IntelliPaat

Designing and Implementing an Azure Data Solution

DP 200 and DP 201

IntelliPaat

Data Lake and Azure Cosmos DB

Agenda

01 Data Lake Key Concepts

02 Loading Data into ADL with Azure Data Factory

03 Azure Cosmos DB

04 Working with Azure Cosmos DB

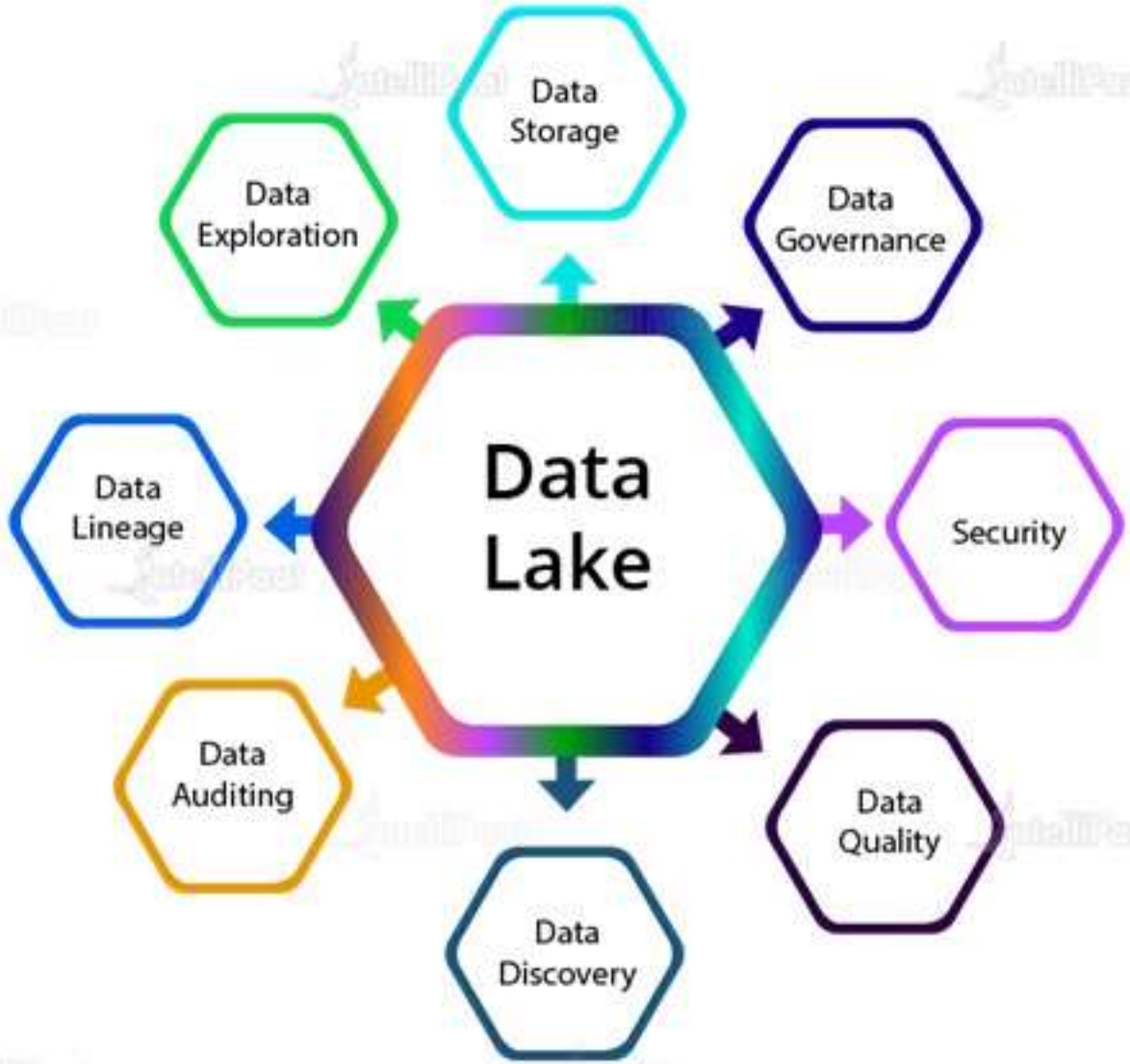
05 Azure Blob Storage

06 Data Partitioning

07 Consistency Levels in Azure Cosmos DB

Data Lake Key Concepts

Data Lake Key Concepts

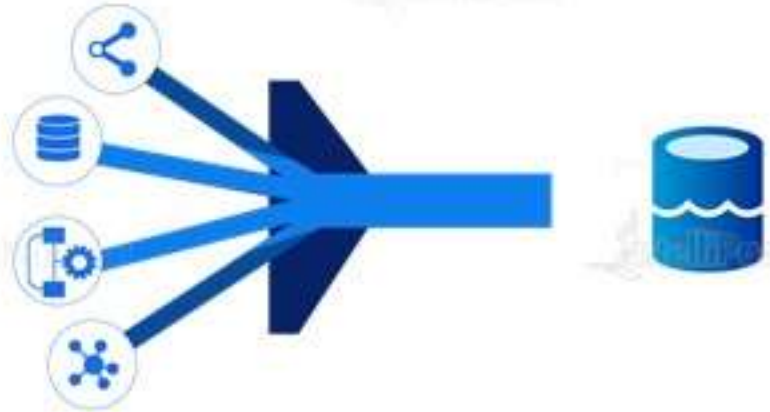


Data Lake Key Concepts

- **Data Ingestion**
- Data Storage
- Data Governance
- Security
- Data Quality
- Data Discovery
- Data Auditing
- Data Lineage
- Data Exploration

Data ingestion allows connectors to get data from different data sources and load the data into the Data Lake. Data ingestion supports:

- All types of structured, semi-structured, and unstructured data
- Multiple ingestions such as batch, real-time, and one-time load
- Many types of data sources such as databases, webservers, emails, IoT, and FTP



Data Lake Key Concepts

- Data Ingestion
- Data Storage
- Data Governance
- Security
- Data Quality
- Data Discovery
- Data Auditing
- Data Lineage
- Data Exploration

- Data storage is scalable and offers cost-effective storage
- It allows faster access to data exploration
- It supports various data formats

Data Lake Key Concepts

- Data Ingestion
- Data Storage
- **Data Governance**
- Security
- Data Quality
- Data Discovery
- Data Auditing
- Data Lineage
- Data Exploration

★ Data governance is a process of managing the availability, usability, security, and integrity of the data used in an organization

Data Lake Key Concepts

- Data Ingestion
- Data Storage
- Data Governance
- **Security**
- Data Quality
- Data Discovery
- Data Auditing
- Data Lineage
- Data Exploration

- Security is implemented in every layer of the Data Lake
- It starts with storage, unearthing, and consumption
- The basic need is to block access for unauthorized users
- It supports different tools to access data easier and to navigate GUI and dashboards
- Authentication, accounting, authorization, and data protection are some important features of Data Lake security

Data Lake Key Concepts

- Data Ingestion
- Data Storage
- Data Governance
- Security
- **Data Quality**
- Data Discovery
- Data Auditing
- Data Lineage
- Data Exploration

- Data quality is an essential component of the Data Lake architecture
- The data is used to extract business value
- Extracting insights from poor-quality data will lead to poor-quality insights

Data Lake Key Concepts

- Data Ingestion
- Data Storage
- Data Governance
- Security
- Data Quality
- **Data Discovery**
- Data Auditing
- Data Lineage
- Data Exploration

- Data discovery is another important stage before we can begin with preparing data for analysis
- In this stage, the tagging technique is used to express data understanding, i.e., organizing and interpreting the data ingested in the Data Lake

Data Lake Key Concepts

- Data Ingestion
- Data Storage
- Data Governance
- Security
- Data Quality
- Data Discovery
- **Data Auditing**
- Data Lineage
- Data Exploration

- Data comes from various sources, various departments, and various asset classifications (secret, public, etc.)
- Because of these variations, some data needs special security requirements and handling
- Certain data in the Data Lake needs the tracking of changes, it undergoes, and of who accesses it for various legal and contractual aspects

Data Lake Key Concepts

- Data Ingestion
- Data Storage
- Data Governance
- Security
- Data Quality
- Data Discovery
- Data Auditing
- **Data Lineage**
- Data Exploration

- Data lineage deals with the origin of data, what happens to it, and where it moves over time
- It simplifies tracing errors in the data back to data sources in a data analytics process, from the origin to the destination, visualized through appropriate tools

Data Lake Key Concepts

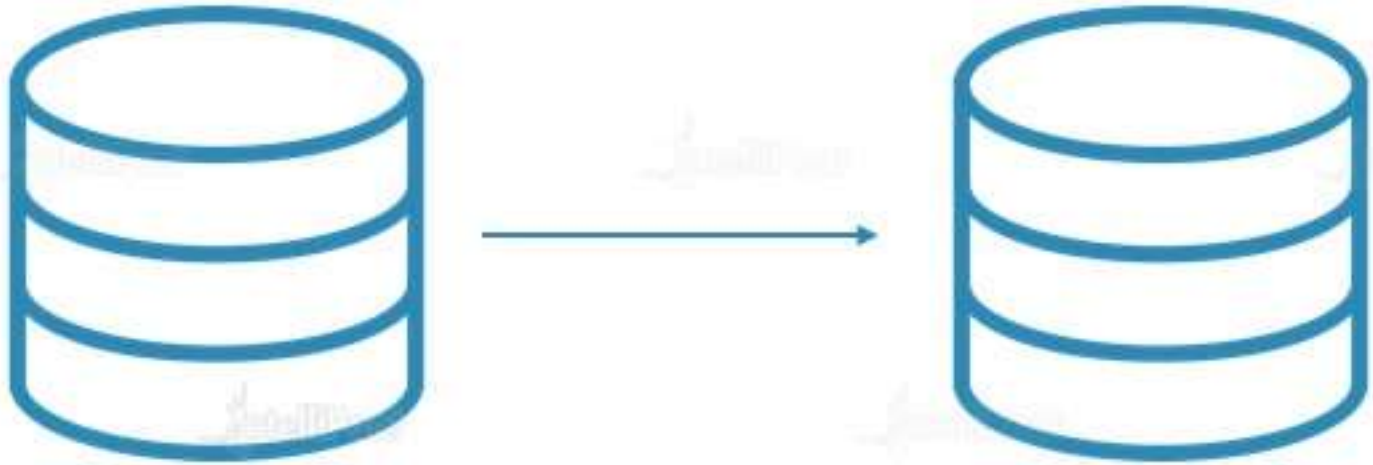
- Data Ingestion
- Data Storage
- Data Governance
- Security
- Data Quality
- Data Discovery
- Data Auditing
- Data Lineage
- **Data Exploration**

- Data exploration is the beginning stage of data analysis
- This process enables deeper data analysis as patterns and trends are identified.

Hands-on: Loading Data into Azure Data Lake Storage Gen2 with Azure Data Factory

Azure Cosmos DB

Azure Cosmos DB



Azure Cosmos DB is an improvised version of the Document DB

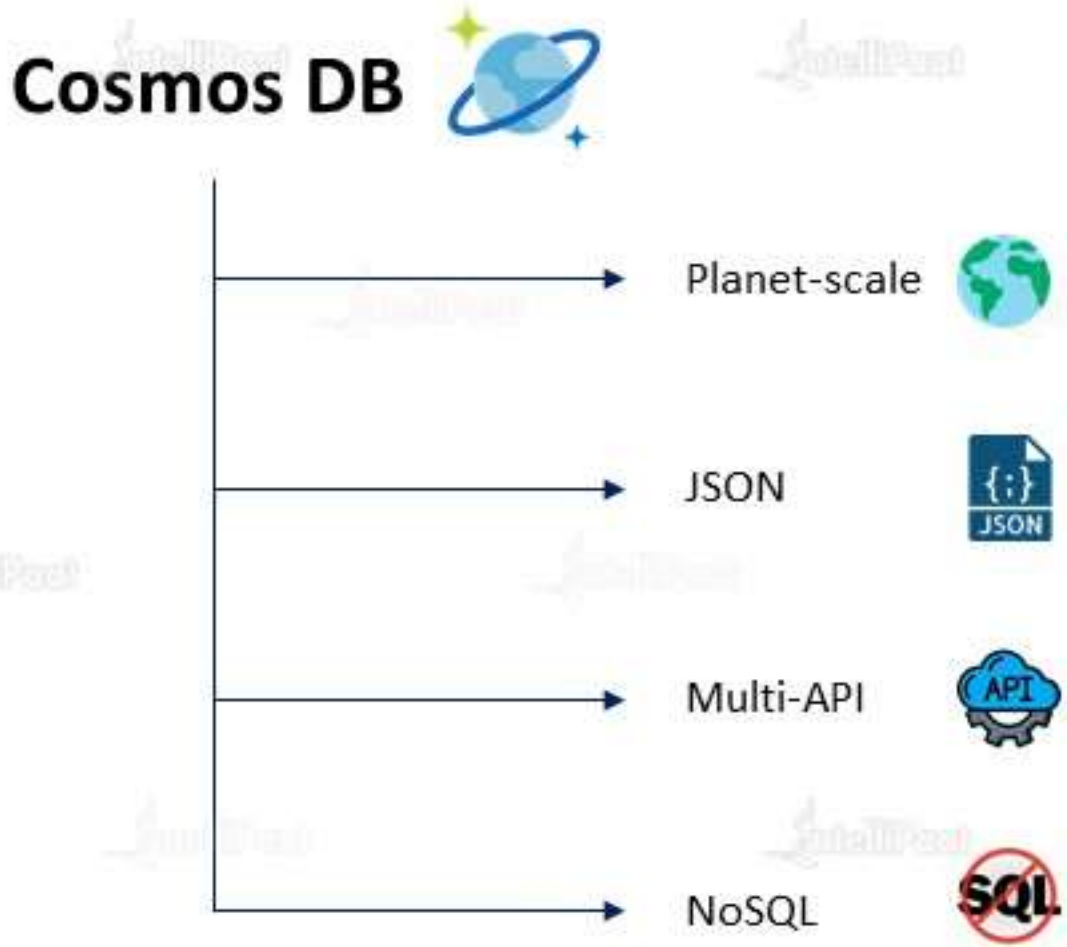
Azure Cosmos DB

Today's applications are required to be highly responsive and always online

To achieve low latency and high availability, instances of these applications need to be deployed in data centers close to their users

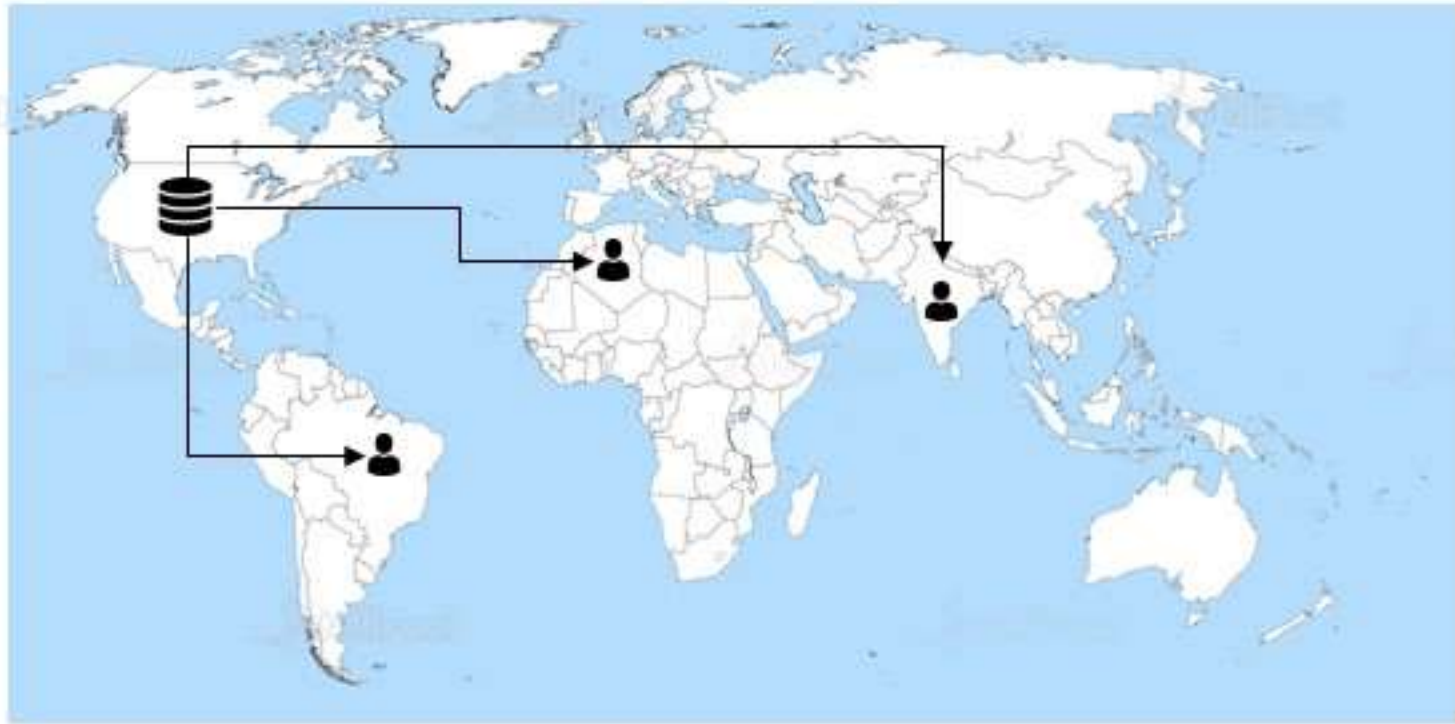
- Azure Cosmos DB is Microsoft's globally distributed, multi-model database service
- With a single click, Cosmos DB enables us to elastically and independently scale throughput and storage across any number of Azure regions worldwide

Azure Cosmos DB



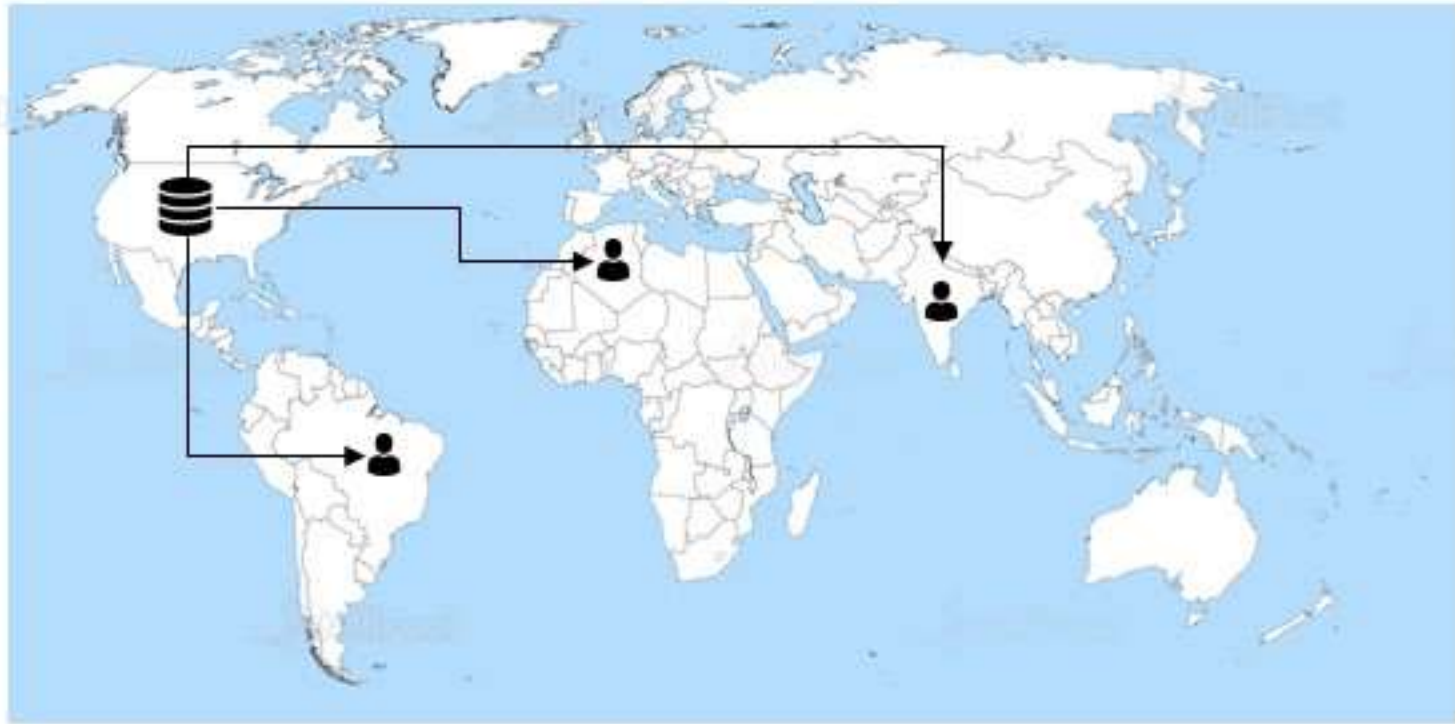
Cosmos DB is a planet-scale, NoSQL, and JSON database with multi-API support

Planet-scale Database



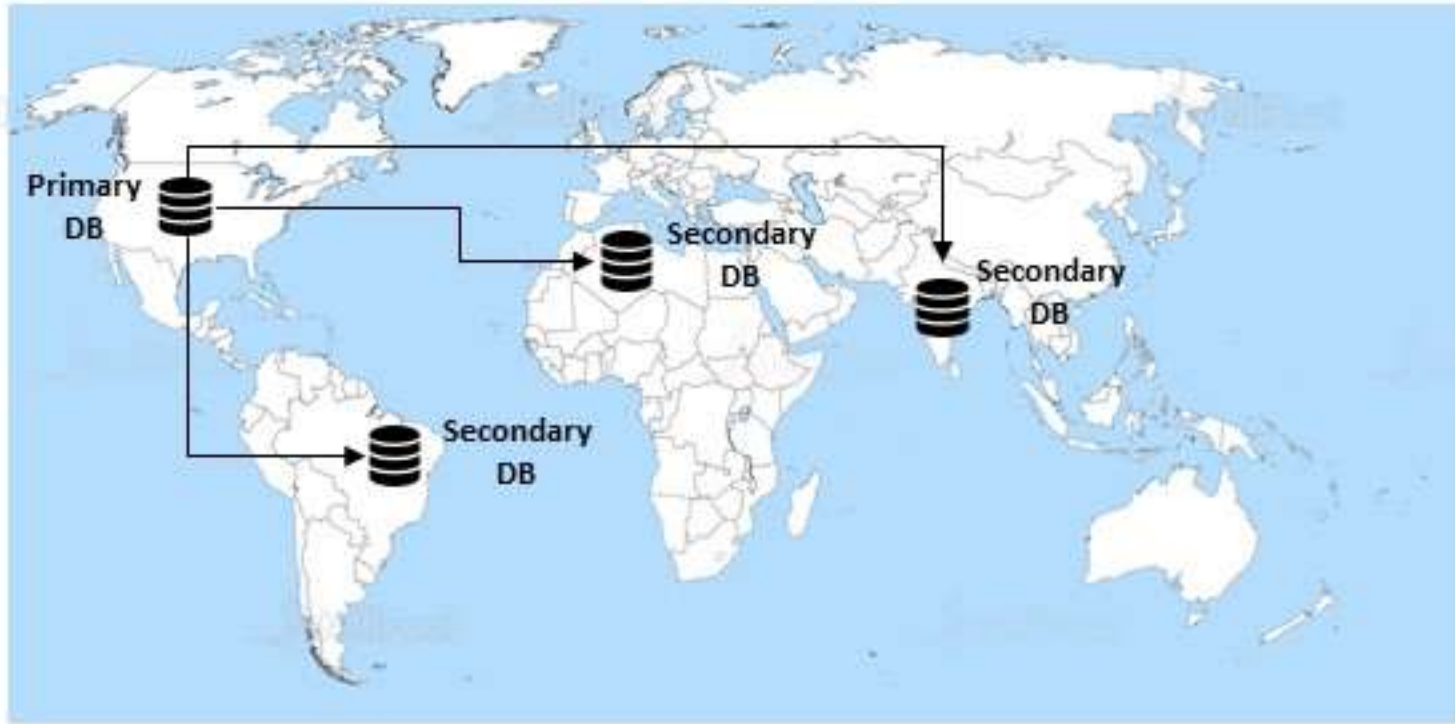
- Suppose, we have a database in the US and there are users who are using it across the globe
- Any user who is in the US can access the database faster as compared to the users who reside in the other geographical areas

Planet-scale Database



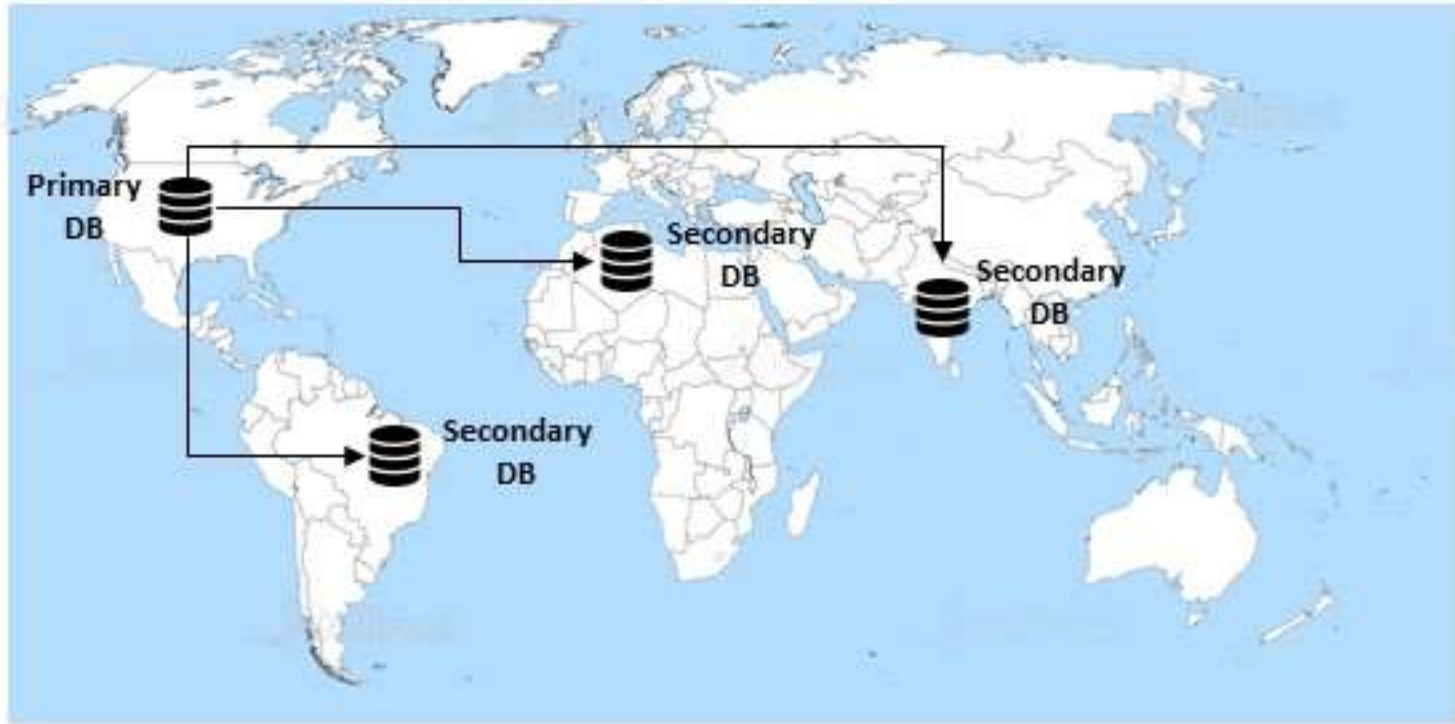
- So, the performance for the users, of other than the US location, is not up to the mark

Planet-scale Database



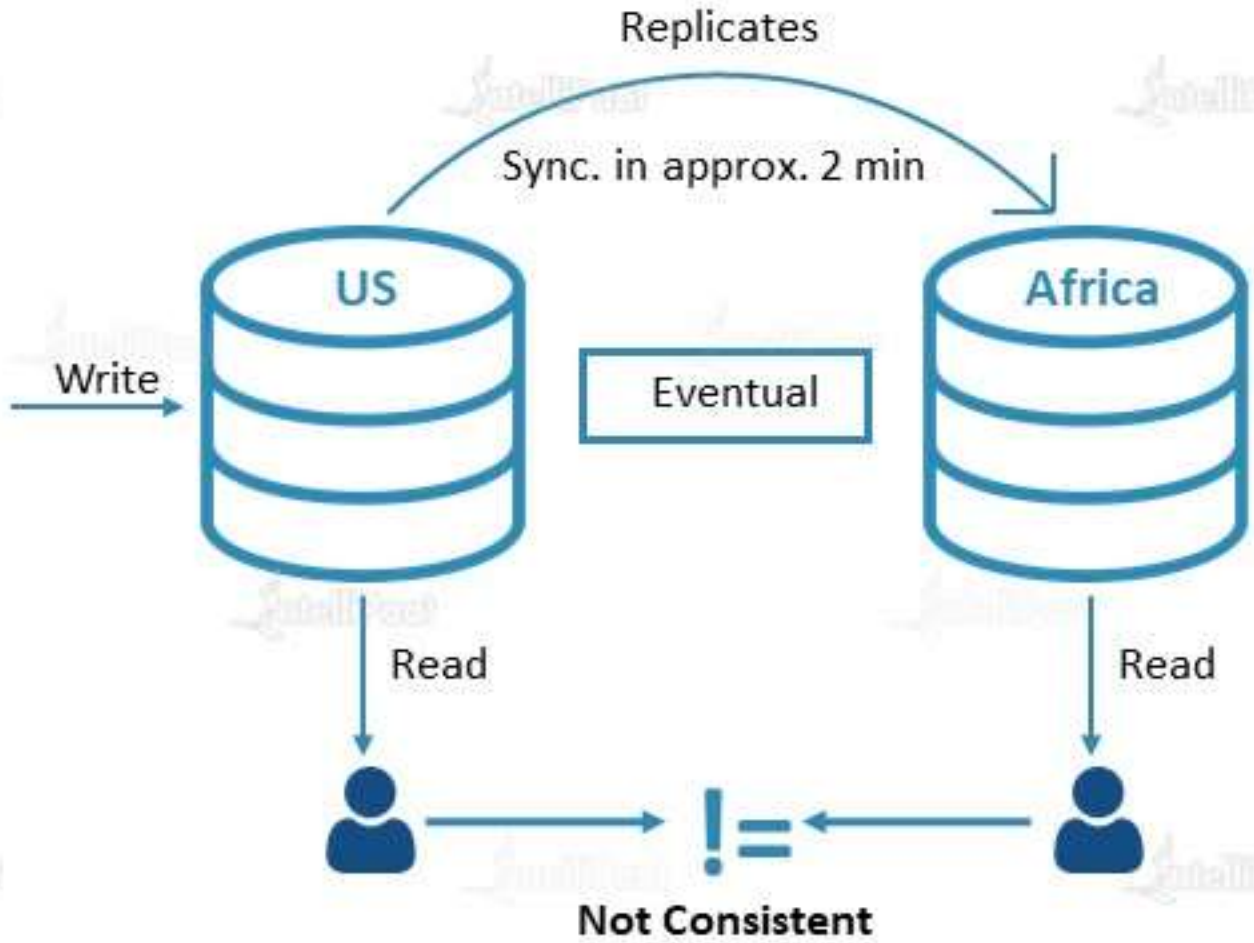
- Here, we can set up secondary databases (read databases), at whichever places we have users
- Meanwhile, the primary database (write database) is in the US

Planet-scale Database



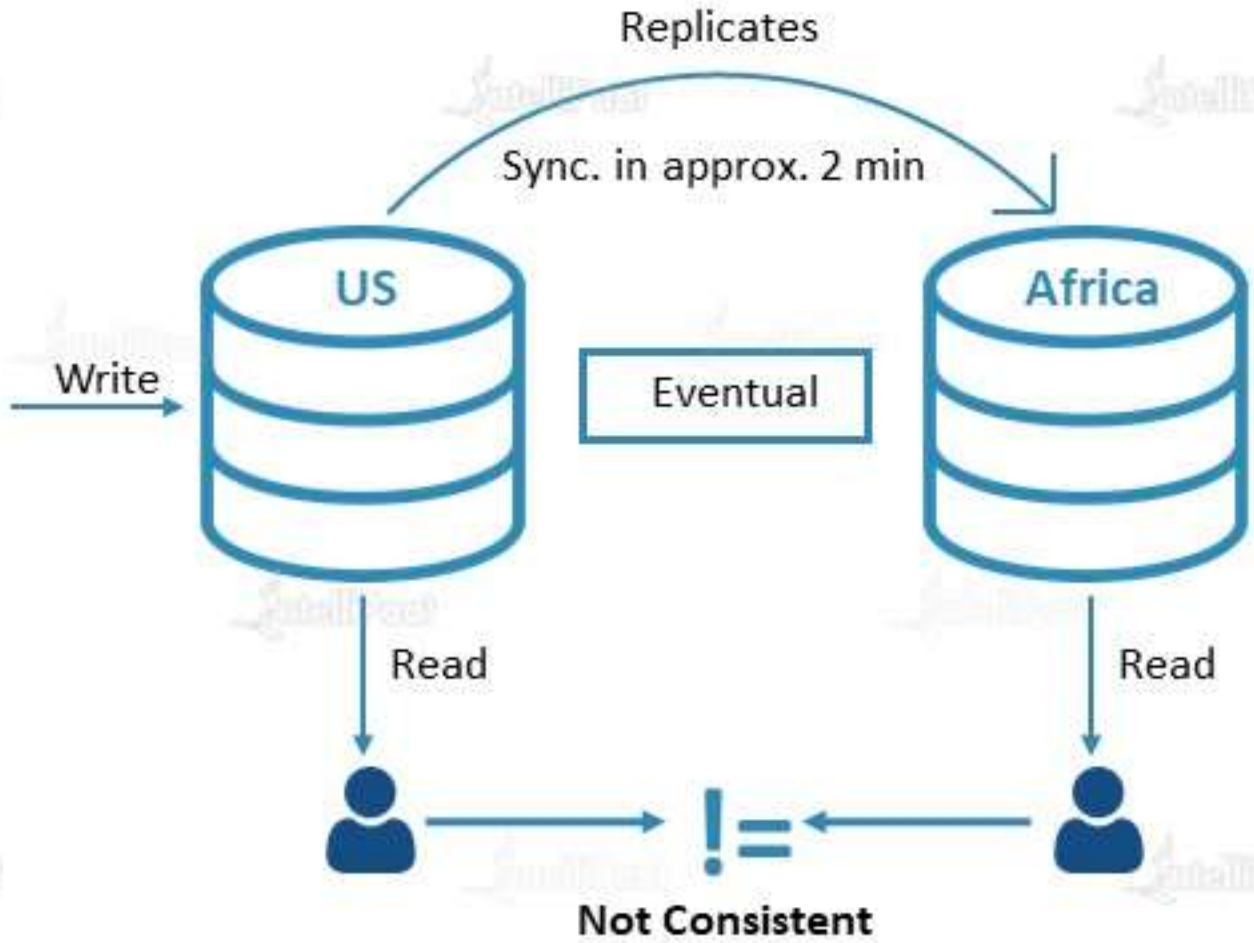
- When someone writes in the US database, the replications of it are sent to the secondary databases that are placed at the other geographical areas

Planet-scale Database



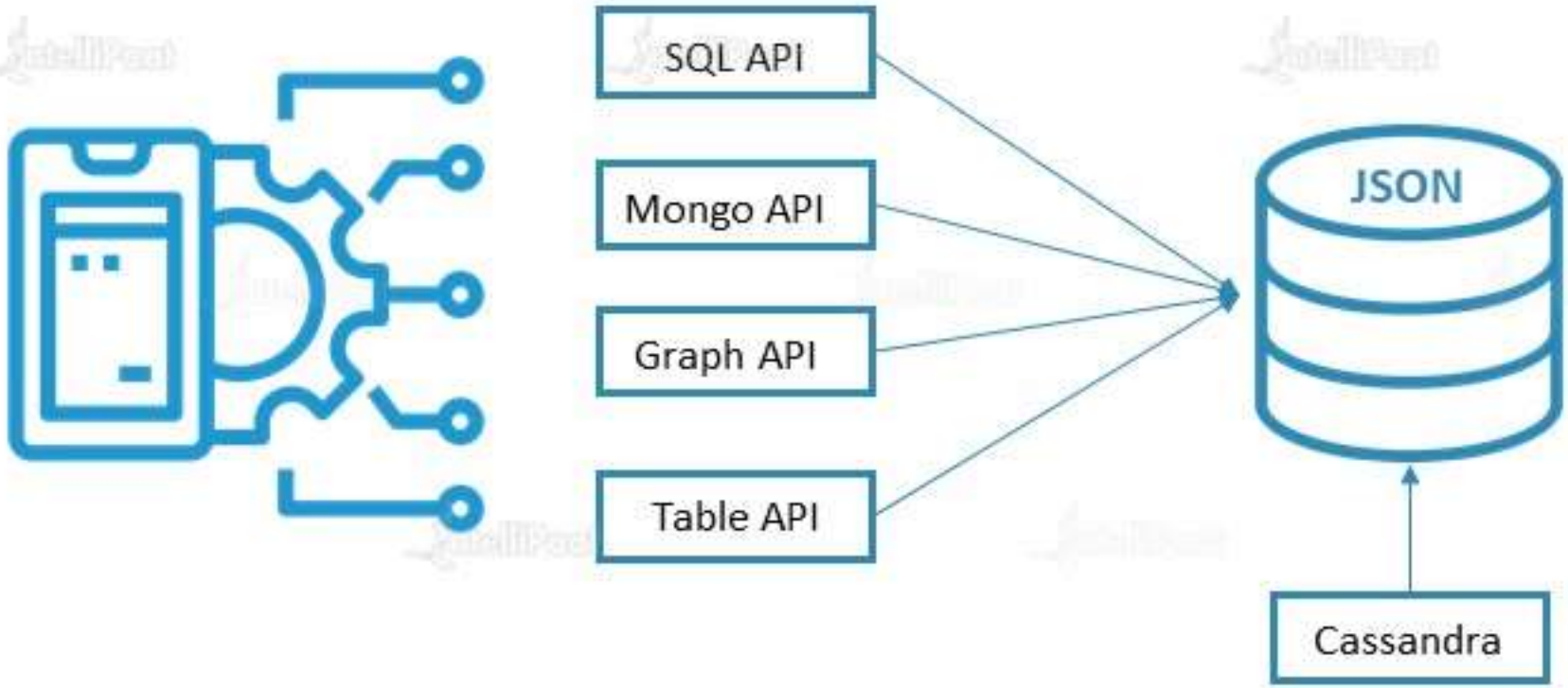
★ One of the biggest problems to achieve a planet-scale database architecture is **'inconsistency'**

Planet-scale Database



★ Before synchronization finishes, if an end user reads data from the US DB, he/she would see a different data as compared to the user who reads from the Africa DB

Multi-API Support



- It is a planet-scale database that stores data in a JSON format but with multi-API support
- Multi-API support means, the data is stored in a JSON format, but we can use any API format to retrieve data.

Why Azure Cosmos DB?

Why Azure Cosmos DB?

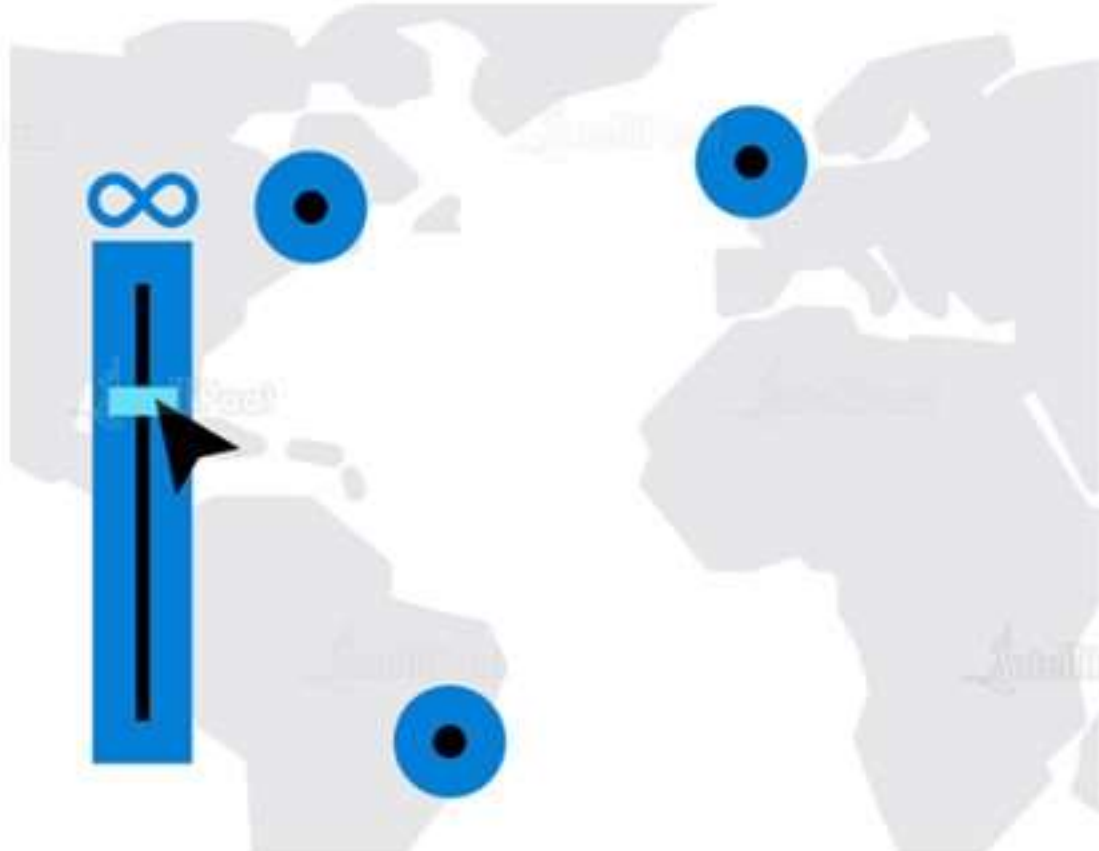
Turnkey global distribution with transparent multi-master replication and five well-defined consistency choices

Why Azure Cosmos DB?

Easily adds or removes any number of regions worldwide at any time

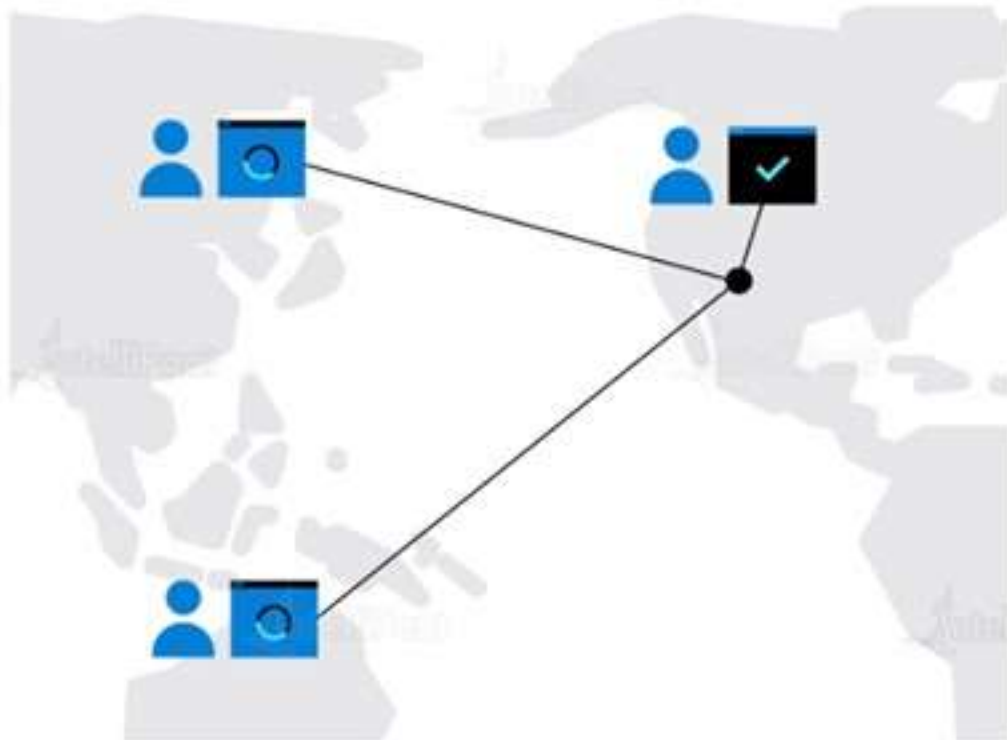
Why Azure Cosmos DB?

Elastically scales throughput and storage across all regions at any time



Why Azure Cosmos DB?

Avoids the speed-of-light penalty for our globally distributed apps



Why Azure Cosmos DB?

- Serves reads and writes locally for guaranteed low latency and 99.999% high availability around the world
- Avoids extreme trade-offs in consistency, performance, and availability with its five consistency models

Why Azure Cosmos DB?

A multi-model with wire-protocol-compatible API endpoints for Cassandra, MongoDB, SQL, Gremlin, etc. and a table with built-in support for Apache Spark and Jupyter Notebook

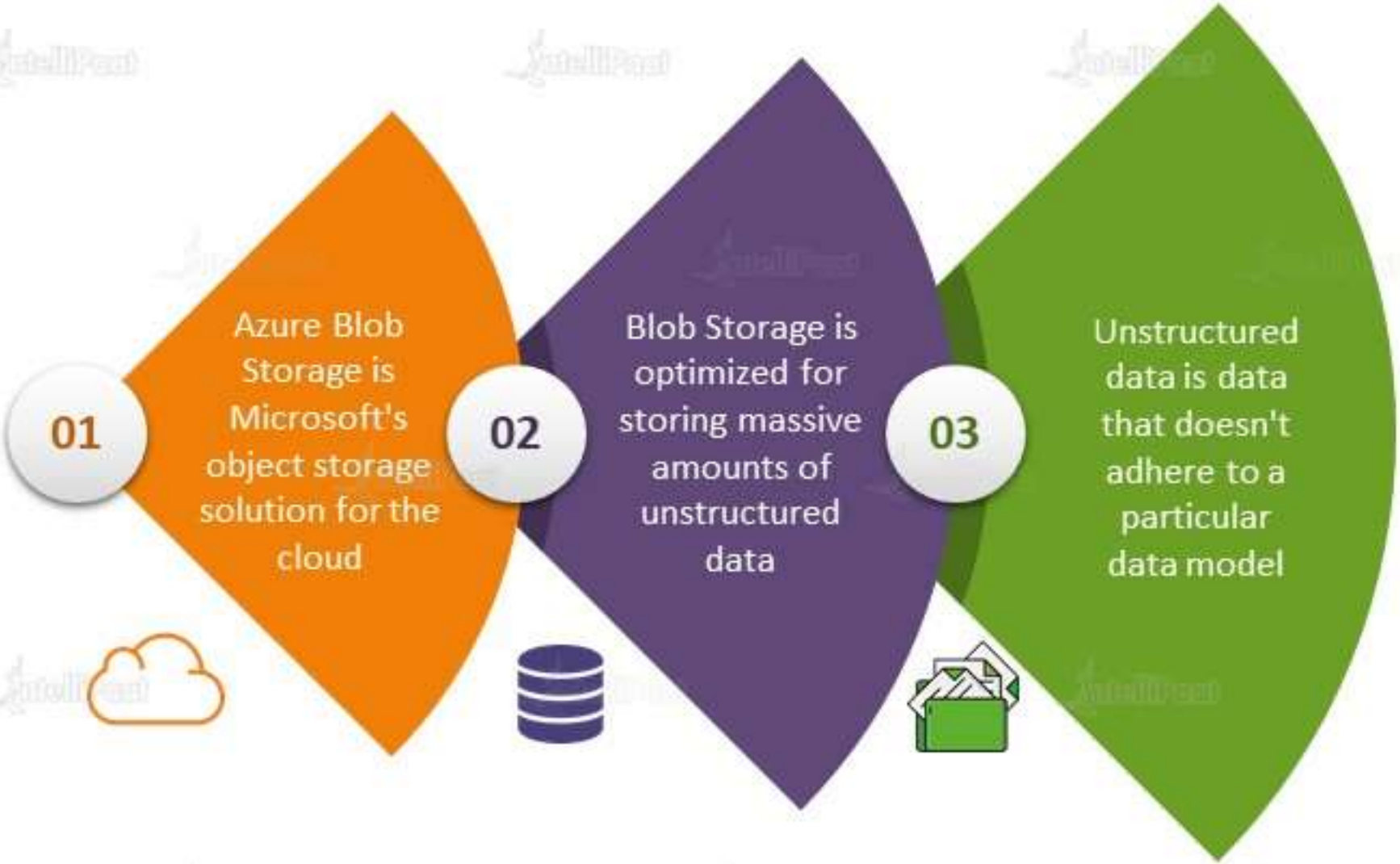
Why Azure Cosmos DB?

Enterprise-grade security by default

Hands-on: Working with Azure Cosmos DB

Azure Blob Storage

Azure Blob Storage



Azure Blob Storage

Blob Storage is designed for:

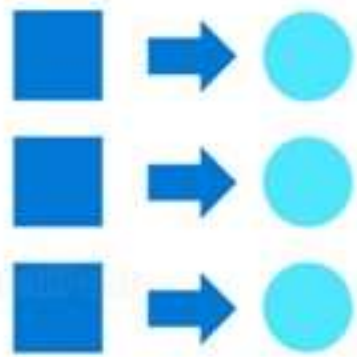
- Serving images or documents directly to a browser
- Storing files for distributed access
- Streaming video and audio
- Writing to log files
- Storing data for backup and restore, disaster recovery, and archiving
- Storing data for analysis by an on-premises or Azure-hosted service

Why Azure Blob Storage?

Why Azure Blob Storage?

Strong Consistency

When an object is changed, it is verified everywhere for superior data integrity, ensuring that we always have access to the latest version



Why Azure Blob Storage?

Object Mutability

Get the flexibility to perform edits in place, which can improve our application performance and reduce bandwidth consumption

Why Azure Blob Storage?

Multiple Blob Types

Block, page, and append blobs give us maximum flexibility to optimize our storage as per our needs

Why Azure Blob Storage?

Easy-to-use Geo-redundancy

It automatically configures geo-replication options in a single menu to easily empower enhanced global and local access and business continuity

Why Azure Blob Storage?

One Infrastructure; Worldwide Access

- It makes our unstructured data available to customers anywhere through the REST-based object storage
- With regions around the world, it is ideal for streaming and storing media, be it a live broadcast event or a long-term archive of petabytes of movies and television shows
- It searches for hidden insights in massive object storage, through Big Data Analytics, and gives access to the right people at the right time, no matter where they are

Why Azure Blob Storage?

Any Data; Cloud Scale

- It stores any type of unstructured data—including images, videos, audio, documents, and backups—with a proven technology at an exabyte scale
- Blob Storage handles trillions of stored objects, with millions of average requests per second, for customers around the world
- Blob storage also supports Azure Data Share, a simple and safe service for sharing big data

Data Partitioning

Data Partitioning

Partitioning means the process of physically dividing data into separate data stores

It can improve scalability, reduce contention, and optimize performance

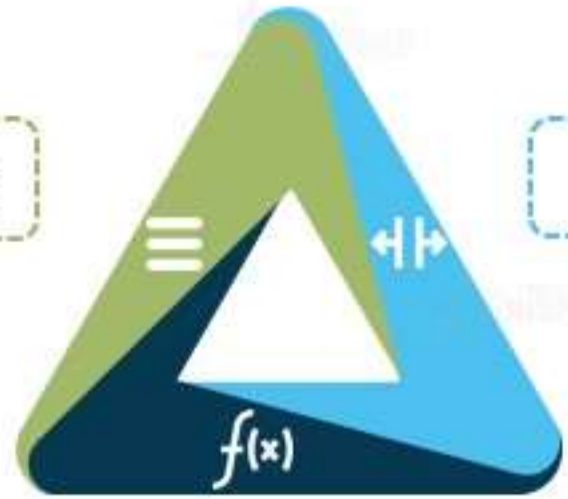
Partitioning can also provide a mechanism for dividing data by the usage pattern

For example, we can archive the older data in a cheaper data storage

Data Partitioning

There are three typical strategies for partitioning data:

Horizontal partitioning



Vertical Partitioning

Functional Partitioning

Designing Partitions

- Horizontal Partitioning
- Vertical Partitioning
- Functional Partitioning

- It is often called sharding
- In this strategy, each partition is a separate data store, but all partitions have the same schema
- Each partition is known as a shard and holds a specific subset of the data, e.g., all the orders for a specific set of customers

Designing Partitions

- Horizontal Partitioning
- Vertical Partitioning
- Functional Partitioning

- In this example, the product inventory data is divided into shards based on the product key

Horizontally Partitioning (Sharding) Data Based on a Partition Key

Key	Name	Description	Stock	Price	LastOrdered
ARC1	Arc Welder	250 Amps	8	119.00	25-Nov-2019
BRK8	Bracket	250 mm	46	5.66	18-Nov-2019
BRK9	Bracket	400 mm	82	6.98	1-July-2019
HOS8	Hose	1/2"	27	27.50	18-Aug-2019
WGT4	Widget	Green	16	13.99	3-Feb-2019
WGT6	Widget	Purple	76	13.99	31-Mar-2019

Key	Name	Description	Stock	Price	LastOrdered
ARC1	Arc Welder	250 Amps	8	119.00	25-Nov-2019
BRK8	Bracket	250 mm	46	5.66	18-Nov-2019
BRK9	Bracket	400 mm	82	6.98	1-July-2019

Key	Name	Description	Stock	Price	LastOrdered
HOS8	Hose	1/2"	27	27.50	18-Aug-2019
WGT4	Widget	Green	16	13.99	3-Feb-2019
WGT6	Widget	Purple	76	13.99	31-Mar-2019

Designing Partitions

- Horizontal Partitioning
- Vertical Partitioning
- Functional Partitioning

- Each shard holds data for a contiguous range of shard keys (A–G and H–Z), organized alphabetically

Horizontally Partitioning (Sharding) Data Based on a Partition Key

Key	Name	Description	Stock	Price	LastOrdered
ARC1	Arc Welder	250 Amps	8	119.00	25-Nov-2019
BRK8	Bracket	250 mm	46	5.66	18-Nov-2019
BRK9	Bracket	400 mm	82	6.98	1-July-2019
HOS8	Hose	1/2"	27	27.50	18-Aug-2019
WGT4	Widget	Green	16	13.99	3-Feb-2019
WGT6	Widget	Purple	76	13.99	31-Mar-2019

Key	Name	Description	Stock	Price	LastOrdered
ARC1	Arc Welder	250 Amps	8	119.00	25-Nov-2019
BRK8	Bracket	250 mm	46	5.66	18-Nov-2019
BRK9	Bracket	400 mm	82	6.98	1-July-2019

Key	Name	Description	Stock	Price	LastOrdered
HOS8	Hose	1/2"	27	27.50	18-Aug-2019
WGT4	Widget	Green	16	13.99	3-Feb-2019
WGT6	Widget	Purple	76	13.99	31-Mar-2019

Designing Partitions

- Horizontal Partitioning
- Vertical Partitioning
- Functional Partitioning

- Sharding spreads the load over many computers, which reduces contention and improves performance

Horizontally Partitioning (Sharding) Data Based on a Partition Key

Key	Name	Description	Stock	Price	LastOrdered
ARC1	Arc Welder	250 Amps	8	119.00	25-Nov-2019
BRK8	Bracket	250 mm	46	5.66	18-Nov-2019
BRK9	Bracket	400 mm	82	6.98	1-July-2019
HOS8	Hose	1/2"	27	27.50	18-Aug-2019
WGT4	Widget	Green	16	13.99	3-Feb-2019
WGT6	Widget	Purple	76	13.99	31-Mar-2019

Key	Name	Description	Stock	Price	LastOrdered
ARC1	Arc Welder	250 Amps	8	119.00	25-Nov-2019
BRK8	Bracket	250 mm	46	5.66	18-Nov-2019
BRK9	Bracket	400 mm	82	6.98	1-July-2019

Key	Name	Description	Stock	Price	LastOrdered
HOS8	Hose	1/2"	27	27.50	18-Aug-2019
WGT4	Widget	Green	16	13.99	3-Feb-2019
WGT6	Widget	Purple	76	13.99	31-Mar-2019

Designing Partitions

- Horizontal Partitioning
- **Vertical Partitioning**
- Functional Partitioning

- In this strategy, each partition holds a subset of the fields for items in the data store
- The fields are divided according to their pattern of use
- For example, frequently accessed fields might be placed in one vertical partition and less-frequently accessed fields in another

Designing Partitions

- Horizontal Partitioning
- Vertical Partitioning
- Functional Partitioning

- In this example, different properties of an item are stored in different partitions

Vertically Partitioning Data by Its Pattern of Use

Key	Name	Description	Stock	Price	LastOrdered
ARC1	Arc Welder	250 Amps	8	119.00	25-Nov-2019
BRK8	Bracket	250 mm	46	5.66	18-Nov-2019
BRK9	Bracket	400 mm	82	6.98	1-July-2019
HOS8	Hose	1/2"	27	27.50	18-Aug-2019
WGT4	Widget	Green	16	13.99	3-Feb-2019
WGT6	Widget	Purple	76	13.99	31-Mar-2019

Key	Name	Description	Price
ARC1	Arc Welder	250 Amps	119.00
BRK8	Bracket	250 mm	5.66
BRK9	Bracket	400 mm	6.98
HOS8	Hose	1/2"	27.50
WGT4	Widget	Green	13.99
WGT6	Widget	Purple	13.99

Key 0	Stock	LastOrdered
ARC1	8	25-Nov-2019
BRK8	46	18-Nov-2019
BRK9	82	1-July-2019
HOS8	27	18-Aug-2019
WGT4	16	3-Feb-2019
WGT6	76	31-Mar-2019

Designing Partitions

- Horizontal Partitioning
- Vertical Partitioning
- Functional Partitioning

- The partition on the left holds data that is accessed more frequently, including product name, description, and price

Vertically Partitioning Data by Its Pattern of Use

Key	Name	Description	Stock	Price	LastOrdered
ARC1	Arc Welder	250 Amps	8	119.00	25-Nov-2019
BRK8	Bracket	250 mm	46	5.66	18-Nov-2019
BRK9	Bracket	400 mm	82	6.98	1-July-2019
HOS8	Hose	1/2"	27	27.50	18-Aug-2019
WGT4	Widget	Green	16	13.99	3-Feb-2019
WGT6	Widget	Purple	76	13.99	31-Mar-2019

Key	Name	Description	Price
ARC1	Arc Welder	250 Amps	119.00
BRK8	Bracket	250 mm	5.66
BRK9	Bracket	400 mm	6.98
HOS8	Hose	1/2"	27.50
WGT4	Widget	Green	13.99
WGT6	Widget	Purple	13.99

Key 0	Stock	LastOrdered
ARC1	8	25-Nov-2019
BRK8	46	18-Nov-2019
BRK9	82	1-July-2019
HOS8	27	18-Aug-2019
WGT4	16	3-Feb-2019
WGT6	76	31-Mar-2019

Designing Partitions

- Horizontal Partitioning
- Vertical Partitioning
- Functional Partitioning

- The partition on the right holds inventory data: the stock count and last-ordered date

Vertically Partitioning Data by Its Pattern of Use

Key	Name	Description	Stock	Price	LastOrdered
ARC1	Arc Welder	250 Amps	8	119.00	25-Nov-2019
BRK8	Bracket	250 mm	46	5.66	18-Nov-2019
BRK9	Bracket	400 mm	82	6.98	1-July-2019
HOS8	Hose	1/2"	27	27.50	18-Aug-2019
WGT4	Widget	Green	16	13.99	3-Feb-2019
WGT6	Widget	Purple	76	13.99	31-Mar-2019

Key	Name	Description	Price
ARC1	Arc Welder	250 Amps	119.00
BRK8	Bracket	250 mm	5.66
BRK9	Bracket	400 mm	6.98
HOS8	Hose	1/2"	27.50
WGT4	Widget	Green	13.99
WGT6	Widget	Purple	13.99

Key 0	Stock	LastOrdered
ARC1	8	25-Nov-2019
BRK8	46	18-Nov-2019
BRK9	82	1-July-2019
HOS8	27	18-Aug-2019
WGT4	16	3-Feb-2019
WGT6	76	31-Mar-2019

Designing Partitions

- Horizontal Partitioning
- Vertical Partitioning
- Functional Partitioning

- In this example, the application regularly queries the product name, description, and price when displaying the product details to customers

Vertically Partitioning Data by Its Pattern of Use

Key	Name	Description	Stock	Price	LastOrdered
ARC1	Arc Welder	250 Amps	8	119.00	25-Nov-2019
BRK8	Bracket	250 mm	46	5.66	18-Nov-2019
BRK9	Bracket	400 mm	82	6.98	1-July-2019
HOS8	Hose	1/2"	27	27.50	18-Aug-2019
WGT4	Widget	Green	16	13.99	3-Feb-2019
WGT6	Widget	Purple	76	13.99	31-Mar-2019

Key	Name	Description	Price
ARC1	Arc Welder	250 Amps	119.00
BRK8	Bracket	250 mm	5.66
BRK9	Bracket	400 mm	6.98
HOS8	Hose	1/2"	27.50
WGT4	Widget	Green	13.99
WGT6	Widget	Purple	13.99

Key 0	Stock	LastOrdered
ARC1	8	25-Nov-2019
BRK8	46	18-Nov-2019
BRK9	82	1-July-2019
HOS8	27	18-Aug-2019
WGT4	16	3-Feb-2019
WGT6	76	31-Mar-2019

Designing Partitions

- Horizontal Partitioning
- Vertical Partitioning
- Functional Partitioning

- The stock count and last-ordered date are held in a separate partition because these two items are commonly used together

Vertically Partitioning Data by Its Pattern of Use

Key	Name	Description	Stock	Price	LastOrdered
ARC1	Arc Welder	250 Amps	8	119.00	25-Nov-2019
BRK8	Bracket	250 mm	46	5.66	18-Nov-2019
BRK9	Bracket	400 mm	82	6.98	1-July-2019
HOS8	Hose	1/2"	27	27.50	18-Aug-2019
WGT4	Widget	Green	16	13.99	3-Feb-2019
WGT6	Widget	Purple	76	13.99	31-Mar-2019

Key	Name	Description	Price
ARC1	Arc Welder	250 Amps	119.00
BRK8	Bracket	250 mm	5.66
BRK9	Bracket	400 mm	6.98
HOS8	Hose	1/2"	27.50
WGT4	Widget	Green	13.99
WGT6	Widget	Purple	13.99

Key 0	Stock	LastOrdered
ARC1	8	25-Nov-2019
BRK8	46	18-Nov-2019
BRK9	82	1-July-2019
HOS8	27	18-Aug-2019
WGT4	16	3-Feb-2019
WGT6	76	31-Mar-2019

Designing Partitions

- Horizontal Partitioning
- Vertical Partitioning
- **Functional Partitioning**

- In functional partitioning, data is aggregated according to how it is used by each bounded context in the system
- For example, an e-commerce system might store invoice data in one partition and product inventory data in another

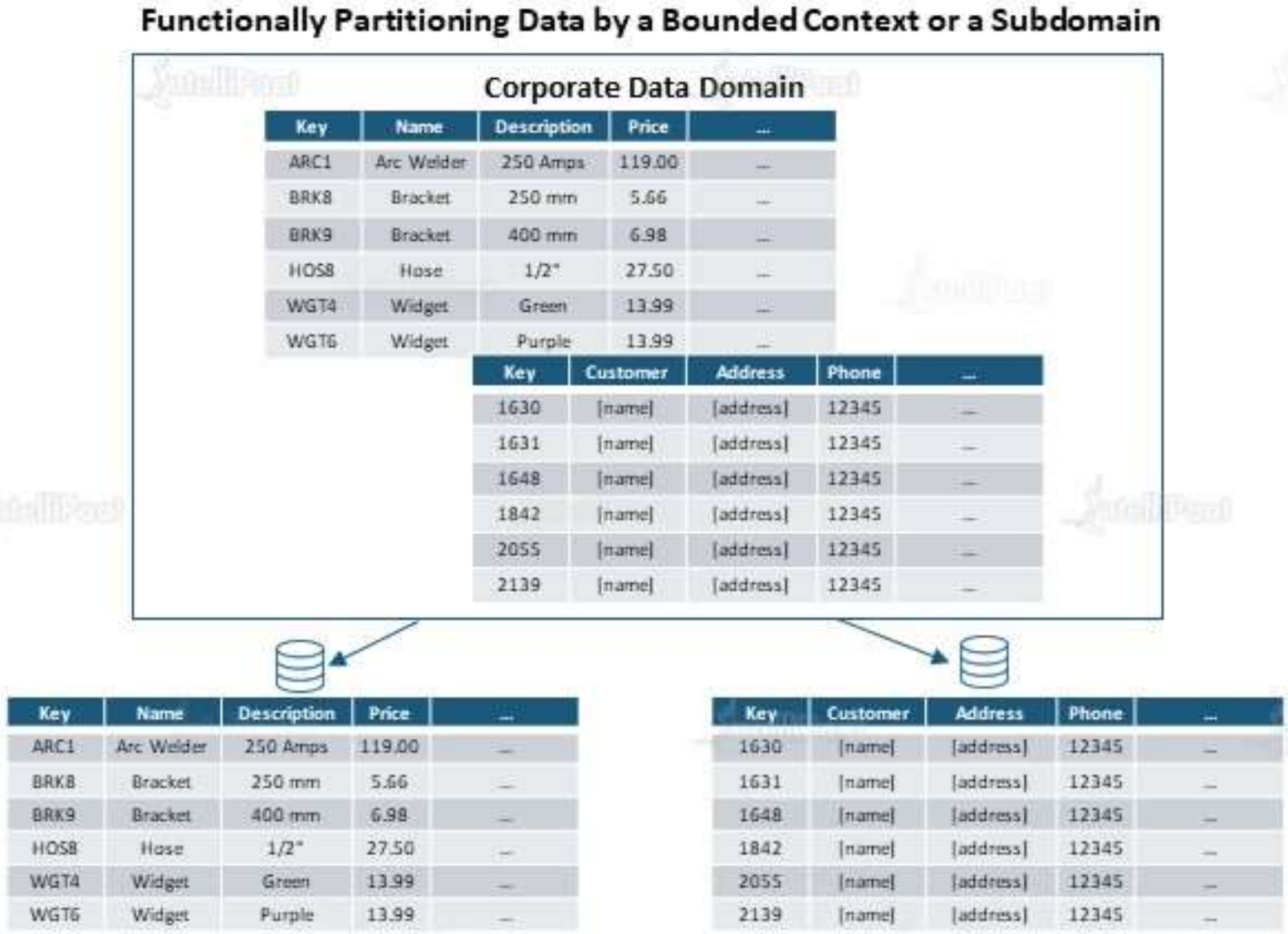
Designing Partitions

- Horizontal Partitioning
- Vertical Partitioning
- **Functional Partitioning**

- When it's possible to identify the bounded context for each distinct business area in an application, functional partitioning is a way to improve isolation and data access performance
- Another common use of functional partitioning is to separate the read-write data from the read-only data

Designing Partitions

- Horizontal Partitioning
- Vertical Partitioning
- **Functional Partitioning**



Designing Partitions

- Horizontal Partitioning
- Vertical Partitioning
- **Functional Partitioning**

- The previous diagram shows the overview of functional partitioning where the inventory data is separated from the customer data
- This partitioning strategy can help reduce data access contention across different parts of the system

Why Partition Data?

Why Partition Data?

Improves Scalability

- When we scale up a single database system, it will eventually reach a physical hardware limit
- If we divide data across multiple partitions, each hosted on a separate server, we can scale up the system almost indefinitely

Why Partition Data?

Improves Performance

- Data access operations on each partition take place over a smaller volume of data
- Partitioning can make our system more efficient. Operations that affect more than one partition can run in parallel

Why Partition Data?

Improves Security

- In some cases, we can separate sensitive and non-sensitive data into different partitions and apply different security controls for them

Why Partition Data?

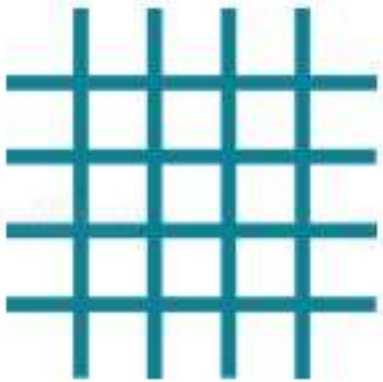
Provides Operational Flexibility

- Partitioning offers many opportunities for fine-tuning operations, maximizing administrative efficiency, and minimizing cost

Why Partition Data?

Matches the Data Store to the Pattern of Use

- Partitioning allows each partition to be deployed on a different type of data store, based on cost and the built-in features that the data store offers
- **E.g.:** Large binary data can be stored in a Blob Storage, while structured data can be held in a Document DB



Why Partition Data?

Improves Availability

- Separating data across multiple servers avoids a single point of failure
- If one instance fails, only the data in that partition is unavailable. Operations on other partitions will continue

Consistency Levels in Azure Cosmos DB

Consistency Levels in Azure Cosmos DB

Most commercially available distributed databases ask developers to choose between two extreme consistency models: **Strong Consistency** and **Eventual Consistency**

The linearizability or the **strong consistency** model is the gold standard of data programmability

Strong consistency model adds a price of higher latency in the steady state and reduced availability during failures. **Eventual consistency** offers higher availability and better performance but makes it hard to program applications

Consistency Levels in Azure Cosmos DB

Azure Cosmos DB approaches **data consistency** as a spectrum of choices instead of two extremes

Strong consistency and **eventual consistency** are at the ends of the spectrum, but there are many consistency choices along the spectrum

Consistency Levels in Azure Cosmos DB

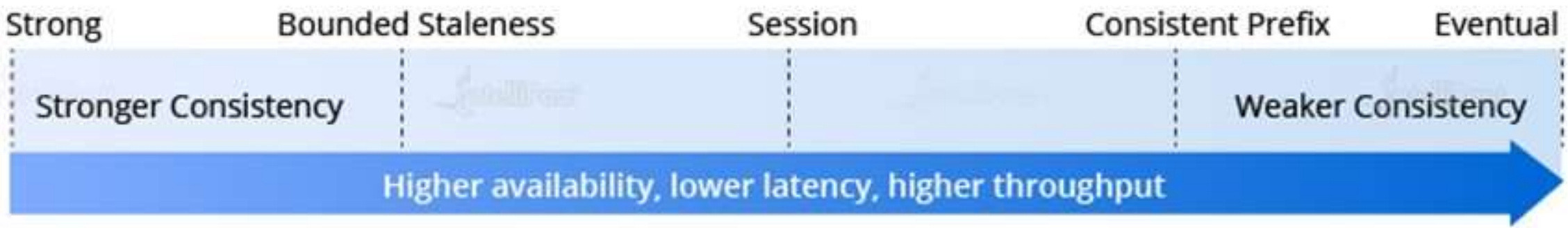
The models are well-defined and intuitive and can be used for specific real-world scenarios

From the strongest to the more relaxed, the models include **strong**, **bounded staleness**, **session**, **consistent prefix**, and **eventual**

Each model provides availability and performance tradeoffs and is backed by SLAs

Consistency Levels in Azure Cosmos DB

The image shows the different consistency levels as a spectrum



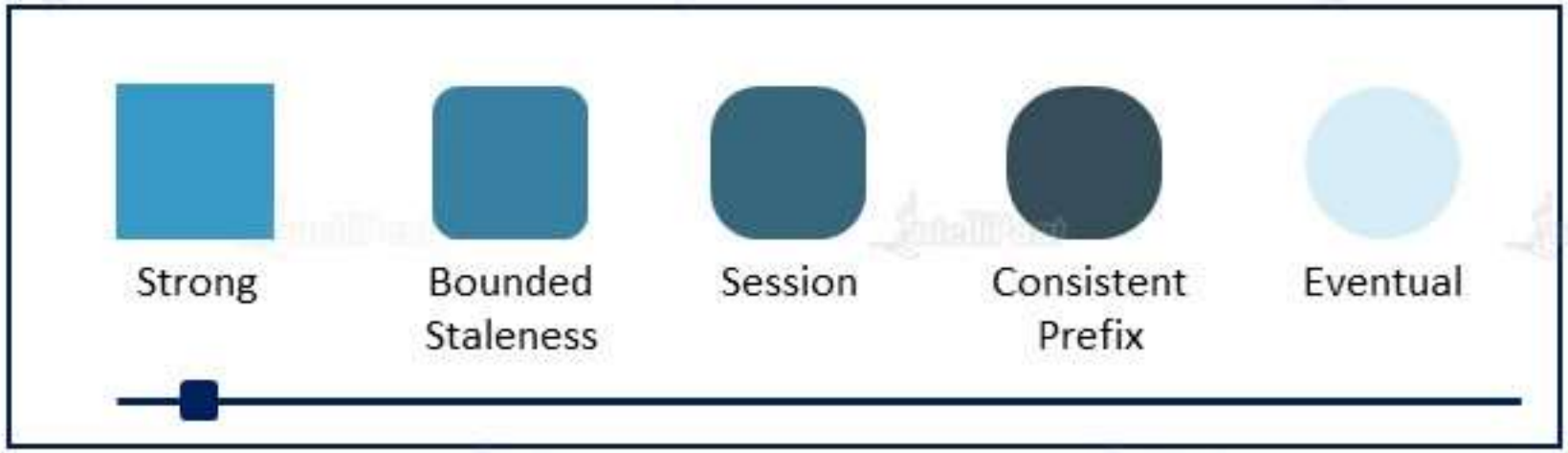
Consistency Levels in Azure Cosmos DB

Semantics of the Five Consistency Levels

Consistency Levels in Azure Cosmos DB

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

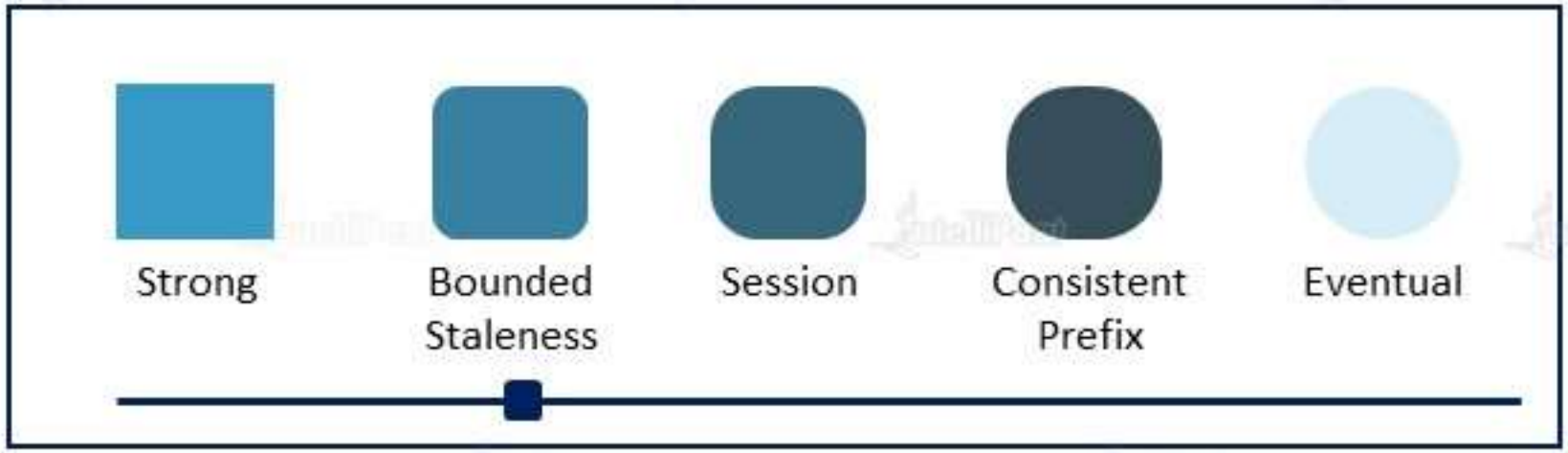
- Strong consistency offers a linearizability guarantee
- Linearizability refers to serving requests concurrently
- Reads are guaranteed to return the most recent committed version of an item
- A client never sees an uncommitted or partial write
- Users are always guaranteed to read the latest committed write



Consistency Levels in Azure Cosmos DB

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

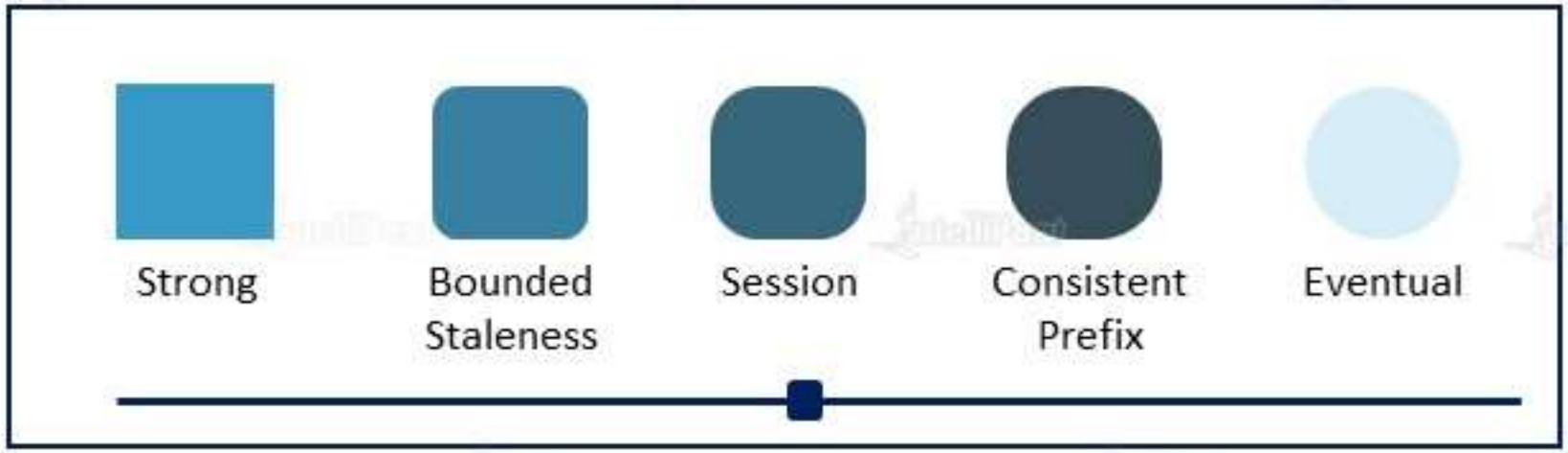
- Reads are guaranteed to honor the consistent-prefix guarantee
- Bounded staleness offers total global order, except within the 'staleness window'
- Monotonic read guarantees exist within a region both inside and outside the staleness window
- When a client performs read operations within a region that accepts writes, the guarantees provided by this consistency are identical to those guarantees by the strong consistency



Consistency Levels in Azure Cosmos DB

- Strong
- Bounded Staleness
- **Session**
- Consistent Prefix
- Eventual

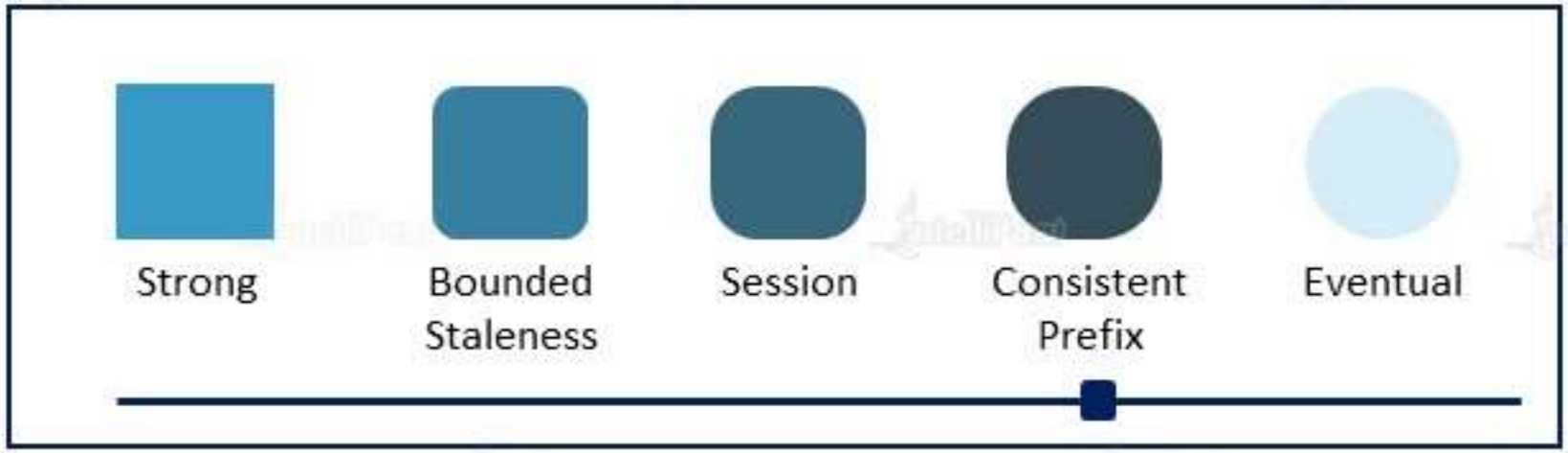
- Within a single client, session reads are guaranteed to honor the consistent-prefix, monotonic reads, monotonic writes, read-your-writes, and write-follows-reads guarantees
- Clients outside of the session, performing writes, will see eventual consistency



Consistency Levels in Azure Cosmos DB

- Strong
- Bounded Staleness
- Session
- **Consistent Prefix**
- Eventual

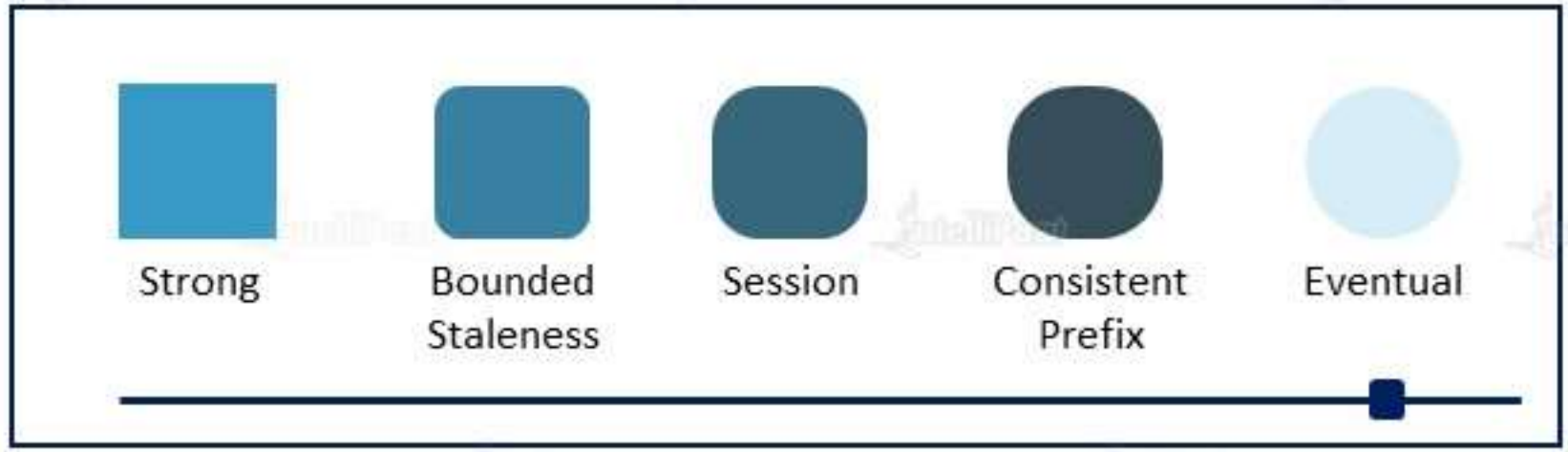
- Updates that are returned contain some prefixes of all updates, with no gaps
- Consistent prefix consistency level guarantees that reads never see out-of-order writes



Consistency Levels in Azure Cosmos DB

- Strong
- Bounded Staleness
- Session
- Consistent Prefix
- **Eventual**

- There's no ordering guarantee for reads
- In the absence of any further writes, the replicas eventually converge



India: +91-7847955955

US: 1-800-216-8930 (TOLL FREE)

sales@intellipaat.com

24/7 Chat with Our Course Advisor