Caltech | Center for Technology & Management Education

Post Graduate Program in Cloud Computing

Powered by simplilearn

Caltech | Center for Technology & Management Education

PG CC - Microsoft Azure Architect Technologies: AZ:303

Powerd by simplilearn

Implement NoSQL Databases

Learning Objectives

By the end of this lesson, you will be able to:

- Configure Azure storage account tables
- Select appropriate Cosmos DB APIs

A Day in the Life of an Azure Architect

You are working for an organization which majorly deals with mobile applications that handle massive amounts of data. The applications should be highly responsive and always online.

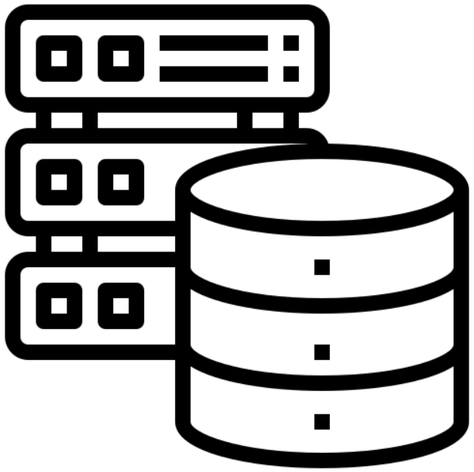
As an Architect you have been asked to suggest your organization an azure database solution that can help achieve low latency so that the application will be able to respond in real time to large changes in usage at peak hours, can store ever increasing volumes of data, and can make the data available to users in milliseconds.

To achieve all of the above along with some additional features, we will be learning a few concepts in this lesson that will help you find a solution for the given scenario.

Azure Storage Account Table

Azure Table Storage

Azure table is a NoSQL datastore ideal for storing structured and non-relational data.



It provides a key or attribute store with a schemaless design.

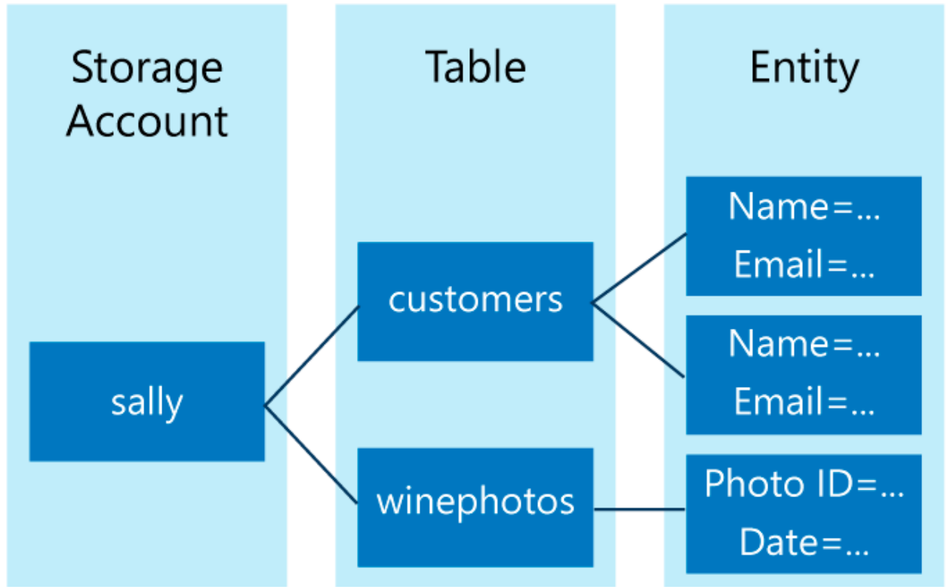
Use Cases for Table Storage

Table storage is commonly used for the following purposes:

- Storing terabytes of structured data to support web-scale applications
- Keeping denormalized datasets that don't require complex joins, foreign keys, or stored procedures that can be accessed quickly
- Querying data quickly with a clustered index
- Accessing data using the OData protocol and LINQ queries with the WCF Data Service.NET libraries

Table Storage Components

The table storage components are the following:



- *URL format* used by Azure Table Storage accounts is http://<storage account>.table.core.windows.net/<table>
- *Storage Accounts* is used to access Azure Storage
- *Table* is a collection of entities
- *Entity* is a set of properties, similar to a database row
- *Properties* are name-value pairs

image source: https://docs.microsoft.com/en-in/

Table Service Data Model

Table service can be accessed using the URI: https://myaccount.table.core.windows.net/

An entity can be assigned upto 255 properties.

Tables store data as collections of entities.

Property is a name, typed-value pair.

An entity is identified by a primary key and a set of properties.

Table Service Data Model

The following rules apply to table names:

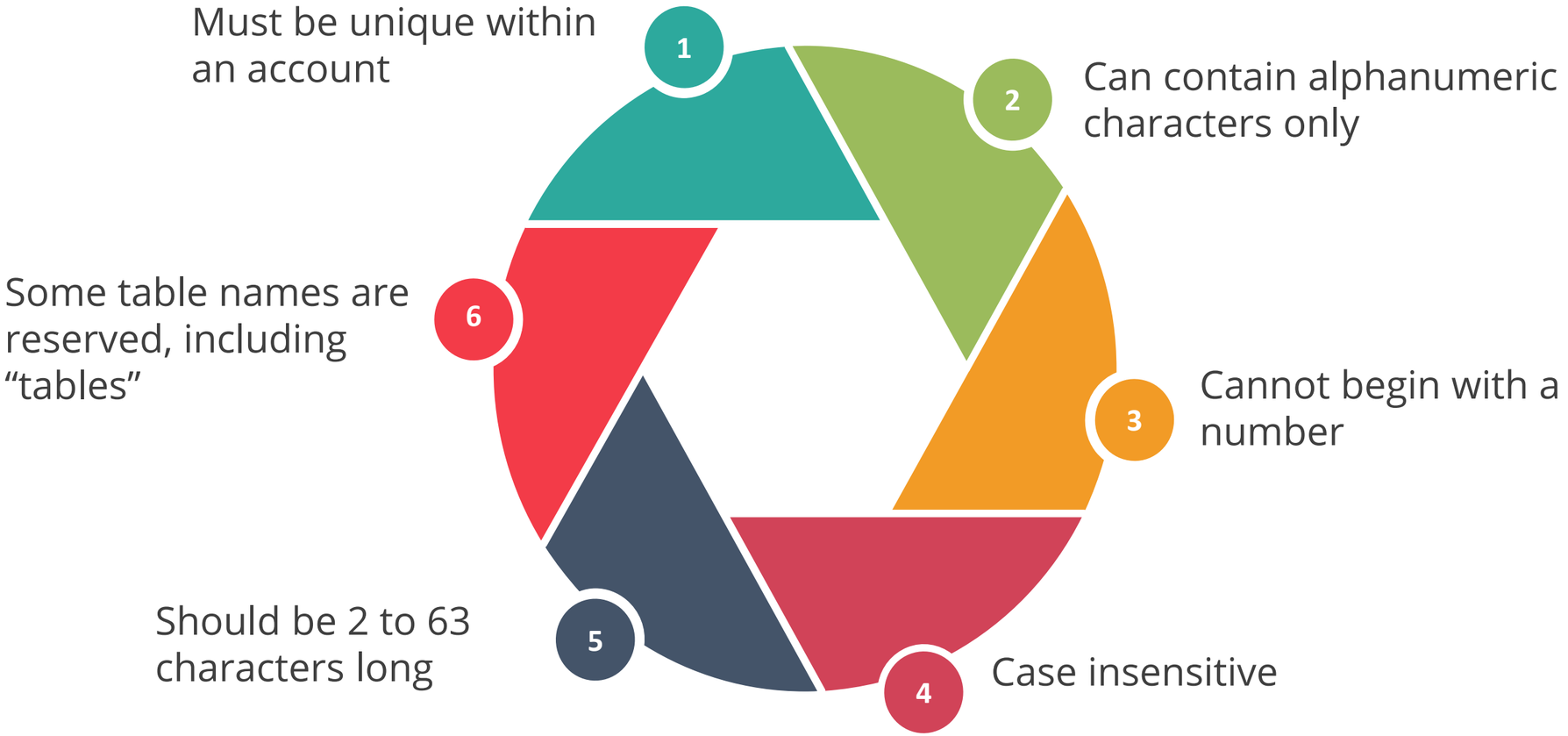


Table Service Data Model

System properties that are automatically included for every entity in a table are:

PartitionKey

This property serves as the basis for table partitioning.

RowKey

This property is a unique identifier for an entity within a partition.

Timestamp

This property is a date-time value that is kept on the server to keep track of when an entity was last changed.

Cosmos DB

Overview of Azure Cosmos DB

Azure Cosmos DB is a globally distributed database that is elastically scalable.

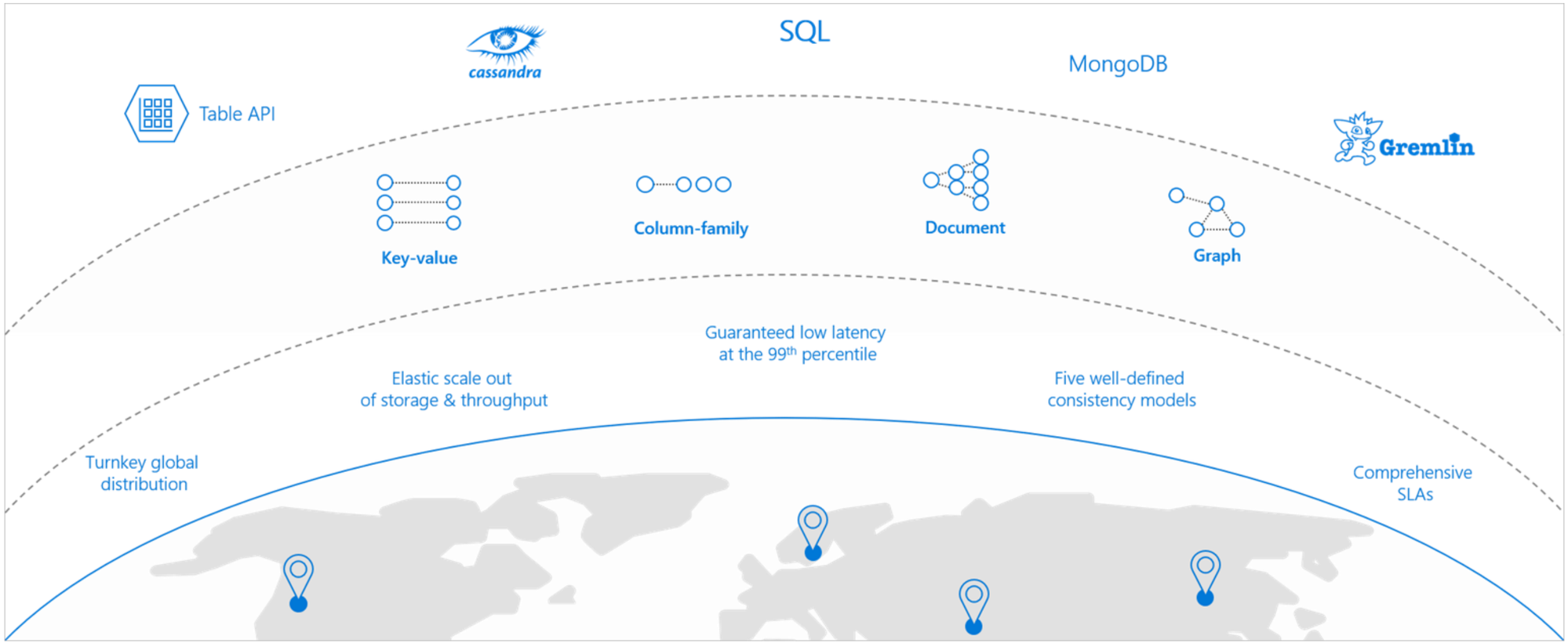


image source: https://docs.microsoft.com/en-in/

Overview of Azure Cosmos DB

Azure Cosmos DB is a NoSQL database with SLA-backed speed and availability, instant scalability, and open-source APIs.

Cosmos DB supports a number of APIs namely:

SQL

MongoDB

Table API

High Availability with Cosmos DB

Azure Cosmos DB transparently replicates users' data across all the Azure regions, associated with user's Cosmos account.

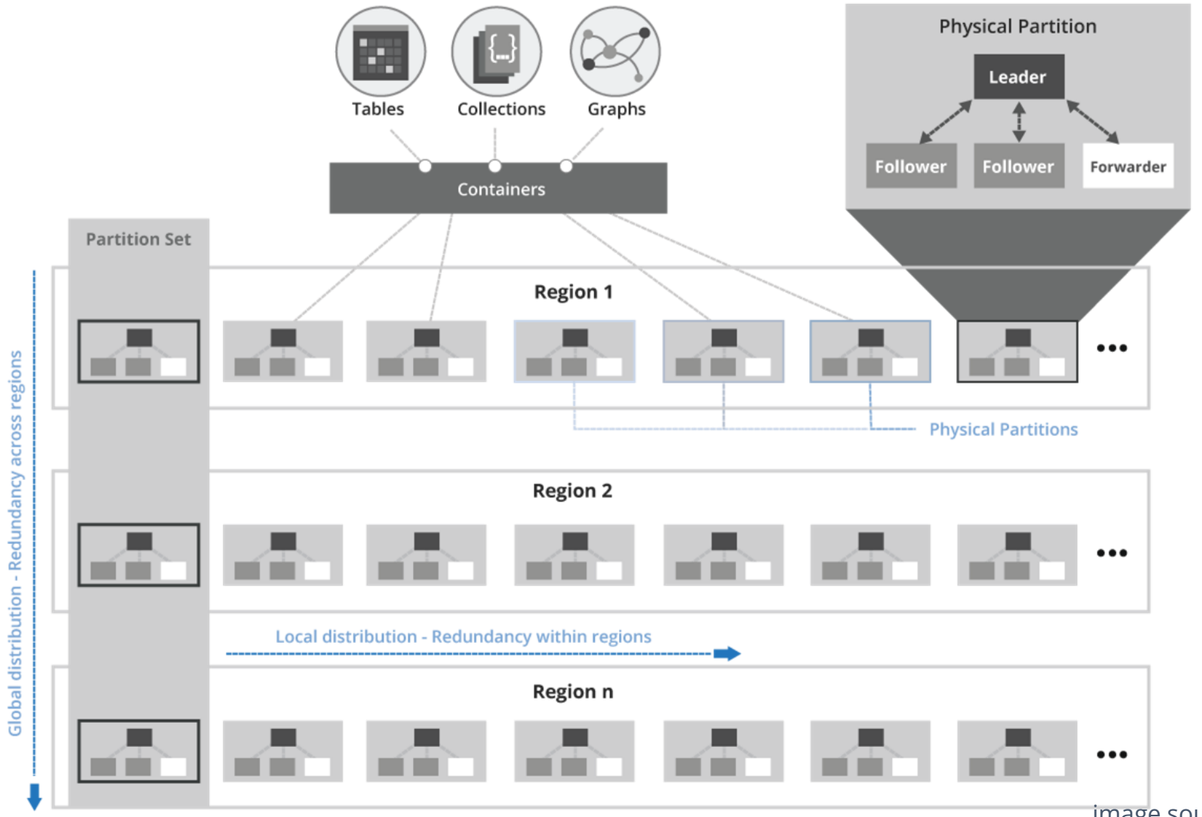


image source: https://docs.microsoft.com/en-in/

High Availability with Cosmos DB

SLAs for high availability is shown in the table below:

Operation Type	Single region	Multi-region (single region writes)	Multi-region (multi-region writes)
Writes	99.99	99.99	99.999
Reads	99.99	99.999	99.999

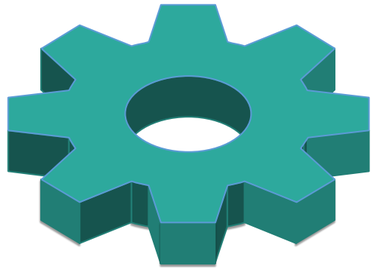
High Availability with Cosmos DB

To ensure high availability with Cosmos DB in the event of regional outages:

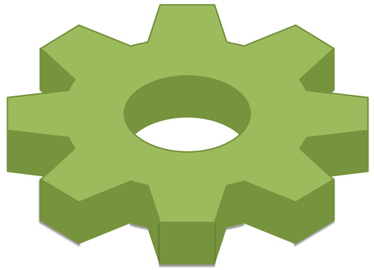
- Before a write operation is acknowledged to the client in Cosmos DB, the data is durably committed by a quorum of replicas in the area that is accepting the write operations.
- Both writes and reads will be highly accessible for multi-region accounts setup with multiple-write regions.

High Availability with Cosmos DB

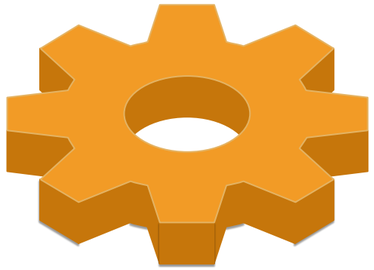
Multi-region accounts with a single-write region (write region outage):



Cosmos account promotes a secondary region to be the new primary write region during a write region outage.



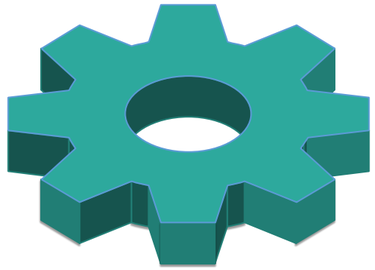
Conflicts feed enables available any write data that was not replicated when the region failed.



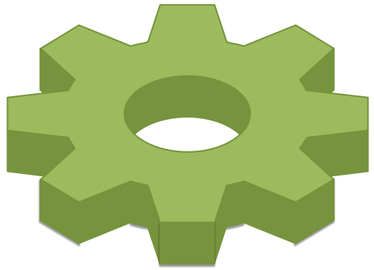
Write region that was impacted is automatically made available as a read region after recovery.

High Availability with Cosmos DB

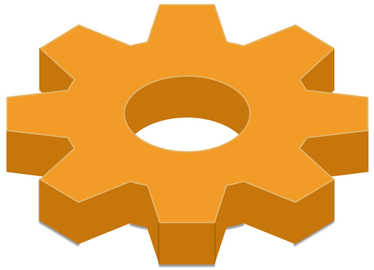
Multi-region accounts with a single-write region (read region outage):



Cosmos accounts using strong consistency with three or more read regions will remain highly available for reads and writes.



The impacted region will be marked as offline and will be automatically disconnected.



Calls automatically fall back to the current write region if none of the areas in the preferred region list are accessible.

Cosmos DB also offers availability zone support.

Overview of Azure Cosmos DB Supported APIs

MongoDB API

- Uses the Azure Cosmos DB platform to provide a massively scalable MongoDB service.
- Existing MongoDB libraries, drivers, tools, and applications are compatible.

Table API

A key-value database service designed to enhance existing Azure Table storage applications without requiring any changes to the apps.

Overview of Azure Cosmos DB Supported APIs

Gremlin API

- Graph database service that is fully managed and horizontally scalable.
- Open Graph APIs - compatible applications that are simple to design, run and work with highly connected datasets (based on the Apache TinkerPop specification, and Apache Gremlin).

Cassandra API

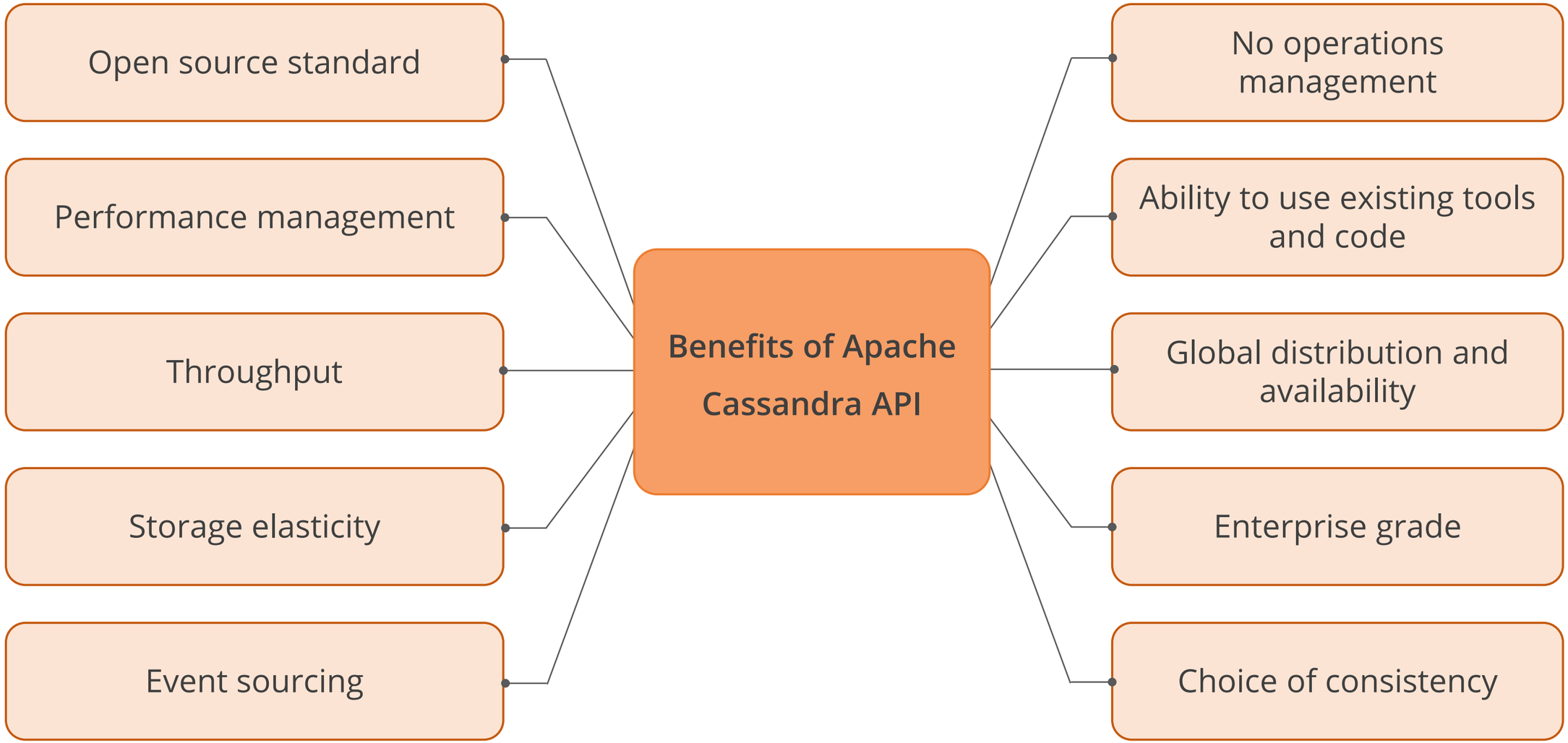
- The Azure Cosmos DB platform powers a globally distributed Apache Cassandra service that is compatible with existing Apache Cassandra frameworks, drivers, tools, and applications.

Overview of Azure Cosmos DB Supported APIs

SQL API

- Based on the Azure Cosmos DB database engine, a native JavaScript and JavaScript Object Notation (JSON) API is supported.
- Provides SQL-based query capabilities for documents based on their identifiers.
- Supports the execution of JavaScript logic in the form of stored procedures, triggers, and user-defined functions within the database.

Azure Cosmos DB Cassandra API

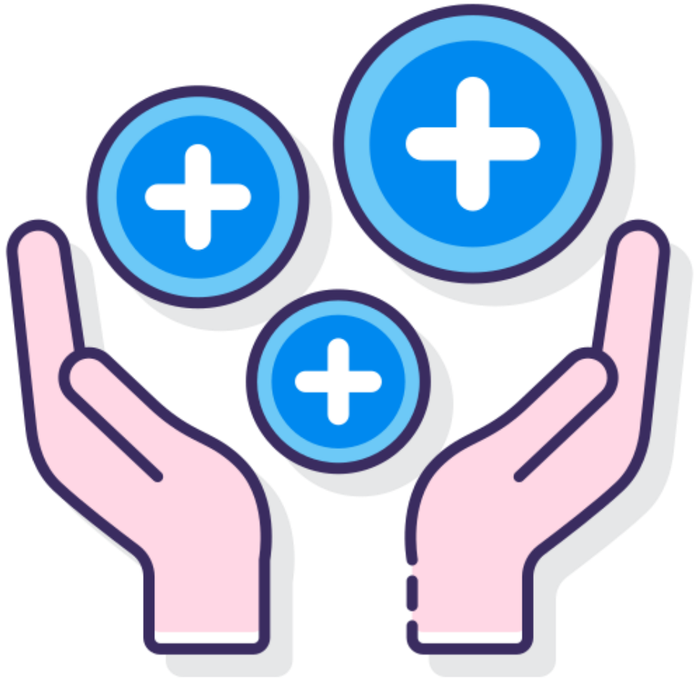


Azure Cosmos DB API for MongoDB

The MongoDB wire protocol is implemented by the Azure Cosmos DB API.

Azure Cosmos DB API for MongoDB

The key benefits of using Azure Cosmos DB API for MongoDB are:



- Having ability to migrate applications to Cosmos DB while preserving significant portions of the existing application logic.
- Keeping applications portable and continuing to remain cloud vendor-agnostic.
- supporting SLAs by financial resources for standard NoSQL APIs, powered by Cosmos DB.

Azure Cosmos DB Gremlin API

Features of Azure Cosmos DB graph database are listed below:

01 Elastically scalable throughput and storage

02 Multi-region replication

03 Fast queries and traversals

04 Fully managed graph database

05 Automatic indexing

06 Tunable consistency levels

Azure Cosmos DB Gremlin API

Azure Cosmos DB offers a graph database service via the Gremlin API.

- Graph databases rely on persisting relationships in the storage layer, for efficient graph retrieval operations.
- Property graph objects include:
  - Vertices
  - Edges
  - Properties

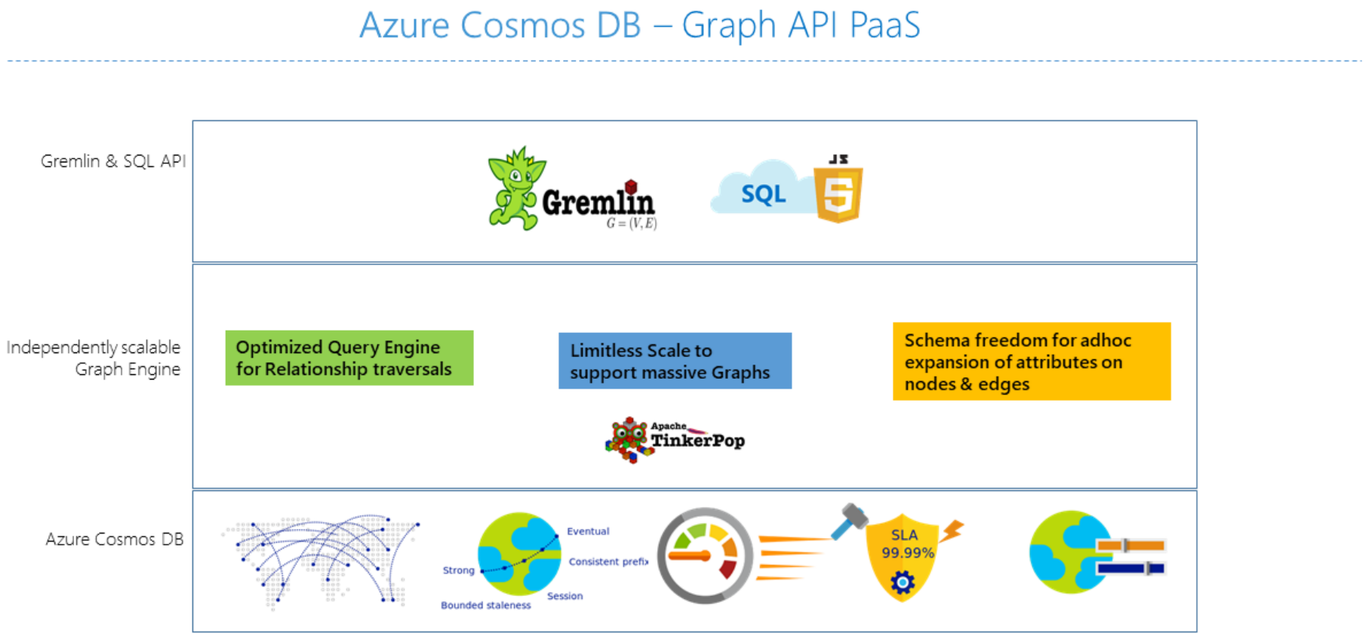
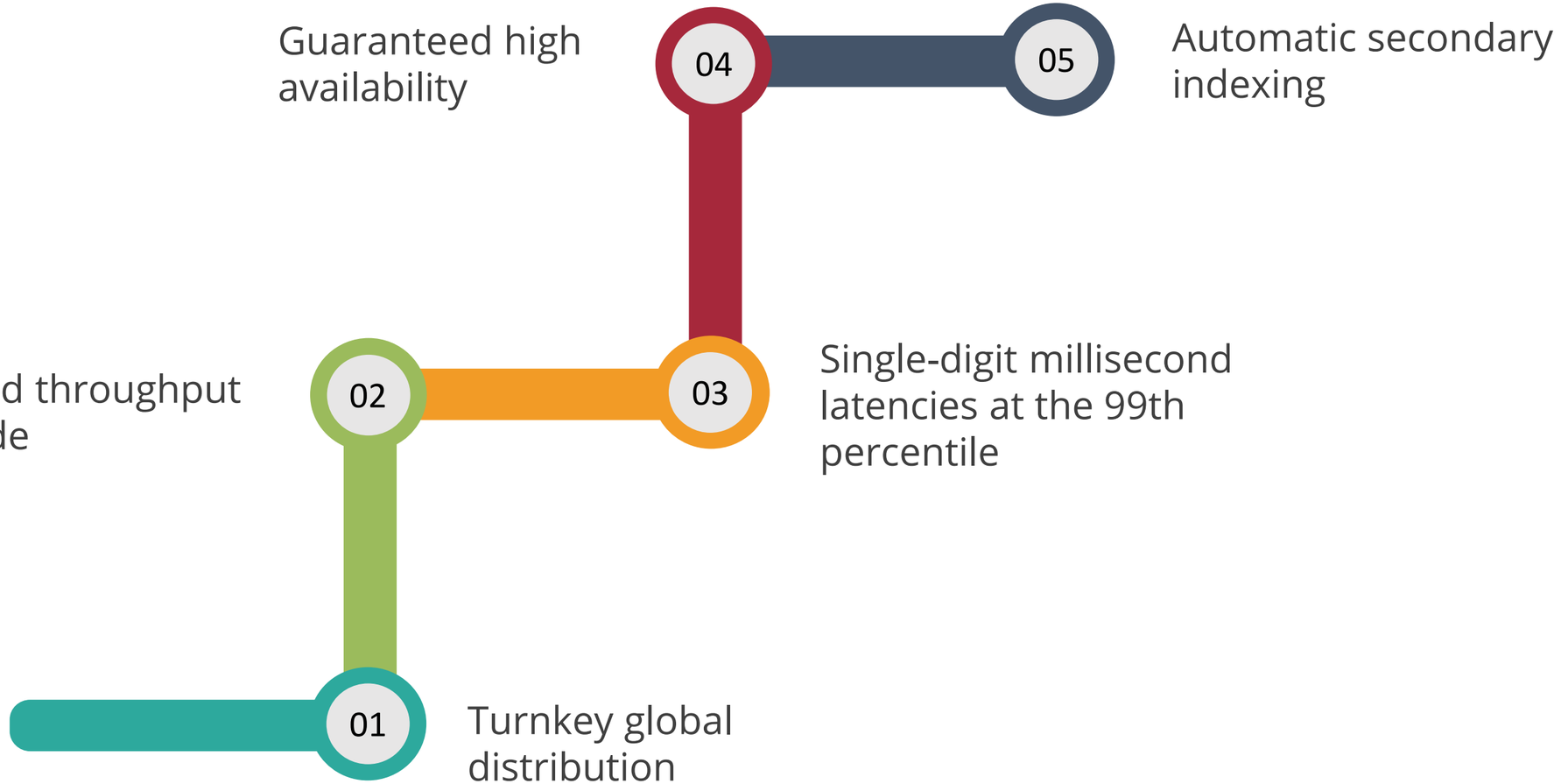


image source: https://docs.microsoft.com/en-in/

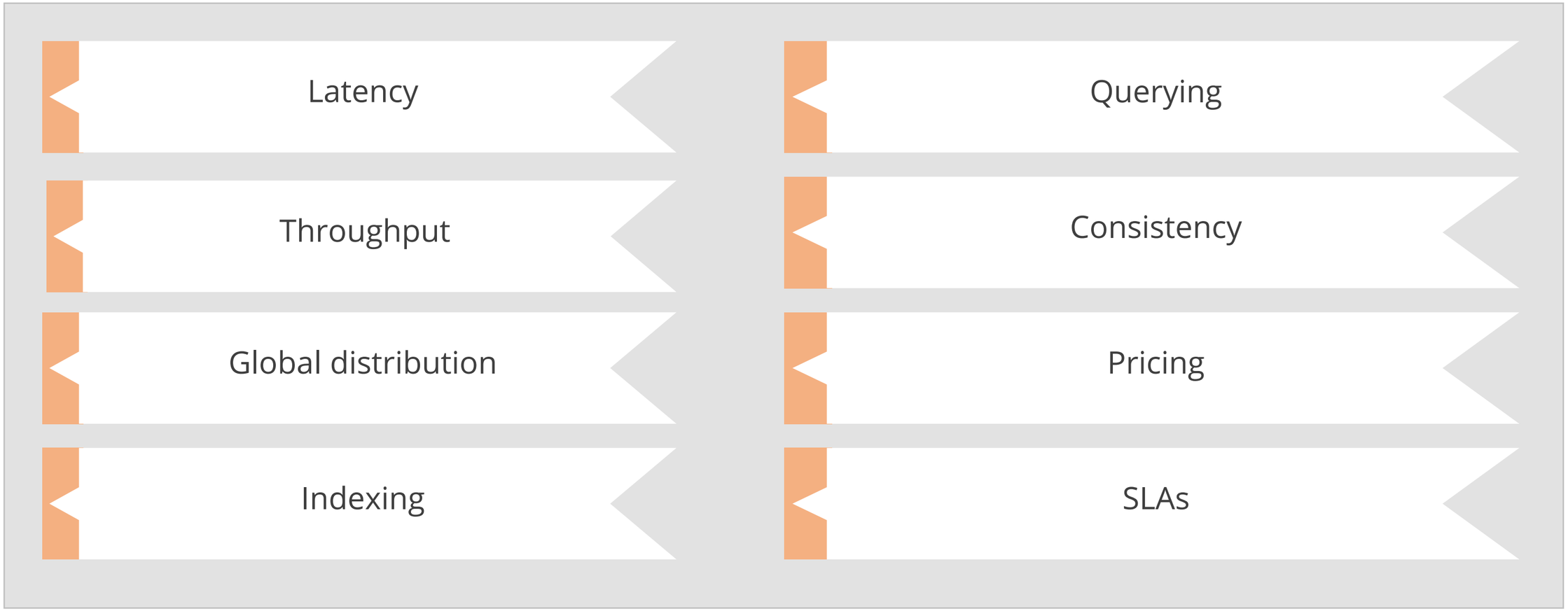
Azure Cosmos DB Table API

It is suitable for applications designed for Azure Table storage and require premium features such as:



Azure Cosmos DB Table API

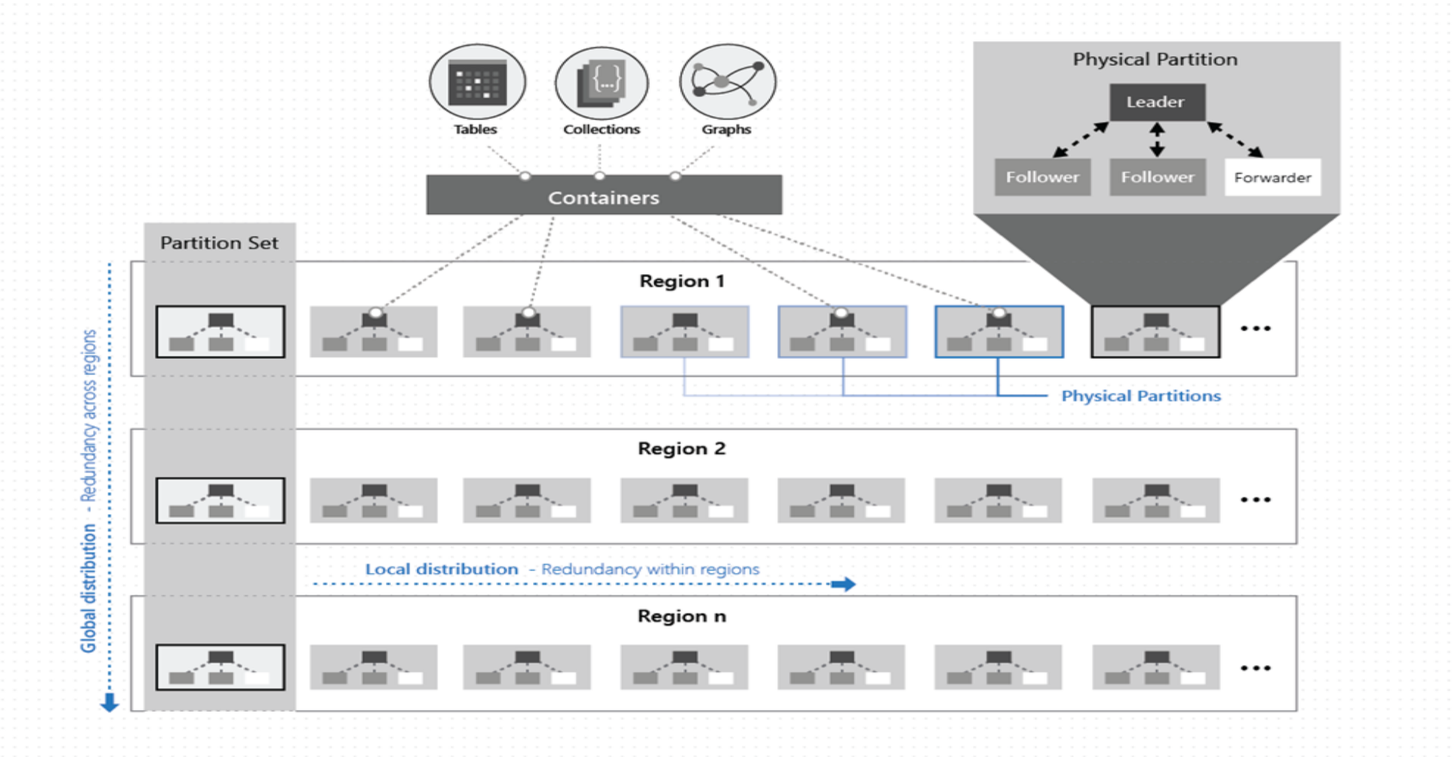
Switching to the Azure Cosmos DB Table API from Azure Table Storage offers the following advantages:



Cosmos DB Replicas

Cosmos DB Replicas

Azure Cosmos DB provides high availability in two primary ways.

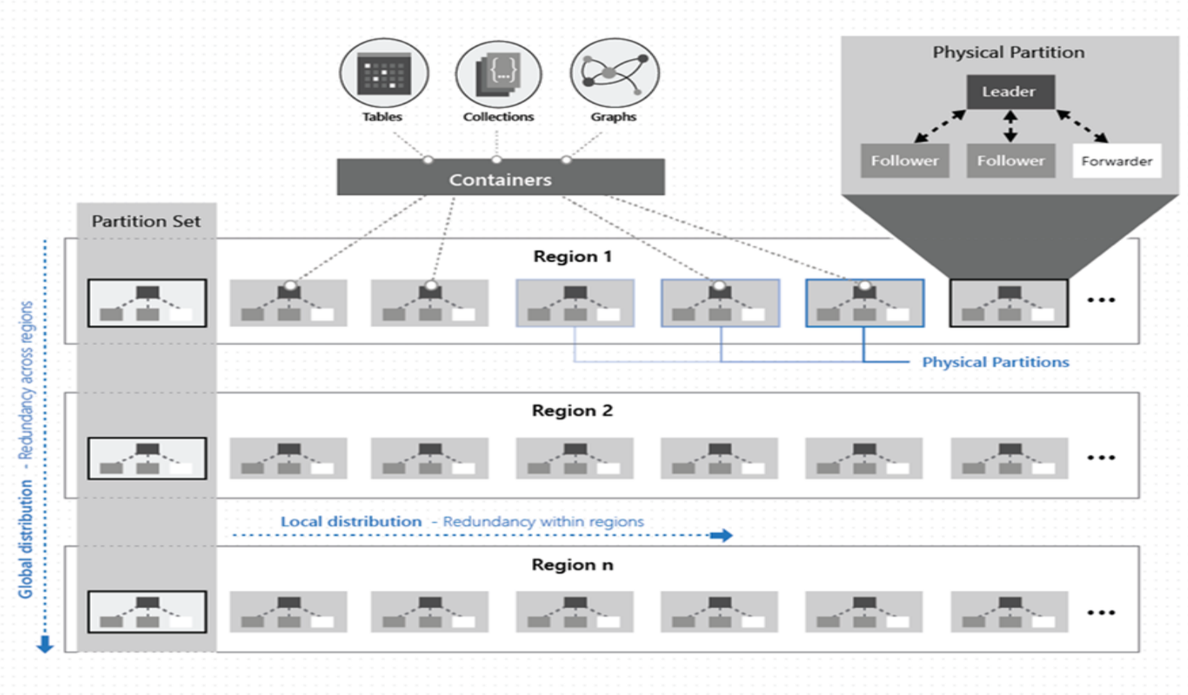


Azure Cosmos DB maintains four copies of your data as replicas within physical partitions.

image source: https://docs.microsoft.com/en-in/

Cosmos DB Replicas

The data is horizontally partitioned within Azure Cosmos containers.



A partition-set is a collection of multiple replica-sets.

- Each partition across all the regions is replicated.
- Each region contains all the data partitions of an Azure Cosmos container.

It can serve reads as well as writes when multi-region writes is enabled.

image source: https://docs.microsoft.com/en-in/

Assisted Practice

Azure Cosmos DB

Duration: 15 Min

Problem Statement:

You've been given the task of creating an Azure Cosmos DB to help your application achieve low latency and high availability.

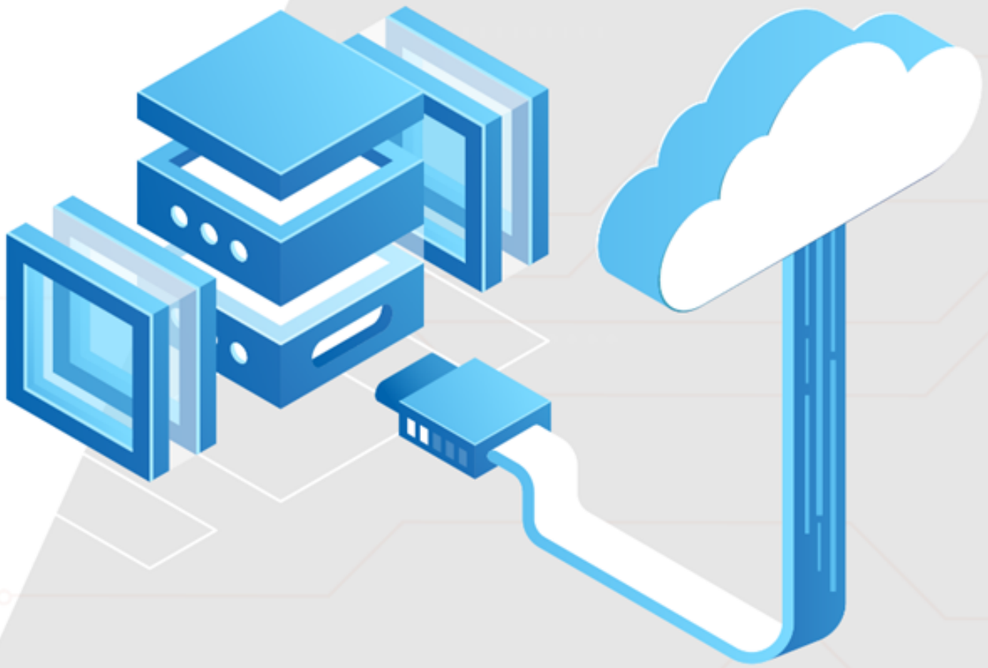
Assisted Practice: Guidelines

Steps to create an Azure Cosmos DB account on Azure Platform:

1. Login to your azure portal
2. Search and Select Azure Cosmos DB
3. Select New in Azure Cosmos DB pane and enter required fields

Key Takeaways

- Azure table is a NoSQL datastore ideal for storing structured and non-relational data.
- PartitionKey, RowKey, and Timestamp are System properties that are automatically included for every entity in a table.
- Azure Cosmos DB transparently replicates user's data across all Azure regions associated with user's Cosmos account.
- The table API is provided by Azure Cosmos DB for applications that are designed for Azure Table storage and require premium features.



Thank you