Data Warehouse

- OLAP vs OLTP
- What is data warehouse
- BigQuery
  - Cost
  - Partitions and Clustering
  - Best practices
  - Internals
  - ML in BQ

Data Warehouse

OLTP: Online Transaction Processing

- OLTP is used for backend databases, where we want to group SQL queries together.

OLAP: Online Analytical Processing

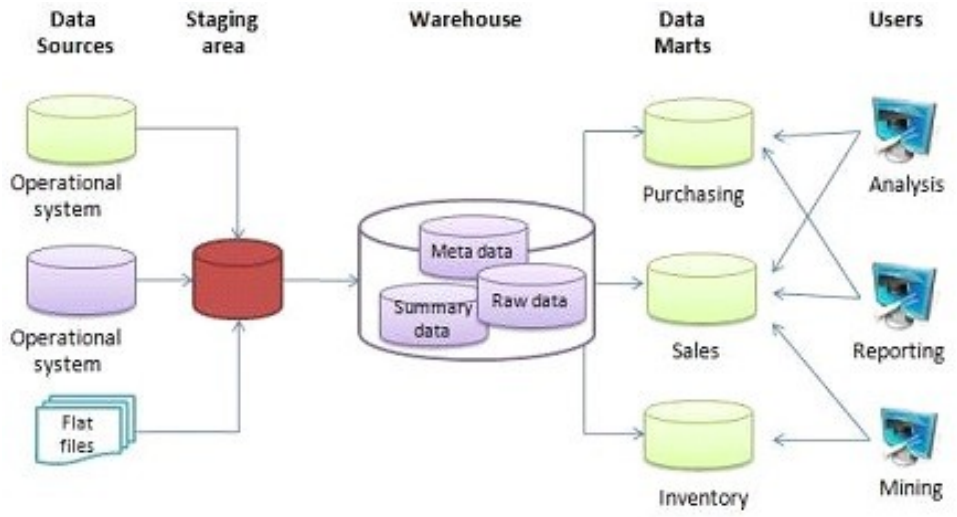
- OLAP is used for putting a lot of data in and discovering insights in the data(used by data analysts or data scientists).

	OLTP	**OLAP**
Purpose	Control and run essential business operations in real time	Plan, solve problems, support decisions, discover hidden insights
Data updates	Short, fast updates initiated by user	Data periodically refreshed with scheduled, long-running batch jobs
Database design	Normalized databases for efficiency	Denormalized databases for analysis
Space requirements	Generally small if historical data is archived	Generally large due to aggregating large datasets

	OLTP	OLAP
Backup and recovery	Regular backups required to ensure business continuity and meet legal and governance requirements	Lost data can be reloaded from OLTP database as needed in lieu of regular backups
Productivity	Increases productivity of end users	Increases productivity of business managers, data analysts, and executives
Data view	Lists day-to-day business transactions	Multi-dimensional view of enterprise data
User examples	Customer-facing personnel, clerks, online shoppers	Knowledge workers such as data analysts, business analysts, and executives

What is a data warehouse

- OLAP solution
- Used for reporting and data analysis



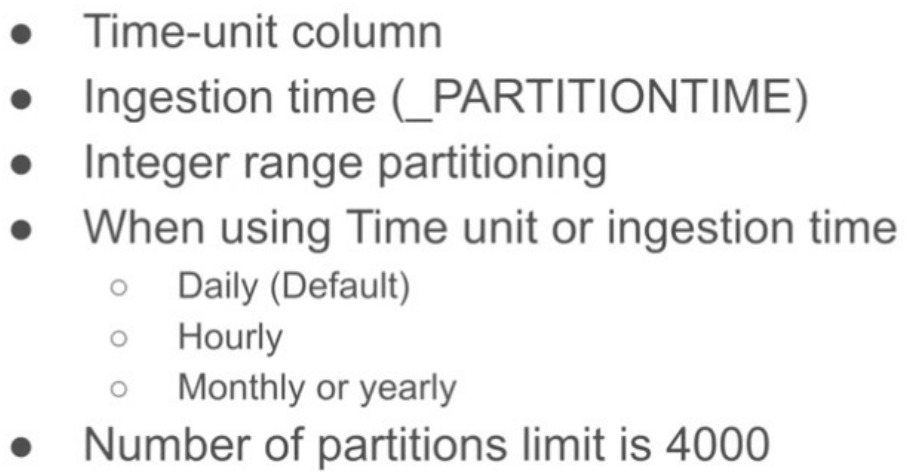
Data Partitioning and Clustering

- Data can be partitioned (e.g. by date) to decrease the size of the processed data for each query
- Clustering the data (e.g. by VendorID) reduces even more the amount of processed data needed for each query.
- This is important because BigQuery charges by the amount of data processed:

  - On demand pricing
    - 1 TB of data processed is $5
  - Flat rate pricing
    - Based on number of pre requested slots
    - 100 slots → $2,000/month = 400 TB data processed on demand pricing

Partitioning Data

- How to partition data?
  - Partitions should be of equivalent size
  - Good suggested partitioning criteria:



Clustering Data

- How to cluster data?
  - Max of four(4) columns

- Columns you specify are used to colocate related data
- Order of the column is important
- The order of the specified columns determines the sort order of the data.
- Clustering improves
  - Filter queries
  - Aggregate queries
- Table with data size < 1 GB, don’t show significant improvement with partitioning and clustering
- You can specify up to four clustering columns

Clustering columns must be top-level, non-repeated columns

- DATE
- BOOL
- GEOGRAPHY
- INT64
- NUMERIC
- BIGNUMERIC
- STRING
- TIMESTAMP
- DATETIME

Partitioning vs Clustering

Clustering	**Partitoning**
Cost benefit unknown	Cost known upfront
You need more granularity than partitioning alone allows	You need partition-level management.
Your queries commonly use filters or aggregation against multiple particular columns	Filter or aggregate on single column
The cardinality of the number of values in a column or group of columns is large	

Clustering over paritioning

- Partitioning results in a small amount of data per partition (approximately less than 1 GB)
- Partitioning results in a large number of partitions beyond the limits on partitioned tables
- Partitioning results in your mutation operations modifying the majority of partitions in the table frequently (for example, every few minutes)

Automatic reclustering

As data is added to a clustered table

- the newly inserted data can be written to blocks that contain key ranges that overlap with the key ranges in previously written blocks
- These overlapping keys weaken the sort property of the table

To maintain the performance characteristics of a clustered table

- BigQuery performs automatic re-clustering in the background to restore the sort property of the table
- For partitioned tables, clustering is maintained for data within the scope of each partition.

BigQuery-Best Practice

- Cost reduction
  - Avoid SELECT *
  - Price your queries before running them
  - Use clustered or partitioned tables
  - Use streaming inserts with caution
  - Materialize query results in stages

BigQuery-Best Practice

- Query performance
  - Filter on partitioned columns
  - Denormalizing data
  - Use nested or repeated columns
  - Use external data sources appropriately
  - Don't use it, in case u want a high query performance
  - Reduce data before using a JOIN
  - Do not treat WITH clauses as prepared statements
  - Avoid oversharding tables

BigQuery-Best Practice

- Query performance
  - Avoid JavaScript user-defined functions
  - Use approximate aggregation functions (HyperLogLog++)
  - Order Last, for query operations to maximize performance
  - Optimize your join patterns
  - As a best practice, place the table with the largest number of rows first, followed by the table with the fewest rows, and then place the remaining tables by decreasing size.

Internals of Big Query

- Big Query separates the Storage (Colossus) from the Compute resources, which reduces the cost(e.g. adding more data only incurs in additional costs for Colossus and not Compute).
- However, having Storage and Compute in different Hardware means that the communication between them can have a lag.
- This lag is resolved by BigQuery using the Jupyter network, which can transmit/receive 1TB per second.
- Dremel is the query execution engine, which divides queries into a tree structure so that each node can execute an indivudual subset of the query.

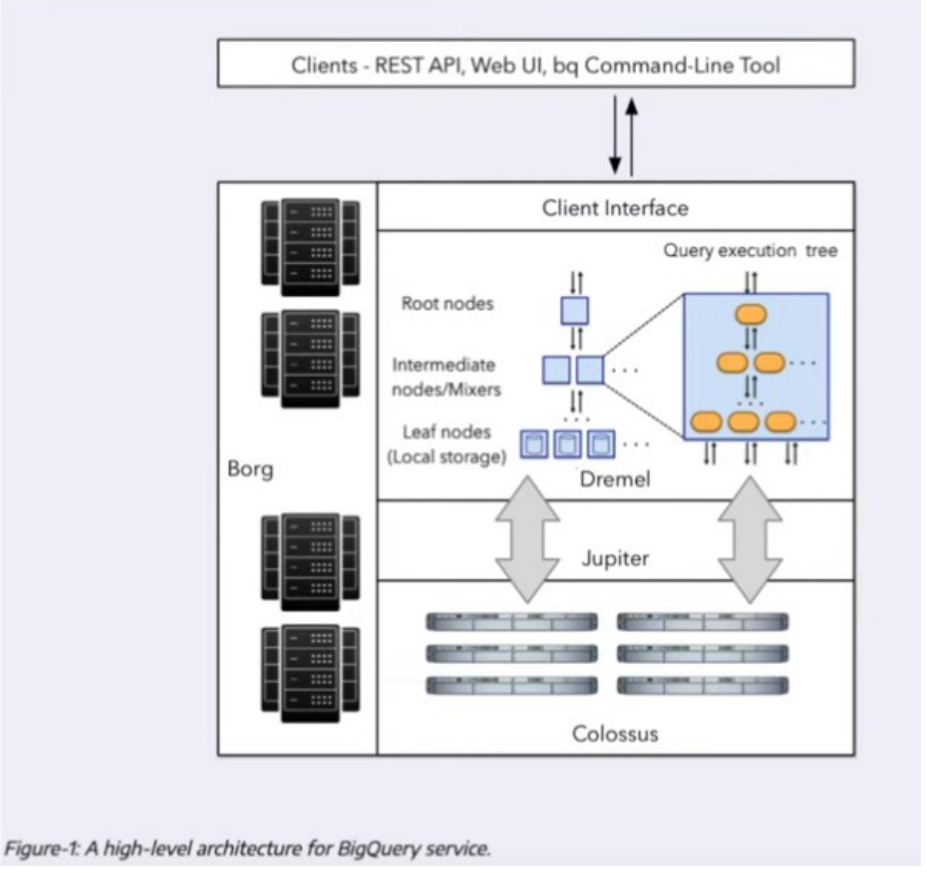
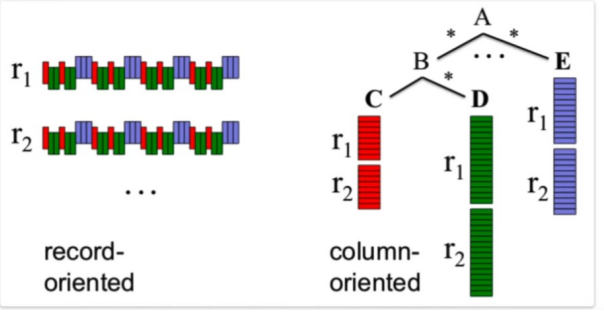


Figure-1: A high-level architecture for BigQuery service.

Column and Record oriented storage

- Record Oriented (e.g. CSV)
- Column Oriented
  - Used by BigQuery
  - Helps with aggregation of columns



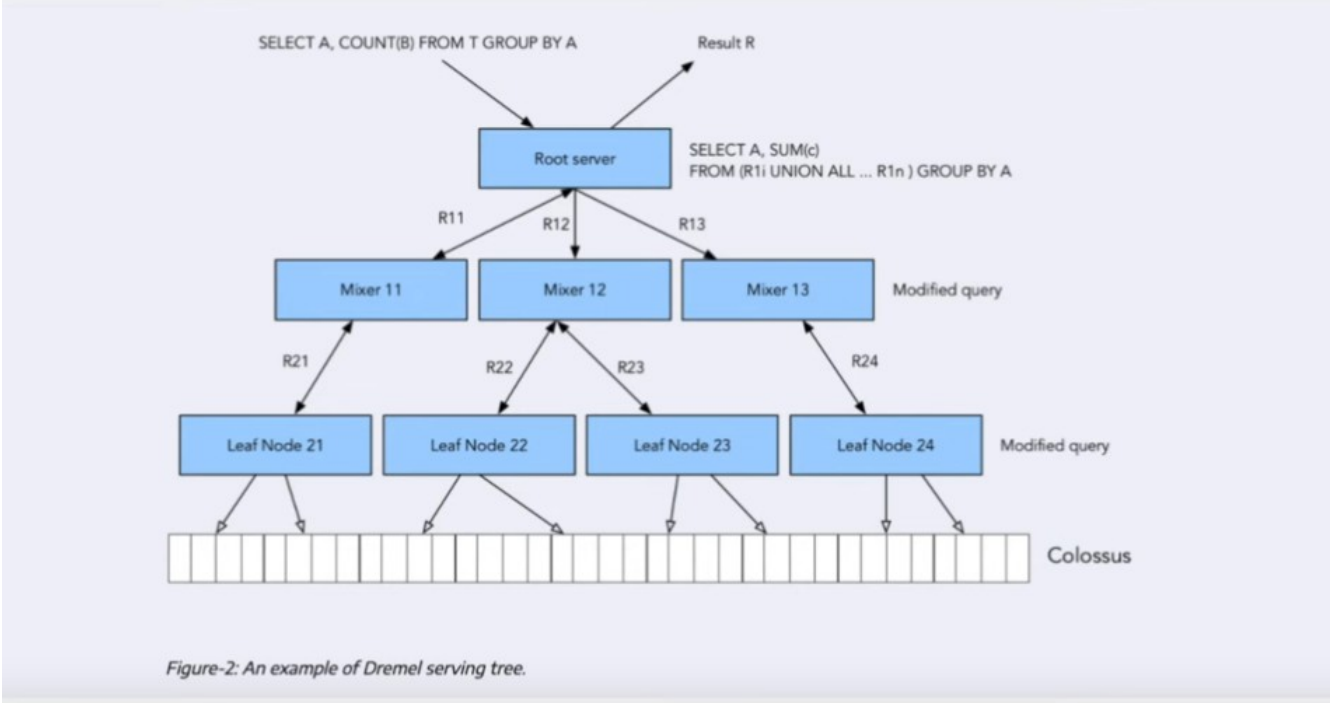


Figure-2: An example of Dremel serving tree.