Day 1 Module 1:

Introduction to Azure Cosmos DBD • Review of NoSQL database structures • Migrating data and applications to Cosmos DB • Managing data in Cosmos DB • Creating and using a SQL API database in Cosmos DB

Module 2: Designing and Implementing SQL API Database Applications • Document models in Cosmos DB • Querying data in a SQL API database • Querying and maintaining data programmatically • Designing and implementing SQL API database applications

Day 2 Module 3: Implementing Server-Side Operations • Server-side programming with Cosmos DB • Creating and using stored procedures • Using triggers to maintain data integrity • Writing user-defined functions, stored procedures, and triggers

Module 4: Optimizing and monitoring performance • Optimizing database performance • Monitoring the performance of a database • Tuning a database and monitoring performance

Day 3 Module 5: Designing and Implementing a Graph Database • Graph database models in Cosmos DB • Designing Graph database models for efficient operation • Designing and implementing a Graph database

Module 6: Querying and Analyzing Big Data with Cosmos DB • Integrating Cosmos DB with Azure search to optimize queries • Analyzing data in a Cosmos DB database using Apache Spark • Visualizing data in a Cosmos DB database • Querying and Analyzing Big Data with Cosmos DB

Day 4 Module 7: Implementing Stream Processing with Cosmos DB • Working with the Cosmos DB change feed • Integrating Cosmos DB into streaming solutions • Using Cosmos DB with stream processing

Pre-assessment quiz: <https://www.sanfoundry.com/mongodb-questions-answers-nosql-databases/>

**# Data**

- A data is a piece of information which we want to store

- it is fact which might be important which we need to preseve for future reference

**# Type of Data**

1. structured - table - RDBMS - MySQL, SQL Server, Oracle etc

2. semi-structured - JSON - NoSQL - Schemaless - MongoDB, Cassandra, Redis

3. unstructured - Phots, Video, files - AWS S3, Azure Blobs, Google Drive, OneDrive

**SQL Database**

Order

- orderId

- data

- amount

- custId

- addressId

OrderItem

- orderId

- pid

- pname

- price

- qty

Customer

- custId

- name

- email

- phone

Address

- addressId

- houseNumber

- city

- country

- pincode

NoSQL

{

 "orderId": "",

 "data": "",

 "amount": 1000,

 "customer": {

    "custId": 101,

    "name": "",

    "email": ""

 },

 "orderItems": [

    {

        }

 ]

}

# RDBMS vs NoSQL

RDBMS

- data is store in tables

- Vertically scalable

- Predefined Schema

- support powerfull query language

- can handle data in modrnate volumes

- has a centralised structure

- data can be written from one or few location

NoSQL

- Data can be stored as documents, graph, key value pair etc

- Horizontally scalable

- No predefined schema, hence easier to update

- Support simple query laguage

- can handle data in very high volumn

- has a decentralised structure

- data can be written from many location

# Azure Cosmo DB

It is globally distributed, low latency, multi-model database for managaing data at large scale

It is a cloud based NoSQL database offered as a PaaS (platform as a service) from Microsoft Azure

it is highly available

hight throughputs

reliable database and

is often called as a serverless database

# Azure Cosmos DB SQL API

it is fast No SQL database service that offers rich querying over diverse data

helps deliver configuration and reliable performance,

globally distributed and enable rapid development

# Advantages

- Guaranteed speed at any scale

- fast, flexible app development with SDKs

- ready for mission critical application

- fully managed and cist effective servless database

# How it works

# Components of Azure CosmoDB

1. Database Account

2. Database

3. Container

# Database Account

- it is fundamental unit of distribution and hight availablity

- we can configure the region for your data in Azure Cosmos DB

- globally unique DNS name used for API requst

# Database

each account can contain one or more then one database. a dataabse is a logical unit for management for container

in azure cosmos db

# containers

it is fundamental unit of scalabilityt in azure cosmos db SQL API where you can provision throuighput at the container

level, you can also optianally configure indexing policy

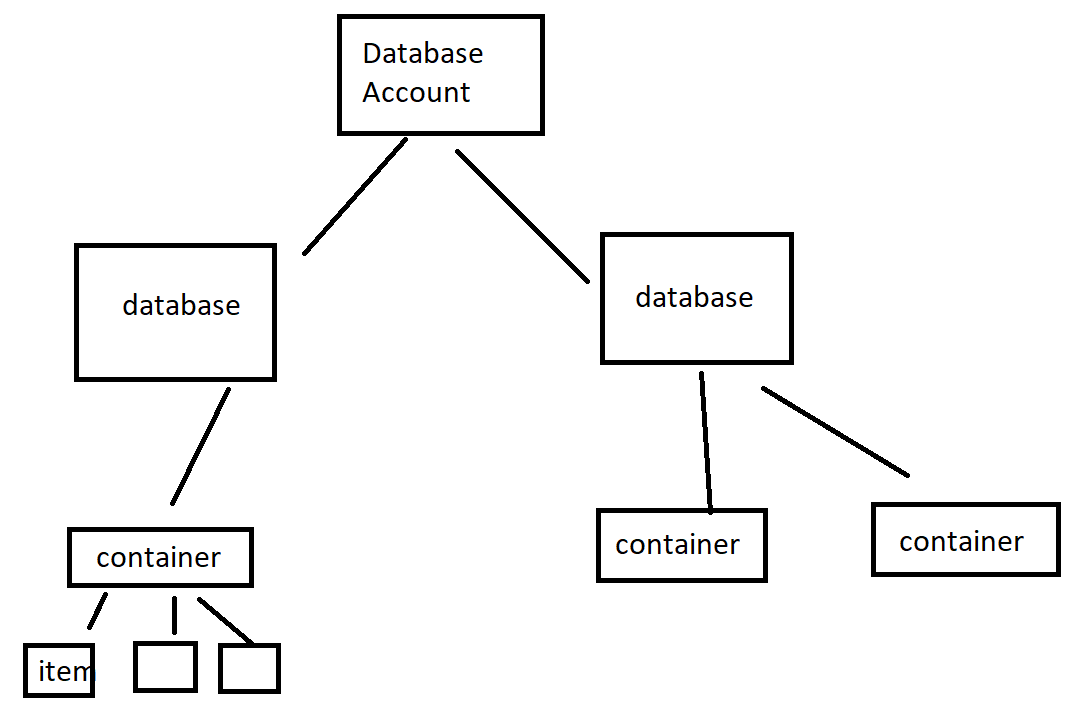
it can automatically partition data in a container

# items

it store individual documents in JSON format as items within the container

it support jSON files and can provide fast and predicatable performance bcoz write opeartion on json document

are atomic



# partitioning key

the container required to specify a partion key path

{

    pid: "",

    name: "",

    sku: "",

    price: 100

   }

   /products/sku