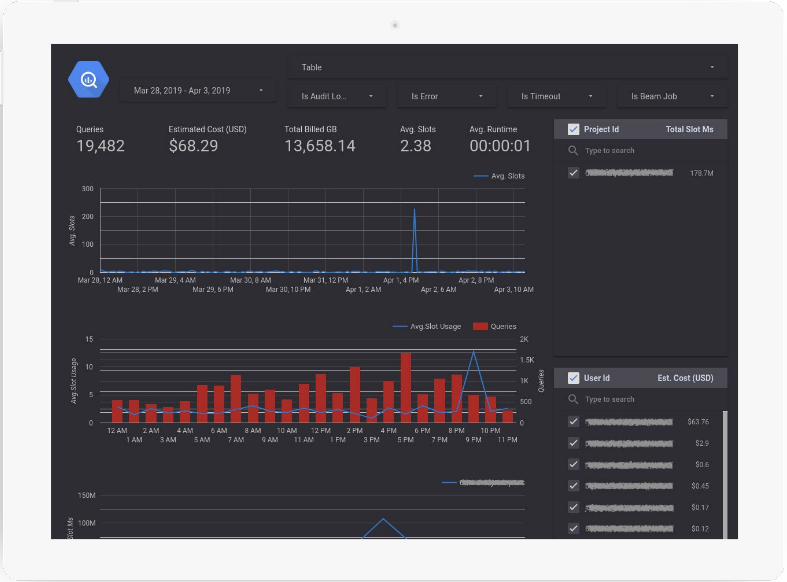
Sink Destination: mydataset

Creating a log sink will export future matching logs to the selected destination.

Data access is exported in real-time to BigQuery and includes metadata such as query executed, bytes scanned, and calling user.

Create **aggregated exports** to BigQuery at the organization or folder level to view data access across projects

Best Practice | Visualize Audit Logs



Mar 28, 2019 - Apr 3, 2019

Table

Is Audit Lo... | Is Error | Is Timeout | Is Beam Job

Queries	Estimated Cost (USD)	Total Billed GB	Avg. Slots	Avg. Runtime
19,482	$68.29	13,658.14	2.38	00:00:01

☑ Project Id	Total Slot Ms
✓ [redacted]	178.7M

☑ User Id	Est. Cost (USD)
✓ [redacted]	$63.76
✓ [redacted]	$2.9
✓ [redacted]	$0.6
✓ [redacted]	$0.45
✓ [redacted]	$0.17
✓ [redacted]	$0.12

bit.ly/bq-audit-codelab

Analyze spending trends & query volume over time

Breakdown costs by project & user

Be proactive about tracking expensive queries and optimizing them

Google Cloud