

Cosmos DB

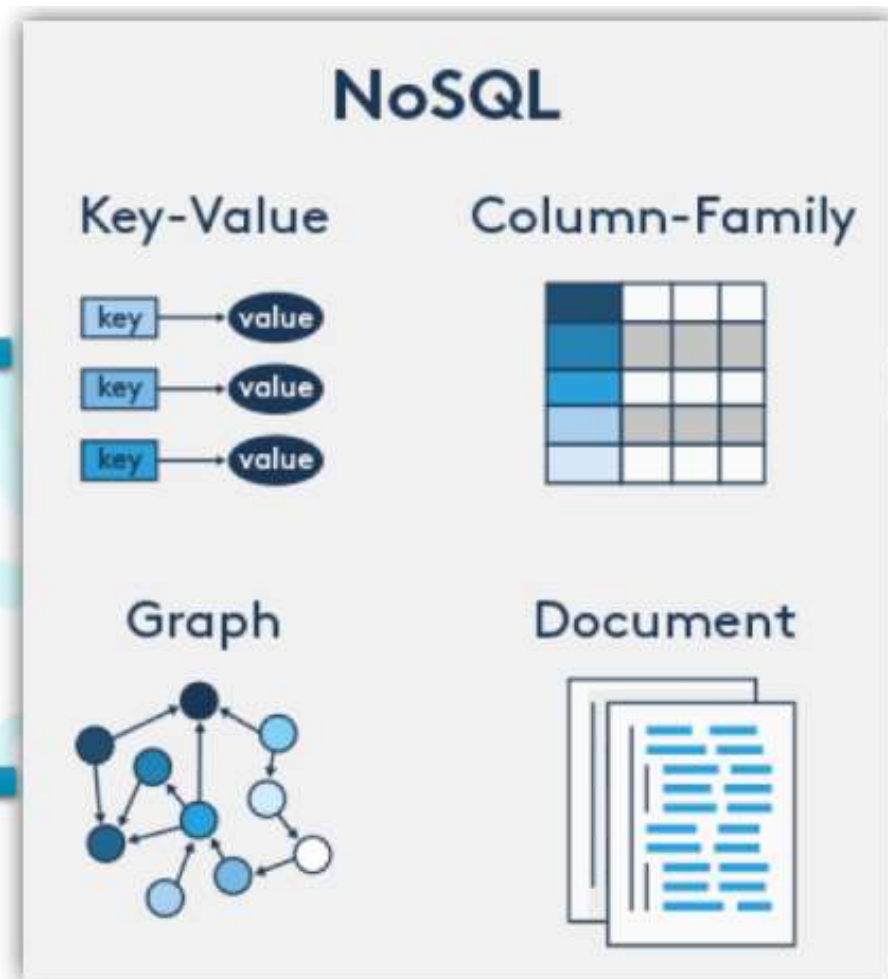
Challenges with globally distributed Databases

- Long time
- Lot of effort
- Need own infrastructure
 - Teams
 - Data centers etc.



Why Cosmos DB?

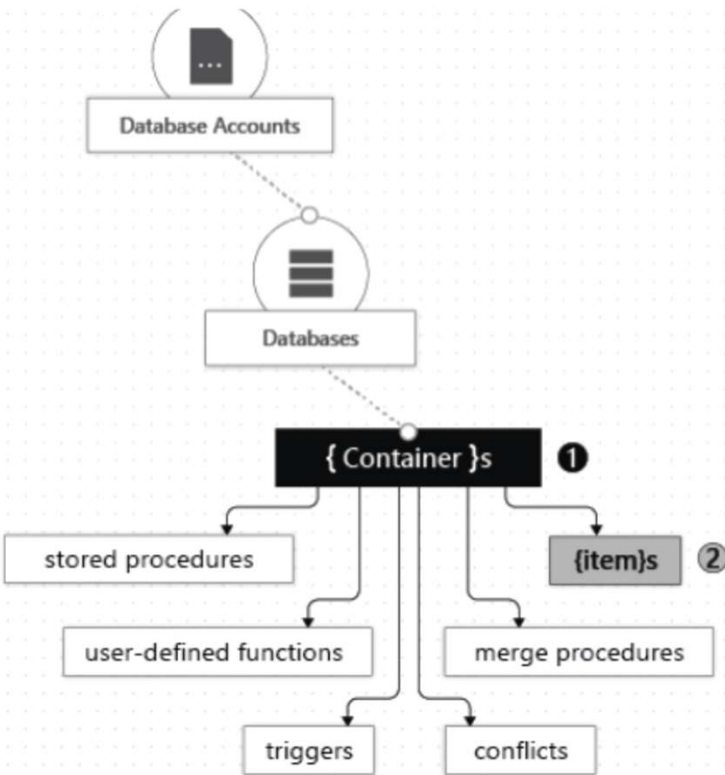
- FULLY MANAGED
- FULLY MANAGED
- GLOBALLY DISTRIBUTED
- CONSISTENCY CHOICES
- SCALABLE
- HIGHLY AVAILABLE, RELIABLE & SECURE



Analyze the decision criteria

	Core (SQL)	MongoDB	Cassandra	Azure Table	Gremlin
New projects being created from scratch	✓				
Existing MongoDB, Cassandra, Azure Table, or Gremlin data		✓	✓	✓	✓
Analysis of the relationships between data					✓
All other scenarios	✓				

Database Containers and Items



① { Container }s
Depending on the Cosmos API, a container is realized as:

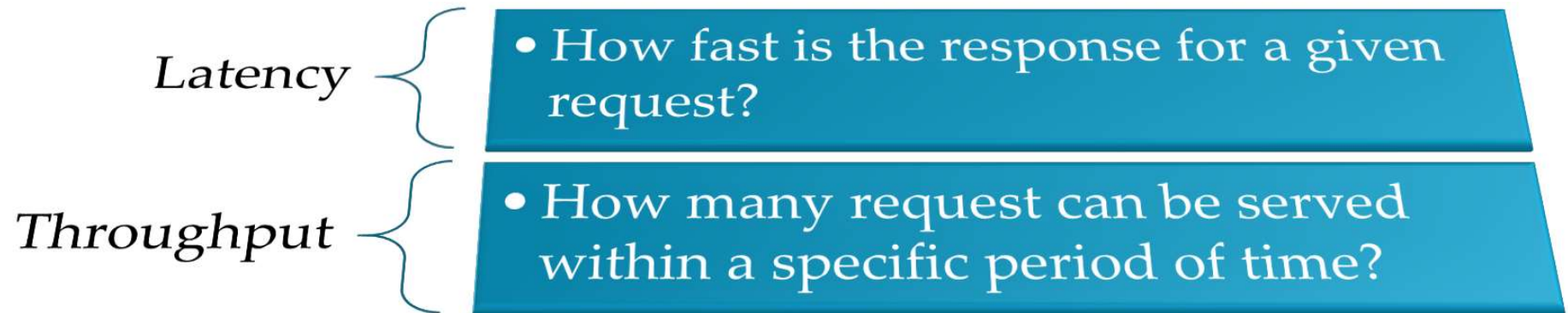


② { item }s
Depending on the Cosmos API, an item is realized as:



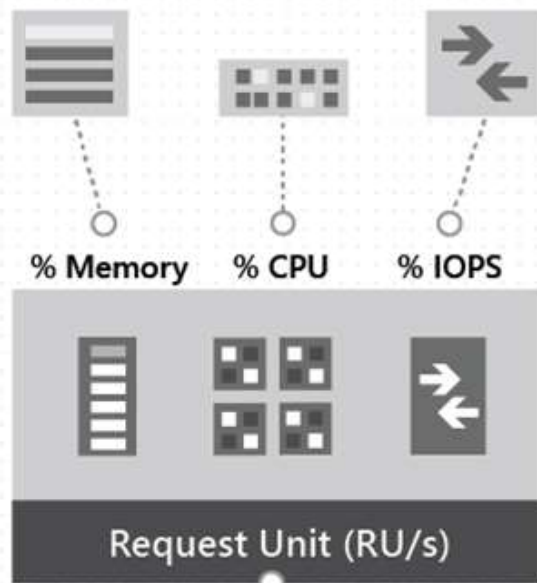
Azure Cosmos entity	SQL API	Cassandra API	MongoDB API	Gremlin API	Table API
Azure Cosmos database	Database	Keyspace	Database	Database	NA
Azure Cosmos container	Container	Table	Collection	Graph	Table
Azure Cosmos item	Document	Row	Document	Node or edge	Item

Measuring Performance

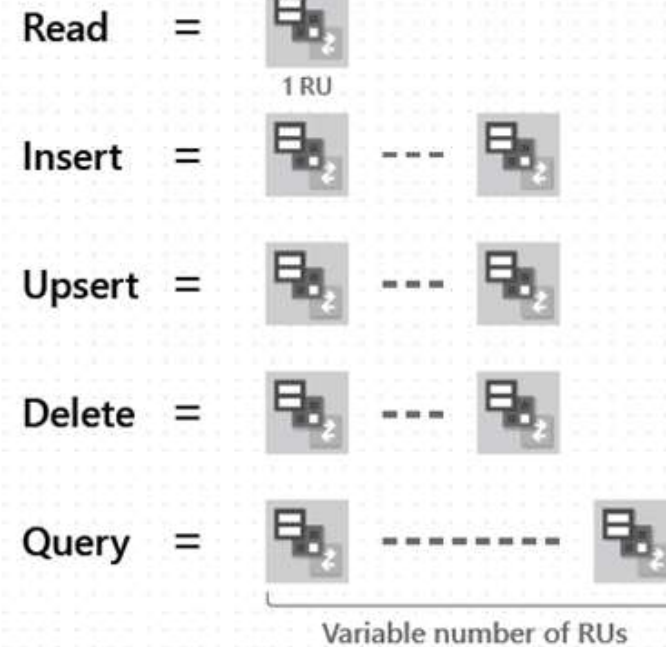


Introducing Request Units

Usage is expressed in Request Units



Database operations consume a variable number of RUs



Introducing Request Units

The screenshot shows the Azure Data Studio interface. At the top, there is a toolbar with buttons for 'New Database', 'New Collection', 'Open Full Screen', 'New SQL Query' (highlighted with a red box), and 'Feedback'. Below the toolbar, the left sidebar shows a tree view with 'SQL API' expanded, and 'flights' selected. Under 'flights', 'departuredelays' is selected, showing sub-items: 'Documents', 'Scale & Settings', 'Stored Procedures', 'User Defined Functions', and 'Triggers'. The main editor area has a tab for 'Query 1' and a button 'Execute Query' (highlighted with a red box). The query text is: `1 SELECT * FROM d WHERE d.origin = 'ABE'`. Below the query, there are two tabs: 'Results' and 'Query Stats' (highlighted with a red box). The 'Query Stats' tab displays a table with the following data:

METRIC	VALUE
Request Charge	46.180000000000001 RUs
Showing Results	1 - 100
Round Trips	1

Reserving requests units

- Provision Request units per second (RU/s)
 - How many request units (not requests) per second are available to your application
- Exceeding reserved throughput limits
 - Requests are “throttled” (HTTP 429)

* Container id ⓘ

B

* Partition key ⓘ

/id

☐ My partition key is larger than 100 bytes

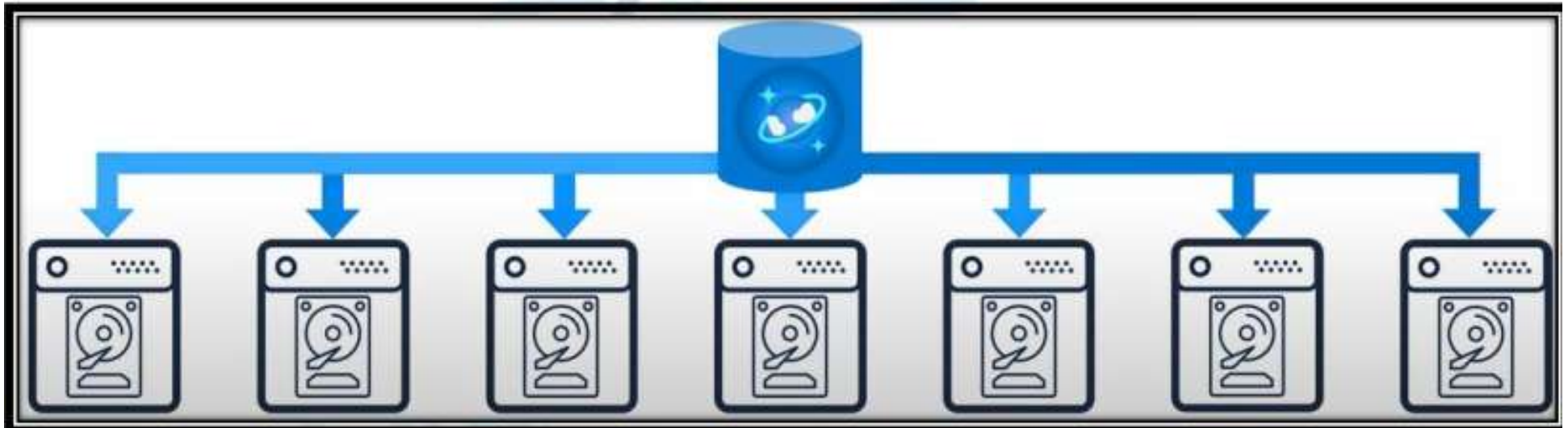
☒ Provision dedicated throughput for this container ⓘ

* Throughput (400 - 100,000 RU/s) ⓘ

400 − +

Estimated spend (USD): **\$0.58 hourly / \$13.82 daily** (8 regions, 400RU/s, \$0.00016/RU)

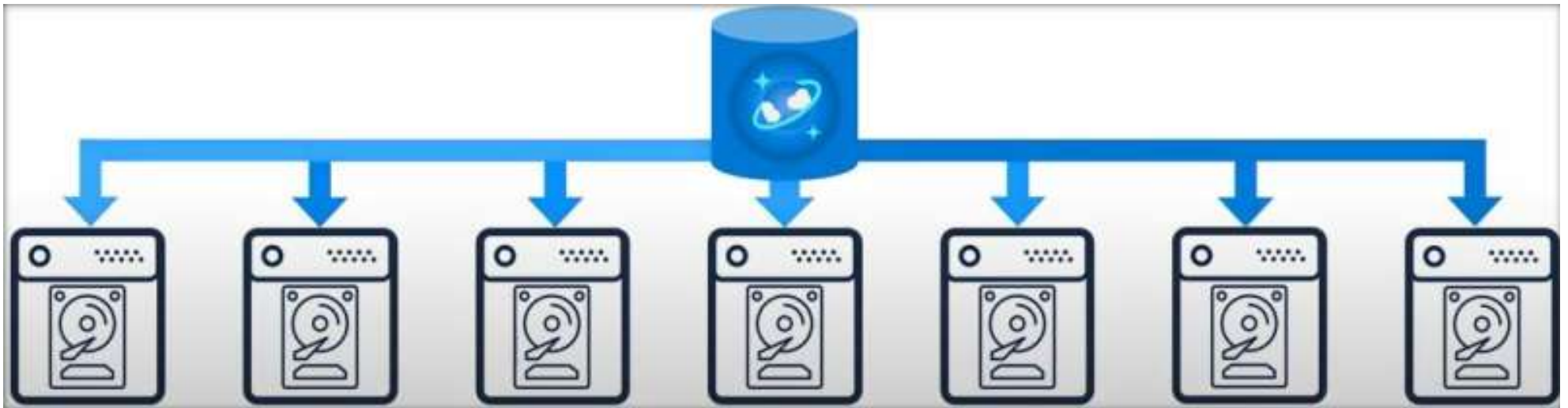
Horizontally Scalable



Unlimited Storage

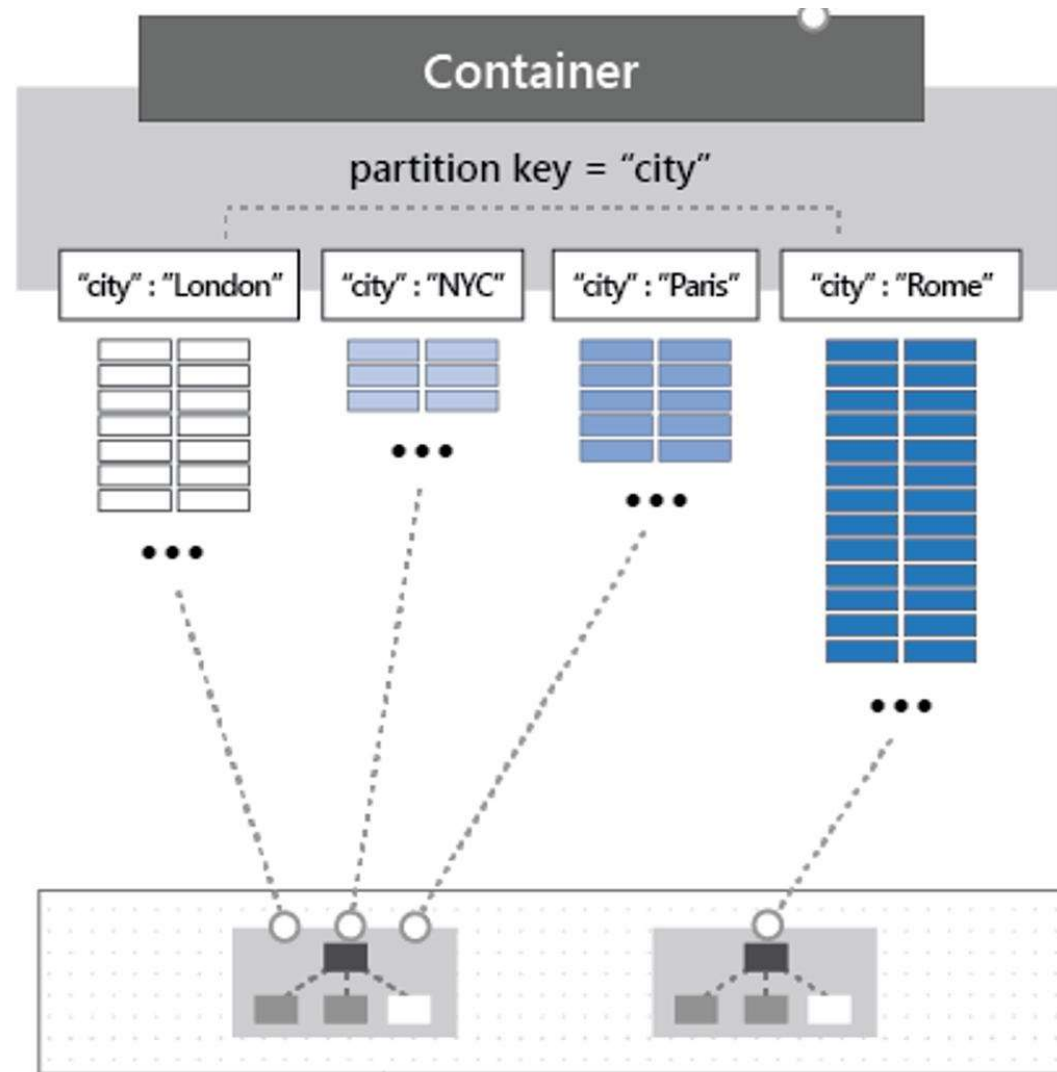
Unlimited Throughput

Partitioning

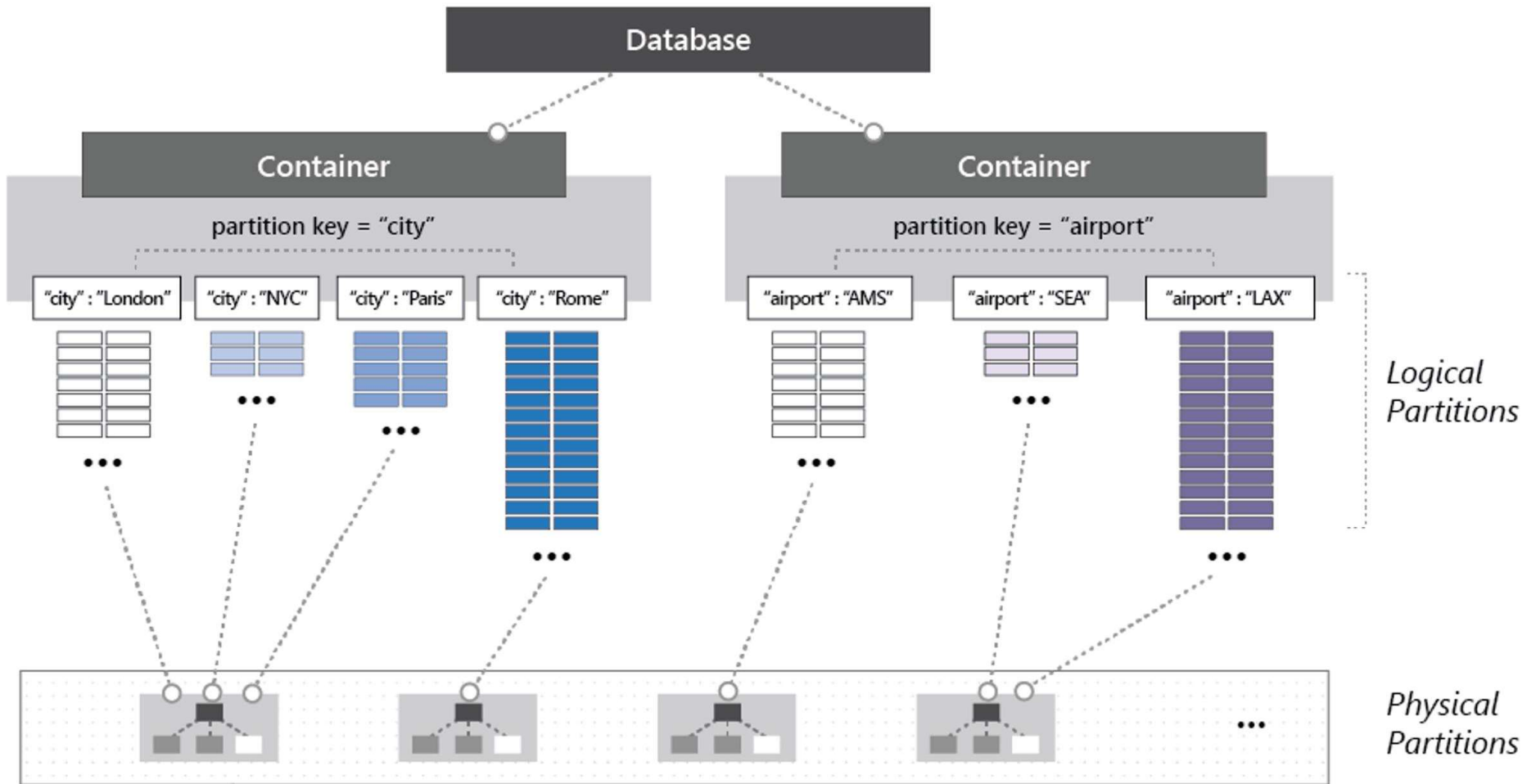


Partitioning

- Partitioning
 - The items in a container are divided into distinct subsets called logical partitions.
- Partition key
 - The value by which Azure organizes your data into logical divisions.
- Logical partitions
 - Are formed based on the value of a partition key that is associated with each item in a container.
- Physical partitions
 - Internally, one or more logical partitions are mapped to a single physical partition



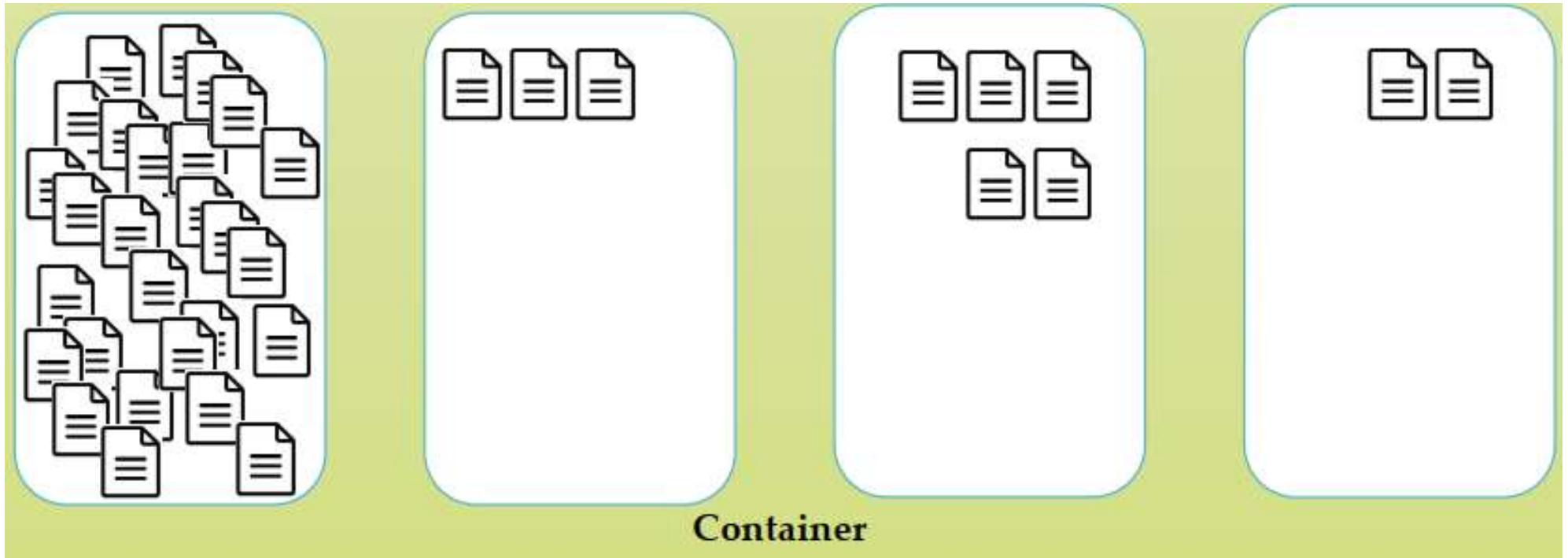
Partitioning



Dedicated vs Shared throughput

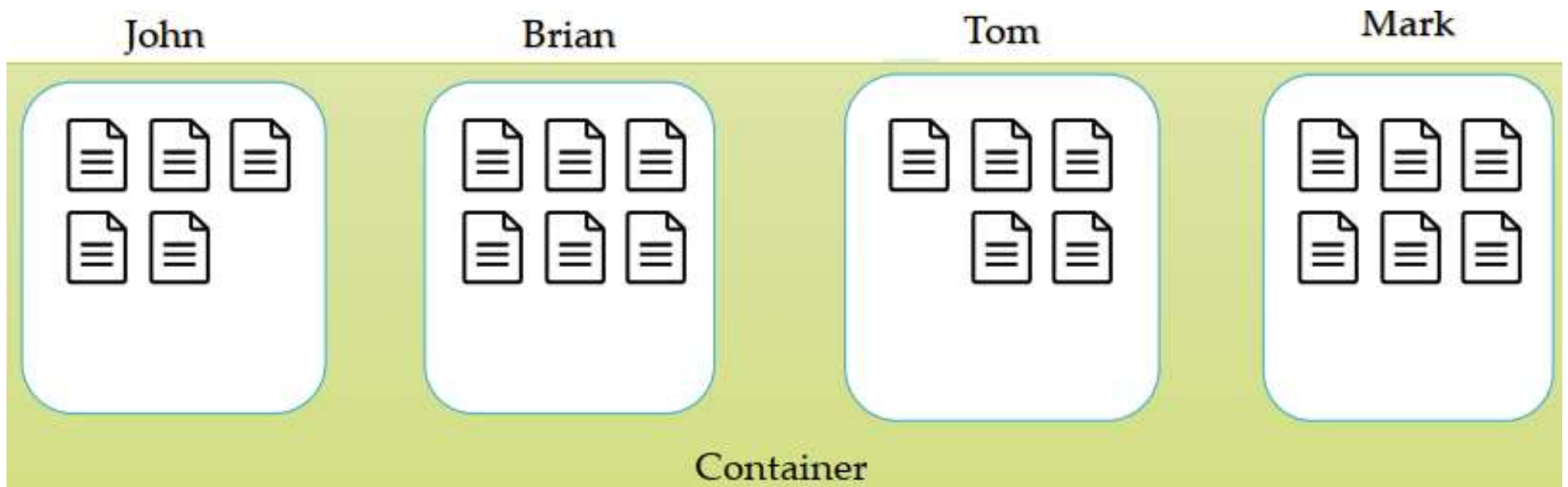
- You can set throughput at:
 - Database level – Shared throughput
 - Container level – Dedicated throughput
- It is recommend to set throughput at container level.
- Choose at the time of creation

Avoid Hot partitions on storage



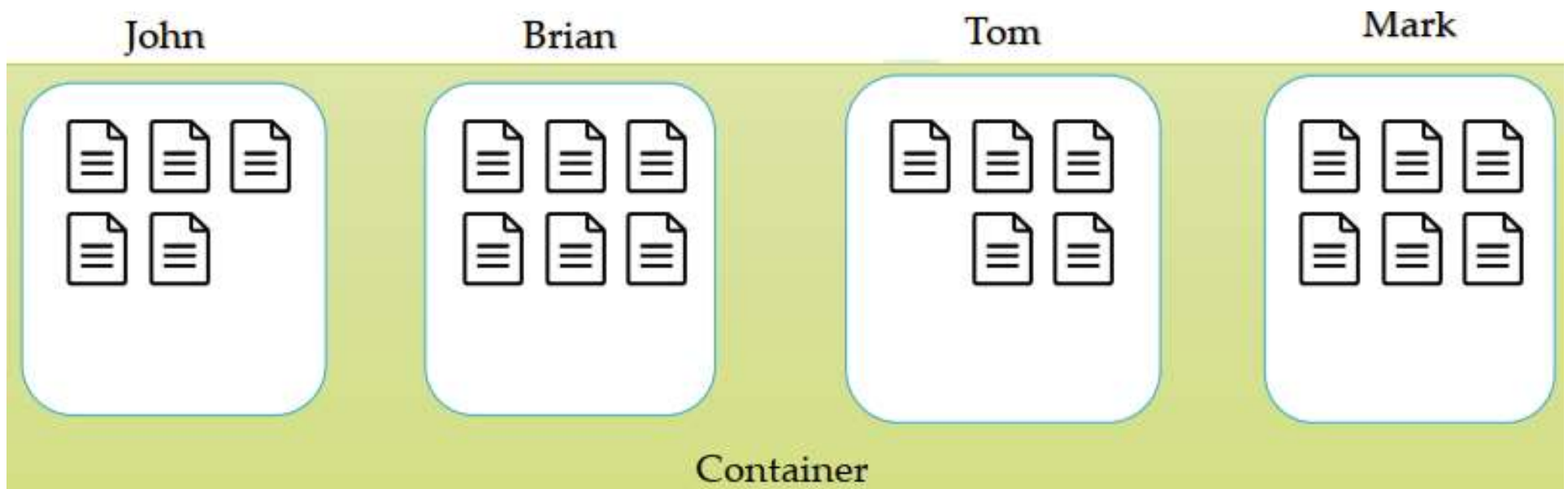
Single partition Query

- `SELECT * FROM c WHERE c.username = 'Brian'`



Cross partition Queries (fan out queries)

- `SELECT * FROM c WHERE c.favoritecolor= 'Blue'`



Automatic Indexing

- Index all data without requiring Index management
- Every property of every record automatically index
- Index update synchronously as you create, update or delete items
- Not specific for SQL, but available for all APIs

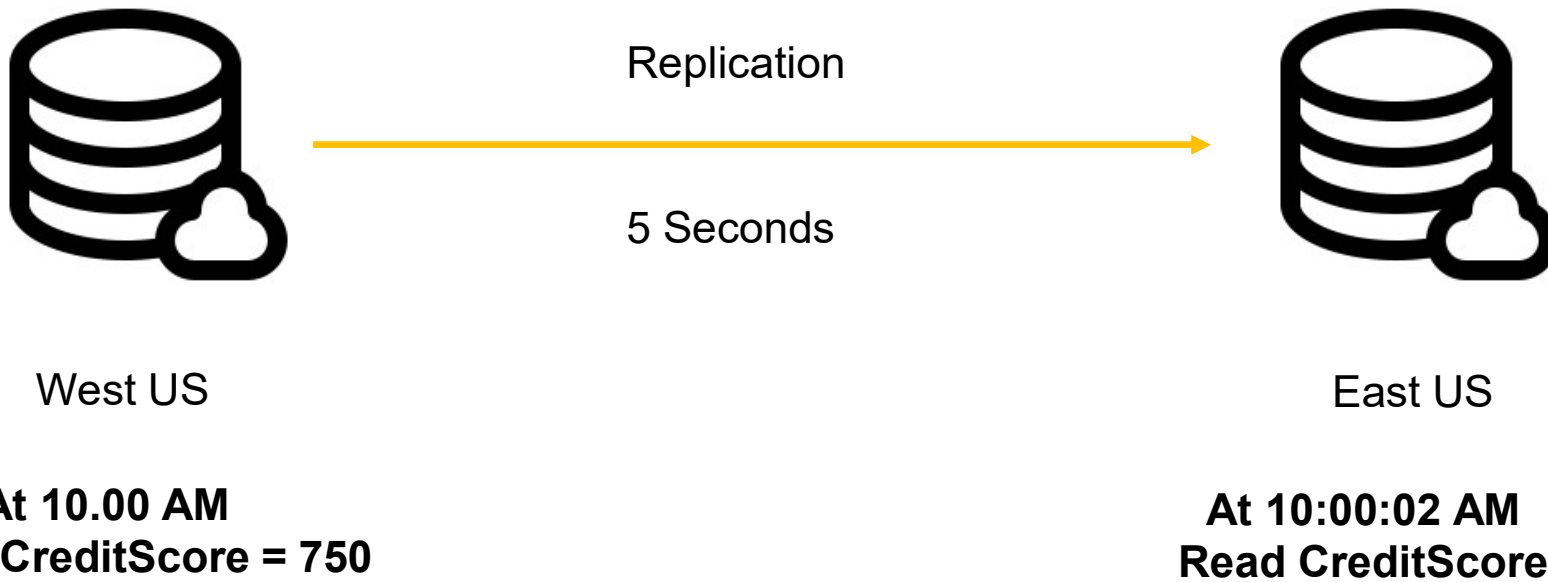
Time to Live (TTL)

- You can set the expiry time for Cosmos DB data items
- Time to live value is configured in seconds.
- The system will automatically delete the expired items based on the TTL value
- Consume only leftover Request units
- Data deletion delay if not enough Request units
- Though the data deletion is delayed, data is not returned by any queries (by any API) after the TTL has expired.

Global Distribution benefits

- Performance
- Business continuity

Data consistency





Five consistency Levels

- Strong
 - No dirty reads, high latency, cost highest, closest to RDBMS
- Bounded staleness
 - Dirty reads possible, bounded by time and updates
- Session
 - No dirty reads for writers (within same session), dirty read possible for other users
- Consistency prefix
 - Dirty reads possible but sequence maintain, reads never see out-of-order writes
- Eventual
 - No guaranteed order, but eventually everything gets in order

Setting the consistency level

- Set default for entire account
 - Can be changed any time
- Override at request level
 - Any request can weaken the default consistency level

Thanks