BigQuery Features

Bigquery Table Partitions

A partitioned table is divided into segments, called partitions, that make it easier to manage and query your data. By dividing a large table into smaller partitions, you can improve query performance and control costs by reducing the number of bytes read by a query. You partition tables by specifying a partition column which is used to segment the table.

If a query uses a qualifying filter on the value of the partitioning column, BigQuery can scan the partitions that match the filter and skip the remaining partitions. This process is called *partition pruning*.

Partition pruning is the mechanism BigQuery uses to eliminate unnecessary partitions from the input scan. The pruned partitions are not included when calculating the bytes scanned by the query. In general, partition pruning helps reduce query cost.

When to use partitioning?

- Want to improve the query performance by scanning portion of table
- To reduce the cost of bytes read by query

Limitations

- We cannot use legacy SQL to query partitioned tables or to write query results to partitioned tables.
- BigQuery does not support partitioning by multiple columns. Only one column can be used to partition a table.
- Bigquery supports only 4000 partitions per table at present

Supported Types of Table Partitions

Integer Range

We can partition a table based on ranges of values in a specific INTEGER column. To create an integer-range partitioned table, you provide:

- The partitioning column.
- The starting value for range partitioning (inclusive).
- The ending value for range partitioning (exclusive).
- The interval of each range within the partition.

Example:(1,100,10)-->(start,end,interval)

Time Unit Column

We can partition a table on a DATE,TIMESTAMP, or DATETIME column in the table. When we write data to the table, BigQuery automatically puts the data into the correct partition, based on the values in the column. The partitions can have either hourly, daily, monthly, or yearly granularity.Partition boundaries are based on UTC time.
Example: 2019-01-01,2021-05-07 17:22:00

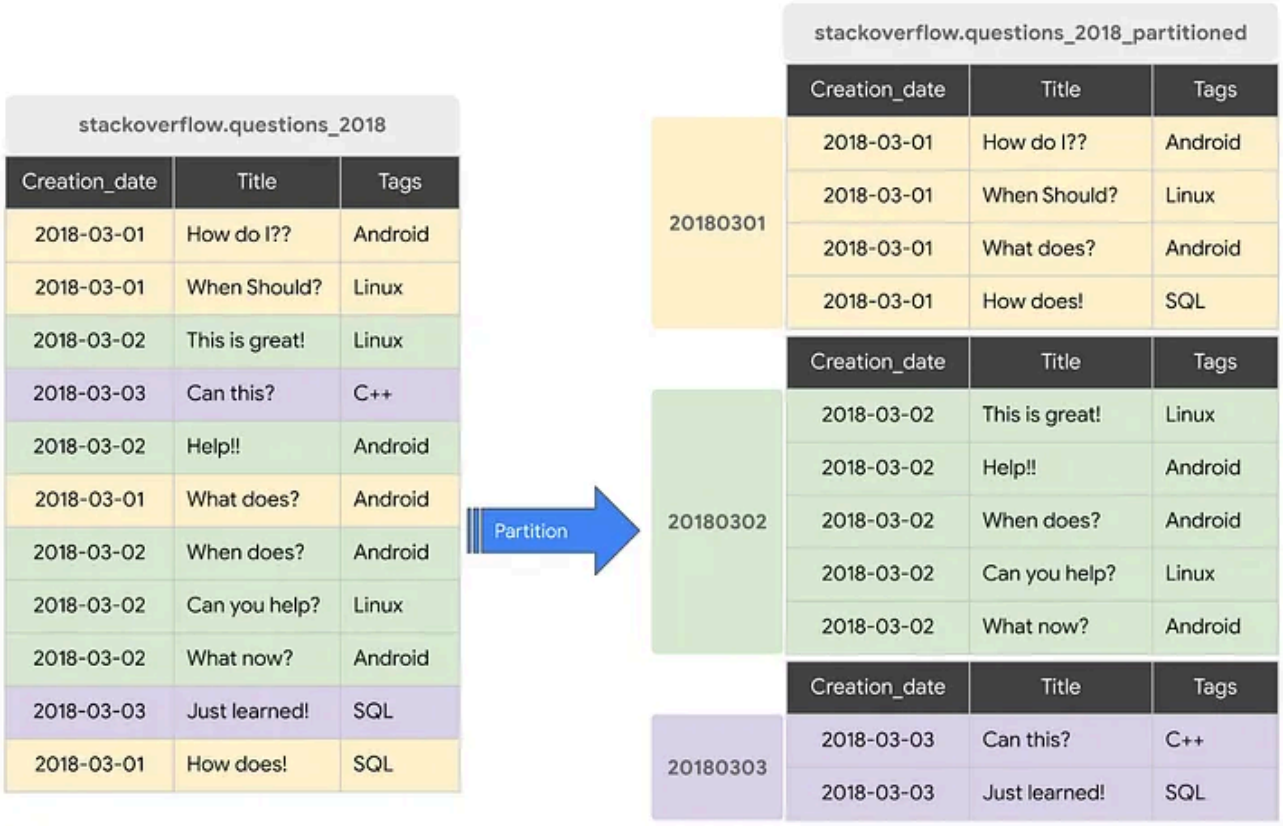
Ingestion Time

When you create a table partitioned by ingestion time, BigQuery automatically assigns rows to partitions based on the time when BigQuery ingests the data. You can choose hourly, daily, monthly, or yearly granularity for the partitions. Partition boundaries are based on UTC time.
An ingestion-time partitioned table has a pseudocolumn named _PARTITIONTIME. The value of this column is the ingestion time for each row, truncated to the partition boundary (such as hourly or daily).
Instead of using _PARTITIONTIME, you can also use _PARTITIONDATE. The _PARTITIONDATE pseudocolumn contains the UTC date corresponding to the value in the _PARTITIONTIME pseudocolumn.

Example: 2021-05-07 17:40:00

Special Partitions

- **__NULL__**: Contains rows with NULL values in the partitioning column.
- **__UNPARTITIONED__**: Contains rows where the value of the partitioning column is outside the range



Bigquery Table Clustering

Clustered tables in BigQuery are tables that have a user-defined column sort order using *clustered columns*. Clustered tables can improve query performance and reduce query costs.

In BigQuery, a *clustered column* is a user-defined table property that sorts storage blocks based on the values in the clustered columns. The storage blocks are adaptively sized based on the size of the table. A clustered table maintains the sort properties in the context of each operation that modifies it. Queries that filter or aggregate by the clustered columns only scan the relevant blocks based on the clustered columns instead of the entire table or table partition. As a result, BigQuery might not be able to accurately estimate the bytes to be processed by the query or the query costs, but it attempts to reduce the total bytes at execution.

When you cluster a table using multiple columns, the column order determines which columns take precedence when BigQuery sorts and groups the data into storage blocks.

The following example compares the logical storage block layout of an unclustered table with the layout of clustered tables that have one or multiple clustered columns:

When you query a clustered table, you don't receive an accurate query cost estimate before query execution because the number of storage blocks to be scanned is not known before query execution. The final cost is determined after query execution is complete and is based on the specific storage blocks that were scanned.

When to use clustering?

- If our queries commonly filter on particular columns, this clustering accelerates query performance
- If our queries filter on columns that have many distinct values (high cardinality), clustering accelerates these queries by providing BigQuery with detailed metadata.

Limitations

- Only GoogleSQL is supported for querying clustered tables and for writing query results to clustered tables.
- You can only specify up to four clustering columns. If you need additional columns, consider combining clustering with partitioning.

Bigquery GIS functions

Google's BigQuery is a data warehouse tool with first-class support for GIS functions and data types. BigQuery GIS lets us analyze and visualize geospatial data in BigQuery by using geography data types and standard SQL geography functions.

We can visualize our BigQuery GIS Data using BigQuery Geo Viz . BigQuery Geo Viz is a web tool for visualizing geospatial data in BigQuery on a map, one query at a time. We can run a SQL query and display the results on an interactive map.

Features

- BigQuery GIS supports spatial data types such as points, lines, polygons, and multi-geometries, allowing users to represent and analyze various geographic features accurately.
- It provides a rich set of spatial functions for performing geometric operations, including distance calculations, geometric transformations, intersection analysis, and spatial joins.
- BigQuery GIS allows users to create compelling visualizations and dashboards to communicate spatial insights effectively.

Limitations

- Geography functions are available only in GoogleSQL.
- BigQuery GIS primarily supports spatial data stored in GeoJSON or Well-Known Text (WKT) formats, which may require data conversion for users with data in other formats.

Bigquery ML

BigQuery ML is a groundbreaking service offered by Google Cloud Platform that integrates machine learning directly into Google BigQuery, a serverless, highly scalable, and cost-effective data warehouse. This innovative tool empowers data analysts and data scientists to build and deploy machine learning models without the need to transfer data to other environments or learn separate ML frameworks. This document aims to provide a comprehensive overview of BigQuery ML, including its features, supported models, advantages, and limitations.

Supported Models

- Linear Regression: Used for predicting numerical values based on input features.
- Logistic Regression: Employed for binary classification tasks, such as predicting whether an email is spam or not.
- K-means Clustering: Facilitates unsupervised learning by partitioning data into clusters.
- Matrix Factorization: Commonly used for recommendation systems to predict user preferences.

- TensorFlow Deep Neural Networks: Allows for building and training custom deep learning models using TensorFlow syntax.
- AutoML Tables: Provides automated machine learning capabilities for tabular data, enabling users to quickly train high-quality models without extensive manual intervention.

Advantages

- BigQuery ML seamlessly integrates machine learning capabilities into BigQuery, enabling users to leverage their existing SQL skills to create and deploy machine learning models.
- BigQuery ML can handle large datasets efficiently, making it suitable for organizations dealing with massive volumes of data.

Limitations

BigQuery ML is tightly integrated with Google Cloud Platform, which may restrict organizations that prefer multi-cloud or on-premises solutions.