Microsoft

GPS Data/AI Strategy FY23

Delivered by CSA Team

Franck Gaillard
Cloud Solution Architect
Data AI
frgail@microsoft.com

Narjes Majdoub
Cloud Solution Architect
Data AI
nmajdoub@microsoft.com

Ali Bouhaddou
Cloud Solution Architect
Data Analytics
albouhad@microsoft.com

Frederic Gisbert
Cloud Solution Architect
Data Analytics
frgisber@microsoft.com

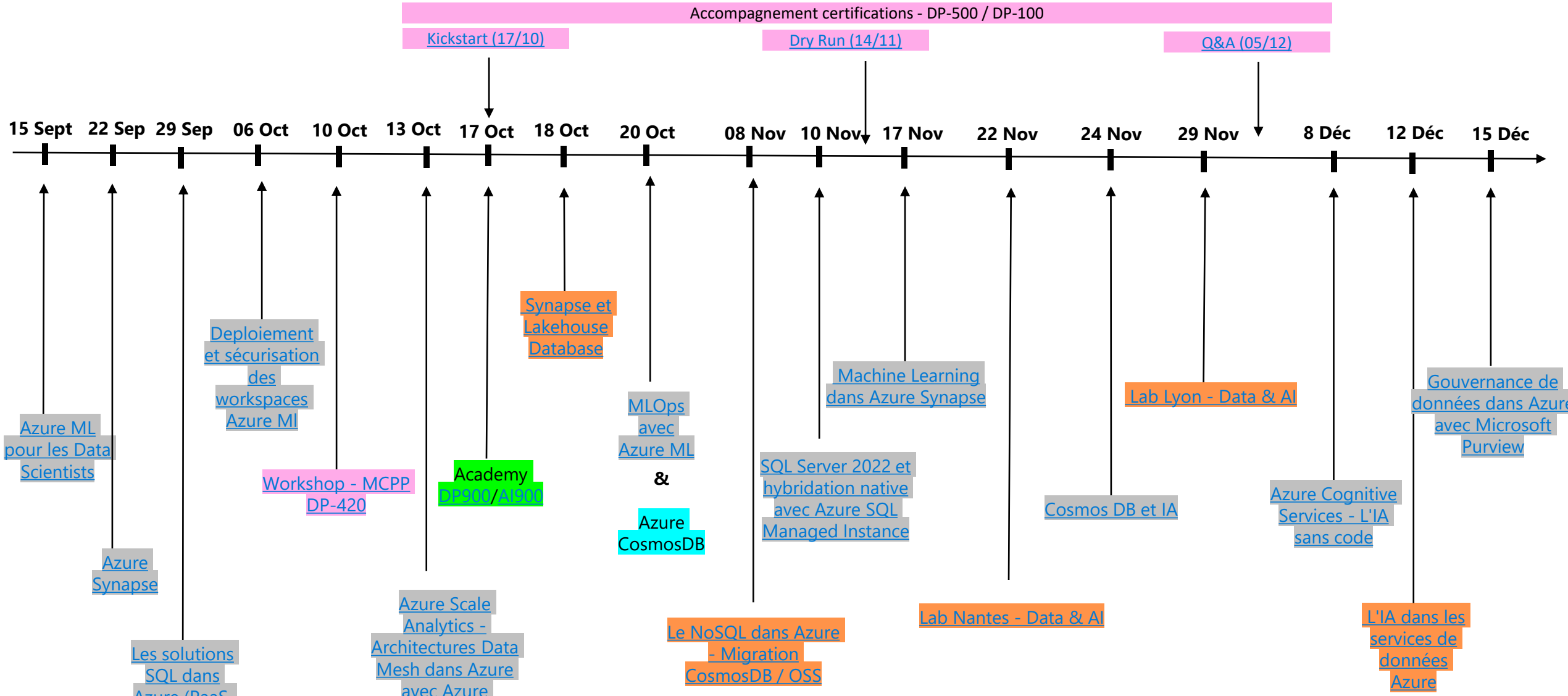
Azure Data & AI technical intensity plan

- From June 2022 to June 2023
- Focus on "**Azure Data & AI**" tech intensity

- Many content, from **L100** Beginner to **L400** Expert level:
  - **Academy** L100
  - **Webinar** L200/L300
  - **Workshop** L300/L400
  - **Certification kickstart** L300/L400
  - **Openhack / Microhack** L400

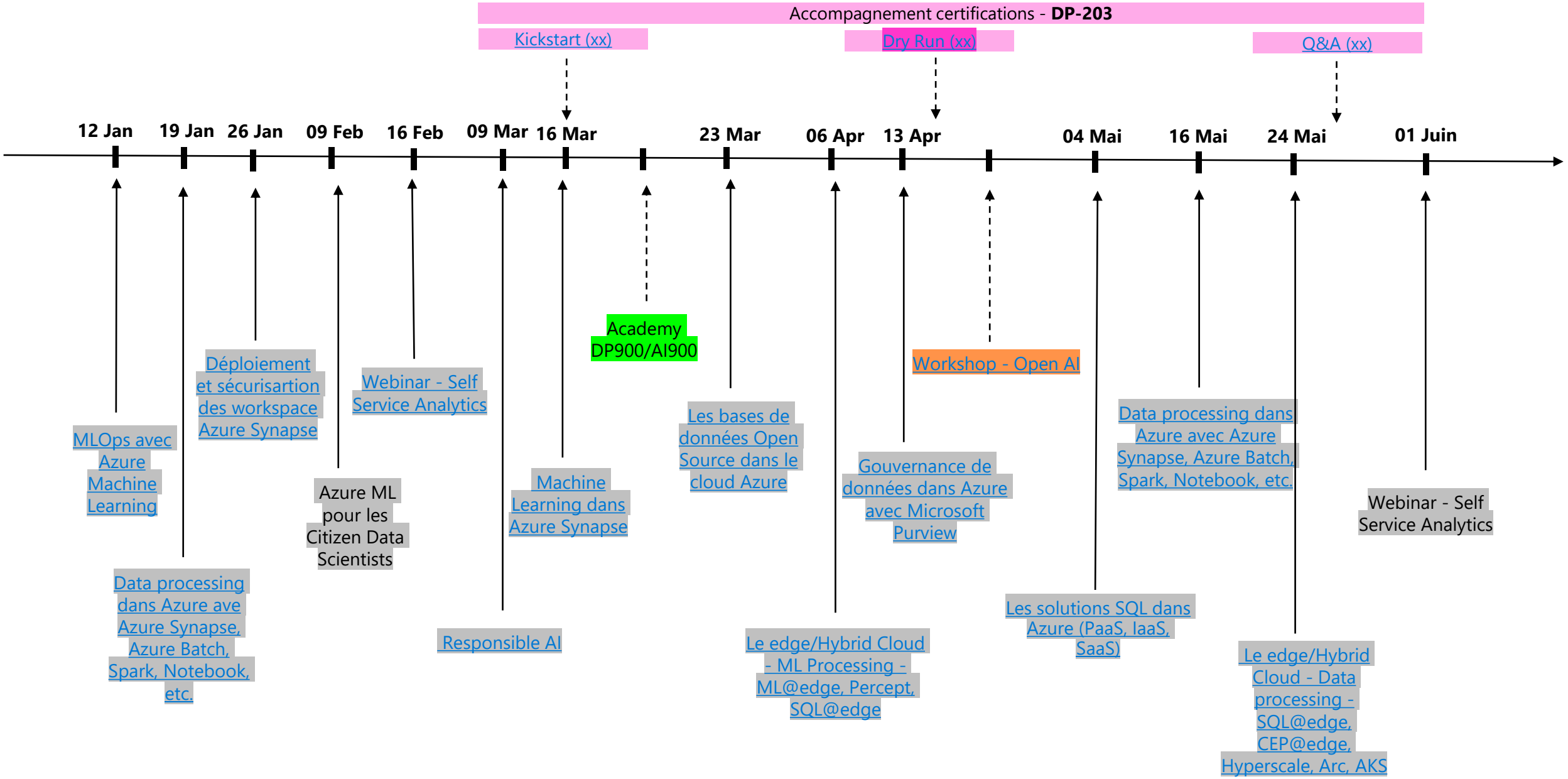
Data & AI events timeline – H1

Webinar/Academy- L 200/300
Workshop/ Openhack/ Certifications – L 300/400



Data & AI events timeline – H2

Webinar/Academy- L 200/300
Workshop/ Openhack/ Certifications – L 300/400



Liste des évènements de type Webinar 2H

Event Webinar (Les jeudis de la Data & AI) - L200/300	Date	Duration (min)	Link
Azure Machine Learning pour les Data Scientists	15/09/2022	120	https://msevents.microsoft.com/event?id=2454281594
Azure Synapse	22/09/2022	120	https://msevents.microsoft.com/event?id=857781749
Les solutions SQL dans Azure (PaaS, IaaS, SaaS)	29/09/2022	120	https://msevents.microsoft.com/event?id=502366997
Déploiement et sécurisation des workspaces Azure Machine learning	06/10/2022	120	https://msevents.microsoft.com/event?id=1505714138
Azure Scale Analytics - Architectures Data Mesh dans Azure avec Azure Synapse, Microsoft Purview et Azure Data Share	13/10/2022	120	https://msevents.microsoft.com/event?id=139685175
MLOps avec Azure Machine Learning	20/10/2022	120	https://msevents.microsoft.com/event?id=1245885767
SQL Server 2022 et hybridation native avec Azure SQL Managed Instance	10/11/2022	120	https://msevents.microsoft.com/event?id=145826476
Machine Learning dans Azure Synapse Analytics	17/11/2022	120	https://msevents.microsoft.com/event?id=3637723312
Azure Cosmos DB et IA	24/11/2022	120	https://msevents.microsoft.com/event?id=2646013445
Azure et les Services Cognitifs	08/12/2022	120	https://msevents.microsoft.com/event?id=3772037220
La gouvernance de données dans Azure avec Microsoft Purview	15/12/2022	120	https://msevents.microsoft.com/event?id=1499560981
MLOps avec Azure Machine Learning	12/01/2023	120	https://msevents.microsoft.com/event?id=4115194515
Data processing dans Azure ave Azure Synapse, Azure Batch, Spark, Notebook, etc.	19/01/2023	120	https://msevents.microsoft.com/event?id=1537241181
Déploiement et sécurisation des workspace Azure Synapse	26/01/2023	120	https://msevents.microsoft.com/event?id=1806467748
Azure Machine Learning pour les Citizen Data Scientists	09/02/2023	120	En cours
PowerBI - Self Service Analytics	16/02/2023	120	https://msevents.microsoft.com/event?id=1401519679
L'IA responsable avec Azure machine learning	09/03/2023	120	https://msevents.microsoft.com/event?id=2072953112
Machine Learning dans Azure Synapse Analytics	16/03/2023	120	https://msevents.microsoft.com/event?id=3413014857
Les bases de données Open Source dans le cloud Azure	23/03/2023	120	https://msevents.microsoft.com/event?id=2727487131
Hybridation des services de Machine Learning Azure	06/04/2023	120	https://msevents.microsoft.com/event?id=1624914222
La gouvernance de données dans Azure avec Microsoft Purview	13/04/2023	120	https://msevents.microsoft.com/event?id=3909342839
Les solutions SQL dans Azure (PaaS, IaaS, SaaS)	04/05/2023	120	https://msevents.microsoft.com/event?id=1162207895
Data processing dans Azure ave Azure Synapse, Azure Batch, Spark, Notebook, etc.	16/05/2023	120	https://msevents.microsoft.com/event?id=3517068442
Hybridation des services de données Azure	24/05/2023	120	https://msevents.microsoft.com/event?id=2996507398
Self Service Analytics	01/06/2023	120	En cours

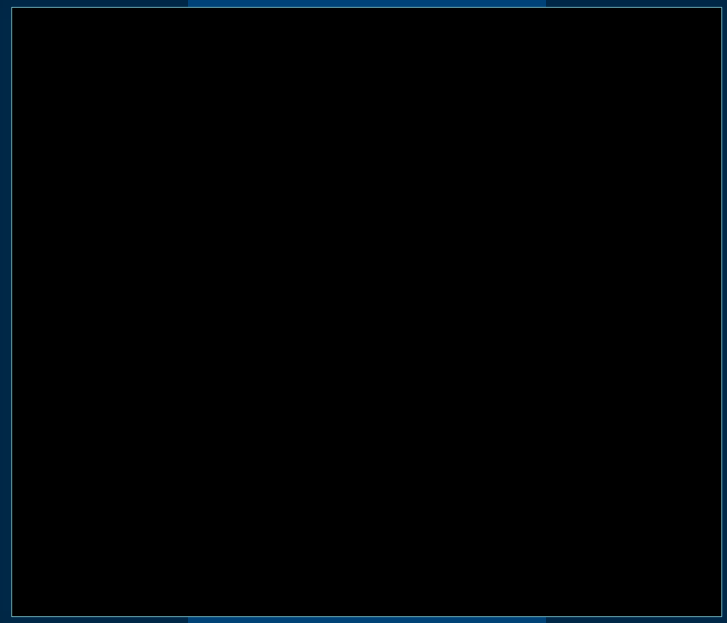
Liste des évènements de type Workshop/Prepa Cert/Academy

Event Workshop L300/400	Date	Duration (min)	Link
Synapse et Lakehouse Database	18/10/2022	120	https://forms.office.com/Pages/ResponsePage.aspx?id=v4j5cvGGr0GRqy180BHbR3zwJTO3s11AuaqpNnBbrwdURE1RMVgwTDNISTE1TDFYSDVLR0cyS1kwWS4u
Le NoSQL dans Azure - Migration CosmosDB / OSS	08/11/2022	120	https://forms.office.com/Pages/ResponsePage.aspx?id=v4j5cvGGr0GRqy180BHbR3zwJTO3s11AuaqpNnBbrwdURE1RMVgwTDNISTE1TDFYSDVLR0cyS1kwWS4u
Lab Lyon - Data & AI	22/11/2022	240	https://forms.office.com/Pages/ResponsePage.aspx?id=v4j5cvGGr0GRqy180BHbR3zwJTO3s11AuaqpNnBbrwdUMlZZOURET0RSWjcyTERYRkJGTlFFUjJaUi4u
Lab Nantes - Data & AI	29/11/2022	240	https://forms.office.com/Pages/ResponsePage.aspx?id=v4j5cvGGr0GRqy180BHbR3zwJTO3s11AuaqpNnBbrwdUMlZZOURET0RSWjcyTERYRkJGTlFFUjJaUi4u
L'IA dans les services de données Azure	12/12/2022	120	https://forms.office.com/Pages/ResponsePage.aspx?id=v4j5cvGGr0GRqy180BHbR3zwJTO3s11AuaqpNnBbrwdURE1RMVgwTDNISTE1TDFYSDVLR0cyS1kwWS4u
Open AI	H2	120	https://forms.office.com/Pages/ResponsePage.aspx?id=v4j5cvGGr0GRqy180BHbR3zwJTO3s11AuaqpNnBbrwdURE1RMVgwTDNISTE1TDFYSDVLR0cyS1kwWS4u

Event Academy, kickstart certifications, workshop certifications	Date	Duration (min)	Link
MCPP - DP-420	10/10/2022	420	https://forms.office.com/Pages/ResponsePage.aspx?id=v4j5cvGGr0GRqy180BHbR3zwJTO3s11AuaqpNnBbrwdUMkJISlRKSU1RRFA0OVgzSFdTSTY0RE9WQy4u
Micro Hack CosmosDB	20/10/2022	420	H1 - Inscriptions PTA
Academy DP900	17-21/10/2022	300	https://msevents.microsoft.com/event?id=3250818161
Academy AI900	17-21/10/2022	300	https://msevents.microsoft.com/event?id=2717528090
Kickstart DP-500	17/10/2022	60	https://forms.office.com/Pages/ResponsePage.aspx?id=v4j5cvGGr0GRqy180BHbR3zwJTO3s11AuaqpNnBbrwdUNEk3WFQ1TEdNNTQ2Uk85V0cxQzM3TE9ZRS4u
Dry Run DP-500	14/11/2022	120	https://forms.office.com/Pages/ResponsePage.aspx?id=v4j5cvGGr0GRqy180BHbR3zwJTO3s11AuaqpNnBbrwdUNEk3WFQ1TEdNNTQ2Uk85V0cxQzM3TE9ZRS4u
Q&A DP-500	05/12/2022	90	https://forms.office.com/Pages/ResponsePage.aspx?id=v4j5cvGGr0GRqy180BHbR3zwJTO3s11AuaqpNnBbrwdUNEk3WFQ1TEdNNTQ2Uk85V0cxQzM3TE9ZRS4u
Kickstart DP-100	17/10/2022	60	https://forms.office.com/Pages/ResponsePage.aspx?id=v4j5cvGGr0GRqy180BHbR3zwJTO3s11AuaqpNnBbrwdUNDAxV0hSN0FHM1YzUzI3OUNMUFYxSkRIMi4u
Dry Run DP-100	14/11/2022	120	https://forms.office.com/Pages/ResponsePage.aspx?id=v4j5cvGGr0GRqy180BHbR3zwJTO3s11AuaqpNnBbrwdUNDAxV0hSN0FHM1YzUzI3OUNMUFYxSkRIMi4u
Q&A DP-100	05/12/2022	90	https://forms.office.com/Pages/ResponsePage.aspx?id=v4j5cvGGr0GRqy180BHbR3zwJTO3s11AuaqpNnBbrwdUNDAxV0hSN0FHM1YzUzI3OUNMUFYxSkRIMi4u
Kickstart DP-203	17/10/2022	60	https://forms.office.com/Pages/ResponsePage.aspx?id=v4j5cvGGr0GRqy180BHbR3zwJTO3s11AuaqpNnBbrwdUOVFWOUVCNFcyQk5SVjFBUFczNktCUFpLMi4u
Dry Run DP-203	14/11/2022	120	https://forms.office.com/Pages/ResponsePage.aspx?id=v4j5cvGGr0GRqy180BHbR3zwJTO3s11AuaqpNnBbrwdUOVFWOUVCNFcyQk5SVjFBUFczNktCUFpLMi4u
Q&A DP-203	05/12/2022	90	https://forms.office.com/Pages/ResponsePage.aspx?id=v4j5cvGGr0GRqy180BHbR3zwJTO3s11AuaqpNnBbrwdUOVFWOUVCNFcyQk5SVjFBUFczNktCUFpLMi4u

Microsoft

Kick Start Certification DP 420 Designing and Implementing Cloud-Native Applications Using Microsoft Azure Cosmos DB



10/10/2022

Speaker info

Frederic Gisbert

Cloud Solutions Architect
Frederic.gisbert@microsoft.com

Ali Bouhaddou

Cloud Solutions Architect
albouhad@microsoft.com

Emmanuel Deletang

Global Black Belt Cosmos DB
edeletang@microsoft.com

Agenda

Kick start

Dry Run

QnA

9h30-10h How to prepare Microsoft Certified Specialist Professional

10h-12h Kick start

1. Reviewing Azure Cosmos DB
2. Partitioning
3. Querying
4. Programming

12h-13h Break

13h-14h Kick start

1. Troubleshooting
2. Modeling & Planning

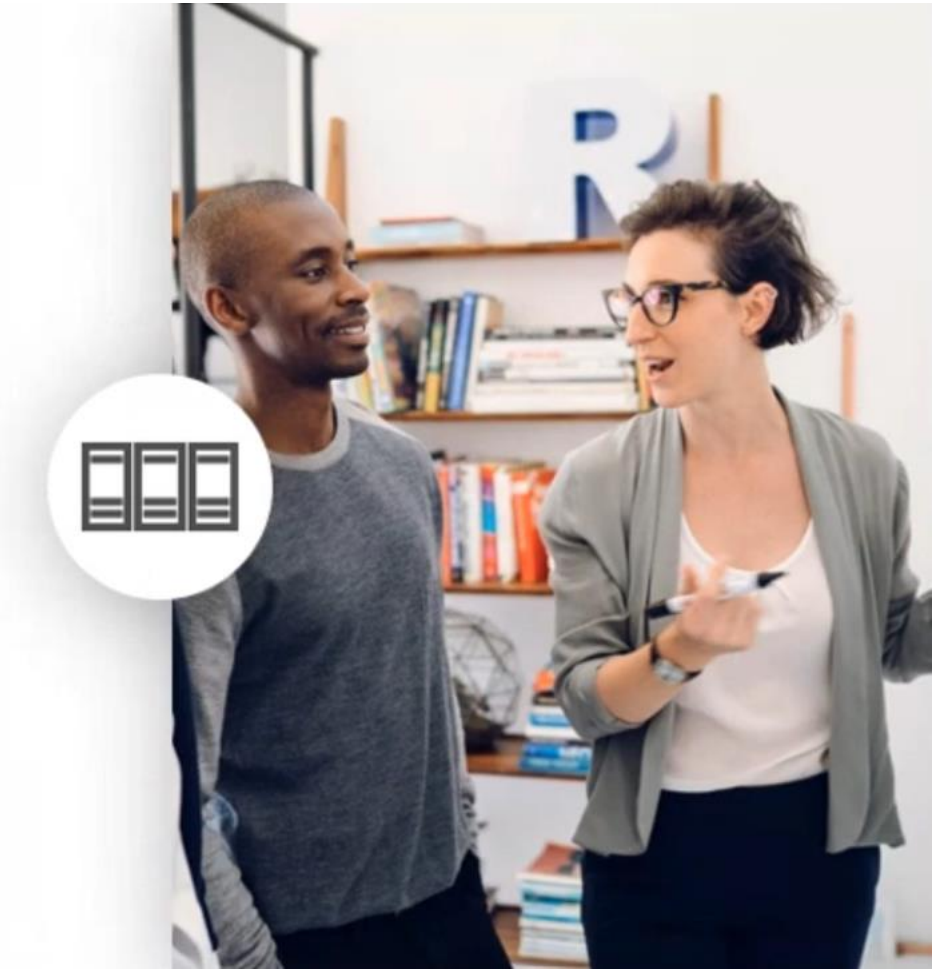
14h-16h Dry Run

16h-17h QnA

Exam overview and objective domain DP-420

Become Microsoft Certified

Microsoft Certified: Azure Cosmos DB Developer Specialty



Cosmos DB Developer

Audience profile

Responsibilities for this role include designing and implementing data models and data distribution, loading data into an Azure Cosmos DB database, and optimizing and maintaining the solution.
These professionals integrate the solution with other Azure services. They also design, implement, and monitor solutions that consider security, availability, resilience, and performance requirements.

A candidate for this certification should be proficient at developing applications that use the Azure **Cosmos DB .NET SDK** for SQL API. They should be able to write efficient Azure **Cosmos DB SQL queries** and be able to create appropriate **index policies**. They should have experience creating **server-side objects** with JavaScript. Additionally, they should be familiar with **provisioning and managing resources** in Azure. They should be able to interpret JSON, read C# or Java code, and use PowerShell.

Microsoft Certified: Azure Cosmos DB Developer Specialty

Take one exam

CERTIFICATION EXAM

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

Earn the certification

Earn the certification

ASSOCIATE CERTIFICATION

Microsoft Certified: Azure Cosmos DB Developer Specialty

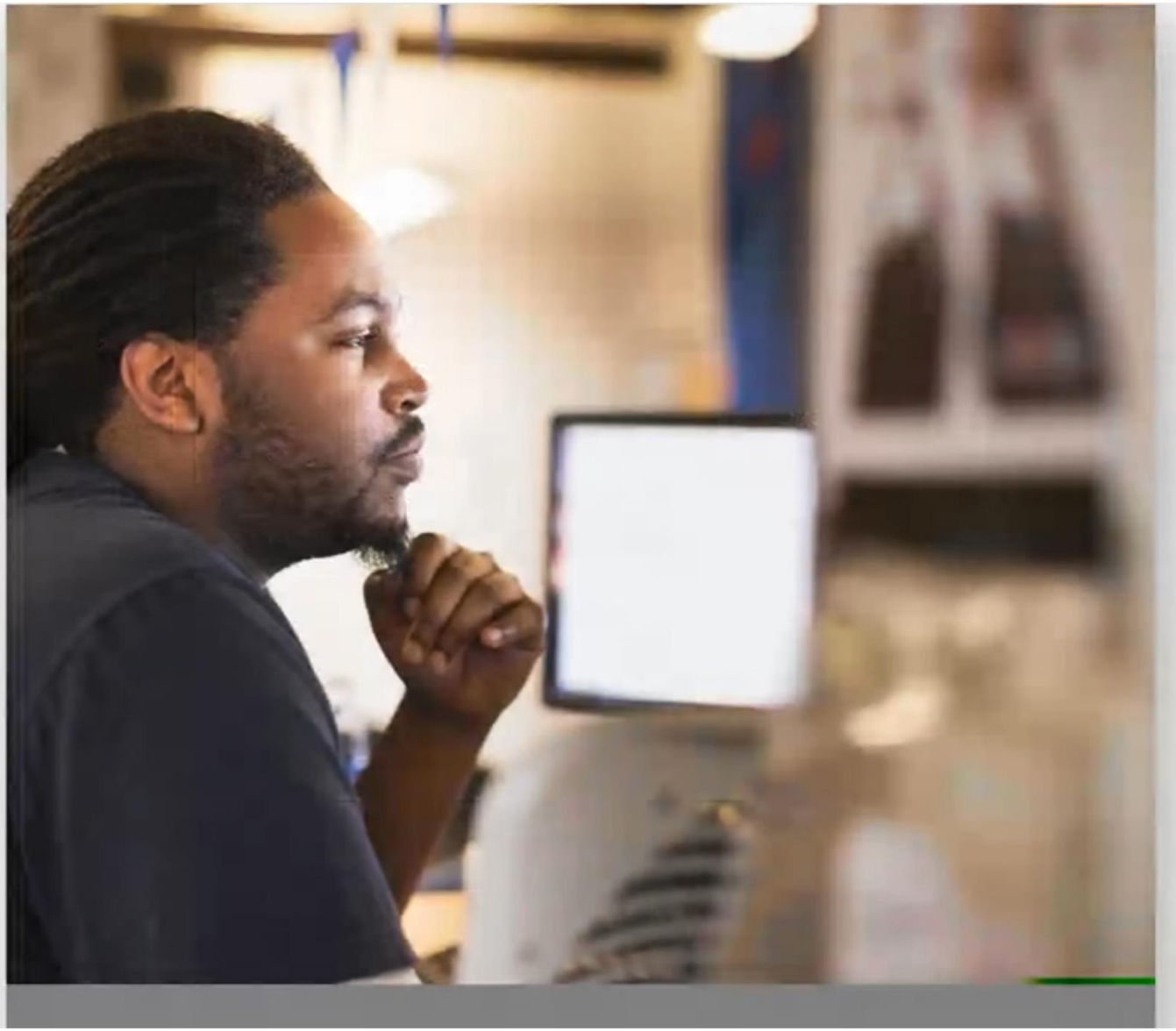
Exam objectives

Skills Measured	Weights
Design and implement data models	35–40%
Design and implement data distribution	5-10%
Integrate an Azure Cosmos DB solution	5-10%
Optimize an Azure Cosmos DB solution	15-20%
Maintain an Azure Cosmos DB solution	25-30%

- Percentages indicate the relative weight of each area on the exam
- The higher the percentage, the more questions you are likely to see in that area

Passing score is 700

How to prepare



Training and certification

Microlearning

- Step-by-step, bite-sized tutorials by product, skill level, and job role
- Hands-on learning with interactive browser-based scripting environments
- Immediate assessment via knowledge quizzes

❑ Microsoft Learn

Online Courses

- Self-paced, structured courses with the ability to track progress and maintain a transcript
- Hands-on learning with hands-on labs
- Knowledge assessments

❑ Microsoft Learning Partners
❑ LinkedIn Learning
❑ Pluralsight

Instructor-led Training

- Blended learning, in-person, and online to suit learning needs
- Delivered by Microsoft Certified Trainers
- Deep technical training to give yo the technical expertise

❑ Microsoft Learning Partners

Certification

- New role-based certifications, including varying levels (Fundamentals, Associate, Expert)
- Industry-recognized technical certifications
- Share "credentials" with your professional network with an online badge

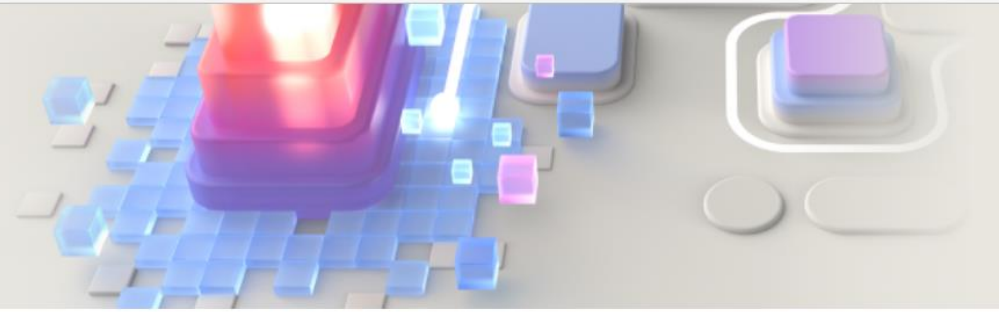
❑ aka.ms/MSCert
❑ Microsoft Learning Partners

https://learn.microsoft.com/en-us/certifications/browse/?roles=developer&terms=cosmos

Not syncing

Exams

Learn new skills to boost your productivity and enable your organization to accomplish more with Microsoft Certifications.



Filter

Clear all

Products

- [ ] Azure

Roles

- [ ] Data Engineer
- [x] Developer
- [ ] Solution Architect

Levels

- [ ] Intermediate

Certification Types

- [ ] Specialty

Types

- [ ] Certification
- [ ] Exam

cosmos

Search

Developer ×

2 results for "cosmos"

CERTIFICATION

Microsoft Certified: Azure Cosmos DB Developer Specialty

ExamDP-420

Azure • Developer • Intermediate

Save

EXAM

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

Azure • Developer • Intermediate

Save

https://learn.microsoft.com/en-us/certifications/azure-cosmos-db-developer-specialty/

Two ways to prepare

Online - Free | Instructor-led - Paid

LEARNING PATH

Get started with Azure Cosmos DB SQL API

0 of 2 modules completed

Intermediate • Data Engineer • Azure

Start

Save

LEARNING PATH

Plan and implement Azure Cosmos DB SQL API

0 of 3 modules completed

Intermediate • Developer • Cosmos DB

Save

LEARNING PATH

Connect to Azure Cosmos DB SQL API with the SDK

0 of 2 modules completed

Intermediate • Developer • Cosmos DB

Save

LEARNING PATH

Access and manage data with the Azure Cosmos DB SQL API SDKs

0 of 3 modules completed

Intermediate • Developer • Cosmos DB

Save

LEARNING PATH

Execute queries in Azure Cosmos DB SQL API

0 of 2 modules completed

Intermediate • Developer • Cosmos DB

Save

LEARNING PATH

Define and implement an indexing strategy for Azure Cosmos DB SQL API

0 of 2 modules completed

Intermediate • Developer • Cosmos DB

Save

LEARNING PATH

Integrate Azure Cosmos DB SQL API with Azure services

0 of 3 modules completed

Intermediate • Developer • Cosmos DB

Save

LEARNING PATH

Implement a data modeling and partitioning strategy for Azure Cosmos DB SQL API

0 of 2 modules completed

Intermediate • Data Engineer • Azure

Save

LEARNING PATH

Design and implement a replication strategy for Azure Cosmos DB SQL API

0 of 3 modules completed

Intermediate • Developer • Cosmos DB

Save

LEARNING PATH

Optimize query and operation performance in Azure Cosmos DB SQL API

0 of 3 modules completed

Intermediate • Developer • Cosmos DB

Save

LEARNING PATH

Monitor and troubleshoot an Azure Cosmos DB SQL API solution

0 of 4 modules completed

Intermediate • Developer • Cosmos DB

Save

LEARNING PATH

Manage an Azure Cosmos DB SQL API solution using DevOps practices

0 of 2 modules completed

Intermediate • Developer • Cosmos DB

Save

LEARNING PATH

Create server-side programming constructs in Azure Cosmos DB SQL API

0 of 2 modules completed

Intermediate • Developer • Cosmos DB

Save

https://www.starttest.com/ITDVersions/22.0.0.0/ITDStart.aspx?SVC=9b7e8f8d-2d1a-4d9f-a77c-cc0ca3a842bc

Microsoft

Exam AA-001_Sandbox

AA-001_Sandbox

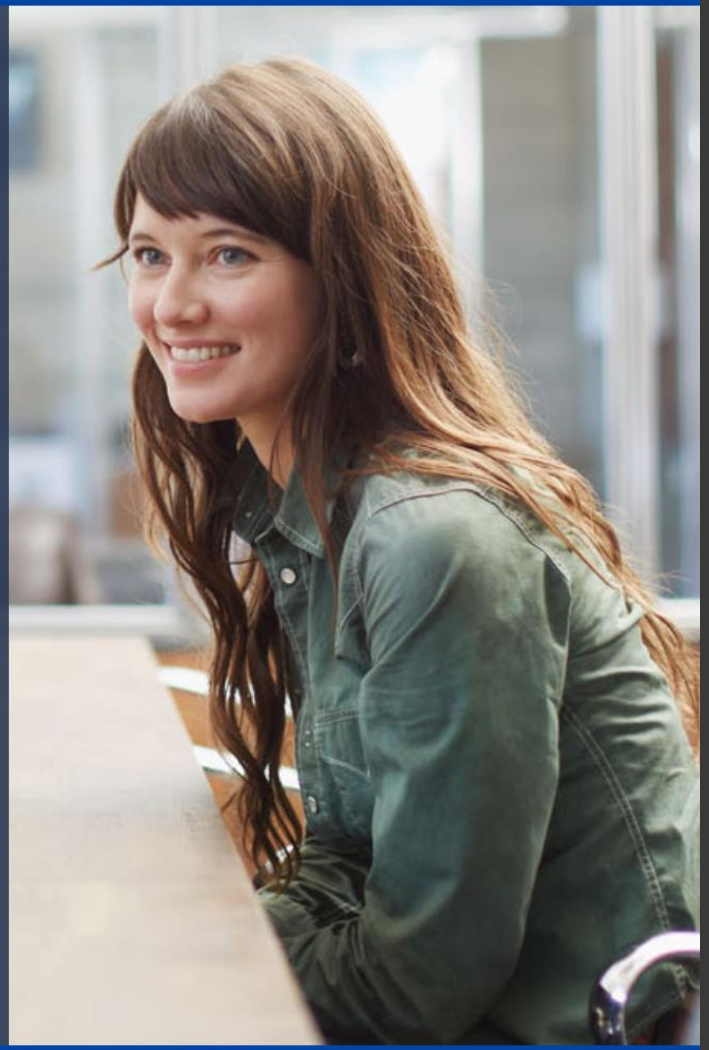
Welcome !

Maximum time for this session, including instructions, survey, and exam: 480 minutes

Number of exam questions: 10

Maximum time for exam: 480 minutes

Minimum score required to pass this exam: 700



Color scheme

Demo the exam experience by visiting our **Exam Sandbox**

Microsoft

Design and implement data models(35–40%)

Design and implement data distribution (5–10%

Integrate an Azure Cosmos DB solution (5–10%)

Optimize an Azure Cosmos DB solution (15–20%)

Maintain an Azure Cosmos DB solution (25–30%)

Design and implement a non-relational data model for Azure Cosmos DB for NoSQL
• 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 for NoSQL
• 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 Exam DP-420: Designing and Implementing Cloud-Native Applications Using Microsoft Azure Cosmos DB 10 • 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
• 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 SQL language for Azure Cosmos DB for NoSQL
• 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 Azure Cosmos DB for NoSQL 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 • Override default consistency by using query request options • Implement session consistency by using session tokens • 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
• Write, deploy, and call a stored procedure • Design stored procedures to work with multiple items transactionally • Implement and call triggers
• Implement a user-defined functionCosmos DB for NoSQL by using JavaScript

Microsoft

Design and implement data models(35–40%)

Design and implement data distribution (5–10%

Integrate an Azure Cosmos DB solution (5–10%)

Optimize an Azure Cosmos DB solution (15–20%)

Maintain an Azure Cosmos DB solution (25–30%)

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

Microsoft

Design and implement data models(35–40%)

Design and implement data distribution (5–10%

Integrate an Azure Cosmos DB solution (5–10%)

Optimize an Azure Cosmos DB solution (15–20%)

Maintain an Azure Cosmos DB solution (25–30%)

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
- 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

Microsoft

Design and implement data models(35–40%)

Design and implement data distribution (5–10%

Integrate an Azure Cosmos DB solution (5–10%)

Optimize an Azure Cosmos DB solution (15–20%)

Maintain an Azure Cosmos DB solution (25–30%)

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 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

Microsoft

Design and implement data models(35–40%)

Design and implement data distribution (5–10%

Integrate an Azure Cosmos DB solution (5–10%)

Optimize an Azure Cosmos DB solution (15–20%)

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

Modern apps face new challenges

Managing and syncing data distributed around the globe

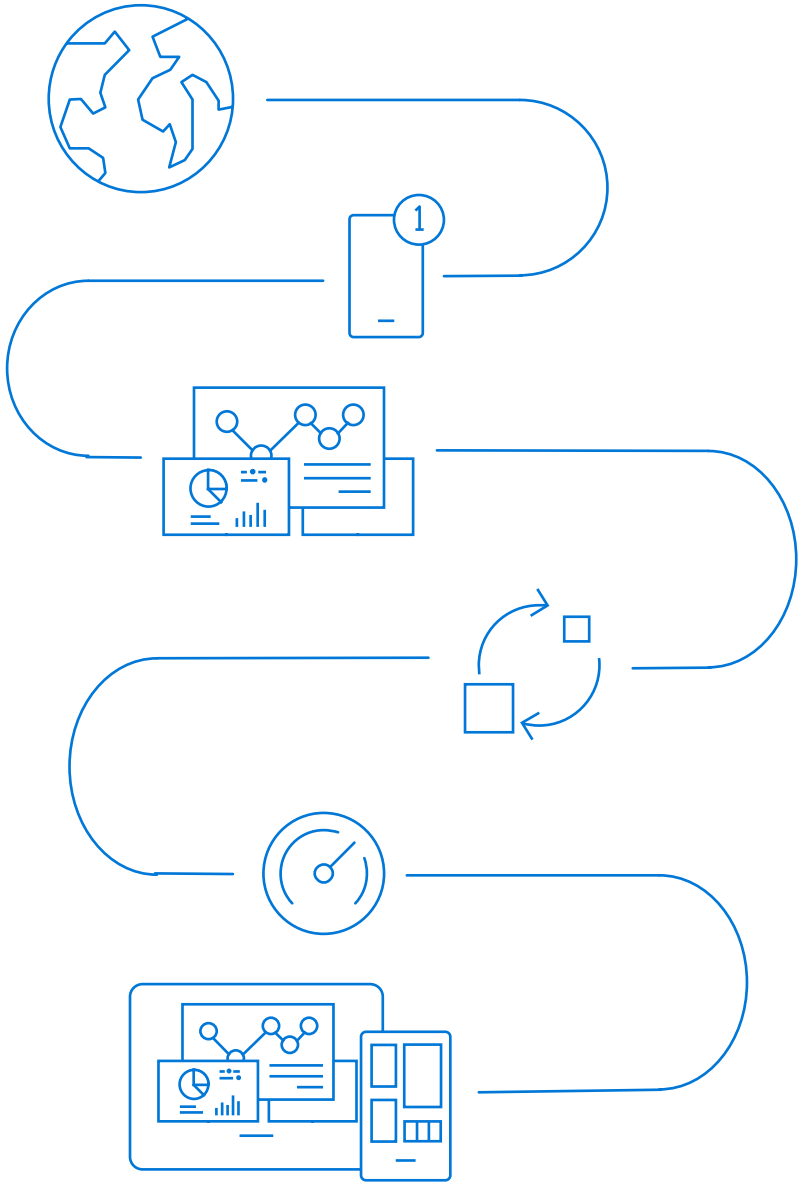
Delivering highly-responsive, real-time personalization

Processing and analyzing large, complex data

Scaling both throughput and storage based on global demand

Offering low-latency to global users

Modernizing existing apps and data



Azure Cosmos DB (1 of 3)

A fully managed NoSQL database for modern app development with SLA-backed speed and availability, automatic and instant scalability, and open source APIs for MongoDB, Cassandra, and other NoSQL engines.

Guaranteed speed at any scale

Gain unparalleled SLA-backed speed and throughput, fast global access, and instant elasticity.

Simplified application development

Build fast with open source APIs, multiple SDKs, schemaless data, and no-ETL analytics over operational data.

Mission-critical ready

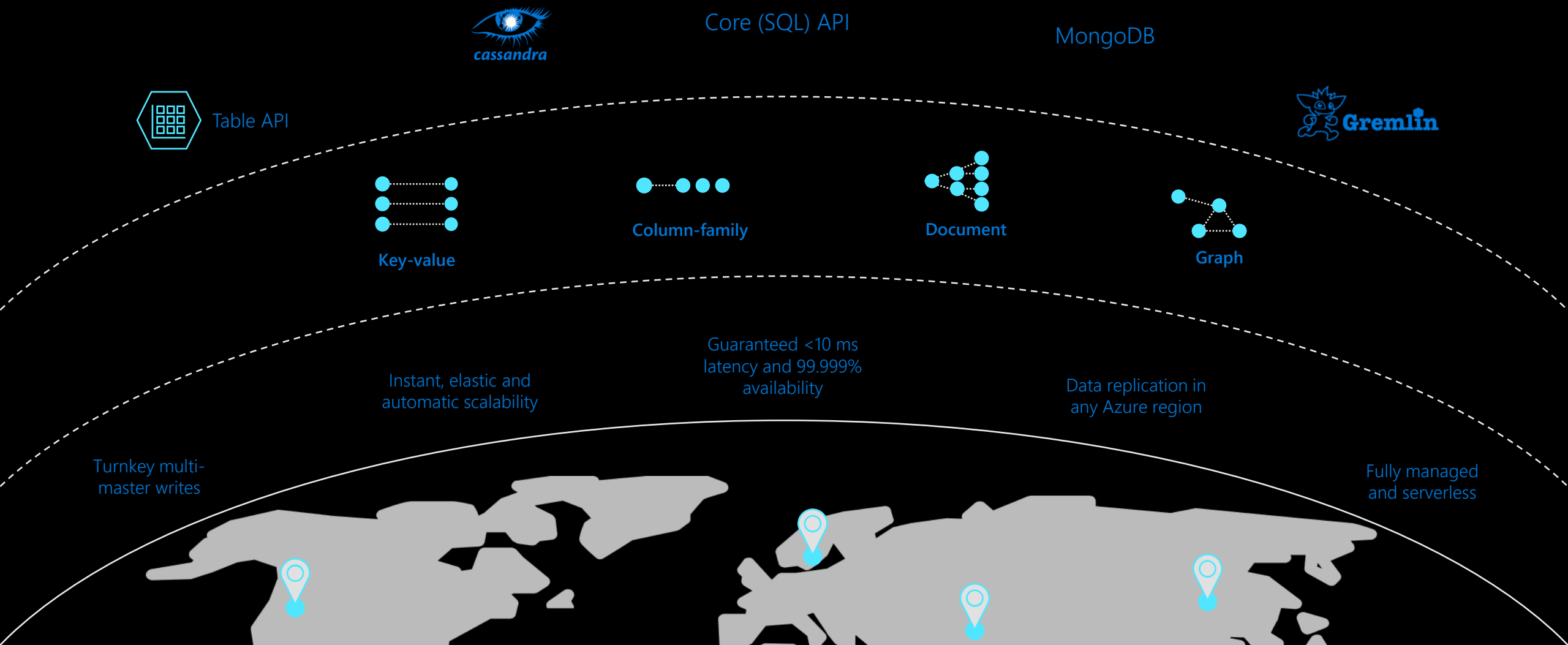
Guarantee business continuity, 99.999% availability, and enterprise-level security for every application.

Fully managed and cost-effective

End-to-end database management with serverless and automatic scaling matching your application and TCO needs.

AZURE COSMOS DB (2 of 3)

Fast NoSQL database with open APIs for any scale

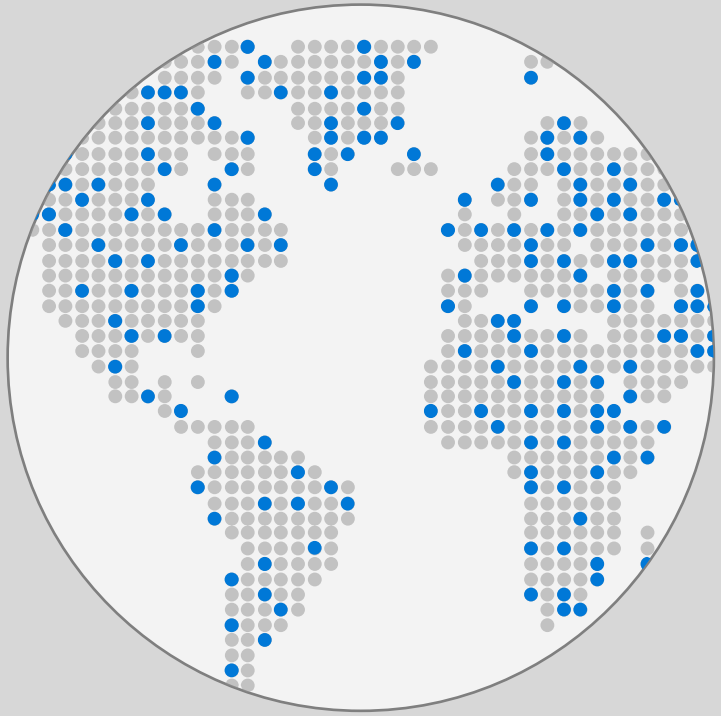


TURNKEY GLOBAL DISTRIBUTION (1 of 3)

PUT YOUR DATA WHERE YOUR USERS ARE IN MINUTES

Automatically replicate all your data around the world, and across more regions than AWS and Google Cloud Platform combined.

- Available in all Azure regions
- Manual and automatic failover
- Automatic & synchronous multi-region replication

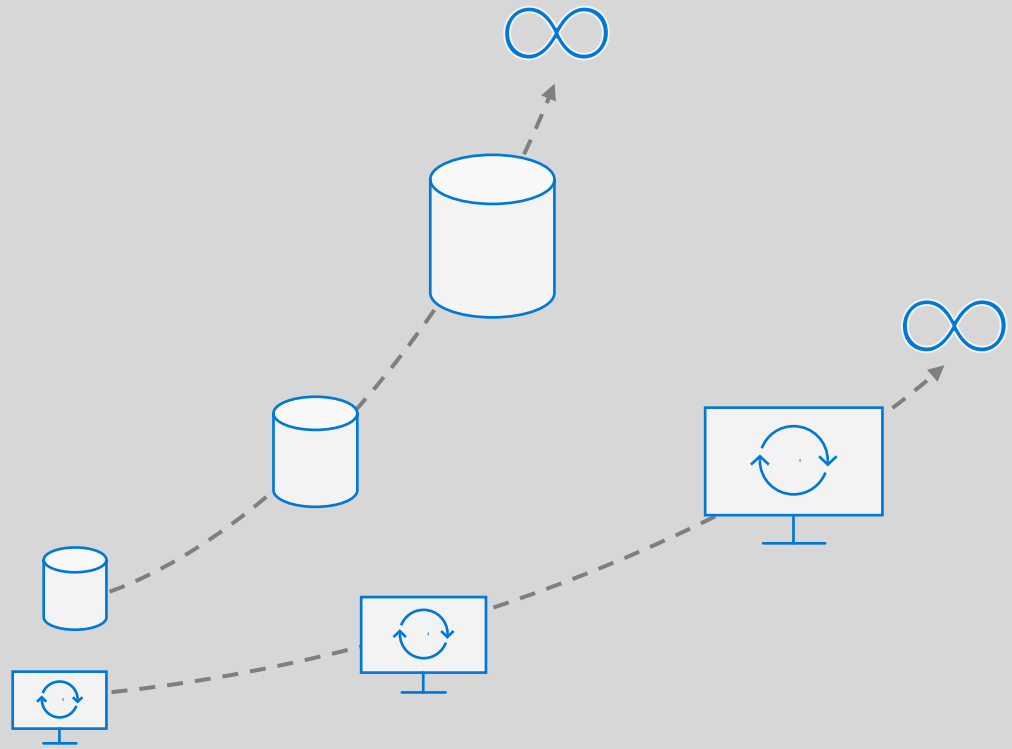


ELASTIC SCALE OUT OF STORAGE AND THROUGHPUT

SCALES AS YOUR APPS' NEEDS CHANGE

Independently and elastically scale storage and throughput across regions – even during unpredictable traffic bursts – with a database that adapts to your app's needs.

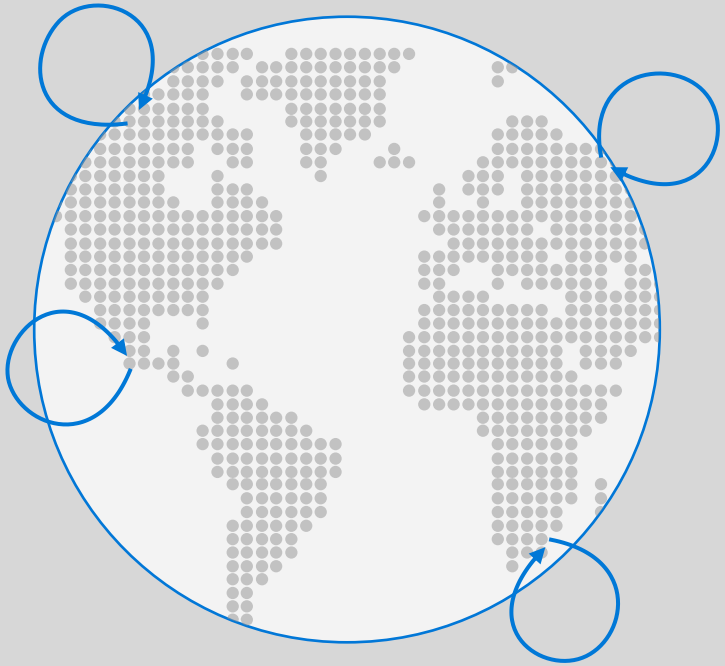
- Elastically scale throughput from 10 to 100s of millions of requests/sec across multiple regions
- Support for requests/sec for different workloads
- Pay only for the throughput and storage you need



GUARANTEED LOW LATENCY

PROVIDE USERS AROUND THE WORLD WITH FAST ACCESS TO DATA

Serve <10 millisecond read and write requests at the 99th percentile from the region nearest to users, while delivering data globally.



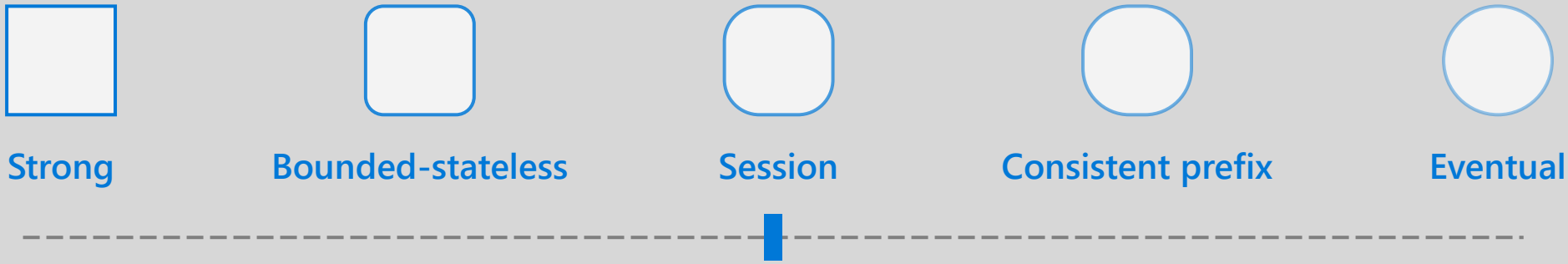
FIVE WELL-DEFINED CONSISTENCY MODELS (1 of 2)

CHOOSE THE BEST CONSISTENCY MODEL FOR YOUR APP

Offers five consistency models

Provides control over performance-consistency tradeoffs, backed by comprehensive SLAs.

An intuitive programming model offering low latency and high availability for your planet-scale app.



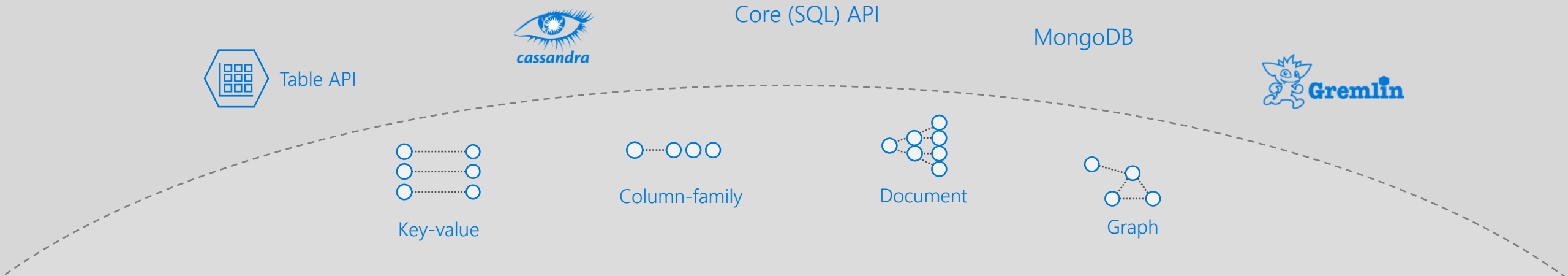
MULTIPLE DATA MODELS AND APIs

USE THE MODEL THAT FITS YOUR REQUIREMENTS, AND THE APIS, TOOLS, AND FRAMEWORKS YOU PREFER

Choose from multiple APIs to access and query data, including Core (SQL), MongoDB, Cassandra, Gremlin, and Table.

Use key-value, tabular, graph, and document data

Data is automatically indexed, with no schema or secondary indexes required.

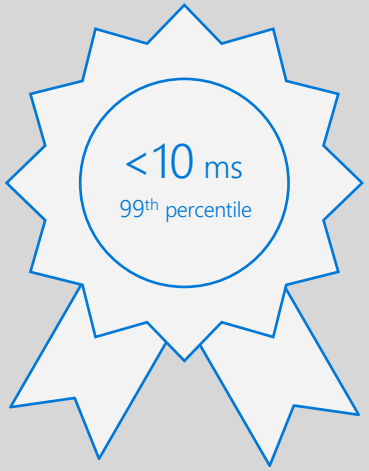


COMPREHENSIVE SLAs

RUN YOUR APP ON WORLD-CLASS INFRASTRUCTURE

Azure Cosmos DB is the only service with financially-backed SLAs for single-digit millisecond read and write latency at the 99th percentile, 99.999% high availability and guaranteed throughput and consistency

Latency



High Availability

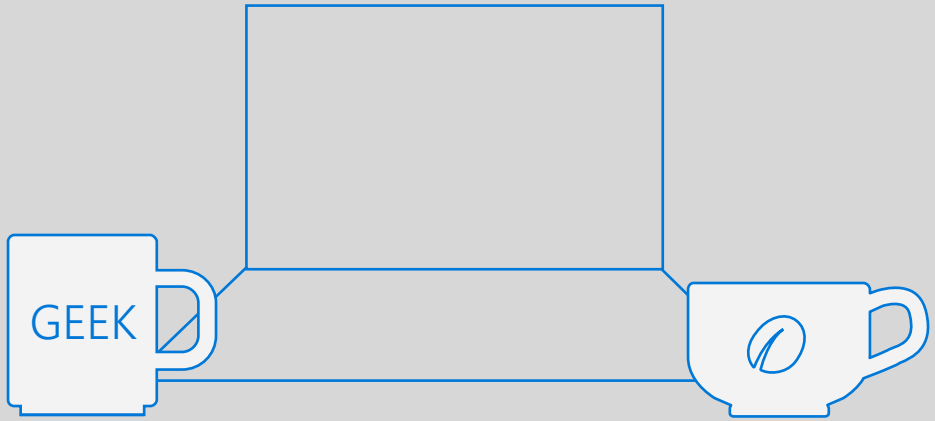
Throughput

Consistency

HANDLE ANY DATA WITH NO SCHEMA OR INDEXING REQUIRED (1 of 2)

Azure Cosmos DB's schema-less service automatically indexes all your data, regardless of the data model, to delivery blazing fast queries.

- Automatic index management
- Synchronous auto-indexing
- No schemas or secondary indices needed
- Works across every data model



Item	Color	Microwave safe	Liquid capacity	CPU	Memory	Storage
Geek mug	Graphite	Yes	16ox	???	???	???
Coffee Bean mug	Tan	No	12oz	???	???	???
Surface book	Gray	???	???	3.4 GHz Intel Skylake Core i7-6600U	16GB	1 TB SSD

TRUST YOUR DATA TO INDUSTRY-LEADING SECURITY & COMPLIANCE

Azure is the world's most trusted cloud, with more certifications than any other cloud provider.

- Enterprise grade security
- Encryption at Rest
- Encryption is enabled automatically by default
- Comprehensive Azure compliance certification

TOP 10 REASONS WHY CUSTOMERS USE AZURE COSMOS DB

The 1st and only database with global distribution turnkey capability

Deliver massive storage/throughput scalability database

Provides guaranteed single digit millisecond latency at 99th percentile worldwide

Natively supports different types of data at massive scale

Boasts 5 well-defined consistency models to pick the right consistency/latency/throughput tradeoff

Enables mission critical intelligent applications

Gives high flexibility to optimize for speed and cost

Tackles big data workloads with high availability and reliability

Provides multi-tenancy and enterprise-grade security

Naturally analytics-ready and perfect for event-driven architectures

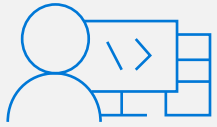
POWERING GLOBAL SOLUTIONS

Azure Cosmos DB was built to support modern app patterns and use cases.

It enables industry-leading organizations to unlock the value of data, and respond to global customers and changing business dynamics in real-time.

Data distributed and available globally

Puts data where your users are

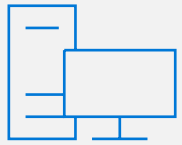


Build real-time customer experiences

Enable latency-sensitive personalization, bidding, and fraud detection.

Ideal for gaming, IoT & eCommerce

Predictable and fast service, even during traffic spikes



Simplified development with serverless architecture

Fully-managed event-driven micro-services with elastic computing power

Run Spark analytics over operational data

Accelerate insights from fast, global data

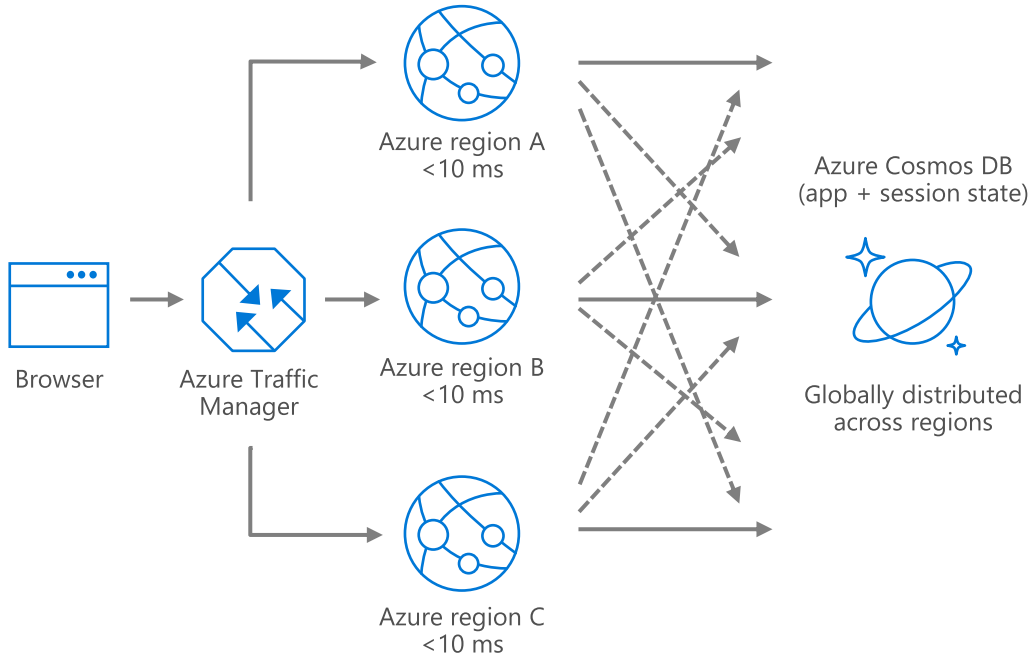
Lift and shift NoSQL data

Lift and shift MongoDB and Cassandra workloads

DATA DISTRIBUTED AND AVAILABLE GLOBALLY

Put your data where your users are to give real-time access and uninterrupted service to customers anywhere in the world.

- Turnkey global data replication across all Azure regions
- Guaranteed low-latency experience for global users
- Resiliency for high availability and disaster recovery



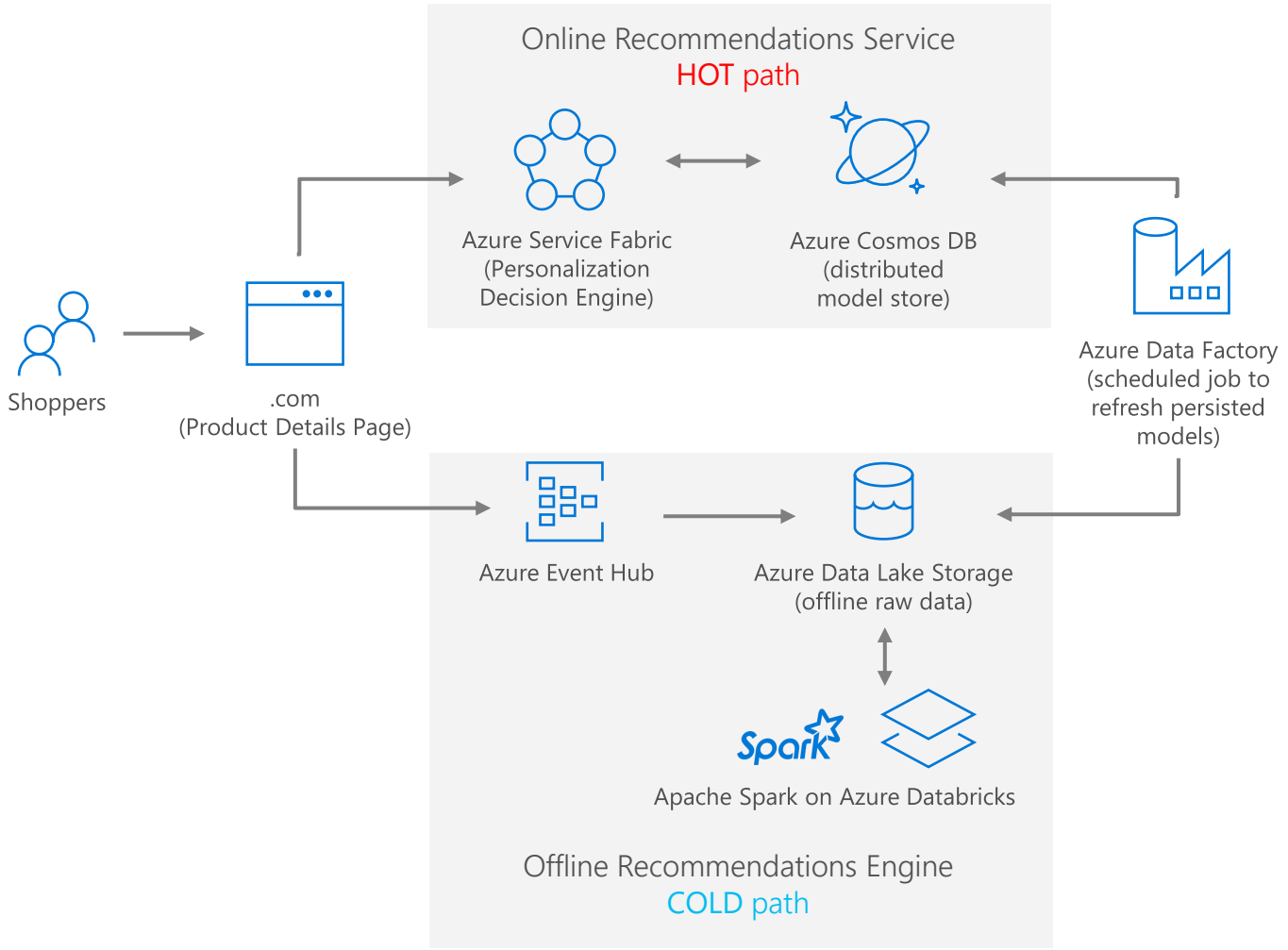
NEXT GAMES

Domino's

BUILD REAL-TIME CUSTOMER EXPERIENCES

Offer latency-sensitive applications with personalization, bidding, and fraud-detection.

- Machine learning models generate real-time recommendations across product catalogues
- Product analysis in milliseconds
- Low-latency ensures high app performance worldwide
- Tunable consistency models for rapid insight

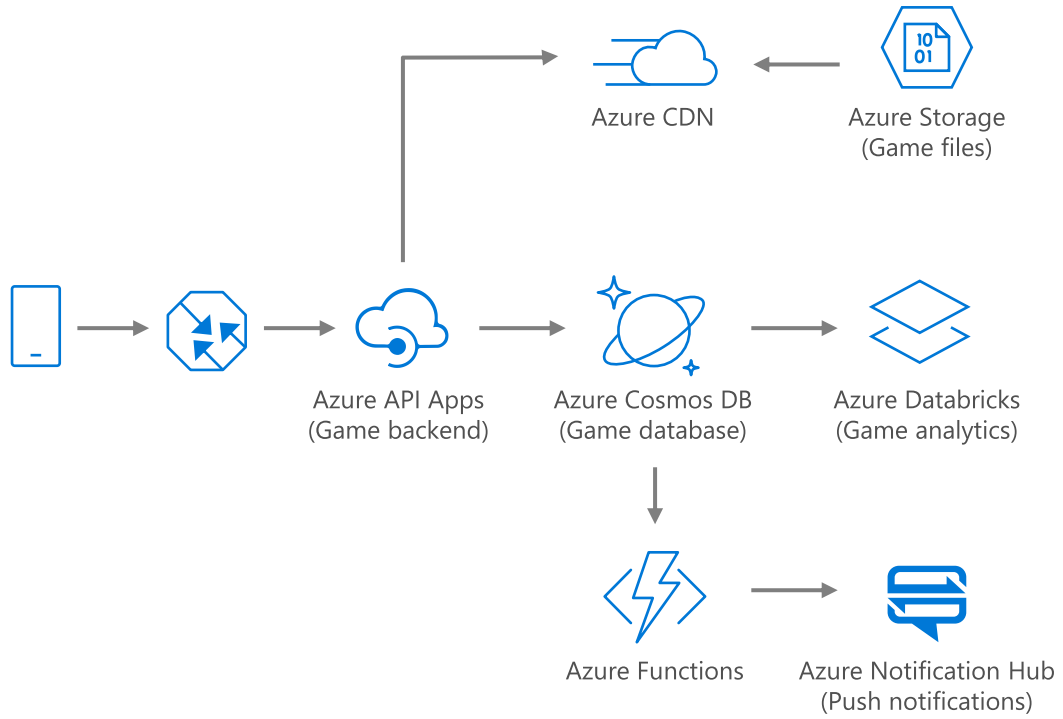


asos

IDEAL FOR GAMING, IOT AND ECOMMERCE

Maintain service quality during high-traffic periods requiring massive scale and performance.

- Instant, elastic scaling handles traffic bursts
- Uninterrupted global user experience
- Low-latency data access and processing for large and changing user bases
- High availability across multiple data centers

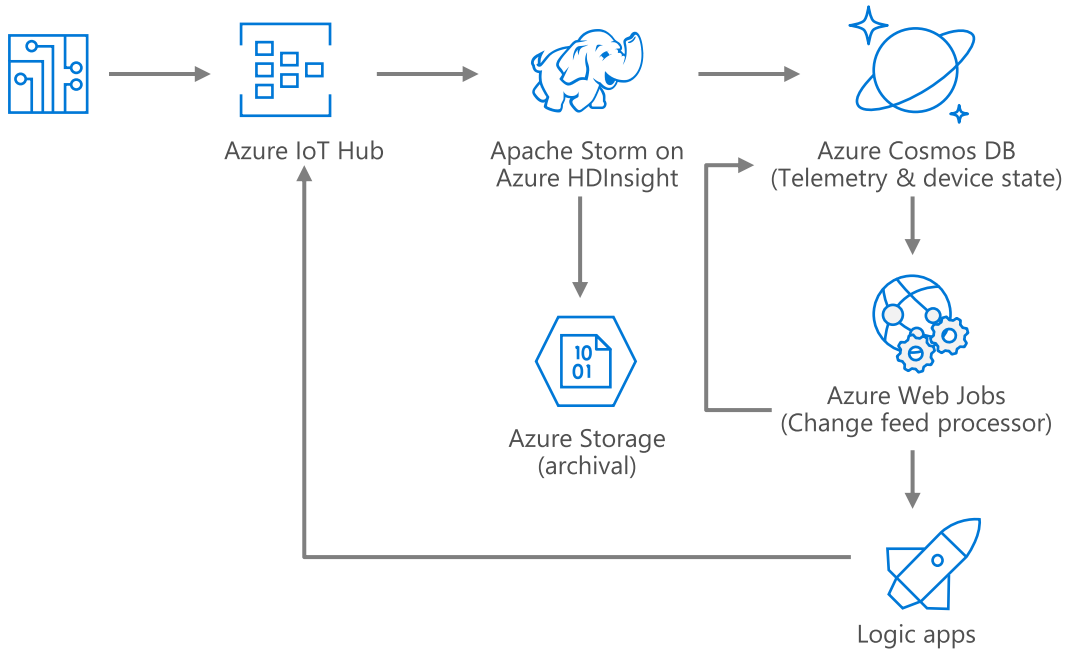


NEXT GAMES
Domino's
jet
XBOX LIVE

MASSIVE SCALE TELEMETRY STORES FOR IOT

Diverse and unpredictable IoT sensor workloads require a responsive data platform

- Seamless handling of any data output or volume
- Data made available immediately, and indexed automatically
- High writes per second, with stable ingestion and query performance

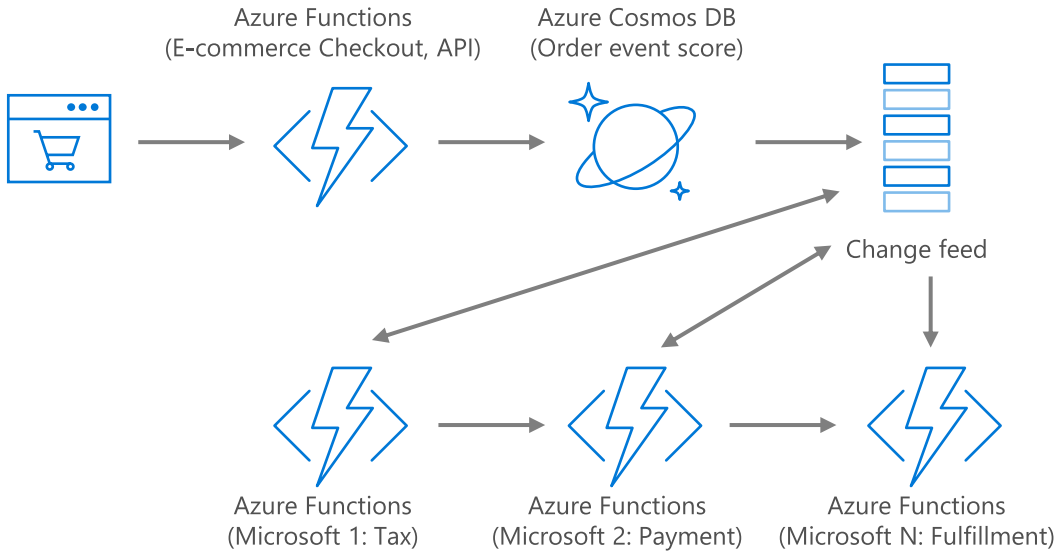


TOYOTA Honeywell LG CNS Johnson Controls

SIMPLIFIED DEVELOPMENT WITH SERVERLESS ARCHITECTURE

Experience decreased time-to-market, enhanced scalability, and freedom from framework management with event-driven micro-services.

- Seamless handling of any data output or volume
- Data made available immediately, and indexed automatically
- High writes per second, with stable ingestion and query performance
- Real-time, resilient change feeds logged forever and always accessible
- Native integration with Azure Functions

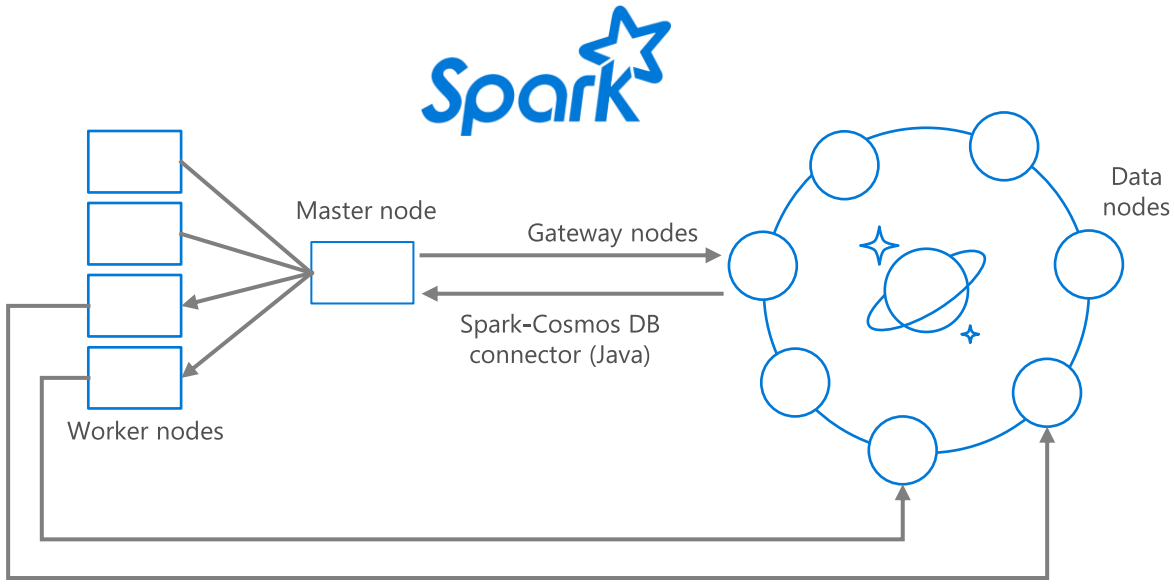


asos jet

RUN SPARK OVER OPERATIONAL DATA

Accelerate analysis of fast-changing, high-volume, global data.

- Real-time big data processing across any data model
- Machine learning at scale over globally-distributed data
- Speeds analytical queries with automatic indexing and push-down predicate filtering
- Native integration with Spark Connector

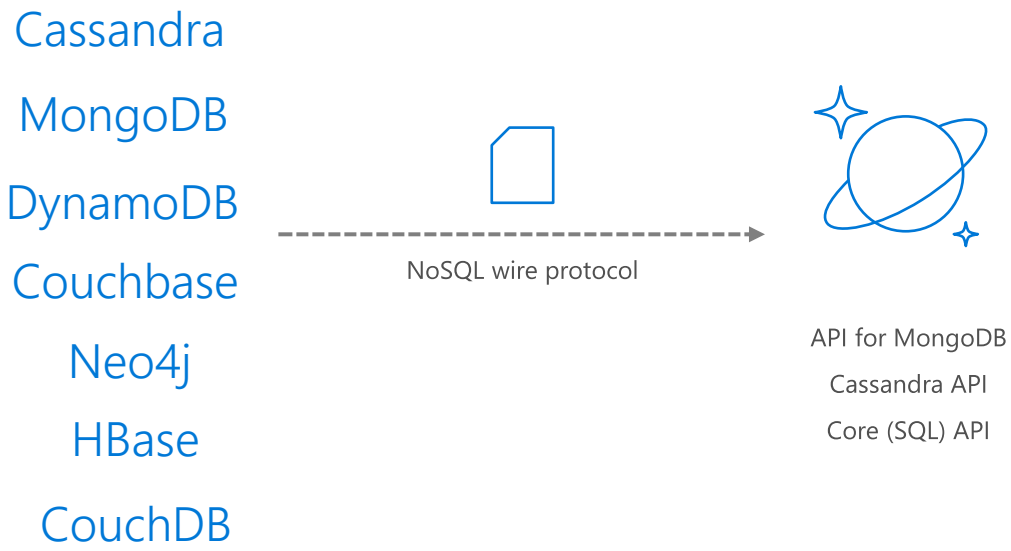


Affinio

MIGRATE NOSQL APPS

Make data modernization easy with seamless migration of NoSQL workloads to the cloud.

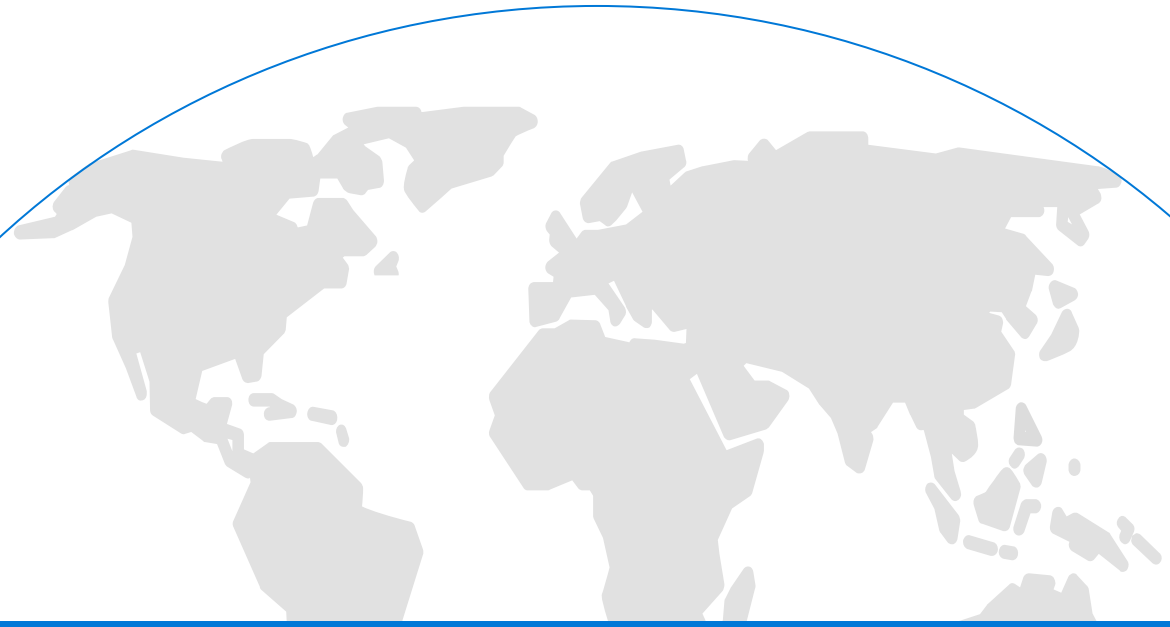
- Bring app data from existing NoSQL deployments in MongoDB, Cassandra, Gremlin, and more
- Leverage existing tools, drivers, and libraries, and continue using existing apps' current SDKs
- Turnkey geo-replication
- No infrastructure or VM management required



AZURE COSMOS DB (3 of 3)

A globally distributed, massively scalable, multi-model database service.

- Multi-model data with your favorite API
- Elastically scale storage and throughput
- Multiple, well-defined consistency levels
- <10ms latency guarantees at the 99th percentile
- Industry-leading SLAs across performance, latency, availability and throughput



Lift and shift MongoDB apps

Run Spark over operational data

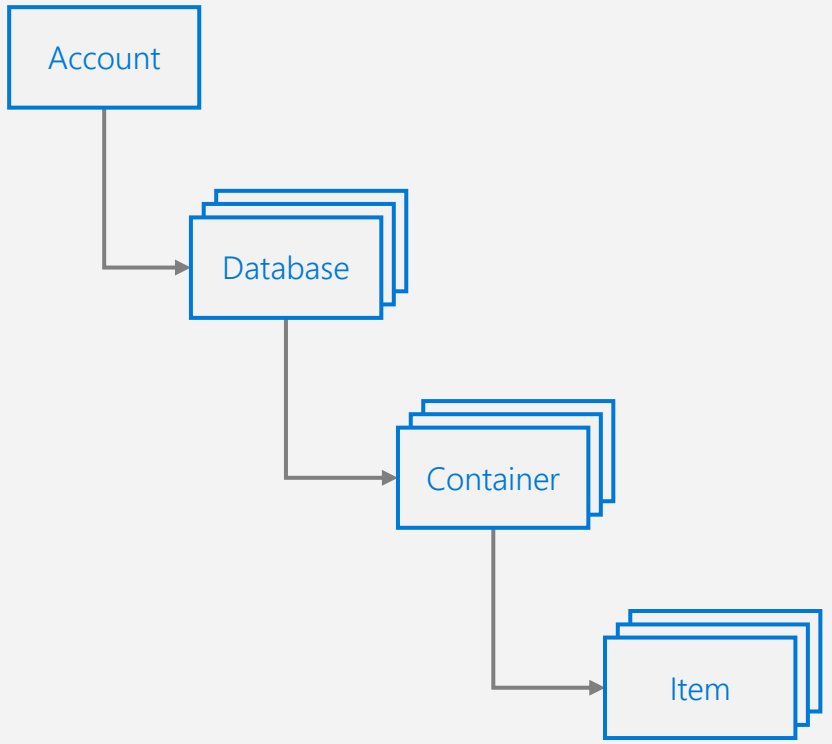
Build real-time customer experiences

Ideal for IoT, gaming and eCommerce

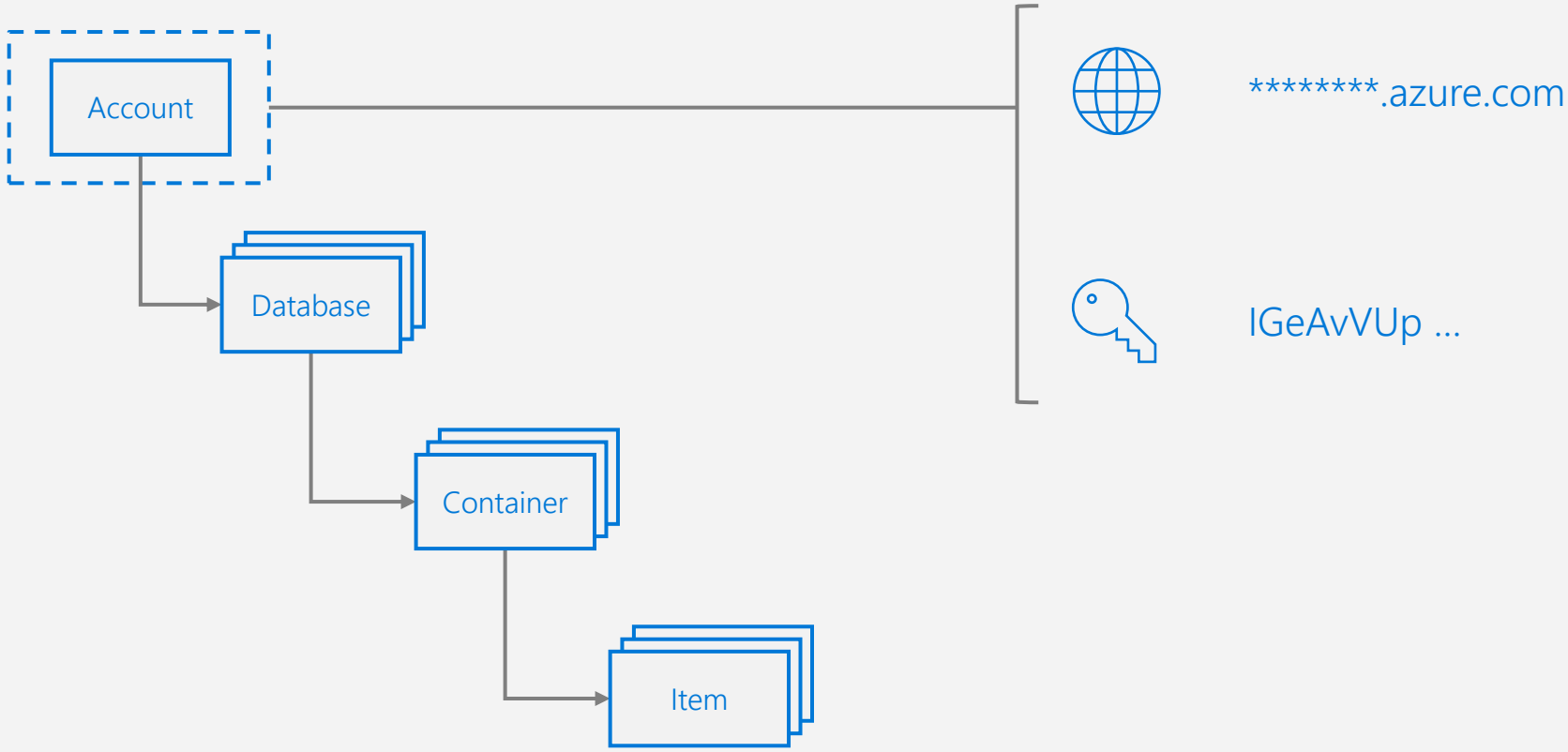
RESOURCE MODEL

Leveraging Azure Cosmos DB to automatically scale your data across the globe

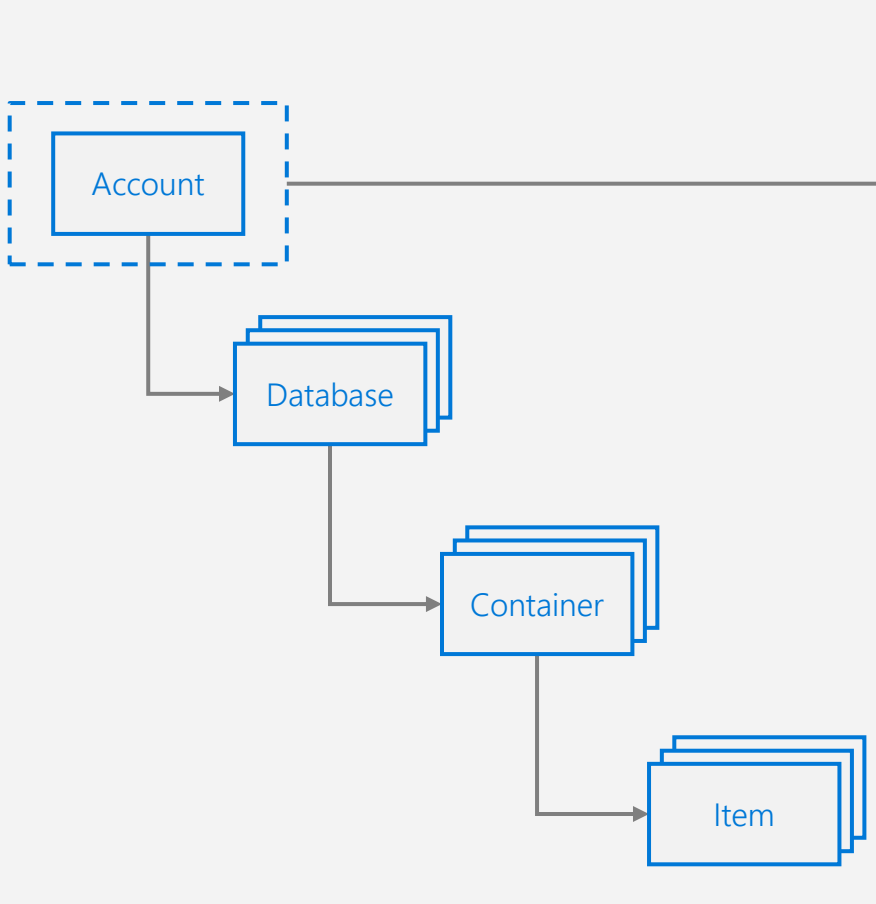
This module will reference partitioning in the context of all Azure Cosmos DB modules and APIs.

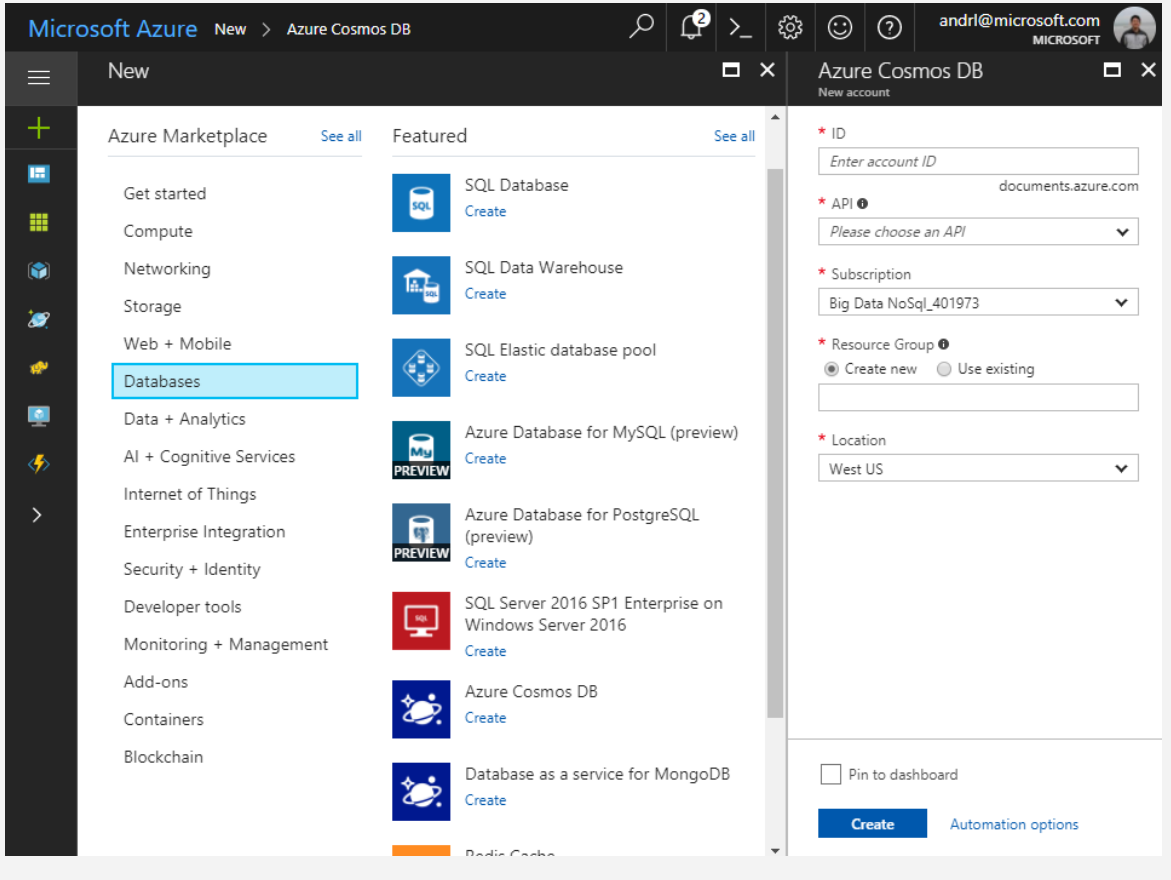


ACCOUNT URI AND CREDENTIALS

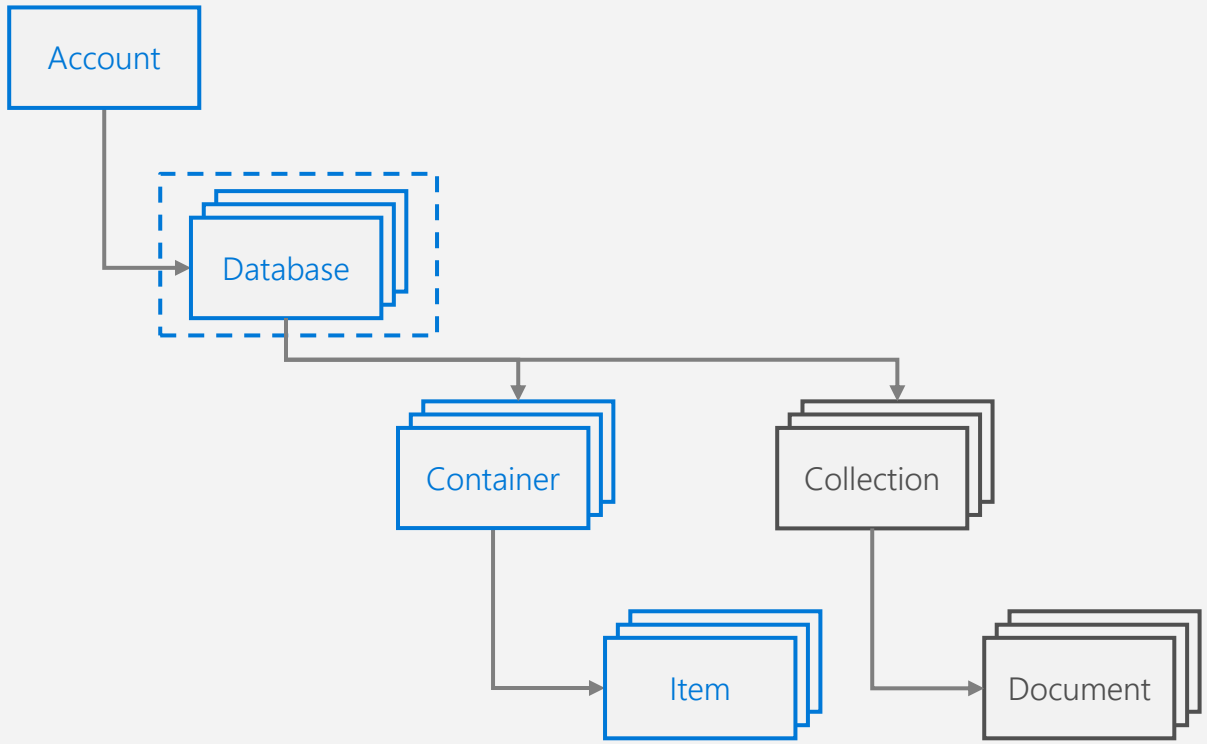


CREATING ACCOUNT

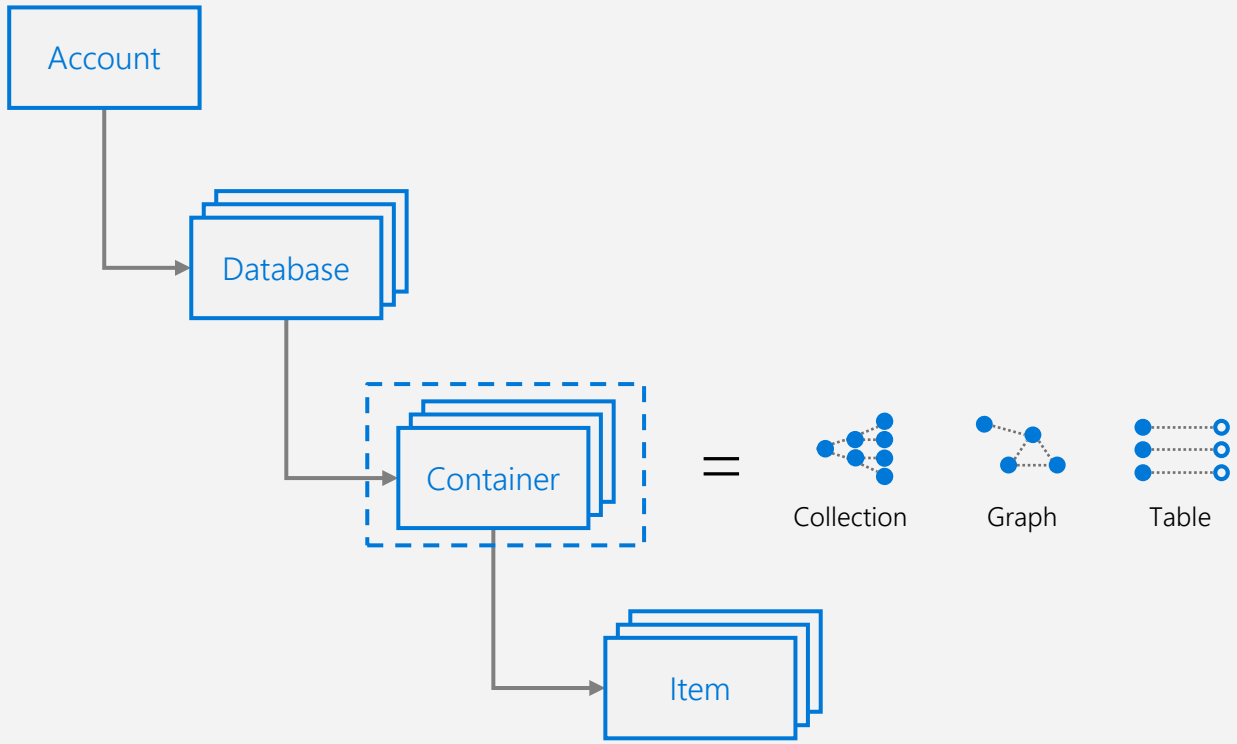




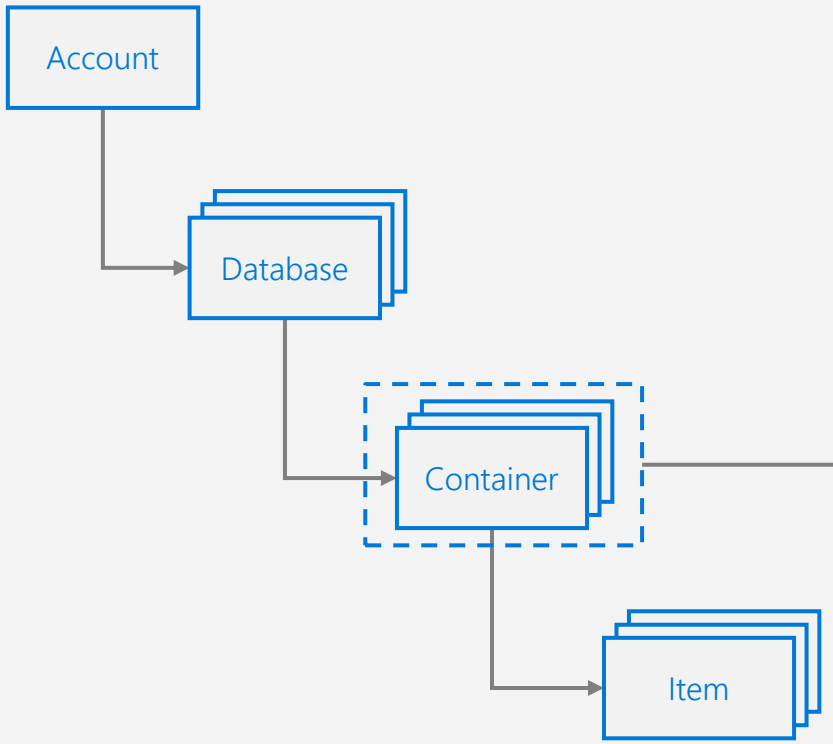
DATABASE REPRESENTATIONS

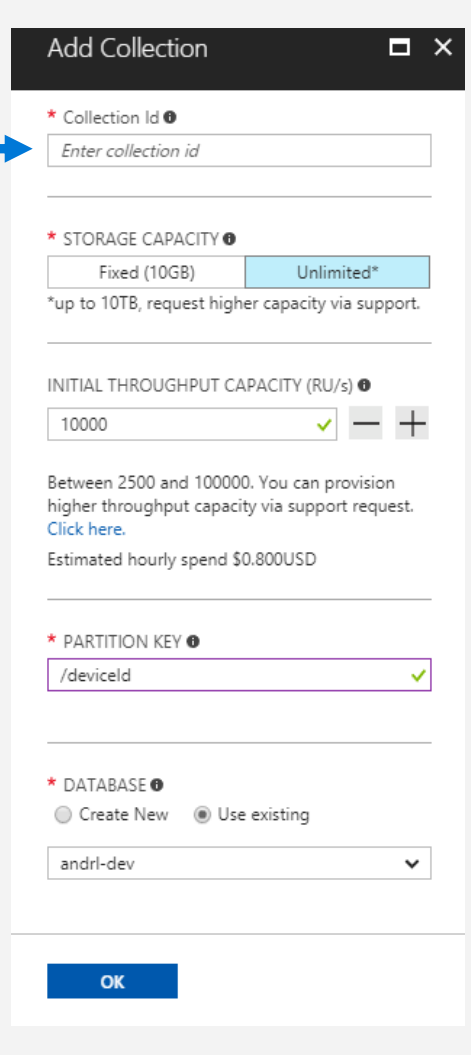


CONTAINER REPRESENTATIONS

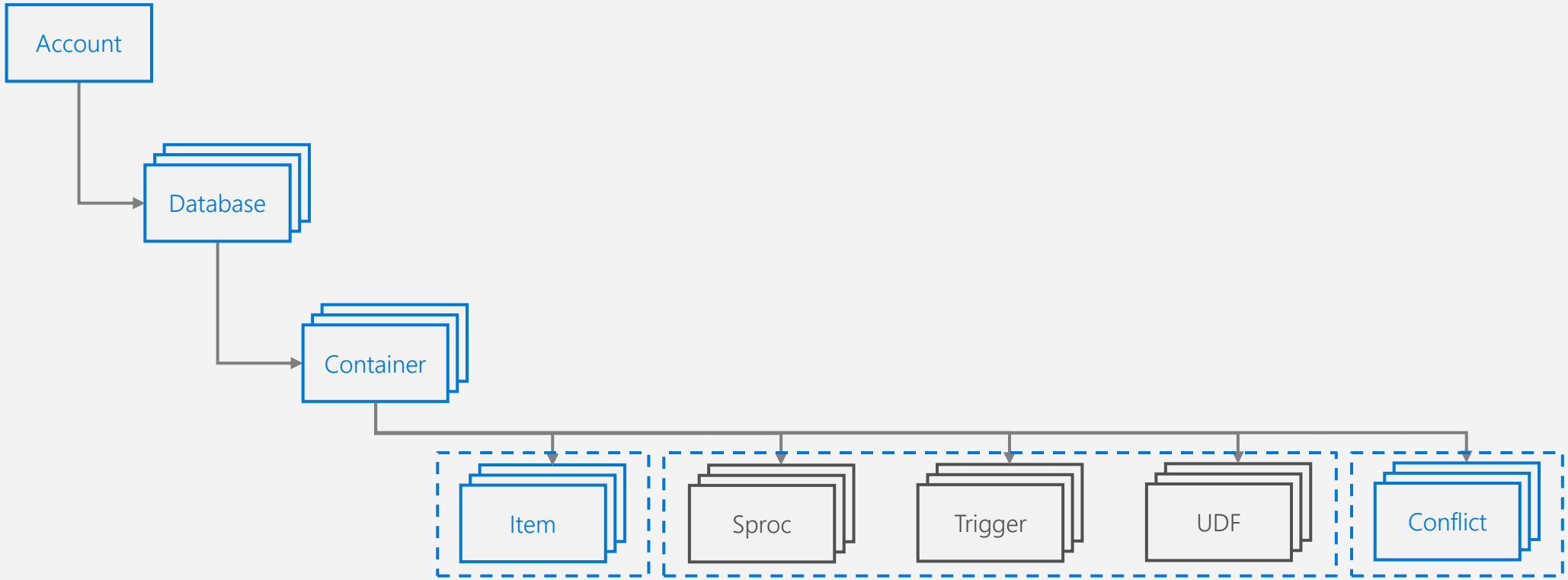


CREATING COLLECTIONS – SQL API

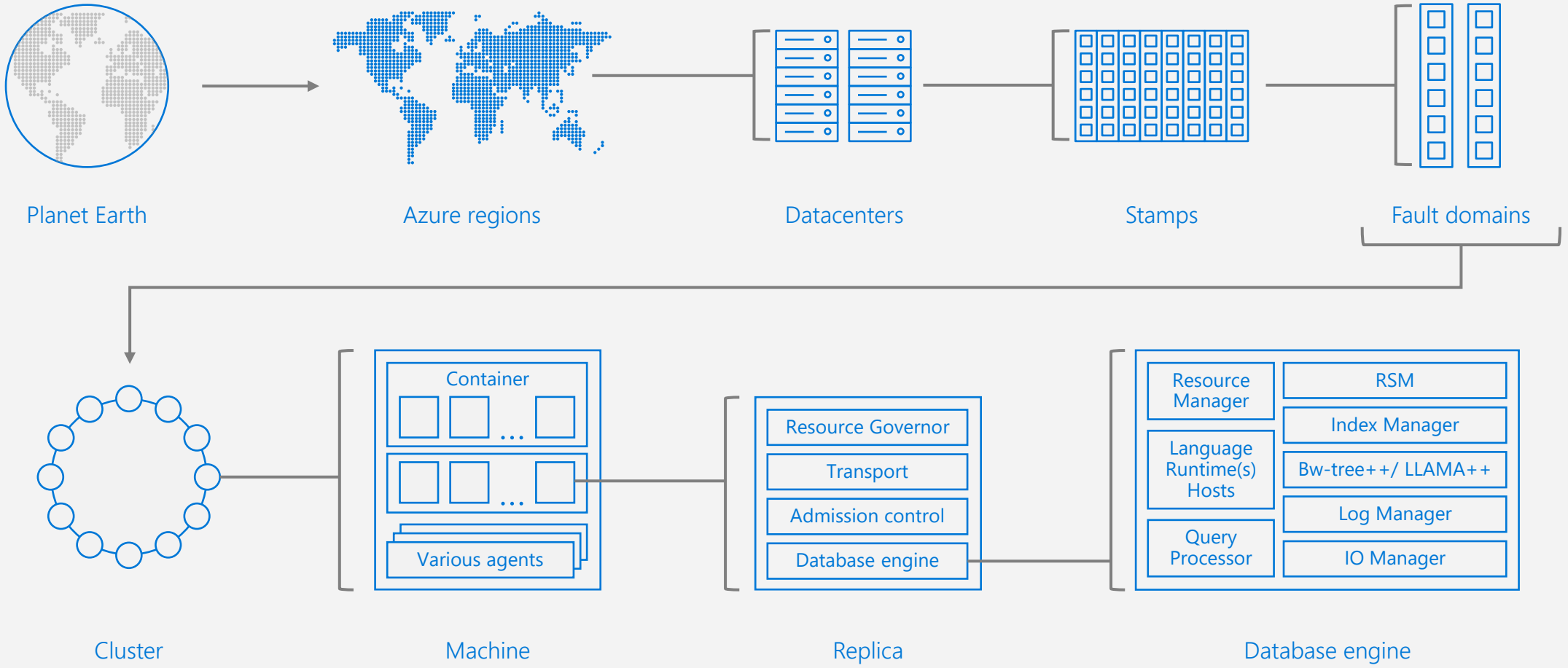




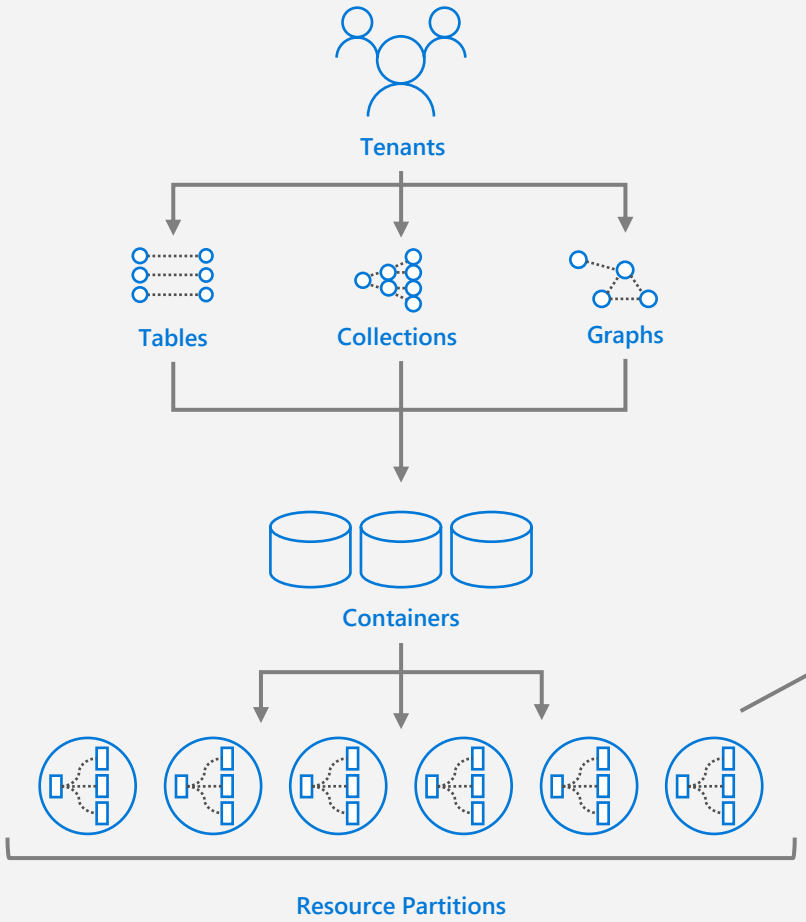
CONTAINER-LEVEL RESOURCES



SYSTEM TOPOLOGY (BEHIND THE SCENES)

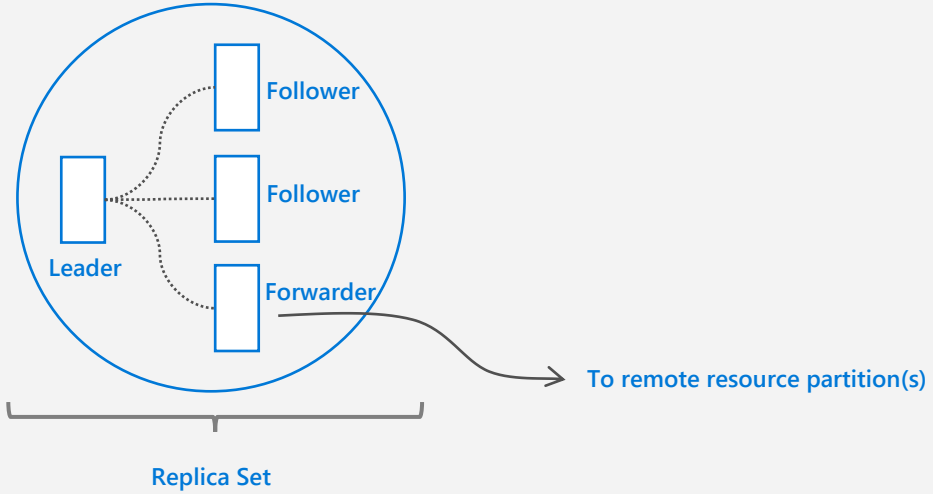


RESOURCE HIERARCHY



CONTAINERS

Logical resources “surfaced” to APIs as tables, collections or graphs, which are made up of one or more physical partitions or servers.



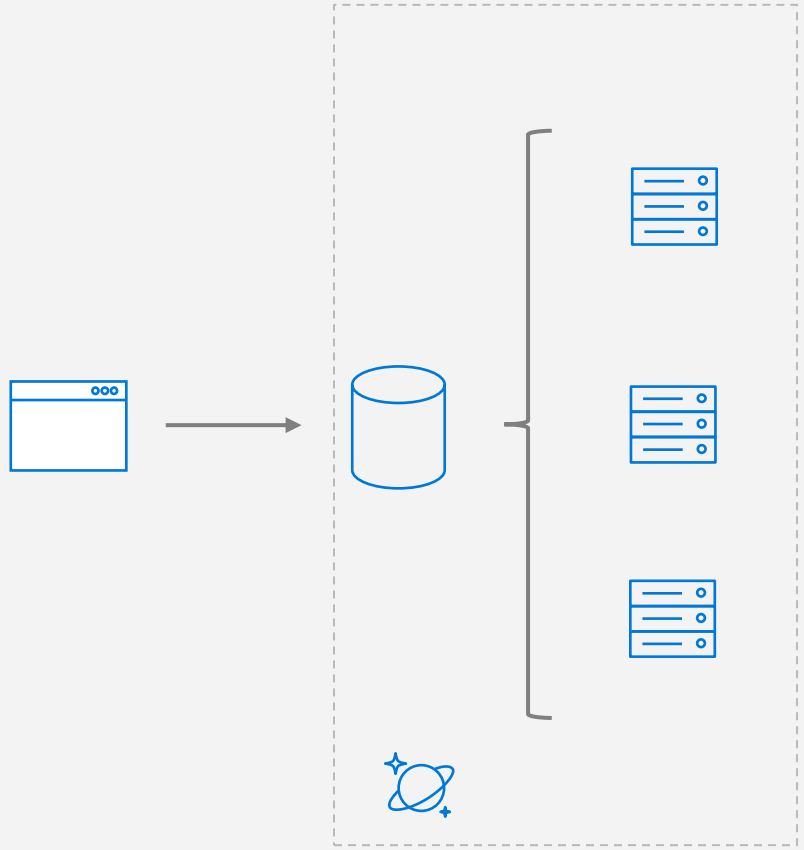
RESOURCE PARTITIONS

- Consistent, highly available, and resource-governed coordination primitives
- Consist of replica sets, with each replica hosting an instance of the database engine

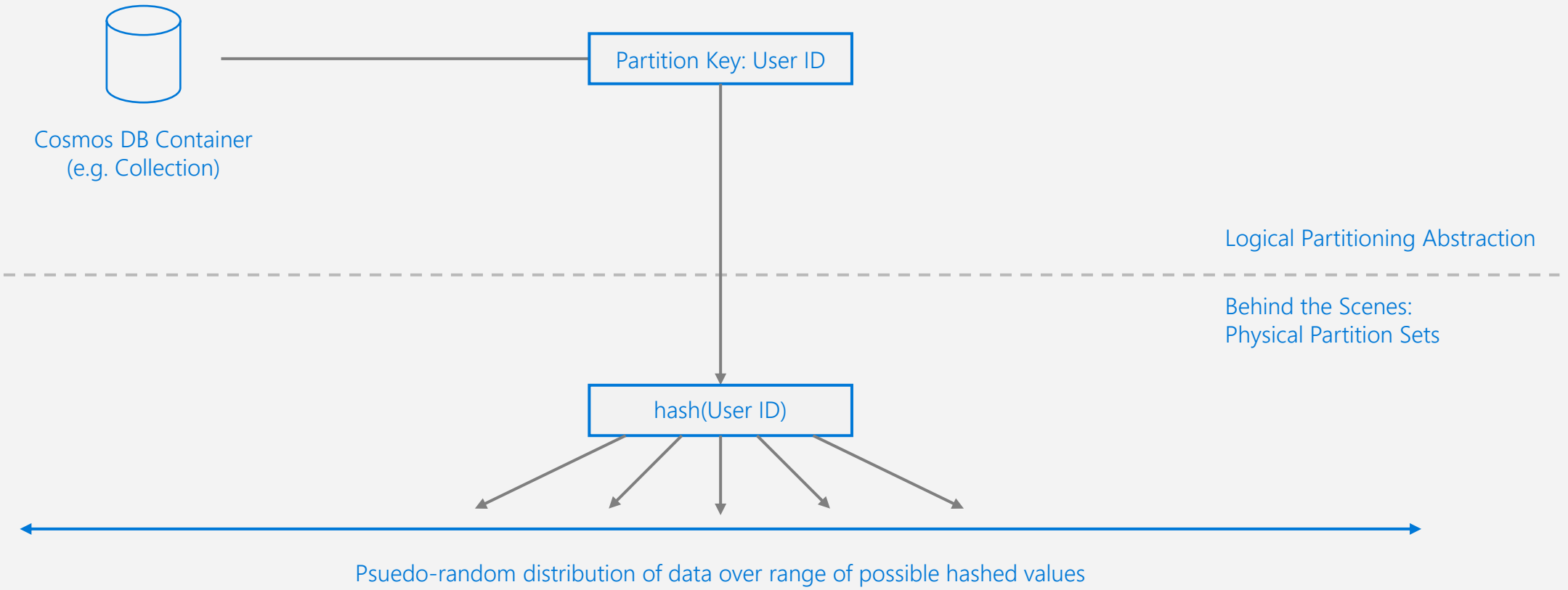
PARTITIONING

Leveraging Azure Cosmos DB to automatically scale your data across the globe

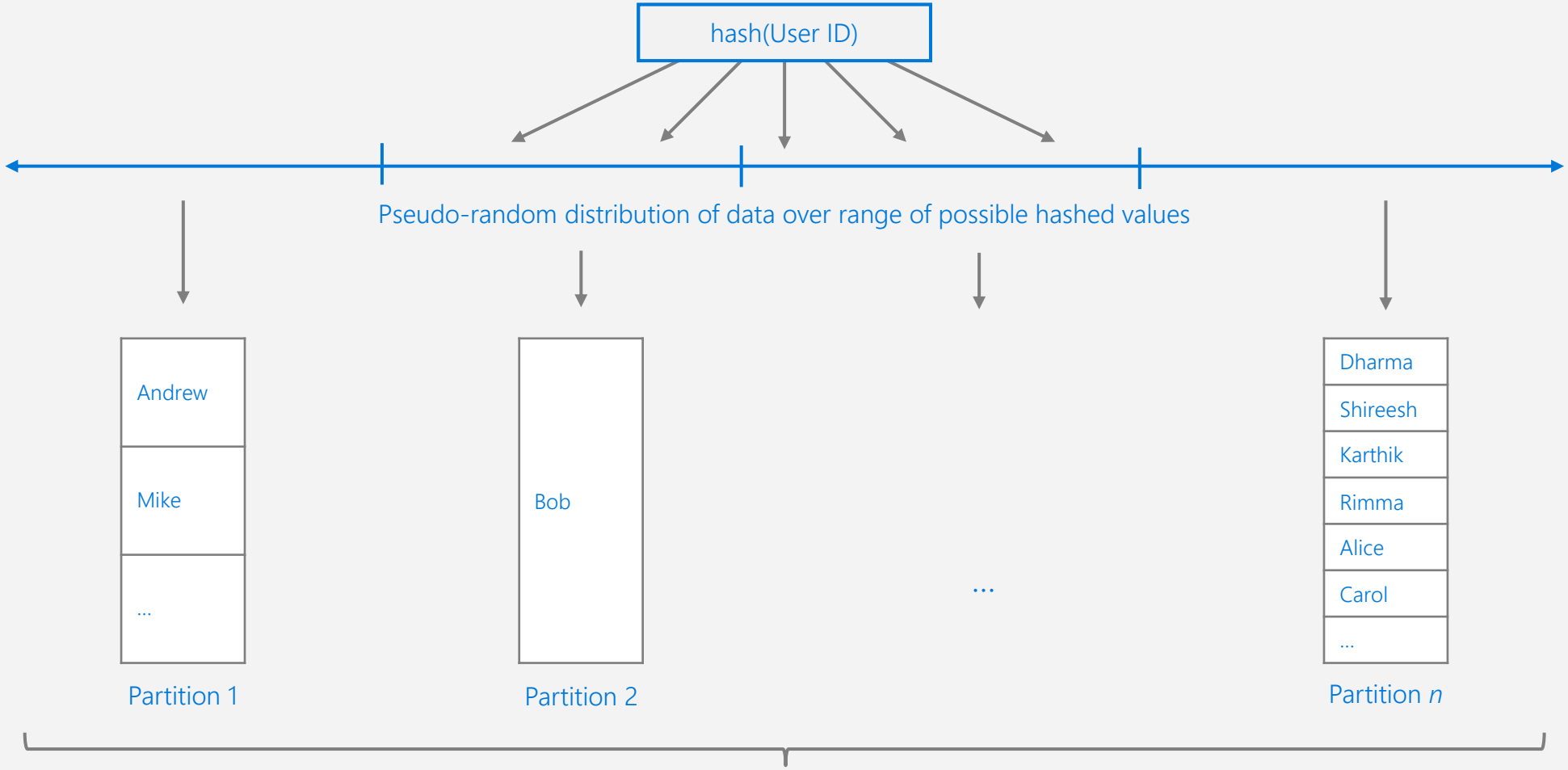
This module will reference partitioning in the context of all Azure Cosmos DB modules and APIs.



PARTITIONS (1 of 5)

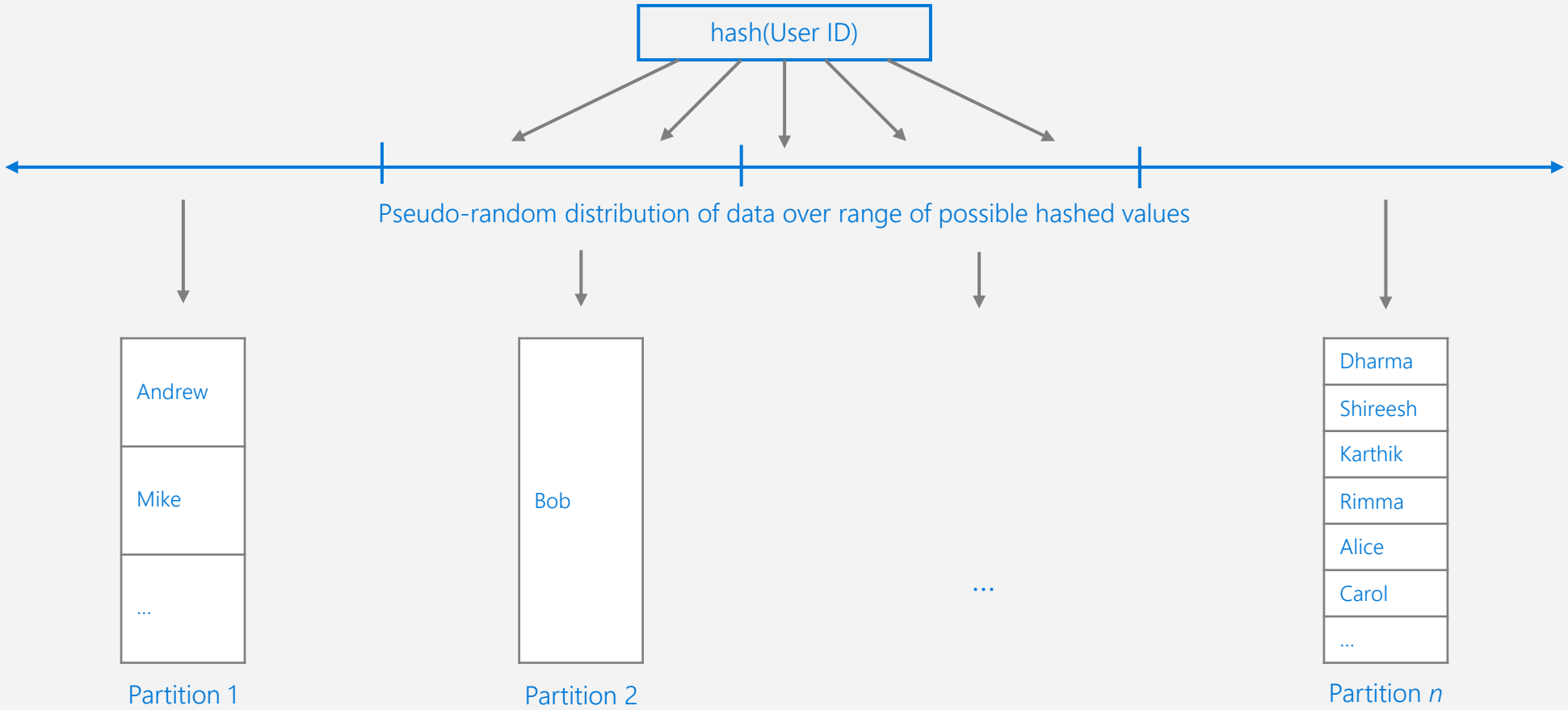


PARTITIONS (2 of 5)



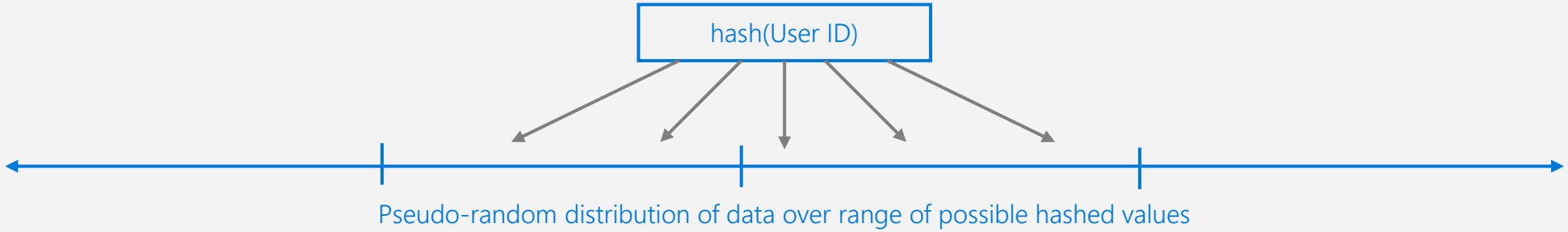
Frugal # of Partitions based on actual storage and throughput needs
(yielding scalability with low total cost of ownership)

PARTITIONS (3 of 5)



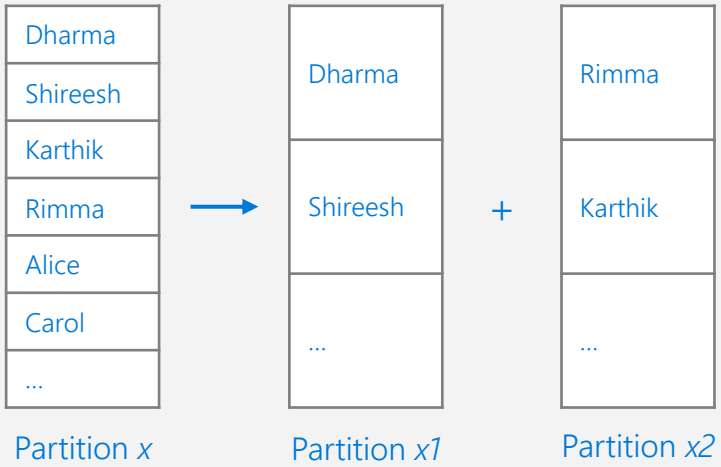
What happens when partitions need to grow?

PARTITIONS (4 of 5)



Partition Ranges can be dynamically sub-divided to seamlessly grow database as the application grows while simultaneously maintaining high availability.

Partition management is fully managed by Azure Cosmos DB, so you don't have to write code or manage your partitions.



DEMO

Creating Unlimited Containers

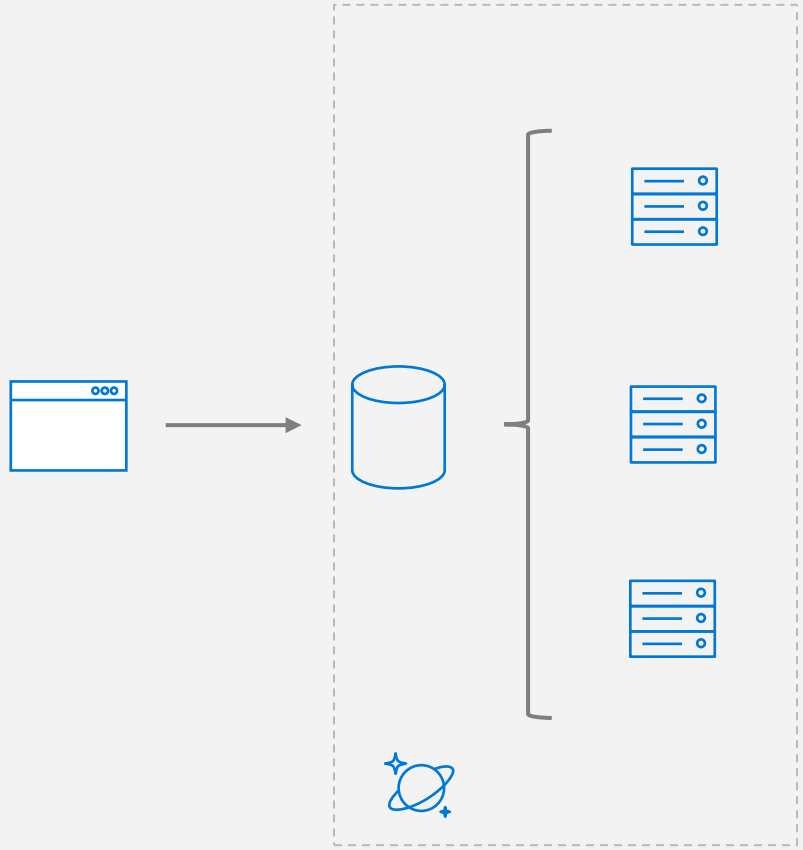
PARTITION DESIGN (1 of 2)

IMPORTANT TO SELECT THE "RIGHT" PARTITION KEY

Partition keys acts as a **means for efficiently routing queries** and as a boundary for **multi-record** transactions.

KEY MOTIVATIONS

- Distribute Requests
- Distribute Storage
- Intelligently Route Queries for Efficiency



PARTITION DESIGN (2 of 2)

EXAMPLE SCENARIO

Contoso Connected Car is a vehicle telematics company. They are planning to store vehicle telemetry data from millions of vehicles every second in Azure Cosmos DB to power predictive maintenance, fleet management, and driver risk analysis.

The partition key we select will be the scope for multi-record transactions.

WHAT ARE A FEW POTENTIAL PARTITION KEY CHOICES?

- Vehicle Model
- Current Time
- Device Id
- Composite Key – Device ID + Current Time

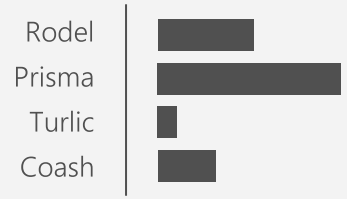
PARTITION KEY CHOICES (1 of 2)

VEHICLE MODEL (e.g. Model A)

Most auto manufactures only have a couple dozen models. This will create a fixed number of logical partition key values; and is potentially the least granular option.

Depending how uniform sales are across various models – this introduces possibilities for hot partition keys on both storage and throughput.

Storage Distribution



Throughput Distribution

CURRENT MONTH (e.g. 2018-04)

Auto manufacturers have transactions occurring throughout the year. This will create a more balanced distribution of storage across partition key values.

However, most business transactions occur on recent data creating the possibility of a hot partition key for the current month on throughput.

Storage Distribution

Throughput Distribution

Example – Contoso Connected Car

PARTITION KEY CHOICES (2 of 2)

DEVICE ID (e.g. Device123)

Each car would have a unique device ID. This creates a large number of partition key values and would have a significant amount of granularity.

Depending on how many transactions occur per vehicle, it is possible to a specific partition key that reaches the storage limit per partition key

Storage Distribution

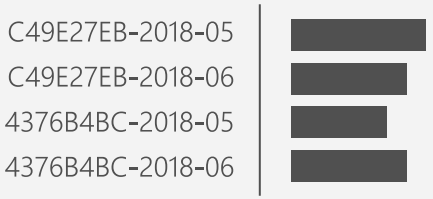
Throughput Distribution

COMPOSITE KEY (Device ID + Time)

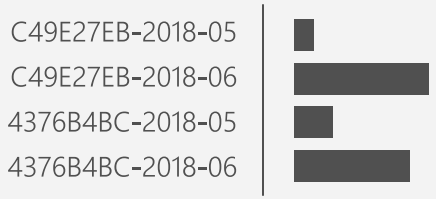
This composite option increases the granularity of partition key values by combining the current month and a device ID. Specific partition key values have less of a risk of hitting storage limitations as they only relate to a single month of data for a specific vehicle.

Throughput in this example would be distributed more to logical partition key values for the current month.

Storage Distribution



Throughput Distribution



Example – Contoso Connected Car

PARTITION GRANULARITY (1 of 3)

SELECT THE "RIGHT" LEVEL OF GRANULARITY FOR YOUR PARTITIONS

Partitions should be based on your most often occurring query and transactional needs. The goal is to **maximize granularity** and **minimize cross-partition requests**.

Don't be afraid to have more partitions!

More partition keys = More scalability

Example – Contoso Connected Car

PARTITION GRANULARITY (2 of 3)

SELECT THE "RIGHT" LEVEL OF GRANULARITY FOR YOUR PARTITIONS

Consider storage & throughput thresholds

Consider cross-partition query likelihood

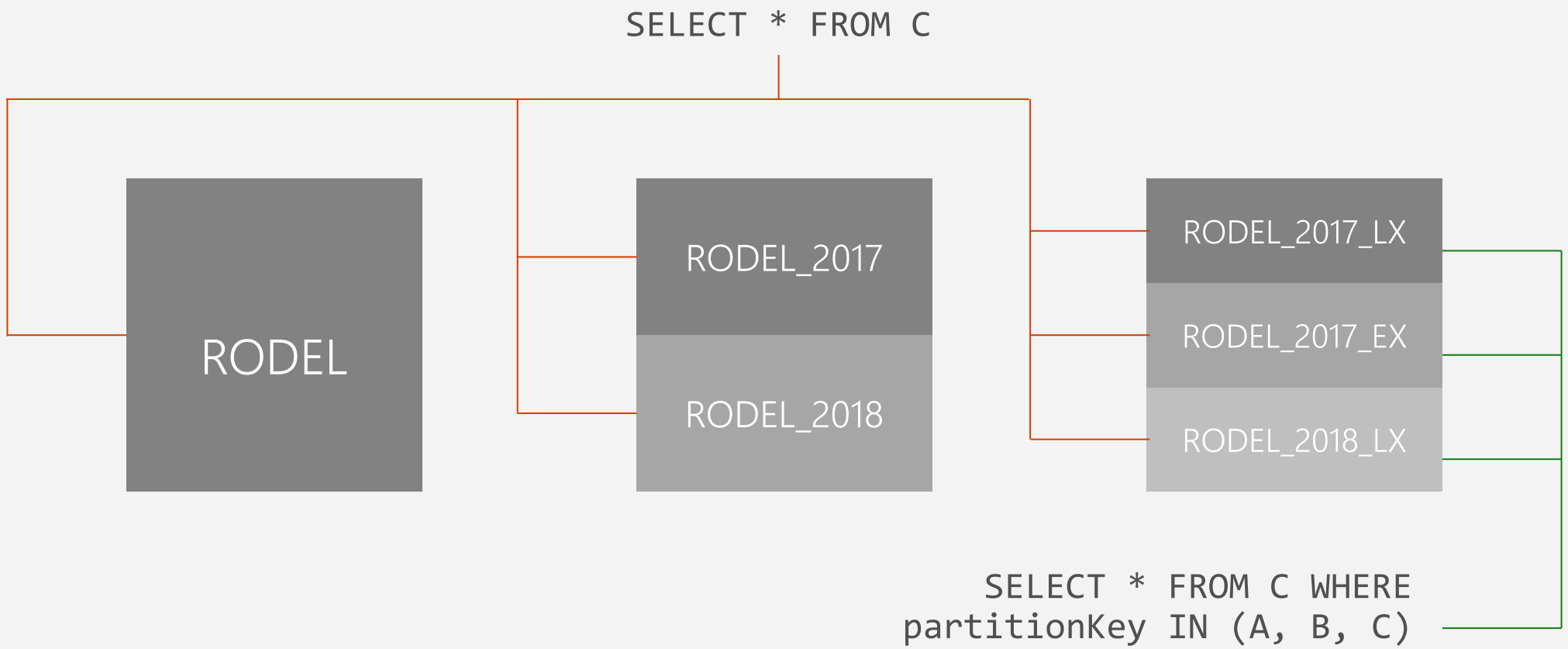
Don't be afraid to have more partitions!

More partition keys = More scalability

Example – Contoso Connected Car

PARTITION GRANULARITY (3 of 3)

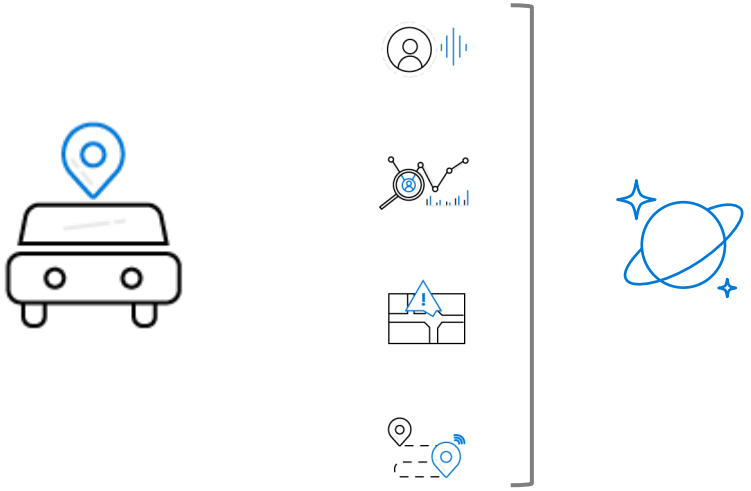
A CROSS-PARTITION QUERY IS NOT ALWAYS A BLIND FAN OUT QUERY

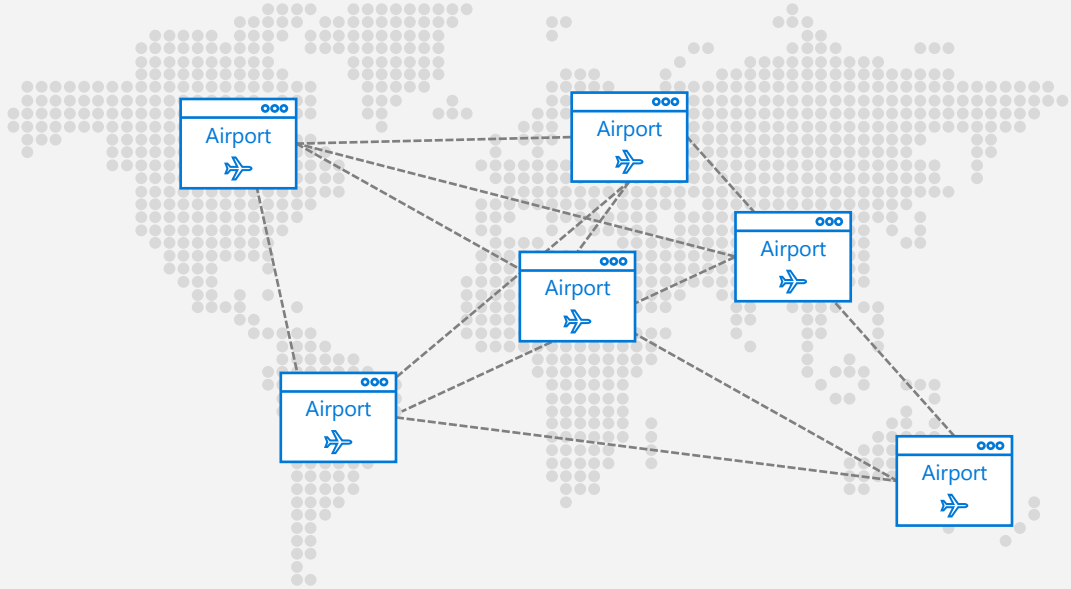


Example – Contoso Connected Car

PARTITION KEY SELECTION

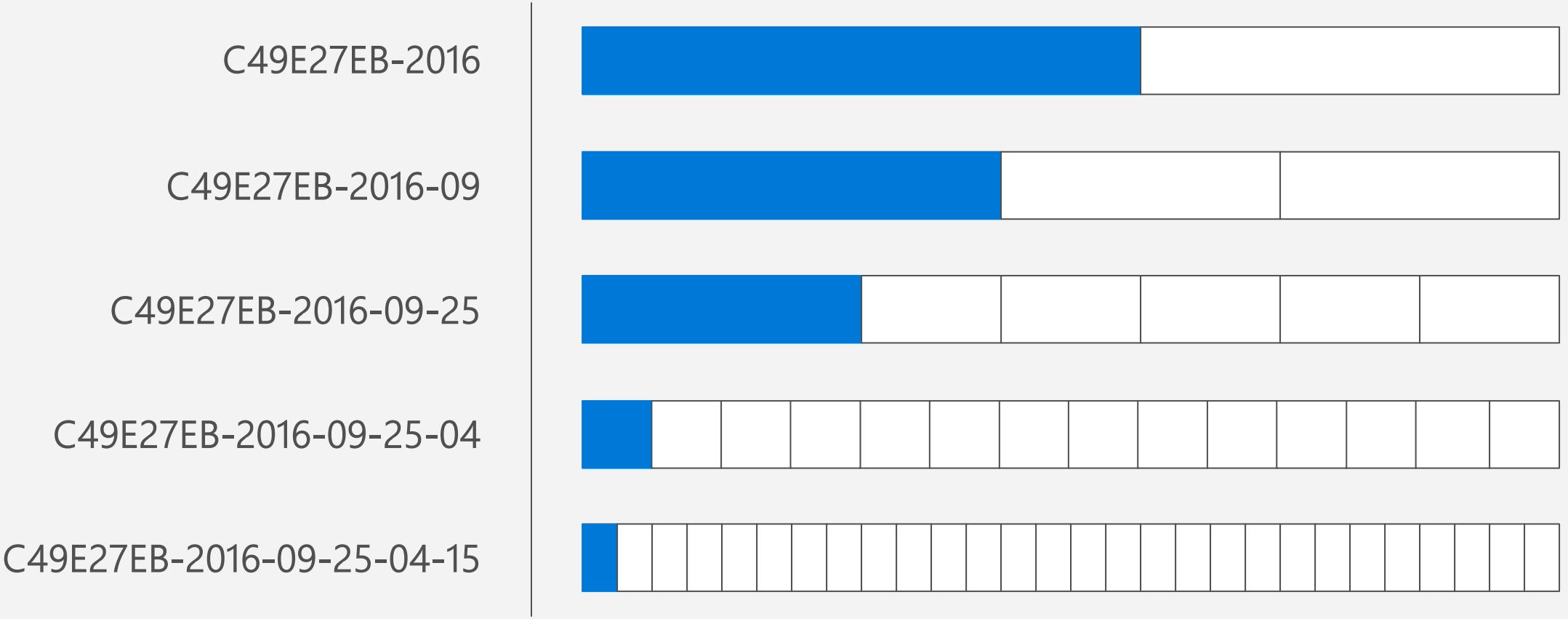
Contoso Connected Car is collecting and storing vehicle telemetry data from millions of vehicles. The team has decided to partition based on a composite key consisting of device id + current time when the interaction occurred.





PARTITION KEY SCENARIO (1 of 2)

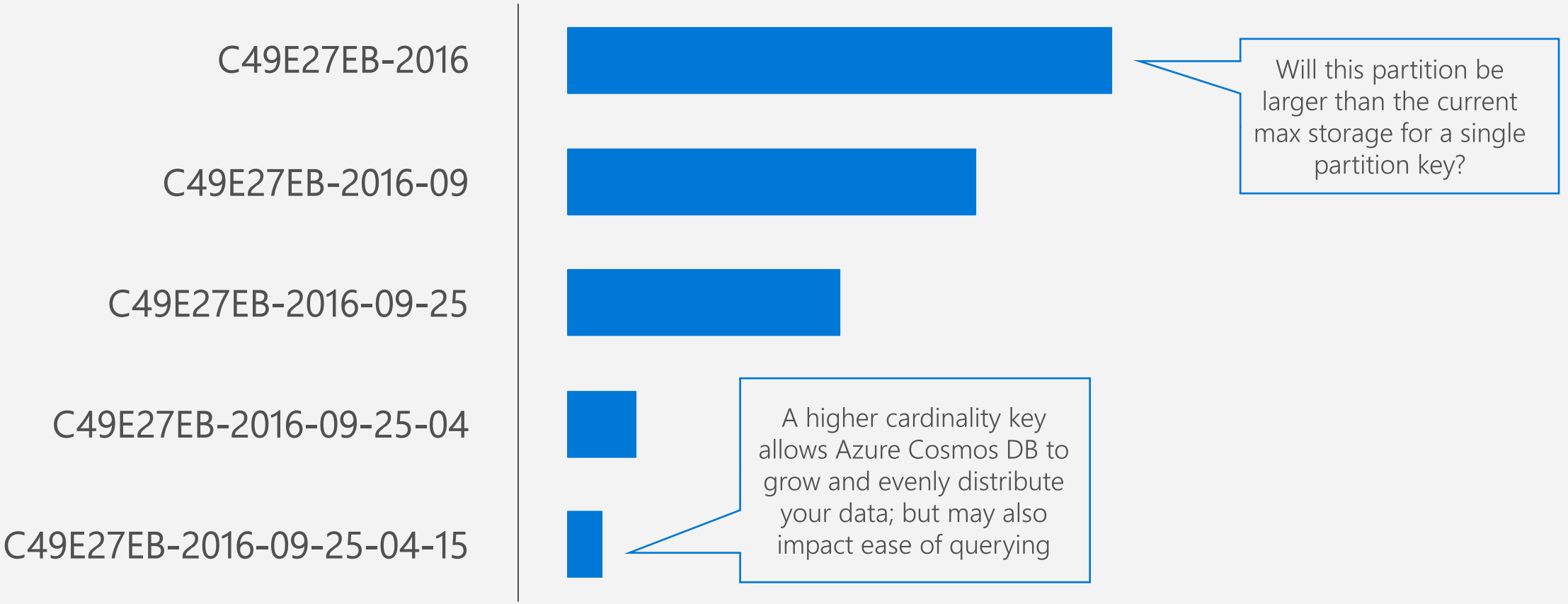
Interaction that occurred on: **September 25, 2016 at 4:15 AM UTC**



Example – Contoso Connected Car

PARTITION KEY SCENARIO (2 of 2)

Interaction that occurred on: **September 25, 2016 at 4:15 AM UTC**



Example – Contoso Connected Car

PARTITIONS (5 of 5)

Best Practices: Design Goals for Choosing a Good Partition Key

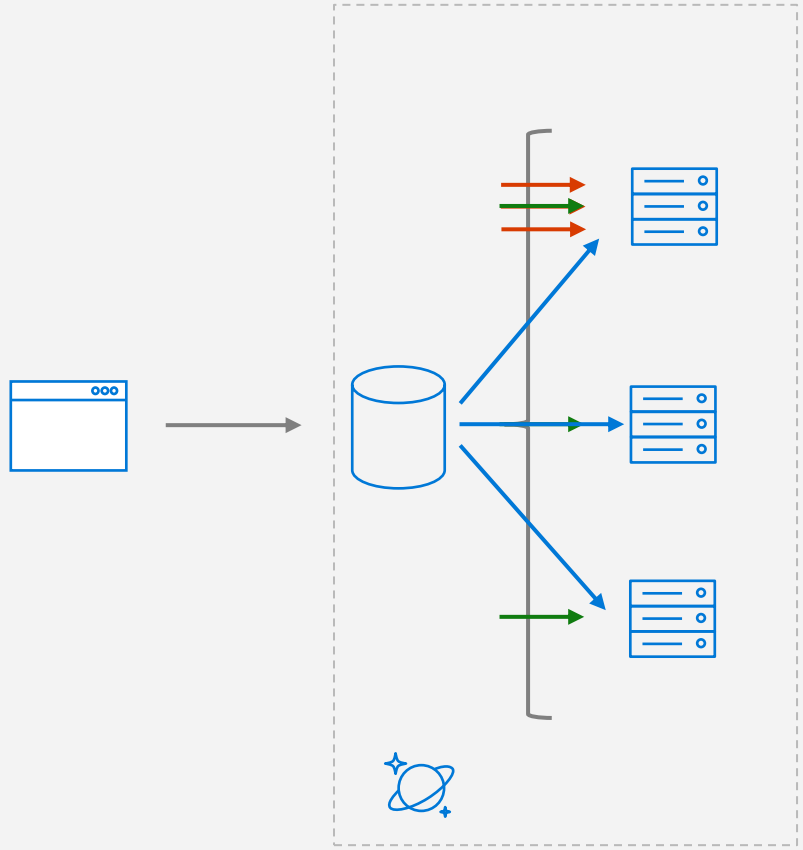
- Distribute the overall request + storage volume
  - Avoid "hot" partition keys
- Partition Key is scope for multi-record transactions and routing queries
  - Queries can be intelligently routed via partition key
  - Omitting partition key on query requires fan-out

Steps for Success

- Ballpark scale needs (size/throughput)
- Understand the workload
- # of reads/sec vs writes per sec
  - Use pareto principal (80/20 rule) to help optimize bulk of workload
  - For reads – understand top 3-5 queries (look for common filters)
  - For writes – understand transactional needs

General Tips

- Build a POC to strengthen your understanding of the workload and iterate (avoid analyses paralysis)
- Don't be afraid of having too many partition keys
  - Partitions keys are logical
  - More partition keys → more scalability



PARTITION KEY STORAGE LIMITS

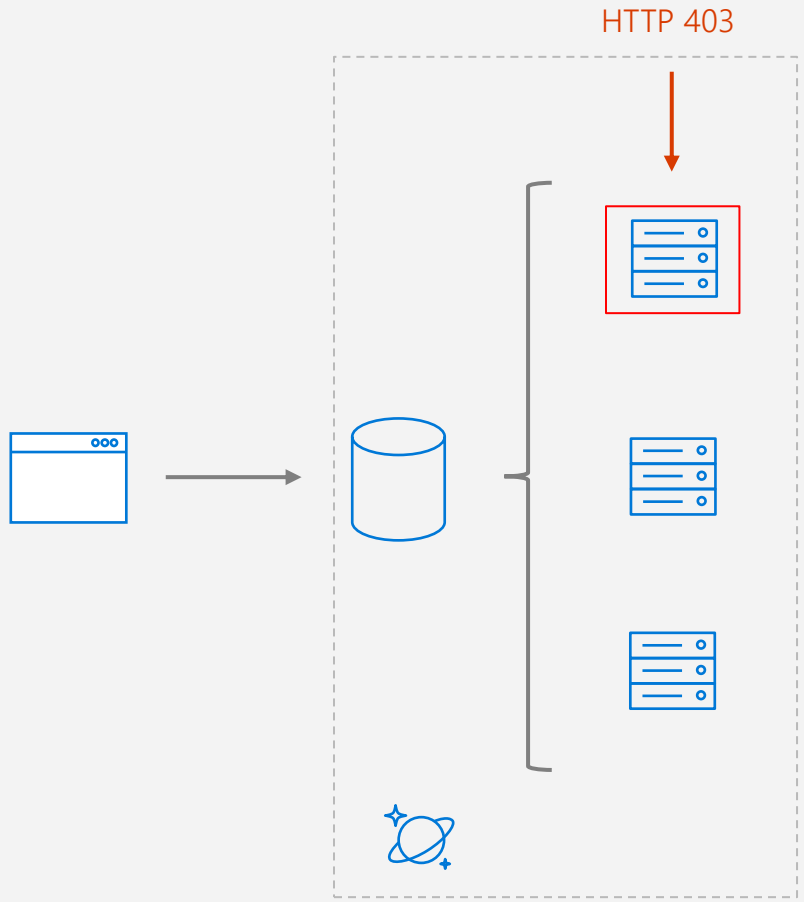
Containers support unlimited storage by dynamically allocating additional physical partitions

Storage for single partition key value (logical partition) is quota'ed to 20GB.

When a partition key reaches its provisioned storage limit, requests to create new resources will return a HTTP Status Code of 403 (Forbidden).

Azure Cosmos DB will automatically add partitions, and may also return a 403 if:

- An authorization token has expired
- A programmatic element (UDF, Stored Procedure, Trigger) has been flagged for repeated violations

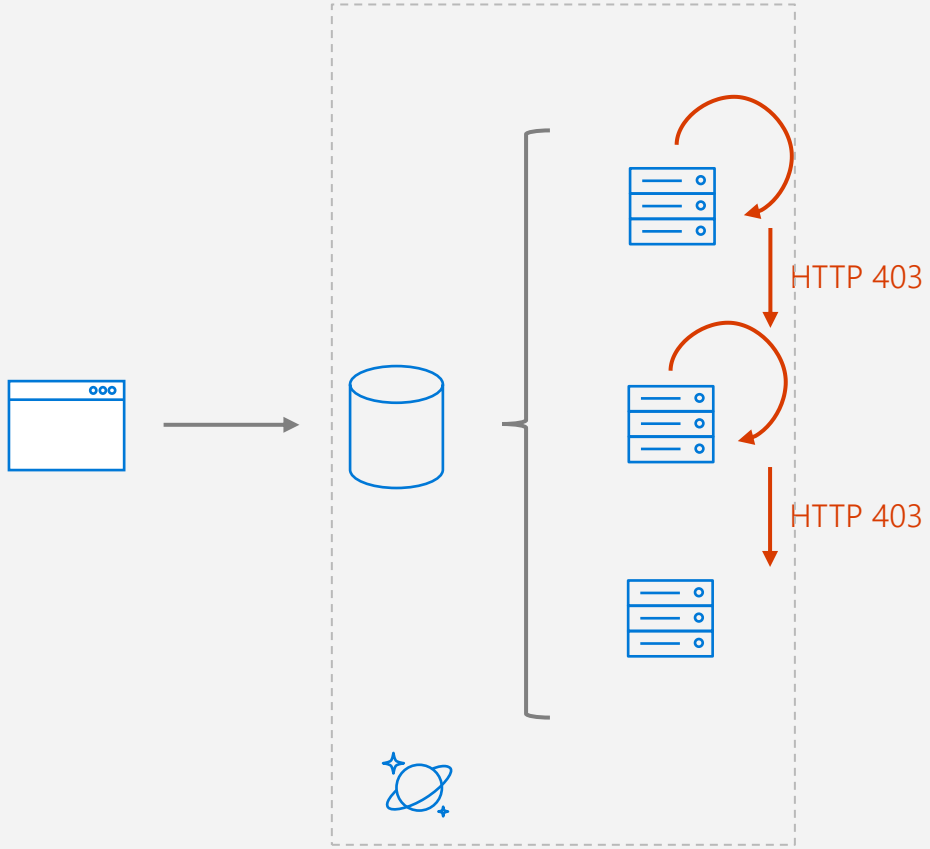


DESIGN PATTERNS FOR LARGE PARTITION KEYS (1 of 2)

"LINKED LIST APPROACH" BY SREADING DATA ACROSS INCREMENTAL PARTITION KEY VALUES

For workloads that exceed quotas for a single partition key value, you can logically spread items across multiple partition keys within a container by using a suffix on the partition key value.

As a partition fills up, you can determine when to **increment** the partition key value by looking for the 403 status code in your application's logic.

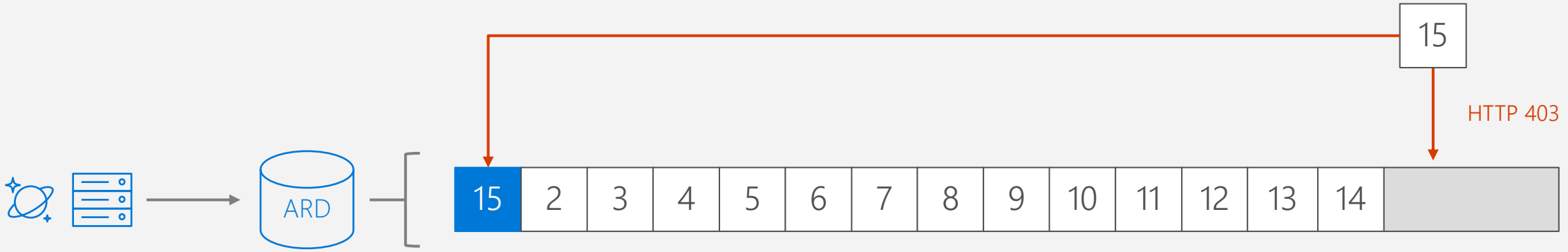


DESIGN PATTERNS FOR LARGE PARTITION KEYS (2 of 2)

"CIRCULAR BUFFER" APPROACH BY REUSING UNIQUE IDS

As you insert new items into a container's partition, you can increment the unique id for each item in the partition.

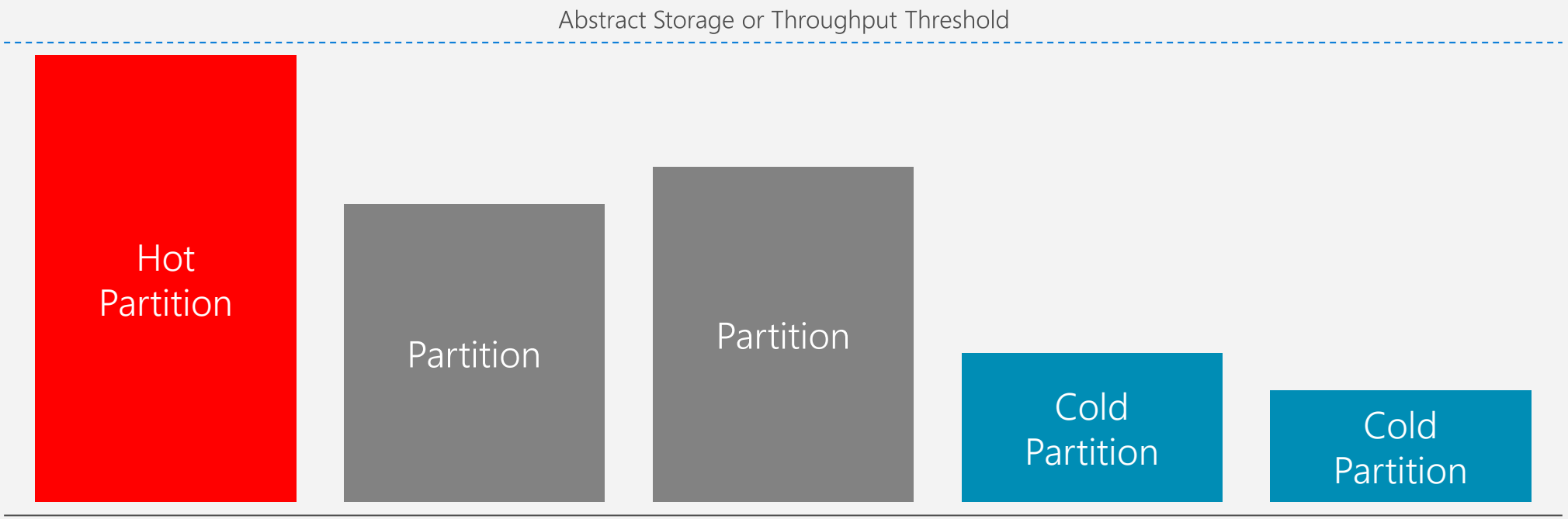
When you get a 403 status code, indicating the partition is full, you can restart your unique id and upsert the items to replace older documents.



HOT/COLD PARTITIONS

PARTITION USAGE CAN VARY OVER TIME

Partitions that are approaching thresholds are referred to as **hot**. Partitions that are underutilized are referred to as **cold**.



QUERY FAN-OUT

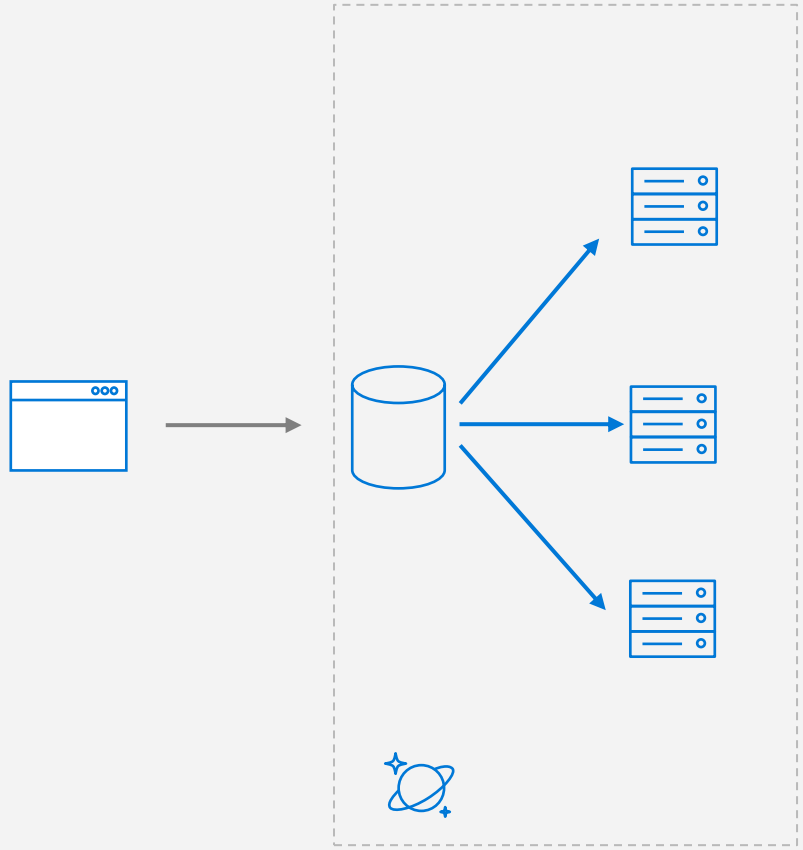
CROSS-PARTITION QUERIES CAN BE PERFORMED SERVER-SIDE OR CLIENT-SIDE

Cross-partition queries are opt-in

- Cross-partition queries can be tuned and parallelized

Creates a bottleneck

- Must wait for all partitions to return before the query is "done"



CROSS-PARTITION SDK EXAMPLE (1 of 3)

```
IQueryable<DeviceReading> crossPartitionQuery = client.CreateDocumentQuery<DeviceReading>(
    UriFactory.CreateDocumentCollectionUri("db", "coll"),
    new FeedOptions {
          EnableCrossPartitionQuery = true,
          MaxDegreeOfParallelism = 10,
          MaxBufferedItemCount = 100
    })
    .Where(m => m.MetricType == "Temperature" && m.MetricValue > 100)
    .OrderBy(m => m.MetricValue);
```

.NET

CROSS-PARTITION SDK EXAMPLE (2 of 3)

```
var querySpec = {
          query: 'SELECT * FROM container c’
};

var feedOptions = {
          enableCrossPartitionQuery = true
          maxDegreeOfParallelism = 10
};

client.queryDocuments(collectionLink, querySpec, feedOptions)
          .toArray(function (err, results) {
          }
```

CROSS-PARTITION SDK EXAMPLE (3 of 3)

```
FeedOptions options = new FeedOptions();
options.setEnableCrossPartitionQuery(true);

Iterator<Document> it = client.queryDocuments(
    collectionLink,
    "SELECT * from r",
    options
).getQueryIterator();
```

DEMO

Cross-Partition Query

QUERY FAN OUT (1 of 3)

QUERYING ACROSS PARTITIONS IS NOT ALWAYS A BAD THING

If you have **relevant data to return**, creating a cross-partition query is a perfectly acceptable workload with a predictable throughput.

In an ideal situation, queries are **filtered to only include relevant partitions**.

BLIND QUERY FAN-OUTS CAN ADD UP

You are charged **~1 RU** for each partition that doesn't have any relevant data.

Multiple fan-out queries can quickly max out RU/s for each partition

QUERY FAN OUT (2 of 3)

CONCURRENCY AND FAN-OUT QUERIES

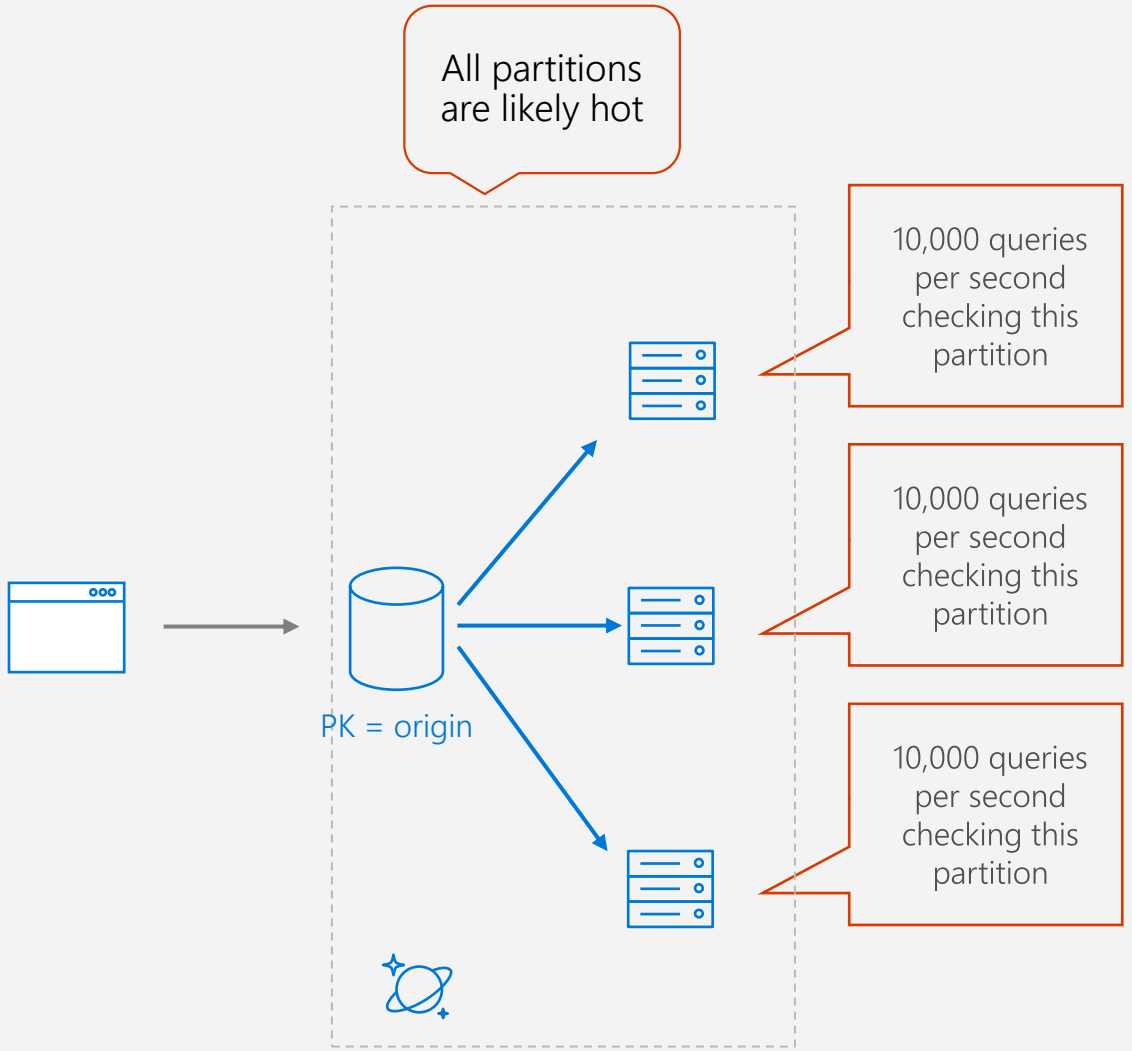
>10,000 fan-out queries in a second can leave all partitions hot

Example: Query on a vehicle database, partitioned by model name, where the query is filtering by year without **filtering to only include relevant partitions**.

```
SELECT * FROM car a WHERE a.year = “2015”
```

>10,000 more queries per second

```
SELECT * FROM car a WHERE a.year = “2016”
```

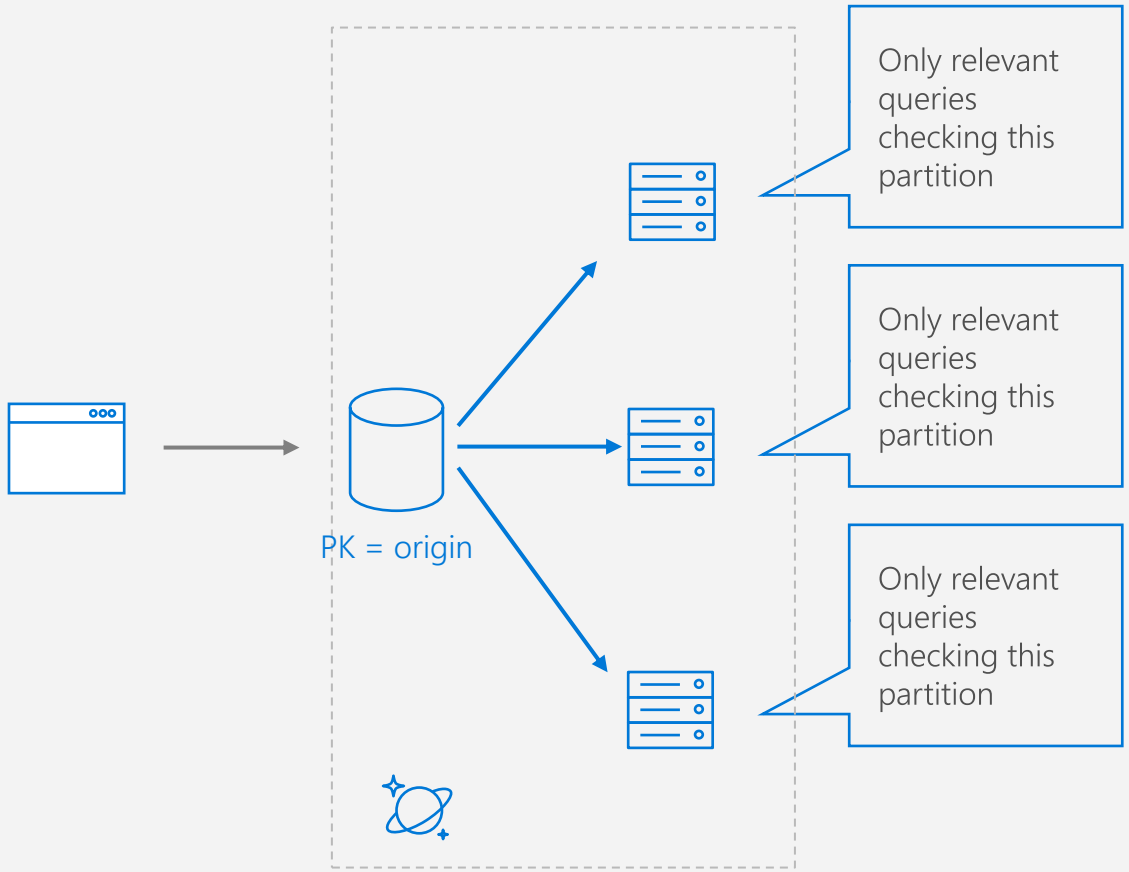


QUERY FAN OUT (3 of 3)

CONCURRENCY AND FAN-OUT QUERIES

Filtering queries to only include relevant partition key values **reduces the amount of wasted effort** and focuses queries on those relevant partitions.

```
SELECT * FROM car a
        WHERE a.model = “TURLIC” AND a.year = “2015”

                    >10,000 more queries
                         per second

SELECT * FROM car a
        WHERE a.model = “COASH” AND a.year = “2016”
```

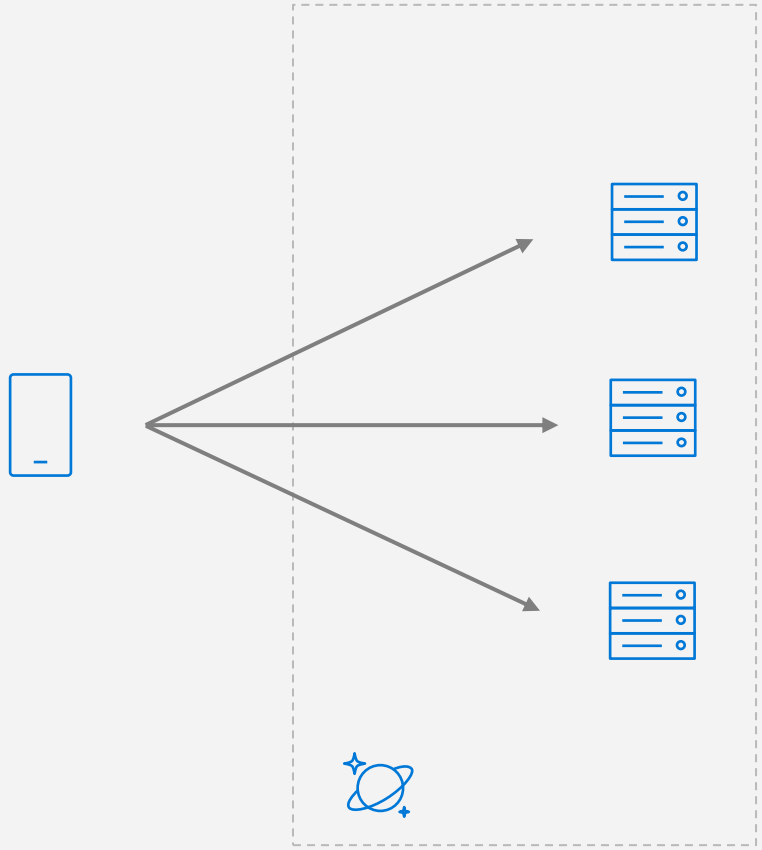


Example – Contoso Connected Car

CLIENT-SIDE QUERY FAN-OUT

USING CLIENT-SIDE THREADS AND PARRALELISM TO FAN-OUT QUERIES ACROSS PARTITIONS

- Leverage existing client device power
- Perform queries in parallel across units
- Potentially process early-finish results before parallel requests are complete



HANDS-ON EXERCISE (1 of 4)

CREATING A MULTI-PARTITION SOLUTION

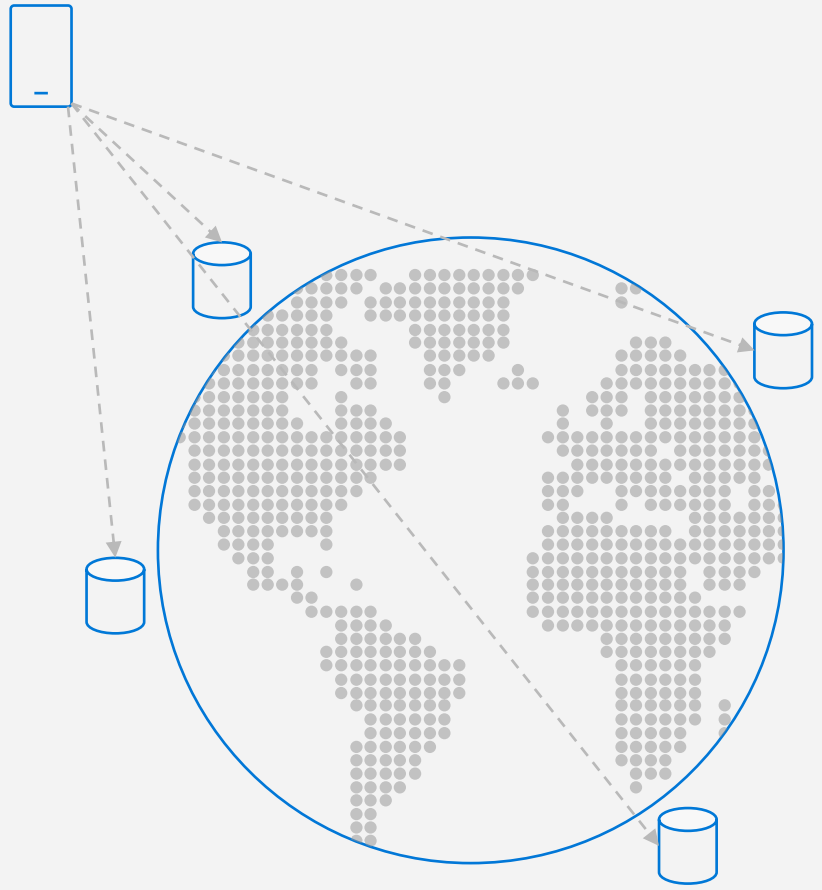
Tasks:

1. Create Unlimited Container
2. Execute Cross-Partition Queries

QUERYING

Tuning query techniques and parameters to make the most efficient use of a globally distributed database service.

This module will reference querying in the context of the SQL API for Azure Cosmos DB



QUERY TUNING (1 of 2)

MULTIPLE THINGS CAN IMPACT THE PERFORMANCE OF A QUERY RUNNING IN AZURE COSMOS DB. A FEW IMPORTANT QUERY PERFORMANCE FACTORS INCLUDE:

Provisioned throughput

Measure RU per query, and ensure that you have the required provisioned throughput for your queries

Partitioning and partition keys

Favor queries with the partition key value in the filter clause for low latency

SDK and query options

Follow SDK best practices like direct connectivity, and tune client-side query execution options

QUERY TUNING (2 of 2)

MANY THINGS CAN IMPACT THE PERFORMANCE OF A QUERY RUNNING IN AZURE COSMOS DB. IMPORTANT PERFORMANCE FACTORS INCLUDE:

Network latency

Account for network overhead in measurement, and use multi-homing APIs to read from the nearest region

Indexing Policy

Ensure that you have the required indexing paths/policy for the query

Query Complexity

Use simple queries to enable greater scale.

Query execution metrics

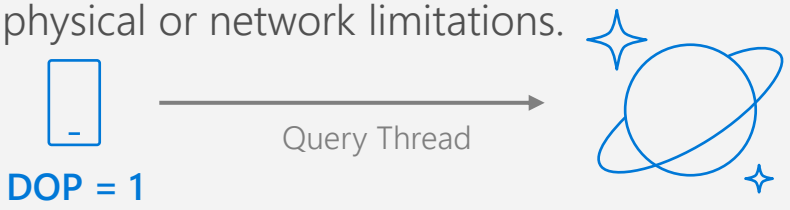
Analyze the query execution metrics to identify potential rewrites of query and data shapes

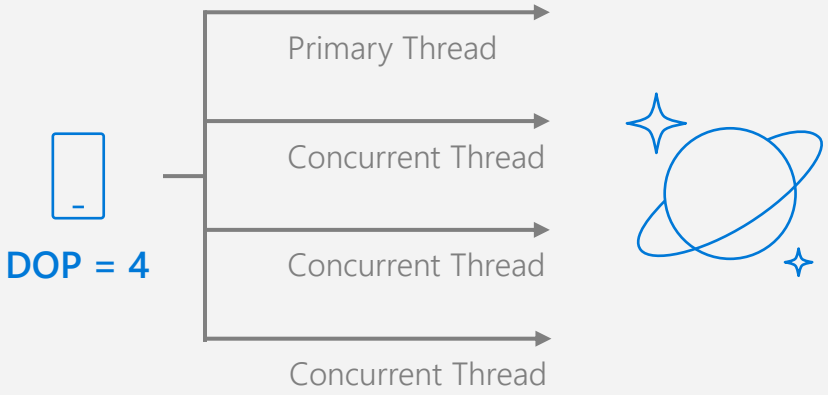
CLIENT QUERY PARALLELISM

CROSS-PARTITOIN QUERIES CAN BE PARRALELIZED TO USE AS MANY THREADS AS POSSIBLE

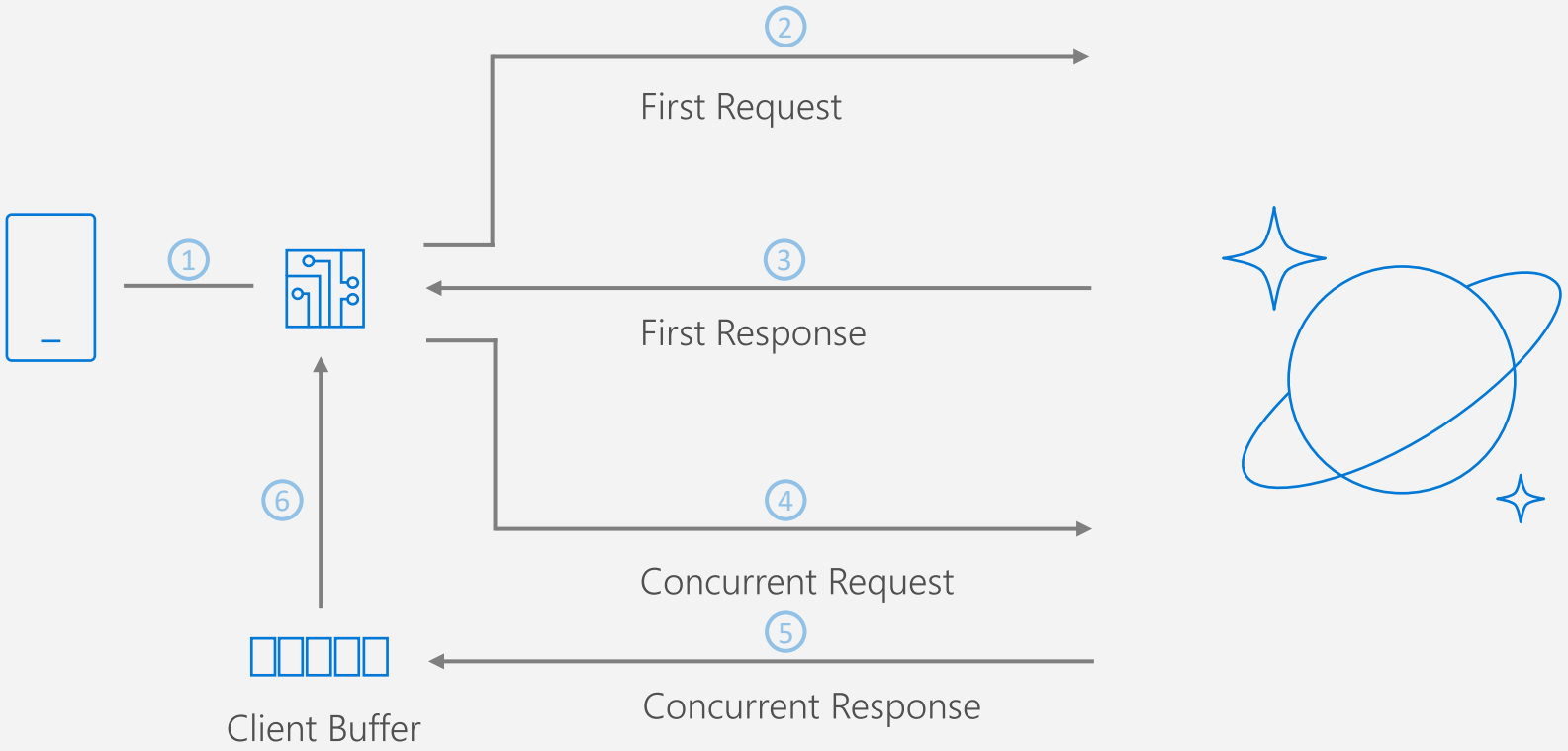
Modern processors ship with both physical and virtual (hyper-threading) cores. For any given cross-partition query, the SDK can use concurrent threads to issue the query across the underlying partitions.

By default, the SDK uses a **slow start algorithm** for cross-partition queries, increasing the amount of threads over time. This increase is exponential up to any physical or network limitations.



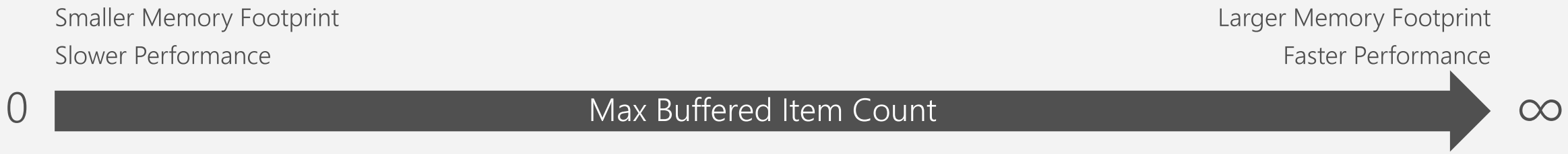


CLIENT RESPONSE BUFFER



SDK QUERY OPTIONS (1 of 2)

ACHIEVING OPTIMAL PERFORMANCE IS OFTEN A BALANCING ACT BETWEEN THESE TWO PROPERTIES



SDK QUERY OPTIONS (2 of 2)

Setting	Value	Effect
MaxDegreeofParallelism	`-1`	The system will automatically decide the number of items to buffer
MaxDegreeofParallelism	`0`	Do not add any additional concurrent threads
MaxDegreeofParallelism	`>= 1`	Add the specified number of additional concurrent threads
MaxBufferedItemCount	`-1`	The system will automatically decide the number of concurrent operations to run
MaxBufferedItemCount	`0`	Do not maintain a client-side buffer
MaxBufferedItemCount	`>= 1`	Specify the maximum size (items) of the client-side buffer

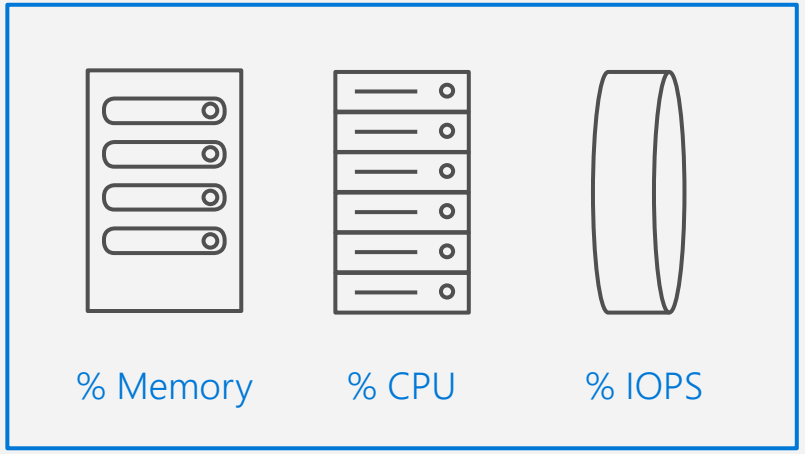
REQUEST UNITS (1 of 3)

Request Units (RUs) is a rate-based currency

Abstracts physical resources for performing requests

Key to multi-tenancy, SLAs, and COGS efficiency

Foreground and background activities



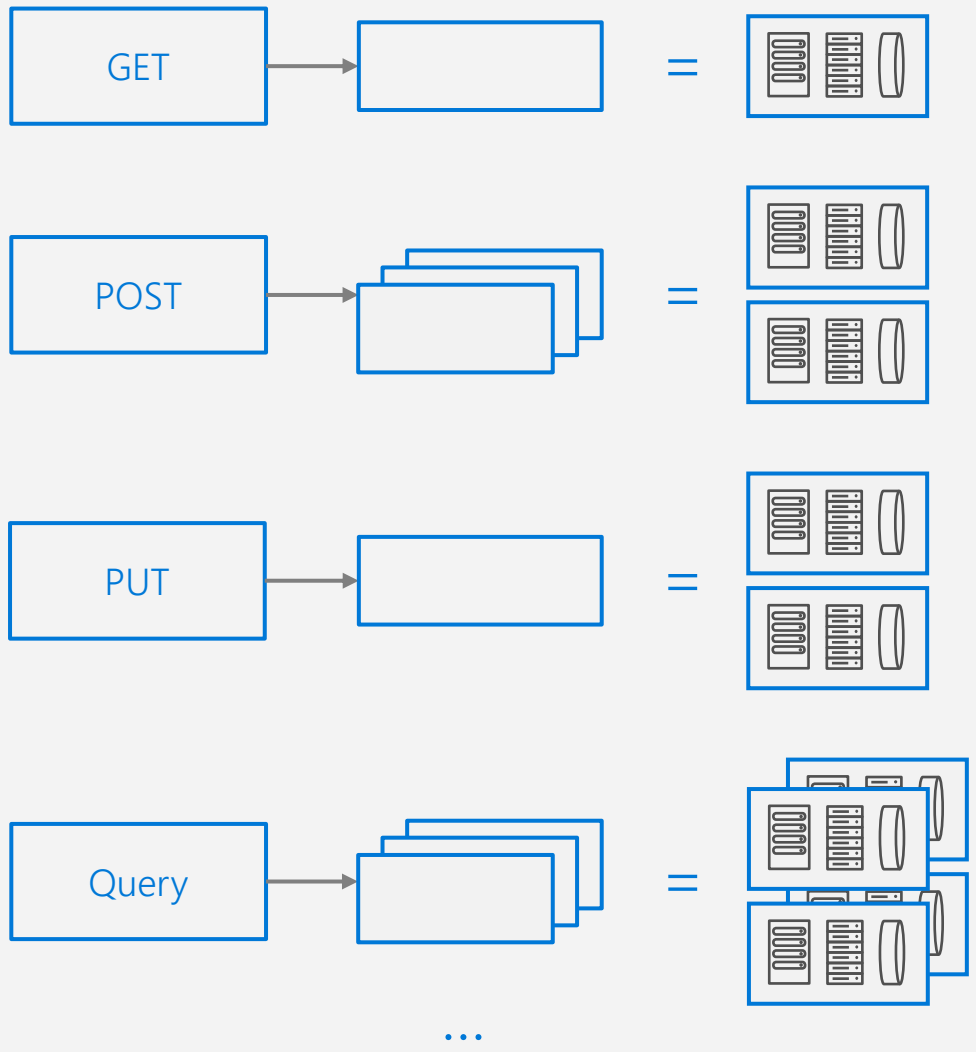
REQUEST UNITS (2 of 3)

Normalized across various access methods

1 RU = 1 read of 1 KB document

Each request consumes fixed RUs

Applies to reads, writes, query, and stored procedures



REQUEST UNITS (3 of 3)

Provisioned in terms of RU/sec

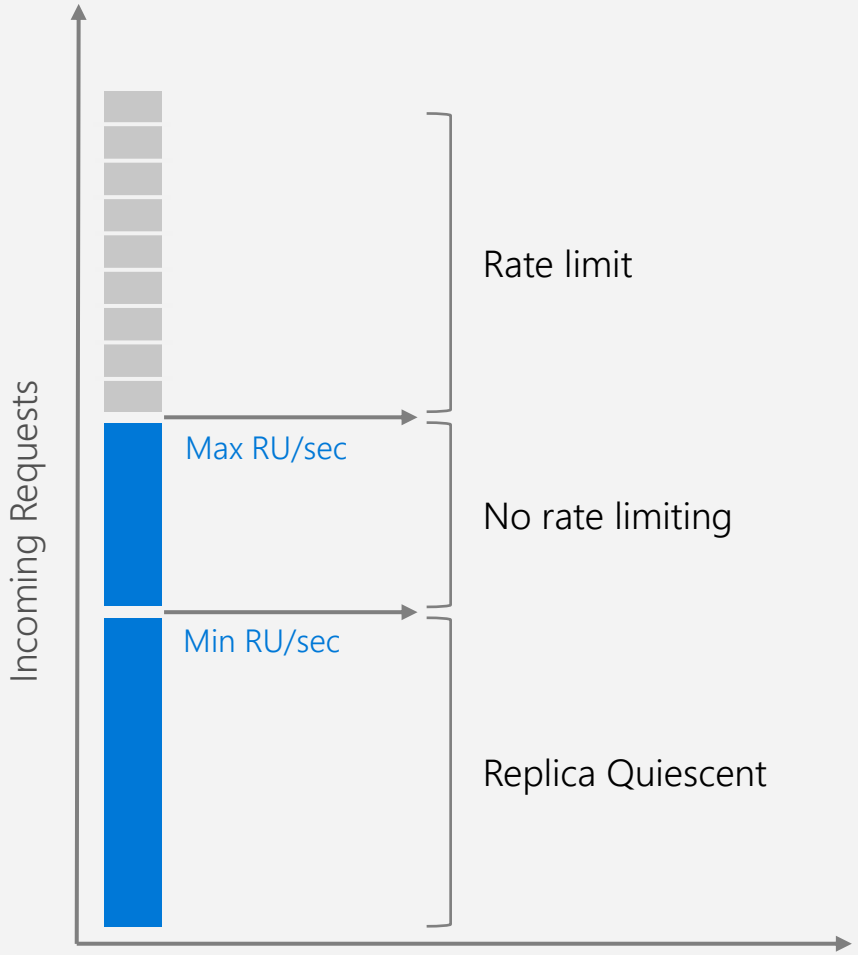
Rate limiting based on amount of throughput provisioned

Can be increased or decreased instantaneously

Metered Hourly

Background processes like TTL expiration, index

transformations scheduled when quiescent



MEASURING RU CHARGE (1 of 2)

ANALYZE QUERY COMPLEXITY

The complexity of a query impacts how many Request Units are consumed for an operation. The number of predicates, nature of the predicates, number of system functions, and the number of index matches / query results all influence the cost of query operations.

MEASURE QUERY COST

To measure the cost of any operation (create, update, or delete):

- Inspect the x-ms-request-charge header
- Inspect the RequestCharge property in ResourceResponse or FeedResponse in the SDK

NUMBER OF INDEXED TERMS IMPACTS WRITE RU CHARGES

Every write operation will require the indexer to run. The more indexed terms you have, the more indexing will be directly having an effect on the RU charge.

You can optimize for this by fine-tuning your index policy to include only fields and/or paths certain to be used in queries.

MEASURING RU CHARGE (2 of 2)

STABILIZED LOGICAL CHARGES

Azure Cosmos DB uses information about past runs to produce a stable logical charge for the majority of CRUD or query operations.

Since this stable charge exists, we can rely on our operations having a **high degree of predictability** with very little variation. We can use the predictable RU charges for future capacity planning.

BULK OF QUERY RU CHARGES IS IO

Query RU is directly proportional to the quantity of query results.

RU CHARGE MEASUREMENT EXAMPLE (1 of 3)

```
ResourceResponse<Document> response = await client.CreateDocumentAsync(
        collectionLink,
        document
);
var requestUnits = response.RequestCharge;
```

.NET

RU CHARGE MEASUREMENT EXAMPLE (2 of 3)

```
client.createDocument(
        collectionLink,
        documentDefinition,
        function (err, document, headers) {
                if (err) {
                        console.log(err);
                }
                var requestData = headers['x-ms-request-charge'];
        }
);
```

RU CHARGE MEASUREMENT EXAMPLE (3 of 3)

```
ResourceResponse<Document> response = client.createDocument(
    collectionLink,
    documentDefinition,
    null,
    false
);

Double requestCharge = response.getRequestCharge();
```

REQUEST UNIT PRICING EXAMPLE

Storage Cost

Avg Record Size (KB)	1
Number of Records	100,000
Total Storage (GB)	100
Monthly Cost per GB	$0.25
Expected Monthly Cost for Storage	$25.00

Throughput Cost

Operation Type	Number of Requests per Second	Avg RU's per Request	RU's Needed
Create	100	5	500
Read	400	1	400

Total RU/sec	900
Monthly Cost per 100 RU/sec	$6.00
Expected Monthly Cost for Throughput	$54.00

Total Monthly Cost

[Total Monthly Cost] = [Monthly Cost for Storage] + [Monthly Cost for Throughput]
= $25 + $54
= $79 per month

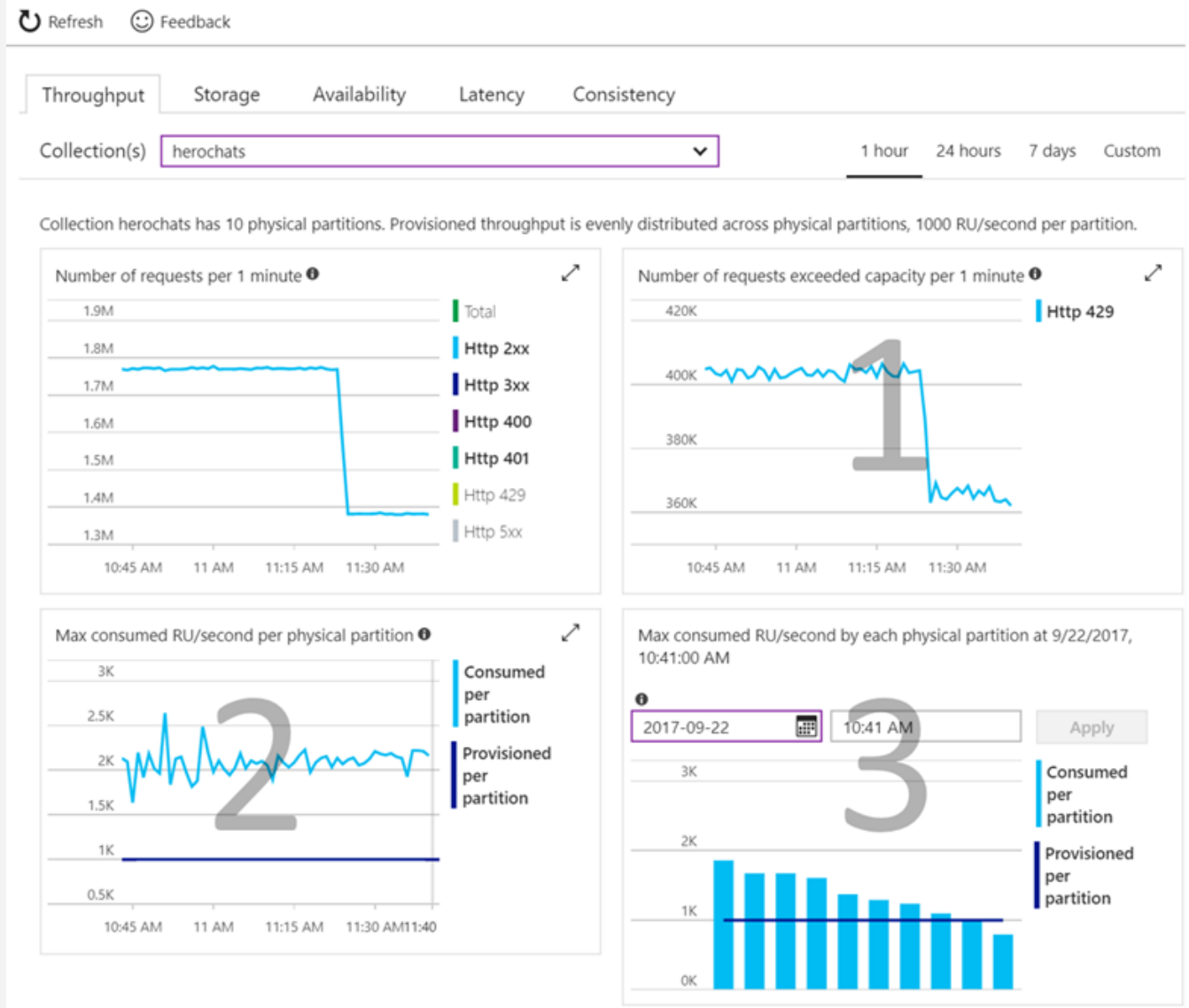
* pricing may vary by region; for up-to-date pricing, see: https://azure.microsoft.com/pricing/details/cosmos-db/

DEMO

Tuning a Query

VALIDATING THROUGHPUT LEVEL CHOICE

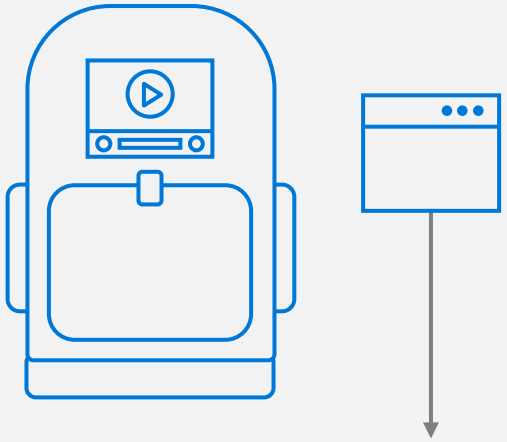
1. Check if your operations are getting rate limited.
    - Requests exceeding capacity chart
2. Check if consumed throughput exceeds the provisioned throughput on any of the physical partitions
    - Max RU/second consumed per partition chart
3. Select the time where the maximum consumed throughput per partition exceeded provisioned on the chart
    - Max consumed throughput by each partition chart



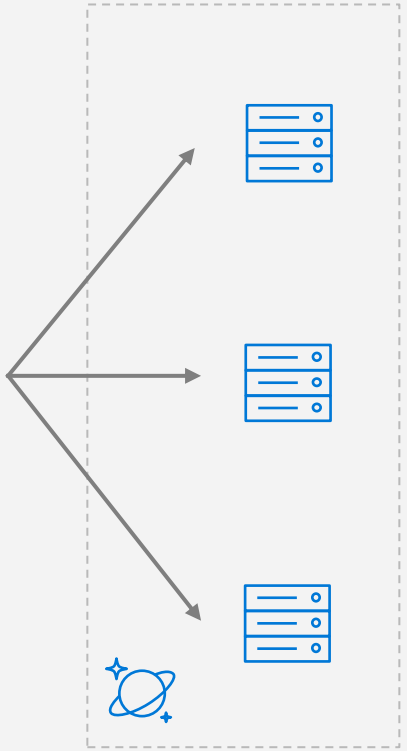
CROSS-PARTITION AGGREGATE

LOW-LATENCY AGGREGATION WORKS ACROSS MULTIPLE PARTITIONS

You can submit a simple SQL query and Azure Cosmos DB handles the routing of the query among data partitions and merges results to return the final aggregate values.



```
SELECT VALUE
    COUNT(1)
FROM
    telemetry t
WHERE
    t.deviceId = "craft267_seat17"
```



Example – Contoso Connected Car

BOUNDED LOCATION SEARCH USING GEO-DATA

GEOJSON SPECIFICATION

Azure Cosmos DB supports indexing and querying of geospatial point data that's represented using the GeoJSON specification.

SEARCH BY DISTANCE FROM POINT

The **ST_DISTANCE** built-in function returns the distance between the two GeoJSON Point expressions.

SEARCH WITHOUT BOUNDED POLYGON

The **ST_WITHIN** built-in function returns a Boolean indicating whether the first GeoJSON Point expression is within a GeoJSON Polygon expression.

```
{
        "type":"Point",
        "coordinates":[ 31.9, -4.8 ]
}

{
        "type":"Polygon",
        "coordinates":[[
                [ 31.8, -5 ],
                [ 31.8, -4.7 ],
                [ 32, -4.7 ],
                [ 32, -5 ],
                [ 31.8, -5 ]
    ]]
}
```

DISTANCE FROM CENTER POINT SEARCH

ST_DISTANCE

ST_DISTANCE can be used to measure the distance between two points. Commonly this function is used to determine if a point is within a specified range (meters) of another point.

```
SELECT *
FROM flights f
WHERE ST_DISTANCE(f.origin.location, {
        "type": "Point",
        "coordinates": [-122.19, 47.36]
}) < 100 * 1000
```

POLYGON SHAPE SEARCH

ST_WITHIN

ST_WITHIN can be used to check if a point lies within a Polygon. Commonly Polygons are used to represent boundaries like zip codes, state boundaries, or natural formations.

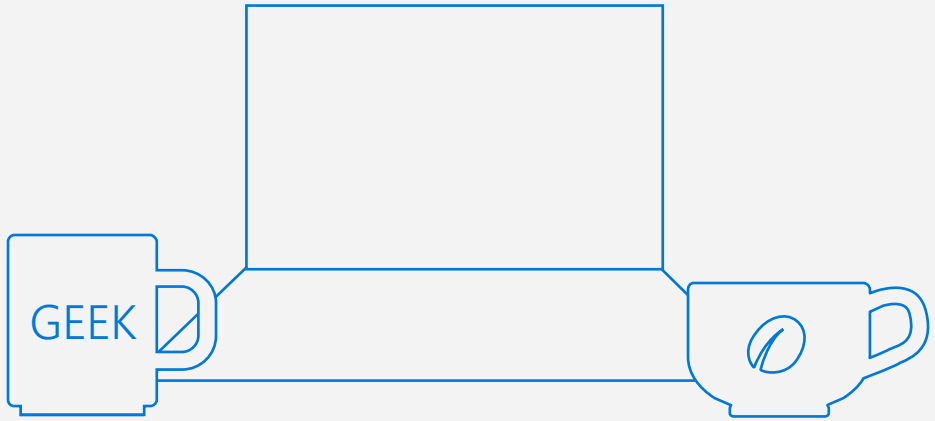
Polygon arguments in ST_WITHIN can contain only a single ring, that is, the Polygons must not contain holes in them.

```
SELECT *
FROM flights f
WHERE ST_WITHIN(f.destination.location,
{
        "type": "Polygon",
        "coordinates": [[
                [-124.63, 48.36],
                [-123.87, 46.14],
                [-122.23, 45.54],
                [-119.17, 45.95],
                [-116.92, 45.96],
                [-116.99, 49.00],
                [-123.05, 49.02],
                [-123.15, 48.31],
                [-124.63, 48.36]
        ]]
})
```

HANDLE ANY DATA WITH NO SCHEMA OR INDEXING REQUIRED (2 of 2)

Azure Cosmos DB's schema-less service automatically indexes all your data, regardless of the data model, to delivery blazing fast queries.

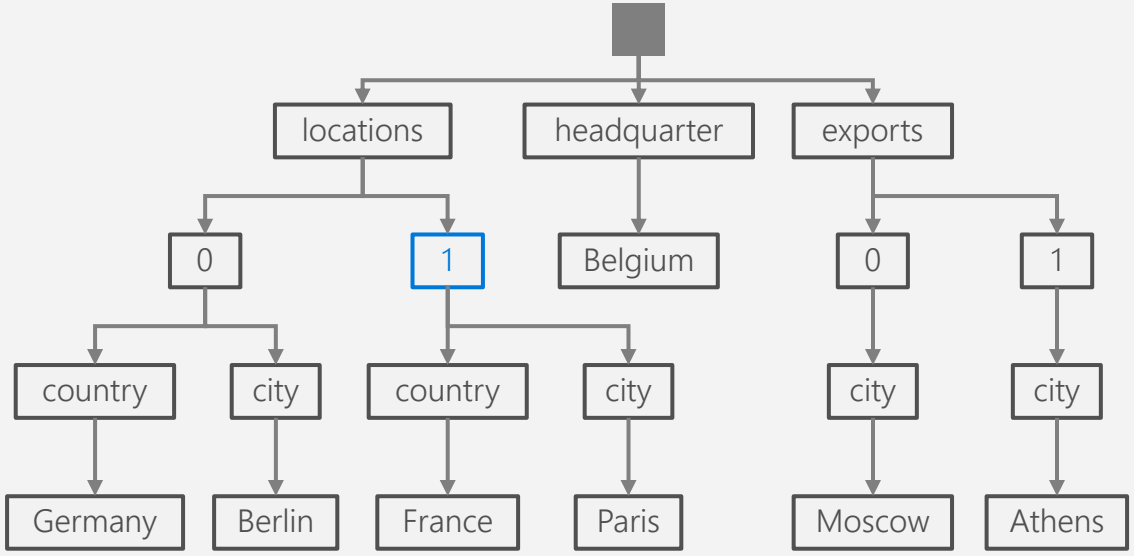
- Automatic index management
- Synchronous auto-indexing
- No schemas or secondary indices needed
- Works across every data model



Item	Color	Microwave safe	Liquid capacity	CPU	Memory	Storage
Geek mug	Graphite	Yes	16ox	???	???	???
Coffee Bean mug	Tan	No	12oz	???	???	???
Surface book	Gray	???	???	3.4 GHz Intel Skylake Core i7-6600U	16GB	1 TB SSD

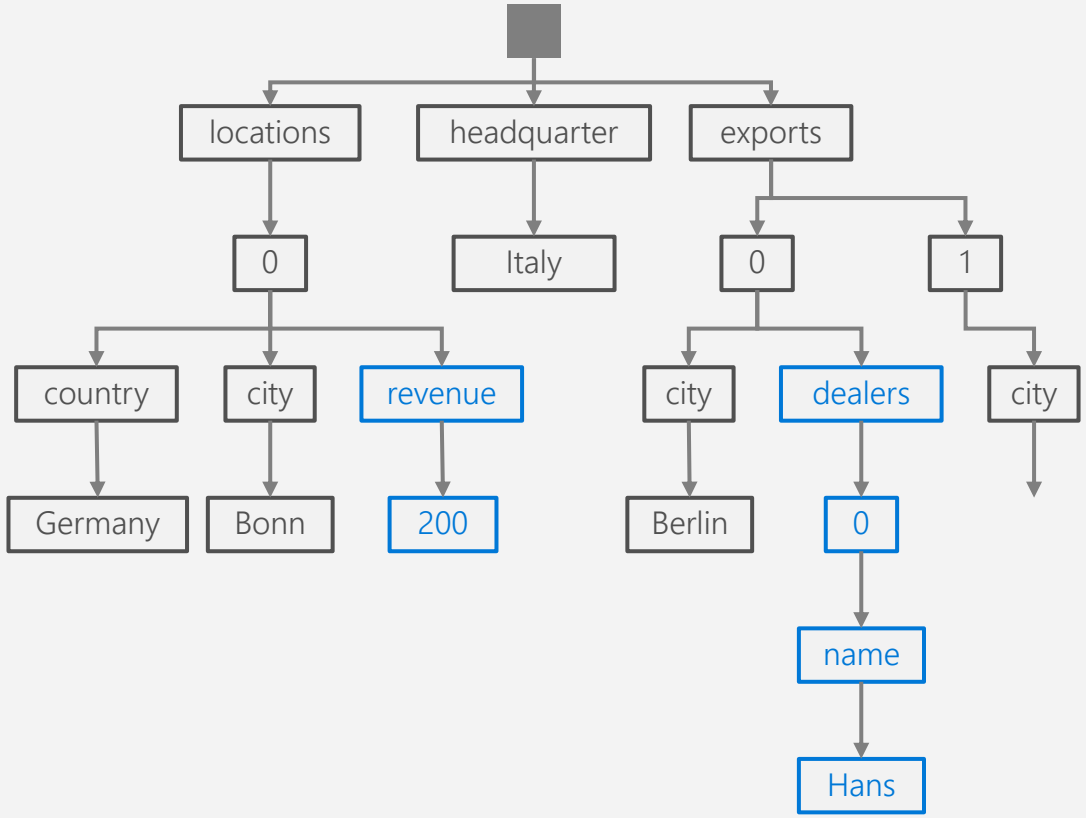
INDEXING JSON DOCUMENTS (1 of 3)

```
{
    "locations": [
        {
            "country": "Germany",
            "city": "Berlin"
        },
        {
            "country": "France",
            "city": "Paris"
        }
    ],
    "headquarter": "Belgium",
    "exports": [
        { "city": "Moscow" },
        { "city": "Athens" }
     ]
}
```

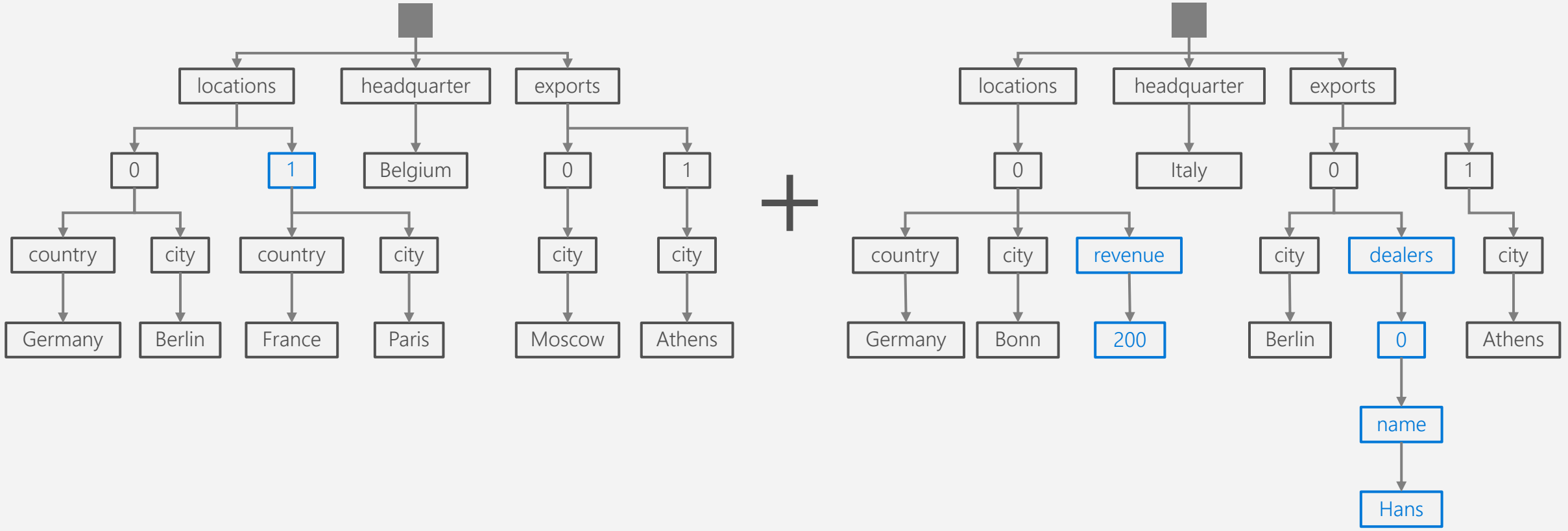


INDEXING JSON DOCUMENTS (2 of 3)

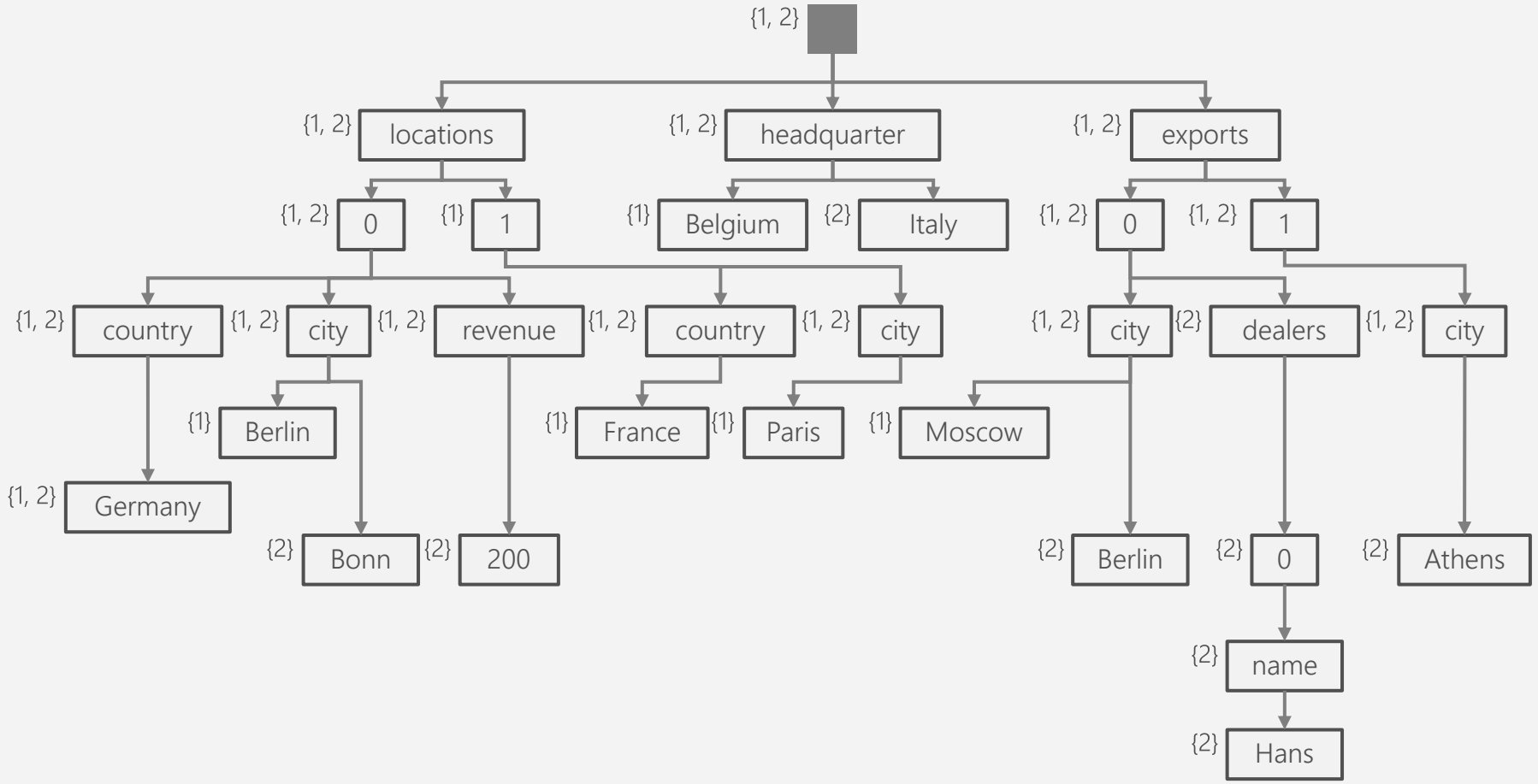
```
{
    "locations": [
        {
            "country": "Germany",
            "city": "Bonn",
            "revenue": 200
        }
    ],
    "headquarter": "Italy",
    "exports": [
        {
            "city": "Berlin",
            "dealers": [
                { "name": "Hans" }
            ]
        },
        { "city": "Athens" }
    ]
}
```



INDEXING JSON DOCUMENTS (3 of 3)



INVERTED INDEX



INDEX POLICIES

CUSTOM INDEXING POLICIES

Though all Azure Cosmos DB data is indexed by default, you can specify a custom indexing policy for your collections. Custom indexing policies allow you to design and customize the shape of your index while maintaining schema flexibility.

- Define trade-offs between storage, write and query performance, and query consistency
- Include or exclude documents and paths to and from the index
- Configure various index types

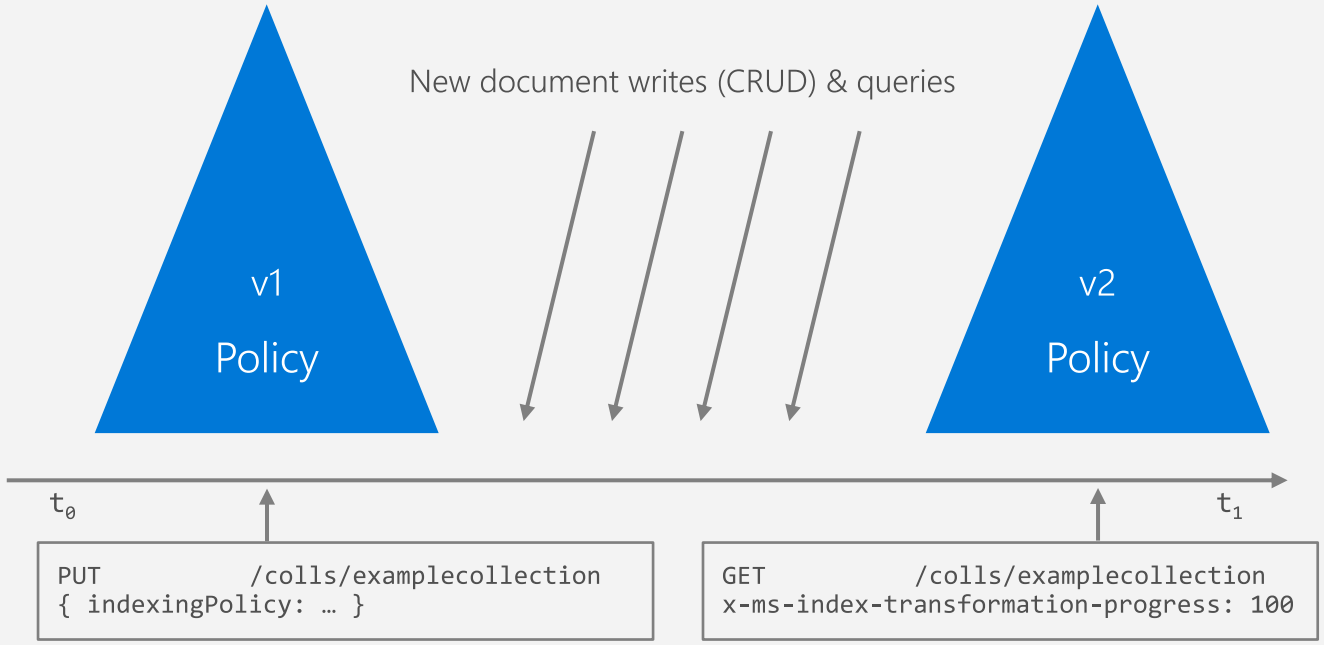
```
{
    "automatic": true,
    "indexingMode": "Consistent",
    "includedPaths": [{
        "path": "/*",
        "indexes": [{
            "kind": "Hash",
            "dataType": "String",
            "precision": -1
        }, {
            "kind": "Range",
            "dataType": "Number",
            "precision": -1
        }, {
            "kind": "Spatial",
            "dataType": "Point"
        }]
    }],
    "excludedPaths": [{
        "path": "/nonIndexedContent/*"
    }]
}
```

ONLINE INDEX TRANSFORMATIONS

ON-THE-FLY INDEX CHANGES

In Azure Cosmos DB, you can make changes to the indexing policy of a collection on the fly. Changes can affect the shape of the index, including paths, precision values, and its consistency model.

A change in indexing policy effectively requires a transformation of the old index into a new index.



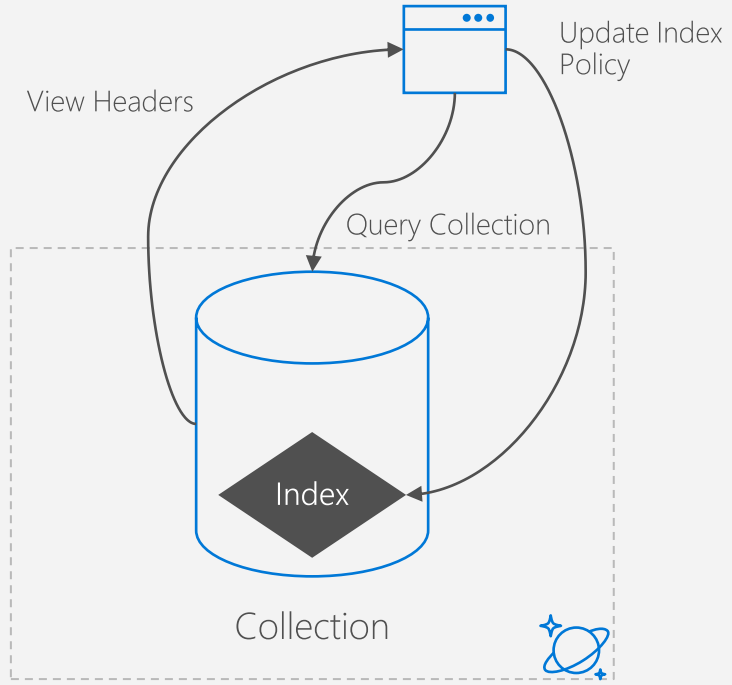
INDEX TUNING

METRICS ANALYSIS

The SQL APIs provide information about performance metrics, such as the index storage used and the throughput cost (request units) for every operation. You can use this information to compare various indexing policies, and for performance tuning.

When running a **HEAD** or **GET** request against a collection resource, the **x-ms-request-quota** and the **x-ms-request-usage** headers provide the **storage quota** and **usage** of the collection.

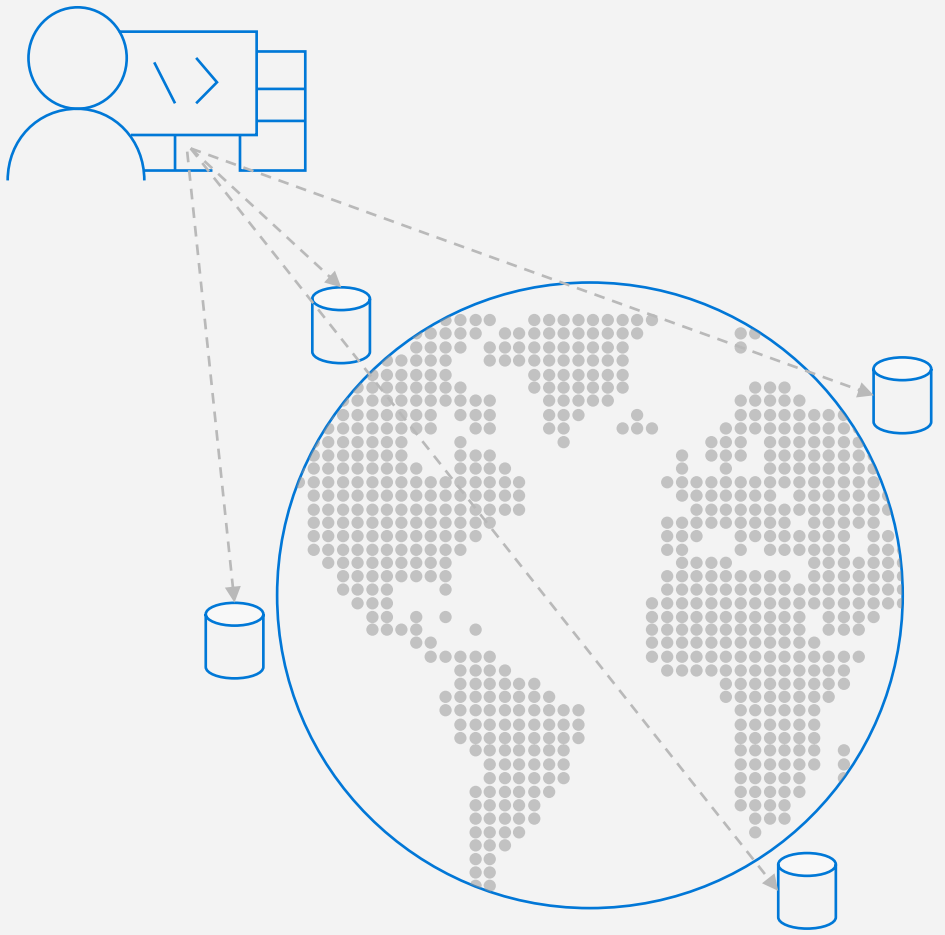
You can use this information to compare various indexing policies, and for performance tuning.



SQL SYNTAX

Using the popular query language, SQL, to access semi-structured JSON data.

This module will reference querying in the context of the SQL API for Azure Cosmos DB.



SQL QUERY SYNTAX

BASIC QUERY SYNTAX

The **SELECT** & **FROM** keywords are the basic components of every query.

```
SELECT
    tickets.id,
    tickets.pricePaid
FROM tickets
```

```
SELECT
    t.id,
    t.pricePaid
FROM tickets t
```

SQL QUERY SYNTAX - WHERE

FILTERING

WHERE supports complex scalar expressions including arithmetic, comparison and logical operators

```
SELECT
    tickets.id,
    tickets.pricePaid
FROM tickets
WHERE
    tickets.pricePaid > 500.00 AND
    tickets.pricePaid <= 1000.00
```

SQL QUERY SYNTAX - PROJECTION (1 of 2)

JSON PROJECTION

If your workloads require a specific JSON schema, Azure Cosmos DB supports JSON projection within its queries

```
SELECT {
    "id": tickets.id,
    "flightNumber": tickets.assignedFlight.flightNumber,
    "purchase": {
        "cost": tickets.pricePaid
    },
    "stops": [
        tickets.assignedFlight.origin,
        tickets.assignedFlight.destination
    ]
} AS ticket
FROM tickets
```

```
[
    {
        "ticket": {
            "id": "6ebe1165836a",
            "purchase": {
                "cost": 575.5
            },
            "stops": [
                "SEA",
                "JFK"
            ]
        }
    }
]
```

SQL QUERY SYNTAX - PROJECTION (2 of 2)

SELECT VALUE

The **VALUE** keyword can further flatten the result collection if needed for a specific application workload

```
SELECT VALUE {
    "id": tickets.id,
    "flightNumber": tickets.assignedFlight.flightNumber,
    "purchase": {
        "cost": tickets.pricePaid
    },
    "stops": [
        tickets.assignedFlight.origin,
        tickets.assignedFlight.destination
    ]
}
FROM tickets
```

```
[
    {
        "id": "6ebe1165836a",
        "purchase": {
            "cost": 575.5
        },
        "stops": [
            "SEA",
            "JFK"
        ]
    }
]
```

INTRA-DOCUMENT JOIN (1 of 4)

Azure Cosmos DB supports intra-document JOIN's for de-normalized arrays

Let's assume that we have two JSON documents in a collection:

```
{
    "pricePaid": 575.5,
    "assignedFlight": {
        "number": "F125",
        "origin": "SEA",
        "destination": "JFK"
    },
    "seat": "12A",
    "requests": [
        "kosher_meal",
        "aisle_seat"
    ],
    "id": "6ebe1165836a"
}
```

```
{
    "pricePaid": 234.75,
    "assignedFlight": {
        "number": "F752",
        "origin": "SEA",
        "destination": "LGA"
    },
    "seat": "14C",
    "requests": [
        "early_boarding",
        "window_seat"
    ],
    "id": "c4991b4d2efc"
}
```

We are interested in querying an array internal to the document

SQL

INTRA-DOCUMENT JOIN (2 of 4)

We can filter on a particular array index position without JOIN:

```
SELECT
    tickets.assignedFlight.number,
    tickets.seat,
    ticket.requests
FROM
    tickets
WHERE
    ticket.requests[1] == "aisle_seat"
```

```
[
    {
        "number":"F125","seat":"12A",
        "requests": [
            "kosher_meal",
            "aisle_seat"
        ]
    }
]
```

SQL

INTRA-DOCUMENT JOIN (3 of 4)

JOIN allows us to merge embedded documents or arrays across multiple documents and returned a flattened result set:

```
SELECT
    tickets.assignedFlight.number,
    tickets.seat,
    requests
FROM
    tickets
JOIN
    requests IN tickets.requests
```

```
[
    {
        "number":"F125","seat":"12A",
        "requests":"kosher_meal"
    },
    {
        "number":"F125","seat":"12A",
        "requests":"aisle_seat"
    },
    {
        "number":"F752","seat":"14C",
        "requests":"early_boarding"
    },
    {
        "number":"F752","seat":"14C",
        "requests":"window_seat"
    }
]
```

SQL

INTRA-DOCUMENT JOIN (4 of 4)

Along with JOIN, we can also filter the cross products without knowing the array index position:

```
SELECT
    tickets.id, requests
FROM
    tickets
JOIN
    requests IN tickets.requests
WHERE
    requests
    IN ("aisle_seat", "window_seat")
```

```
[
    {
        "number":"F125","seat":"12A“,
        "requests": "aisle_seat"
    },
    {
        "number":"F752","seat":"14C",
        "requests": "window_seat"
    }
]
```

SQL

PAGINATED QUERY RESULTS (1 of 3)

Straightforward approach to paginate the results:

```
var query = client.CreateDocumentQuery<ExampleEntity>(collectionUri, options);
var docQuery = query.AsDocumentQuery();

List<ExampleEntity> results = new List<ExampleEntity>();
while (docQuery.HasMoreResults)
{
    foreach (ExampleEntity item in await query.ExecuteNextAsync())
    {
        results.Add(item);
    }
}
```

.NET

PAGINATED QUERY RESULTS (2 of 3)

Pagination with ToList():

```
var query = client.CreateDocumentQuery<ExampleEntity>(collectionUri, options);
var docQuery = query.AsDocumentQuery();

List<ExampleEntity> results = new List<ExampleEntity>();
results = query.ToList();
```

ToList() automatically iterates through all pages

.NET

PAGINATED QUERY RESULTS (3 of 3)

Pagination with hasNext() in Java:

```
Iterator<Document> documents = client.queryDocuments(
    collectionLink,
    queryString,
    options
).getQueryIterator();

while(documents.hasNext()) {
    Document current = documents.next();
}
```

SQL QUERY PARAMETRIZATION

```
var query = client.CreateDocumentQuery<ExampleEntity>(collectionUri,
    new SqlQuerySpec
    {
        QueryText = "SELECT * FROM dataset s WHERE (s.id = @id)",
        Parameters = new SqlParameterCollection
        {
            new SqlParameter("@id", “exampleIdentifier")
        }
    }, options
);

List<ExampleEntity> results = new List<ExampleEntity>();
results = query.ToList<ExampleEntity>();
```

.NET

SQL QUERY IN LINQ

```
var query = client.CreateDocumentQuery<ExampleEntity>(collectionUri, options);

var docQuery = query
                .Where(s => s.LastName == "Andersen")
                .Select(s => new { Name = s.LastName })
                .AsDocumentQuery();

List<ExampleEntity> results = new List<ExampleEntity>();
results = query.ToList<ExampleEntity>();
```

.NET

HANDS-ON EXERCISE (2 of 4)

QUERYING THE DATABASE USING SQL

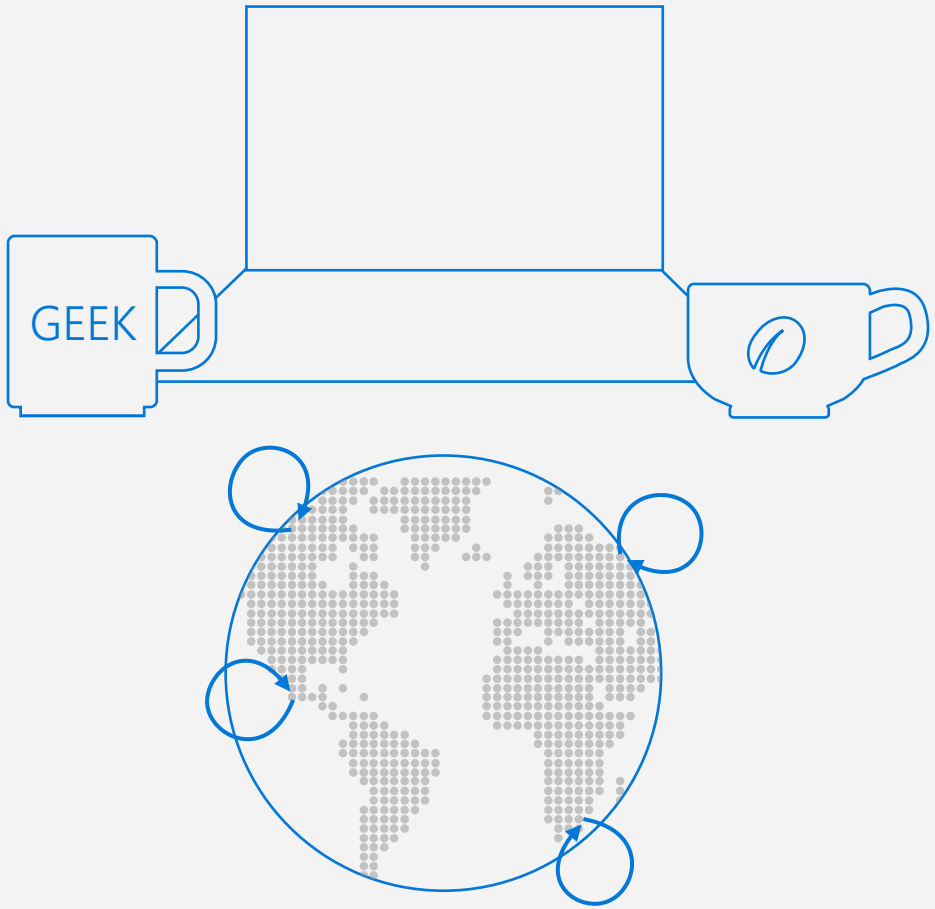
Tasks:

1. Executing Simple Queries
2. Running Intra-document Queries
3. Projecting Query Results

PROGRAMMING

Run native JavaScript server-side programming logic to performic atomic multi-record transactions.

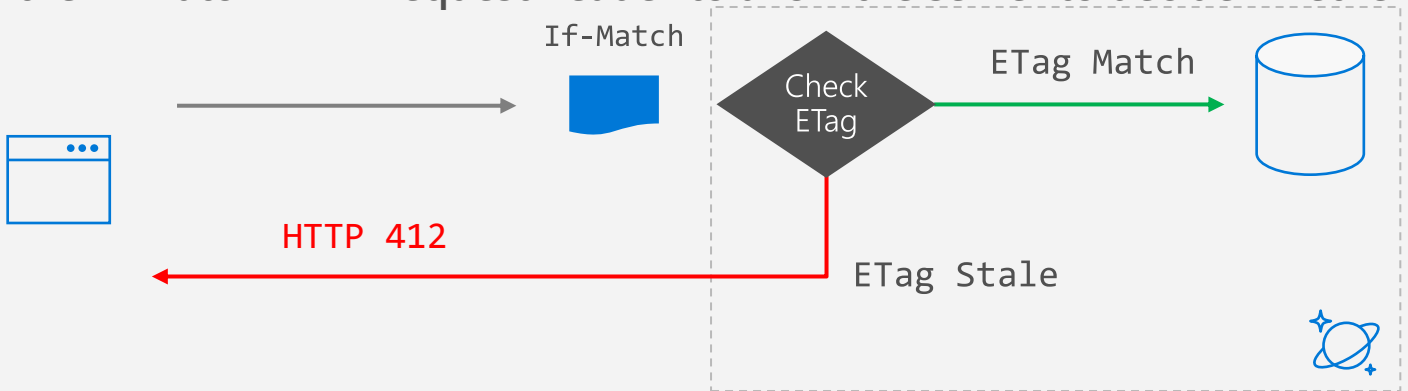
This module will reference programming in the context of the SQL API.



CONTROL CONCURRENCY USING ETAGS (1 of 2)

OPTIMISTIC CONCURRENCY

- The SQL API supports optimistic concurrency control (OCC) through HTTP entity tags, or ETags
- Every SQL API resource has an ETag system property, and the ETag value is generated on the server every time a document is updated.
- If the ETag value stays constant – that means no other process has updated the document. If the ETag value unexpectedly mutates – then another concurrent process has updated the document.
- **ETags can be used with the If-Match HTTP request header to allow the server to decide whether a resource should be updated:**



CONTROL CONCURRENCY USING ETAGS (2 of 2)

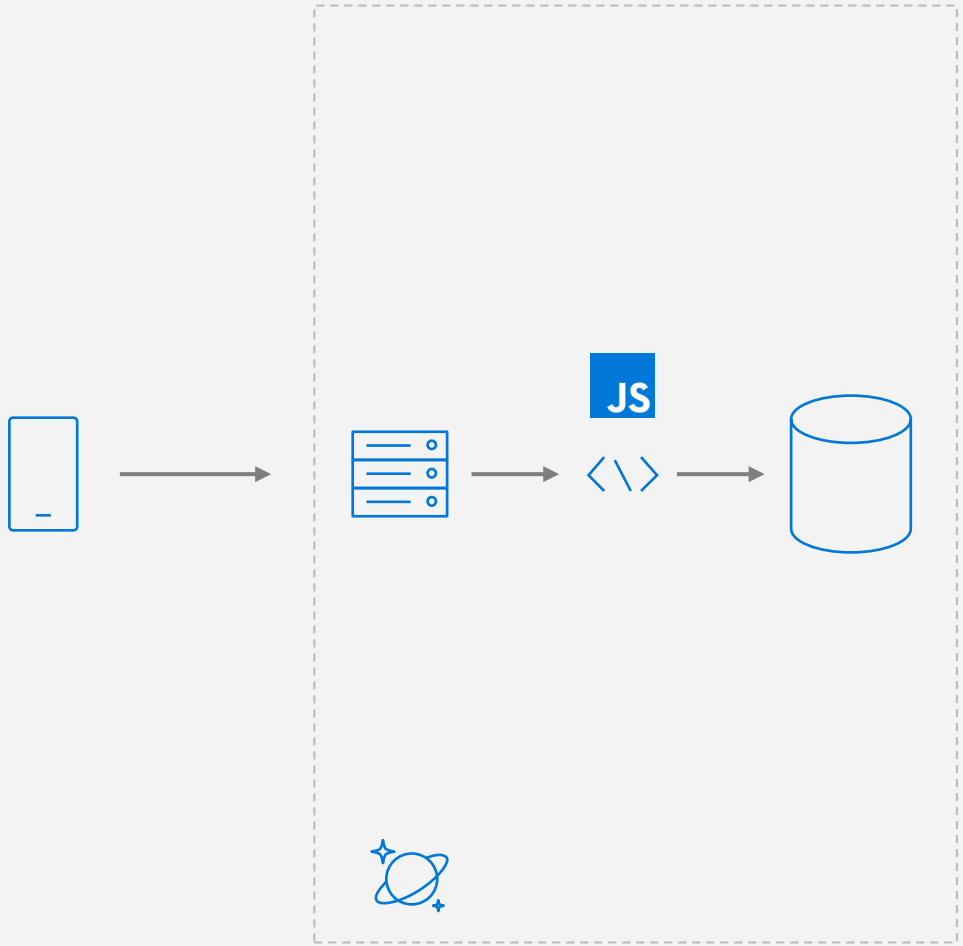
```
try
{
    var ac = new AccessCondition { Condition = readDoc.ETag, Type = AccessConditionType.IfMatch };
    updatedDoc = await client.ReplaceDocumentAsync(readDoc, new RequestOptions { AccessCondition = ac });
}
catch (DocumentClientException dce)
{
    if (dce.StatusCode == HttpStatusCode.PreconditionFailed)
    {
        Console.WriteLine("Another concurrent process has updated the record");
    }
}
```

.NET

STORED PROCEDURES

BENEFITS

- Familiar programming language
- Atomic Transactions
- Built-in Optimizations
- Business Logic Encapsulation



SIMPLE STORED PROCEDURE

.NET SDK v2 | .NET SDK v3 | Java SDK | JavaScript SDK | Python SDK

The following example shows how to register a stored procedure by using the JavaScript SDK

JavaScript Copy

```
const container = client.database("myDatabase").container("myContainer");
const sprocId = "spCreateToDoItems";
await container.scripts.storedProcedures.create({
    id: sprocId,
    body: require(`../js/${sprocId}`)
});
```

The following code shows how to call a stored procedure by using the JavaScript SDK:

JavaScript Copy

```
const newItem = [{
    category: "Personal",
    name: "Groceries",
    description: "Pick up strawberries",
    isComplete: false
}];
const container = client.database("myDatabase").container("myContainer");
const sprocId = "spCreateToDoItems";
const {resource: result} = await container.scripts.storedProcedure(sprocId).execute(newItem, {partitionKe
```

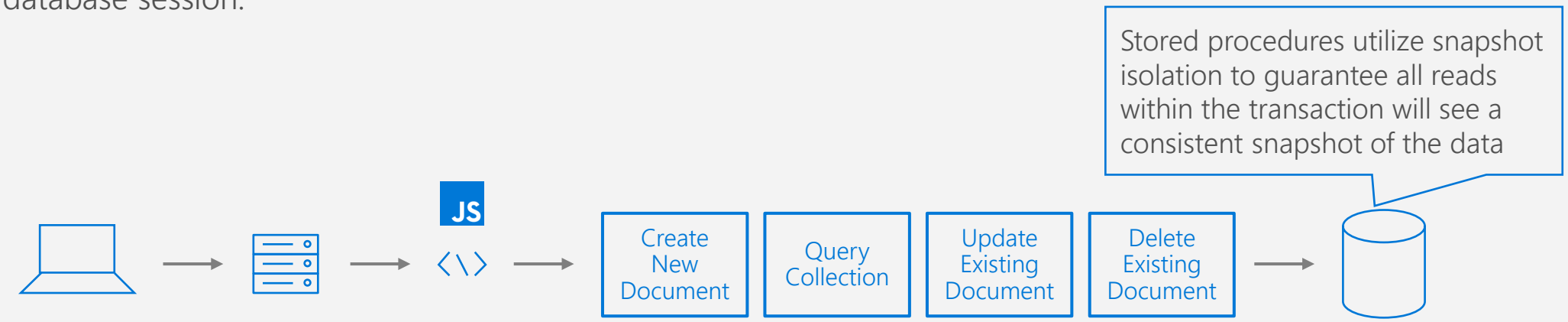
```
function createSampleDocument(documentToCreate) {
            var context = getContext();
            var collection = context.getCollection();
            var accepted = collection.createDocument(
                    collection.getSelfLink(),
                    documentToCreate,
                    function (error, documentCreated) {

            context.getResponse().setBody(documentCreated.id)
                    }
            );
            if (!accepted) return;
}
```

MULTI-DOCUMENT TRANSACTIONS

DATABASE TRANSACTIONS

In a typical database, a transaction can be defined as a sequence of operations performed as a single logical unit of work. Each transaction provides ACID guarantees.

In Azure Cosmos DB, JavaScript is hosted in the same memory space as the database. Hence, requests made within stored procedures and triggers execute in the same scope of a database session.



BOUNDED EXECUTION

EXECUTION WITHIN TIME BOUNDARIES

All Azure Cosmos DB operations must complete within the server-specified request timeout duration. If an operation does not complete within that time limit, the transaction is rolled back.

HELPER BOOLEAN VALUE

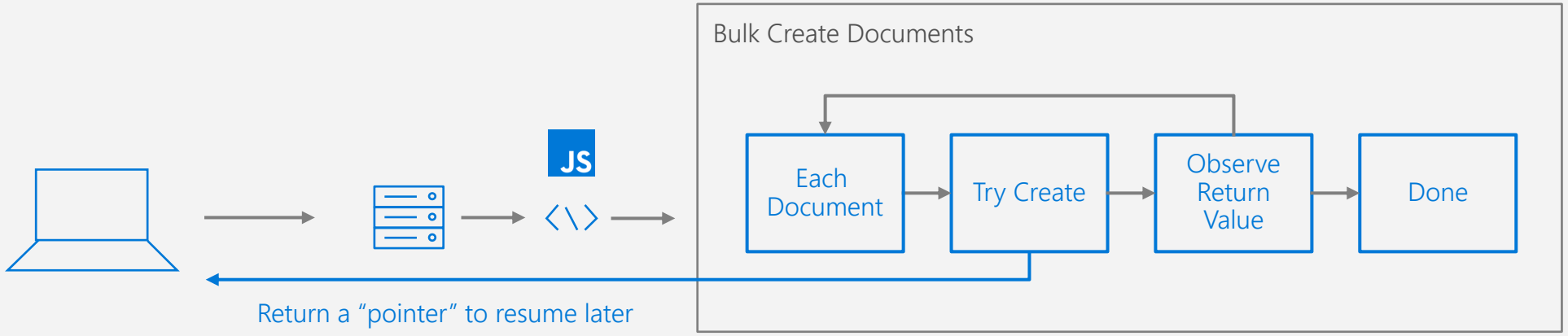
All functions under the collection object (for create, read, replace, and delete of documents and attachments) return a **Boolean value** that represents whether that operation will complete:

- **If true**, the operation is expected to complete
- **If false**, the time limit will soon be reached and your function should end execution as soon as possible.

TRANSACTION CONTINUATION MODEL

CONTINUING LONG-RUNNING TRANSACTIONS

- JavaScript functions can implement a continuation-based model to batch/resume execution
- The continuation value can be any value of your own choosing. This value can then be used by your applications to **resume a transaction from a new "starting point"**



CONTROL FLOW

JAVASCRIPT CONTROL FLOW

Stored procedures allow you to naturally express control flow, variable scoping, assignment, and integration of exception handling primitives with database transactions directly in terms of the JavaScript programming language.

ES6 PROMISES

ES6 promises can be used to implement promises for Azure Cosmos DB stored procedures. Unfortunately, **promises "swallow" exceptions by default**. It is recommended to use callbacks instead of ES6 promises.

STORED PROCEDURE CONTROL FLOW (1 of 2)

```
function createTwoDocuments(docA, docB) {
    var ctxt = getContext(); var coll = context.getCollection(); var collLink = coll.getSelfLink();
    var aAccepted = coll.createDocument(collLink, docA, docACallback);
    function docACallback(error, created) {
        var bAccepted = coll.createDocument(collLink, docB, docBCallback);
        if (!bAccepted) return;
    };
    function docBCallback(error, created) {
        context.getResponse().setBody({
            "firstDocId": created.id,
            "secondDocId": created.id
       });
    };
}
```

STORED PROCEDURE CONTROL FLOW (2 of 2)

```
function createTwoDocuments(docA, docB) {
    var ctxt = getContext(); var coll = context.getCollection(); var collLink = coll.getSelfLink();
    var aAccepted = coll.createDocument(collLink, docA, function(docAError, docACreated) {
        var bAccepted = coll.createDocument(collLink, docB, function(docBError, docBCreated) {
            context.getResponse().setBody({
                "firstDocId": docACreated.id,
                "secondDocId": docBCreated.id
            });
        });
        if (!aAccepted) return;
    });
    if (!bAccepted) return;
}
```

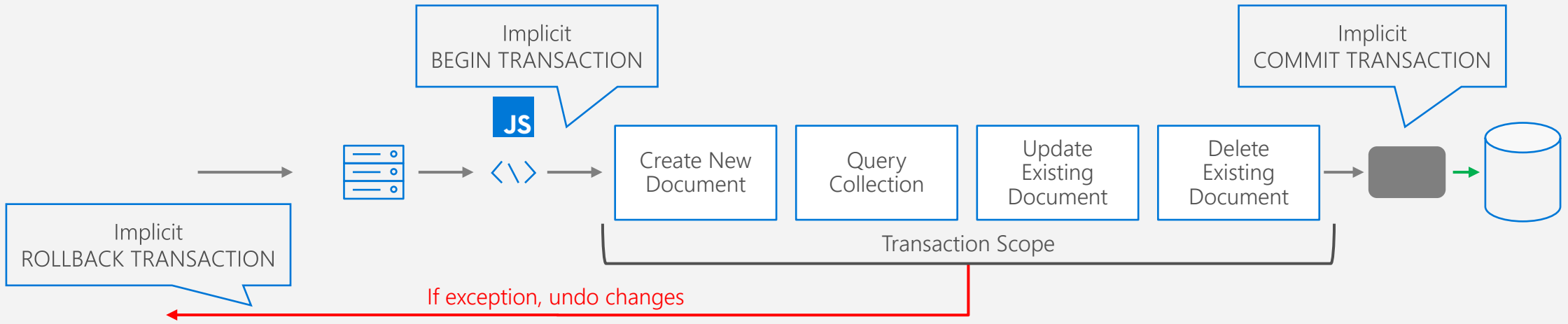
Nesting your callbacks is just as valid of a method as using named callback functions

ROLLING BACK TRANSACTIONS

TRANSACTION ROLL-BACK

Inside a JavaScript function, all operations are automatically wrapped under a single transaction:

- If the function completes without any exception, all data changes are committed
- If there is any exception that's thrown from the script, Azure Cosmos DB's JavaScript runtime will roll back the whole transaction.



TRANSACTION ROLLBACK IN STORED PROCEDURE

```
collection.createDocument(
        collection.getSelfLink(),
        documentToCreate,
        function (error, documentCreated) {
                if (error) throw "Unable to create document, aborting...";
        }
);
collection.createDocument(
        documentToReplace._self,
        replacementDocument,
        function (error, documentReplaced) {
                if (error) throw "Unable to update document, aborting...";
        }
);
```

DEBUGGING STORED PROCEDURES (1 of 3)

CONSOLE LOGGING

Much like with traditional JavaScript applications, you can use **console.log()** to capture various telemetry and data points for your running code.

VIEWING SCRIPT LOGS

.NET

You must **opt-in** to viewing and capturing console output using the **EnableScriptLogging** boolean property available in the client SDK. The SDK has a **ScriptLog** property on the **StoredProcedureResponse** class that contains the captured output of the JavaScript console log.

DEBUGGING STORED PROCEDURES (2 of 3)

```
var response = await client.ExecuteStoredProcedureAsync(
    document.SelfLink,
    new RequestOptions
    {
        EnableScriptLogging = true
    }
);
String logData = response.ScriptLog;
```

.NET

DEBUGGING STORED PROCEDURES (3 of 3)

```
RequestOptions requestOptions = new RequestOptions();
requestOptions.EnableScriptLogging = true;

StoredProcedureResponse response = client.executeStoredProcedure(
    storedProcLink,
    requestOptions,
    new Object[]{}
);
```

DEMO

Authoring Stored Procedures

USER-DEFINED FUNCTIONS

UDF

- User-defined functions (UDFs) are used to extend the Azure Cosmos DB SQL API's query language grammar and implement custom business logic. UDFs can only be called from inside queries

- They do not have access to the context object and are meant to be used as compute-only code

USER-DEFINED FUNCTION DEFINITION

```
var taxUdf = {
        id: "tax",
        serverScript: function tax(income) {
                if (income == undefined)
                        throw 'no input';
                if (income < 1000)
                        return income * 0.1;
                else if (income < 10000)
                        return income * 0.2;
                else
                        return income * 0.4;
        }
}
```

USER-DEFINED FUNCTION USAGE IN QUERIES

```
SELECT
	*
FROM
	TaxPayers t
WHERE
	udf.tax(t.income) > 20000
```

SQL

DEMO

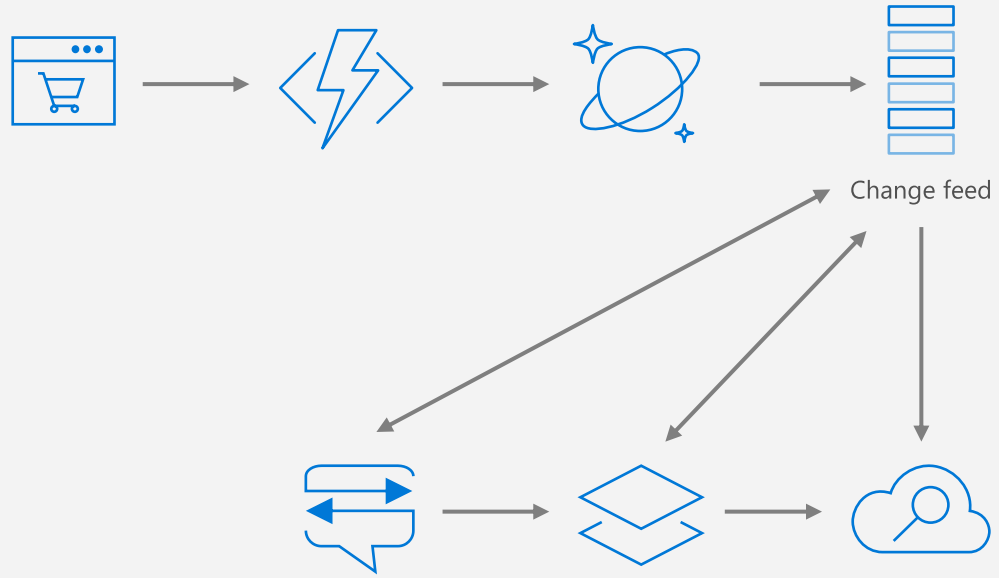
Authoring User-Defined Functions

MODERN REACTIVE APPLICATIONS

IoT, gaming, retail and operational logging applications need to **track and respond to tremendous amount of data** being ingested, modified or removed from a globally-scaled database.

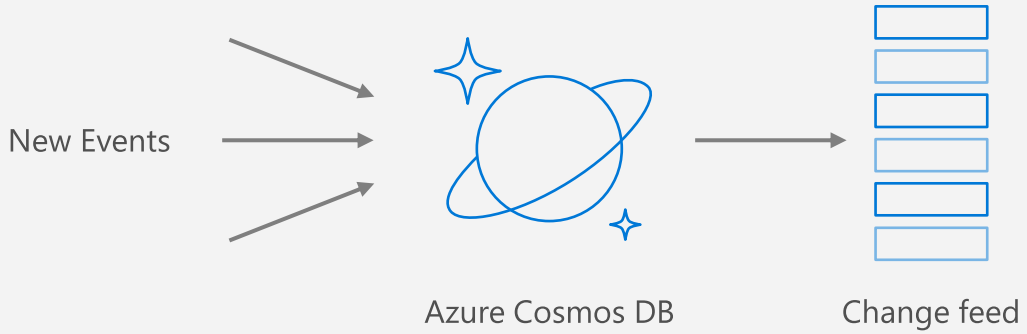
COMMON SCENARIOS

- Trigger notification for new items
- Perform real-time analytics on streamed data
- Synchronize data with a cache, search engine or data warehouse.

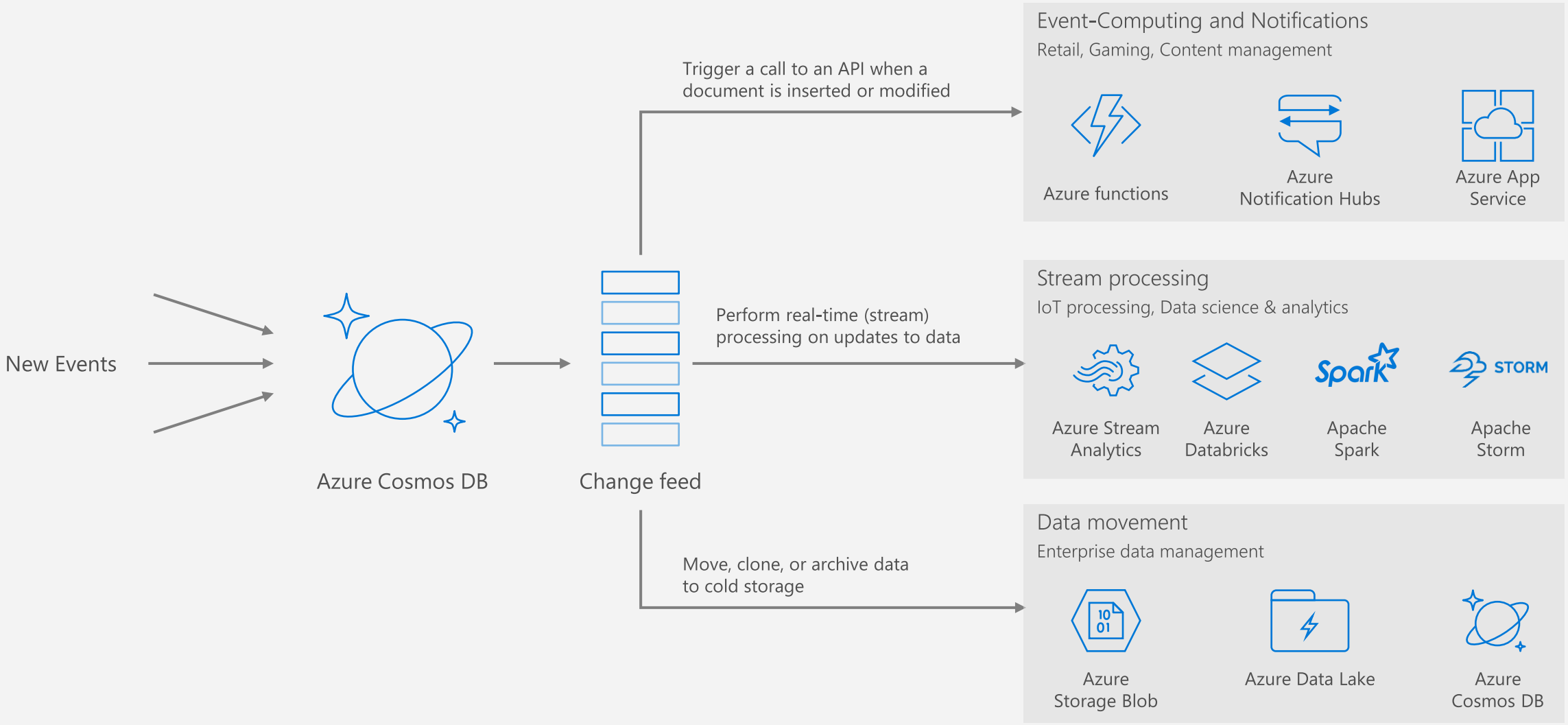


CHANGE FEED

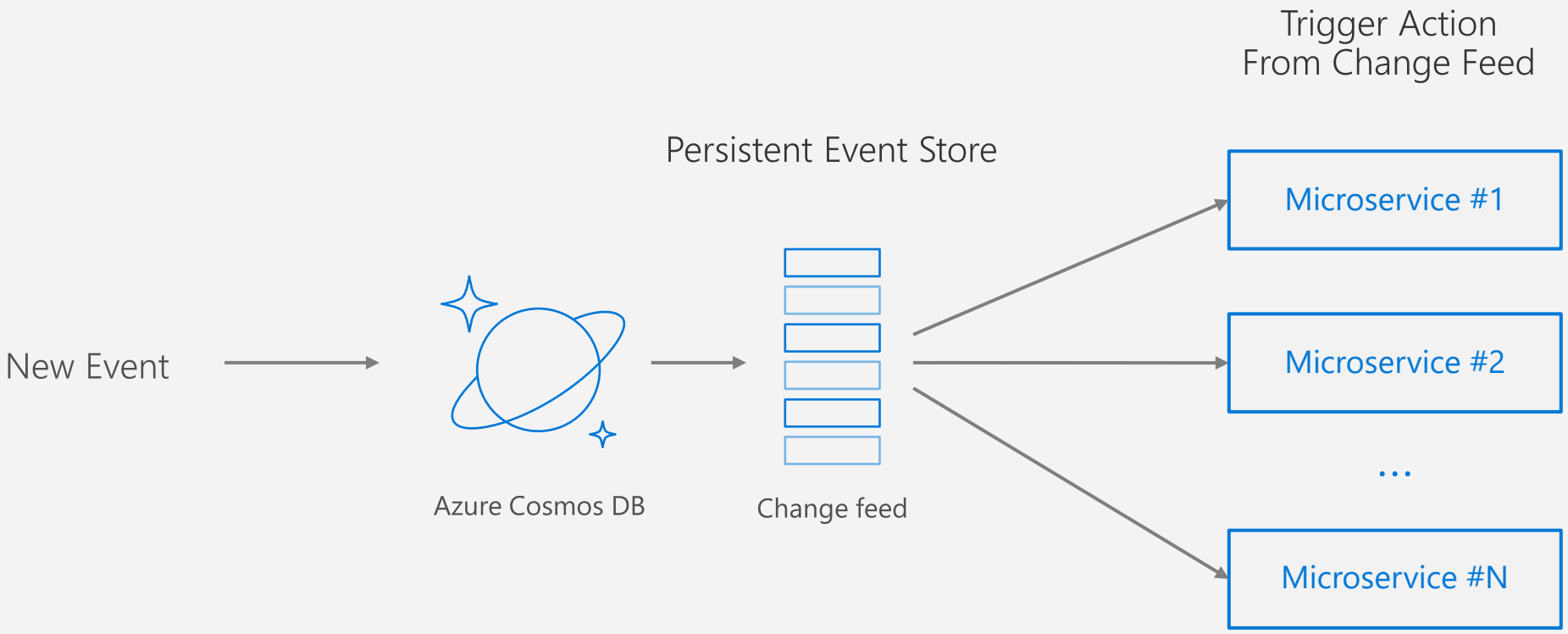
Persistent log of records within an Azure Cosmos DB container. Preseneted in the order in which they were modified



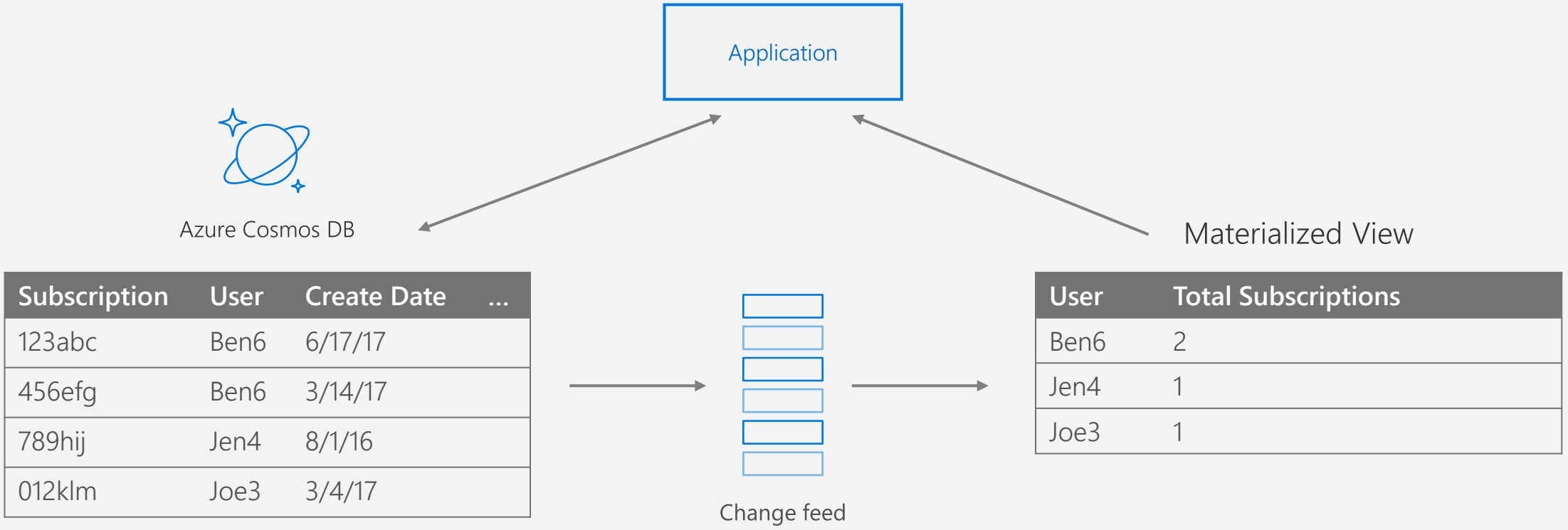
CHANGE FEED SCENARIOS



EVENT SOURCING FOR MICROSERVICES



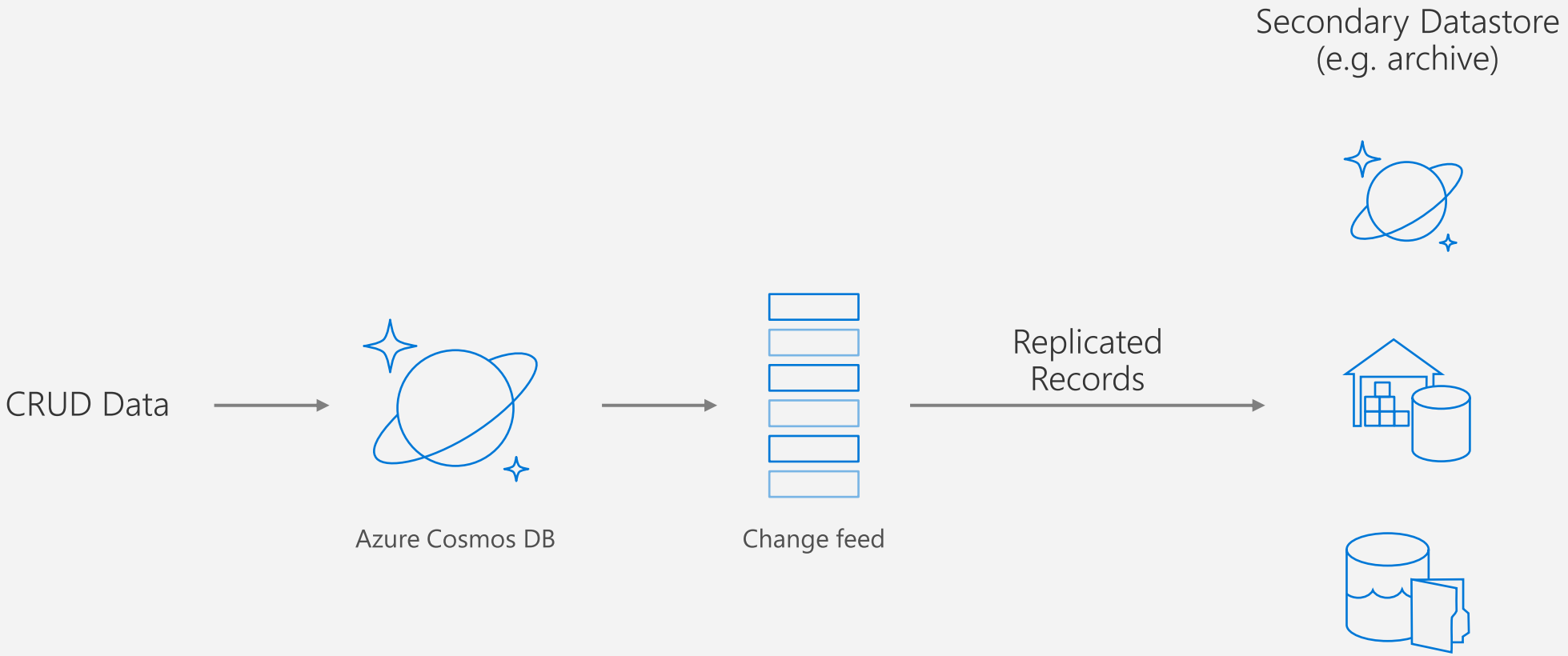
MATERIALIZING VIEWS



Subscription	User	Create Date	...
123abc	Ben6	6/17/17	
456efg	Ben6	3/14/17	
789hij	Jen4	8/1/16	
012klm	Joe3	3/4/17	

User	Total Subscriptions
Ben6	2
Jen4	1
Joe3	1

REPLICATING DATA

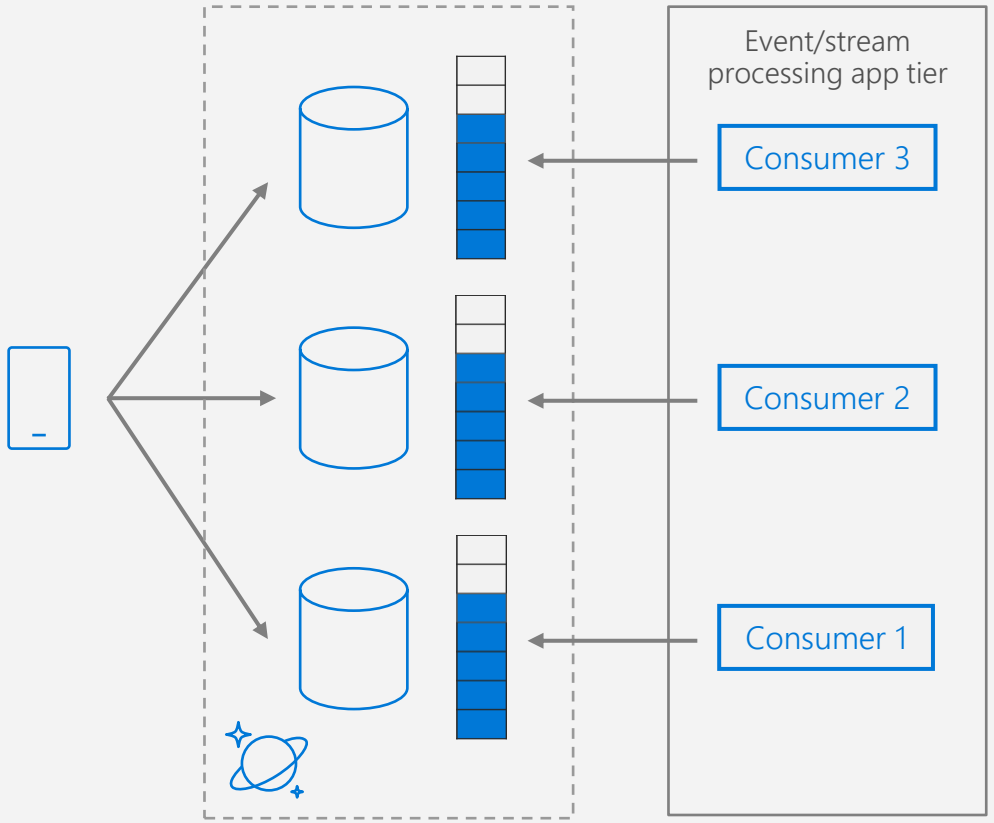


CHANGE FEED WITH PARTITIONS

Consumer parallelization

Change feed listens for any changes in Azure Cosmos DB collection. It then outputs the sorted list of documents that were changed in the order in which they were modified.

The changes are persisted, can be processed asynchronously and incrementally, and the output can be distributed across one or more consumers for parallel processing. The change feed is available for each partition key range within the document collection, and thus **can be distributed across one or more consumers for parallel processing.**



CHANGE FEED – RETRIEVING KEY RANGES

```
string pkRangesResponseContinutation = null;
List<PartitionKeyRange> partitionKeyRanges = new List<PartitionKeyRange>();
do
{
        FeedResponse<PartitionKeyRange> pkPagesResponse = await client.ReadPartitionKeyRangeFeedAsync(
                collectionUri,
                new FeedOptions { RequestContinutation = pkRangesResponseContinutation }
        );
        partitionKeyRanges.AddRange(pkRangesResponse);
        pkRangesResponseContinutation = pkRangesResponse.ResponseContinutation;
}
while (pkRangesResponseContinutation != null);
```

.NET

CHANGE FEED - CONSUMING CHANGE FEED

```
foreach (PartitionKeyRange pkRange in partitionKeyRanges)
{
        string continuation = null;
        checkpoints.TryGetValue(pkRange.Id, out continutation);
        IDocumentQuery<Document> query = client.CreateDocumentChangeFeedQuery(
                collection,
                new ChangeFeedOptions
                {
                        PartitionKeyRangeId = pkRange.Id,
                        StartFromBeginning = true,
                        RequestContinutation = continuation,
                        MaxItemCount = 1
                }
        );
...
```

CONTINUED ON NEXT PAGE

.NET

CHANGE FEED PRIMITIVES - CONSUME CHANGE FEED

```
...
        while (query.HasMoreResults)
        {
                FeedResponse<DeviceReading> readChangesResponse = query.ExecuteNextAsync<DeviceReading>().Result;
                foreach (DeviceReading changedDocument in readChangesResponse)
                {
                        Console.WriteLine(changedDocument.Id);
                }
                checkpoints[pkRange.Id] = readChangesResponse.ResponseContinuation;
        }
}
```

.NET

CHANGE FEED PROCESSOR LIBRARY

https://www.nuget.org/packages/Microsoft.Azure.DocumentDB.ChangeFeedProcessor/

Microsoft.Azure.DocumentDB.ChangeFeedProcessor 1.3.1

Microsoft Azure Cosmos DB Change Feed Processor library

This library provides a host for distributing change feed events in partitioned collection across multiple observers. Instances of the host can scale up (by adding) or down (by removing) dynamically, and the load will be automatically distributed among active instances in about-equal way.

Package Manager | .NET CLI | Paket CLI

```
PM> Install-Package Microsoft.Azure.DocumentDB.ChangeFeedProcessor -Version 1.3.1
```

Dependencies

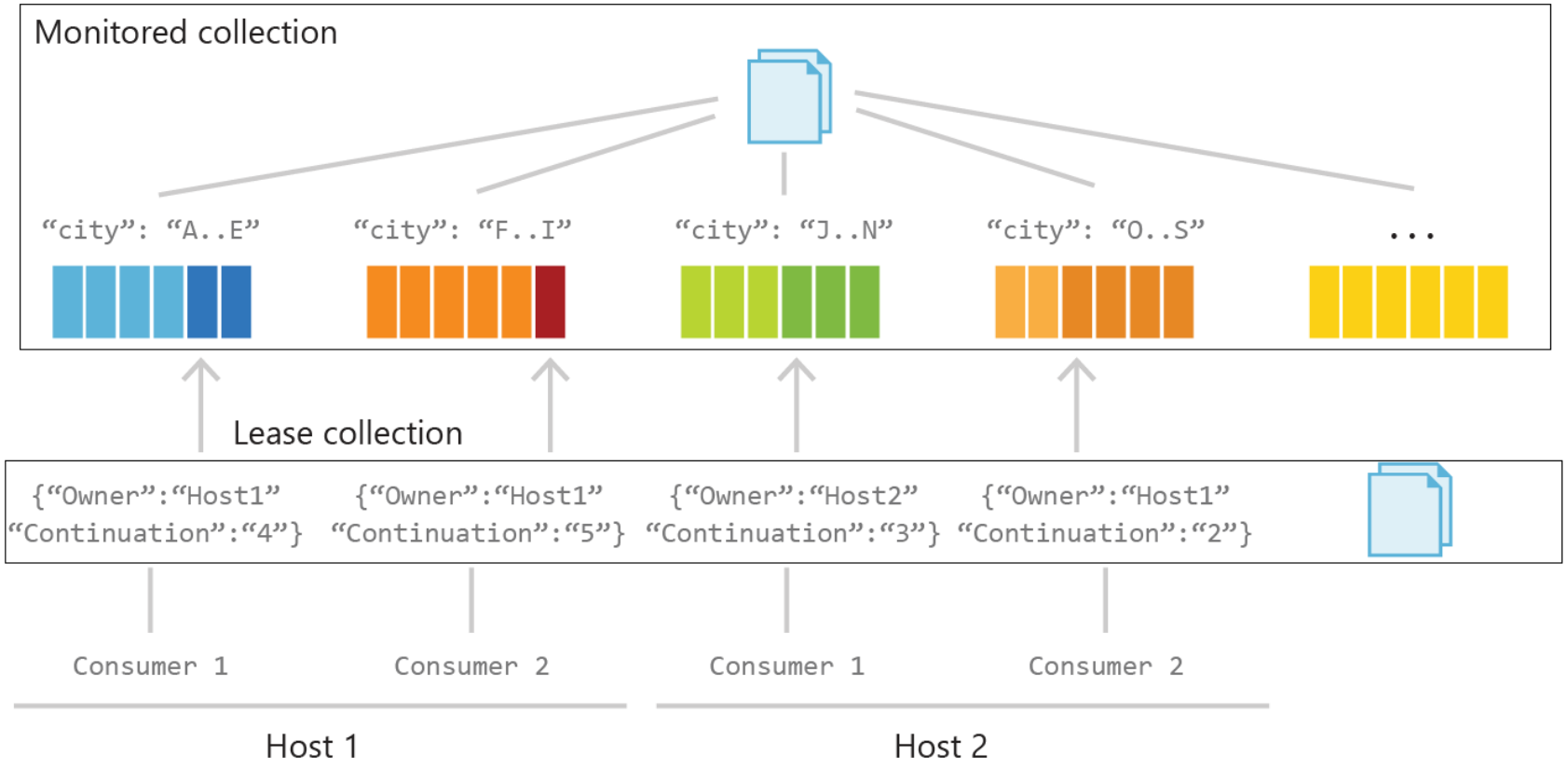
.NETFramework 4.5.2

Microsoft.Azure.DocumentDB (>= 1.20.2)
Newtonsoft.Json (>= 9.0.1)

.NETStandard 2.0

Microsoft.Azure.DocumentDB.Core (>= 1.8.2)
Newtonsoft.Json (>= 9.0.1)
System.Collections.Concurrent (>= 4.3.0)

CHANGE FEED PROCESSOR – BEHIND THE SCENES



CHANGE FEED PROCESSOR – INTERFACE IMPLEMENTATION

```
public class DocumentFeedObserver : IChangeFeedObserver
{
    ...
    public Task IChangeFeedObserver.ProcessChangesAsync(ChangeFeedObserverContext context, IReadOnlyList<Document>
    docs)
    {
          Console.WriteLine("Change feed: {0} documents", Interlocked.Add(ref totalDocs, docs.Count));
          foreach(Document doc in docs)
          {
                   Console.WriteLine(doc.Id.ToString());
          }
          return Task.CompletedTask;
    }
}
```

.NET

CHANGE FEED PROCESSOR - REGISTRATION

```
DocumentFeedObserver docObserver = new DocumentFeedObserver();
ChangeFeedEventHost host = new ChangeFeedEventHost(
          hostName,
          documentCollectionLocation,
          leaseCollectionLocation,
          feedOptions,
          feedHostOptions
);
await host.RegisterObserverAsync(docObserverFactory);
```

.NET

DEMO

Change Feed Usage

HANDS-ON EXERCISE (3 of 4)

AUTHORING STORED PROCEDURES USING JAVASCRIPT

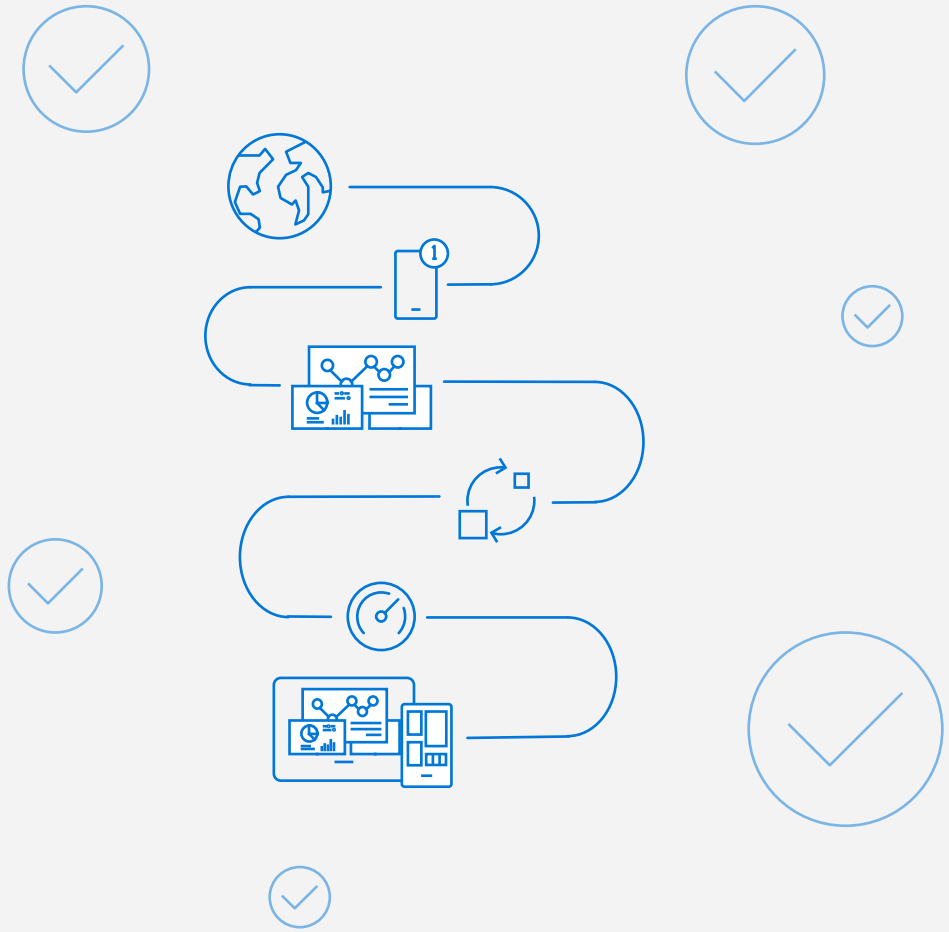
Tasks:

1. Create Simple Stored Procedure
2. Manipulate Multiple Documents in Stored Procedure
3. Implement Custom Continuation Model

TROUBLESHOOTING

Using HTTP/REST response status codes and headers to diagnose and troubleshoot requests.

This module will reference troubleshooting in the context of all Azure Cosmos DB modules and APIs.



ANALYZING HTTP RESPONSES

RESPONSE STATUS CODES

When a request is unsuccessful, Azure Cosmos DB responds using well-defined HTTP status codes that can provide more detail into exactly why a specific request failed.

RESPONSE HEADERS

Azure Cosmos DB uses a variety of HTTP headers to offer insight into the result of requests, error conditions, and useful metadata to perform actions such as:

- Resume request
- Measure RU/s charge associated with request
- Access newly created resource directly.

HTTP response codes	
2xx	Success
4xx	Client Errors
5xx	Server Errors

RESPONSE STATUS CODES (1 of 2)

Code	Meaning
200 OK	GET, PUT or POST operation was successful
201 Created	Resource created successfully using a POST operation
204 No Content	Resource deleted successfully using a DELETE operation
401 Unauthorized	Invalid Authorization header
403 Forbidden	Authorization token expired Resource quota reached when attempting to create a document Stored Procedure, Trigger or UDF is blacklisted from executed
409 Request Timeout	Stored Procedure, Trigger or UDF exceeded maximum execution time

RESPONSE STATUS CODES (2 of 2)

Header	Value
409 Conflict	The item Id for a PUT or POST operation conflicts with an existing item
412 Precondition Failure	The specified eTag is different from the version on the server (optimistic concurrency error)
413 Entity Too Large	The item size exceeds maximum allowable document size of 2MB
429 Too Many Requests	Container has exceeded provisioned throughput limit
449 Retry With	Transient error has occurred, safe to retry
50x	Server-side error. If effort persists, contact support

RESPONSE HEADERS (1 of 2)

Header	Value
x-ms-activity-id	Unique identifier for the operation
x-ms-serviceversion	Service Version used for request/response
x-ms-schemaversion	Schema Version used for request/response
x-ms-item-count	In a query (or read-feed), the number of items returned
x-ms-alt-content-path	REST URI to access resource using user-supplied IDs
etag	The same value as the _etag property of the requested item

RESPONSE HEADERS (2 of 2)

Header	Value
`x-ms-continuation`	Token returned if a query (or read-feed) has more results and is resubmitted by clients as a request header to resume execution
`x-ms-session-token`	Used to maintain session consistency. Clients much echo this as a request header in subsequent operations to the same container
`x-ms-request-charge`	Number of normalized RU/s for the operation
`x-ms-resource-quota`	Allotted quota for the specified resource in the account
`x-ms-resource-usage`	Current usage count of the specified resource in the account
`x-ms-retry-after-ms`	If rate limited, the number of milliseconds to wait before retrying the operation

DEMO

Viewing REST API Response Metadata

IDENTIFYING RATE LIMITING

HTTP RESPONSE STATUS CODE

A rate limited request will return a HTTP status code of **429 (Too Many Requests)**. This response indicates that the container has exceeded provisioned throughput limit.

HTTP RESPONSE HEADER

A **rate limited** request will also have a **x-ms-retry-after-ms** header. This header gives the number of milliseconds your application should wait before retrying the current request.

AUTOMATIC RETRY ON THROTTLE

The SDK automatically retries any throttled requests. This can **potentially create a long-running client-side method** that is attempting to retry throttled requests.

LOGGING

WHAT IS LOGGED BY AZURE DIAGNOSTIC LOGS

All authenticated backend requests across all protocols and APIs

- Includes failed requests

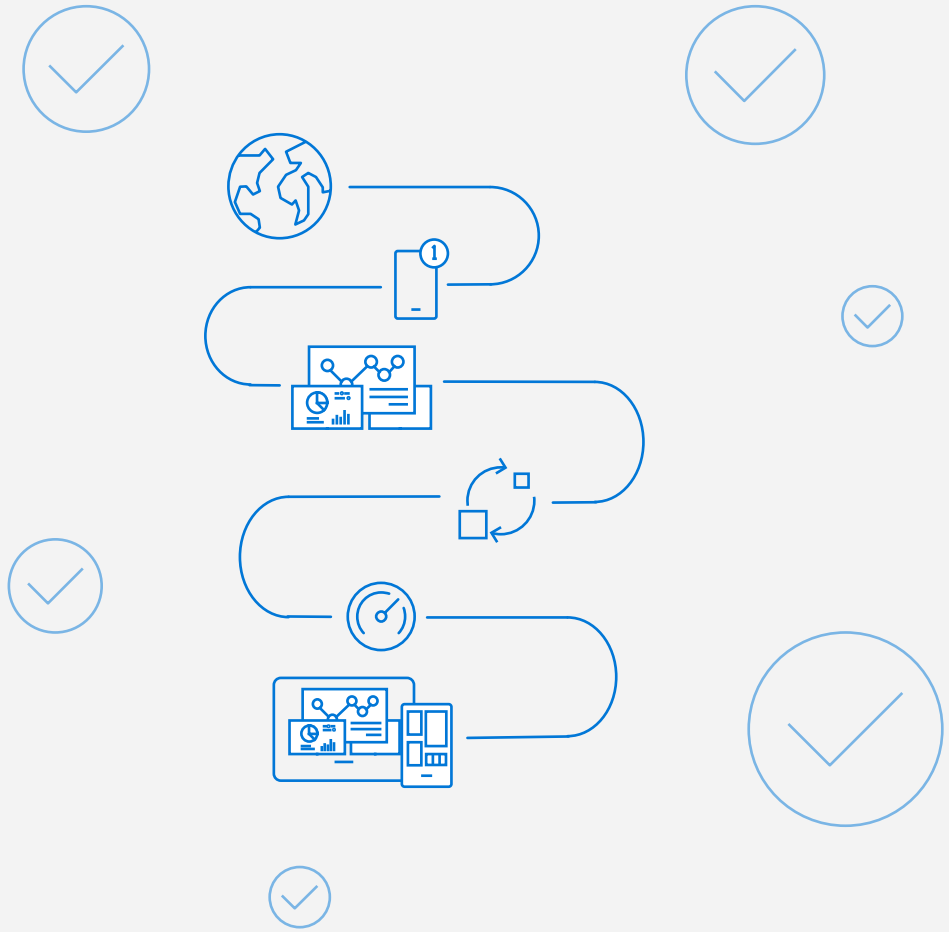
Database operations

- Includes CRUD operations on all resources

Account Key operations

Unauthenticated requests

- Requests that result in a 401 response



VIEWING LOGS IN LOG ANALYTICS

ENABLE LOGGING

Diagnostic Logs for Azure Services are opt-in. You should first enable logging (using the Portal, CLI or PowerShell).

Logs take, on average, about two hours to be made available.

LOG ANALYTICS

If you selected the **Send to Log Analytics** option when you turned on diagnostic logging, diagnostic data from your collection is forwarded to Log Analytics.

From within the Log Analytics portal experience, you can:

- Search logs using the expanded Log Analytics query language
- Visualize the results of a query as a graph
- Pin a graph visualization of a query to your Azure portal dashboard

DEMO

Viewing Logs in Log Analytics

HANDS-ON EXERCISE (4 of 4)

TROUBLESHOOTING FAILED AND RATE LIMITED REQUESTS

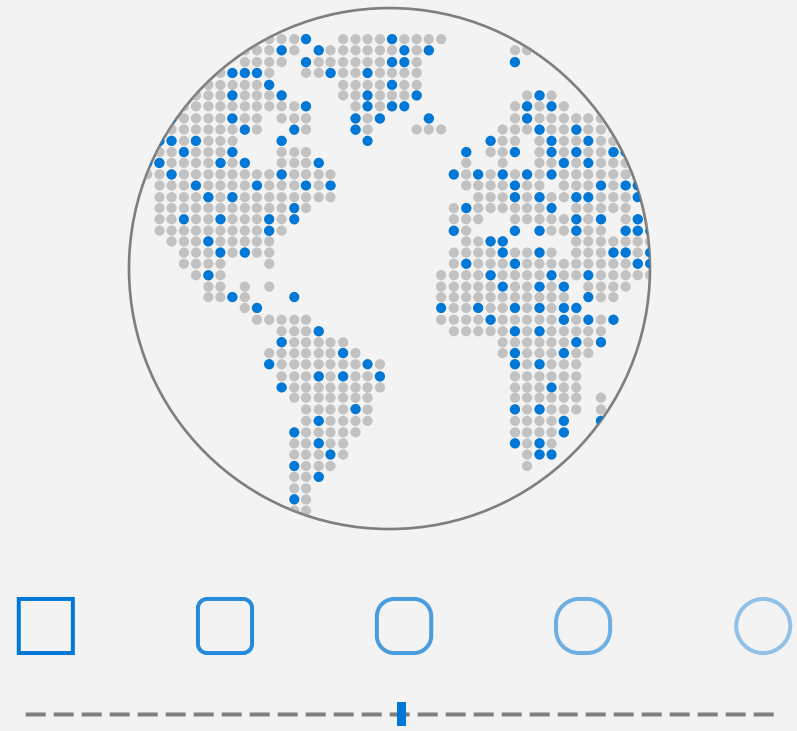
Tasks:

1. Simulate a Usage Spike in Requests
2. View Response Metadata
3. Third exercise task

MODELING & PLANNING

Modeling data & configuring containers to take advantage of Azure Cosmos DB strengths.

This module will reference modeling in the context of all Azure Cosmos DB modules and APIs.



CONTAINERS

CONTAINERS

- Containers do NOT enforce schema
- There are benefits to co-locate multiple types in a container
- Annotate records with a "type" property

CO-LOCATING TYPES IN THE SAME CONTAINER

- Ability to query across multiple entity types with a single network request.
- Ability to perform transactions across multiple types
- Cost: reduce physical partition footprint

CO-LOCATING TYPES

Ability to query across multiple entity types with a single network request.

For example, we have two types of documents: cat and person.

```
{
   "id": "Andrew",
   "type": "Person",
   "familyId": "Liu",
   "worksOn": "Azure Cosmos DB"
}
```

```
{
   "id": "Ralph",
   "type": "Cat",
   "familyId": "Liu",
   "fur": {
         "length": "short",
         "color": "brown"
   }
}
```

We can query both types of documents without needing a JOIN simply by running a query without a filter on type:

```
SELECT * FROM c WHERE c.familyId = "Liu"
```

If we wanted to filter on type = "Person", we can simply add a filter on type to our query:

```
SELECT * FROM c WHERE c.familyId = "Liu" AND c.type = "Person"
```

UPDATING NORMALIZED DATA (1 of 3)

UPDATES ARE ATOMIC

Update operations update the entire document, not specific fields or “parts” of the document.

DE-NORMALIZED DOCUMENTS CAN BE EXPENSIVE TO UPDATE

De-normalization has benefits for read operations, but you must weigh this against the costs in write operations.

De-normalization may require fanning out update operations.

Normalization may require chaining a series of requests to resolve relationships.

```
{
        "id": "08259",
        "ticketPrice": 255.00,
        "flightCode": "3754",
        "origin": {
                "airport": "SEA",
                "gate": "A13",
                "departure": "2014-09-15T23:14:25.7251173Z"
        },
        "destination": {
                "airport": "JFK",
                "gate": "D4",
                "arrival": "2014-09-16T02:10:10.2379581Z"
        },
        "pilot": [{
                "id": "EBAMAO",
                “name": "Hailey Nelson"
        }]
}
```

UPDATING NORMALIZED DATA (2 of 3)

Normalized: Optimized for writes over reads

```
{
        "id": "08259",
        "pilot": [{ "id": "EBAMAO", “name": "Hailey Nelson“ }]
},
{
        "id": "08259",
        "ticketPrice": 255.00,
        "flightCode": "3754"
},
{
        "id": "08259",
        "origin": {
                "airport": "SEA", "gate": "A13",
                "departure": "2014-09-15T23:14:25.7251173Z"
        },
        "destination": {
                "airport": "JFK", "gate": "D4",
                "arrival": "2014-09-16T02:10:10.2379581Z"
        }
}
```

De-normalized: Optimized for reads over writes

```
{
        "id": "08259",
        "ticketPrice": 255.00,
        "flightCode": "3754",
        "origin": {
                "airport": "SEA",
                "gate": "A13",
                "departure": "2014-09-15T23:14:25.7251173Z"
        },
        "destination": {
                "airport": "JFK",
                "gate": "D4",
                "arrival": "2014-09-16T02:10:10.2379581Z"
        },
        "pilot": [{
                "id": "EBAMAO",
                “name": "Hailey Nelson"
        }]
}
```

UPDATING NORMALIZED DATA (3 of 3)

THE SOLUTION IS TYPICALLY A COMPROMISE BASED ON YOUR WORKLOAD

Examine your workload. Answer the following questions:

- Which fields are commonly updated together?
- What are the most common fields included in all queries?

Example: The ticketPrice, origin and destination fields are often updated together. The pilot field is only rarely updated. The flightCode field is included in almost all queries across the board.

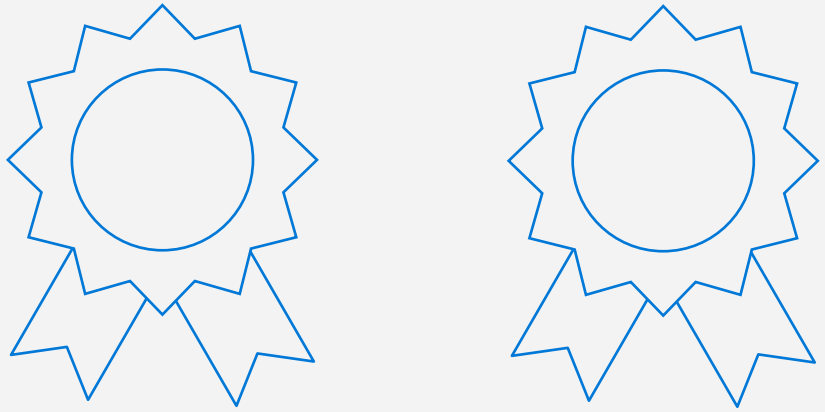
```
{
        "id": "08259",
        "flightCode": "3754",
        "pilot": [{ "id": "EBAMAO", “name": "Hailey Nelson“ }]

},
{
        "id": "08259",
        "flightCode": "3754",
        "ticketPrice": 255.00,
        "origin": {
                "airport": "SEA", "gate": "A13",
                "departure": "2014-09-15T23:14:25.7251173Z"
        },
        "destination": {
                "airport": "JFK", "gate": "D4",
                "arrival": "2014-09-16T02:10:10.2379581Z"
        }
}
```

SHORT-LIFETIME DATA

Some data produced by applications are only useful for a finite period of time:

- Machine-generated event data
- Application log data
- User session information

It is important that the database system systematically purges this data at pre-configured intervals.



TIME-TO-LIVE (TTL)

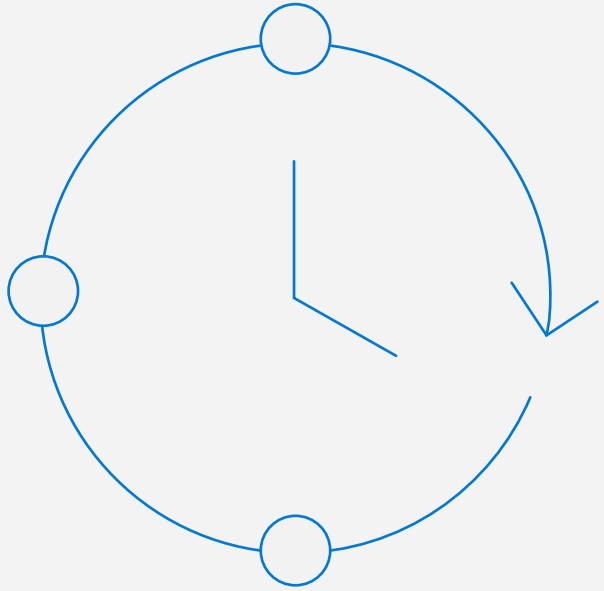
AUTOMATICALLY PURGE DATA

Azure Cosmos DB allows you to set the length of time in which documents live in the database before being automatically purged. A document's "time-to-live" (TTL) is measured in seconds from the last modification and can be set at the collection level with override on a per-document basis.

The TTL value is specified in the _ts field which exists on every document.

- The _ts field is a unix-style epoch timestamp representing the date and time. The _ts field is updated every time a document is modified.

Once TTL is set, Azure Cosmos DB will automatically remove documents that exist after that period of time.



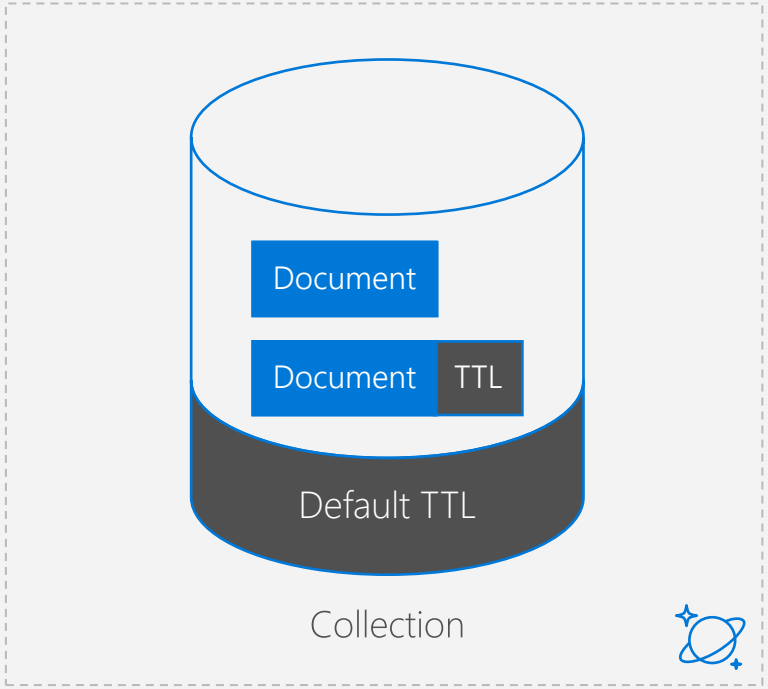
EXPIRING RECORDS USING TIME-TO-LIVE

TTL BEHAVIOR

The TTL feature is controlled by TTL properties at two levels - the collection level and the document level.

- DefaultTTL for the collection
  - If missing (or set to null), documents are not deleted automatically.
  - If present and the value is "-1" = infinite – documents don't expire by default
  - If present and the value is some number ("n") – documents expire "n" seconds after last modification
- TTL for the documents:
  - Property is applicable only if DefaultTTL is present for the parent collection.
  - Overrides the DefaultTTL value for the parent collection.

The values are set in seconds and are treated as a delta from the _ts that the document was last modified at.



DEMO

Pruning old records using TTL Values

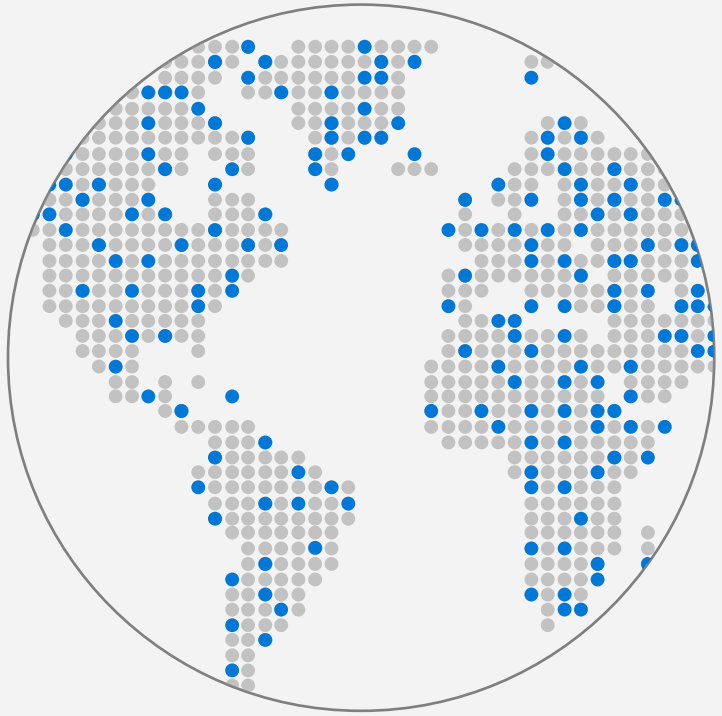
TURNKEY GLOBAL DISTRIBUTION (2 of 3)

High Availability

- Automatic and Manual Failover
- Multi-homing API removes need for app redeployment

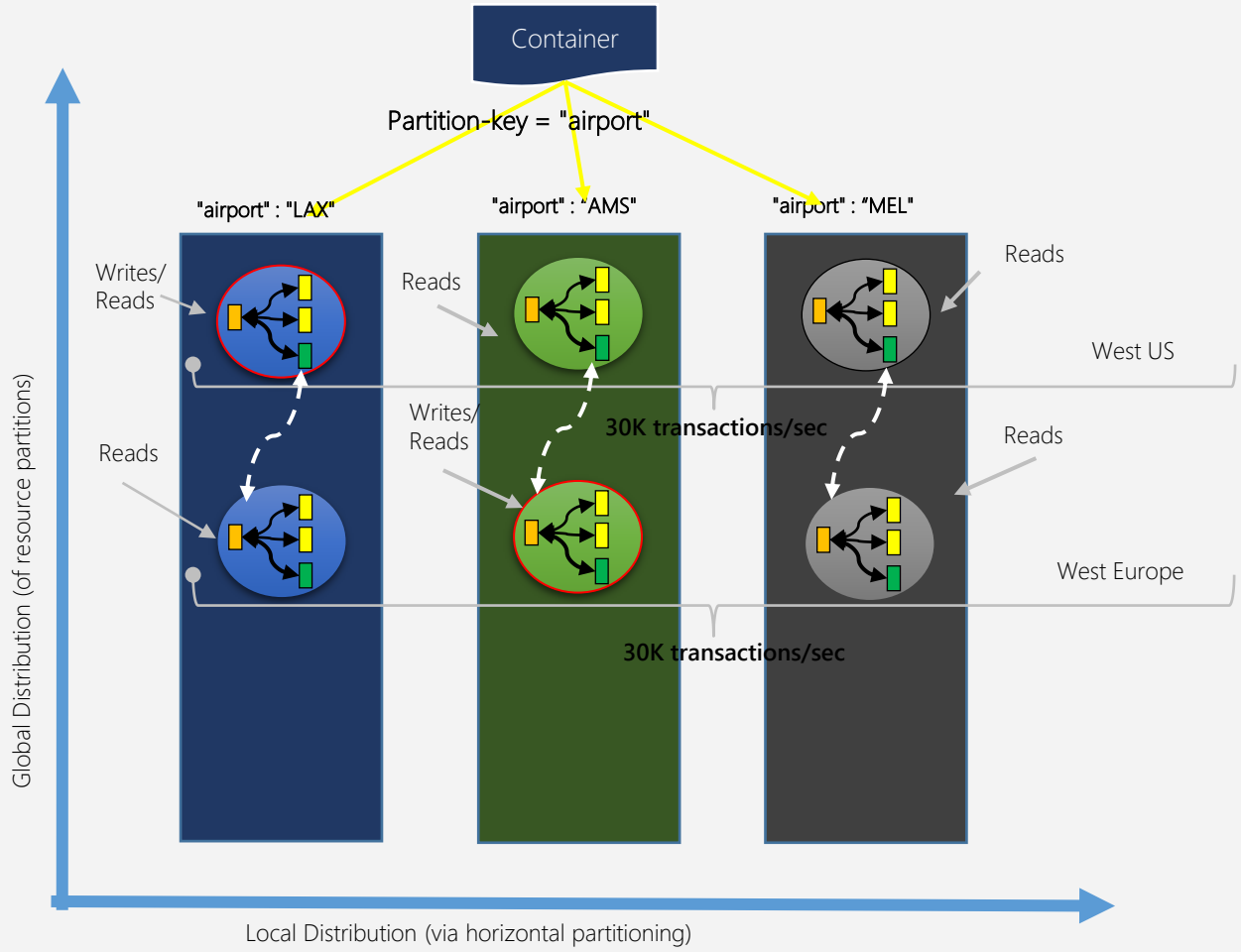
Low Latency (anywhere in the world)

- Packets cannot move fast than the speed of light
- Sending a packet across the world under ideal network conditions takes 100's of milliseconds
- You can cheat the speed of light – using data locality
    - CDN's solved this for static content
    - Azure Cosmos DB solves this for dynamic content

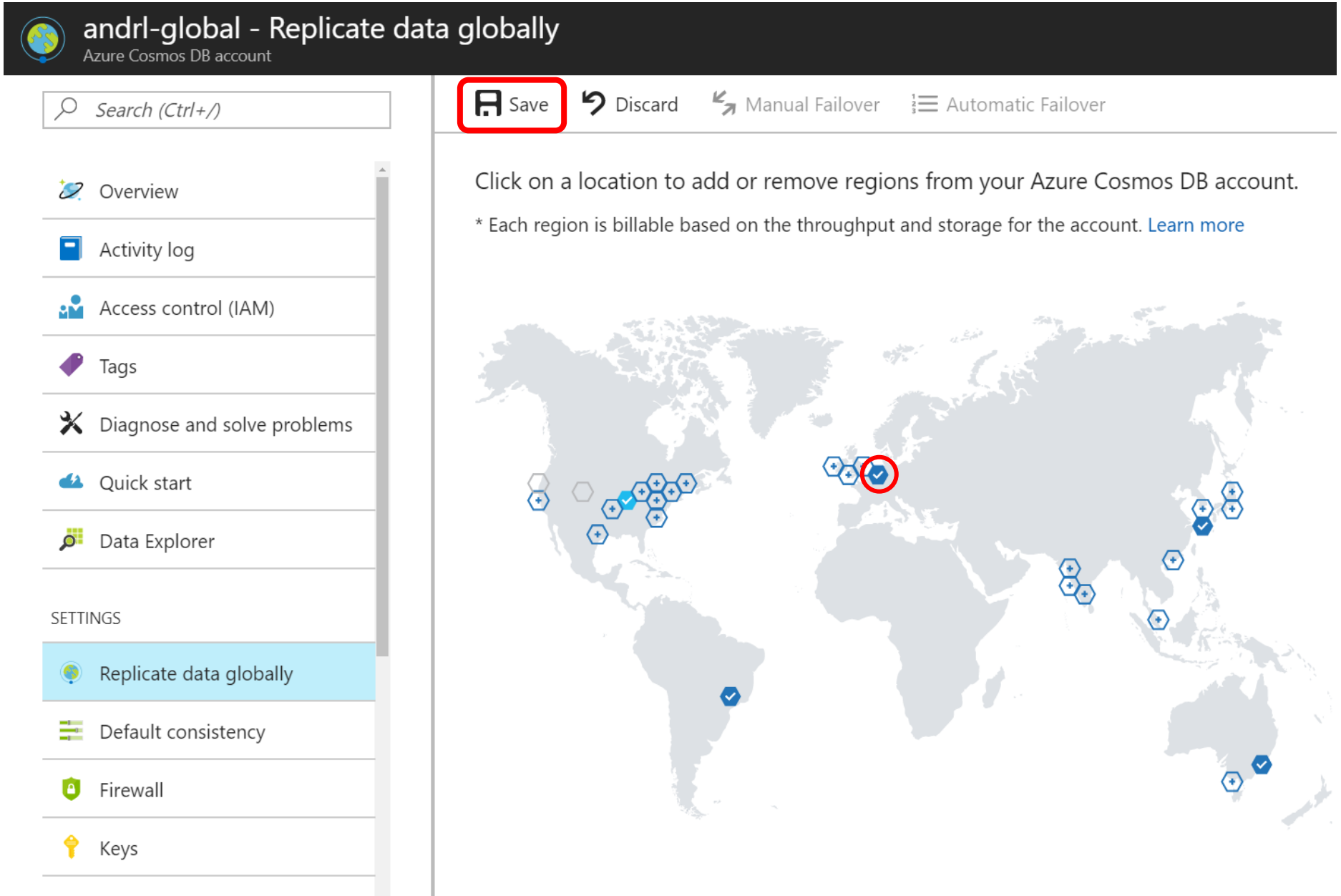


TURNKEY GLOBAL DISTRIBUTION (3 of 3)

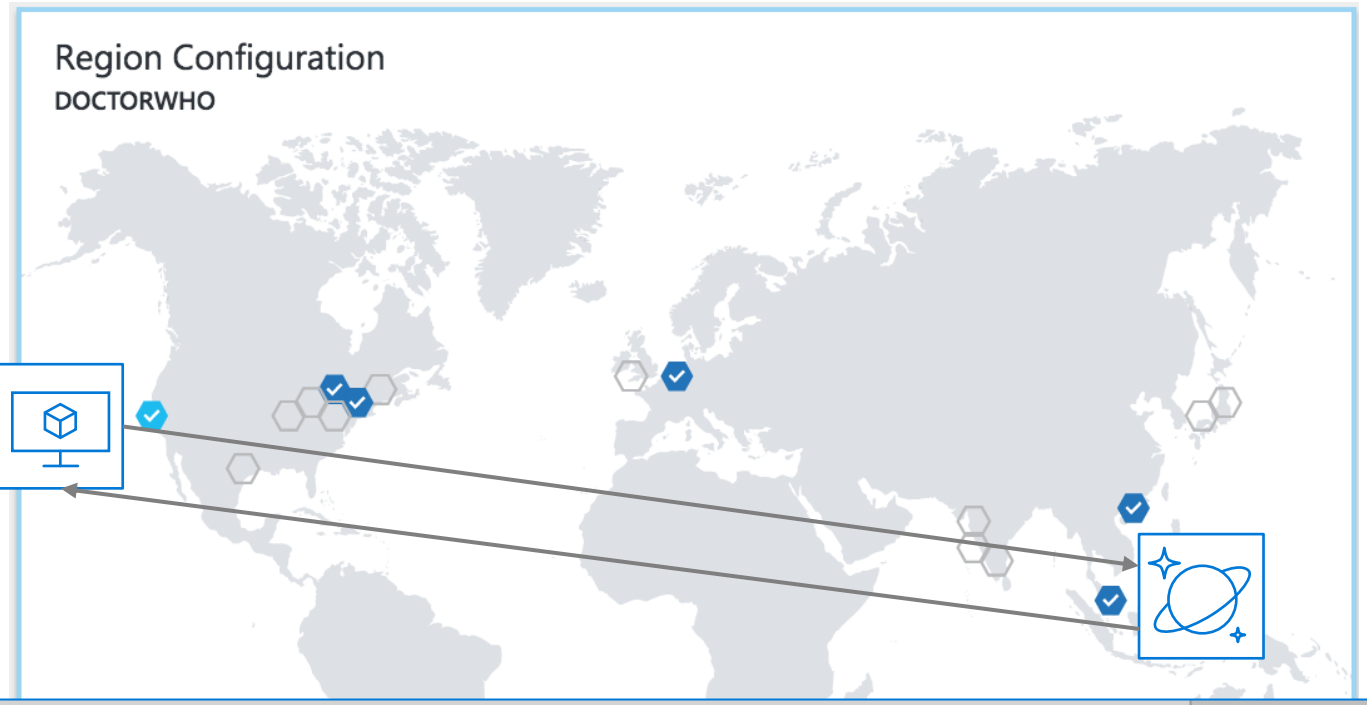
- Automatic and transparent replication worldwide
- Each partition hosts a replica set per region
- Customers can test end to end application availability by programmatically simulating failovers
- All regions are hidden behind a single global URI with multi-homing capabilities
- Customers can dynamically add / remove additional regions at any time



REPLICATING DATA GLOBALLY (1 of 3)



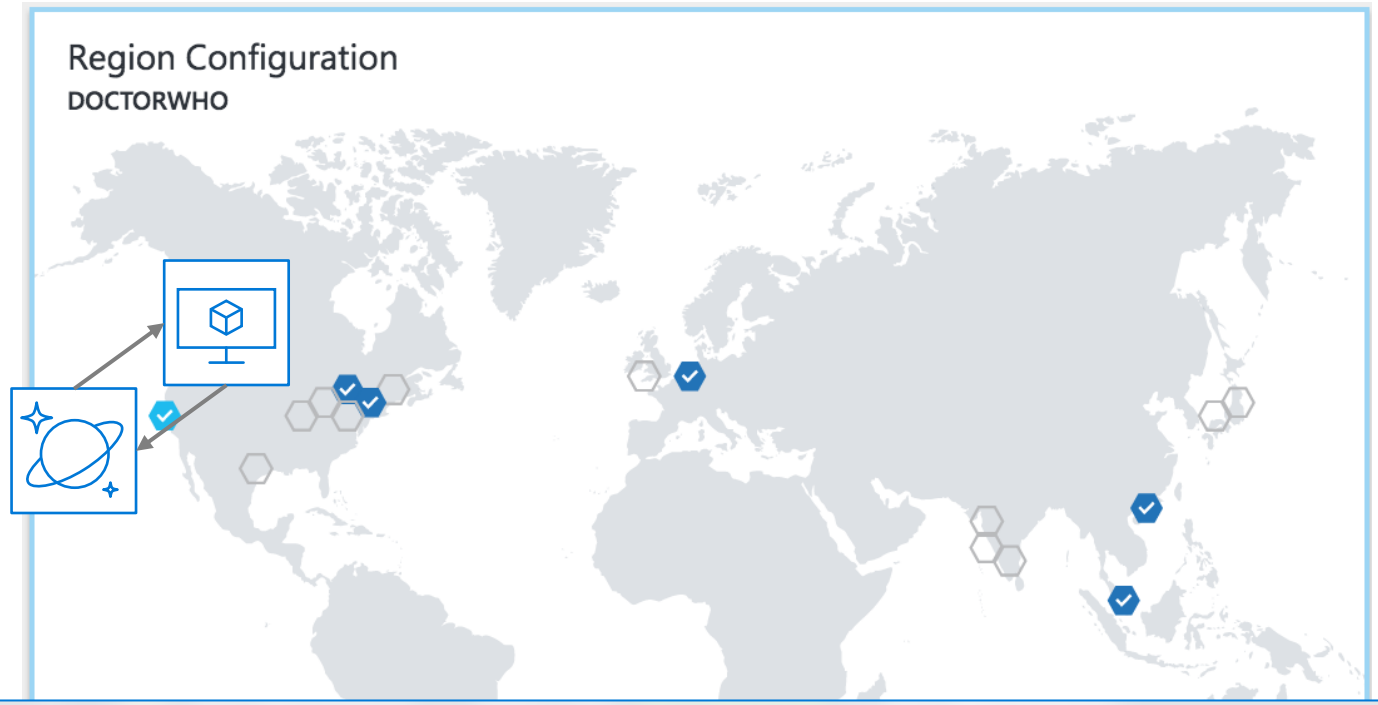
REPLICATING DATA GLOBALLY (2 of 3)



...rkspace/docdb/docdb — amypond@blink: ~/docdb — ssh amypond@104.42.108.173 -p 22

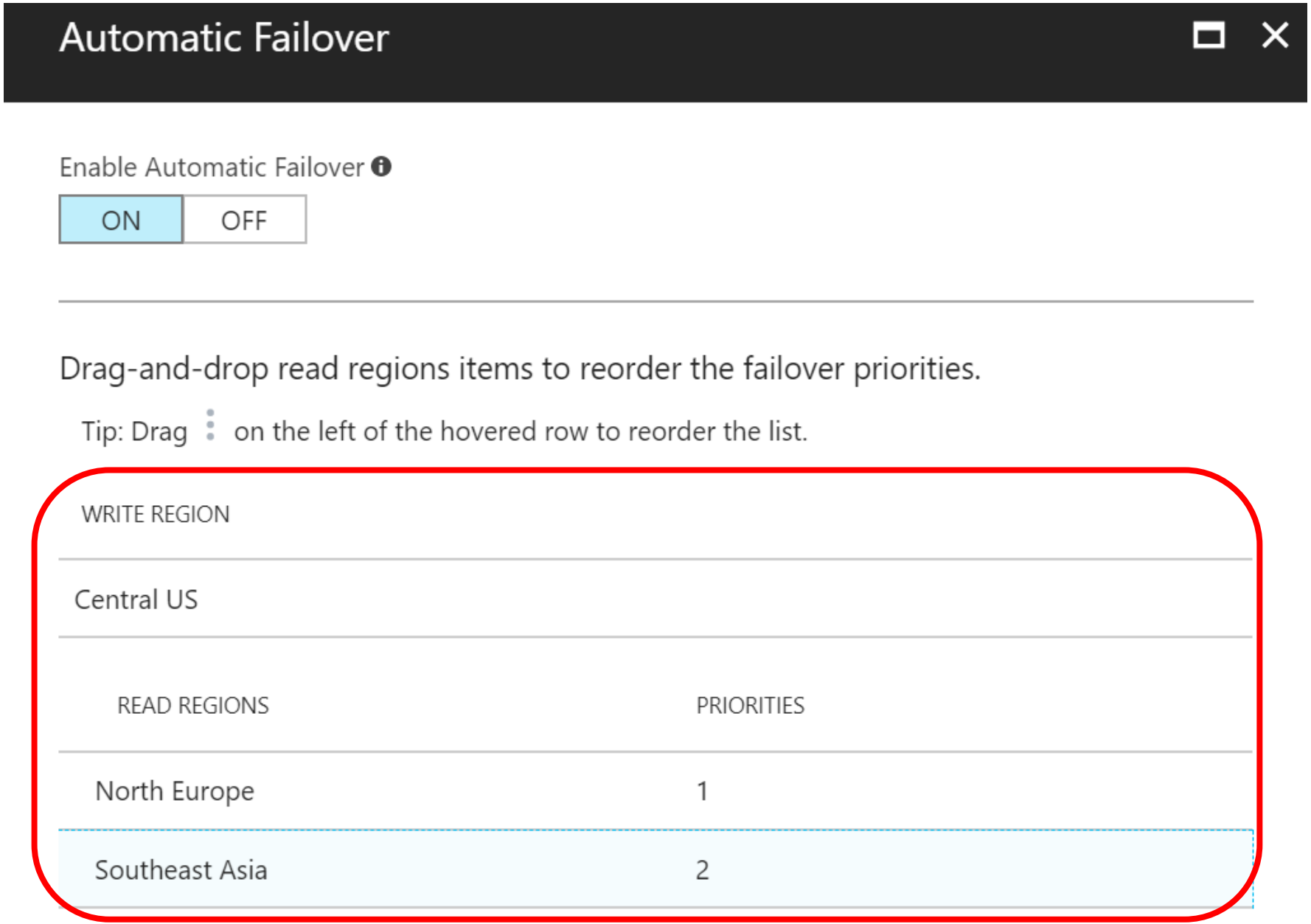
```
amypond@blink:~/docdb$ node testQ2.js
210.788804 milliseconds, 2 RUs, ActivityId: 9025eac6-eb74-4a07-94cf-f2383caffbb3
178.825773 milliseconds, 2 RUs, ActivityId: ea53b736-b629-4290-9cdf-cf80c6139461
178.839173 milliseconds, 2 RUs, ActivityId: f143c992-ba67-4c7b-b7b8-0b5db8df4dbf
178.564573 milliseconds, 2 RUs, ActivityId: 1a8d7b5b-42c5-4c39-9160-d1e5ab2200d0
179.229073 milliseconds, 2 RUs, ActivityId: 483b85de-74e0-4f48-9206-70ac1268c60e
178.653772 milliseconds, 2 RUs, ActivityId: 50fbfe91-f41e-4f14-8a15-0b344c894727
178.464572 milliseconds, 2 RUs, ActivityId: cac6446a-79d4-4886-81d8-1dda835daa72
180.708475 milliseconds, 2 RUs, ActivityId: d40af8e4-582b-4479-bb9c-e3582eac6774
```

REPLICATING DATA GLOBALLY (3 of 3)




```
amypond@blink:~/docdb$ node testQ2.js
12.736112 milliseconds, 2 RUs, ActivityId: dd3c17b9-1b76-445a-8e27-29b7486bd7e4
4.947605 milliseconds, 2 RUs, ActivityId: e2f4c899-9fb1-4f76-a4ab-e5718fac5742
5.044005 milliseconds, 2 RUs, ActivityId: 0fc5d216-78a0-4d92-a3d0-63efd9dd6552
5.351205 milliseconds, 2 RUs, ActivityId: 861155f0-81ba-4c8a-9933-e50d8708cc21
4.553505 milliseconds, 2 RUs, ActivityId: 3db9641f-70f1-4ef1-84bb-809280bbe1a5
5.427506 milliseconds, 2 RUs, ActivityId: 10d1b2e5-e795-4c77-8655-815f410ba11e
5.900106 milliseconds, 2 RUs, ActivityId: bda1df86-c5ad-45c5-bcfb-93d417c54751
4.895405 milliseconds, 2 RUs, ActivityId: f206d58d-64a2-47f2-9653-145eaf47ff97
5.244306 milliseconds, 2 RUs, ActivityId: 3aa7e177-b1a9-413d-8023-55f5054d1b74
```

AUTOMATIC FAILOVER (1 of 2)

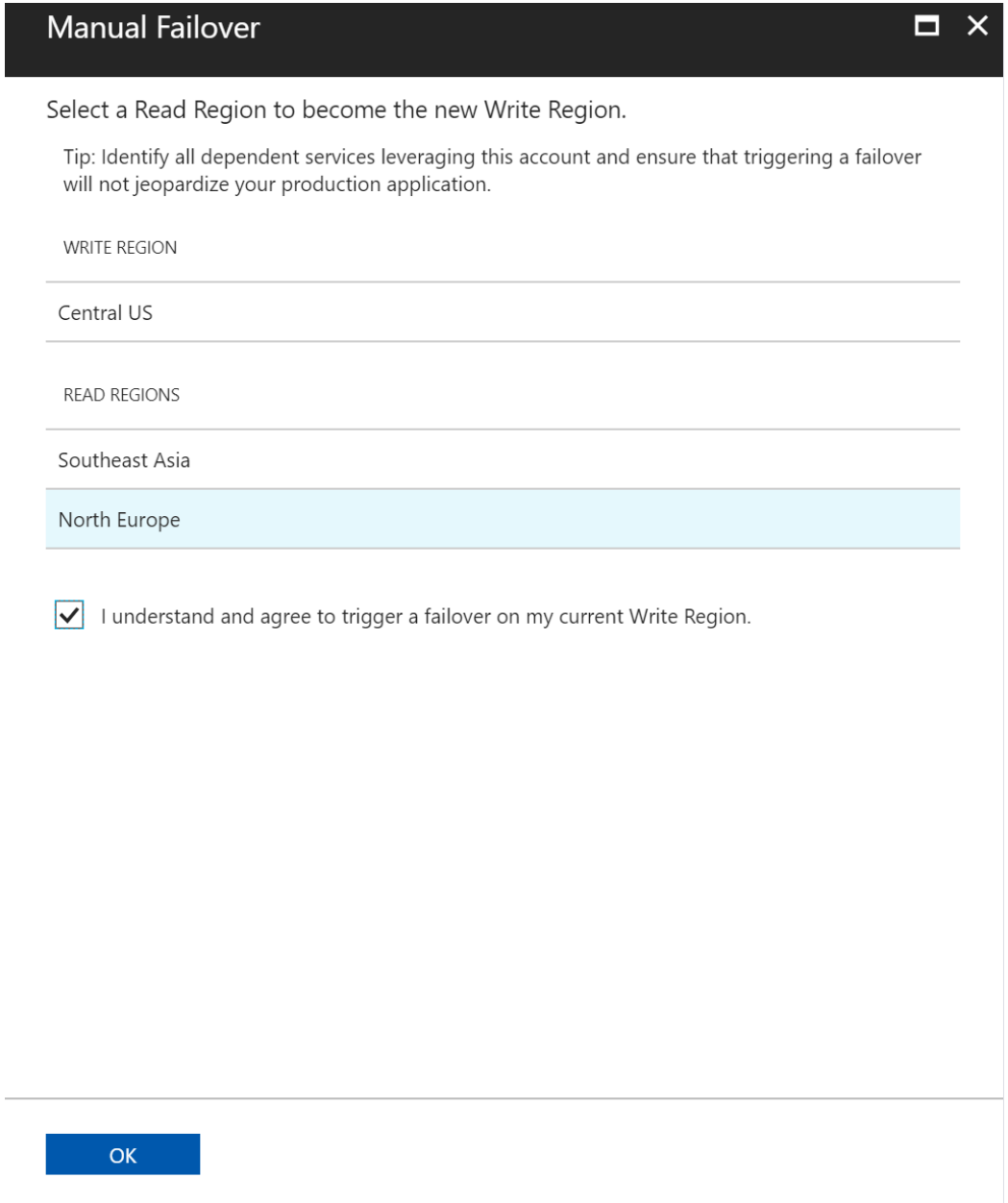


AUTOMATIC FAILOVER (2 of 2)

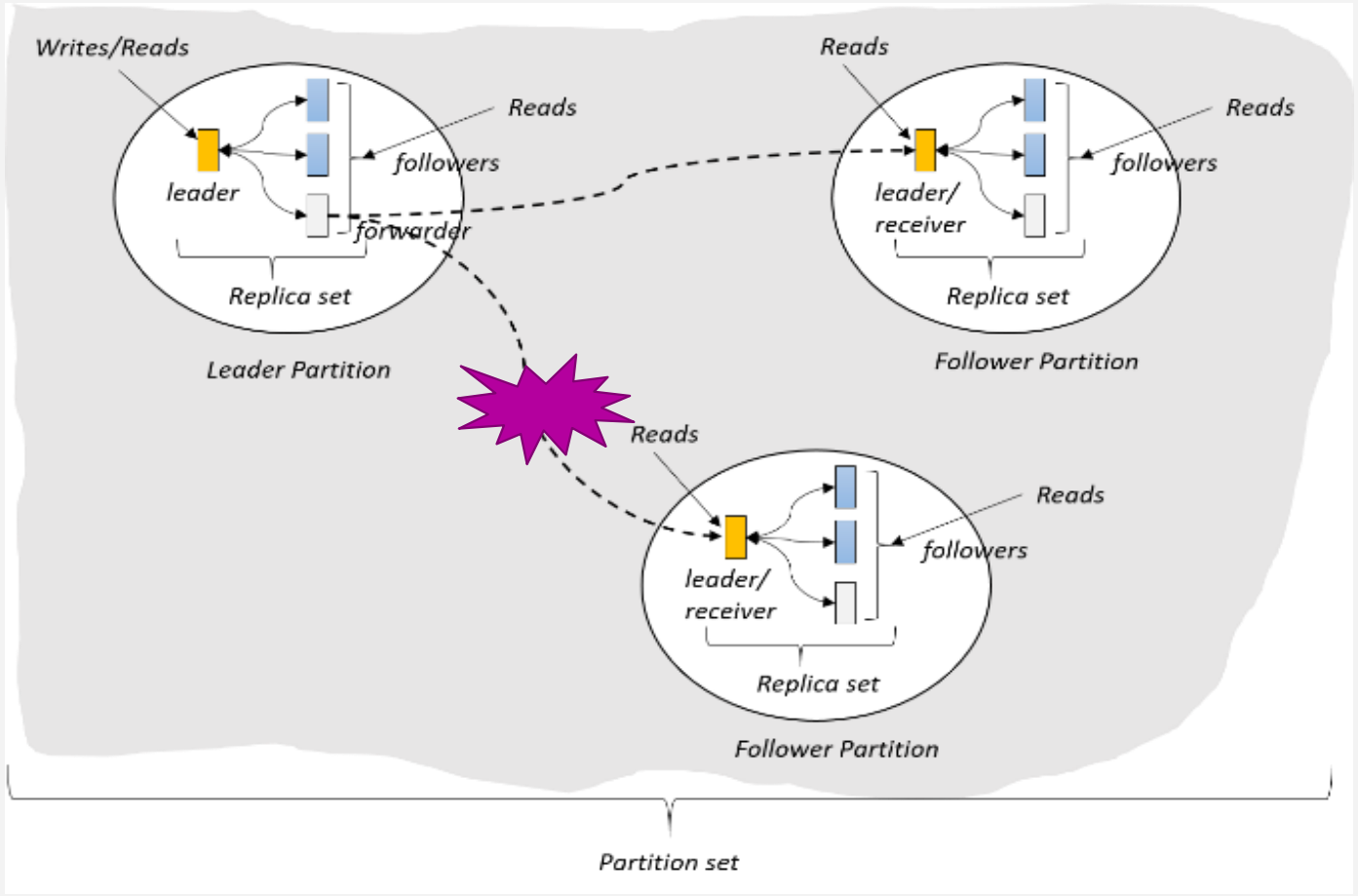
```
DocumentClient client;
ConnectionPolicy policy = new ConnectionPolicy()
{
    ConnectionMode = ConnectionMode.Direct,
    ConnectionProtocol = Protocol.Tcp
};
policy.PreferredLocations.Add(LocationNames.CentralUS); //first preference
policy.PreferredLocations.Add(LocationNames.SoutheastAsia); //second preference
policy.PreferredLocations.Add(LocationNames.NorthEurope); //third preference

client = new DocumentClient(new Uri(endpointUri), PrimaryKey, connectionPolicy: policy);
```

.NET

MANUAL FAILOVER



BREWER'S CAP THEOREM



Impossible for distributed data store to simultaneously provide more than 2 out of the following 3 guarantees:

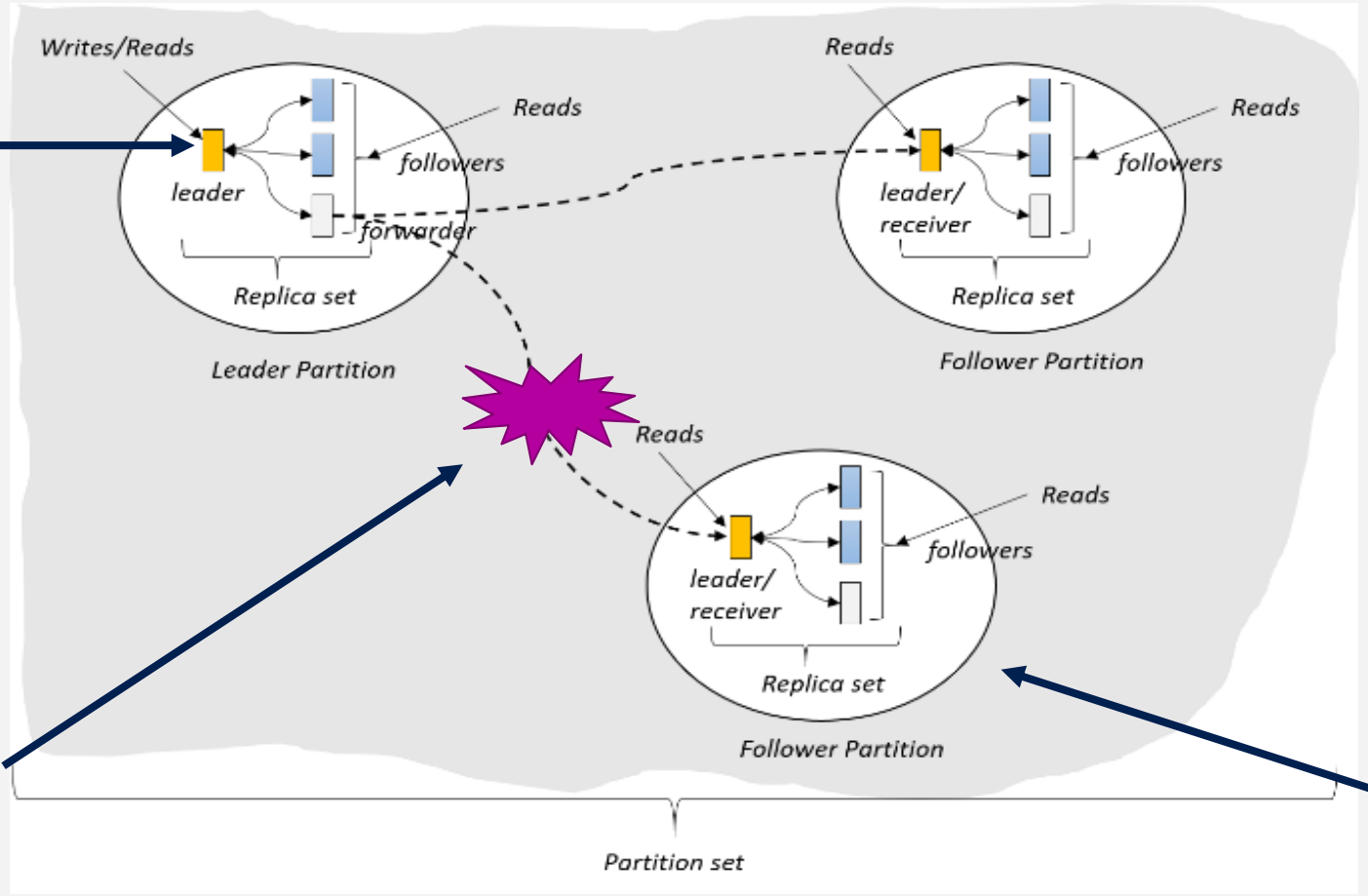
- Consistency
- Availability
- Partition Tolerance

CONSISTENCY (1 of 3)

(West US)

Value = ~~5~~ 6

Update 5 => 6



(East US)

Value = ~~5~~ 6

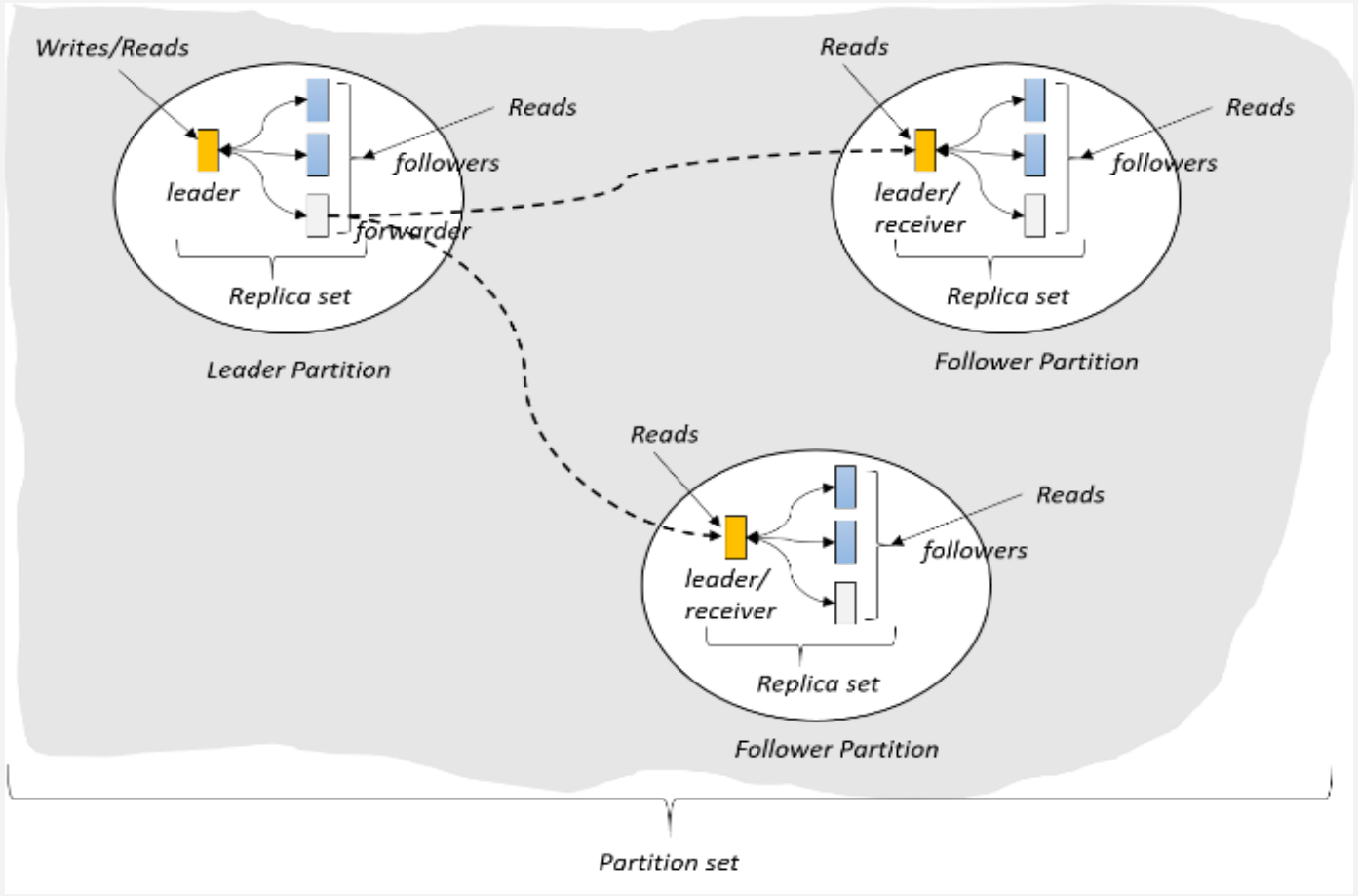
(North Europe)

Value = 5

What happens when a network partition is introduced?

Reader: What is the value? Should it see 5? (prioritize availability) Or does the system go offline until network is restored? (prioritize consistency)

PACELC THEOREM



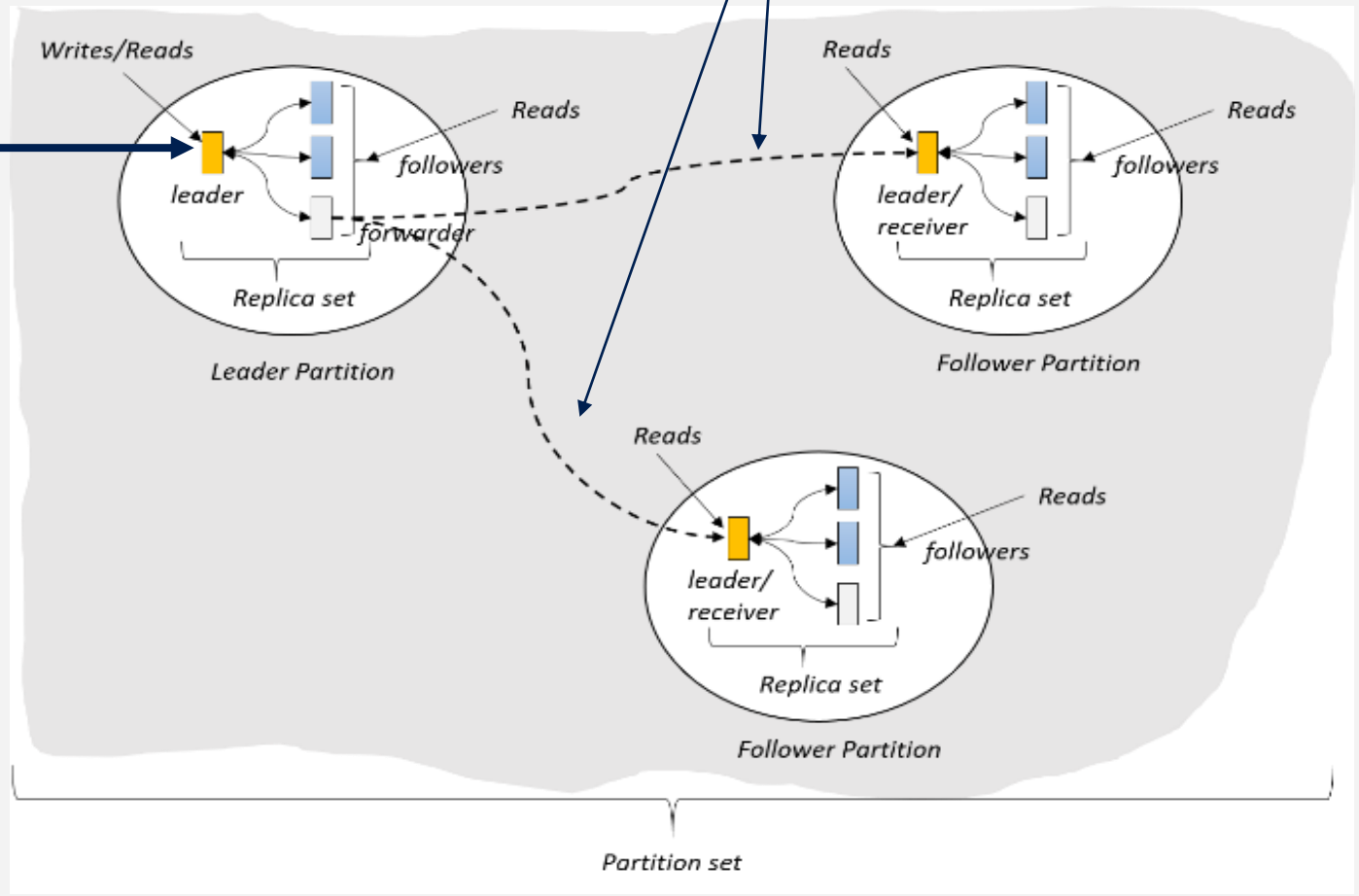
In the case of network partitioning (P) in a distributed computer system, one has to choose between availability (A) and consistency (C) (as per the CAP theorem), but else (E), even when the system is running normally in the absence of partitions, one has to choose between latency (L) and consistency (C).

CONSISTENCY (2 of 3)

Latency: packet of information can travel as fast as speed of light. Replication between distant geographic regions can take 100's of milliseconds

Value = ~~5~~ 6

Update 5 => 6



Value = ~~5~~ 6

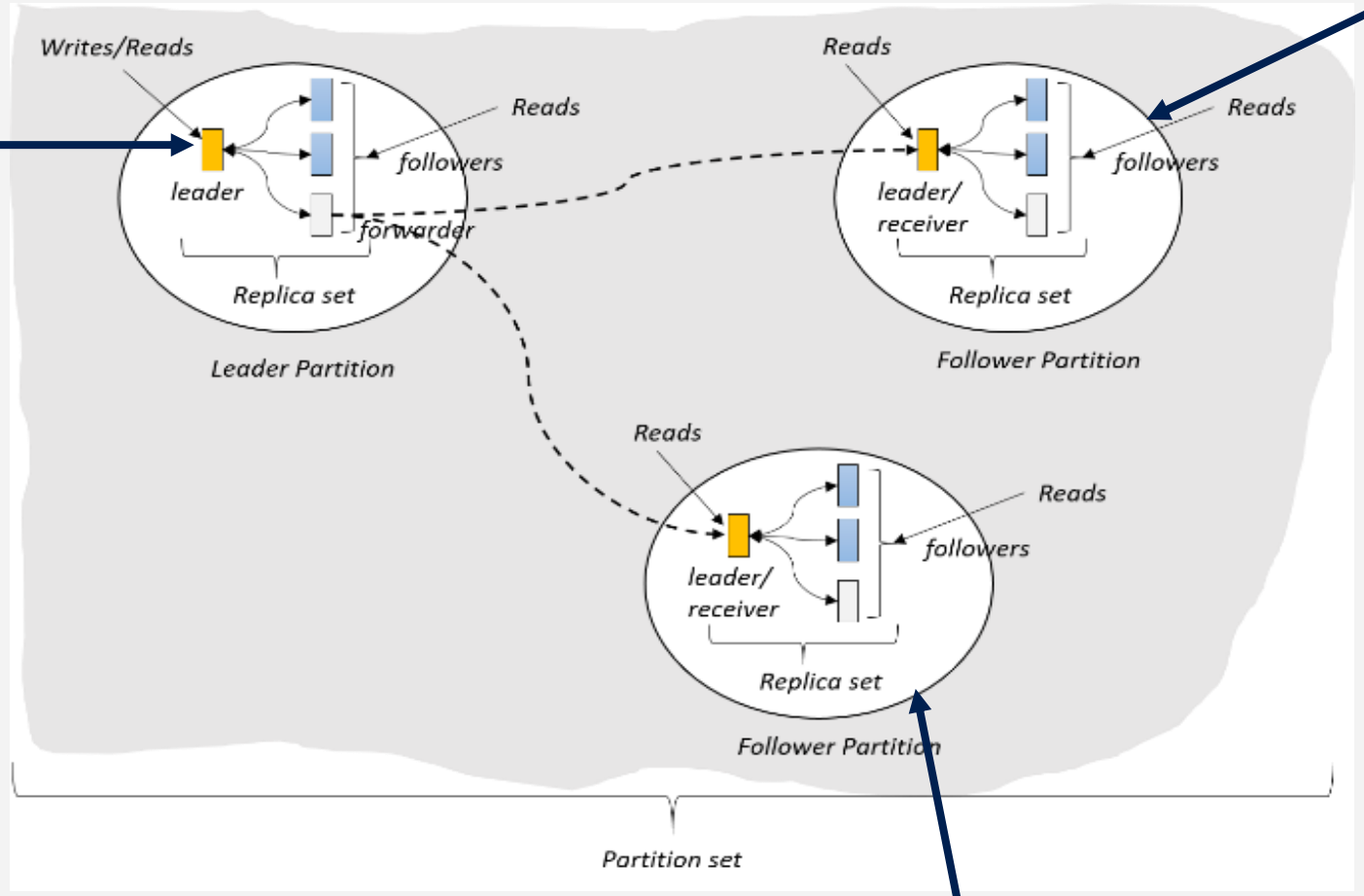
Value = 5

CONSISTENCY (3 of 3)

Reader A: What is the value?

Value = ~~5~~ 6

Update 5 => 6



Value = ~~5~~ 6

Value = 5

Should it Reader B see 5 immediately? (prioritize latency)

Does it see the same result as reader A? (quorum impacts throughput)

Does it sit and wait for 5 => 6 propagate? (prioritize consistency)

Reader B: What is the value?

FIVE WELL-DEFINED CONSISTENCY MODELS (2 of 2)

CHOOSE THE BEST CONSISTENCY MODEL FOR YOUR APP

Five well-defined, consistency models

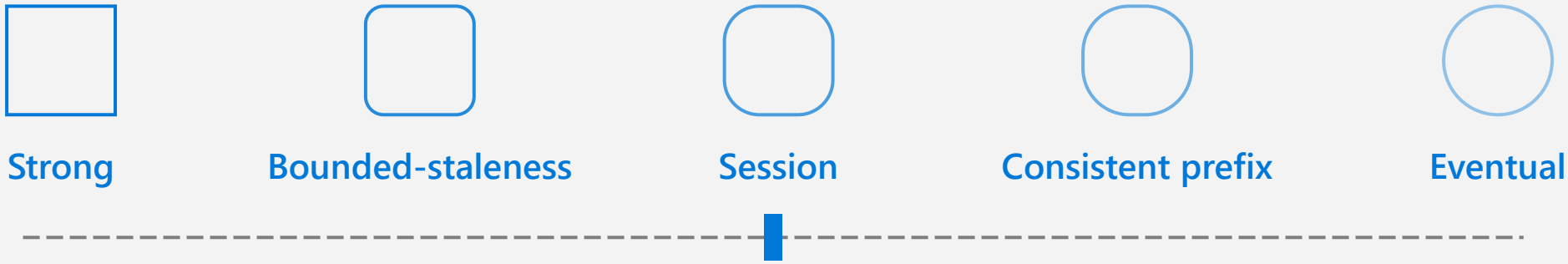
Overridable on a per-request basis

Provides control over performance-consistency tradeoffs, backed by comprehensive SLAs.

An intuitive programming model offering low latency and high availability for your planet-scale app.

CLEAR TRADEOFFS

- **Latency**
- **Availability**
- **Throughput**



CONSISTENCY MODELS - BREAKDOWN

Consistency Level	Guarantees
Strong	Linearizability (once operation is complete, it will be visible to all)
Bounded Staleness	Consistent Prefix. Reads lag behind writes by at most k prefixes or t interval Similar properties to strong consistency (except within staleness window), while preserving 99.99% availability and low latency.
Session	Consistent Prefix. Within a session: monotonic reads, monotonic writes, read-your-writes, write-follows-reads Predictable consistency for a session, high read throughput + low latency
Consistent Prefix	Reads will never see out of order writes (no gaps).
Eventual	Potential for out of order reads. Lowest cost for reads of all consistency levels.

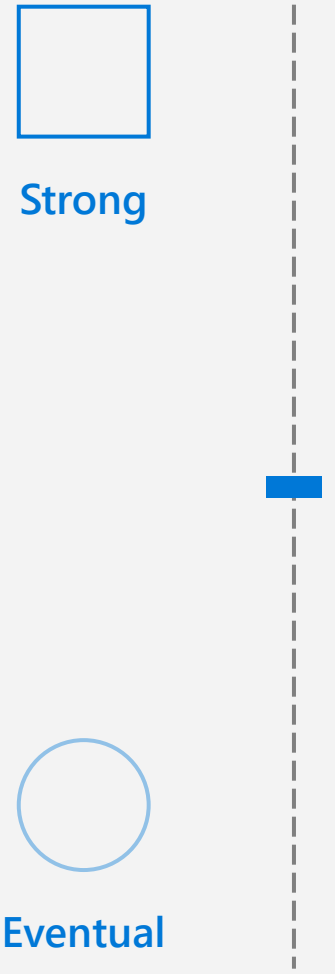
DEMYSTIFYING CONSISTENCY MODELS (1 of 2)

Strong consistency

Guarantees linearizability. Once an operation is complete, it will be visible to all readers in a strongly-consistent manner across replicas.

Eventual consistency

Replicas are eventually consistent with any operations. There is a potential for out-of-order reads. Lowest cost and highest performance for reads of all consistency levels.



STRONG CONSISTENCY

- Strong consistency offers a linearizability guarantee with the reads guaranteed to return the most recent version of an item.
- Strong consistency guarantees that a write is only visible after it is committed durably by the majority quorum of replicas.
- A client is always guaranteed to read the latest acknowledged write.
- The cost of a read operation (in terms of request units consumed) with strong consistency is higher than session and eventual, but the same as bounded staleness.

EVENTUAL CONSISTENCY

- Eventual consistency guarantees that in absence of any further writes, the replicas within the group eventually converge.
- Eventual consistency is the weakest form of consistency where a client may get the values that are older than the ones it had seen before.
- Eventual consistency provides the weakest read consistency but offers the lowest latency for both reads and writes.
- The cost of a read operation (in terms of RUs consumed) with the eventual consistency level is the lowest of all the Azure Cosmos DB consistency levels.

DEMYSTIFYING CONSISTENCY MODELS (2 of 2)

Bounded-staleness

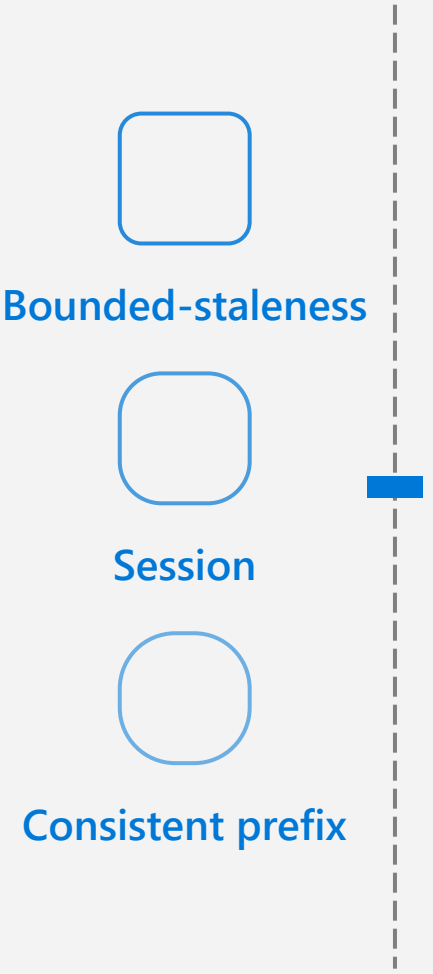
Consistent prefix. Reads lag behind writes by at most **k** prefixes or **t** interval. Similar properties to strong consistency except within staleness window.

Session

Consistent prefix. Within a session, reads and writes are monotonic. This is referred to as "read-your-writes" and "write-follows-reads". Predictable consistency for a session. High read throughput and low latency outside of session.

Consistent Prefix

Reads will never see out of order writes.



BOUNDED STALENESS CONSISTENCY

- Bounded staleness consistency guarantees that the reads may lag behind writes by at most K versions or prefixes of an item or t time-interval.
- Bounded staleness offers total global order except within the "staleness window." The monotonic read guarantees exist within a region both inside and outside the "staleness window."
- Bounded staleness provides a stronger consistency guarantee than session, consistent-prefix, or eventual consistency.
- The cost of a read operation (in terms of RUs consumed) with bounded staleness is higher than session and eventual consistency, but the same as strong consistency.

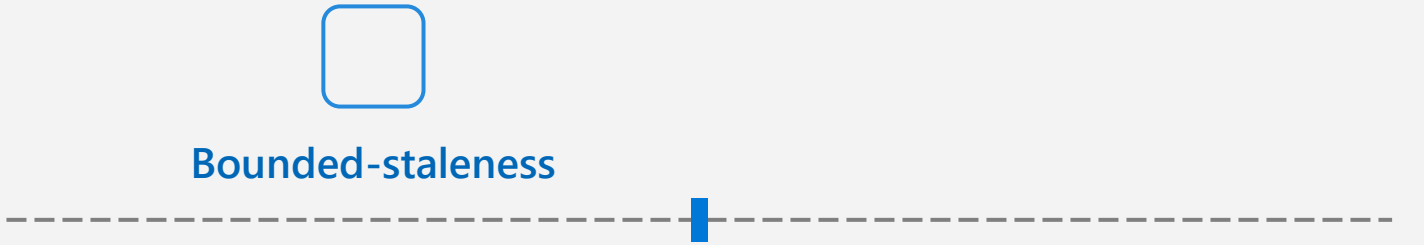
SESSION CONSISTENCY

- Unlike the global consistency models offered by strong and bounded staleness consistency levels, session consistency is scoped to a client session.
- Session consistency is ideal for all scenarios where a device or user session is involved since it guarantees monotonic reads, monotonic writes, and read your own writes (RYW) guarantees.
- Session consistency provides predictable consistency for a session, and maximum read throughput while offering the lowest latency writes and reads.
- The cost of a read operation (in terms of RUs consumed) with session consistency level is less than strong and bounded staleness, but more than eventual consistency.

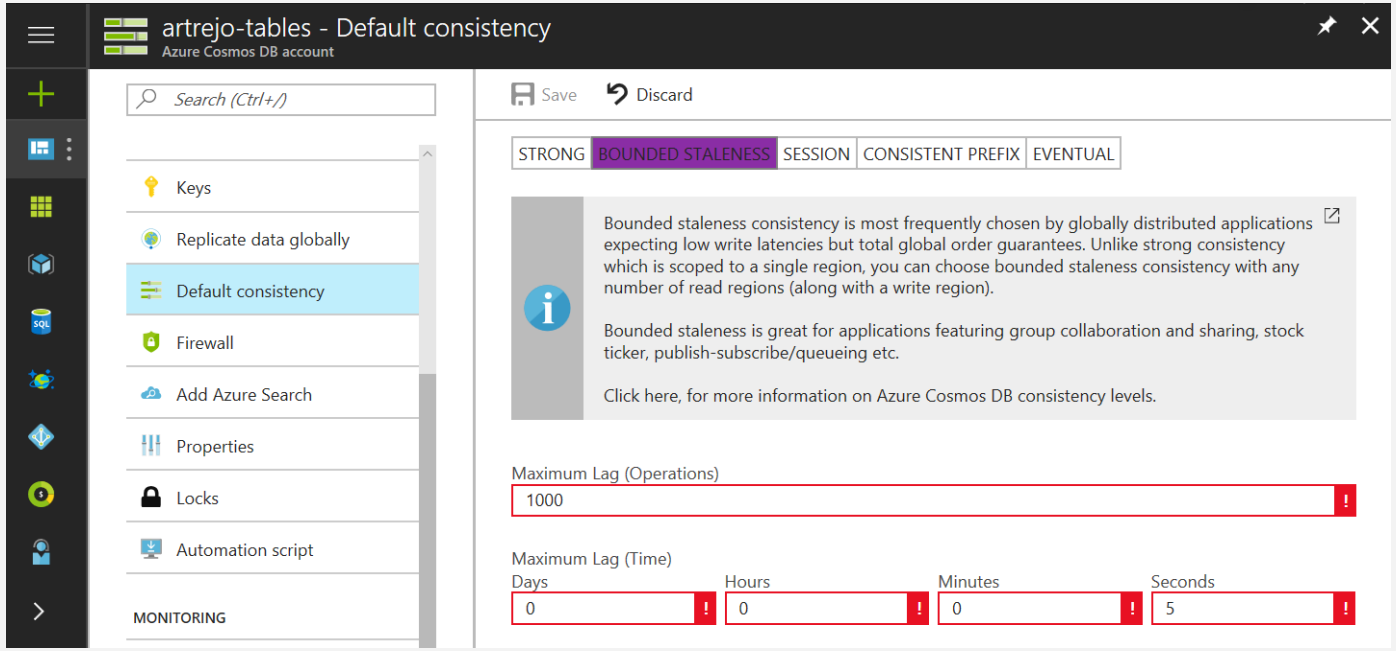
CONSISTENT PREFIX CONSISTENCY

- Consistent prefix guarantees that in absence of any further writes, the replicas within the group eventually converge.
- Consistent prefix guarantees that reads never see out of order writes. If writes were performed in the order **A**, **B**, **C**, then a client sees either **A**, **A**,**B**, or **A**,**B**,**C**, but never out of order like **A**,**C** or **B**,**A**,**C**.

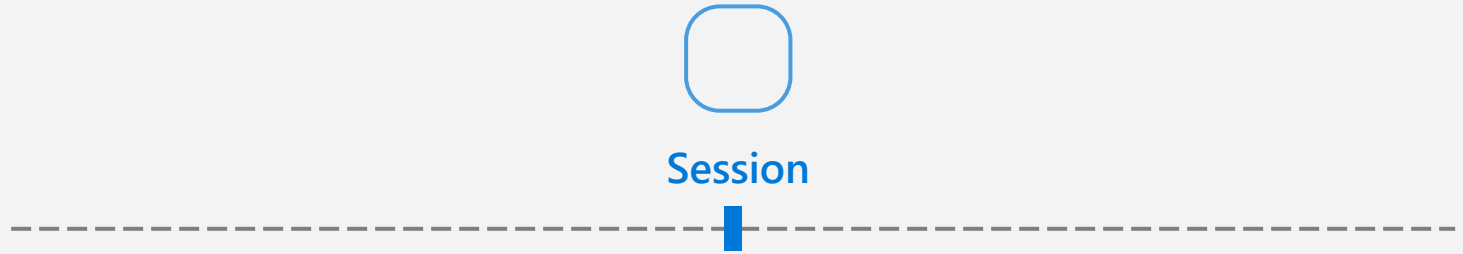
BOUNDED STALENESS IN THE PORTAL



BOUNDS ARE SET SERVER-SIDE VIA THE AZURE PORTAL



SESSION CONSISTENCY IN CODE



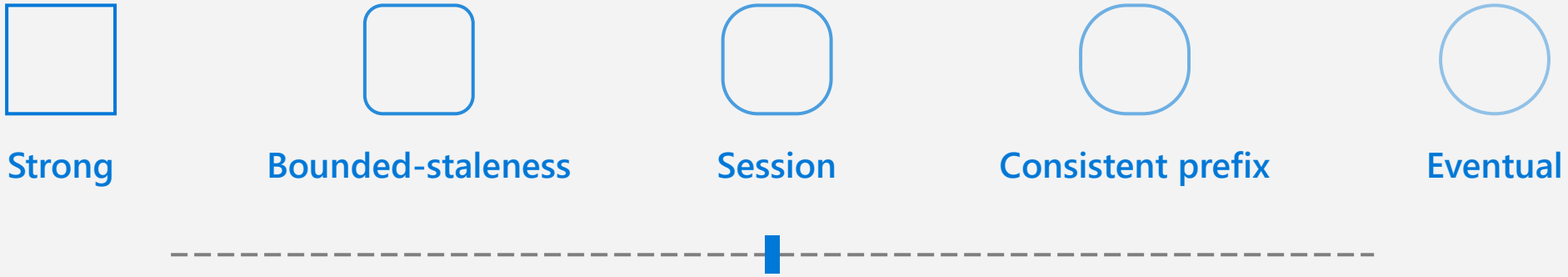
SESSION IS CONTROLLED USING A "SESSION TOKEN"

- Session tokens are automatically cached by the Client SDK
- Can be pulled out and used to override other requests (to preserve session between multiple clients)

```
string sessionToken;

using (DocumentClient client = new DocumentClient(new Uri(""), ""))
{
  ResourceResponse<Document> response = client.CreateDocumentAsync(
      collectionLink,
      new { id = "an id", value = "some value" }
  ).Result;
  sessionToken = response.SessionToken;
}
```

```
using (DocumentClient client = new DocumentClient(new Uri(""),
""))
{
  ResourceResponse<Document> read = client.ReadDocumentAsync(
      documentLink,
      new RequestOptions { SessionToken = sessionToken }
  ).Result;
}
```

RELAXING CONSISTENCY IN CODE



CONSISTENCY CAN BE RELAXED ON A PER-REQUEST BASIS

```
client.ReadDocumentAsync(
    documentLink,
    new RequestOptions { ConsistencyLevel = ConsistencyLevel.Eventual }
);
```

Dry Run MCPP
DP 420

Microsoft

Microsoft Confidential

All content is NDA unless otherwise stated.

To answer the questions

- please use 👍 💗 👏 feature if you want to answer a question versus posting it in the chat window

Ctrl+Shift+K is a quick way to vote or lower your hand

"Raise Your Hand" feature if you want to ask a question

Q&A

Microsoft Enregistrez vous dès maintenant au prochain Webinars Data AI

Event Webinar (Les jeudis de la Data & AI) - L200/300	Date	Duration (min)	Link
Azure Synapse	22/09/2022	120	https://msevents.microsoft.com/event?id=857781749
Les solutions SQL dans Azure (PaaS, IaaS, SaaS)	29/09/2022	120	https://msevents.microsoft.com/event?id=502366997
Déploiement et sécurisation des workspaces Azure Machine learning	06/10/2022	120	https://msevents.microsoft.com/event?id=1505714138
Azure Scale Analytics - Architectures Data Mesh dans Azure avec Azure Synapse, Microsoft Purview et Azure Data Share	13/10/2022	120	https://msevents.microsoft.com/event?id=139685175
MLOps avec Azure Machine Learning	20/10/2022	120	https://msevents.microsoft.com/event?id=1245885767
SQL Server 2022 et hybridation native avec Azure SQL Managed Instance	10/11/2022	120	https://msevents.microsoft.com/event?id=145826476
Machine Learning dans Azure Synapse Analytics	17/11/2022	120	https://msevents.microsoft.com/event?id=3637723312
Azure Cosmos DB et IA	24/11/2022	120	https://msevents.microsoft.com/event?id=2646013445
Azure et les Services Cognitifs	08/12/2022	120	https://msevents.microsoft.com/event?id=3772037220
La gouvernance de données dans Azure avec Microsoft Purview	15/12/2022	120	https://msevents.microsoft.com/event?id=1499560981
MLOps avec Azure Machine Learning	12/01/2023	120	https://msevents.microsoft.com/event?id=4115194515
Data processing dans Azure ave Azure Synapse, Azure Batch, Spark, Notebook, etc.	19/01/2023	120	https://msevents.microsoft.com/event?id=1537241181
Déploiement et sécurisation des workspace Azure Synapse	26/01/2023	120	https://msevents.microsoft.com/event?id=1806467748
Azure Machine Learning pour les Citizen Data Scientists	09/02/2023	120	En cours
PowerBI - Self Service Analytics	16/02/2023	120	https://msevents.microsoft.com/event?id=1401519679
L'IA responsable avec Azure machine learning	09/03/2023	120	https://msevents.microsoft.com/event?id=2072953112
Machine Learning dans Azure Synapse Analytics	16/03/2023	120	https://msevents.microsoft.com/event?id=3413014857
Les bases de données Open Source dans le cloud Azure	23/03/2023	120	https://msevents.microsoft.com/event?id=2727487131
Hybridation des services de Machine Learning Azure	06/04/2023	120	https://msevents.microsoft.com/event?id=1624914222
La gouvernance de données dans Azure avec Microsoft Purview	13/04/2023	120	https://msevents.microsoft.com/event?id=3909342839
Les solutions SQL dans Azure (PaaS, IaaS, SaaS)	04/05/2023	120	https://msevents.microsoft.com/event?id=1162207895
Data processing dans Azure ave Azure Synapse, Azure Batch, Spark, Notebook, etc.	16/05/2023	120	https://msevents.microsoft.com/event?id=3517068442
Hybridation des services de données Azure	24/05/2023	120	https://msevents.microsoft.com/event?id=2996507398
Self Service Analytics	01/06/2023	120	En cours

Annexes