Data Lakes and Data warehousing

Favorite	☐
Archived	☐
Fleeting	☐
Area/Resource	
Project	
Type	
Review Date	
Image	
URL	
Created	@November 24, 2022 10:25 AM
Updated	@February 11, 2023 7:38 PM
Root Area	
Project Area	
Updated (short)	02/11/2023
Pulls	
Resource Pulls	
Project Archived	
URL Base	
🍳 Recipe Divider	🥗🥗🥗 RECIPE BOOK PROPERTIES 🥗🥗🥗
🍳 Recipe Tags	
📚 Book Divider	📚📚📚 BOOK TRACKER PROPERTIES 📚📚📚
📚 Author	

📚 Date Started	
📚 Date Finished	
📚 Book Status	
📚 Rating	

Introduction

💡 A data engineer gets the data where it can be useful, and in usable condition.

A data lake brings data from across the organisation into a single location. It usually has to be replicated in storage.

Considerations when making a data lake:

- Can your lake handle all types of data?
- Can it scale to meet demand
- Does it support High-throughput ingestion
- Is there fine grained access control to objects
- Can other tools access it easily

- Cloud storage i usually good for raw data storage

💡 Data processing is used to make data usable in current form.

Data engineering challenges

1) Access to data

Tools to analyse data across several organisations

2) Data accuracy and quality

Data warehouse stores cleaned and transformed data

- Consolidated place to store data
- Any raw data from source systems need to be cleaned and transformed
- ETL clean up to push into warehouse

3) Availability of compute

4) Optimize performance through queries

Big query

💡 Server-less data warehouse, used as a collective home for all analytical data

Data marts: BigQuery organises data tables into units called datasets

Grants: IAM grants permissions to perform specific actions

Considerations with data warehouses

- Can it server as a sink for both batch and streaming data
- Can the data warehouse scale to meet my needs
- Is it designed for performance

Partner effectively with other teams

As a data engineer you have to add value to data

- ML teams want to have documented data that is clean
- Data analysts need clean data with visible schemas
- Data engineers need data availability

For all of this you need to make you can:

- Monitor your data

- Scale health
- cloud monitoring
    - Track spending
    - Create alerts

Manage access and data governance

- Who should have access
- How is PII handled
- How can we educate end to end users on our data catalog
    - Cloud data catalog is a managed data discovery + loss prevention API for guarding PII

Build production ready pipelines

1) Data lakes

2) Data warehouse

3) Governance policy

- [ ] What does it mean to productionalize

End to end and scalable:

- Include monitoring
- Automate as much as possible

- How can we ensure pipeline health and cleanliness
- How can productionalize these pipelines to min maintenance and max uptime

- How can we respond and adapt to schema changes
- Cloud composer orchestrates workflow

Big Query SQL Queries

Data lakes

Introduction to data lake

💡 A scalable and secure data platform that allows enterprises to ingest, store, and process and analyze any type of volume.

- Data sources
    - Data sinks: Reliable way to store and retrieve data
    - Data warehouse: Data that has been transformed into a useful format
- Data pipelines
    - Does the transformation and processing of data at scale, batch and streaming
- High-level orchestration workflows
    - You need a way for the pipeline to be orchestrated
- Analogy (Civil engineer)
    - Raw materials need to be brought to the job site (in the data lake)
    - Materials need to be cut and transformed for purpose and stored (pipelines to data sinks)
        - Workers behind the scene are VMs
    - The actual building is the new insight or the ML model

    - The supervisor directs all aspects and teams on the project (workflow orchestration)
        - new CSV file just dropped so orchestrate the extraction
        - Orchestrator is apache airflow
- It all starts with getting that data into your lake first

Data lake vs Data warehouse

- A data lake is a capture of every apsect of your business operation
    - The data is stored in its natural/raw format, usually as object blobs or files
    - Retain all data in its native format
    - Support all data types and all users
    - Adapt to changes easily
    - Tends to be application specific
- A data warehouse typically has the following characteristics
    - Typically loaded only after a use case is defined
    - Processed/organised/transformed
    - Good for querying and analysis

Data storage and ETL options on GCP

- Cloud storage for catch all
- Cloud SQL and Spanner for relational data
- Firestore and Bigtable for noSQL
- The path your data takes to get to the cloud depends on:
    - Where your data is now
    - How big your data is
    - Where it has to go
    - How much transformation is needed

The method you used to load data into the cloud depends on how much transformation is needed

- If the data is readily available and formatted for analysis then you use **EL**
    - Parquet is supported for federated querying
- **ELT** if the data is not formatted the way you want it for analysis
    - When the amount of transformation that is needed is not very high
    - SQL can be used to transform the data
- **ETL** is used when there is a lot of data to be dropped, and to be transformed.
    - If your data is in a binary format then it must be converted
    - If this transformation is essentially or reduces the size
        - Reduce the network bandwith you need to load in the data
    - ETL transformation takes place in a staging area before being loaded into the warehouse

Build a data lake using cloud storage

> 💡 Cloud storage is a common use case for a data lake this is because the data is usually cheaper, persistent and allows for high throughput.

- Strong consistency
    - Can be reached anywhere

How does cloud storage work?

> 💡 It is an object store. Buckets are container for objects. Objects only exist within buckets.

- A bucket name is a global name

    - simplifies locating buckets
- Each bucket is associated with a particular region
- When an object is stored cloud storage replicates it. This makes the data much more durable.
    - Objects replicated across regions and zones
- High throughput is achieved as replicas can serve data
- Objects contain metadata

Storage classes:

- Standard storage: Hot data, frequently accessed
- Nearline storage: Infrequently accessed data
    - 30 day minimum duration
- Coldline storage: Accessed once every 90 days
- Archive storage: Only access data once a year

- [ ] What is bandwith, throughput and latency

Cloud storage simulates a file system

- Can be accessed via the web or gc cli

Cloud storage has object management

Secure cloud storage

Controlling access with IAM and access lists

💡 IAM provides access control on object level while access lists provides access on object level

- Bucket roles
    - Read, writer, owner and set acl policy

💡 All data in google cloud is encrypted at rest, and in transit. Google uses GMEK (google-managed encryption keys).
You can use CMEK (Customer managed encryption keys
You can use CSEK customer supplied encryption keys
Client side encryption: Encrypted before it is uploaded.

Store all types of data

💡 Cloud storage is not optimized for high frequency writes, the latency is not low enough, for transactional workloads use cloud SQL. You also do not want to use cloud storage for analytics on structured data. If you do that you will spend a significant amount of compute on parsing data.

- Use BigQuery for analytics workloads.

Transactional systems are write heavy; 80% writes and 20% reads

Cloud spanner: Better for globally available data, also useful if database is too big to fit into a single cloud SQL instance

Cloud Bigtable: Analytical workloads for high throughput

Cloud SQL as a relational data lake

💡 Cloud sql makes managing a database easy without having to handle compute and servers. GCP will backup, security updates. You can treat a SQL database as a service.

- Can be integrated with other google services
- Backup, recovery, scaling and security is managed for you

- [ ] Vertical scaling vs horizontal scaling

- Automatic replication
- Cloud SQL can handle fail overs through fail over replicas

Fully managed vs Server-less

Fully managed: No setup, automated backups, updates, etc. Replicated, highly available

Server-less: No server management, Fully managed security, pay for what you use

- Always choose the server less product if all other things are equal

Data warehouses

> 💡 A data warehouse should consolidate data from many sources. Which means it imposes a schema. The analyst should know the schema to analyze it.

- The data should be optimized for simplicity of access and high-speed query performance
- A data lake is made for storage
- Data warehouse data should be available for querying and be quick

> 💡 A modern data warehouse can handle huge amounts of data that do not fit into memory. Server-less and no-ops, you are not limited by servers or indexes to maintain.

- 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 GCP data warehouse, which can handle large PB scale datasets in seconds.

- Cost effectively handles large datasets, which is almost as cost effective as cloud storage
- Serverless and no-ops, including ad hoc queries
- All other benefits
    - Supports SQL queries
- BigQuery is server-less and fully-managed service
    - People are not having to worry about common tasks
    - The storage engine handles and how data is stored and replicated
- BigQuery tables are column-oriented, which is efficient for high-read throughput
- BigQuery tables are immutable, and are not optimized for updating
- The data is physically stored in a redundant way separate from the computer cluster
    - Uses colossus to store data and jupiter to communicate between storage and processing
- BigQuery allocates storage and resources based on usage patterns
- A bigquery slot is a combination of CPU, memory and networking resources
    - allocates slots to queries based on intensity
- Implemented using a micro-service architecture
- Analytics throughput is measured via bigquery slots

- [ ] Look into bigquery slots more
- [ ] Look into github for google cloud training data analysts

Big query demo

All datasets are under resources

- Some operations are more expensive than others
- Slot time consumed: the linear time it would take to do the work in a query, on a single machine

Get started with BigQuery

BigQuery organizes data tables into units called datasets. You refer to tables as project.dataset.table

- Helps you structure your information logically, you can also isolate your data according to it's needs.
- BigQuery datasets belong to a project
    - IAM permissions are needed to submit a job to the service
        - You need at least read management
        - Access control is through IAM, can be configured on dataset, table, view or column
        - A user has to run jobs to query
- Bigquery datasets can be multiregional
- Every table has a schema
    - You can enter a schema manually or by supplying a JSON file
- BigQuery storage encrypts data at rest
- Use google workspace or Gmail accounts for these tasks
- Access controls is at the level of dataset table, view or columns
- Cloud Audit logs
    - Admin activity
    - System event
        - Table exploration
    - Data access

- BigQuery provides predefined roles for controlling access to resources
    - IAM grants permission to perform specific actions
    - Access control is to datasets
- Row-level security
    - Create row access policy
    - based on regions etc…
- Creating an authorized view
    - Share query results with a particular group without giving them the underlying access to the data
- Share datasets with other analysts is easy
    - Grant access to relevant projects
    - Share queries to get acquainted with the data
    - Analysts can control tasks on the console based on the permission you gave them

- [ ] What is a view?

💡 A view is a SQL query that looks like a table, but only shows the results of a query.
A materialized view: A pre-computer data set derived from a query specification.

- Can improve the performance of workloads that have the characteristic of common, and repeated queries

BigQuery query service

- Querying native tables is the most performant way to use bigquery
- Data can be queried in eternal sources: They are called federated queries
- The query job can also write to a destination table

- Use query validator with pricing calculator for estimates.

BigQuery pricing

- **Analysis pricing** is the cost to process queries, including SQL queries, user-defined functions, scripts, and certain data manipulation language (DML) and data definition language (DDL) statements that scan tables.
- **Storage pricing** is the cost to store data that you load into BigQuery.

- [ ] Look into slot pricing for flat-rate pricing

BigQuery slots: You have access to up to 2000 concurrent slots, share among all queries in a single project. BigQuery will periodically burst beyond this limit to accelerate smaller queries, in addition you might occasionally have fewer slots available if there is a high amount of contention for on-demand capacity in a specific location.

Pricing

- BigQuery uses a columnar data structure. You're charged according to the total data processed in the columns you select, and the total data **per column** is calculated based on the **types** of data in the column.
- You aren't charged for queries that return an error or for queries that retrieve results from the cache. For procedural language jobs this consideration is provided at a per-statement level.
- Canceling a running query job might incur charges up to the full cost for the query if you let the query run to completion.
- You are charged for the amount of data processed in the columns you select, even if you set an explicit LIMIT on the results
- Partitioning and clustering your tables can help reduce the amount of data processed by queries. As a best practice, use partitioning and clustering whenever possible.
- When querying an external data source from BigQuery, you are charged for the number of bytes read by the query. If the external data is stored in another Google

Cloud product such as Cloud Storage, any storage costs for that product apply as well. For more information, see Google Cloud pricing.

- Is it more efficient in terms of speed and cost to store data in big query?

BigQuery provides cost control mechanisms that enable you to cap your query costs. You can set:

- User-level and project-level custom cost controls
- The maximum bytes billed by a query

For detailed examples of how to calculate the number of bytes processed, see Query size calculation.

> ⚠️ You are able to separate the cost of storage and the cost of queries

Load new data into big query

EL: Data is imported as is. Source and target has same schema

ELT: Less transformation necessary

ETL: Staged and transformed in intermediate form

Batch loading data into BQ supports different file formats:

- CSV: Auto detect, manual verification
- NEWLINE_DELIMITED_JSON: Auto detech, manual verification
- AVRO: Self describing, determined directly
- DATASTORE_BACKUP
- PARQUET: Needs to load schema as JSON

- Load job creates a destination table
- BigQuery sets daily limits on the sizes of the number of load jobs
- Use cloud functions, to listen to cloud storage event that is associated to new files arriving and launch a load job

- BigQuery can import data from mostly everywhere
- Loading data through Cloud storage is useful in the EL process

Automate the execution of queries based on a schedule

- Can schedule queries to cache for later use

Data transfer service

> 💡 Schedule automatic transfer of data from wherever you need. Stage the data, so it can be cleaned and transformed.

- Use the data transfer service for all repeated jobs that extract data from a software somewhere

> ✏️ Data backfill: Detect missing data, and request that all analytic processes work as expected

Loading and transforming data in BigQuery, you can load the data into a staging area, and then use SQL to transform the data into usable data from SQL commands.

- BigQuery supports standard DML statements
- Also supports DDL statements
- Supports UDF, user defined functions
    - Basically macros with dbt
- UDF working with javascripts can include libraries
- You are able to share your UDFs with other members or publicly

Schemas

💡 As an engineer you usually have very little time to learn key information about new data tables, in this case you need to know how to query metadata, to quickly learn about tables.

- Use INFORMATION_SCHEMA and TABLES to explore metadata

DEMO

- You can query datasets using the __TABLES__ from <project>.<dataset> will give you great information about the metadata, including:
    - size of data
    - creation time
    - modified time
    - number of rows
    - view or table
- You can query datasets using INFORMATION_SCHEMA.COLUMNS which will show you the number of columns in a table, and other column data
- You can UNION tables metadata together to see all dataset metadata in a project

Schema design

💡 Transactional databases often use normal form. Normalizing the data means turning it into a relational system.
Denormalizing means you allow duplicate values in column fields to gain processing performance. It takes more storage, but it can be processed in parallel. BigQuery can better query data, that is denormalized.

Denormalizing data on a 1-to-many field in flattened data can cause shuffling of data over the network.

- Avoid shuffling the data

- Nested and repeated data is allowed in BigQuery, it is useful for working with data that is used in relational databases.
    - It is the best alternative for data that already has a relational pattern to it
    - This increases bigquery performance

Nested and repeated fields

A usual question is should we normalize or de-normalized.

> 💡 Normalized schemas, are join intensive which is computationally expensive. RDBMS are records based which means each row has to be pulled out.

> ⚠️ In de-normalized we must be careful of data in different letters of granularity. When aggregating we might be at risk of replicating several same data points.

> ⚠️ Nested data allows you to benefit from de-normalized data while having more granular data being handled via nested fields.

Struct: A SQL datatype that can be thought of as pre-joined tables within a table.

- Struct = Record on BigQuery
- Array datatype is seen as a REPEATED mode
    - Can be of any single type
    - Can be part of regular fields or structs
- A single table can have many structs

DEMO with nested and repeated columns

> 💡 Nested data, and repeated columns allows you to handle different levels of granularity in your data.

- Structs are much faster, and use less processing power

- [ ] What is a correlated cross join?
- [ ] Watch demo again

> 💡 It is much more efficient to design schemas using nested repeated fields in de-normalized tables. If the data is smaller than 10 gb keep them normalized

- As a dataset increases in size, the performance impact of a join can increase substantially.

Working with JSON and Array Data

> ⚠️ Data in an array must only have one type. They must all be the same.

- Finding the number of elements with `ARRAY_LENGTH(<array>)`
- Deduplicating elements with `ARRAY_AGG(DISTINCT <field>)`
- Ordering elements with `ARRAY_AGG(<field> ORDER BY <field>)`
- Limiting `ARRAY_AGG(<field> LIMIT 5)`

Use the UNNEST function on your array field to query it.

- You need to UNNEST() arrays to bring the array elements back into rows
- UNNEST() always follows the table name in your FROM clause (think of it conceptually like a pre-joined table)

A STRUCT can have:

- One or many fields in it
- The same or different data types for each field

- Its own alias

Storing your large reporting tables as STRUCTs (pre-joined "tables") and ARRAYs (deep granularity) allows you to:

- Gain significant performance advantages by avoiding 32 table JOINs
- Get granular data from ARRAYs when you need it but not be punished if you don't (BigQuery stores each column individually on disk)
- Have all the business context in one table as opposed to worrying about JOIN keys and which tables have the data you need

💡 A cross join generates a paired combination of each row of the first table with each row of the second table.

Partitioning and clustering optimization

💡 A partitioned table allows you to optimize your queries based on chosen partitions and with massively reduce the amount of data to process.

- A partitioned table requires bigquery to maintain more metadata.
- Bigquery allows you to partition tables through:
    - Ingestion time
    - Any column that is of type datetime
    - Integer-typed column
- A good practice is to require the query to always include the partition filter

- [ ] Look at the blog optimizing big query

💡 Clustering can improve the performance of certain types of queries. Filtering or Aggregation.

- BigQuery automatically sorts the data based on values in the clustering columns
- Set up clustering at table creation time

- [ ] Look more ignore clustering

⚠️ As more and more operations modify a table, the degree of which data is sorted weakens.

- BigQuery performs auto re-clustering
- Use clustering when you commonly use filters or aggregation against particular columns in your queries.