Google Discussions

Exam Professional Data Engineer All Questions

View all questions & answers for the Professional Data Engineer exam

Go to Exam

EXAM PROFESSIONAL DATA ENGINEER TOPIC 1 QUESTION 302 DISCUSSION

Actual exam question from Google's Professional Data Engineer

Question #: 302

Topic #: 1

[All Professional Data Engineer Questions]

You work for a farming company. You have one BigQuery table named sensors, which is about 500 MB and contains the list of your 5000 sensors, with columns for id, name, and location. This table is updated every hour. Each sensor generates one metric every 30 seconds along with a timestamp, which you want to store in BigQuery. You want to run an analytical query on the data once a week for monitoring purposes. You also want to minimize costs. What data model should you use?

A. 1. Create a metrics column in the sensors table.
2. Set RECORD type and REPEATED mode for the metrics column.
3. Use an UPDATE statement every 30 seconds to add new metrics.

B. 1. Create a metrics column in the sensors table.
2. Set RECORD type and REPEATED mode for the metrics column.
3. Use an INSERT statement every 30 seconds to add new metrics.

C. 1. Create a metrics table partitioned by timestamp.
2. Create a sensorId column in the metrics table, that points to the id column in the sensors table.
3. Use an INSERT statement every 30 seconds to append new metrics to the metrics table.
4. Join the two tables, if needed, when running the analytical query.

D. 1. Create a metrics table partitioned by timestamp.
2. Create a sensorId column in the metrics table, which points to the id column in the sensors table.
3. Use an UPDATE statement every 30 seconds to append new metrics to the metrics table.
4. Join the two tables, if needed, when running the analytical query.

Show Suggested Answer

by scaenruy at *Jan. 4, 2024, 1:54 p.m.*

Comments

Type your comment...

Submit

raaad Highly Voted 1 year, 4 months ago

Partitioned Metrics Table: Creating a separate metrics table partitioned by timestamp is a standard practice for time-series data like sensor readings. Partitioning by timestamp allows for more efficient querying, especially when you're only interested in a specific time range (like weekly monitoring).
Reference to Sensors Table: Including a sensorId column that references the id column in the sensors table allows you to maintain a relationship between the metrics and the sensors without duplicating sensor information.
INSERT Every 30 Seconds: Using an INSERT statement every 30 seconds to the partitioned metrics table is a standard approach for time-series data ingestion in BigQuery. It allows for efficient data storage and querying.
Join for Analysis: When you need to analyze the data, you can join the metrics table with the sensors table based on the sensorId, allowing for comprehensive analysis with sensor details.

upvoted 10 times

rajshiv Most Recent 3 weeks, 2 days ago

Selected Answer: C

C is the best answer.
It cannot be A and B - as Embedding a RECORD type (nested structure) in the sensors table and updating it every 30 seconds is inefficient and expensive. Moreover, BigQuery is not designed for frequent updates or modifying nested fields repeatedly. Plus It increases storage and write costs significantly.
It cannot be D - Even though it uses a good table design (separate metrics table with timestamp partition), using UPDATE every 30 seconds to append data is inefficient as BigQuery is not optimized for UPDATE-heavy workloads.

upvoted 1 times

plum21 3 months ago

Selected Answer: C

C. B is not feasible – update on the metrics column will be required in such a case or an insert with all sensor data with one-element array of metrics which does not make any sense.

upvoted 1 times

Pime13 3 months, 4 weeks ago

Selected Answer: C

This approach offers several advantages:

Cost Efficiency: Partitioning the metrics table by timestamp helps reduce query costs by allowing BigQuery to scan only the relevant partitions.
Data Organization: Keeping metrics in a separate table maintains a clear separation between sensor metadata and sensor metrics, making it easier to manage and query the data2.
Performance: Using INSERT statements to append new metrics ensures efficient data ingestion without the overhead of frequent updates

upvoted 2 times

7787de3 7 months, 3 weeks ago

Selected Answer: C

Because "Minimize costs" was requested, i would go for C.
Storage cost will be lower for partitions where no writes took place for a certain amount of time, see https://cloud.google.com/bigquery/pricing#storage
Partitioning by timestamp can be configured to use daily, hourly, monthly, or yearly partitioning - so if you choose daily partitioning, the number of partitions should not be an issue.
Working with RECORDS (A,B) would be an option if performance was in focus.

upvoted 2 times

dac9215 9 months ago

Option C will not violate partitioning limit of 4000 as the lowest grain of partitioning is hourly

upvoted 3 times

vbrege 10 months, 2 weeks ago

Selected Answer: B

Here's my logic (some people have already said same thing)

Cannot be C and D
- Total 5000 sensors are sending new timestamp every 30 seconds. If you partition this table with timestamp, you are getting partitions above 4000 (single job) or 10000 (partition limit) so option C and D don't look correct
- For C and D, also need to consider that BigQuery best practices advise to avoid JOINs and use STRUCT and RECORD types to solve the parent-child join issue.

Now coming back to A and B, we will be adding sensor readings for every sensor. I don't think this is a transactional type database where you need to update data. You will add new data for more accurate analysis later so A is discarded. BigQuery best practices also advise to avoid UPDATE statements since its an Analytical columnar database

B is the correct option.

upvoted 4 times

apoio.certificacoes.closer 4 months, 1 week ago

Avoid Joins when tables are large. The sensors table is 500mb, hardly anything. The only watchout is for multiplication of columns when joining.

upvoted 1 times

Gloups 11 months, 1 week ago

Selected Answer: A

Since BigQuery tables are limited to 4000 partitions, options C & D are discarded. Option B is wrong as insertion is invalid too. So option A.

upvoted 3 times

gabrielosluz 3 months, 2 weeks ago

I also thought about this limitation. But researching about partitioning with timestamp, I found this in the documentation:

"For TIMESTAMP and DATETIME columns, the partitions can have either hourly, daily, monthly, or yearly granularity. For DATE columns, the partitions can have daily, monthly, or yearly granularity."

In other words, I believe that even with timestamp partitioning, it would not reach this limit. What do you think?

upvoted 1 times

anushree09 1 year ago

I'm in favor of Option B
Reason: BQ has nested columns feature specifically to address these scenarios where a join would be needed in a traditional/ relational data model. Nesting field will reduce the need to join tables, performance will be high and design will be simple

upvoted 4 times

96f3bfa 1 year, 2 months ago

Selected Answer: C

Option C

upvoted 1 times

Matt_108 1 year, 3 months ago

Selected Answer: C

Option C

upvoted 2 times

SanjeevRoy91 1 year, 1 month ago

Why C. Partitioning by timestamp could breach the 4000 cap of number of partitions easily. And with soo much less data, why partitioning is required in the first place. Ans should be B

upvoted 3 times

raaad 1 year, 4 months ago

Selected Answer: C

Option C

upvoted 4 times

scaenruy 1 year, 4 months ago

Selected Answer: C

C.
1. Create a metrics table partitioned by timestamp.
2. Create a sensorId column in the metrics table, that points to the id column in the sensors table.
3. Use an INSERT statement every 30 seconds to append new metrics to the metrics table.
4. Join the two tables, if needed, when running the analytical query.

upvoted 1 times

EXAMTOPICS

Platform

- Home
- Examtopics PRO
- All Exams
- Training Courses

© 2024 ExamTopics