

Cosmos DB Guide

Partitions

Survival Guide



Survival Guide: Partitioning in Azure Cosmos DB

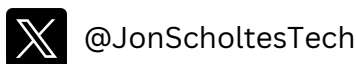
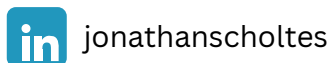
THE GUIDE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE GUIDE OR THE USE OR OTHER DEALINGS IN THE GUIDE

Follow Us

Stay connected and follow us for more insightful content, tutorials, and discussions!

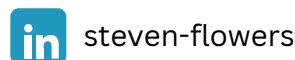
Jonathan Scholtes

Website: stochasticcoder.com



Steve Flowers

Website: thoughtreplica.com



Microsoft Azure Discord Server

Join the Microsoft [Azure Discord Server](#) for community support, discussions, and networking with other Azure enthusiasts.

Version: 1



Survival Guide: Partitioning in Azure Cosmos DB

Partitioning in Azure Cosmos DB is essential for efficient database management and scaling. By understanding and applying best practices, you can optimize both performance and cost. This guide will help you get started with partition keys, hierarchical partition keys, and changing partition keys when necessary. Partitioning distributes your documents across physical nodes, allowing Cosmos DB to scale out. Your partition key creates logical groupings of documents, which are then evenly distributed across these physical nodes.

Partitions

Physical Partitions

Physical partitions are the actual storage and compute resources used by Azure Cosmos DB. managed by Azure and are abstracted from the user, ensuring seamless scaling and performance optimization



Max 10,000 RUs per
physical partition



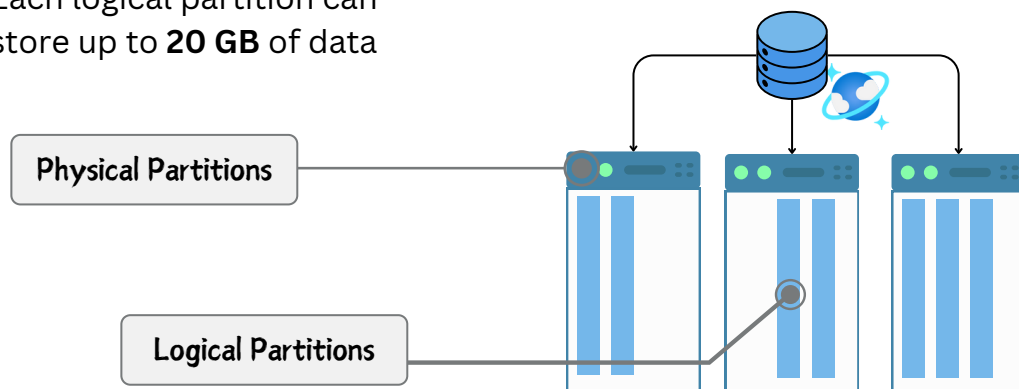
Each physical partition can
store up to **50 GB** of data.

Logical partitions

All items within a logical partition share the same partition key value. Azure Cosmos DB automatically manages logical partitions, distributing them across physical partitions to balance load and storage



Each logical partition can
store up to **20 GB** of data

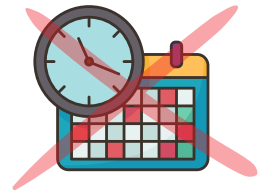


Partition Keys Best Practices

High Cardinality

A partition key should have a broad range of values. High cardinality ensures that data is evenly distributed across different partitions, avoiding scenarios where one partition is overloaded, known as hot partitions

Though they have a high cardinality, date time stamps are generally not a good candidate for a partition key. If queries often search for data based on a window of time, this can create a hot partition



Immutable Values

To maintain database performance and avoid data retrieval issues, keep the partition key unchanged. Altering a document's partition key is not possible once set, so select a stable property for queries to prevent rewriting the entire document.

Query Patterns

The partition key should distribute operations evenly across all logical partitions. This equal distribution helps utilize the provisioned throughput more efficiently.

Queries should consistently include the partition key in the WHERE clause to maximize efficiency and performance in Azure Cosmos DB.

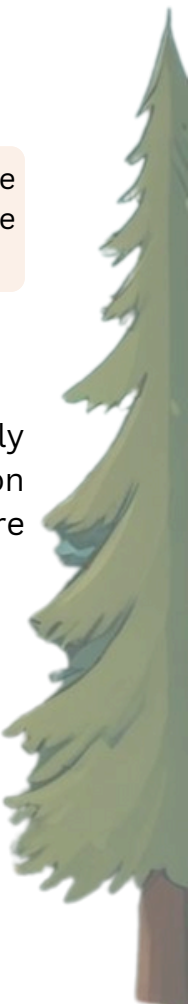


Uniform Distribution

The partition key should distribute operations evenly across all logical partitions. This equal distribution helps utilize the provisioned throughput more efficiently.

Forget About It!

If your container is small (less than 20k RUs), and won't need to scale out in the future, don't stress your partition key selection! Your data footprint is small enough that choosing the correct partition key will be less significant.

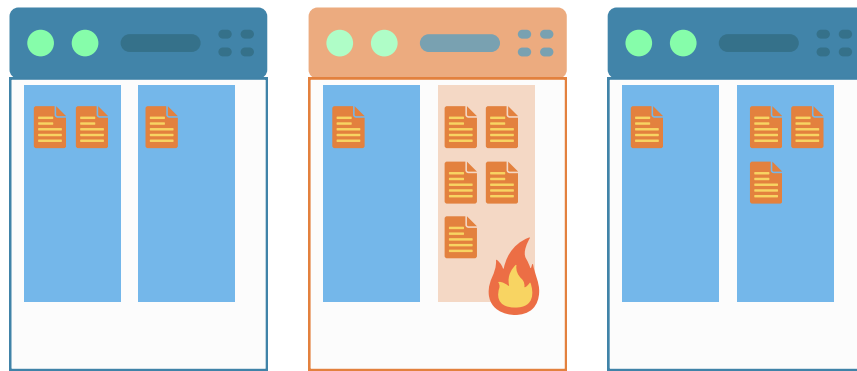


Changing the Partition Key

Changing the partition key in Azure Cosmos DB should be considered when you encounter performance or cost issues related to your current partition key. Specifically, consider changing the partition key if you face the following challenges:

Hot Partitions

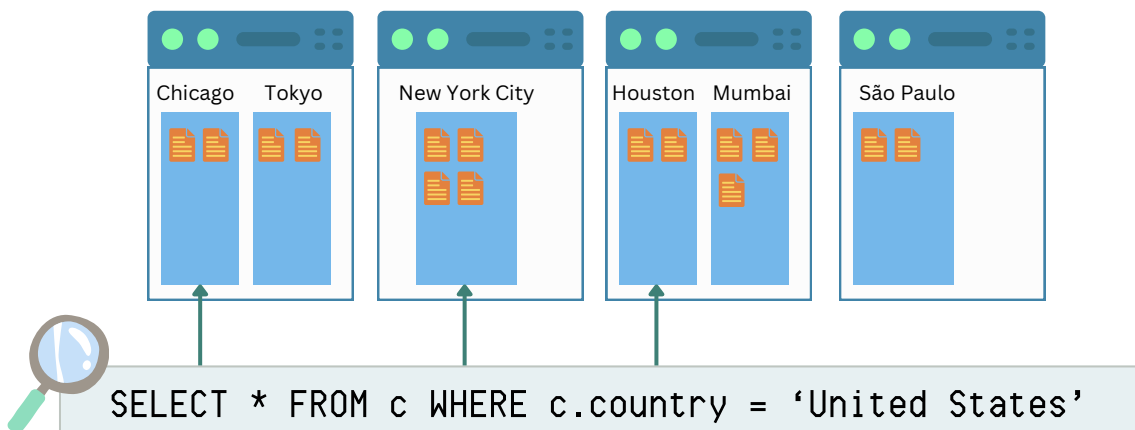
A hot partition occurs when some partitions receive significantly more requests than others, causing throttling and increased costs. This imbalance degrades performance and leads to higher expenses due to overburdened partitions and underutilized storage.



Cross-Partition Queries

Cross-partition queries in Azure Cosmos DB span multiple physical partitions, resulting in higher latency and increased costs compared to queries that target a single partition. Minimize their use to optimize performance and reduce expenses, especially in larger containers where they can significantly impact resource consumption.

If your container is small, you likely won't have enough physical partitions to be concerned about the performance impact of cross-partition queries.



Hierarchical Partition Keys

Hierarchical partition keys (HPK) or sub-partitioning allow you to create up to three levels of partition keys. This is particularly useful if your application handles multitenant environments or if you have large datasets that exceed the storage limits of a single partition



THE OLD WAY

Synthetic partition keys. Before HPK, developers would create a synthetic key by concatenating multiple values such as state, county, and city and store it in a string such as: "state-county-city". The only benefit was achieving greater cardinality based on a known hierarchy.

THE NEW way



HPKs allow us to avoid concatenating multiple fields and store them as standard fields. This makes development more streamlined. Instead of concatenating strings, you can build a hierarchical partition key object in your code! Additionally, you can narrow your search and point the database to your data at different levels of the hierarchy. For example, if your HPK is state/county/city, using "city" and a WHERE clause will have a smaller scan scope than "county", which will have a larger scan scope than city but a smaller scan scope than "state"

Resources



Optimizing database performance: Change partition keys of a container in Azure Cosmos DB for NoSQL API

<https://devblogs.microsoft.com/cosmosdb/optimizing-database-performance-change-partition-keys-of-a-container-in-azure-cosmos-db-for-nosql-api-preview/>



Hierarchical partition keys in Azure Cosmos DB

<https://learn.microsoft.com/en-us/azure/cosmos-db/hierarchical-partition-keys>



Partitioning and horizontal scaling in Azure Cosmos DB

<https://learn.microsoft.com/en-us/azure/cosmos-db/partitioning-overview>



Implement a data modeling and partitioning strategy for Azure Cosmos DB for NoSQL

<https://learn.microsoft.com/en-us/training/paths/implement-modeling-partitioning-azure-cosmos-db-sql-api/>