Microsoft Azure Cosmos DB

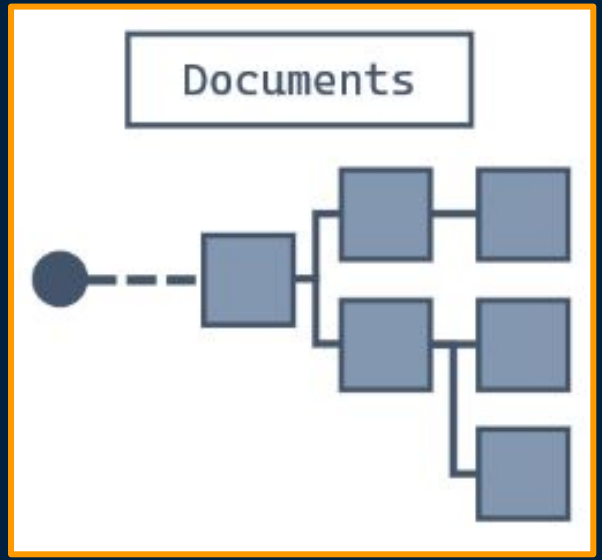
COSC 516 – Cloud Databases

Azure Cosmos DB

Azure Cosmos DB is a fast NoSQL database.

- Single-digit millisecond response times
- Automatic and instant scalability

  - Document data model
  - Supports JSON natively



Source: https://learn.microsoft.com/en-us/azure/cosmos-db/introduction

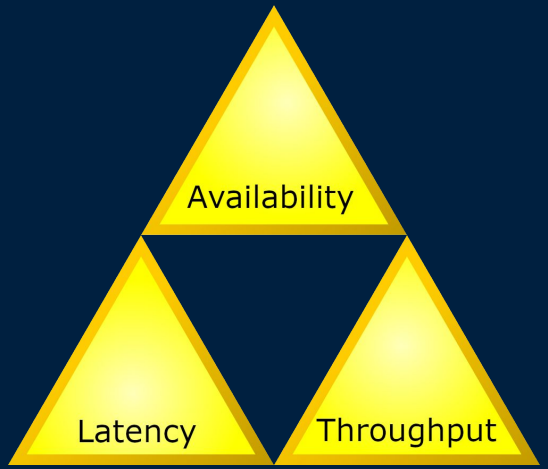
Key Features

- Globally distributed → over all major Azure regions (turnkey)
- High availability → 99.999% for reads and writes
- Elastic scale → potential for 100,000,000+ requests/sec
- Automatic indexing → no need to worry about indexing data
- Low latency → under 10ms for reads and writes
- Flexibility → does not enforce a specific schema
- Consistency models → balance between consistency and performance

Consistency Models

Azure Cosmos DB provides five **consistency models** to choose from:

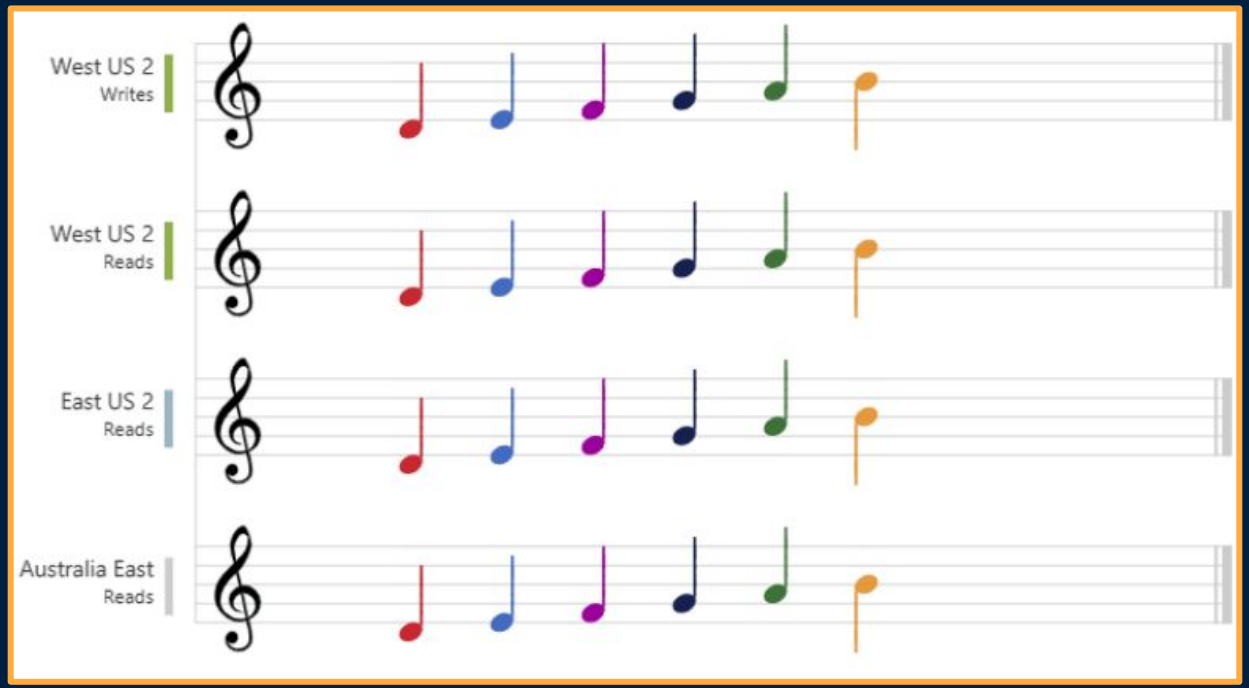
- → Strong Consistency
- → Bounded Staleness
- → Session
- → Consistent Prefix
- → Eventual Consistency



Source: https://learn.microsoft.com/en-us/azure/cosmos-db/consistency-levels

Strong Consistency

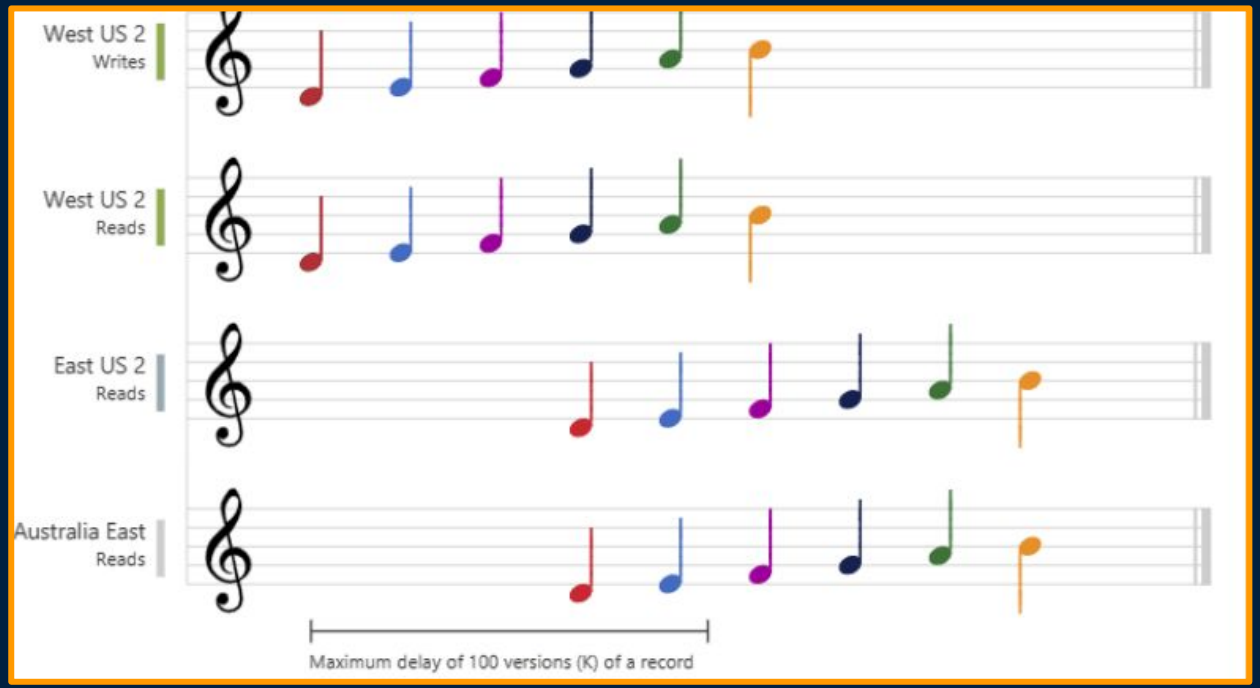
- Reads are guaranteed to return the most recent version of an item
- Higher write latencies due to data replication across large distances



Source: https://learn.microsoft.com/en-us/azure/cosmos-db/consistency-levels

Bounded Staleness

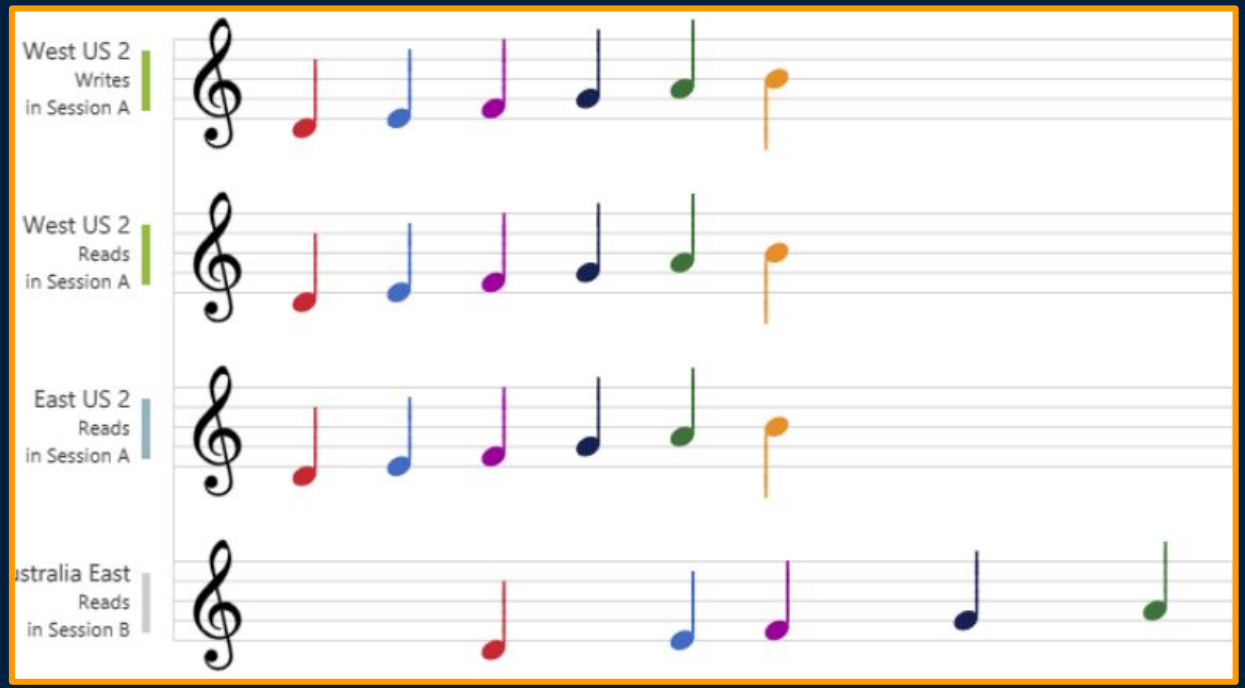
- Data is replicated asynchronously
- Has a staleness window that prioritizes writes



Source: https://learn.microsoft.com/en-us/azure/cosmos-db/consistency-levels

Session

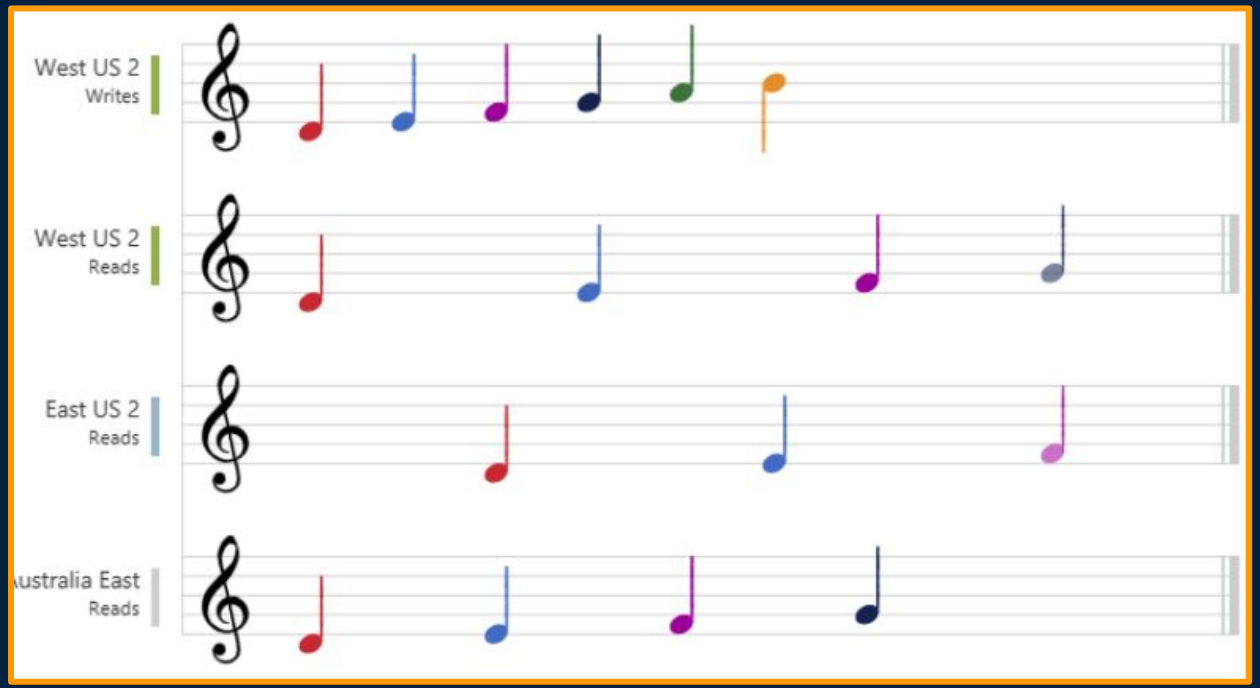
- Client-centric, rather than data-centric
- Most widely used for both single region and globally distributed applications



Source: https://learn.microsoft.com/en-us/azure/cosmos-db/consistency-levels

Consistent Prefix

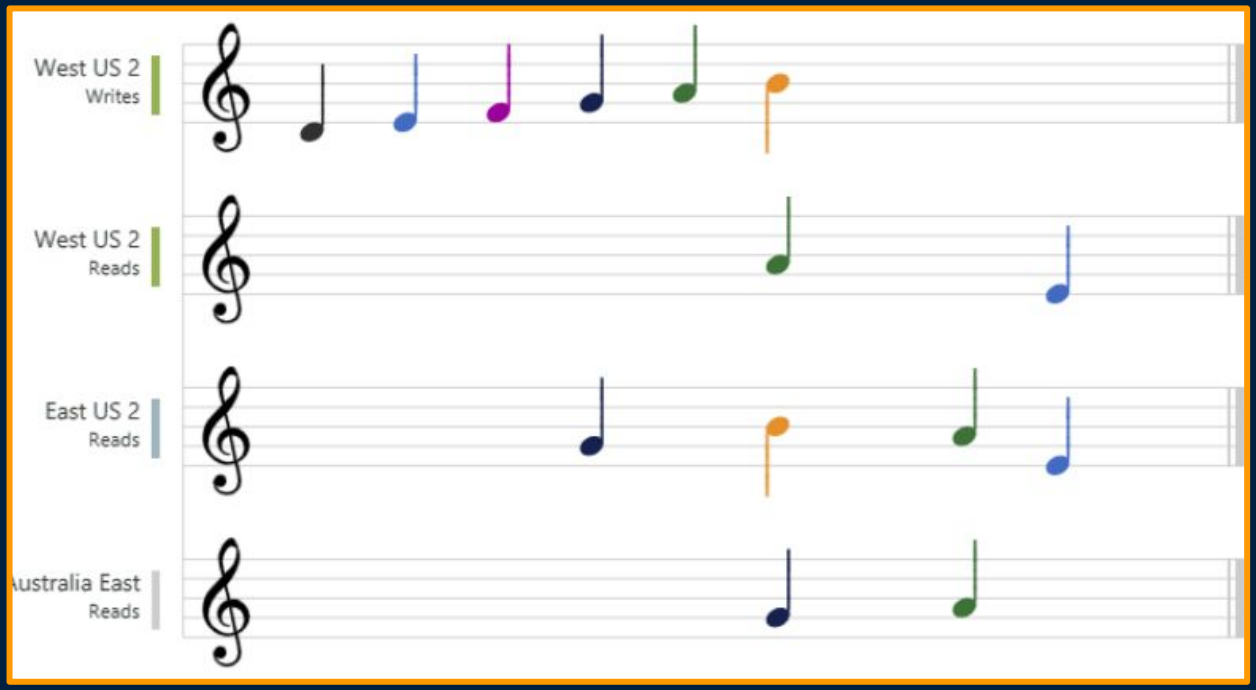
- Guarantees readers will never see out of order writes
- Write latencies, availability, and throughput comparable to eventual consistency



Source: https://learn.microsoft.com/en-us/azure/cosmos-db/consistency-levels

Eventual Consistency

- Weakest form of consistency → replicas converge as **consistent… eventually**
- Ideal for applications where throughput is more important than order



Source: https://learn.microsoft.com/en-us/azure/cosmos-db/consistency-levels

Use Cases

IoT / Telemetry

- Sensor workloads can write massive volumes of data
- Azure Cosmos DB is optimized for ***write-heavy workloads***

Retail / Marketing

- Retail and marketing workloads can experience unexpected swings in usage
- Azure Cosmos DB can handle requests ***during peak usage***

Web / Mobile

- User-generated content differing in size, shape, and volume
- Azure Cosmos DB can store data of ***varying schemas***

Multiple APIs

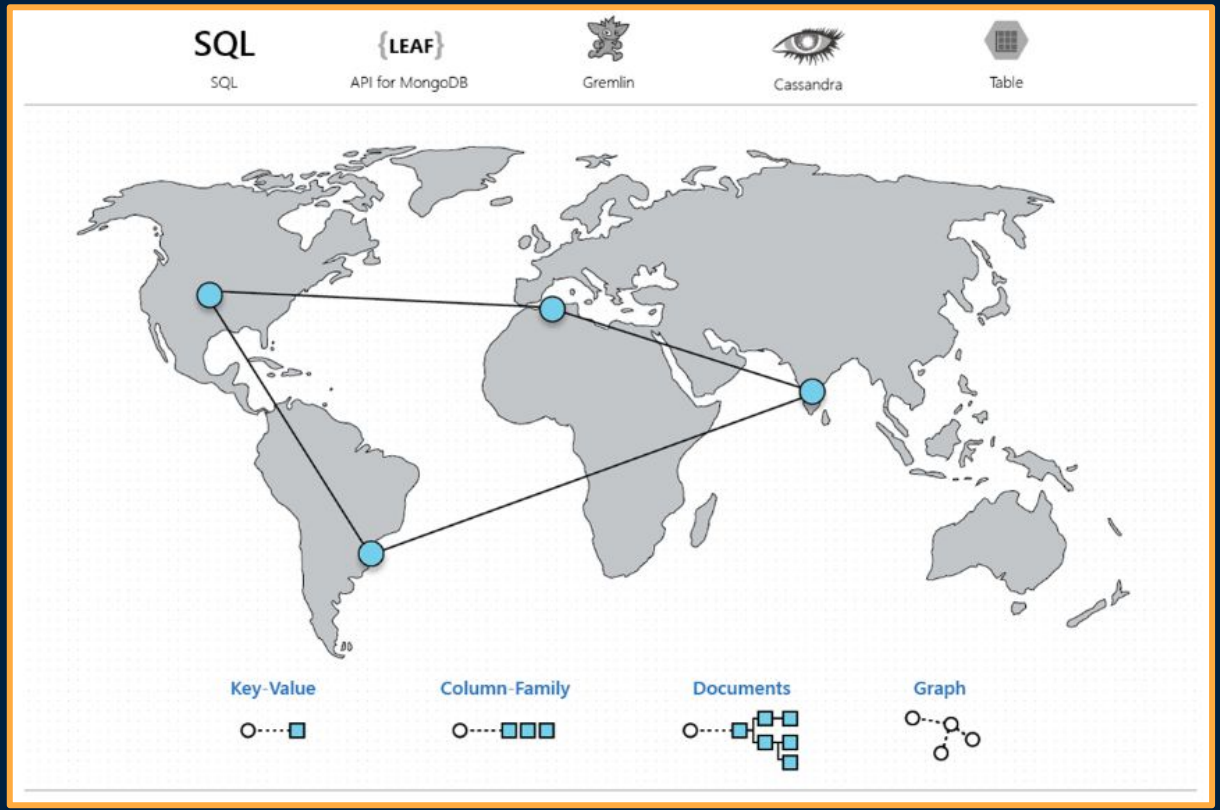
★ NoSQL

→ MongoDB

→ Gremlin

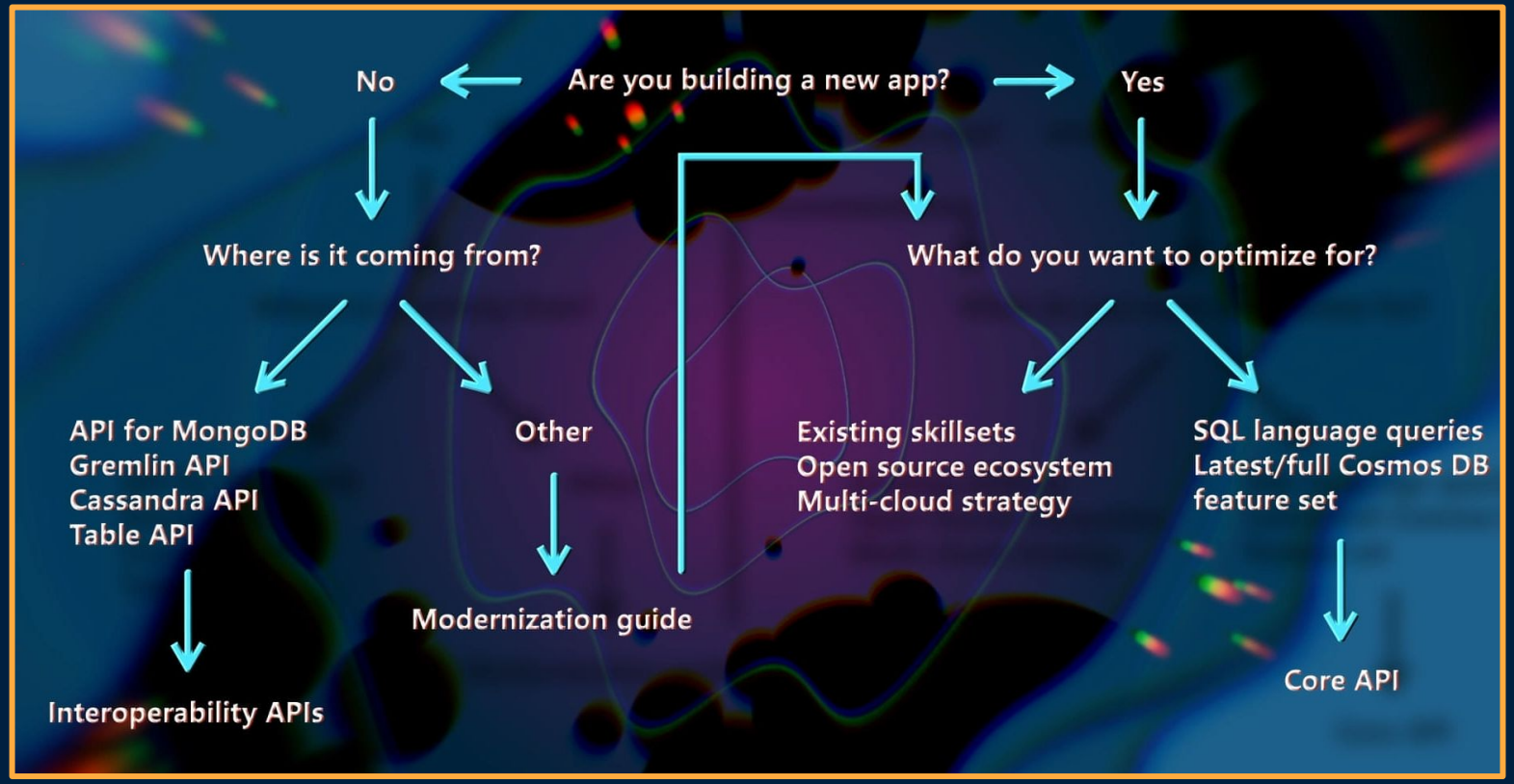
→ Cassandra

→ Azure Table



Source: https://learn.microsoft.com/en-us/azure/cosmos-db/introduction

Choosing an API



Source: https://learn.microsoft.com/en-us/azure/cosmos-db/choose-api

Organizations Using Cosmos DB

Source: https://learn.microsoft.com/en-us/azure/cosmos-db/use-cases

Solving Industry Specific Needs

Solution Areas	Financial	Retail	Manufacturing	Government	Health	Education
Transactional	Customer experience management	Retail management system	Connected field service	Citizen services tracking	Care coordination	Educational lifecycle management
Real-time experiences	Seamless services	Commerce experience	Supply chain visibility	Secure global platform	Customer care experience	Global collaboration
Big data processing	Risk compute	Demand forecasting	Server logs	Enterprise grade control	Genomic data	Student analytics
AI	Risk mitigation	Learn customer habits	Predictive processes	Smart buildings	Automated appointments	Intelligent resources for students
Examples of SaaS apps	Customer Service Banking Screen Compliance Assessments	Pricing & Promotion Retail Personalization Inventory Optimization	Quote-to-cash system Sales automation Predictive Maintenance	Citizen Service Requests Grants Management	Patient Coordination Risk prediction Medical Claim Analytics	Learning Management System Personalized learning
Unique need	**Security**	**User experience**	**Scale**	**Security**	**Compliance**	**Innovation**

Source: https://learn.microsoft.com/en-us/azure/cosmos-db/use-cases

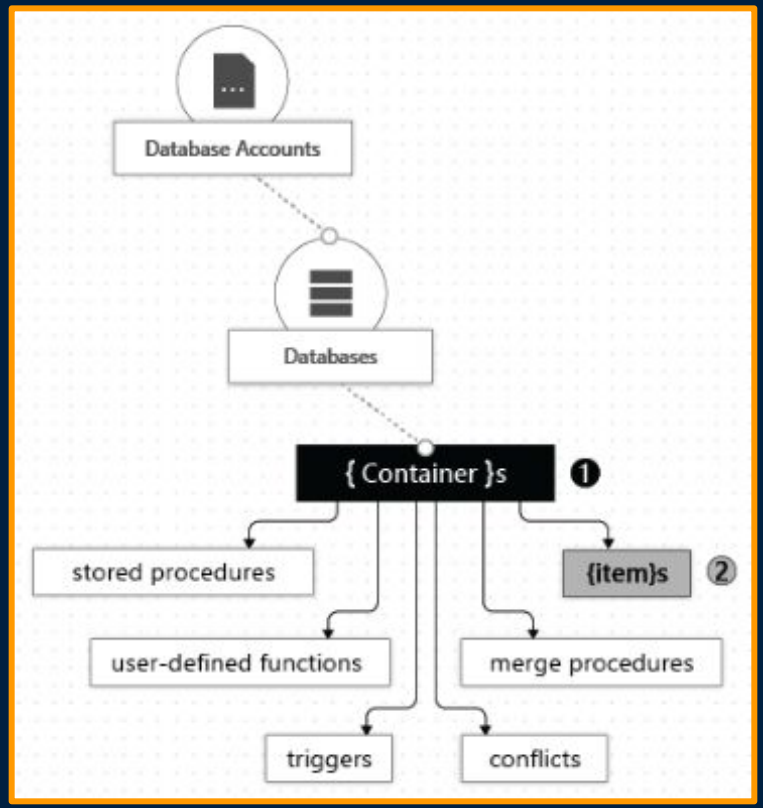
Resource Model

Account

- Can configure the regions for data storage
- Can manage unlimited provisions
- Contains globally unique DNS name
  - *https://{account}.documents.azure.com/*

Database[s]

- Manage users, permissions, and containers
- Can provision throughput

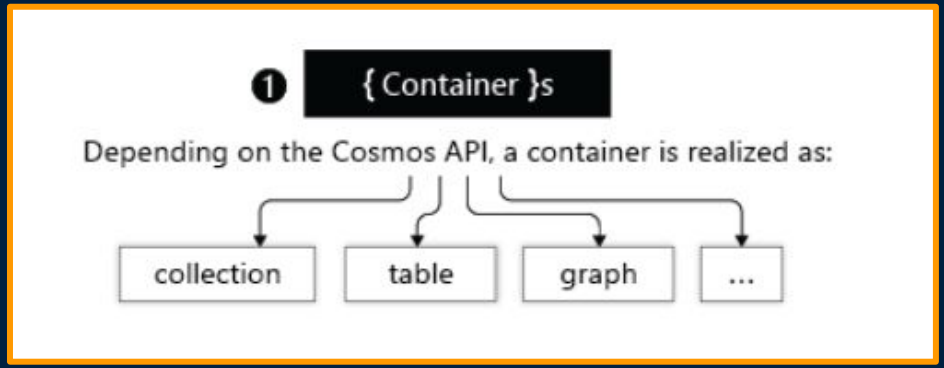


Source: https://learn.microsoft.com/en-us/azure/cosmos-db/account-databases-containers-items

Resource Model Continued…

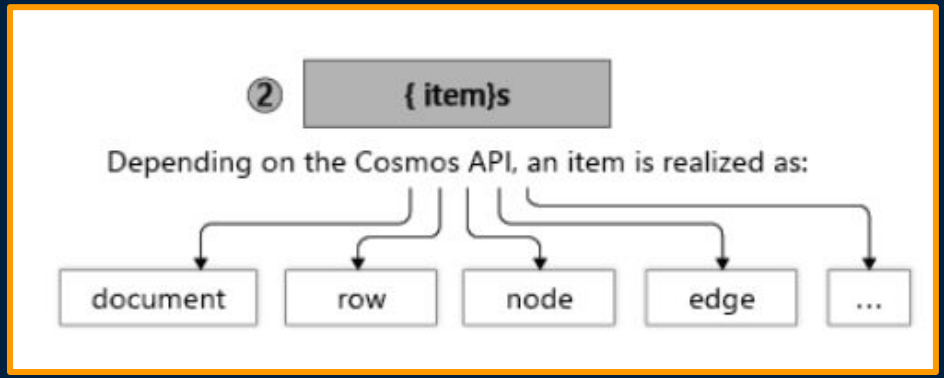
Container[s]

- Normally used to provision throughput
- Can configure indexing policy
- Can configure time-to-live (TTL)



Item[s]

- Individual documents in JSON format
- Write operations are atomic



Source: https://learn.microsoft.com/en-us/azure/cosmos-db/account-databases-containers-items

Partitions

Partitioning is a technique used in Azure Cosmos DB to divide and categorize similar items into different containers known as partitions.

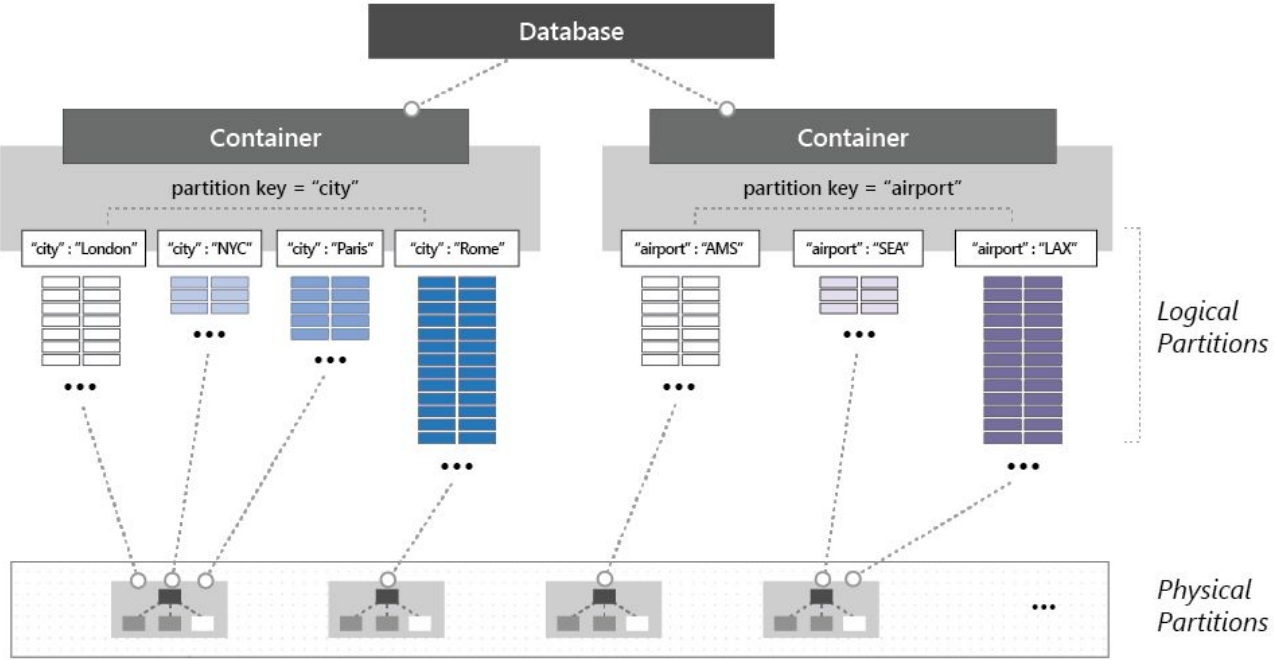
There are ***two*** kinds of partitions:

- ***Logical Partition*** - items in a container are divided into distinct subsets
- ***Physical Partition*** - entirely managed by Azure Cosmos DB

A physical partition can be mapped to one or more logical partitions.

- A logical partition can grow to a maximum size of ***20 GB*** and ***10,000 RU/s.***
- A physical partition can only have a maximum size of ***50 GB*** and ***10,000 RU/s.***

Partition Diagram



Partition Optimization

A ***Hot partition*** refers to having too many requests directed to a partition.

Partition key selection factors:

- Even Distribution
- High cardinality - better distribution and scalability of data
- Spread request units

Partition Key Selection Scenario 1

We want to have a partition on the sales of mobile devices.

1. Mobile device ***model type*** can be: “Apple”, “Google” and “Samsung”.
2. We have another feature called ***Mobile Identification Number (MIN)***

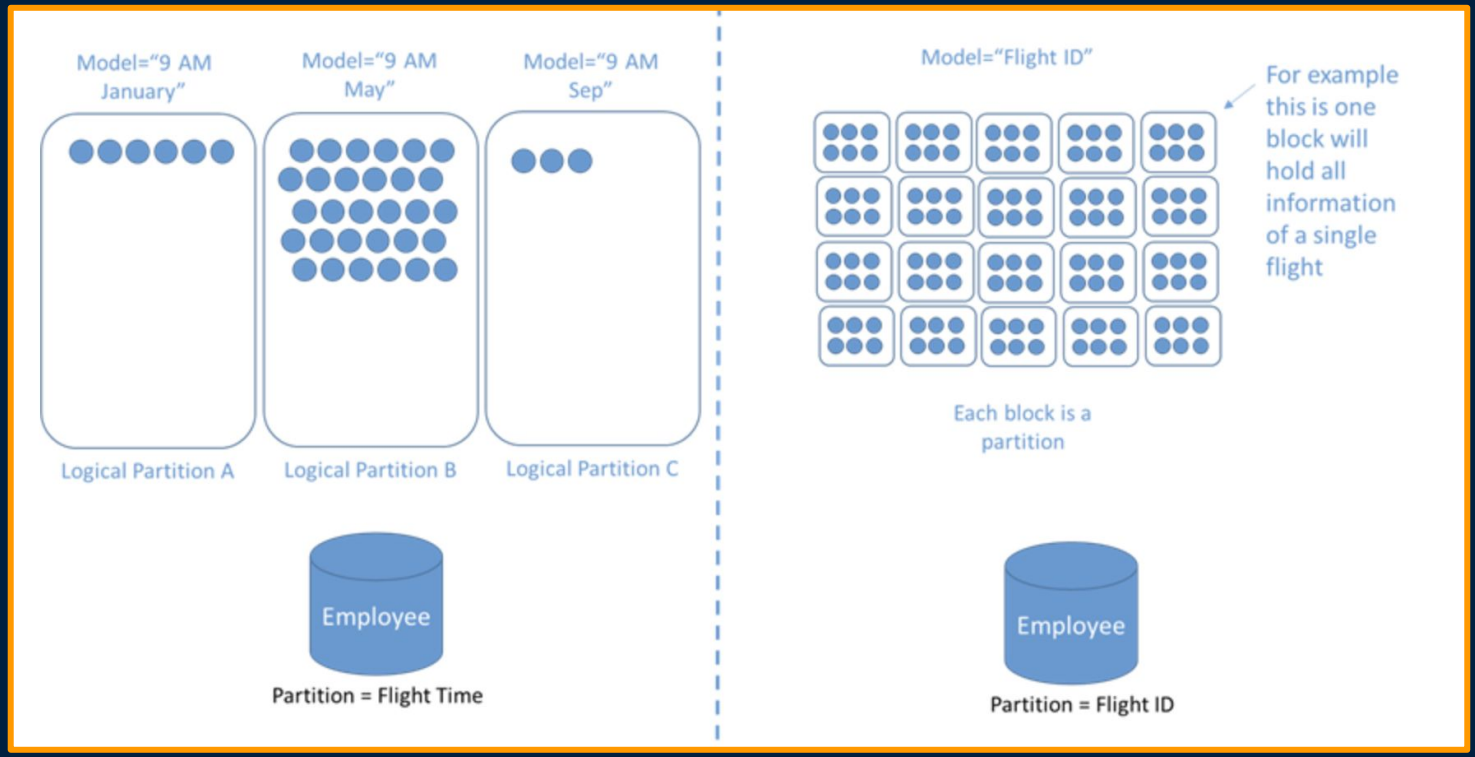
What would be a better choice for the partition key:

Model Type

or

Mobile Identification Number (MIN)?

Partition Key Selection Scenario 2



What would be a better choice for partition key: ***Flight Time*** or ***Flight ID*** ?

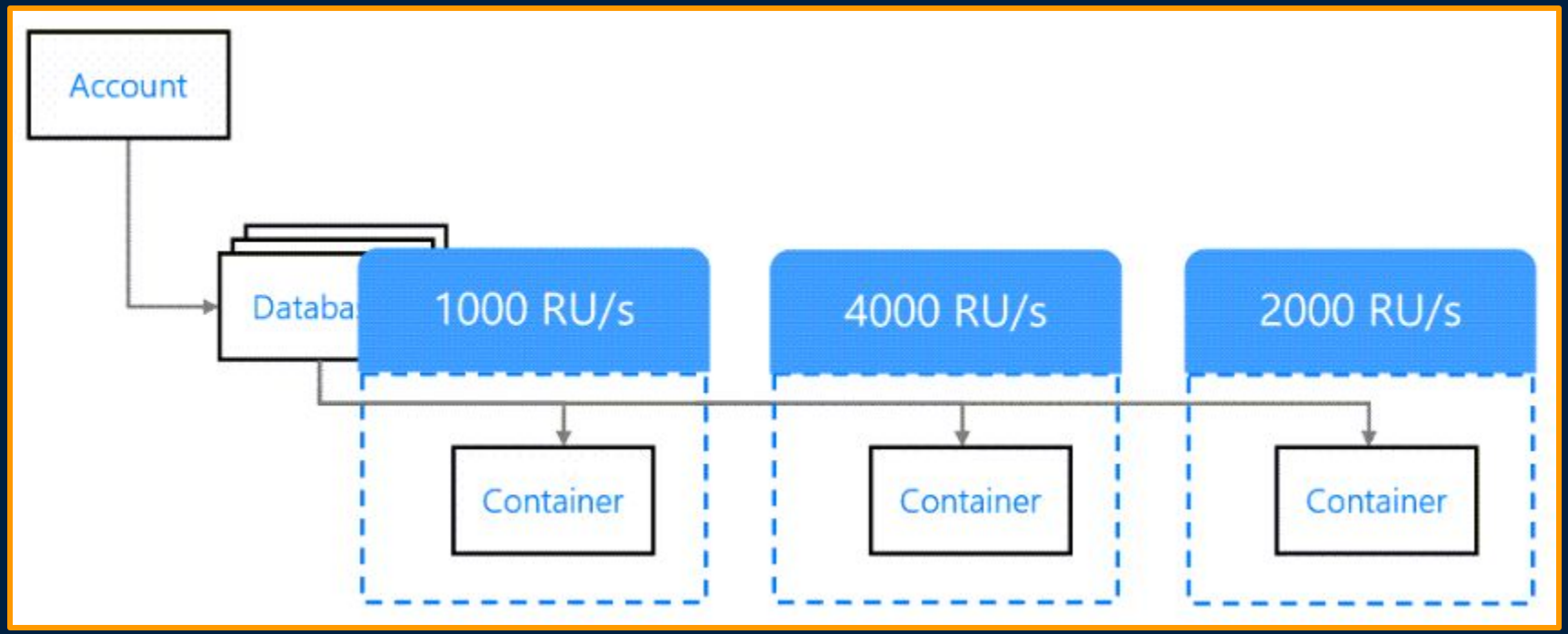
Throughput

- Each container is a unit of scalability for both ***throughput*** and ***storage***
  - Partitioned horizontally within a region
  - Distributed across all regions configured in the account

- Throughput can be configured by:
  - Container (most common)
  - Database
  - Both container and database (container takes precedence)

Container-level Throughput Provisioning

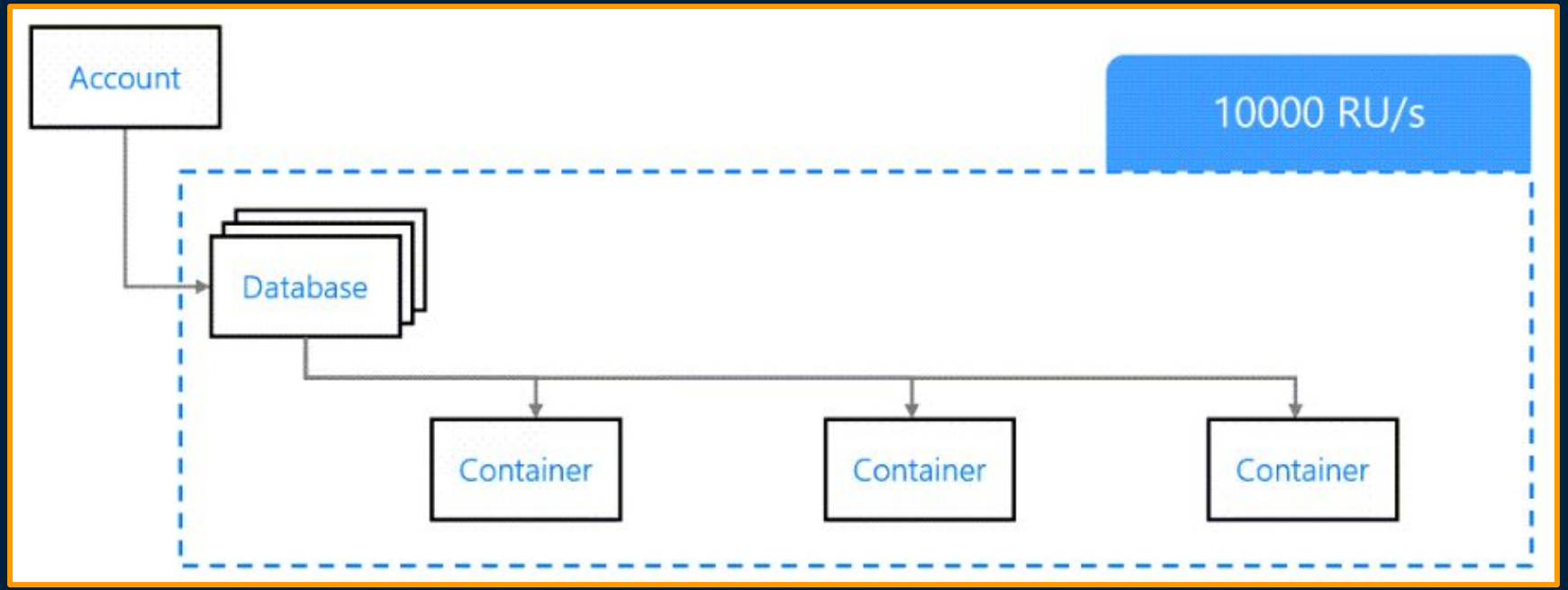
- Containers can be provisioned with throughput independently



Source: https://learn.microsoft.com/en-us/azure/cosmos-db/request-units

Database-level Throughput Provisioning

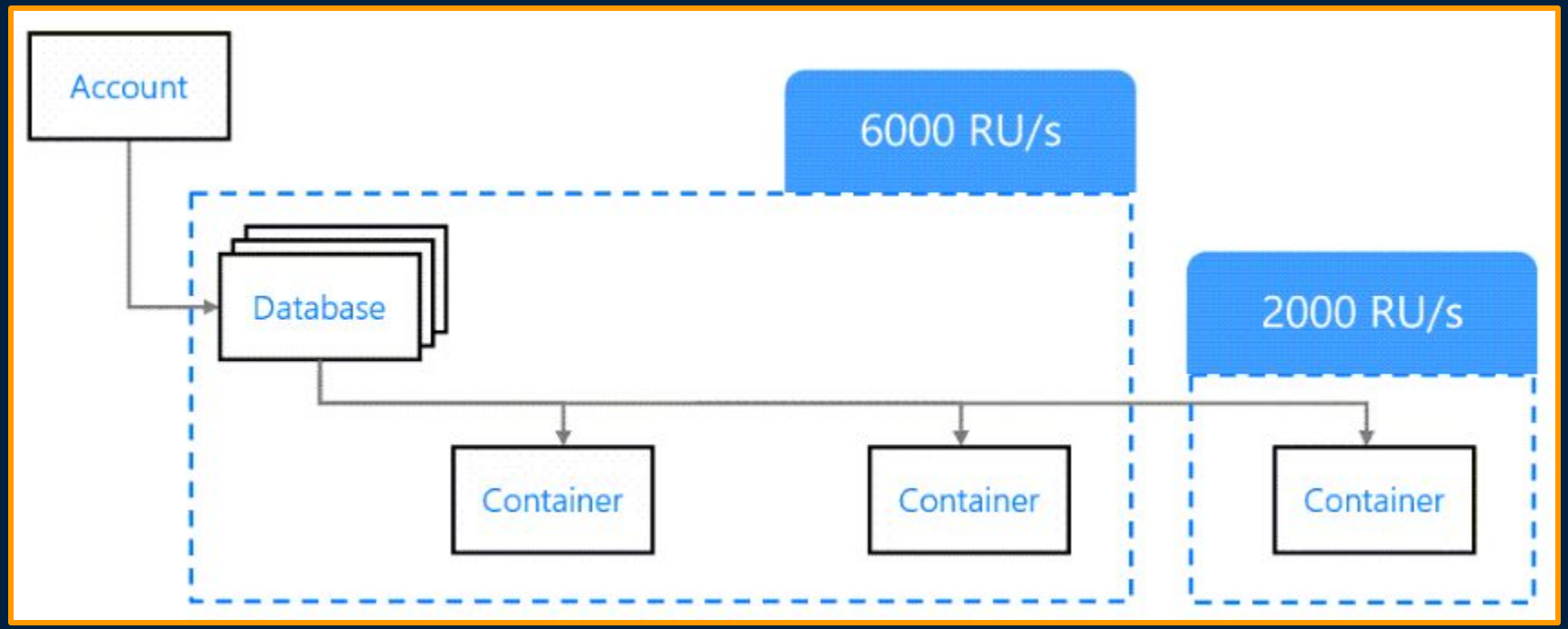
- Containers share the same throughput
  - Can result in unpredictable performance in a specific container



Source: https://learn.microsoft.com/en-us/azure/cosmos-db/request-units

Mixed Throughput Provisioning

- Specific containers within a database can be provisioned separately



Source: https://learn.microsoft.com/en-us/azure/cosmos-db/request-units

Request Units (RU)

Request Units are a rate-based currency

- Used to simplify costs of using physical resources
- Counting operations can help estimate capacity planning

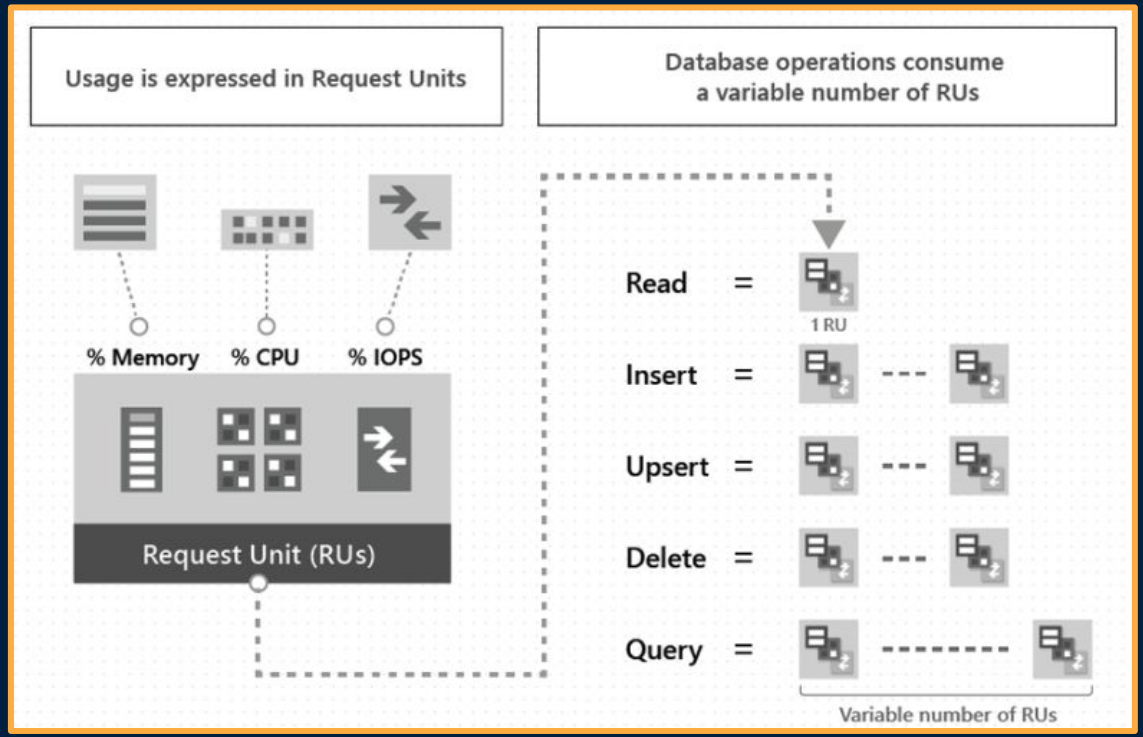
1 RU = 1 point read of 1kb

20 RU = 20 kb point read

100 reads/sec x 4 RU/s = ***400 RU/s***

200 writes/sec x 8 RU/s = ***1600 RU/s***

Capacity planning result = ***2000 RU/s***



Source: https://learn.microsoft.com/en-us/azure/cosmos-db/request-units

Storage and Time-to-live

SSD storage is also billed per GB per month

A document's ***Time-to-live (TTL)*** can be set to purge documents over time

- Reduces overall storage used
- The container JSON property ***DefaultTimeToLive*** can be configured
    - If configured, the contained items property ***ttl*** can also be configured

Container.DefaultTimeToLive	Item.ttl	Expiration in seconds
`1000`	*null*	`1000`
`1000`	`-1`	*This item will never expire*
`1000`	`2000`	`2000`

Free Options

Azure Cosmos DB Emulator

- **Downloadable version** that mimics the cloud service
  - Can develop and test code locally without an Azure subscription

Free Tier

- Provides the first **1000 RU/s** and **25 GB** of storage for free
- Further throughput and storage is billed at regular price

Azure Free Account

- Free Azure resources and credits for a limited time
- **12 months** of additional **400 RU/s** and **25 GB** of storage for free
- **30 days** to spend up to **$200 in Azure credits**

Cosmos DB Price models

- 3 year reserved capacity; 5,000 RU/sec; 25 GB transactional storage, 2 copies of periodic backup storage ($0.250 Per GB/month)
  - Single Region Monthly: $225.25
  - Multiple Region Monthly: $415.06
- 3 year reserved capacity; 50,000 RU/sec; 25 GB transactional storage, 2 copies of periodic backup storage
  - Single Region Monthly: $2,196.25
  - Multiple Region Monthly: $4094.25

Source: https://learn.microsoft.com/en-us/azure/cosmos-db/how-pricing-works

Azure Cosmos DB vs. Google Cloud Firestore

- Azure Cosmos DB is a highly flexible NoSQL database with numerous APIs and compatibility modes, whereas Google Cloud Firestore has only one.

- Azure Cosmos DB offers a simplified SQL-like query language that supports filters, sorts and aggregates. Google Cloud Firestore supports many of the same query patterns but cannot perform aggregations at the database level.

- Azure Cosmos DB is much cheaper than Google Cloud Firestore for the same workload.

- If aggregating data across collections is a requirement, Cosmos DB will be a better choice. Cosmos is also better for cost savings.

Conclusion

- Azure Cosmos DB highly available, reliable, and high throughput database.
- Supports Multiple APIs: SQL, Cassandra, MongoDB, Gremlin, Azure Table
- Offers five well-defined consistency models.
- Choosing the right API model and the right level of consistency for your application will help improve the performance of the queries.

Objectives

- Describe Cosmos DB and its key features
- Explain the 5 consistency models
- Discuss the use cases Cosmos DB
- Discuss the native API and the compatibility APIs
- Discuss considerations when selecting those API
- Define the Cosmos DB resource model for each API
- Describe partitions and partition key optimization
- Describe the Cosmos DB throughput provisioning
- Understand container, database and mixed level throughput provisioning
- Explain the storage and time-to-live
- Explain the Request Units and how that impacts pricing
- Understand the Cosmos DB pricing models
- Compare Azure Cosmos DB with other Cloud Databases


UBC
THE UNIVERSITY OF BRITISH COLUMBIA