BigQuery Datawarehouse

BigQuery - Datawarehouse

In 28 Minutes

- **Exabyte scale modern Datawarehousing** solution from GCP
  - Relational database (SQL, schema, consistency etc)
    - Use **SQL-like commands** to query massive datasets
  - Traditional (Storage + Compute) + Modern (Realtime + Serverless)
- When we are talking about a Datawarehouse, **importing and exporting data (and formats) becomes very important**:
  - Load data from a **variety of sources, incl. streaming data**
    - Variety of import formats - CSV/JSON/Avro/Parquet/ORC/Datastore backup
  - Export to Cloud Storage (long term storage) & Data Studio (visualization)
    - Formats - CSV/JSON (with Gzip compression), Avro (with deflate or snappy compression)
- Automatically expire data (**Configurable Table Expiration**)
- Query **external data sources** without storing data in BigQuery
  - Cloud Storage, Cloud SQL, BigTable, Google Drive
  - Use **Permanent or Temporary** external tables

BigQuery - Accessing and Querying Data

In28 Minutes

- **Access databases using:**
  - Cloud Console
  - bq command-line tool (NOT gcloud)
  - BigQuery Rest API OR
  - HBase API based libraries (Java, .NET & Python)

- (Remember) BigQuery queries **can be expensive** as you are running them on large data sets!
- (BEST PRACTICE) **Estimate BigQuery queries before running:**
  - **1:** Use UI(console)/bq(--dry-run) - Get scanned data volume (estimate)
  - **2:** Use Pricing Calculator: Find price for scanning 1 MB data. Calculate cost.

Partitioning and Clustering BigQuery Tables - Use Case

In 28 Minutes

- You pay for BigQuery queries by the amount of data scanned
- How do you reduce your costs of querying BigQuery and improve performance?
- Scenario: Imagine a Questions table with millions of rows
  - You want to find all questions asked between a date range (date between `2022-10-02` and `2028-10-02`) belonging to a specific category
    - If you have a single questions table you need to scan all the rows
      - **Partitioning** - Divide table into multiple segments (example: by date)
      - **Clustering** - Group related data (example: by category)

Questions		
Date	Question	Category
2025-10-02	Question Detail ...	GCP
2025-10-02	Question Detail ...	AWS
2025-10-02	Question Detail ...	GCP
2025-10-03	Question Detail ...	Azure
2025-10-03	Question Detail ...	GCP
2025-10-03	Question Detail ...	Azure

Partitioning and Clustering BigQuery Tables

In 28 Minutes

- **Partitioning**: Table is divided into segments
  - Makes it easy to manage and query your data
  - Improves performance and reduces costs
  - Partition based on Ingestion time (arrival time) OR a column (TIMESTAMP, DATE, or DATETIME , or INTEGER)
  - (DEFAULT) All partitions will share same schema as table
  - Allows you to expire (delete) parts of table data easily (partition_expiration_days)
- **Clustering**: Organize table data based on the contents of one or more columns
  - Goal: colocate related data and eliminate scans of unnecessary data
  - Avoid creating too many small partitions (of less than 1 GB). In those cases, prefer Clustering.

Questions_2025_10_02

Date	Question	Category
2025-10-02	Question Detail	AWS
2025-10-02	Question Detail	GCP
2025-10-02	Question Detail	GCP

Questions_2025_10_03

Date	Question	Category
2025-10-03	Question Detail	Azure
2025-10-03	Question Detail	Azure
2025-10-03	Question Detail	GCP

Partitioning and Clustering BigQuery Tables - Syntax

In 28 Minutes

```
CREATE TABLE `my_data_set.questions_partitioned_and_clustered`
...
...
PARTITIONED BY
    DATE(created_date)
    CLUSTER BY category
...
OPTIONS (
  expiration_timestamp=TIMESTAMP "2025-01-01 00:00:00 UTC",
  partition_expiration_days=7
)
```

Expiring Data in BigQuery

In 28 Minutes

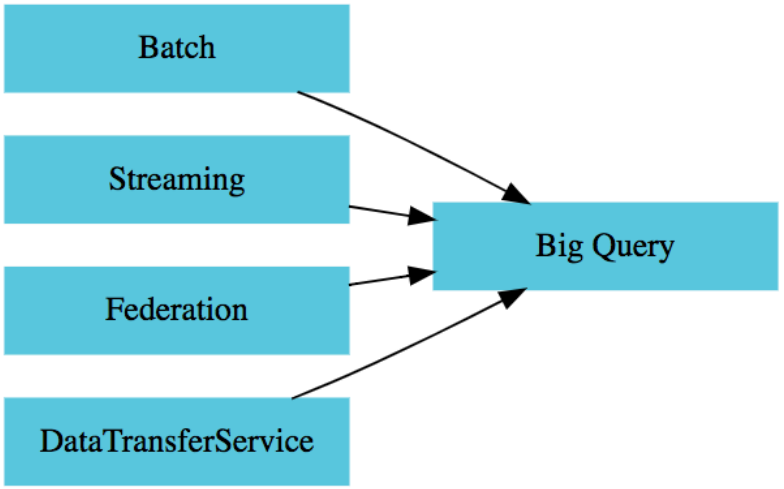
```
CREATE SCHEMA mydataset
OPTIONS(
  default_table_expiration_days=3.75
  )

ALTER TABLE mydataset.mytable
SET OPTIONS (
  expiration_timestamp=TIMESTAMP "2025-01-01 00:00:00 UTC",
  partition_expiration_days=7
)
```

- You pay for data stored in BigQuery:
  - How can you automatically delete (expire) data which is not needed?
- Big Query Hierarchy : Data Set > Table > Partitions
  - You can configure expiration at each level
    - Configure default table expiration (default_table_expiration_days) for datasets
    - Configure expiration time (expiration_timestamp) for tables
    - Configure partition expiration (partition_expiration_days) for partitioned tables
- Best Practice: Expire Tables and Partitions you are NOT using!

Importing Data into BigQuery

In28 Minutes

- **Batch Import** (FREE):
  - Import from Cloud Storage and local files
  - Import after processing by Cloud Dataflow and Cloud Dataproc
- **Streaming Import** ($$$$):
  - From Cloud Pub/Sub, Streaming Inserts
  - Import after processing by Cloud Dataflow and Cloud Dataproc
- **Federated Queries**: Query **external data**
  - Cloud Storage, Cloud SQL, BigTable, Google Drive
- **BigQuery Data Transfer Service**: Import from
  - Google SaaS apps (Google Ads, Cloud Storage etc)
  - External cloud storage providers - Amazon S3
  - Data warehouses - Teradata, Amazon Redshift



Streaming Data into BigQuery

- Loading data in bulk is free **but streaming data is NOT FREE**
  - AND there are a lot of limitations (Use with caution!)
- Streaming data can contain duplicates. How can you avoid duplicates?
  - Add **insertId** with each streaming insert:
    - insertId is used to provide best effort de-duplication (for up to one minute)
      - For strict de-duplication and transactions, try Google Cloud Datastore
- There are **strict streaming quotas** with BigQuery:
  - IF you are NOT populating `insertId`:
    - Maximum bytes per second - 1 GB per second, per project (REMEMBER per project - NOT per table)
  - ELSE (i.e. you are using `insertId`)
    - Maximum rows per second per project
      - US and EU multi-regions: 500,000, Other locations: 100,000
      - per table limitation: 100,000
    - Maximum bytes per second: 100 MB
  - If you have streaming of millions of rows per second, prefer BigTable!

Understanding BigQuery Best Practices

In 28 Minutes

- **Estimate your queries** before running them
  - bq --dry_run flag or dryRun API parameter
- Use clustering and partitioning for your tables
- **Avoid streaming inserts** when possible
  - Loading data in bulk is free but streaming data is NOT FREE
  - Offers Best effort de-duplication (when you use `insertId`)
  - Remember Quota limits
- **Expire Data Automatically**:
  - Configure default table expiration (default_table_expiration_days) for datasets
  - Configure expiration time for tables
  - Configure partition expiration for partitioned tables

Understanding BigQuery Best Practices - 2

In 28 Minutes

- Consider **Long-term storage** option
  - Long-term storage: Table in which data is NOT edited for 90 consecutive days
  - Lower Storage cost - Similar to Cloud Storage Nearline
- BigQuery is fast for **complex queries**:
  - BUT it is not as well optimized for narrow-range queries (Prefer Cloud Bigtable)
  - (REMEMBER) Too much complexity in setting up a query
- **Optimize BigQuery usage** using audit logs:
  - Analyze queries/jobs that were run earlier
  - Stream your audit logs (BigQueryAuditMetadata) to BigQuery
    - Understand usage patterns (query costs by user)
    - Optimize (visualize using Google Data Studio)

Cloud Dataproc

In 28 Minutes

- Managed **Spark and Hadoop** service:
  - Variety of jobs are supported:
    - Spark, PySpark, SparkR, Hive, SparkSQL, Pig, Hadoop
  - Perform complex batch processing
- **Multiple Cluster Modes:**
  - Single Node / Standard/ High Availability (3 masters)
  - Use regular/preemptible VMs
- Use case: Move your Hadoop and Spark clusters to the cloud
  - Perform your machine learning and AI development using open source frameworks
- (REMEMBER) Cloud Dataproc is a data analysis platform
  - You can export cluster configuration but NOT data
- (ALTERNATIVE) BigQuery - When you run SQL queries on Petabytes
  - Go for Cloud Dataproc when you need more than queries (Example: Complex batch processing Machine Learning and AI workloads)