MICROSOFT DP-420

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

Get Instant Access to Vital Exam Acing Materials | Study Guide | Sample Questions | Practice Test

DP-420

Microsoft Certified - Azure Cosmos DB Developer Specialty

40-60 Questions Exam – 700 / 1000 Cut Score – Duration of 120 minutes

EDUSUM
#1 Online Certification Guide

Table of Contents:

Discover More about the DP-420 Certification

Are you interested in passing the Microsoft DP-420 exam? First discover, who benefits from the DP-420 certification. The DP-420 is suitable for a candidate if he wants to learn about Microsoft Azure. Passing the DP-420 exam earns you the Microsoft Certified - Azure Cosmos DB Developer Specialty title.

While preparing for the DP-420 exam, many candidates struggle to get the necessary materials. But do not worry; your struggling days are over. The DP-420 PDF contains some of the most valuable preparation tips and the details and instant access to useful **DP-420 study materials just at one click**.

Microsoft DP-420 Designing and Implementing Cloud-Native Applications Using Microsoft Azure Cosmos DB Certification Details:

Exam Name	Microsoft Certified - Azure Cosmos DB Developer Specialty
Exam Code	DP-420
Exam Price	$165 (USD)
Duration	120 mins
Number of Questions	40-60
Passing Score	700 / 1000
Books / Training	**Course DP-420T00: Designing and Implementing Cloud-Native Applications Using Microsoft Azure Cosmos DB**
Schedule Exam	**Pearson VUE**
Sample Questions	**Designing and Implementing Cloud-Native Applications Using Microsoft Azure Cosmos DB Sample Questions**
Practice Exam	**Microsoft DP-420 Certification Practice Exam**

DP-420 Syllabus:

Topic	**Details**
Design and Implement Data Models (35-40%)	
Design and implement a non-relational data model for Azure Cosmos DB Core API	- Develop a design by storing multiple entity types in the same container - Develop a design by storing multiple related entities in the same document - Develop a model that denormalizes data across documents - Develop a design by referencing between documents - Identify primary and unique keys - Identify data and associated access patterns - Specify a default TTL on a container for a transactional store
Design a data partitioning strategy for Azure Cosmos DB Core API	- Choose a partitioning strategy based on a specific workload - Choose a partition key - Plan for transactions when choosing a partition key - Evaluate the cost of using a cross-partition query - Calculate and evaluate data distribution based on partition key selection - Calculate and evaluate throughput distribution based on partition key selection - Construct and implement a synthetic partition key - Design partitioning for workloads that require multiple partition keys
Plan and implement sizing and scaling for a database created with Azure Cosmos DB	- Evaluate the throughput and data storage requirements for a specific workload - Choose between serverless and provisioned models - Choose when to use database-level provisioned throughput - Design for granular scale units and resource governance - Evaluate the cost of the global distribution of data

Topic	**Details**
	- Configure throughput for Azure Cosmos DB by using the Azure portal
Implement client connectivity options in the Azure Cosmos DB SDK	- Choose a connectivity mode (gateway versus direct) - Implement a connectivity mode - Create a connection to a database - Enable offline development by using the Azure Cosmos DB emulator - Handle connection errors - Implement a singleton for the client - Specify a region for global distribution - Configure client-side threading and parallelism options - Enable SDK logging
Implement data access by using the Azure Cosmos DB SQL language	- Implement queries that use arrays, nested objects, aggregation, and ordering - Implement a correlated subquery - Implement queries that use array and type-checking functions - Implement queries that use mathematical, string, and date functions - Implement queries based on variable data
Implement data access by using SQL API SDKs	- Choose when to use a point operation versus a query operation - Implement a point operation that creates, updates, and deletes documents - Implement an update by using a patch operation - Manage multi-document transactions using SDK Transactional Batch - Perform a multi-document load using Bulk Support in the SDK - Implement optimistic concurrency control using ETags - Implement session consistency by using session tokens

Topic	Details
	- Implement a query operation that includes pagination - Implement a query operation by using a continuation token - Handle transient errors and 429s - Specify TTL for a document - Retrieve and use query metrics
Implement server-side programming in Azure Cosmos DB Core API by using JavaScript	- Write, deploy, and call a stored procedure - Design stored procedures to work with multiple items transactionally - Implement and call triggers - Implement a user-defined function
Design and Implement Data Distribution (5-10%)	
Design and implement a replication strategy for Azure Cosmos DB	- Choose when to distribute data - Define automatic failover policies for regional failure for Azure Cosmos DB Core API - Perform manual failovers to move single master write regions - Choose a consistency model - Identify use cases for different consistency models - Evaluate the impact of consistency model choices on availability and associated RU cost - Evaluate the impact of consistency model choices on performance and latency - Specify application connections to replicated data
Design and implement multi-region write	- Choose when to use multi-region write - Implement multi-region write - Implement a custom conflict resolution policy for Azure Cosmos DB Core API
Integrate an Azure Cosmos DB Solution (5-10%)	
Enable Azure Cosmos DB analytical workloads	- Enable Azure Synapse Link - Choose between Azure Synapse Link and Spark Connector - Enable the analytical store on a container

Topic	Details
	- Enable a connection to an analytical store and query from Azure Synapse Spark or Azure Synapse SQL - Perform a query against the transactional store from Spark - Write data back to the transactional store from Spark
Implement solutions across services	- Integrate events with other applications by using Azure Functions and Azure Event Hubs - Denormalize data by using Change Feed and Azure Functions - Enforce referential integrity by using Change Feed and Azure Functions - Aggregate data by using Change Feed and Azure Functions, including reporting - Archive data by using Change Feed and Azure Functions - Implement Azure Cognitive Search for an Azure Cosmos DB solution
Optimize an Azure Cosmos DB Solution (15-20%)	
Optimize query performance in Azure Cosmos DB Core API	- Adjust indexes on the database - Calculate the cost of the query - Retrieve request unit cost of a point operation or query - Implement Azure Cosmos DB integrated cache
Design and implement change feeds for an Azure Cosmos DB Core API	- Develop an Azure Functions trigger to process a change feed - Consume a change feed from within an application by using the SDK - Manage the number of change feed instances by using the change feed estimator - Implement denormalization by using a change feed - Implement referential enforcement by using a change feed - Implement aggregation persistence by using a

Topic	**Details**
	change feed - Implement data archiving by using a change feed
Define and implement an indexing strategy for an Azure Cosmos DB Core API	- Choose when to use a read-heavy versus write-heavy index strategy - Choose an appropriate index type - Configure a custom indexing policy by using the Azure portal - Implement a composite index - Optimize index performance
Maintain an Azure Cosmos DB Solution (25-30%)	
Monitor and troubleshoot an Azure Cosmos DB solution	- Evaluate response status code and failure metrics - Monitor metrics for normalized throughput usage by using Azure Monitor - Monitor server-side latency metrics by using Azure Monitor - Monitor data replication in relation to latency and availability - Configure Azure Monitor alerts for Azure Cosmos DB - Implement and query Azure Cosmos DB logs - Monitor throughput across partitions - Monitor distribution of data across partitions - Monitor security by using logging and auditing
Implement backup and restore for an Azure Cosmos DB solution	- Choose between periodic and continuous backup - Configure periodic backup - Configure continuous backup and recovery - Locate a recovery point for a point-in-time recovery - Recover a database or container from a recovery point
Implement security for an Azure Cosmos DB solution	- Choose between service-managed and customer-managed encryption keys - Configure network-level access control for Azure Cosmos DB - Configure data encryption for Azure Cosmos DB

Topic	Details
	- Manage control plane access to Azure Cosmos DB by using Azure role-based access control (RBAC) - Manage data plane access to Azure Cosmos DB by using keys - Manage data plane access to Azure Cosmos DB by using Azure Active Directory - Configure Cross-Origin Resource Sharing (CORS) settings - Manage account keys by using Azure Key Vault - Implement customer-managed keys for encryption - Implement Always Encrypted
Implement data movement for an Azure Cosmos DB solution	- Choose a data movement strategy - Move data by using client SDK bulk operations - Move data by using Azure Data Factory and Azure Synapse pipelines - Move data by using a Kafka connector - Move data by using Azure Stream Analytics - Move data by using the Azure Cosmos DB Spark Connector
Implement a DevOps process for an Azure Cosmos DB solution	- Choose when to use declarative versus imperative operations - Provision and manage Azure Cosmos DB resources by using Azure Resource Manager templates (ARM templates) - Migrate between standard and autoscale throughput by using PowerShell or Azure CLI - Initiate a regional failover by using PowerShell or Azure CLI - Maintain index policies in production by using ARM templates

Broaden Your Knowledge with Microsoft DP-420 Sample Questions:

Question: 1

You have an Azure Cosmos DB Core (SQL) API account. You run the following query against a container in the account.

SELECT -

IS_NUMBER("1234") AS A,

IS_NUMBER(1234) AS B,

IS_NUMBER({prop: 1234}) AS C -

What is the output of the query?

a) [{"A": false, "B": true, "C": false}]
b) [{"A": true, "B": false, "C": true}]
c) [{"A": true, "B": true, "C": false}]
d) [{"A": true, "B": true, "C": true}]

Answer: a

Question: 2

You have a container named container1 in an Azure Cosmos DB Core (SQL) API account. Upserts of items in container1 occur every three seconds.

You have an Azure Functions app named function1 that is supposed to run whenever items are inserted or replaced in container1. You discover that function1 runs, but not on every upsert. You need to ensure that function1 processes each upsert within one second of the upsert.

Which property should you change in the Function.json file of function1?

a) checkpointInterval
b) leaseCollectionsThroughput
c) maxItemsPerInvocation
d) feedPollDelay

Answer: d

Question: 3

You have the following query.

SELECT * FROM ◌̇

WHERE c.sensor = "TEMP1"

AND c.value < 22 -

AND c.timestamp >= 1619146031231

You need to recommend a composite index strategy that will minimize the request units (RUs) consumed by the query. What should you recommend?

a) a composite index for (sensor ASC, value ASC) and a composite index for (sensor ASC, timestamp ASC)
b) a composite index for (sensor ASC, value ASC, timestamp ASC) and a composite index for (sensor DESC, value DESC, timestamp DESC)
c) a composite index for (value ASC, sensor ASC) and a composite index for (timestamp ASC, sensor ASC)
d) a composite index for (sensor ASC, value ASC, timestamp ASC)

Answer: a

Question: 4

You have an application named App1 that reads the data in an Azure Cosmos DB Core (SQL) API account. App1 runs the same read queries every minute. The default consistency level for the account is set to eventual. You discover that every query consumes request units (RUs) instead of using the cache.

You verify the IntegratedCacheiteItemHitRate metric and the IntegratedCacheQueryHitRate metric. Both metrics have values of 0. You verify that the dedicated gateway cluster is provisioned and used in the connection string.
You need to ensure that App1 uses the Azure Cosmos DB integrated cache. What should you configure?

a) the indexing policy of the Azure Cosmos DB container
b) the connectivity mode of the App1 CosmosClient
c) the consistency level of the requests from App1
d) the default consistency level of the Azure Cosmos DB account

Answer: b

Question: 5

You need to implement a trigger in Azure Cosmos DB Core (SQL) API that will run before an item is inserted into a container. Which two actions should you perform to ensure that the trigger runs?

Each correct answer presents part of the solution. NOTE: Each correct selection is worth one point.

a) Append pre to the name of the JavaScript function trigger.
b) For each create request, set the access condition in RequestOptions.
c) For each create request, set the trigger name in RequestOptions.
d) For each create request, set the consistency level to session in RequestOptions.
e) Register the trigger as a pre-trigger.

Answer: e

Question: 6

Reference Scenario: **click here**

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.

Answer: c

Question: 7

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) Configure the Backup & Restore settings for the account.
b) Create a new account that has a periodic backup policy.
c) Enable Continuous Backup for the account.
d) Configure the Point In Time Restore settings for the account.

Answer: c

Question: 8

You have an Azure Cosmos DB Core (SQL) API account. You run the following query against a container in the account.
SELECT -
IS_NUMBER("1234") AS A,
IS_NUMBER(1234) AS B,
IS_NUMBER({prop: 1234}) AS C -
What is the output of the query?

a) [{"A": false, "B": true, "C": false}]
b) [{"A": true, "B": false, "C": true}]
c) [{"A": true, "B": true, "C": false}]
d) [{"A": true, "B": true, "C": true}]

Answer: a

Question: 9

Reference Scenario: **click here**
You need to select the partition key for con-iot1. The solution must meet the IoT telemetry requirements. What should you select?

a) the timestamp
b) the device ID
c) the temperature
d) the humidity

Answer: b

Question: 10

You are implementing an Azure Data Factory data flow that will use an Azure Cosmos DB (SQL API) sink to write a dataset.
The data flow will use 2,000 Apache Spark partitions. You need to ensure that the ingestion from each Spark partition is balanced to optimize throughput.
Which sink setting should you configure?

a) Throughput
b) Write throughput budget
c) Batch size
d) Collection action

Answer: c

Avail the Study Guide to Pass Microsoft DP-420 Designing and Implementing Cloud-Native Applications Using Microsoft Azure Cosmos DB Exam:

- Find out about the DP-420 syllabus topics. Visiting the official site offers an idea about the exam structure and other important study resources. Going through the syllabus topics help to plan the exam in an organized manner.
- Once you are done exploring the **DP-420 syllabus**, it is time to plan for studying and covering the syllabus topics from the core. Chalk out the best plan for yourself to cover each part of the syllabus in a hassle-free manner.
- A study schedule helps you to stay calm throughout your exam preparation. It should contain your materials and thoughts like study hours, number of topics for daily studying mentioned on it. The best bet to clear the exam is to follow your schedule rigorously.
- The candidate should not miss out on the scope to learn from the DP-420 training. Joining the Microsoft provided training for DP-420 exam helps a candidate to strengthen his practical knowledge base from the certification.
- Learning about the probable questions and gaining knowledge regarding the exam structure helps a lot. Go through the **DP-420 sample questions** and boost your knowledge
- Make yourself a pro through online practicing the syllabus topics. DP-420 practice tests would guide you on your strengths and weaknesses regarding the syllabus topics. Through rigorous practicing, you can improve the weaker sections too. Learn well about time management during exam and become confident gradually with practice tests.

Career Benefits:

- Passing the DP-420 exam, helps a candidate to prosper highly in his career. Having the certification on the resume adds to the candidate's benefit and helps to get the best opportunities.

Here Is the Trusted Practice Test for the DP-420 Certification

EduSum.Com is here with all the necessary details regarding the DP-420 exam. We provide authentic practice tests for the DP-420 exam. What do you gain from these practice tests? You get to experience the real exam-like questions made by industry experts and get a scope to improve your performance in the actual exam. Rely on EduSum.Com for rigorous, unlimited two-month attempts on the **DP-420 practice tests**, and gradually build your confidence. Rigorous practice made many aspirants successful and made their journey easy towards grabbing the Microsoft Certified - Azure Cosmos DB Developer Specialty.

Start Online Practice of DP-420 Exam by visiting URL
https://www.edusum.com/microsoft/dp-420-designing-and-implementing-cloud-native-applications-using-microsoft-azure-cosmos