

[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 3 DISCUSSION

Actual exam question from Google's Professional Data Engineer

Question #: 3

Topic #: 1

[\[All Professional Data Engineer Questions\]](#)

You designed a database for patient records as a pilot project to cover a few hundred patients in three clinics. Your design used a single database table to represent all patients and their visits, and you used self-joins to generate reports. The server resource utilization was at 50%. Since then, the scope of the project has expanded. The database must now store 100 times more patient records. You can no longer run the reports, because they either take too long or they encounter errors with insufficient compute resources. How should you adjust the database design?




- A. Add capacity (memory and disk space) to the database server by the order of 200.
- B. Shard the tables into smaller ones based on date ranges, and only generate reports with prespecified date ranges.
- C. Normalize the master patient-record table into the patient table and the visits table, and create other necessary tables to avoid self-join.
- D. Partition the table into smaller tables, with one for each clinic. Run queries against the smaller table pairs, and use unions for consolidated reports.

[Show Suggested Answer](#)

by [deleted] at *March 15, 2020, 8:14 a.m.*

Comments

Type your comment...

  **MaxNRG** Highly Voted  3 years, 5 months ago

C is correct because this option provides the least amount of inconvenience over using pre-specified date ranges or one table per clinic while also increasing performance due to avoiding self-joins.

A is not correct because adding additional compute resources is not a recommended way to resolve database schema problems.

B is not correct because this will reduce the functionality of the database and make running reports more difficult.

D is not correct because this will likely increase the number of tables so much that it will be more difficult to generate reports vs. the correct option.

<https://cloud.google.com/bigquery/docs/best-practices-performance-patterns>

<https://cloud.google.com/bigquery/docs/reference/standard-sql/query-syntax#explicit-alias-visibility>

   upvoted 7 times



  **gord_nat** 4 months, 1 week ago

Why are we assuming the database in question is BigQuery? There are several other RDBMS options in GCP. ..

Also, why was the original db pushed to prod without being normalized first? Typically, the normalized db is released to prod. When the data set becomes larger, you would add partitioning.

The scenario being presented is unrealistic.

   upvoted 1 times

  **balserson99** Highly Voted  4 years, 3 months ago

A is incorrect because adding space won't solve the problem of query performance.




B is incorrect because there is nothing related to the report generation which is specified and sharding tables on date ranges is not a good option as it will create many tables.

C is CORRECT because the statement says "the scope of the project has expanded. The database must now store 100 times more patient records". As the data increases there would be difficulty in managing the tables and querying it. Hence creating different table is correct as per the need.

D is Incorrect as it Partitions on each clinic. We have to adjust the database design so that it performs optimally when generating reports.

Also nothing is specified for generation of reports in the required statement.



   upvoted 6 times

  **Ahamada** Most Recent  2 months, 1 week ago

Selected Answer: C

answer is C, the problem here is the self-join (avoid self-join if possible) on a Denormalized table. So the solution is to Normalize

   upvoted 1 times

  **cqrm3n** 3 months, 3 weeks ago

Selected Answer: C

Normalizing the database into separate Patients and Visits tables, along with creating other necessary tables, is the best solution for handling the increased data size while ensuring efficient query performance and maintainability. This approach addresses the root problem instead of applying temporary fixes.

   upvoted 1 times

  **SamuelTsch** 6 months, 2 weeks ago

Selected Answer: C

C is the most suitable solution for this situation. It provides a better way for scalability and monitoring. B has a constraint on predefined date range, which is usually not suitable for reporting.

   upvoted 1 times

  **rocky48** 1 year, 6 months ago

Selected Answer: C

Normalization is a technique used to organize data in a relational database to reduce data redundancy and improve data integrity. Breaking the patient records into separate tables (patient and visits) and eliminating self-joins will make the database more scalable and improve query performance. It also helps maintain data integrity and makes it easier to manage large datasets efficiently.

Options A, B, and D may provide some benefits in specific cases, but for a scenario where the project scope has expanded significantly and there are performance issues with self-joins, normalization (Option C) is the most robust and scalable solution.

   upvoted 4 times

  **rtcpost** 1 year, 6 months ago


Selected Answer: C

Normalization is a technique used to organize data in a relational database to reduce data redundancy and improve data integrity. Breaking the patient records into separate tables (patient and visits) and eliminating self-joins will make the database more scalable and improve query performance. It also helps maintain data integrity and makes it easier to manage

large datasets efficiently.

Options A, B, and D may provide some benefits in specific cases, but for a scenario where the project scope has expanded significantly and there are performance issues with self-joins, normalization (Option C) is the most robust and scalable solution.

   upvoted 3 times

  **vaga1** 1 year, 11 months ago

Selected Answer: C

"100 times more patient records" immediately brings to create a patient dimensional table to save space on disk if a generical relational database is mentioned.

   upvoted 1 times

  **maurilio_cardoso_multiedro** 2 years, 1 month ago

C - <https://cloud.google.com/bigquery/docs/best-practices-performance-patterns>

   upvoted 1 times

  **bha11111** 2 years, 1 month ago

Selected Answer: C

C- This is correct have verified from different sources

   upvoted 1 times

  **Morock** 2 years, 2 months ago

Selected Answer: C

Should be C. Basic ER design...

   upvoted 1 times

  **GCPpro** 2 years, 3 months ago

c - is the correct one.

   upvoted 1 times

  **testoneAZ** 2 years, 3 months ago

C should be the correct answer

   upvoted 1 times

  **Brillianttyagi** 2 years, 4 months ago

Selected Answer: C

C- Is the correct answer!

   upvoted 1 times

  **Arkon88** 3 years, 2 months ago

Selected Answer: C

C - based on Google documentation, self-join is an anti-pattern:

<https://cloud.google.com/bigquery/docs/best-practices-performance-patterns>

   upvoted 2 times

  **ch1nczyk** 3 years, 2 months ago

Selected Answer: C

Correct

   upvoted 1 times

  **samdhimal** 3 years, 3 months ago

correct answer -> Normalize the master patient-record table into the patient table and the visits table, and create other necessary tables to avoid self-join.

Avoid self-join at all cost because that's what google says.

Reference:

<https://cloud.google.com/bigquery/docs/best-practices-performance-patterns>

   upvoted 3 times

  **samdhimal** 2 years, 3 months ago

Normalizing the database design will help to minimize data redundancy and improve the efficiency of the queries. By separating the patient and visit information into separate tables, the database will be able to handle the increased number of records and generate reports more efficiently, because the self-joins will no longer be required.

Option A is not a good solution because adding more capacity to the server will not address the underlying problem of the database design, and it may not be sufficient to handle the increased data volume.

Option B is not a good solution because it limits the flexibility of the queries and reports, and it may not be sufficient to

Option B is not a good solution because it limits the flexibility of the queries and reports, and it may not be sufficient to handle the increased data volume.

Option D is not a good solution because partitioning the table into smaller tables may lead to data redundancy and it may not be sufficient to handle the increased data volume.

   upvoted 3 times

[Load full discussion...](#)



Platform

> [Home](#)

> [Examtopics PRO](#)

> [All Exams](#)

> [Training Courses](#)



© 2024 ExamTopics