DP-420$^{Q\&As}$

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

Pass Microsoft DP-420 Exam with 100% Guarantee

QUESTION 1

You have a database in an Azure Cosmos DB for NoSQL account that is configured for multi-region writes.

You need to use the Azure Cosmos DB SDK to implement the conflict resolution policy for a container. The solution must ensure that any conflict sent to the conflict feed.

Solution:

1.

You set ConfilictResolutionMode to Custom.

2.

You Set ResolutionProcedures to a custom stored procedure.

3.

You configure the custom stored procedure to use the conflictingItems parameter to resolve conflict.

Does this meet the goal?

A. Yes

B. No

Correct Answer: A

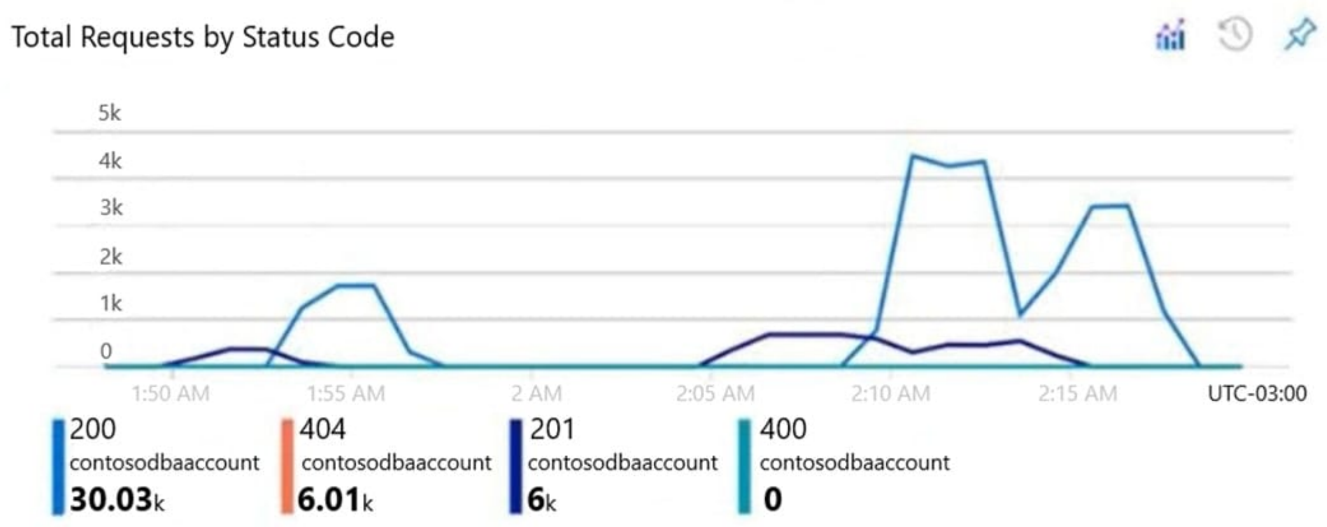
Setting ConflictResolutionMode to Custom and configuring a custom stored procedure with the "conflictingItems" parameter will allow you to implement a custom conflict resolution policy. This will ensure that any conflicts are sent to the conflict feed for resolution.

QUESTION 2

HOTSPOT

You have an Azure Cosmos DB Core (SQL) API account used by an application named App1.

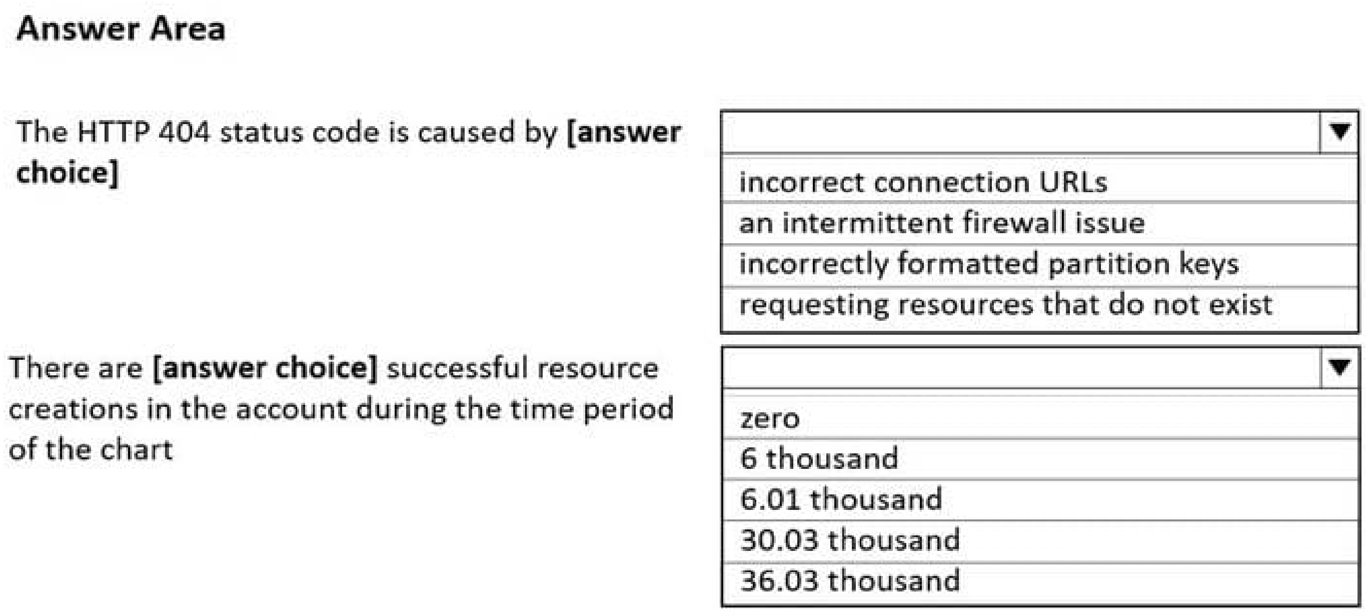
You open the Insights pane for the account and see the following chart.



Use the drop-down menus to select the answer choice that answers each question based on the information presented in the graphic.

NOTE: Each correct selection is worth one point.

Hot Area:



Answer Area

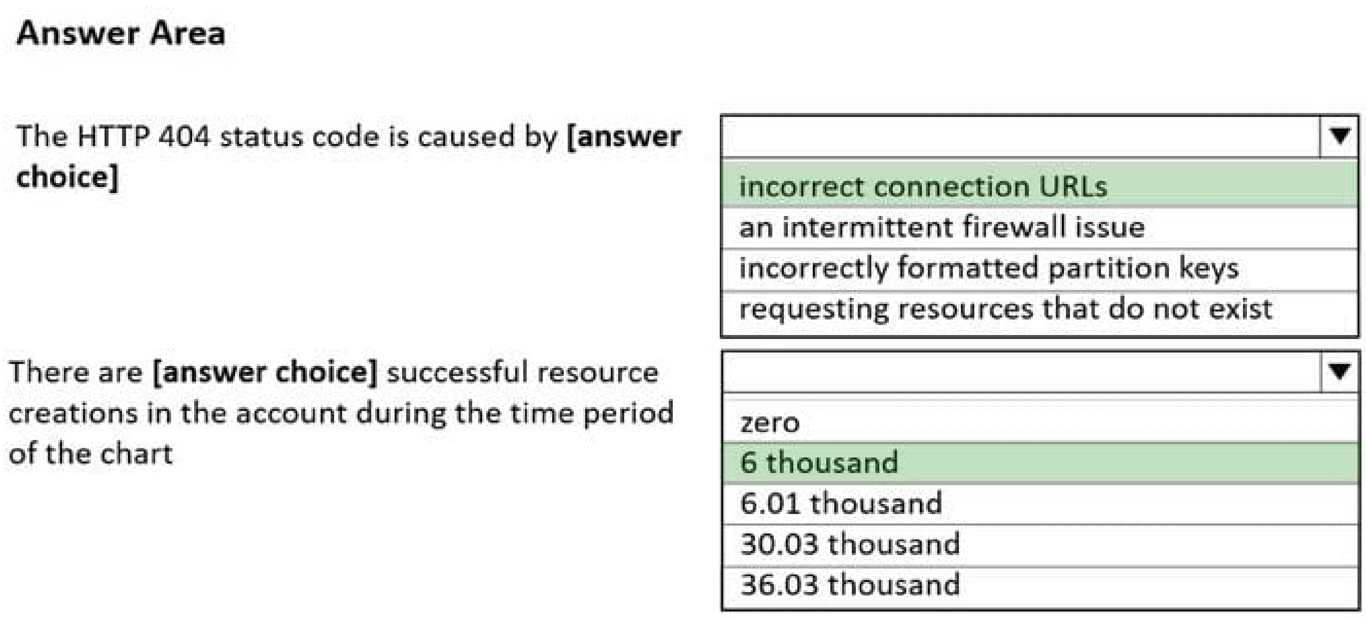
The HTTP 404 status code is caused by **[answer choice]**

- incorrect connection URLs
- an intermittent firewall issue
- incorrectly formatted partition keys
- requesting resources that do not exist

There are **[answer choice]** successful resource creations in the account during the time period of the chart

- zero
- 6 thousand
- 6.01 thousand
- 30.03 thousand
- 36.03 thousand

Correct Answer:



Answer Area

The HTTP 404 status code is caused by **[answer choice]**

- [✓ incorrect connection URLs]
- an intermittent firewall issue
- incorrectly formatted partition keys
- requesting resources that do not exist

There are **[answer choice]** successful resource creations in the account during the time period of the chart

- zero
- [✓ 6 thousand]
- 6.01 thousand
- 30.03 thousand
- 36.03 thousand

Box 1: incorrect connection URLs

400 Bad Request: Returned when there is an error in the request URI, headers, or body. The response body will contain an error message explaining what the specific problem is.

The HyperText Transfer Protocol (HTTP) 400 Bad Request response status code indicates that the server cannot or will not process the request due to something that is perceived to be a client error (for example, malformed request syntax,

invalid request message framing, or deceptive request routing).

Box 2: 6 thousand

201 Created: Success on PUT or POST. Object created or updated successfully.

Note:

200 OK: Success on GET, PUT, or POST. Returned for a successful response.

404 Not Found: Returned when a resource does not exist on the server. If you are managing or querying an index, check the syntax and verify the index name is specified correctly.

Reference:

https://docs.microsoft.com/en-us/rest/api/searchservice/http-status-codes

QUESTION 3

You have an Azure Cosmos DB Core (SQL) API account that is used by 10 web apps.

You need to analyze the data stored in the account by using Apache Spark to create machine learning models. The solution must NOT affect the performance of the web apps.

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

NOTE: Each correct selection is worth one point.

A. In an Apache Spark pool in Azure Synapse, create a table that uses cosmos.olapas the data source.

B. Create a private endpoint connection to the account.

C. In an Azure Synapse Analytics serverless SQL pool, create a view that uses OPENROWSETand the CosmosDB provider.

D. Enable Azure Synapse Link for the account and Analytical store on the container.

E. In an Apache Spark pool in Azure Synapse, create a table that uses cosmos.oltpas the data source.

Correct Answer: AD

Explore analytical store with Apache Spark

1.

Navigate to the Data hub.

2.

Select the Linked tab (1), expand the Azure Cosmos DB group (if you don\\'t see this, select the Refresh button above), then expand the WoodgroveCosmosDb account (2). Right-click on the transactions container (3), select New notebook (4), then select Load to DataFrame (5).

3.

In the generated code within Cell 1 (3), notice that the spark.read format is set to cosmos.olap. This instructs Synapse Link to use the container\\'s analytical store. If we wanted to connect to the transactional store, like to read from the change feed or write to the container, we\\'d use cosmos.oltp instead.

Reference: https://github.com/microsoft/MCW-Cosmos-DB-Real-Time-Advanced-Analytics/blob/main/Hands-on%20lab/HOL%20step-by%20step%20-%20Cosmos%20DB%20real-time%20advanced%20analytics.md

QUESTION 4

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

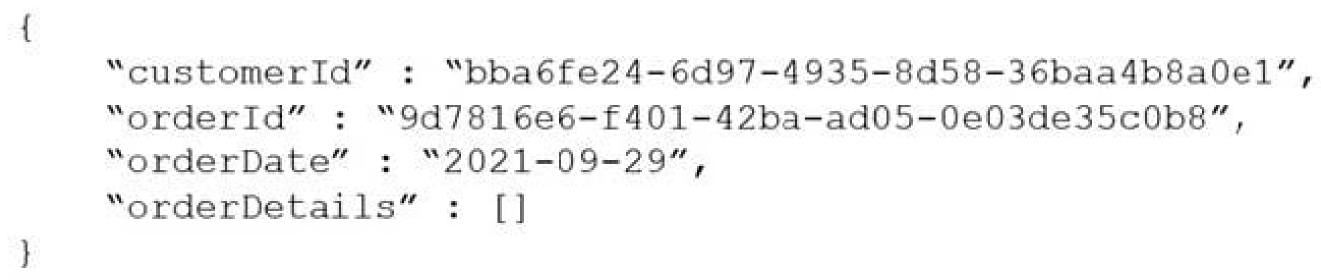
QUESTION 5

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.