Microsoft

Microsoft Azure Training Day: Data and Analytics
Azure Cosmos DB

Marco Parenzan (twitter: @marco_parenzan)
Solution Sales Specialist @ Insight (email: marco.parenzan@insight.com)
Microsoft Azure MVP (mvpid: 5000823)
Community Lead @ 1nn0va (url: https://www.facebook.com/1nn0va)

WHAT IS AZURE COSMOS DB

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

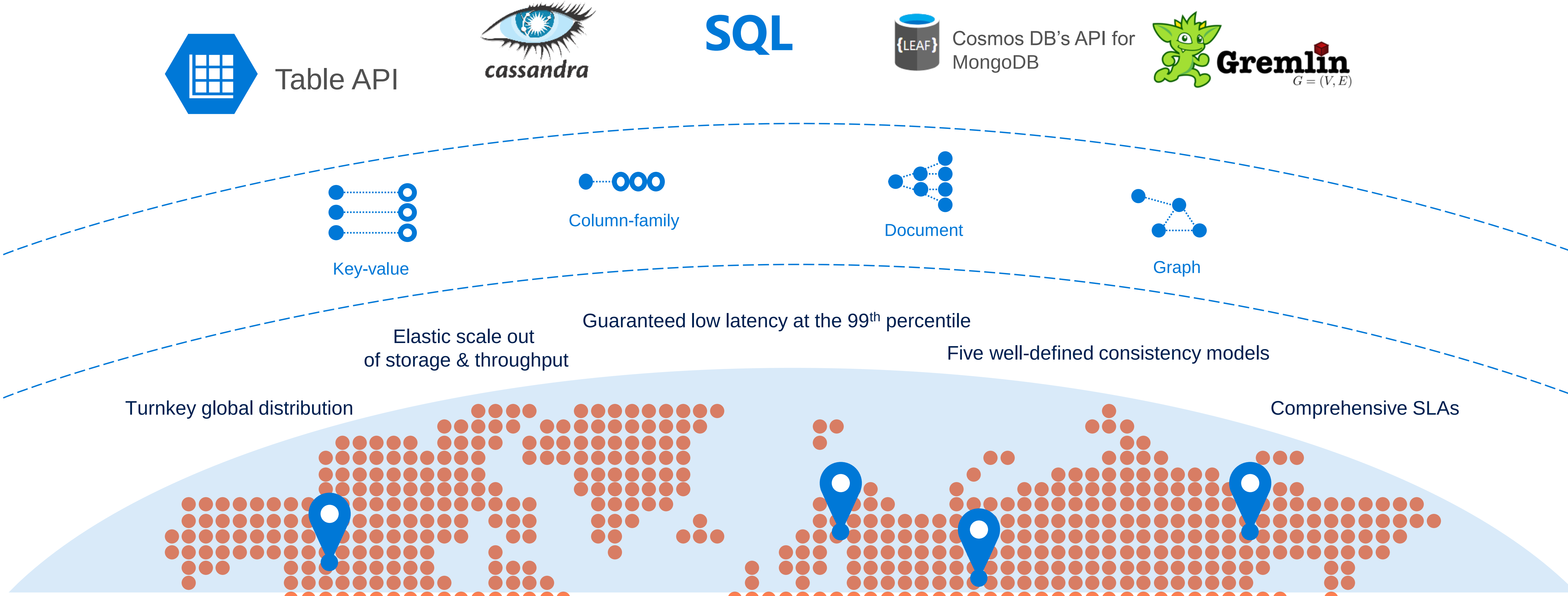


Table API

cassandra

SQL

Cosmos DB’s API for MongoDB

Gremlin $G = (V, E)$

Key-value

Column-family

Document

Graph

Guaranteed low latency at the 99th percentile

Elastic scale out of storage & throughput

Five well-defined consistency models

Turnkey global distribution

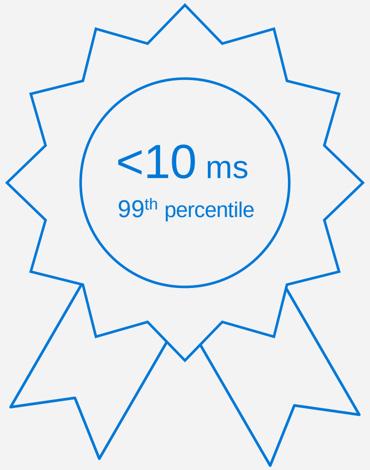
Comprehensive SLAs

Comprehensive SLA’s

Run your app on world-class infrastructure.

Azure Cosmos DB is the only service with financially-backed SLAs for millisecond latency at the 99th percentile, 99.999% HA and guaranteed throughput and consistency

Latency



<10 ms

99th percentile

HA

99.999%

Throughput

Guaranteed

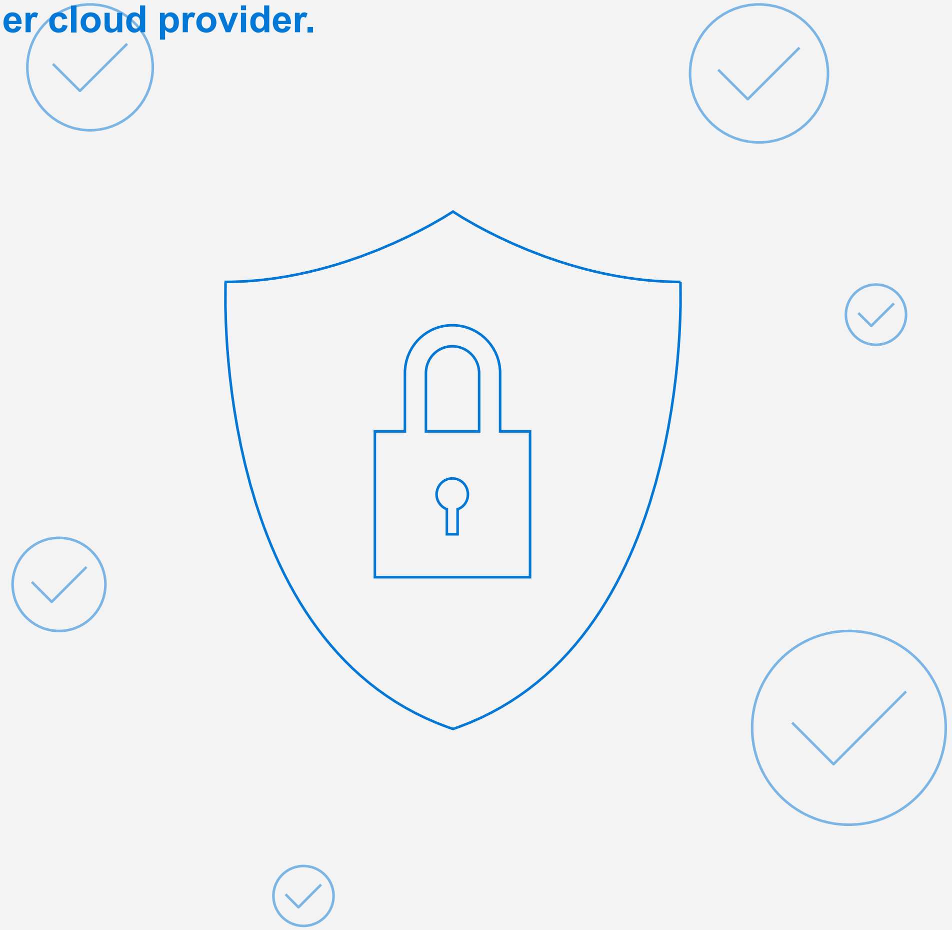
Consistency

Guaranteed

Trust your Data to Industry-Leading Security & Compliance

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

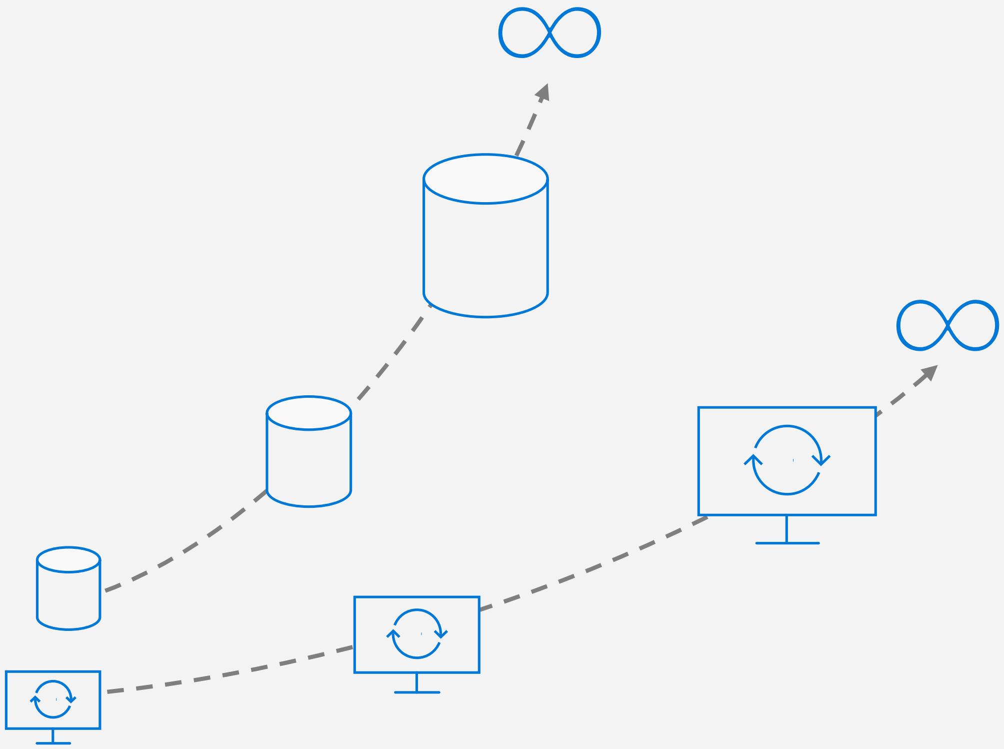
- Enterprise grade security
- Encryption at Rest and Transit
- Encryption is enabled automatically by default
- Comprehensive Azure compliance certification



Elastically Scale Storage and Throughput

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

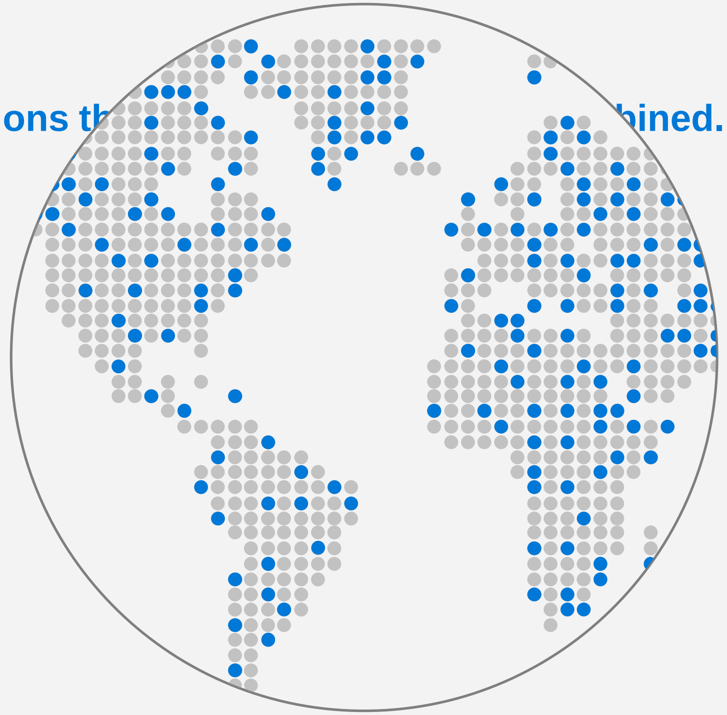
- Elastically scale throughput from 10 to 100s of millions of requests/sec across multiple regions
- Support for requests/sec for different workloads
- Pay only for the throughput and storage you need



Turnkey Global Distribution

Put your data where your users are in minutes

Automatically replicate all your data around the world, and across more regions th bined.



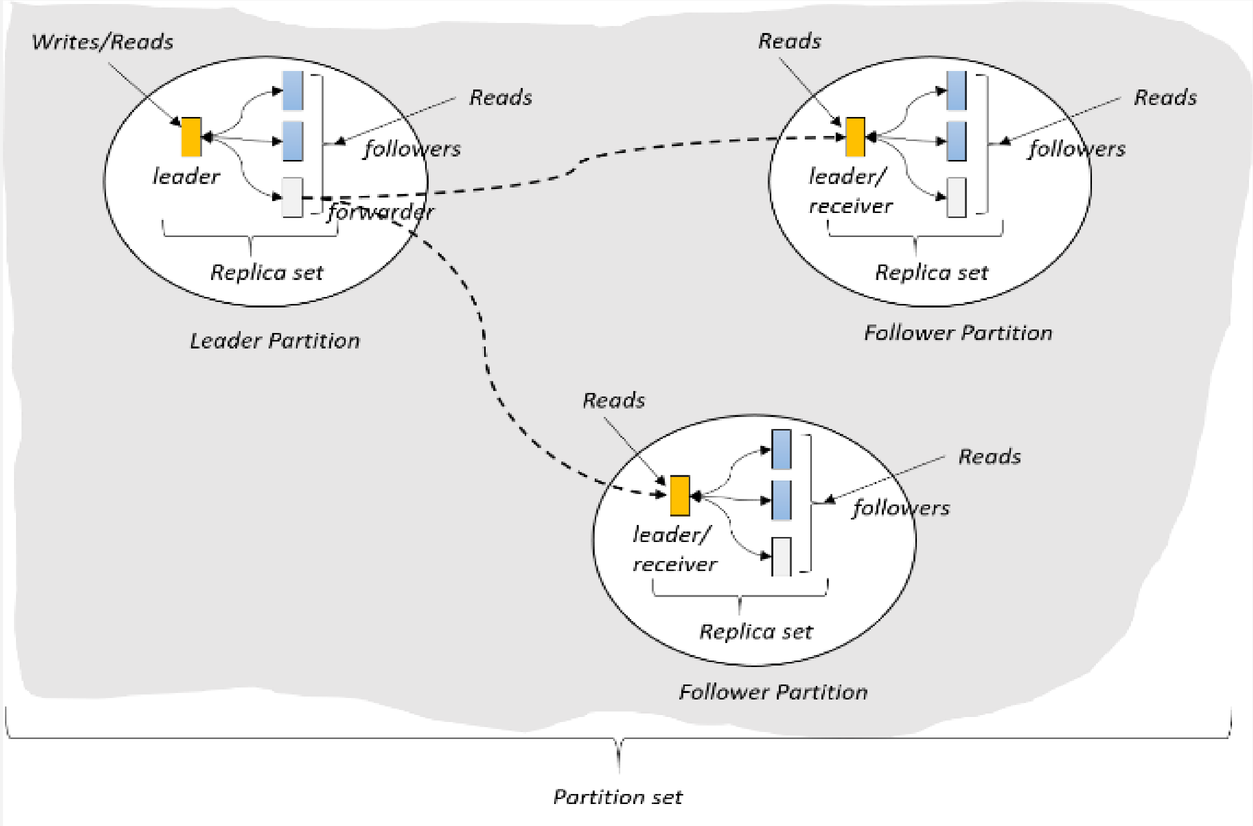
- Available in [all Azure regions]
- Manual and automatic failover
- Automatic & synchronous multi-region replication
- Configure multiple write regions to further reduce latency and increase availability

CONSISTENCY

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



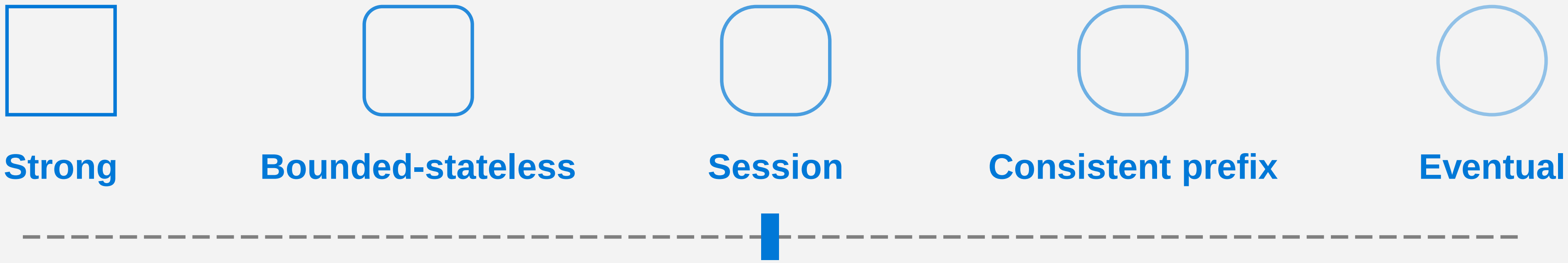
Five Well-Defined Consistency Models

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.



Strong

Bounded-stateless

Session

Consistent prefix

Eventual

Multiple Data Models and API's

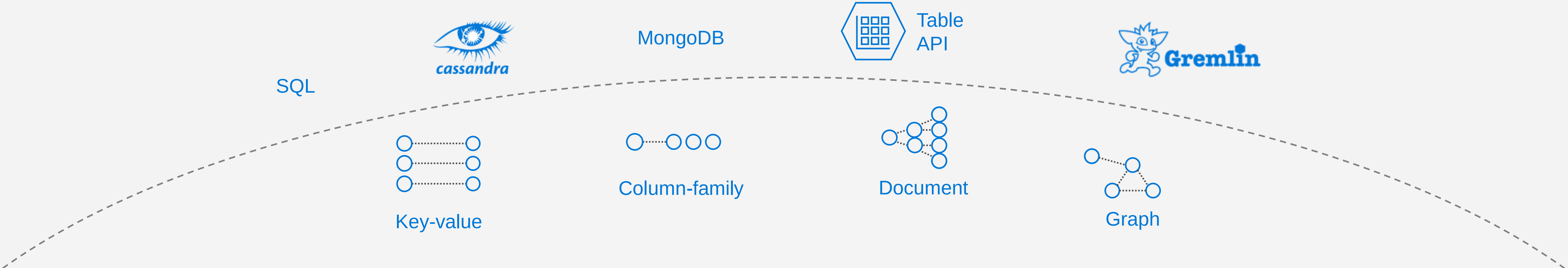
Use the model that fits your requirements, and the apis, tools, and frameworks you prefer

Cosmos DB offers a multitude of APIs to access and query data including, SQL, various popular OSS APIs, and native support for NoSQL workloads.

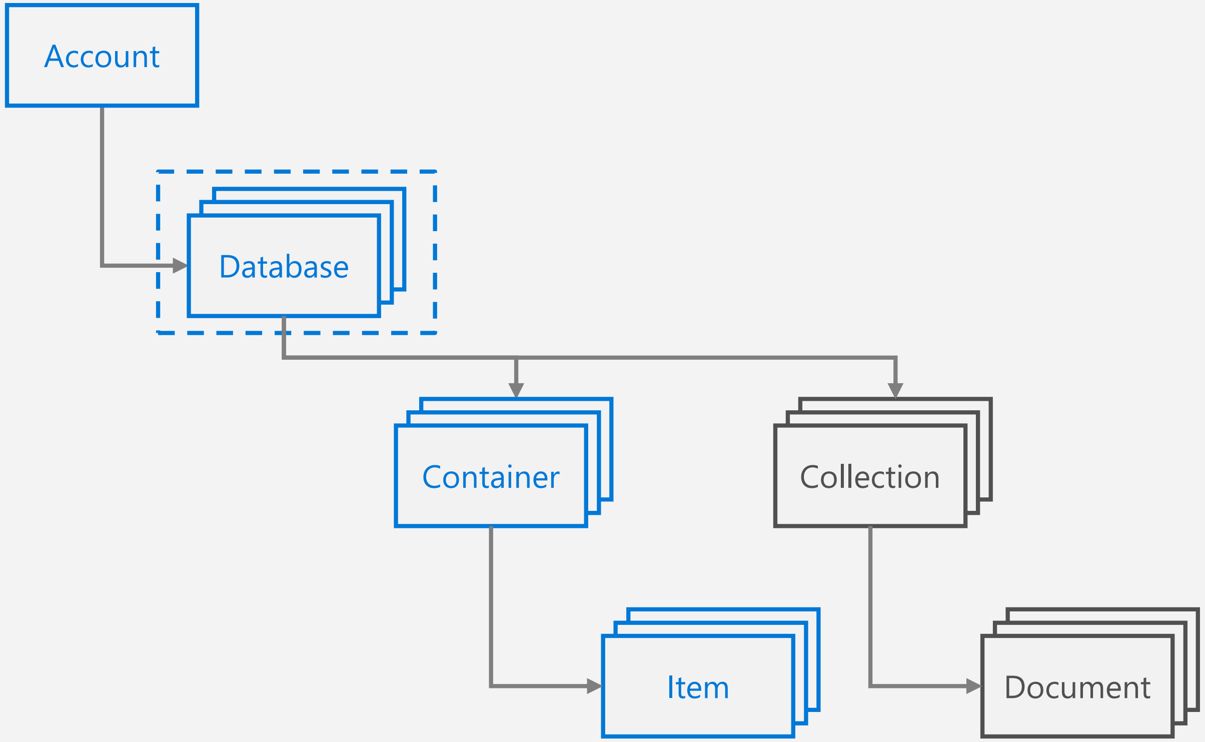
Use key-value, columnar, graph, and document data

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

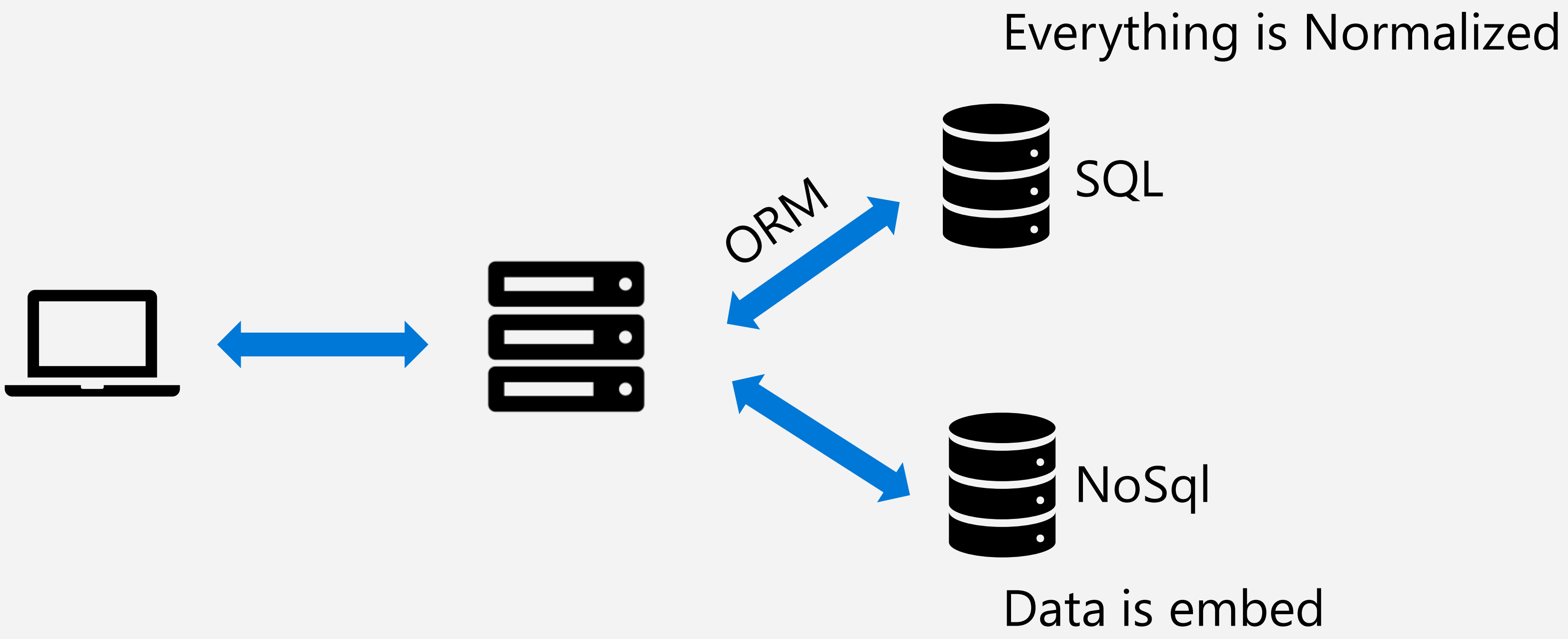
Blazing fast queries with no lag



RESOURCE MODEL

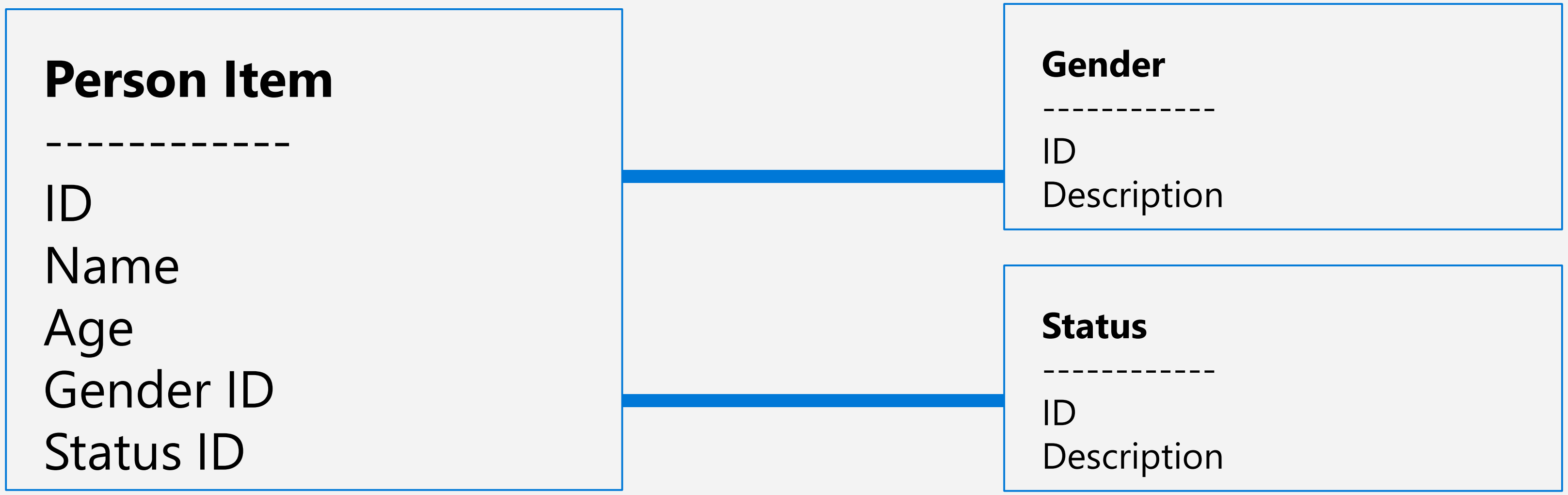


EMBED vs REFERENCE



EMBED vs REFERENCE

Relational modelling



Person Item

ID
Name
Age
Gender ID
Status ID

Gender

ID
Description

Status

ID
Description

Non-relational modeling

```
{
        "ID": 1,
        "Name": “John",
        “Age": “42",
        “GenderID": 1,
        “GenderDescription": “male"
        “StatusID": 2,
        “StatusDescription”: “married”
}
```

EMBED vs REFERENCE

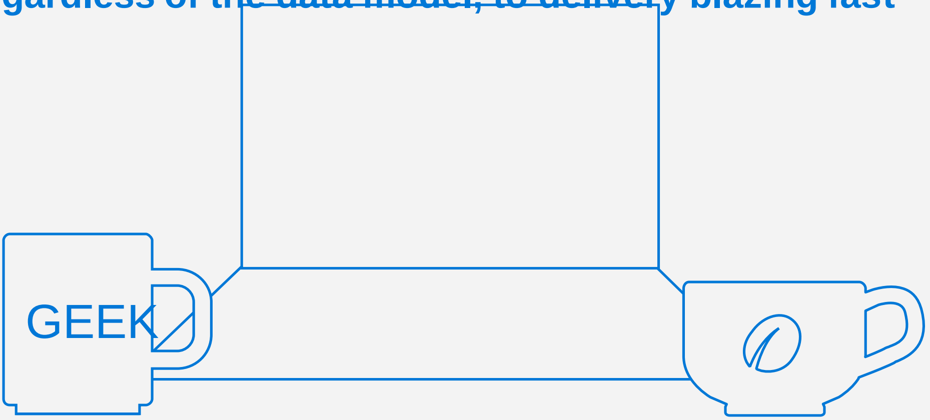
When to embed?
When to reference?
When to put different types in the same collection?
Sometimes the best choice is to do both

- Reference less frequently used data
- Embed more frequently used data

Handle any Data with no Schema or Indexing Required

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
- Freedom from schema + index management
- Works across every data model
- Ingest and serve data back out in milliseconds



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

Index POLICIES

CUSTOM INDEXING POLICIES

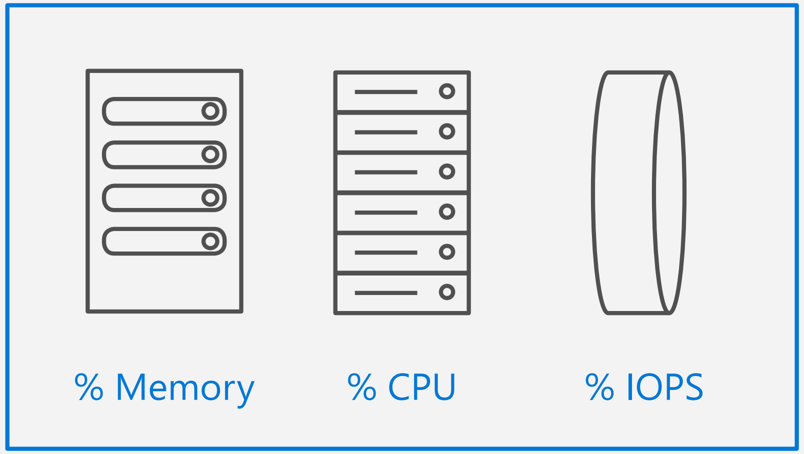
Though all Azure Cosmos DB data is indexed by default, you can specify a custom indexing policy for your collections. Custom indexing policies allow you to design and customize the shape of your index while maintaining schema flexibility.

- 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/*"
    }]
}
```

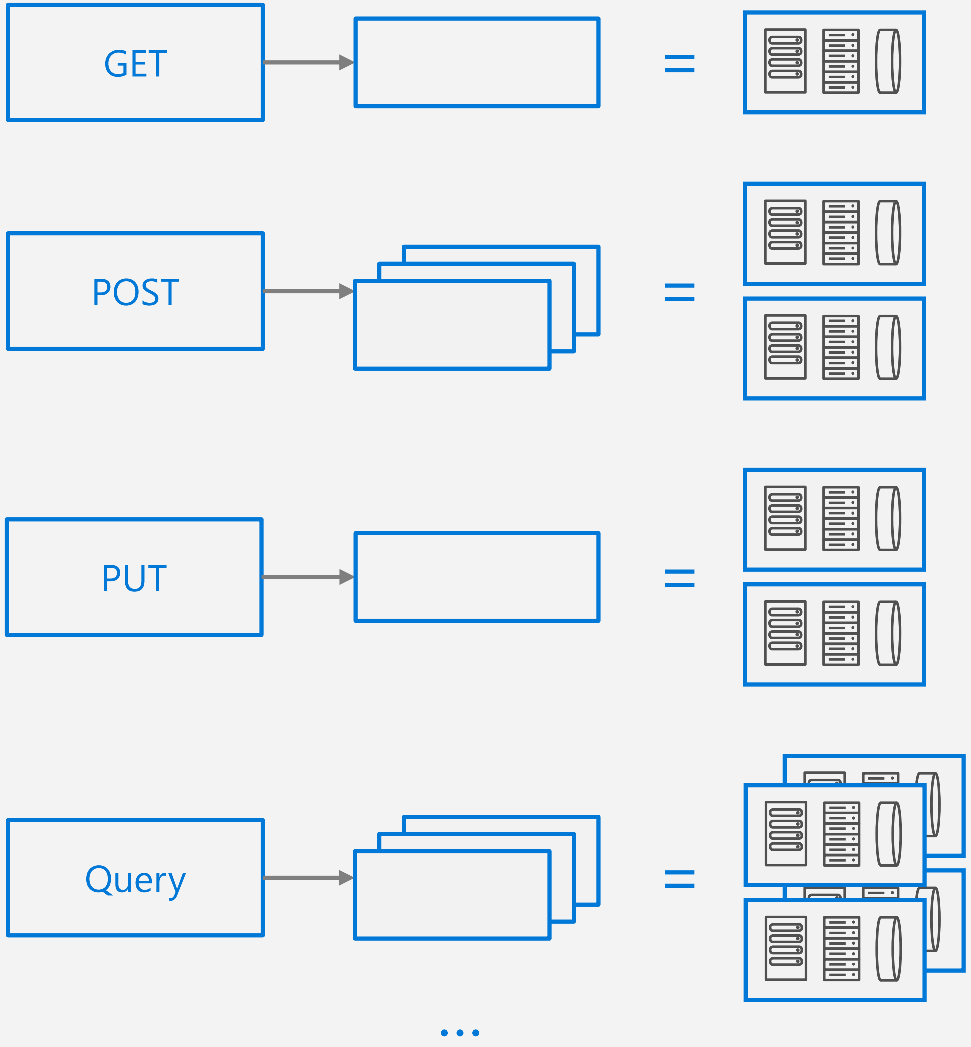
Request Units

- 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

- 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 Unit Pricing Example

Storage Cost

Avg Record Size (KB)	1
Number of Records	100,000,000
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

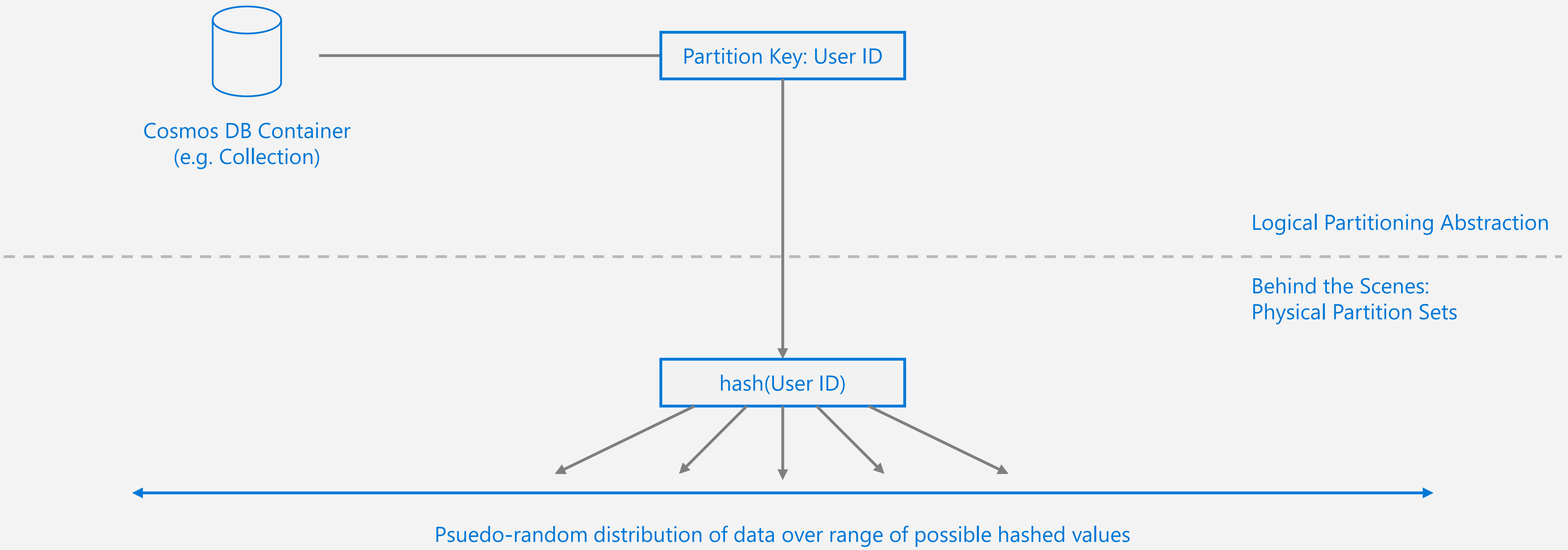
Total RU/sec	900
Monthly Cost per 100 RU/sec	$6.00
Expected Monthly Cost for Throughput	$54.00

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/

Partitions



Cosmos DB Container
(e.g. Collection)

Partition Key: User ID

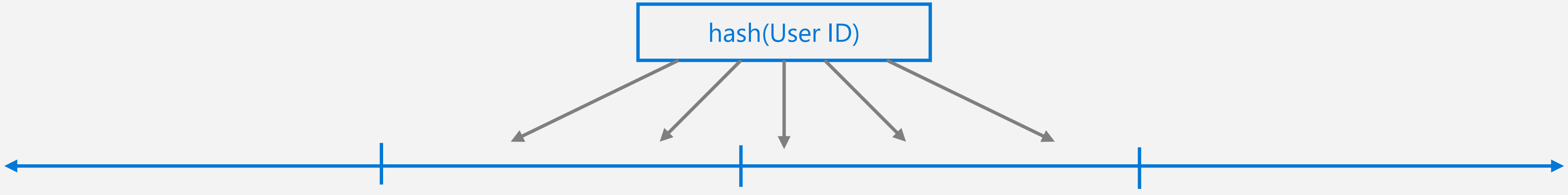
Logical Partitioning Abstraction

Behind the Scenes:
Physical Partition Sets

hash(User ID)

Psuedo-random distribution of data over range of possible hashed values

Partitions

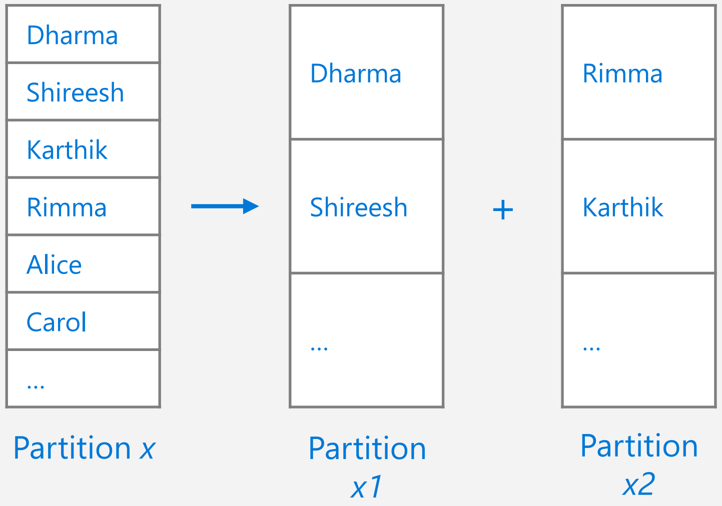


hash(User ID)

Pseudo-random distribution of data over range of possible hashed values

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.



Dharma
Shireesh
Karthik
Rimma
Alice
Carol
…

Partition *x*

Dharma
Shireesh
…

Partition *x1*

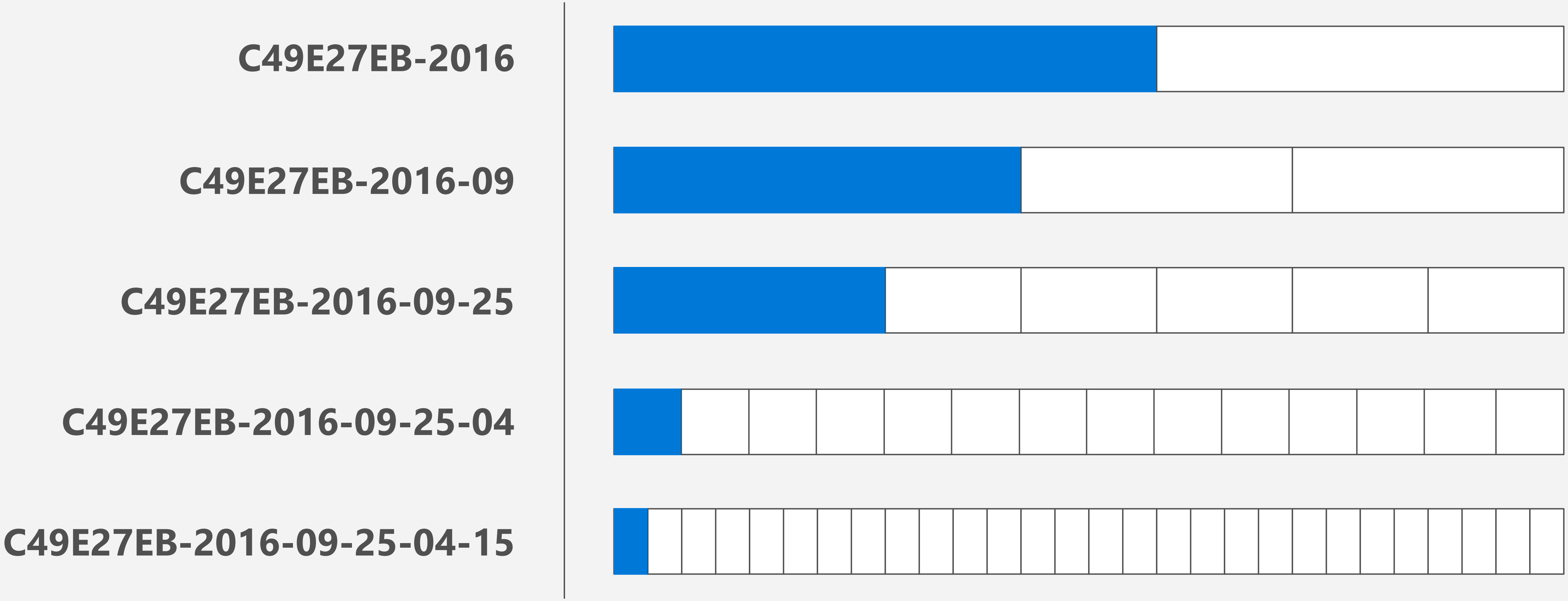
+

Rimma
Karthik
…

Partition *x2*

Partition Key Scenario

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



C49E27EB-2016

C49E27EB-2016-09

C49E27EB-2016-09-25

C49E27EB-2016-09-25-04

C49E27EB-2016-09-25-04-15

Example – Contoso Connected Car

Partitions

Best Practices: Design Goals for Choosing a Good Partition Key

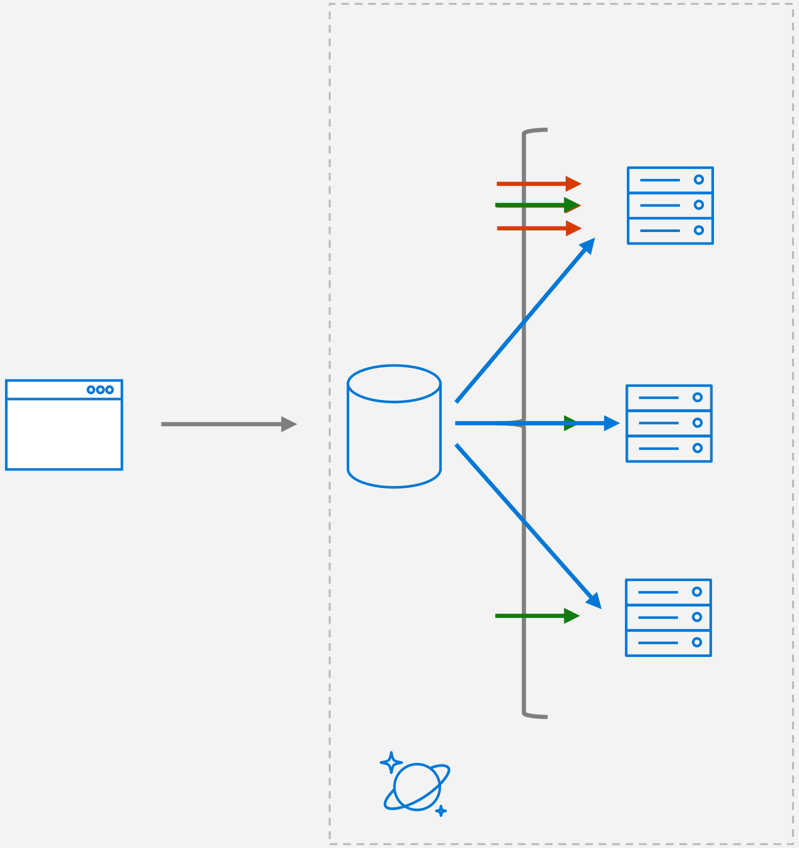
- Distribute the overall request + storage volume
  - Avoid “hot” partition keys
- Partition Key is scope for multi-record transactions and routing queries
  - Queries can be intelligently routed via partition key
  - Omitting partition key on query requires fan-out

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

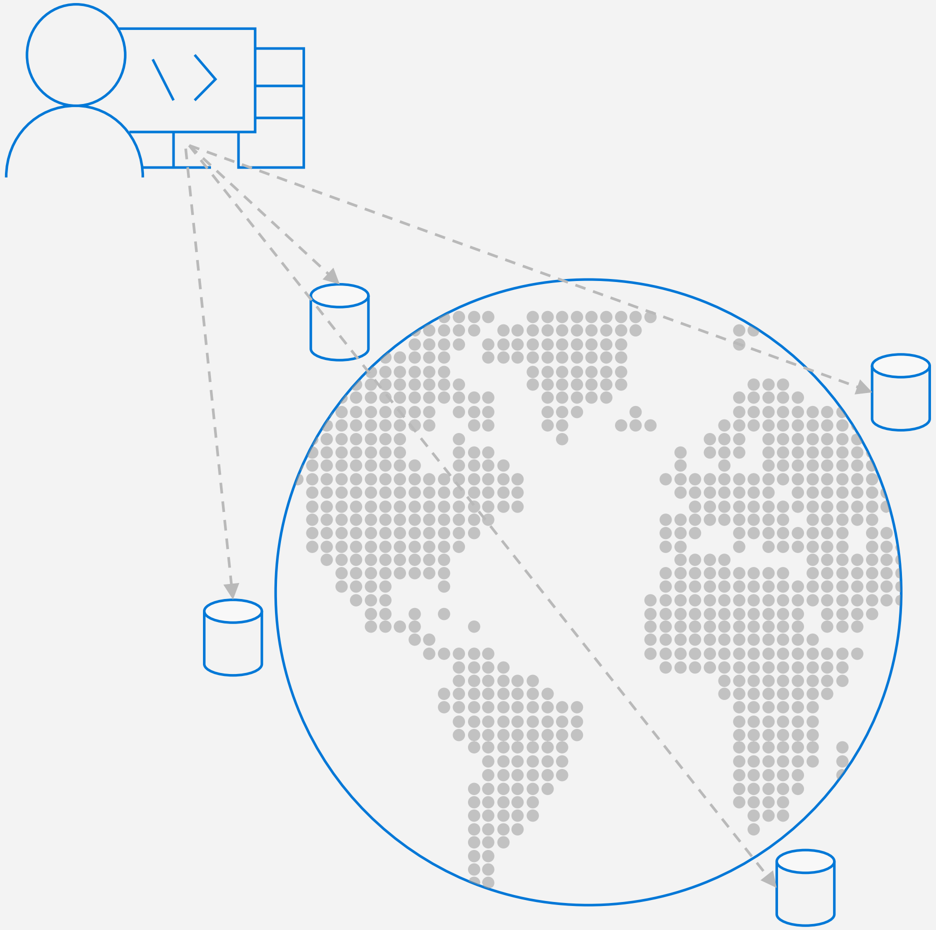
- Build a POC to strengthen your understanding of the workload and iterate (avoid analyses paralysis)
- Don’t be afraid of having too many partition keys
  - Partitions keys are logical
  - More partition keys → more scalability



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 - 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

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"
            ]
        }
    }
]
```

INTRA-DOCUMENT JOIN

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

Paginated Query REsults

Pagination with hasNext() in Java:

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

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

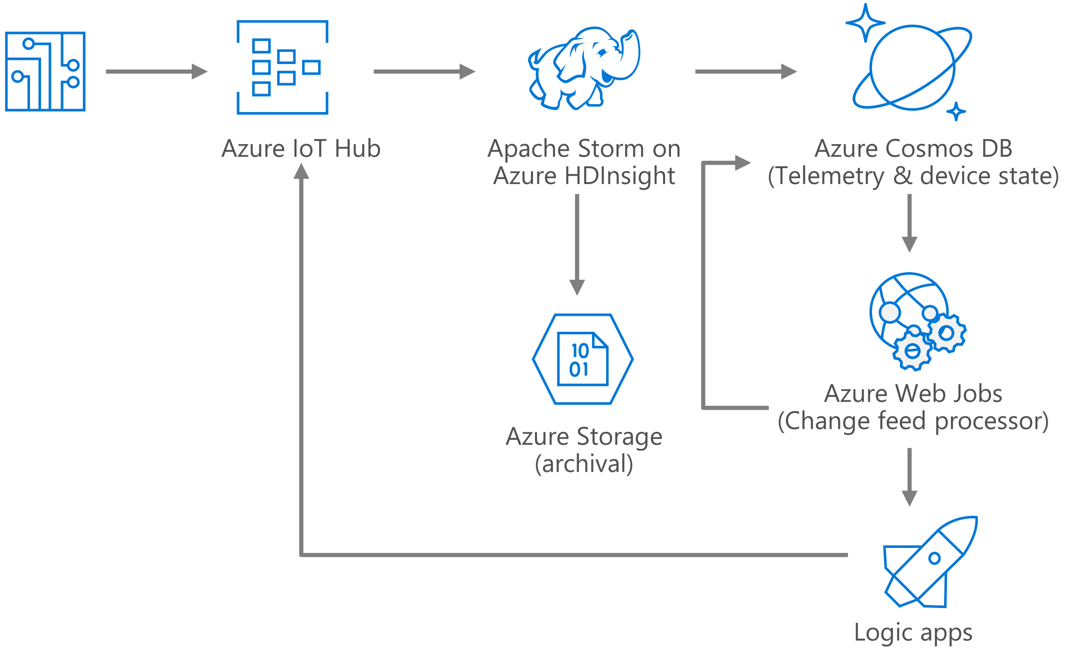
Massive Scale Telemetry Stores for IOT

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


Azure IoT Hub
Apache Storm on Azure HDInsight
Azure Cosmos DB (Telemetry & device state)
Azure Storage (archival)
Azure Web Jobs (Change feed processor)
Logic apps

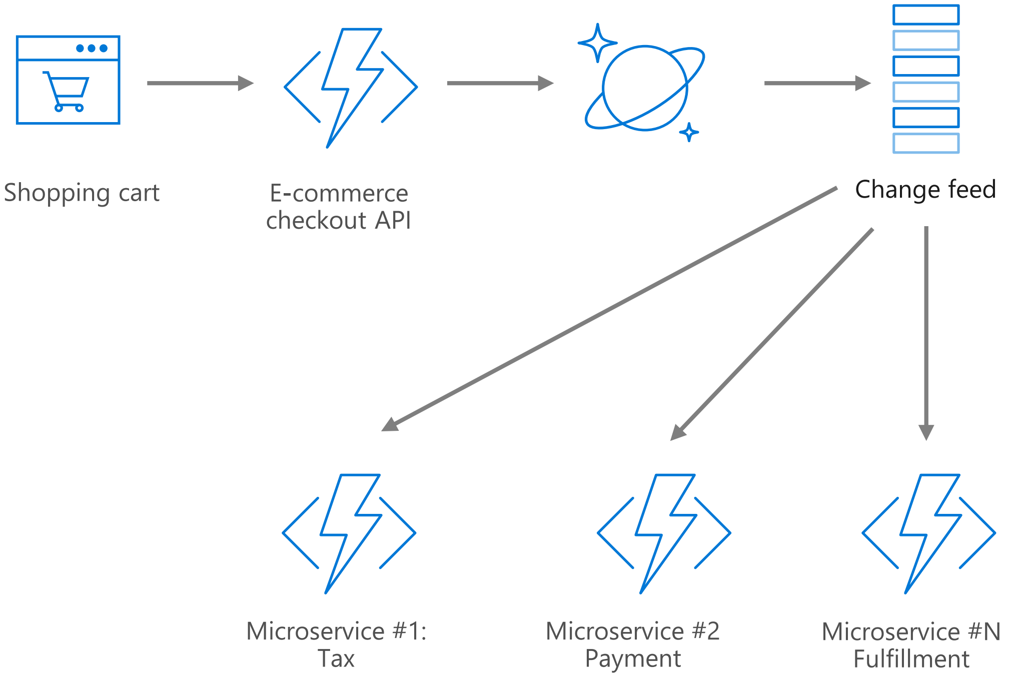
TOYOTA Honeywell LG CNS Johnson Controls

Event Sourcing & Microservice architecture

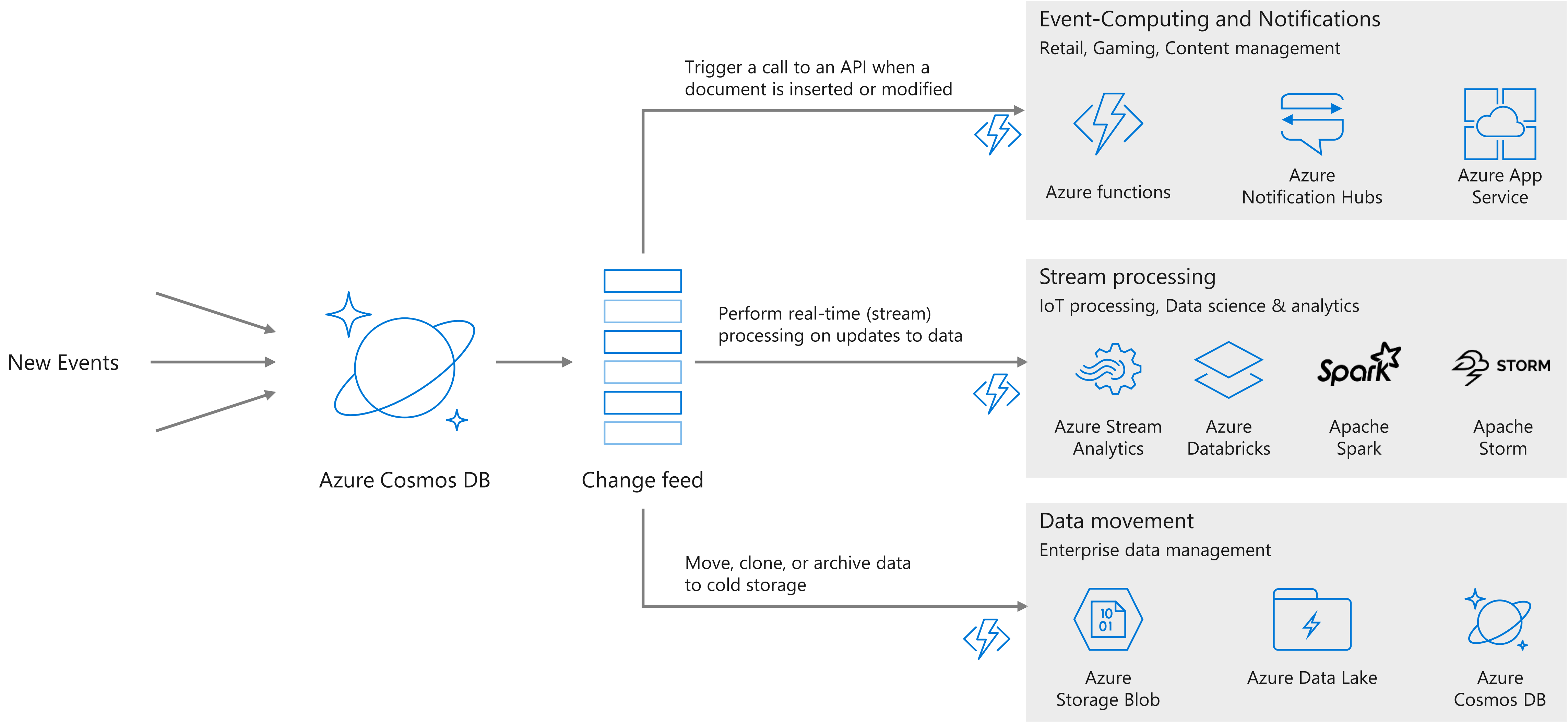
- IoT, gaming, retail and operational logging applications need to track and respond to tremendous amount of data being ingested, modified or removed from a globally-scaled database.

- Using Cosmos DB Change Feed:

- Listen for any changes in a container with Azure Functions
- Process ordered list of changed documents
- Trigger downstream microservices, with Cosmos as "source of truth" for all events in system



Change Feed Scenarios



New Events

Azure Cosmos DB

Change feed

Trigger a call to an API when a document is inserted or modified

Perform real-time (stream) processing on updates to data

Move, clone, or archive data to cold storage

Event-Computing and Notifications

Retail, Gaming, Content management

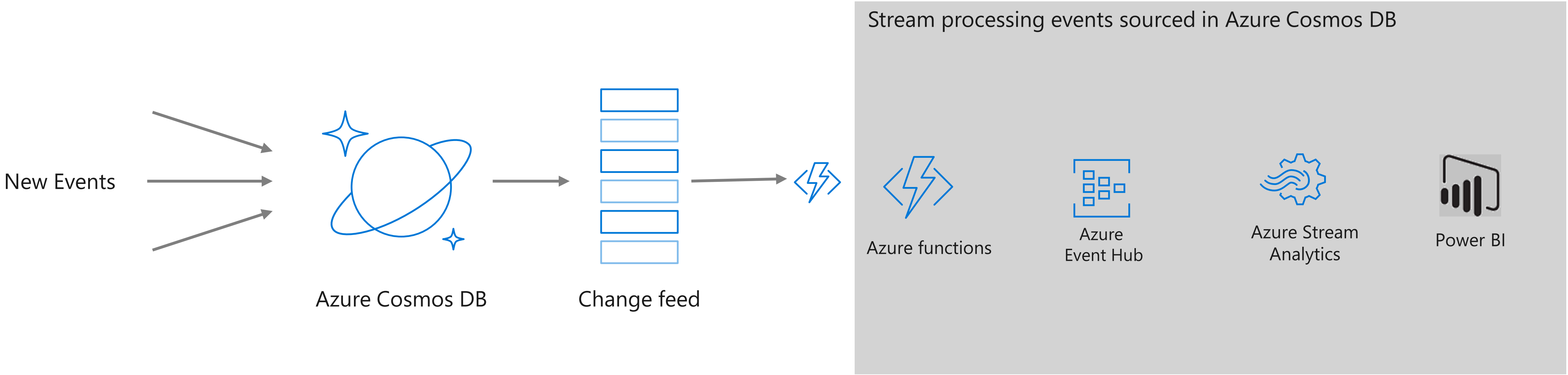
Stream processing

IoT processing, Data science & analytics

Data movement

Enterprise data management

The Architecture for Todays Experience



Ideal for E-commerce, Gaming, and IOT

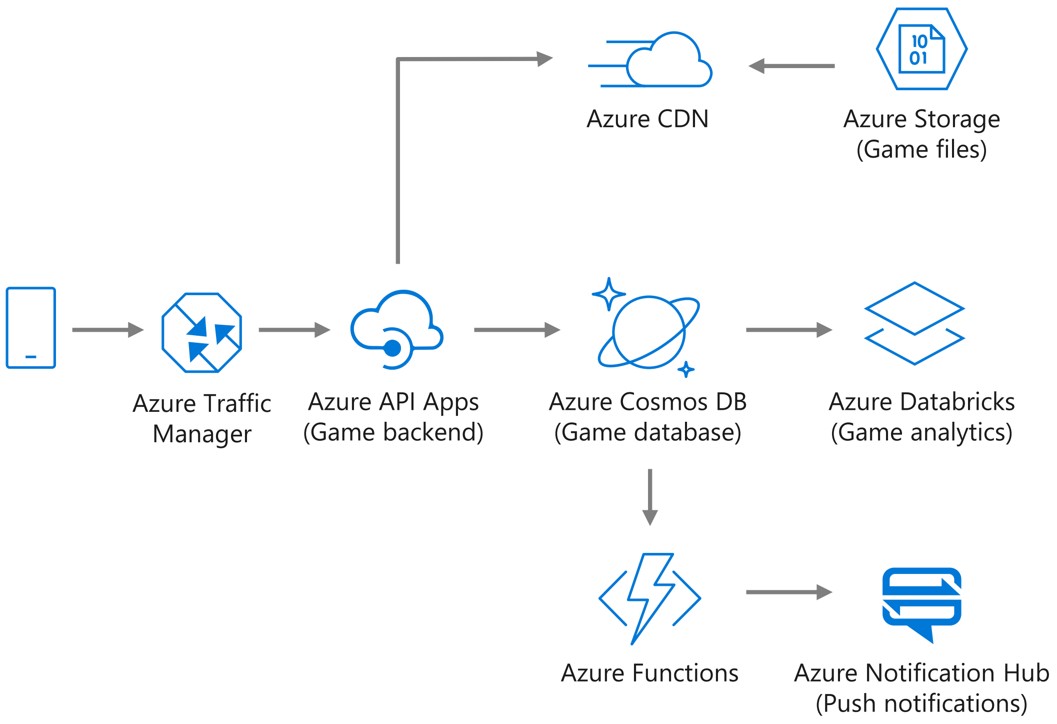
Maintain service quality during high-traffic periods requiring massive scale and performance.

Instant, elastic scaling handles traffic bursts

Uninterrupted global user experience

Low-latency data access (<10ms at P99) to process large and changing user bases

High availability (99.999%) across multiple data centers


Azure CDN
Azure Storage (Game files)
Azure Traffic Manager
Azure API Apps (Game backend)
Azure Cosmos DB (Game database)
Azure Databricks (Game analytics)
Azure Functions
Azure Notification Hub (Push notifications)

NEXT GAMES Domino's jet XBOX LIVE