Google Discussions

EXAM PROFESSIONAL DATA ENGINEER TOPIC 1 QUESTION 41 DISCUSSION

Actual exam question from Google's Professional Data Engineer
Question #: 41
Topic #: 1
[All Professional Data Engineer Questions]

MJTelco Case Study -

Company Overview -
MJTelco is a startup that plans to build networks in rapidly growing, underserved markets around the world. The company has patents for innovative optical communications hardware. Based on these patents, they can create many reliable, high-speed backbone links with inexpensive hardware.

Company Background -
Founded by experienced telecom executives, MJTelco uses technologies originally developed to overcome communications challenges in space. Fundamental to their operation, they need to create a distributed data infrastructure that drives real-time analysis and incorporates machine learning to continuously optimize their topologies. Because their hardware is inexpensive, they plan to overdeploy the network allowing them to account for the impact of dynamic regional politics on location availability and cost.
Their management and operations teams are situated all around the globe creating many-to-many relationship between data consumers and provides in their system. After careful consideration, they decided public cloud is the perfect environment to support their needs.

Solution Concept -
MJTelco is running a successful proof-of-concept (PoC) project in its labs. They have two primary needs:
✑ Scale and harden their PoC to support significantly more data flows generated when they ramp to more than 50,000 installations.
✑ Refine their machine-learning cycles to verify and improve the dynamic models they use to control topology definition.
MJTelco will also use three separate operating environments `" development/test, staging, and production `" to meet the needs of

running experiments, deploying new features, and serving production customers.

Business Requirements -
✑ Scale up their production environment with minimal cost, instantiating resources when and where needed in an unpredictable, distributed telecom user community.
✑ Ensure security of their proprietary data to protect their leading-edge machine learning and analysis.
✑ Provide reliable and timely access to data for analysis from distributed research workers
✑ Maintain isolated environments that support rapid iteration of their machine-learning models without affecting their customers.

Technical Requirements -
Ensure secure and efficient transport and storage of telemetry data
▪
✑ Rapidly scale instances to support between 10,000 and 100,000 data providers with multiple flows each.
✑ Allow analysis and presentation against data tables tracking up to 2 years of data storing approximately 100m records/day
✑ Support rapid iteration of monitoring infrastructure focused on awareness of data pipeline problems both in telemetry flows and in production learning cycles.

CEO Statement -
Our business model relies on our patents, analytics and dynamic machine learning. Our inexpensive hardware is organized to be highly reliable, which gives us cost advantages. We need to quickly stabilize our large distributed data pipelines to meet our reliability and capacity commitments.

CTO Statement -
Our public cloud services must operate as advertised. We need resources that scale and keep our data secure. We also need environments in which our data scientists can carefully study and quickly adapt our models. Because we rely on automation to process our data, we also need our development and test environments to work as we iterate.

CFO Statement -
The project is too large for us to maintain the hardware and software required for the data and analysis. Also, we cannot afford to staff an operations team to monitor so many data feeds, so we will rely on automation and infrastructure. Google Cloud's machine learning will allow our quantitative researchers to work on our high-value problems instead of problems with our data pipelines.
MJTelco needs you to create a schema in Google Bigtable that will allow for the historical analysis of the last 2 years of records. Each record that comes in is sent every 15 minutes, and contains a unique identifier of the device and a data record. The most common query is for all the data for a given device for a given day.
Which schema should you use?

A. Rowkey: date#device_id Column data: data_point

B. Rowkey: date Column data: device_id, data_point

C. Rowkey: device_id Column data: date, data_point

D. Rowkey: data_point Column data: device_id, date

E. Rowkey: date#data_point Column data: device_id

Show Suggested Answer

by jvg637 at *March 15, 2020, 1:42 p.m.*

Comments

Type your comment...

Submit

itche_scratche Highly Voted 5 years ago

None, rowkey should be Device_Id+Date(reverse)

upvoted 95 times

Jlozano 3 years, 4 months ago

A - "Date#Device_Id" is not the same that "Timestamp#Device_Id". If you want to query historical data, rowkey as "2021-12-09#12345device" is optimal design. Nevertheless, "2021-12-09:09:10:47:2000#12345device" isn't it. Each record has a date (2021-12-09) and unique devide id (12345, 12346, 12347...).

upvoted 23 times

Rajuuu 4 years, 9 months ago

A is a better option then other ..though not perfect as you mentioned.

upvoted 6 times

sraakesh95 3 years, 3 months ago

Totally agree if we have to avoid hotspotting! , but, incase we need to choose one of the options below, would you be going for A?

upvoted 1 times

sumanshu 3 years, 10 months ago

For READ operation it's is correct. i.e. Date#Device (so that data read from single node) -
For write operation it should be DeviceID#Date (so that data write via multiple nodes)

upvoted 4 times

Load full discussion...

jvg637 Highly Voted 5 years, 1 month ago

think is A, since “The most common query is for all the data for a given device for a given day”, rowkey should have info for both devcie and date.

upvoted 19 times

michaelkhan3 3 years, 7 months ago

Google specifically mentions that it's a bad idea to use a timestamp at the start of a rowkey
https://cloud.google.com/bigtable/docs/schema-design#row-keys-avoid
The answer really should be Device_id#Timestamp but with the answers we were given you would be better off leaving the timestamp out all together

upvoted 13 times

Whoswho 2 years, 4 months ago

I remember seeing it as well. the answer should be A. (reversed)

upvoted 2 times

wan2three 2 years, 5 months ago

but it didnt say cant use date, date and timestamp are different

upvoted 5 times

FP77 1 year, 8 months ago

The date is even worse than timestamp for the problem of hot-spotting

upvoted 1 times

Ronn27 Most Recent 4 months ago

Selected Answer: A

Its very confusing but what I found is timebucket concept and day can be used instead of timestamp.

https://cloud.google.com/bigtable/docs/schema-design-time-series#time-buckets

upvoted 2 times

cloud_rider 5 months, 1 week ago

Selected Answer: C

The correct option should be device_id#Date as it will distribute the load while writing and also be performant while reading. C is the second best option in my understanding as device Id will ensure that data sent by all the devices on a day is distributed between nodes and will not create hotspot.

distributed between nodes and will not create hotspot.

upvoted 3 times

SamuelTsch 6 months, 2 weeks ago

Selected Answer: A

I would go to date#device_id. However, i don't find this combination. A should be then chosen.

upvoted 3 times

cmira123 8 months, 3 weeks ago

A-https://cloud.google.com/bigtable/docs/schema-design-time-series?hl=es-419#use_tall_and_narrow_tables

upvoted 1 times

Lenifia 10 months ago

Selected Answer: A

showed up in my exam. picked A. passed the exam. still not sure it's correct though

upvoted 2 times

39405bb 11 months, 3 weeks ago

A. Rowkey: date#device_id Column data: data_point

Explanation:

Optimized for Most Common Query: The most common query is for all data for a given device on a given day. This schema directly matches the query pattern by including both date and device_id in the row key. This enables efficient retrieval of the required data using a single row key prefix scan.
Scalability: As the number of devices and data points increases, this schema distributes the data evenly across nodes in the Bigtable cluster, avoiding hotspots and ensuring scalability.
Data Organization: By storing data points as column values within each row, you can easily add new data points or timestamps without modifying the table structure.

upvoted 1 times

mark1223jkh 11 months, 3 weeks ago

Answer C:
https://cloud.google.com/bigtable/docs/schema-design#time-based:~:text=Don%27t%20use%20a%20timestamp%20by%20itself%20or%20at%20the%20beginning%20of%20a%20row%20key%2C

upvoted 1 times

0725f1f 1 year, 2 months ago

Selected Answer: C

c without any doubt

upvoted 2 times

philli1011 1 year, 3 months ago

The right answer should be Reverse A, but since we don't have that, the best answer is C.

upvoted 1 times

gise 1 year, 3 months ago

Selected Answer: C

C. This schema is best suited for historical analysis of device data over time when the most common query is to retrieve all data for a **specific device** on a **given day**.

* **Row Key as `device_id`:** This allows for efficient retrieval of all data points related to a particular device in a single operation. Bigtable sorts data lexicographically by row key, so all data for a single device will be stored together.

* **Column with `date` and `data_point`:**
- Using `date` as a column name or part of the column qualifier allows you to quickly filter and retrieve data for specific date ranges.
- Storing `data_point` as the column value provides the actual data associated with each timestamp.

Example:

With this schema, a query to get all data for `device_12345` on `2023-12-20` would efficiently target the specific row key `device_12345` and fetch the relevant columns (with dates around `2023-12-20`).

upvoted 3 times

JonFrow 1 year, 4 months ago

C - the answer should the right answer.
Key is "all the data for a given device for a given day"
as in, Device first, and all the data + data points after.
This has nothing to do with Date-based search.

upvoted 1 times

rocky48 1 year, 5 months ago

Selected Answer: A

A - Key should be less granular item first to more granular item, there are more devices than date key (every 15 min)

upvoted 1 times

imran79 1 year, 6 months ago

the closest match to this in the provided options is:

C. Rowkey: device_id Column data: date, data_point

Thus, option C would be the best choice from the given option

upvoted 1 times

kenwilliams 1 year, 11 months ago

Selected Answer: A

It all comes down to the most common query

upvoted 3 times

FP77 1 year, 8 months ago

Exactly
"all the data for a given device for a given day"
That's why the answer is C. You start by selecting the device and then the date. This solution is not prone to hot-spotting, yours is.

upvoted 1 times

PolyMoe 2 years, 3 months ago

Selected Answer: A

A. Rowkey: date#device_id Column data: data_point This schema would allow querying all data for a given device for a given day by looking up the row key, which would be the date followed by the device_id. This would be the most efficient way to access the data as it would be stored in sorted order by date and device_id.

upvoted 2 times

Load full discussion...