Return to Table of Contents

Choose a Lesson

- BigQuery Overview
- Interacting with BigQuery
- Load and Export Data
- Optimize for Performance and Costs
- Streaming Insert Example
- BigQuery Logging and Monitoring
- BigQuery Best Practices

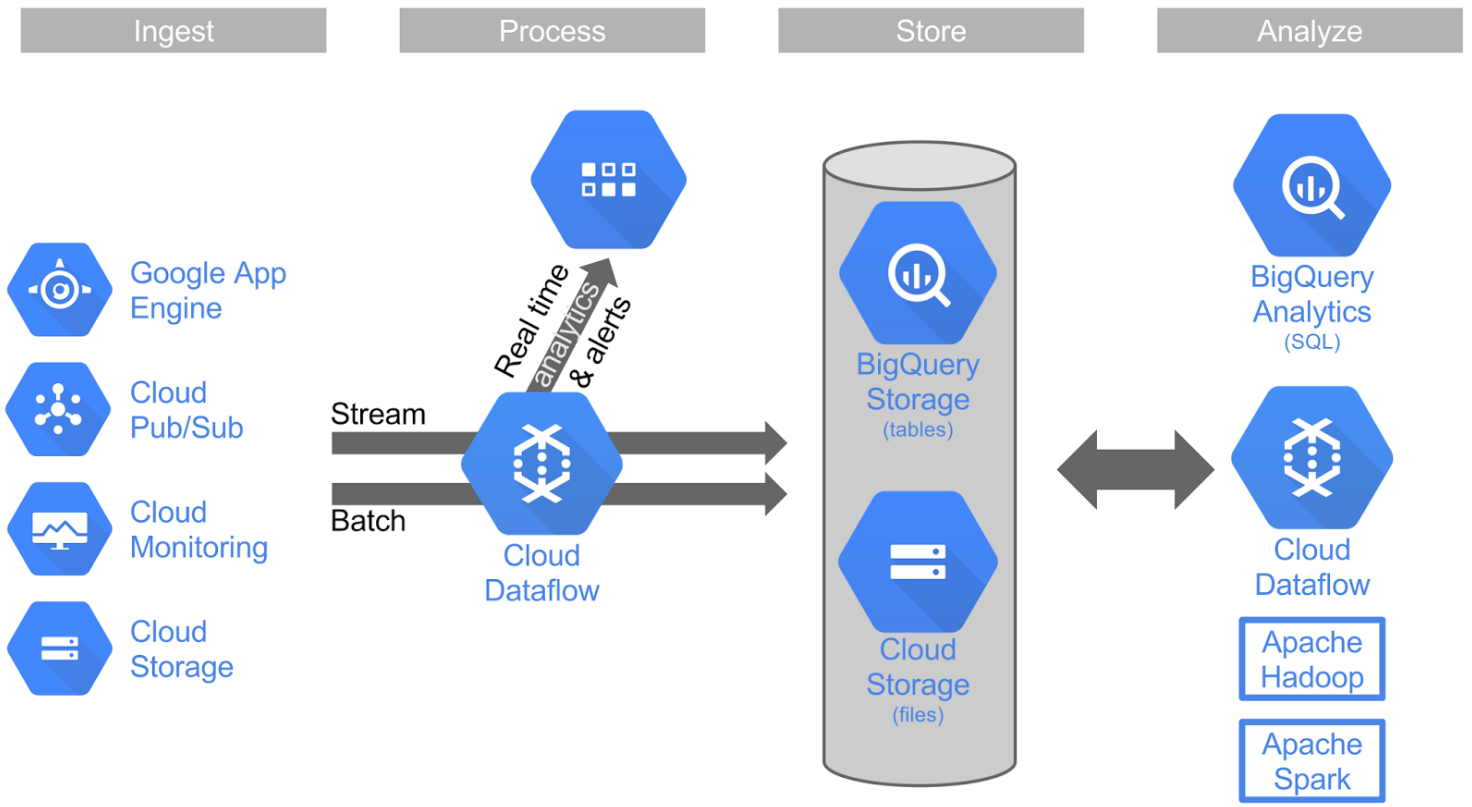
BigQuery Overview

Next

What is BigQuery?

- Fully Managed Data warehousing
  - Near-real time analysis of petabyte scale databases
- Serverless (no-ops)
- Auto-scaling to petabyte range
- Both storage and analysis
- Accepts batch and streaming loads
- Locations = multi-regional (US, EU), Regional (asia-northeast1)
- Replicated, durable
- Interact primarily with standard SQL (also Legacy SQL)
  - SQL Primer course

exam: basic SQL



Return to Table of Contents

Choose a Lesson

- BigQuery Overview
- Interacting with BigQuery
- Load and Export Data
- Optimize for Performance and Costs
- Streaming Insert Example
- BigQuery Logging and Monitoring
- BigQuery Best Practices

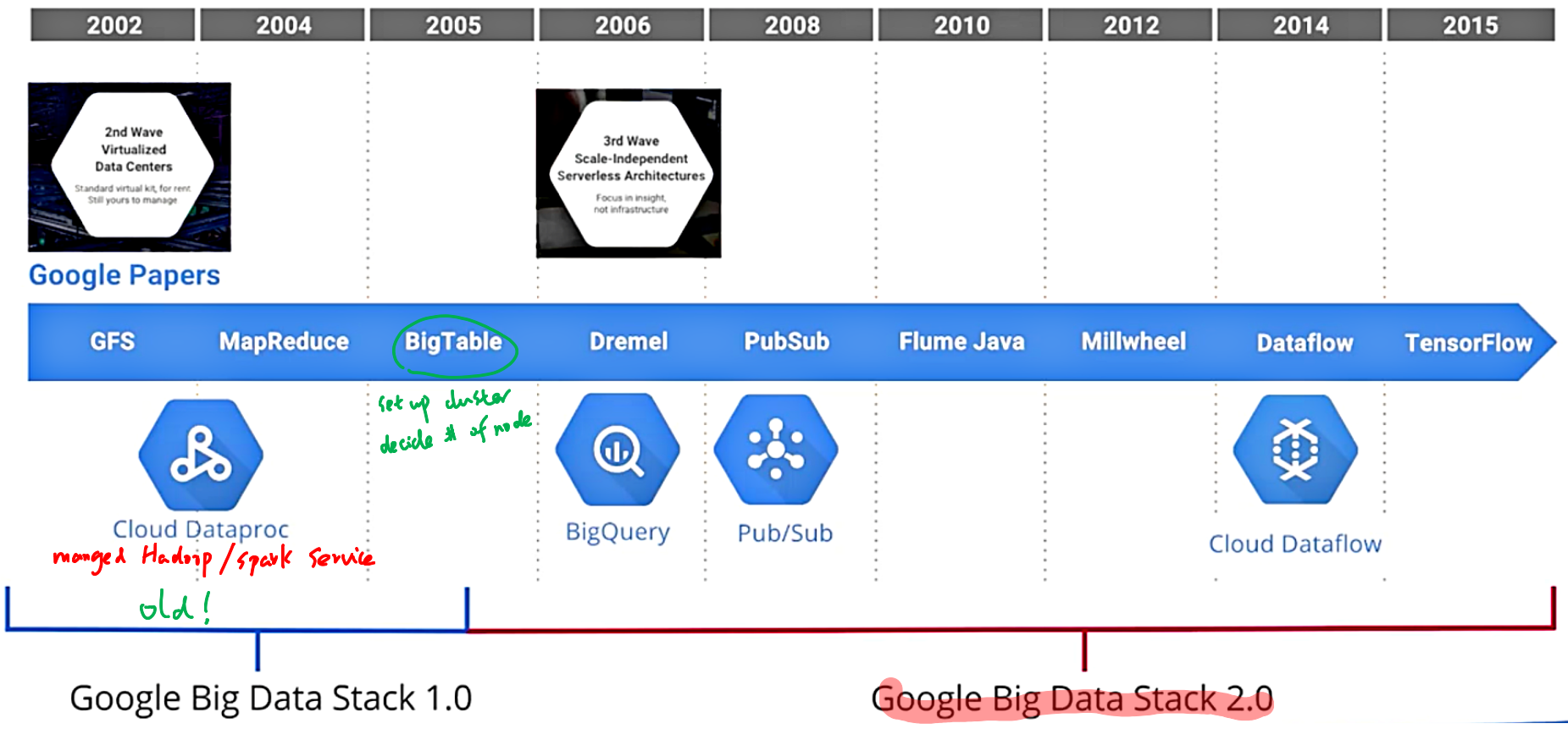
BigQuery Overview

Previous

Next

How BigQuery works

- Part of the "3rd wave" of cloud computing
  - Google Big Data Stack 2.0
- Focus on serverless compute, real time insights, machine learning...
  - ...instead of data placement, cluster configuration
  - No managing of infrastructure, nodes, clusters, etc



Return to Table of Contents

Choose a Lesson

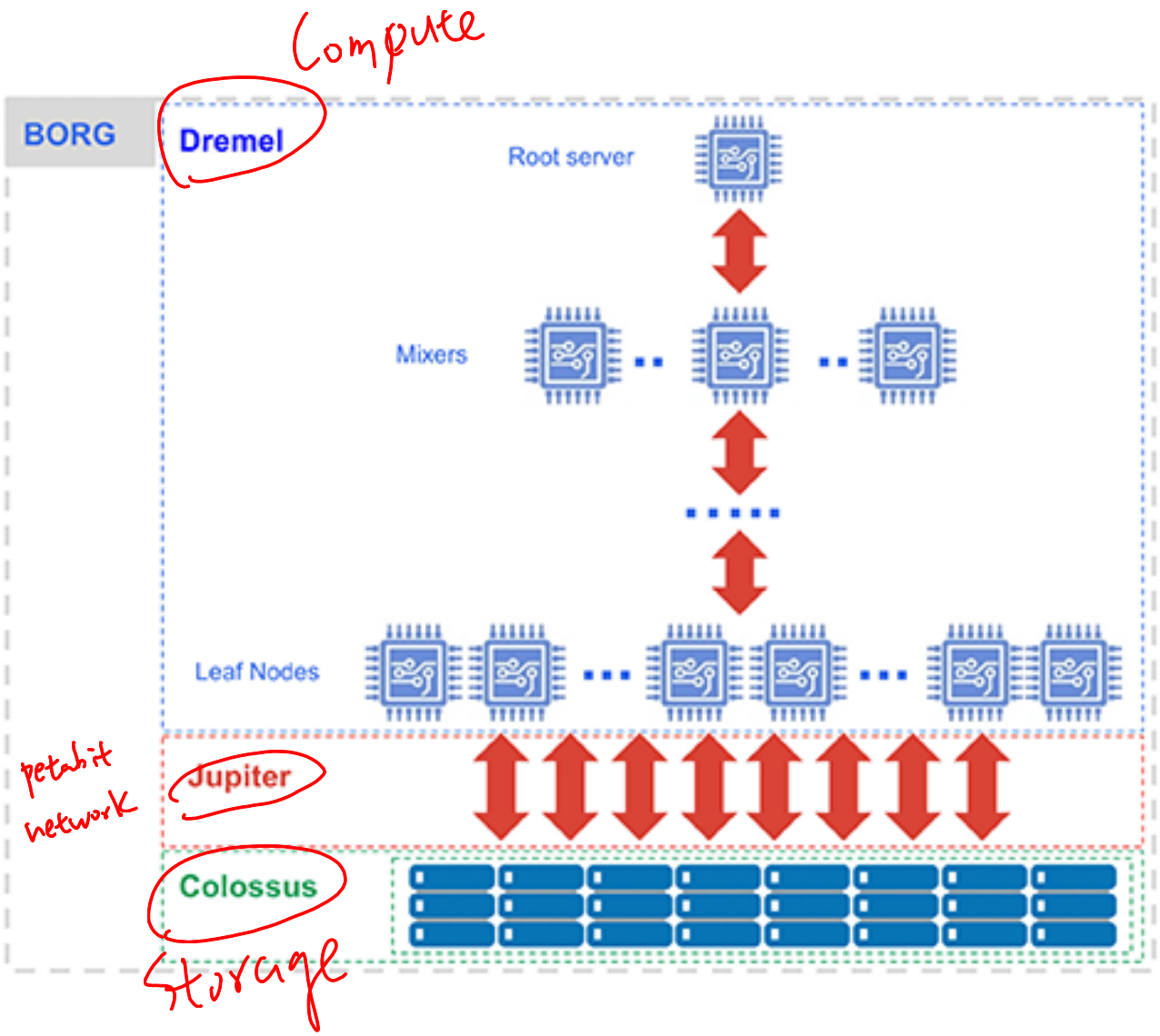
- BigQuery Overview
- Interacting with BigQuery
- Load and Export Data
- Optimize for Performance and Costs
- Streaming Insert Example
- BigQuery Logging and Monitoring
- BigQuery Best Practices

BigQuery Overview

Previous (Not in exam) Next

How BigQuery works (cont)

- Jobs (queries) can scale up to thousands of CPU's across many nodes, but the process is completely invisible to end user
- **Storage** and **compute** are separated, connected by **petabit network**



Return to Table of Contents

Choose a Lesson

- BigQuery Overview
- Interacting with BigQuery
- Load and Export Data
- Optimize for Performance and Costs
- Streaming Insert Example
- BigQuery Logging and Monitoring
- BigQuery Best Practices

BigQuery Overview

Previous

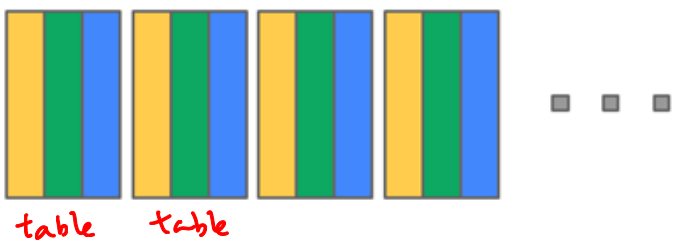
Next

How BigQuery works (cont)

- Columnar data store (different from SQL)
  - Separates records into column values, stores each value on different storage volume
  - Traditional RDBMS stores whole record on one volume
  - Extremely fast read performance, poor write (update) performance - BigQuery does not update existing records
  - Not transactional

just append records

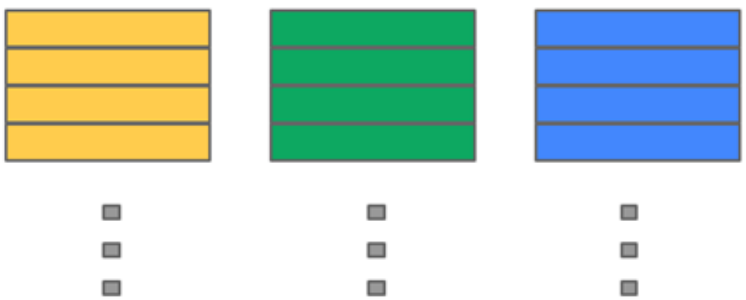
SQL



Record Oriented Storage

entire table with rows

Columnar data store



Column Oriented Storage

Return to Table of Contents

Choose a Lesson

- BigQuery Overview
- Interacting with BigQuery
- Load and Export Data
- Optimize for Performance and Costs
- Streaming Insert Example
- BigQuery Logging and Monitoring
- BigQuery Best Practices

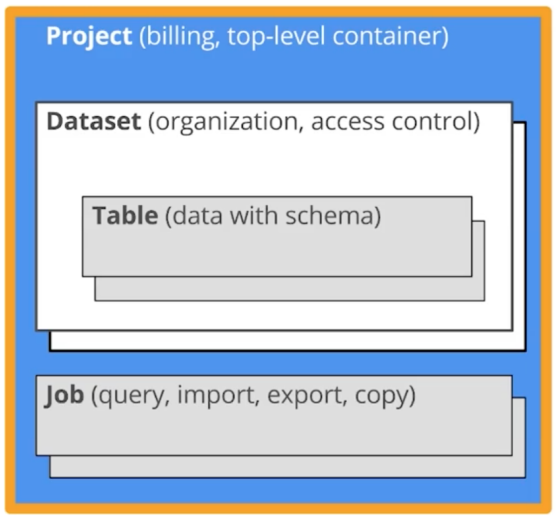
BigQuery Overview

Previous (return topic)

Next

BigQuery structure

- Dataset - contains tables/views
- Table = collection of columns
- Job = long running action/query



Identity and Access Management (IAM)

- Control by project, dataset, view
- Cannot control at table level (only project level, or dataset level)
  - But can control by views via datasets as alternative (virtual table defined by SQL query)
- Predefined roles - BigQuery...
  - Admin - full access
  - Data Owner - full dataset access
  - Data Editor - edit dataset tables
  - Data Viewer - view datasets and tables
  - Job User - run jobs
  - User - run queries and create datasets (but not tables)
- Roles comparison matrix
- Sharing datasets
  - Make public with All Authenticated Users

Return to Table of Contents

Choose a Lesson

BigQuery Overview

Interacting with BigQuery

Load and Export Data

Optimize for Performance and Costs

Streaming Insert Example

BigQuery Logging and Monitoring

BigQuery Best Practices

BigQuery Overview

Previous

Pricing

- Storage, Queries, Streaming insert
- Storage = $0.02/GB/mo (first 10GB/mo free)
  - Long term storage (not edited for 90 days) = $0.01/GB/mo
- Queries = $5/TB (first TB/mo free)
- Streaming = $0.01/200 MB
- Pay as you go, with high end flat-rate query pricing
- Flat rate - starts at $40K per month with 2000 slots

users

Return to Table of Contents

Choose a Lesson

- BigQuery Overview
- Interacting with BigQuery
- Load and Export Data
- Optimize for Performance and Costs
- Streaming Insert Example
- BigQuery Logging and Monitoring
- BigQuery Best Practices

Interacting with BigQuery

Next

exam tip

Interaction methods

- Web UI
- Command line (bq commands)
  - bq query --arguments 'QUERY'

create dataset:
$ bq mk --dataset <project-ID>:new_dat

- Programmatic (REST API, client libraries)
- Interact via queries

Querying tables

- FROM `project.dataset.table` (Standard SQL)
- FROM [project:dataset.table] (Legacy SQL)

Searching multiple tables with wildcards

Query across multiple, similarly named tables

- FROM `project.dataset.table_prefix*`

Filter further in WHERE clause

- AND _TABLE_SUFFIX BETWEEN 'table003' and 'table050'

Advanced SQL queries are allowed

- JOINS, sub queries, CONCAT

Return to Table of Contents

Choose a Lesson

- BigQuery Overview
- Interacting with BigQuery
- Load and Export Data
- Optimize for Performance and Costs
- Streaming Insert Example
- BigQuery Logging and Monitoring
- BigQuery Best Practices

Interacting with BigQuery

Previous

Views

(exam) query inside query

only contains data from previous query.

- Virtual table defined by query
- 'Querying a query'
- Contains data only from query that contains view
- Useful for limiting table data to others

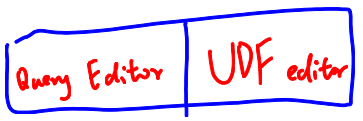
Cached queries

☑ use as cached results

(exam)

- Queries cost money
- Previous queries are cached to avoid charges if ran again
- command line to disable cached results
  - bq query --nouse_cache '(QUERY)'
- Caching is per user only

User Defined Functions (UDF)



- Combine SQL code with JavaScript/SQL functions
- Combine SQL queries with programming logic
- Allow much more complex operations (loops, complex conditionals)
- WebUI only usable with Legacy SQL
- Command Line only with standard SQL

Return to Table of Contents

Choose a Lesson

BigQuery Overview

Interacting with BigQuery

Load and Export Data

Optimize for Performance and Costs

Streaming Insert Example

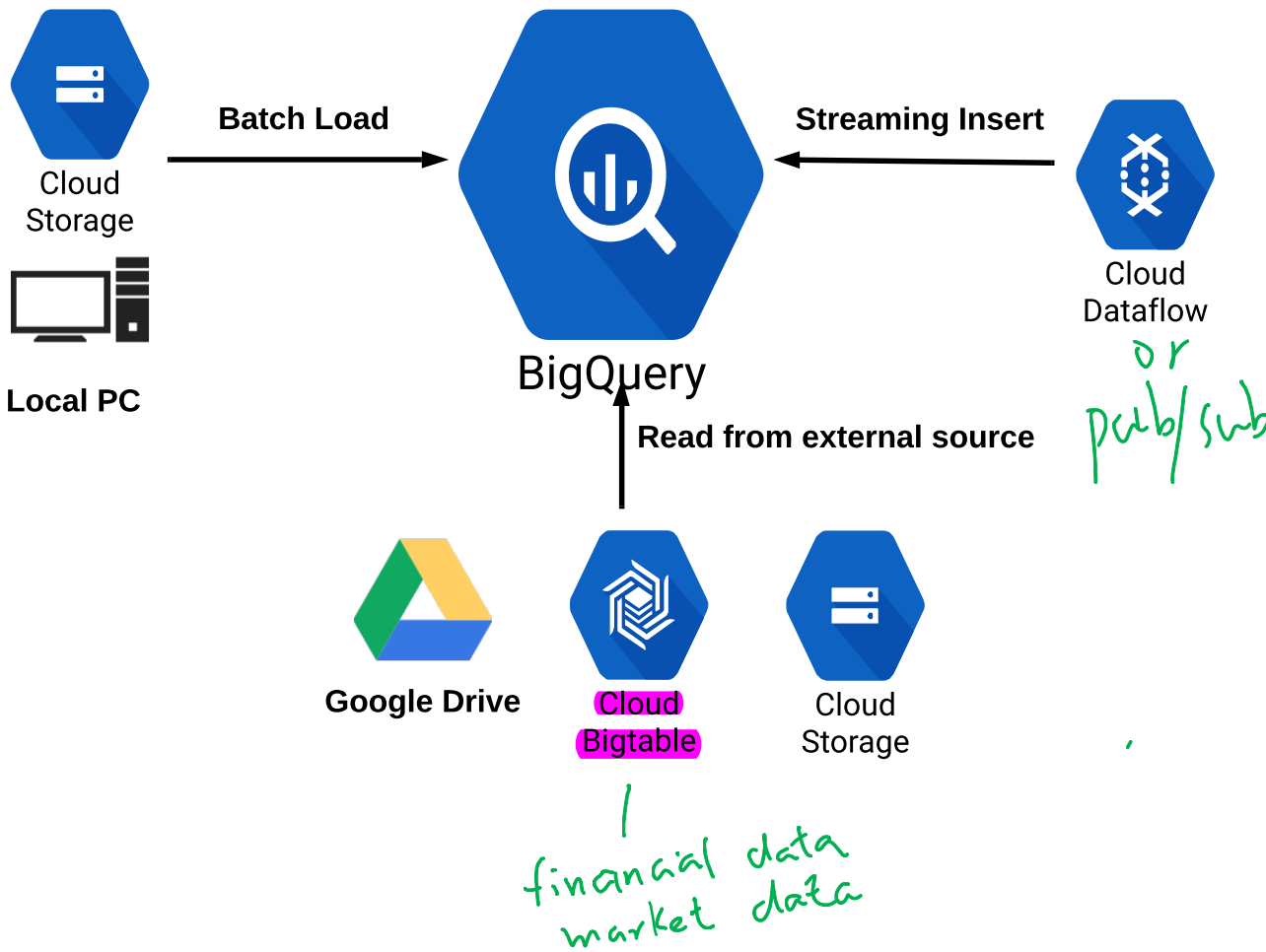
BigQuery Logging and Monitoring

BigQuery Best Practices

Load and Export Data

Loading and reading sources

Next



Data formats:

Load

- CSV
- JSON (Newline delimited)
- Avro - best for compressed files
- Parquet
- Datastore backups

Read

- CSV
- JSON (Newline delimited)
- Avro
- Parquet

Why use external sources?

- Load and clean data in one pass from external, then write to BigQuery
- Small amount of frequently changing data to join to other tables

Loading data with command line

- bq load --source_format=[format] [dataset].[table] [source_path] [schema]
- Can load multiple files with command line (not WebUI)

Return to Table of Contents

Choose a Lesson

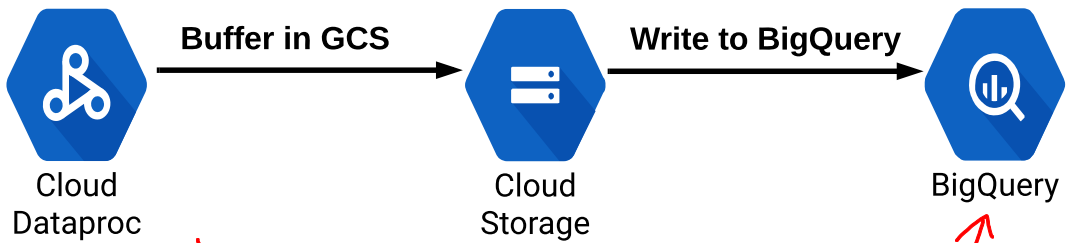
- BigQuery Overview
- Interacting with BigQuery
- Load and Export Data
- Optimize for Performance and Costs
- Streaming Insert Example
- BigQuery Logging and Monitoring
- BigQuery Best Practices

Load and Export Data

Previous

Connecting to/from other Google Cloud services

- Dataproc - Use BigQuery connector (installed by default), job uses Cloud Storage for staging



Exporting tables

vs Copying tables

- Can only export to Cloud Storage
- Can copy table to another BigQuery dataset
- Export formats: CSV, JSON, Avro
- Can export multiple tables with command line, 1 table at a time if use web UI
- Can only export up to 1GB per file, but can split into multiple files with wildcards (*)
- Command line
  - bq extract 'projectid:dataset.table' gs://bucket_name/folder/object_name
  - Can drop 'project' if exporting from same project
  - Default is CSV, specify other format with --destination_format
  - --destination_format=NEWLINE_DELIMITED_JSON

BigQuery Transfer Service

- Import data to BigQuery from other Google advertising SaaS applications
- Google AdWords
- DoubleClick
- YouTube reports

> data ⇒ BigQuery

Return to Table of Contents

Choose a Lesson

- BigQuery Overview
- Interacting with BigQuery
- Load and Export Data
- Optimize for Performance and Costs
- Streaming Insert Example
- BigQuery Logging and Monitoring
- BigQuery Best Practices

Optimize for Performance and Costs

Next

Performance and costs are complementary

- Less work = faster query = less costs
- What is 'work'?
  - I/O - how many bytes read?
  - Shuffle - how much passed to next stage
  - How many bytes written?
  - CPU work in functions

General best practices

1. Avoid using SELECT *
2. Denormalize data when possible (counter to general relational database)
   - Grouping data into single table
   - Often with nested/repeated data (exam question)
   - Good for read performance, not for write (transactional) performance
3. Filter early and big with WHERE clause
4. Do biggest joins first, and filter pre-JOIN } filter out data
5. LIMIT does not affect cost
6. Partition data by date
   - Partition by ingest time – possible to query data by date
   - Parition by specified data columns

Return to Table of Contents

Choose a Lesson

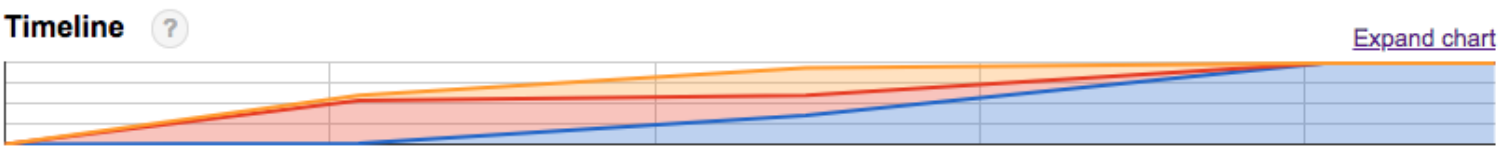
- BigQuery Overview
- Interacting with BigQuery
- Load and Export Data
- Optimize for Performance and Costs
- Streaming Insert Example
- BigQuery Logging and Monitoring
- BigQuery Best Practices

Optimize for Performance and Costs

Previous

Monitoring query performance

- Understand color codes ← for exam question
- Understand 'skew' in difference between average and max time



Execution Plan

many nodes are work in background

	Stage timing				Rows		
	Wait	Read	Compute	Write	Parallel Inputs	Input	Output
S00: Input					105	122 M	5.34 K (78.2 KB)
S02: Coalesce					100	5.34 K	5.34 K (78.2 KB)
S03: Join+					74	22.1 M	37 (925 B)
S04: Aggregate+					17	37	19 (646 B)
S05: Output					1	19	19 (532 B)

faster performance will lower those number

⇒ less cost

Return to Table of Contents

Choose a Lesson

- BigQuery Overview
- Interacting with BigQuery
- Load and Export Data
- Optimize for Performance and Costs
- Streaming Insert Example
- BigQuery Logging and Monitoring
- BigQuery Best Practices

Streaming Insert Example

Quick setup

cd

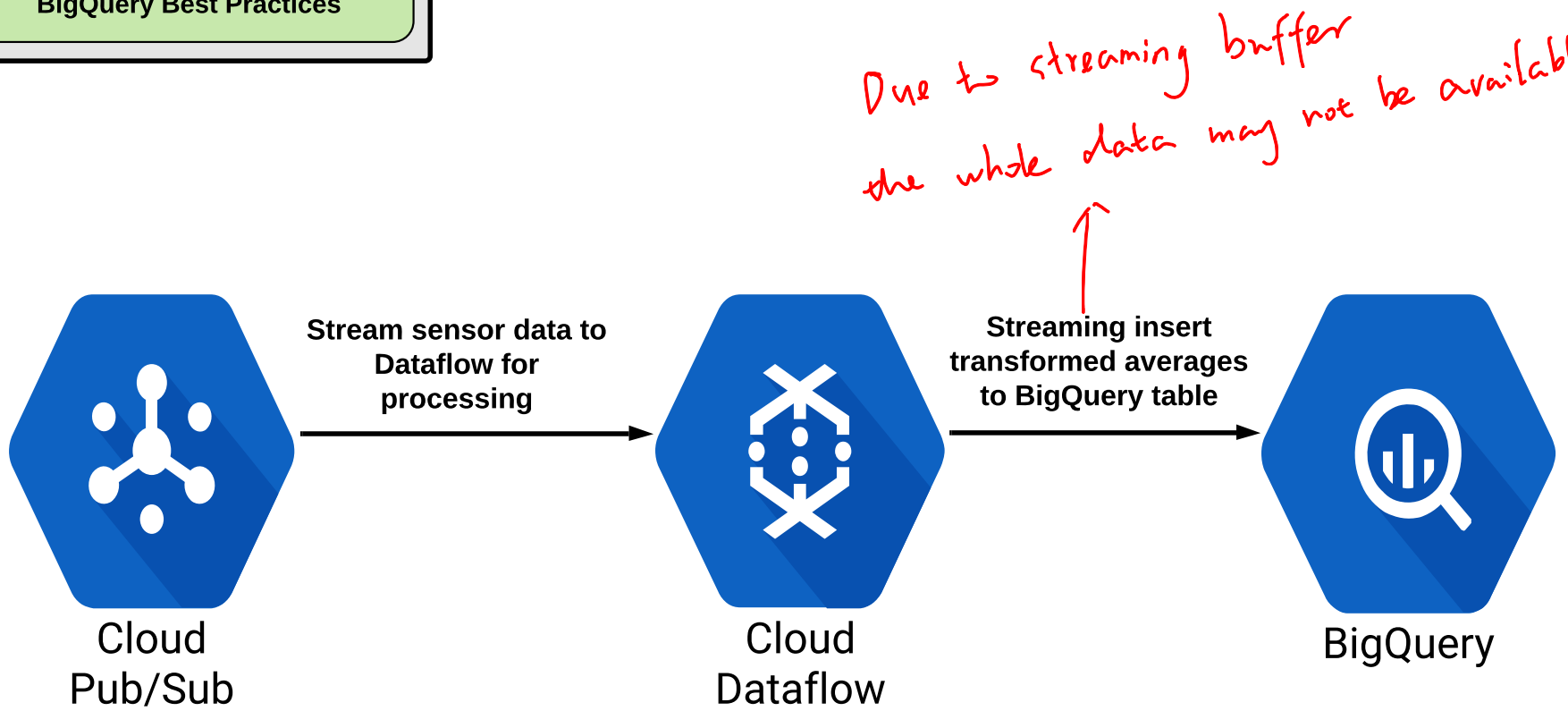
gsutil cp -r gs://gcp-course-exercise-scripts/data-engineer/* .

bash streaming-insert.sh

Clean up

bash streaming-cleanup.sh

Manually stop Dataflow job



Return to Table of Contents

Choose a Lesson

- BigQuery Overview
- Interacting with BigQuery
- Load and Export Data
- Optimize for Performance and Costs
- Streaming Insert Example
- BigQuery Logging and Monitoring
- BigQuery Best Practices

BigQuery Logging and Monitoring

Stackdriver Monitoring and Logging Differences

- Monitoring = performance/resources (usage)
- Logging = who is doing what
  - History of actions

← Two big stackdriver services!

Monitoring BigQuery Performance/Resources

- Monitoring = metrics, performance, resource capacity/usage (slots)
  - Query count, query times, slot utilization
  - Number of tables, stored and uploaded bytes over time
  - Alerts on metrics e.g., long query times
    - Example: Alerts when queries take more than one minute
- No data on **who** is doing what, or query details

find the heavy user

set an alert

(1) (2)

Stackdriver Logging: "A Paper Trail"

- Logging = who is doing what
- Record of jobs and queries associated with accounts

Return to Table of Contents

Choose a Lesson

BigQuery Overview

Interacting with BigQuery

Load and Export Data

Optimize for Performance and Costs

Streaming Insert Example

BigQuery Logging and Monitoring

BigQuery Best Practices

BigQuery Best Practices

Next

Data Format for Import

- Best performance = Avro format
- Scenario: Import multi-TB databases with millions of rows

Faster
Avro - Compressed
Avro - Uncompressed
Parquet
CSV
JSON
CSV - Compressed
JSON - Compressed
Slower

Return to Table of Contents

Choose a Lesson

BigQuery Overview

Interacting with BigQuery

Load and Export Data

Optimize for Performance and Costs

Streaming Insert Example

BigQuery Logging and Monitoring

BigQuery Best Practices

BigQuery Best Practices

Previous

Next

Partitioned Tables

What is a partitioned table?

- Special single table
  - Divided into segments known as “partitions” by time

Why is this important?

- Query only certain rows (partitions) instead of entire table
  - Limits amount of read data
  - Improves performance
  - Reduces costs
- Partition types
  - Ingests time — when the data/row is created
  - Includes **TIMESTAMP** or **DATE** column
- **Scenario**: A large amount of data gets generated every day, and we need to query for only certain time periods within the same table.

Why not use multiple tables (one for each day) plus wildcards?

- Limited to 1000 tables per dataset
- Substantial performance drop vs. a single table

Return to Table of Contents

Choose a Lesson

BigQuery Overview

Interacting with BigQuery

Load and Export Data

Optimize for Performance and Costs

Streaming Insert Example

BigQuery Logging and Monitoring

BigQuery Best Practices

BigQuery Best Practices

Previous

Next

Clustered Tables

- Taking partitioned tables "to the next level"
- Similar to partitioning, divides table reads by a specified column field
  - Instead of dividing by date/time, divides by field
- **Scenario:** Logistics company needs to query by tracking ID
  - Cluster by tracking ID column = only reading table rows with specified tracking ID's
- Restriction: only (currently) available for partitioned tables

* the table must be partitioned before you cluster

Slots

- Computational capacity required to run a SQL query
  - Bigger/more complex queries need more slots
- Default, on-demand pricing allocates 2000 slots
  - Only an issue for extremely complex queries, or high number of simultaneous users
  - If more than 2000 slots required, switch to **flat-rate pricing**

Return to Table of Contents

Choose a Lesson

BigQuery Overview

Interacting with BigQuery

Load and Export Data

Optimize for Performance and Costs

Streaming Insert Example

BigQuery Logging and Monitoring

BigQuery Best Practices

BigQuery Best Practices

Previous

Backup and Recovery

Build-in Snapshot feature

- Highly available = multi-regional dataset vs. regional
- Backup/recovery = BigQuery automatically takes continuous snapshots of tables
  - 7 day history, but 2 days if purposely deleted
- Restore to previous point in time using `@(time)`, in milliseconds
- Example: Get snapshot from one hour ago

```
#legacySQL
SELECT * FROM [PROJECT_ID:DATASET.TABLE@-3600000]
```

1 hour

- Alternatively, export table data to GCS, though not as cost effective