DP-420 Q&As

Designing and Implementing Cloud-Native Applications Using Microsoft Azure Cosmos DB

Pass Microsoft DP-420 Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.pass4itsure.com/dp-420.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Microsoft Official Exam Center

- **Instant Download** After Purchase
- **100% Money Back** Guarantee
- **365 Days** Free Update
- **800,000+** Satisfied Customers

QUESTION 1

You are building an application that will store data in an Azure Cosmos DB for NoSQL account. The account uses the session default consistency level. The account is used by five other applications. The account has a single read-write

region and 10 additional read regions.

Approximately 20 percent of the items stored in the account are updated hourly.

Several users will access the new application from multiple devices.

You need to ensure that the users see the same item values consistently when they browse from the different devices. The solution must not affect the other applications.

Which two actions should you perform? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. Use implicit session management when performing read requests.

B. Provide a stored session token when performing read requests.

C. Associate a session token to the user account.

D. Set the default consistency level to eventual.

E. Associate a session token to the device.

Correct Answer: BC

QUESTION 2

You have operational data in an Azure Cosmos OB for NoSQL database.

Database users report that the performance of the database degrades significantly when a business analytics team runs large Apache Spark-based queries against the database.

You need 10 reduce the impact that running the Spark-based queries has on the database users.

What should you implement?

A. Azure Synapse Link

B. a default consistency level of Consistent Prefix

C. a default consistency level of Strong

D. the Spark connector

Correct Answer: A

QUESTION 3

After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.

You have a container named container1 in an Azure Cosmos DB Core (SQL) API account.

You need to make the contents of container1 available as reference data for an Azure Stream Analytics job.

Solution: You create an Azure function that uses Azure Cosmos DB Core (SQL) API change feed as a trigger and Azure event hub as the output.

Does this meet the goal?

A. Yes

B. No

Correct Answer: B

The Azure Cosmos DB change feed is a mechanism to get a continuous and incremental feed of records from an Azure Cosmos container as those records are being created or modified. Change feed support works by listening to container for any changes. It then outputs the sorted list of documents that were changed in the order in which they were modified.

Reference: https://docs.microsoft.com/en-us/azure/cosmos-db/sql/changefeed-ecommerce-solution

QUESTION 4

You have a database in an Azure Cosmos DB Core (SQL) API account. The database is backed up every two hours.

You need to implement a solution that supports point-in-time restore.

What should you do first?

A. Enable Continuous Backup for the account.

B. Configure the Backup and Restore settings for the account.

C. Create a new account that has a periodic backup policy.

D. Configure the Point In Time Restore settings for the account.

Correct Answer: A

When creating a new Azure Cosmos DB account, in the Backup policy tab, choose continuous mode to enable the point in time restore functionality for the new account. With the point-in-time restore, data is restored to a new account, currently you can\\'t restore to an existing account.

Reference: https://docs.microsoft.com/en-us/azure/cosmos-db/provision-account-continuous-backup

QUESTION 5

You have a database named db1 in an Azure Cosmos DB for NoSQL

You are designing an application that will use dbl.

In db1, you are creating a new container named coll1 that will store in coll1.

The following is a sample of a document that will be stored in coll1.

```
{
    "customerId" : "bba6fe24-6d97-4935-8d58-36baa4b8a0e1",
    "orderId" : "9d7816e6-f401-42ba-ad05-0e03de35c0b8",
    "orderDate" : "2021-09-29",
    "orderDetails" : []
}
```

The application will have the following characteristics:

1.

New orders will be created frequently by different customers.

2.

Customers will often view their past order history.

You need to select the partition key value for coll1 to support the application. The solution must minimize costs. To what should you set the partition key?

A. id

B. customerId

C. orderDate

D. orderId

Correct Answer: B

Based on the characteristics of the application and the provided document structure, the most suitable partition key value for coll1 in the given scenario would be the customerId, Option B. The application frequently creates new orders by different customers and customers often view their past order history. Using customerId as the partition key would ensure that all orders associated with a particular customer are stored in the same partition. This enables efficient querying of past order history for a specific customer and reduces cross-partition queries, resulting in lower costs and improved performance. a partition key is a JSON property (or path) within your documents that is used by Azure Cosmos DB to distribute data among multiple partitions3. A partition key should have a high cardinality, which means it should have many distinct values, such as hundreds or thousands1. A partition key should also align with the most common query patterns of your application, so that you can efficiently retrieve data by using the partition key value1. Based on these criteria, one possible partition key that you could use for coll1 is B:customerId.

This partition key has the following advantages: It has a high cardinality, as each customer will have a unique ID3. It aligns with the query patterns of the application, as customers will often view their past order history3. It minimizes costs, as it reduces the number of cross-partition queries and optimizes the storage and throughput utilization1. This partition key also has some limitations, such as: It may not be optimal for scenarios where orders need to be queried

independently from customers or aggregated by date or other criteria3. It may result in hot partitions or throttling if some customers create orders more frequently than others or have more data than others1. It may not support transactions across multiple customers, as transactions are scoped to a single logical partition2. Depending on your specific use case and requirements, you may need to adjust this partition key or choose a different one. For example, you could use a synthetic partition key that concatenates multiple properties of an item2, or you could use a partition key with a random or pre-calculated suffix to distribute the workload more evenly2.