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.passapply.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 have an Azure Cosmos DB for NoSQL account configured for global distribution across four regions.

At connection time, the SQL SDK needs to identify the optimal endpoint for reading and writing.

Which two factors can influence the SDK? Each correct answer presents a complete solution.

NOTE: Each correct selection is worth one point.

A. the consistency level in RequestOptions

B. regional latency

C. the default consistency level

D. the PreferredLocations configuration

E. a region being available

Correct Answer: DE

Explanation:

Connecting to a preferred region using the API for NoSQL

The SDKs accept an optional parameter PreferredLocations that is an ordered list of Azure regions.

The SDK will automatically send all writes to the current write region. All reads will be sent to the first available region in the preferred locations list. If the request fails, the client will fail down the list to the next region.

The SDK will only attempt to read from the regions specified in preferred locations. So, for example, if the Azure Cosmos DB account is available in four regions, but the client only specifies two read(non-write) regions within the

PreferredLocations, then no reads will be served out of the read region that is not specified in PreferredLocations. If the read regions specified in the PreferredLocations list are not available, reads will be served out of write region.

Reference: https://learn.microsoft.com/en-us/azure/cosmos-db/nosql/tutorial-global-distribution

QUESTION 2

You have a container in an Azure Cosmos DB for NoSQL account that stores data about orders. The following is a sample of an order document.

```
{
    "orderId" : "d4a9179b-5ead-43a3-b851-add9a71ac4b6",
    "customerId" : "f6e39103-bdc7-4346-9cfb-45daa4b2becf",
    "orderDate" : "2021-09-29",
    "orderItems" : [...],
    "total" : 12345
}
```

Documents are up to 2 KB.

You plan to receive one million orders daily.

Customers will frequently view then past order history.

You are the evaluating whether to use orderDate as the partition key.

What are two effects of using orderDate as the partition key? Each correct answer presents a complete solution.

NOTE: Each correct selection is worth one point.

A. You will exceed the maximum number of partition key values.

B. Queries will run cross-partition.

C. You will exceed the maximum storage per partition.

D. There will always be a hot partition.

Correct Answer: BD

QUESTION 3

You have a database in an Azure Cosmos DB Core (SQL) API account.

You need to create an Azure function that will access the database to retrieve records based on a variable named accountnumber. The solution must protect against SQL injection attacks.

How should you define the command statement in the function?

A. cmd = "SELECT * FROM Persons pWHERE p.accountnumber = \\'accountnumber\\'"

B. cmd = "SELECT * FROM Persons pWHERE p.accountnumber = LIKE @accountnumber"

C. cmd = "SELECT * FROM Persons pWHERE p.accountnumber = @accountnumber"

D. cmd = "SELECT * FROM Persons pWHERE p.accountnumber = \\'" + accountnumber + "\\'"

Correct Answer: C

Azure Cosmos DB supports queries with parameters expressed by the familiar @ notation. Parameterized SQL provides robust handling and escaping of user input, and prevents accidental exposure of data through SQL injection.

For example, you can write a query that takes lastName and address.state as parameters, and execute it for various values of lastName and address.state based on user input.

SELECT * FROM Families f WHERE f.lastName = @lastName AND f.address.state = @addressState

Reference: https://docs.microsoft.com/en-us/azure/cosmos-db/sql/sql-query-parameterized-queries

QUESTION 4

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 5

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 Synapse pipeline that uses Azure Cosmos DB Core (SQL) API as the input and Azure Blob Storage as the output.

Does this meet the goal?

A. Yes

B. No

Correct Answer: B

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

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