Microsoft Azure Developer: Develop Solutions with Azure Cosmos DB Storage

CREATING COSMOS DB CONTAINERS

David Tucker
TECHNICAL ARCHITECT & CTO CONSULTANT
@_davidtucker_ davidtucker.net

Objectives

Select the appropriate API for your solution

Create Cosmos DB containers

Interact with data using the appropriate SDK

Implement partitioning schemes

Set the appropriate consistency level for operations

Implement scaling (partitions, containers)

Implement server-side programming including stored procedures, triggers, and change feed notifications

NoSQL Databases

Database Approaches

Relational Databases

NoSQL Databases

NoSQL Differences

Relational Databases	NoSQL Databases
Fixed schema	**Fluid schema**
Table based structure	**Multiple structures** (key-value, graph, document, wide-column)
Vertical scaling and manual sharding for scalability	**Horizontal scaling and data partitioning for scalability**
Provides ACID guarantees (atomicity, consistency, isolation, durability)	**Provides BASE** (basically available, soft state, eventual consistency) **semantics**
Data normalization	**Non-normalized data**

"If your transactional volumes are reaching extreme levels, such as many thousands of transactions per second, you should consider a distributed NoSQL database."

Microsoft, Cosmos DB Documentation

Cosmos DB Capabilities

"**Azure Cosmos DB** is Microsoft's globally distributed, multi-model database service."

Microsoft, Cosmos DB Documentation

Azure Cosmos DB

Provides extremely low latency (single digit millisecond)

Provides SLA for throughput, latency, availability, and consistency

Support multi-region replication at any point

Provides five-nines of high-availability for both reads and writes

Enables elastic scalability

Pricing is for the throughput you provision*

Supports multiple consistency options

Additional Cosmos DB Features

Integrated Analytics

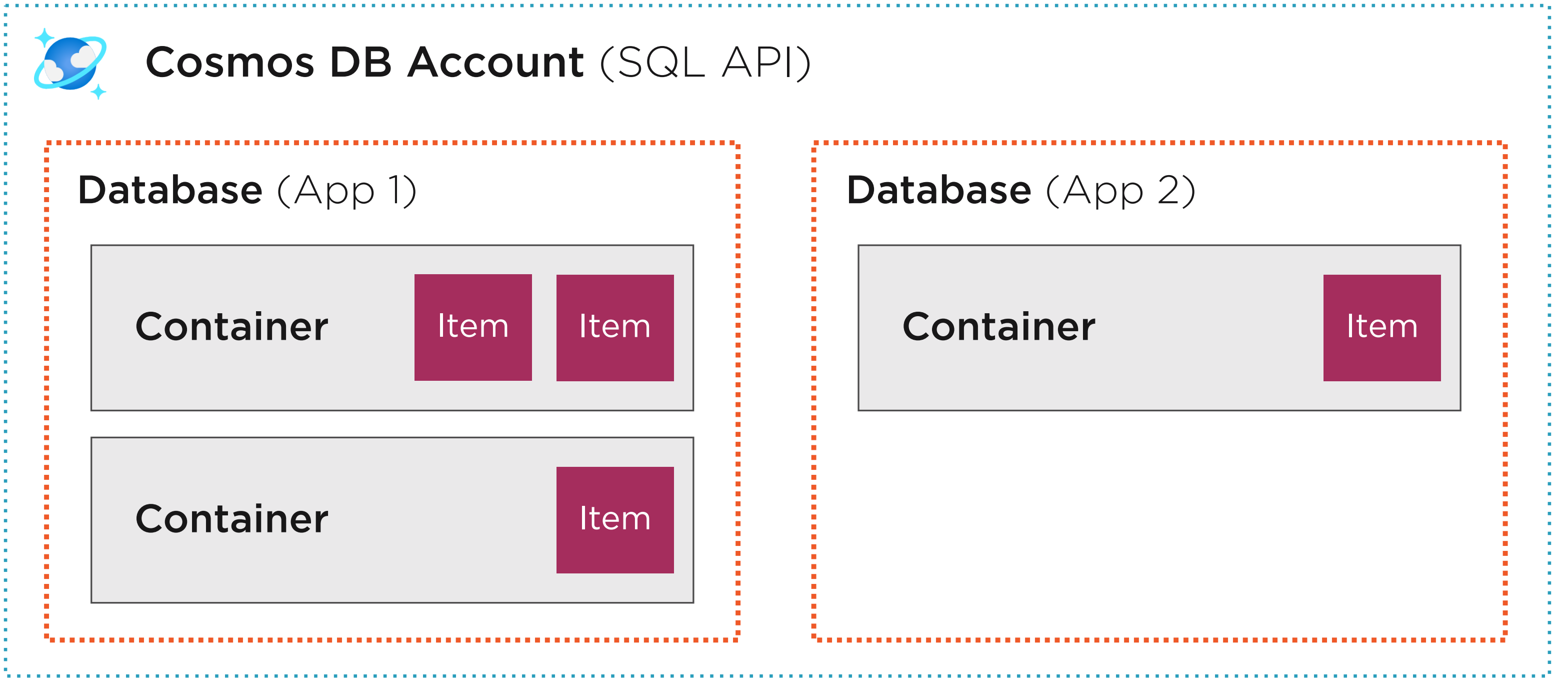
Region Support

Schema-agnostic

Automatic Indexing

Supports Multiple SDK’s

Cosmos DB Organization



Cosmos DB Account (SQL API)

Database (App 1)

Container Item Item

Container Item

Database (App 2)

Container Item

Supported Cosmos DB API’s

Supported Cosmos DB API's

- SQL
- Cassandra
- MongoDB
- Gremlin
- Azure Table

Cassandra API Use Cases

You want to leverage the Cassandra Query Language (CQL) to query data

You want to be able to leverage existing Cassandra tools

You have existing Cassandra databases that you want to migrate to the cloud

You want to store data in a wide-column format (two dimensional key-value store)

MongoDB API Use Cases

You want to leverage MongoDB API to query data

You want to be able to leverage existing MongoDB tools

You have existing MongoDB databases that you want to migrate to the cloud

You want to store data as JSON documents

Gremlin API Use Cases

You need to store graph relationships between data

Can leverage Apache Tinkerpop’s Gremlin language for querying relationships

Azure Table API Use Cases

You have experience with Azure Table Storage

You have applications and data to migrate from Azure Table Storage

You want to query data using OData or LINQ queries

SQL API Use Cases

You want to leverage a SQL-like language to query data

You want to store data as JSON documents

If no other use cases fit, choose the SQL API

Database Entity

SQL
Database

Cassandra
Keyspace

MongoDB
Database

Gremlin
Database

Azure Table
Not Applicable*

Container Entity

SQL
Container

Cassandra
Table

MongoDB
Collection

Gremlin
Graph

Azure Table
Table

Creating a Cosmos DB Container

```
# create a sql api cosmos db account
az cosmosdb create --name pluralsight --resource-group pluralsight

# create a sql database
az cosmosdb sql database create --account-name pluralsight
--name sampledb

# create a sql database container
az cosmosdb sql container create --resource-group pluralsight
--account-name pluralsight --database-name sampledb
--name samplecontainer --partition-key-path “/employeeid”
```

Creating a Cosmos DB Container using the CLI

Azure CLI

Demo

Creating a Cosmos DB account for the SQL API

Creating a Cosmos DB database

Creating a Cosmos DB container

Inserting items into the container

Querying the container

Selecting an SDK for Cosmos DB

Selecting an SDK

When using the SQL API, utilize the latest Cosmos DB SDK for your platform

When using MongoDB, Cassandra, and Gremlin use current SDK’s for those API’s

When leveraging the Azure Table API, leverage the current Table Storage SDK