



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 are troubleshooting the current issues caused by the application updates.

Which action can address the application updates issue without affecting the functionality of the application?

- A. Enable time to live for the con-product container.
- B. Set the default consistency level of account1 to strong.
- C. Set the default consistency level of account1 to bounded staleness.
- D. Add a custom indexing policy to the con-product container.

Correct Answer: C

Bounded staleness is frequently chosen by globally distributed applications that expect low write latencies but require total global order guarantee. Bounded staleness is great for applications featuring group collaboration and sharing, stock ticker, publish-subscribe/queueing etc.

Scenario: Application updates in con-product frequently cause HTTP status code 429 "Too many requests". You discover that the 429 status code relates to excessive request unit (RU) consumption during the updates.

Reference: <https://docs.microsoft.com/en-us/azure/cosmos-db/consistency-levels>

QUESTION 2

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while

others might not have a correct solution.

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 for NoSQL 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 to copy data to another Azure Cosmos DB for NoSQL container.

Does this meet the goal?

- A. Yes
- B. No

Correct Answer: B

QUESTION 3

You plan to store order data in Azure Cosmos DB for NoSQL account. The data contains information about orders and



their associated items.

You need to develop a model that supports order read operations. The solution must minimize the number of requests.

- A. Create a single database that contains one container. Store orders and order items in separate documents in the container.
- B. Create a single database that contains one container. Create a separate document for each order and embed the order items into the order documents.
- C. Create a database for orders and a database for order items.
- D. Create a single database that contains a container for order and a container for order items.

Correct Answer: B

Azure Cosmos DB is a multi-model database that supports various data models, such as documents, key-value, graph, and column-family³. The core content-model of Cosmos DB's database engine is based on atom-record-sequence (ARS), which allows it to store and query different types of data in a flexible and efficient way³. To develop a model that supports order read operations and minimizes the number of requests, you should consider the following factors:

1.

The size and shape of your data

2.

The frequency and complexity of your queries

3.

The latency and throughput requirements of your application

4.

The trade-offs between storage efficiency and query performance

Based on these factors, one possible model that you could implement is B. Create a single database that contains one container. Create a separate document for each order and embed the order items into the order documents.

This model has the following advantages:

1.

It stores orders and order items as self-contained documents that can be easily retrieved by order ID¹.

2.

It avoids storing redundant data or creating additional containers for order items¹.

3.

It allows you to view the order history of a customer with simple queries¹.

4.

It leverages the benefits of embedding data, such as reducing the number of requests, improving query performance,



and simplifying data consistency².

This model also has some limitations, such as: It may not be suitable for some order items that have data that is greater than 2 KB, as it could exceed the maximum document size limit of 2 MB². It may not be optimal for scenarios where order items need to be queried independently from orders or aggregated by other criteria². It may not support transactions across multiple orders or customers, as transactions are scoped to a single logical partition². Depending on your specific use case and requirements, you may need to adjust this model or choose a different one. For example, you could use a hybrid data model that combines embedding and referencing data², or you could use a graph data model that expresses entities and relationships as vertices and edges.

QUESTION 4

You have a database in an Azure Cosmos DB for NoSQL account.

The database contains a container named container1. The indexing mode container1 is set to none.

You configure Azure Cognitive Search to extract data from container1 and make the data searchable.

You discover that the Cognitive Search index is missing all the data from the Azure Cosmos DB index.

What should you do to resolve the issue?

- A. Modify The index attributes in Cognitive Search to searchable.
- B. Modify the index attributes in Cognitive Search to Retrievable.
- C. Change the indexing mode of container 1 to consistent-
- D. Modify the indexing policy of container 1 to exclude the / * path

Correct Answer: C

QUESTION 5

HOTSPOT

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

The following is a sample of a document in container1.

```
{  
  "studentId": "631282",  
  "firstName": "James",  
  "lastName": "Smith",  
  "enrollmentYear": 1990,  
  "isActivelyEnrolled": true,  
  "address": {
```



```
"street": "",  
"city": "",  
"stateProvince": "",  
"postal": "",  
}  
}
```

The container1 container has the following indexing policy.

```
{  
  "indexingMode": "consistent",  
  "includePaths": [  
    {  
      "path": "/*"  
    },  
    {  
      "path": "/address/city/?"  
    }  
  ],  
  "excludePaths": [  
    {  
      "path": "/address/*"  
    },  
    {  
      "path": "/firstName/?"  
    }  
  ]  
}
```

For each of the following statements, select Yes if the statement is true. Otherwise, select No.

NOTE: Each correct selection is worth one point.

Hot Area:



Answer Area

Statements	Yes	No
The /isActivelyEnrolled property is included in the index	<input type="radio"/>	<input type="radio"/>
The /fisrtname property is included in the index	<input type="radio"/>	<input type="radio"/>
The /address/city property is included in the index	<input type="radio"/>	<input type="radio"/>

Correct Answer:

Answer Area

Statements	Yes	No
The /isActivelyEnrolled property is included in the index	<input checked="" type="radio"/>	<input type="radio"/>
The /fisrtname property is included in the index	<input type="radio"/>	<input checked="" type="radio"/>
The /address/city property is included in the index	<input checked="" type="radio"/>	<input type="radio"/>

Box 1: Yes

"path": "/" is in includePaths.

Include the root path to selectively exclude paths that don't need to be indexed. This is the recommended approach as it lets Azure Cosmos DB proactively index any new property that may be added to your model.

Box 2: No

"path": "/firstName/?" is in excludePaths.

Box 3: Yes

"path": "/address/city/?" is in includePaths

Reference:

<https://docs.microsoft.com/en-us/azure/cosmos-db/index-policy>

[Latest DP-420 Dumps](#)

[DP-420 PDF Dumps](#)

[DP-420 Braindumps](#)