Get the best out of Live Sessions

HOW?

e!

Check your Internet Connection

Log in 10 mins before, and check your internet connection to avoid any network issues during the LIVE session.

Speak with the Instructor

By default, you will be on mute to avoid any background noise. However, if required you will be **unmuted by instructor**.

Clear Your Doubts

Feel free to clear your doubts. Use the "**Questions**" tab on your webinar tool to interact with the instructor at any point during the class.

Let us know if you liked our content

Please share feedback after each class. It will help us to enhance your learning experience.

edureka!

edureka!

Microsoft Azure Developer Associate (AZ-204)

COURSE OUTLINE
MODULE 06

- Introduction to Azure IaaS Compute Solutions
- Implementing Azure Batch Service and Disk Encryption
- Designing and Developing Applications That Use Containers
- Implementing Azure App Service Web Apps and Mobile Apps
- Implementing Azure App Service API Apps and Azure Functions
- **Developing Solutions That Use Azure Table Storage and Cosmos DB**
- Developing Solutions That Use Relational Database And Azure Blob Storage
- Implementing Authentication and Access Control In Azure
- Implementing Secure Data Solutions and Integrate Caching & CDN
- Instrument Monitoring, Logging and Scalability Of Apps & Services
- Connecting to and Consuming Azure and Third-party Services
- Developing Event-based and Message-based Solutions in Azure

Module 6 – Developing Solutions That Use Azure Table Storage and Cosmos DB

edureka!

Topics

- Azure Storage overview
- General Purpose Storage accounts
- Replication
- Azure Table Storage
- Azure Table Service Data Model
- Authorization In Table Storage
- Shared Key For Table Service
- Table Service REST API
- Entity Group Transactions
- Azure Cosmos DB
- Cosmos DB Database, Containers and Items
- CRUD operations using appropriate APIs
- Handling Documents
- Implement Scaling in Azure Cosmos DB
- Implement Server-side Programming

Objectives

After completing this module, you should be able to:

- Understand the features and uses of Azure Table storage
- Learn how to utilize Shared Key authorization
- Use the Azure Table storage REST service to manage data
- Understand core features and functionality of Azure Cosmos DB
- Manage containers and items
- Create and update documents
- Implement Scaling in Azure Cosmos DB
- Implement Server-side Programming

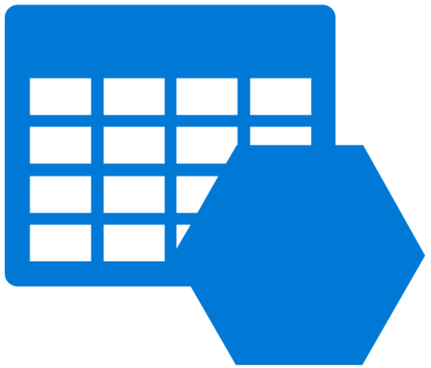
edureka!

Azure Storage

edureka!

Azure Storage Overview

- Cloud storage solution for modern applications that rely on durability, availability, and scalability to meet the needs of their customers
- Massively scalable, so you can store and process hundreds of terabytes of data
- Elastic, so you can design and scale applications according to the amount of data stored and the number of requests made against it
- Uses an auto-partitioning system that automatically load-balances your data based on traffic



Azure Storage

Azure Storage Services

- It stores unstructured object data
- A blob can be any type of text or binary data, such as a document, media file, or application installer
- Blob storage is also referred to as **Object storage**

Blob Storage

- It stores structured datasets
- Table storage is a NoSQL key-attribute data store, which allows for rapid development and fast access to large quantities of data

Table Storage

- It provides reliable messaging for workflow processing and for communication between components of cloud services

Queue Storage

- It offers shared storage for legacy applications using the standard SMB protocol
- Azure services can share file data across application components via mounted shares, and on-premises applications can access file data in a share via the File service REST API

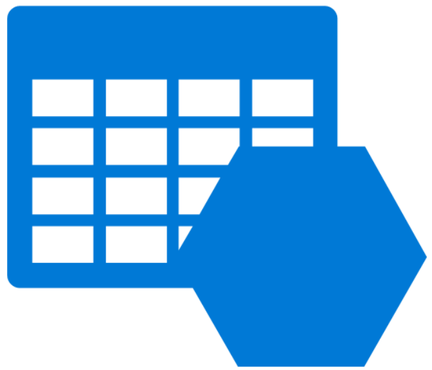
File Storage

edureka!

Azure Storage Accounts

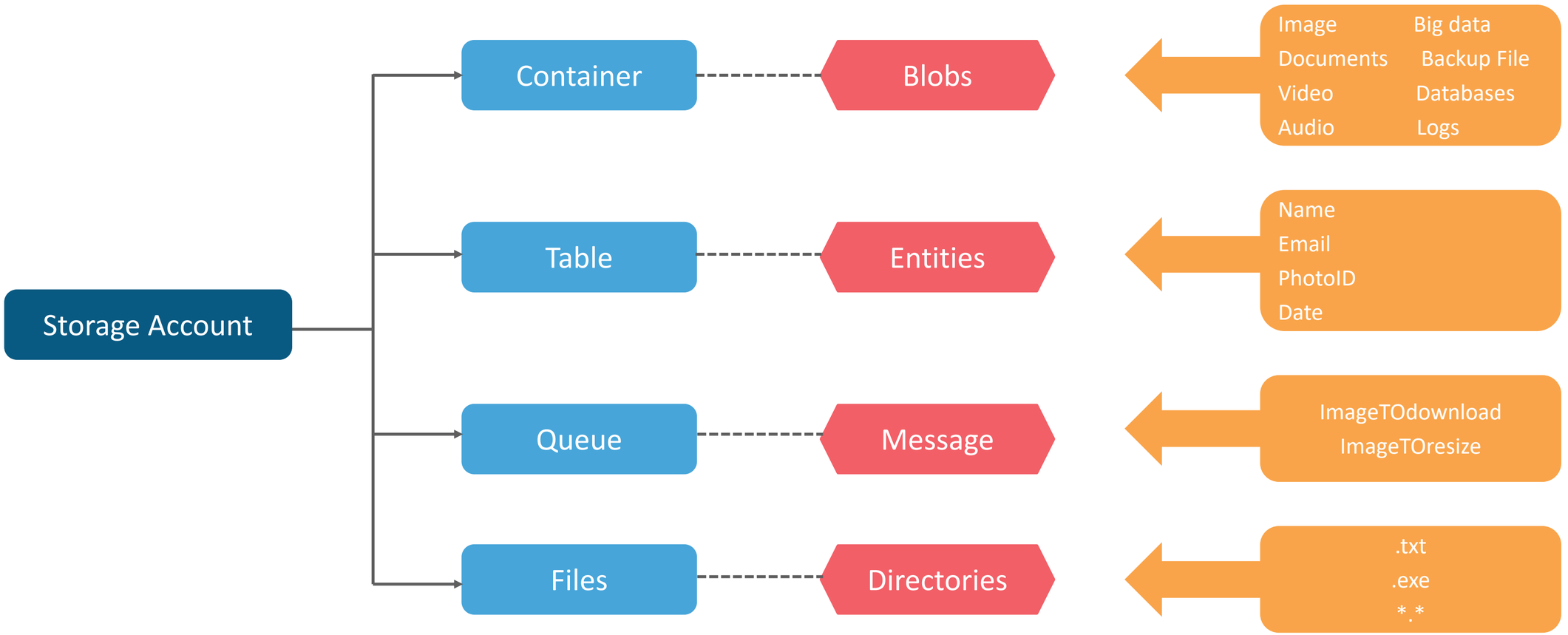
Azure Storage provides three types of storage accounts that support different features and pricing models to determine which one is best for your applications. These are:

- General-purpose v2 accounts (mostly recommended)
- General-purpose v1 accounts
- Blob storage accounts



General-purpose Storage Accounts

A general-purpose storage account gives you access to Azure Storage services such as Tables, Queues, Files, Blobs and Azure virtual machine disks under a single account.



edureka!

Replication

- To ensure data durability, Azure Storage replicates **multiple copies of the data**
- Replication type is selected when storage account is set up and in most cases, this setting can be modified after the storage account has been created

Replication options for a storage account include:

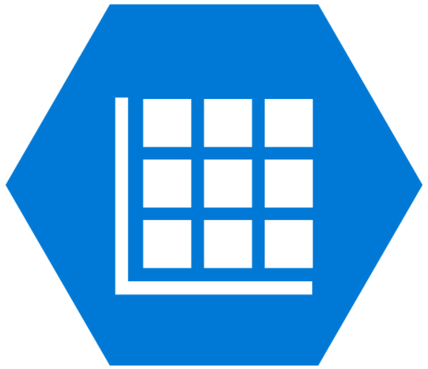
- Locally-redundant storage (LRS)
- Zone-redundant storage (ZRS)
- Geo-redundant storage (GRS)
- Read-access geo-redundant storage (RA-GRS)

Azure Table Storage

edureka!

Azure Table Storage

Azure Table Storage is a NoSQL key-value store for rapid development using massive semi-structured datasets



Azure Table Storage

It embraces a strong consistency model and is most opt for Enterprise-level data stores

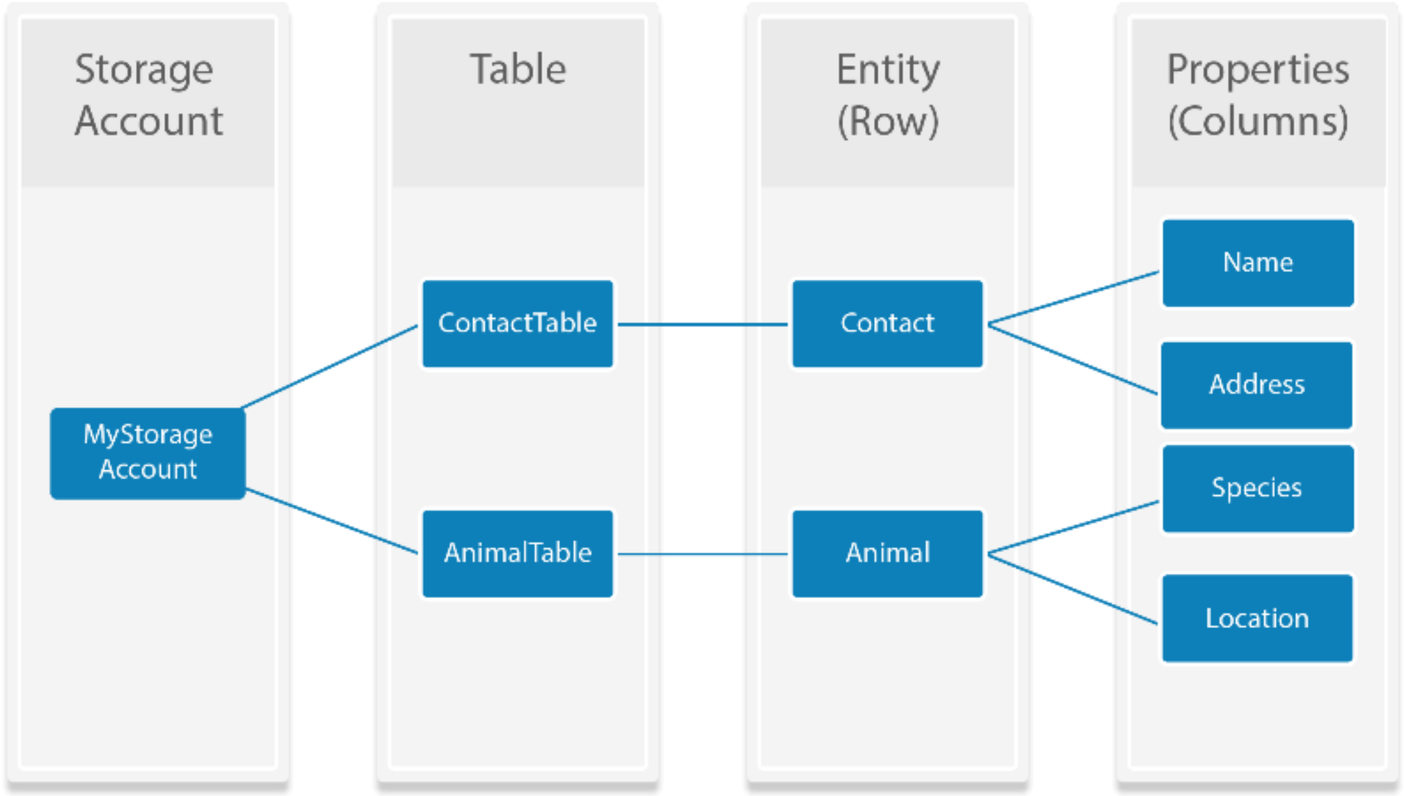
Azure Table Storage – Functionalities

- Stores semi-structured data that is highly available
- Creates apps that require a flexible data schema
- Performs OData-based queries
- Creates massively-scalable apps
- Uses JSON to serialize data

Table Storage Concepts

Table storage contains the following components:

- **Accounts:** All access to Azure Storage is done through a storage account.
- **Tables:** They store data as collections of entities
- **Entities:** These are similar to rows, an entity has a primary key and a set of properties. A table can contain any number of entities
- **Properties:** A property is a name, typed-value pair, similar to a column



Azure Table Service Data Model – Table Name Rules

01 Table names must be unique within an account

02 Table names may contain only alphanumeric characters

03 Table names cannot begin with a numeric character

04 Table names must be from 3 to 63 characters long

05 Some table names. Like ‘tables’ are reserved, attempting to create a table with a reserved table name returns an error

06 These rules are also described by the regular expression "^[A-Za-z][A-Za-z0-9]{2,62}$"

07 Table names preserve the case with which they were created, but are case-insensitive when used

edureka!

Azure Table Service Data Model – Properties

Property Names:

01 Property names are case-sensitive strings, up to 255 characters in size

02 Property names should follow naming rules for C# identifiers

Property Limitations:

01 An entity can have up to 255 properties, including 3 system properties described in the next slide

02 Therefore, the user may include up to 252 custom properties, in addition to the 3 system properties

03 The combined size of all data in an entity's properties cannot exceed 1 MB

edureka!

Azure Table Service Data Model – System Properties

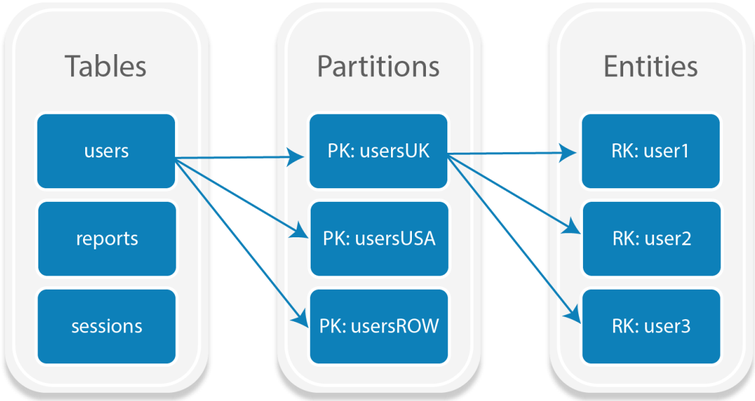
An entity always has the following system properties: **PartitionKey**, **RowKey**, **Timestamp**

These system properties are automatically included for every entity in a table

The names of these properties are reserved and cannot be changed

The developer is responsible for inserting and updating the values of **PartitionKey** and **RowKey**

The server manages the value of **Timestamp**, which cannot be modified

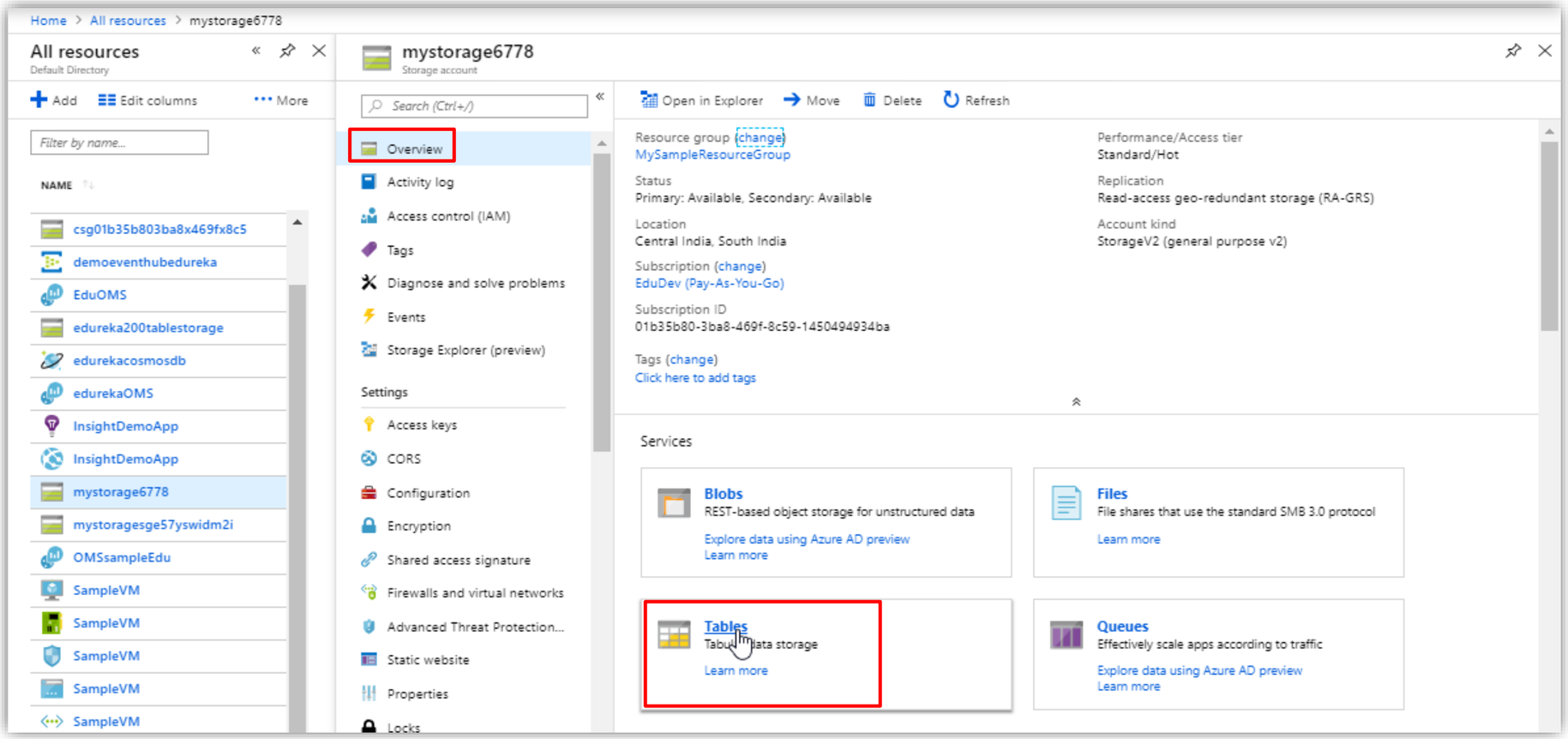


edureka!

Create a Table in Azure Portal

You can use Table service in the Azure portal to create a table.

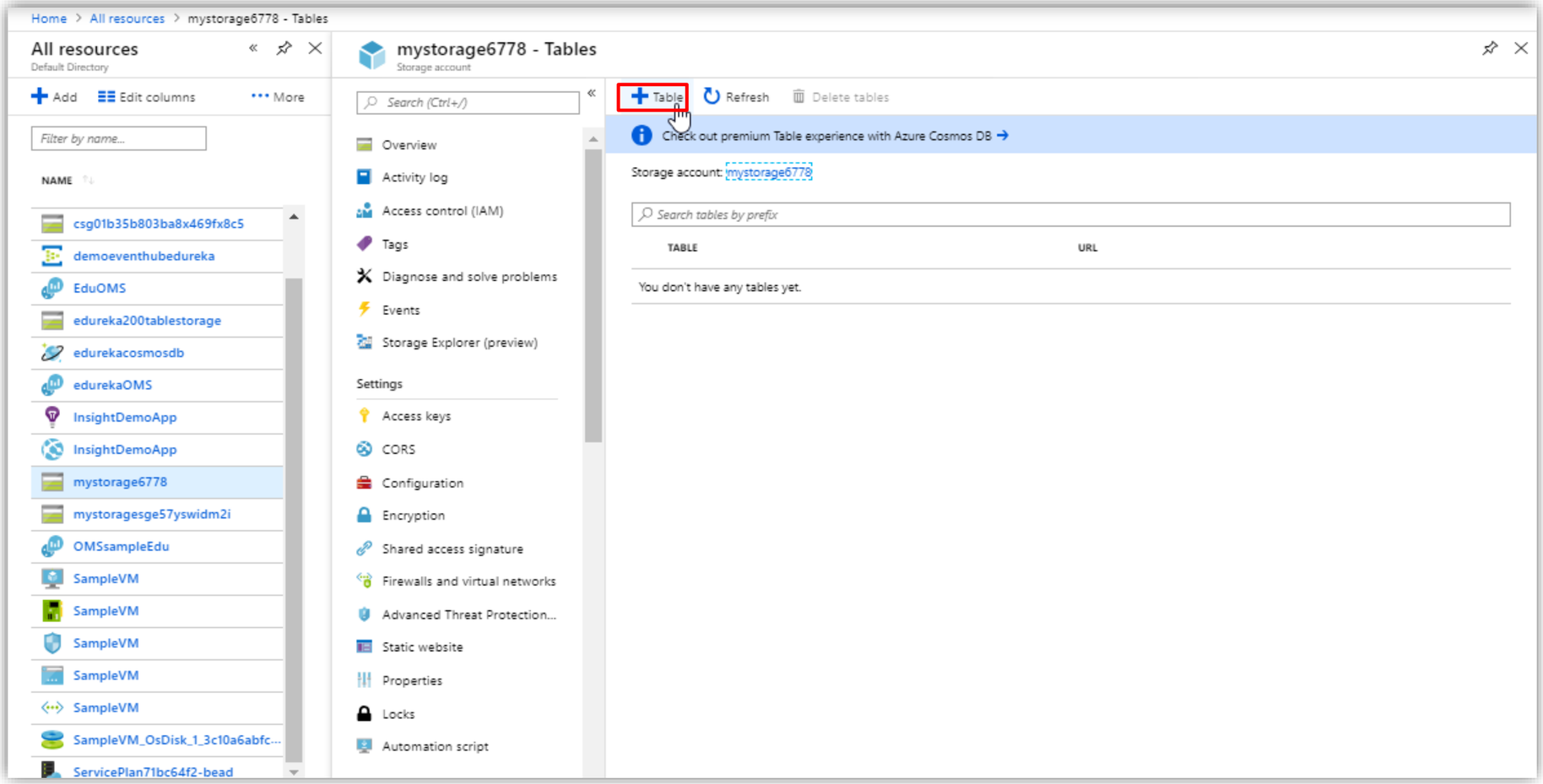
- Click on **Overview** of any storage account > then click on **Tables**



edureka!

Create a Table in Azure Portal (Cont.)

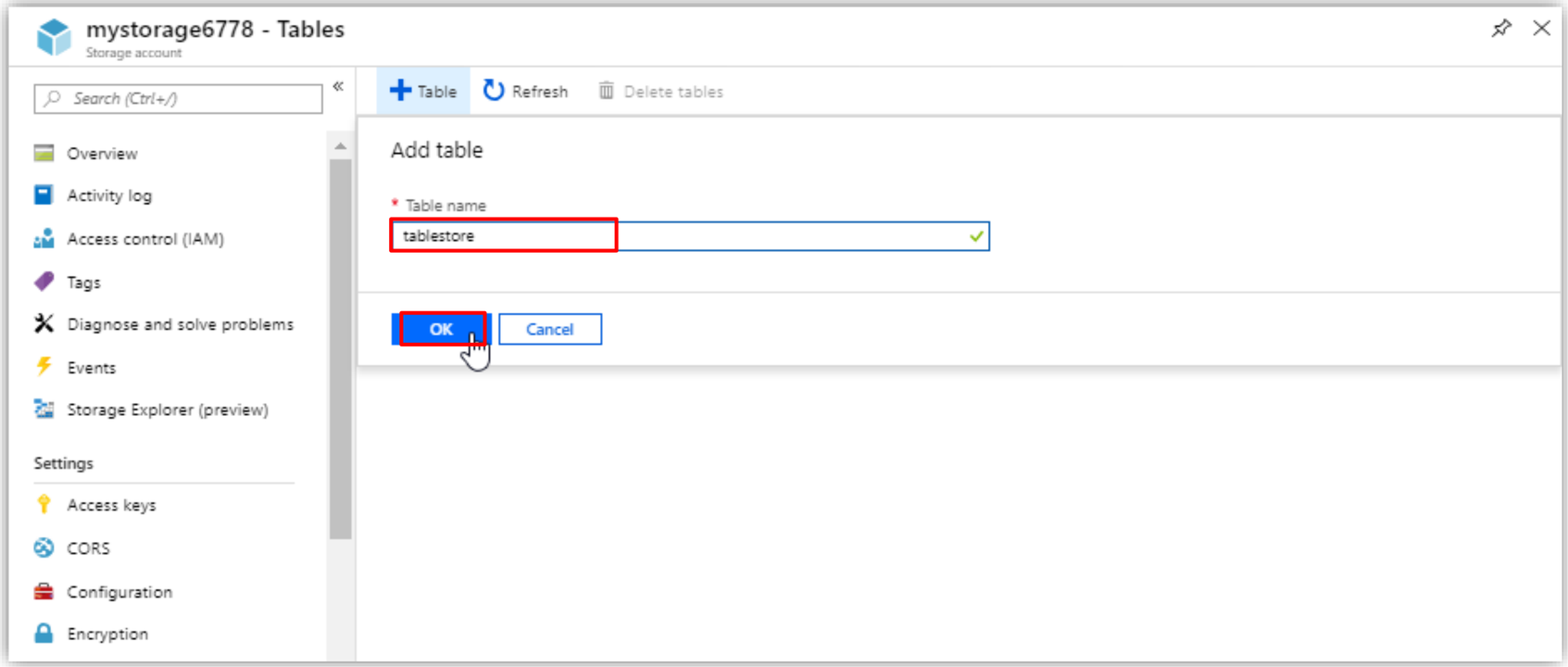
- Click on **Add Table**



edureka!

Create a Table in Azure Portal (Cont.)

- Type a name for your table in the **Table name** box > then click **OK**



Authorization in Table Storage

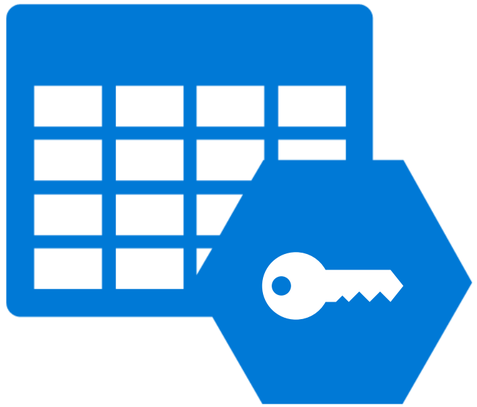
edureka!

Authorization for the Azure Storage Services

- Every request made against a secured resource in the Blob, File, Queue, or Table service must be authorized
- Authorization ensures that resources in your storage account are accessible:
  - only when you want them to be
  - and only to those users or applications to whom you grant access
- Options for authorizing requests to Azure Storage include:
  - **Azure Active Directory (Azure AD)**
  - **Shared Key**
  - **Shared access signatures**
  - **Anonymous access to containers and blobs**

Authorizing Requests Using Shared Key

Shared Key authorization relies on your **account access keys** and other parameters to produce an ***encrypted signature string*** that is passed on the request in the *Authorization* header



Every request made against a storage service must be authorized, unless the request is for a blob or container resource that has been made available for public or signed access

Shared Key for Table Service – Signature String

- Use the Shared Key authorization scheme to make requests against the Table service using the REST API
- The format of the ***signature string*** for Shared Key against the Table service is the same for all versions
- To encode the signature string for a request against the Table service made using the REST API, use the following format:

```
StringToSign = VERB + "\n" +
               Content-MD5 + "\n" +
               Content-Type + "\n" +
               Date + "\n" +
               CanonicalizedResource;
```

Table Service REST API

edureka!

Table Service REST API

- The Table service offers structured storage in the form of tables
- The Table service API is a REST API for working with tables and the data that they contain

Azure Table Service REST API – Operations

Operation	Description
Set Table Service Properties	Sets the properties of the Table service.
Get Table Service Properties	Gets the properties of the Table service.
Query Tables	Enumerates the tables in a storage account.
Create Table	Creates a new table within a storage account.
Delete Table	Deletes a table from a storage account.
Query Entities	Queries data in a table.
Insert Or Merge Entity	Inserts or merges an entity in a table.

Entity Group Transactions

- The Table service supports batch operations for
  - Insert Entity
  - Update Entity
  - Merge Entity
  - Delete Entity

  operations.

Demo 1 – Use Azure Table Storage REST Service to Create and Manage Data

edureka!

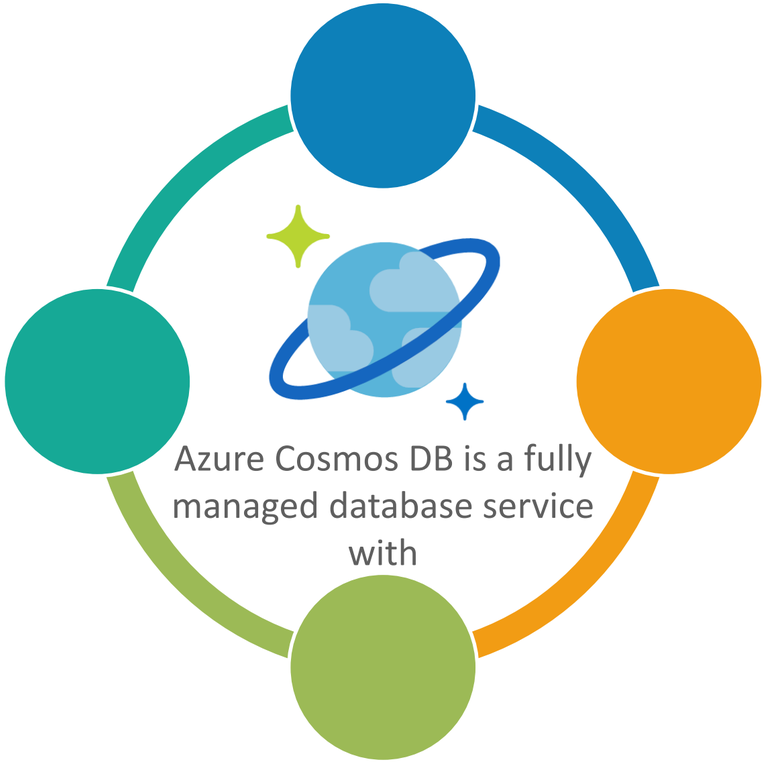
Azure Cosmos DB

edureka!

Azure Cosmos DB – Overview

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

Elastic and unlimited scalability to meet demand with capacity, and payment based on only the throughput and storage you need



Multi-model with wire protocol–compatible API endpoints for Cassandra, MongoDB, SQL, Gremlin, Etcd, and Table along with built-in support for Apache Spark and Jupyter notebooks

Single-digit millisecond latency at the 99th percentile and 99.999-percent high availability, for any scale, backed by SLAs

Azure Cosmos DB – Features and Support

Table

SQL

SQL

JavaScript

{LEAF}

API for MongoDB

Gremlin

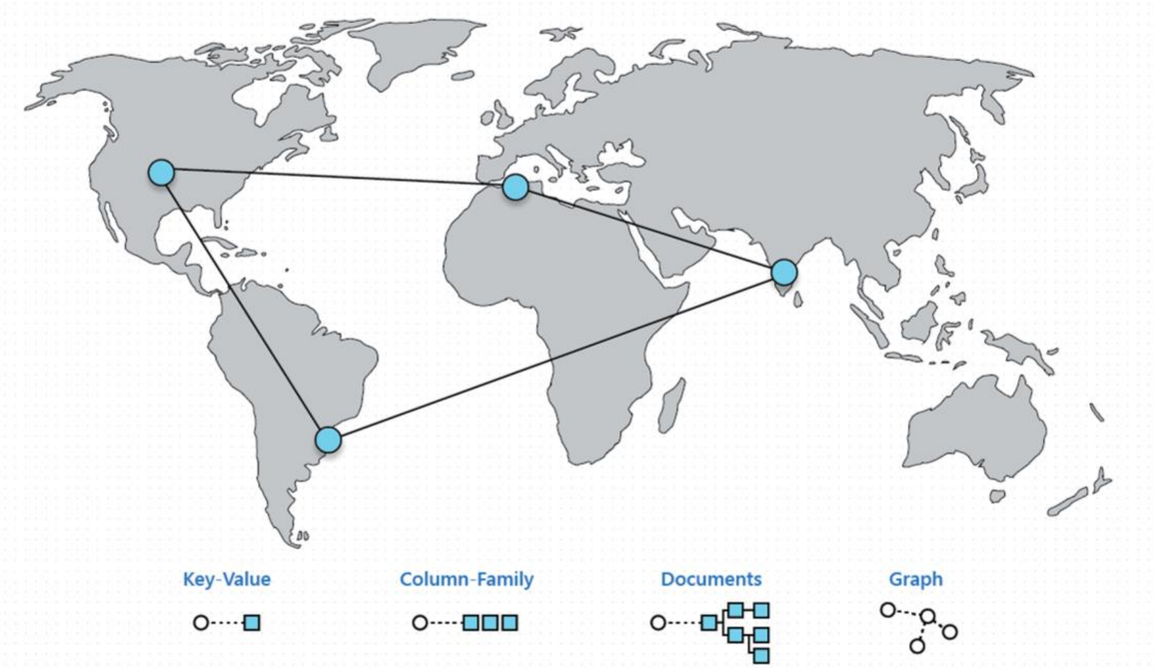
etcd

ETCD

Spark

Cassandra

... more APIs coming

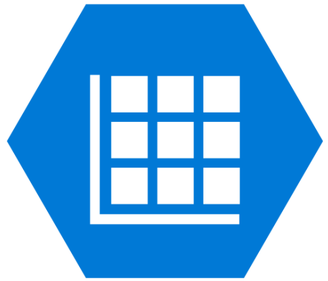


- Global distribution
- Elastic scale-out
- Guaranteed low latency
- Five consistency models
- Comprehensive SLAs

edureka!

Account Types Based on APIs

- The API determines the type of account to create
- Azure Cosmos DB provides five APIs:
  - Core(SQL) for ***document*** databases
  - Gremlin for ***graph*** databases
  - MongoDB for ***document*** databases
  - Azure Table, and Cassandra for ***NoSQL***
- Currently, you must create a separate account for each API

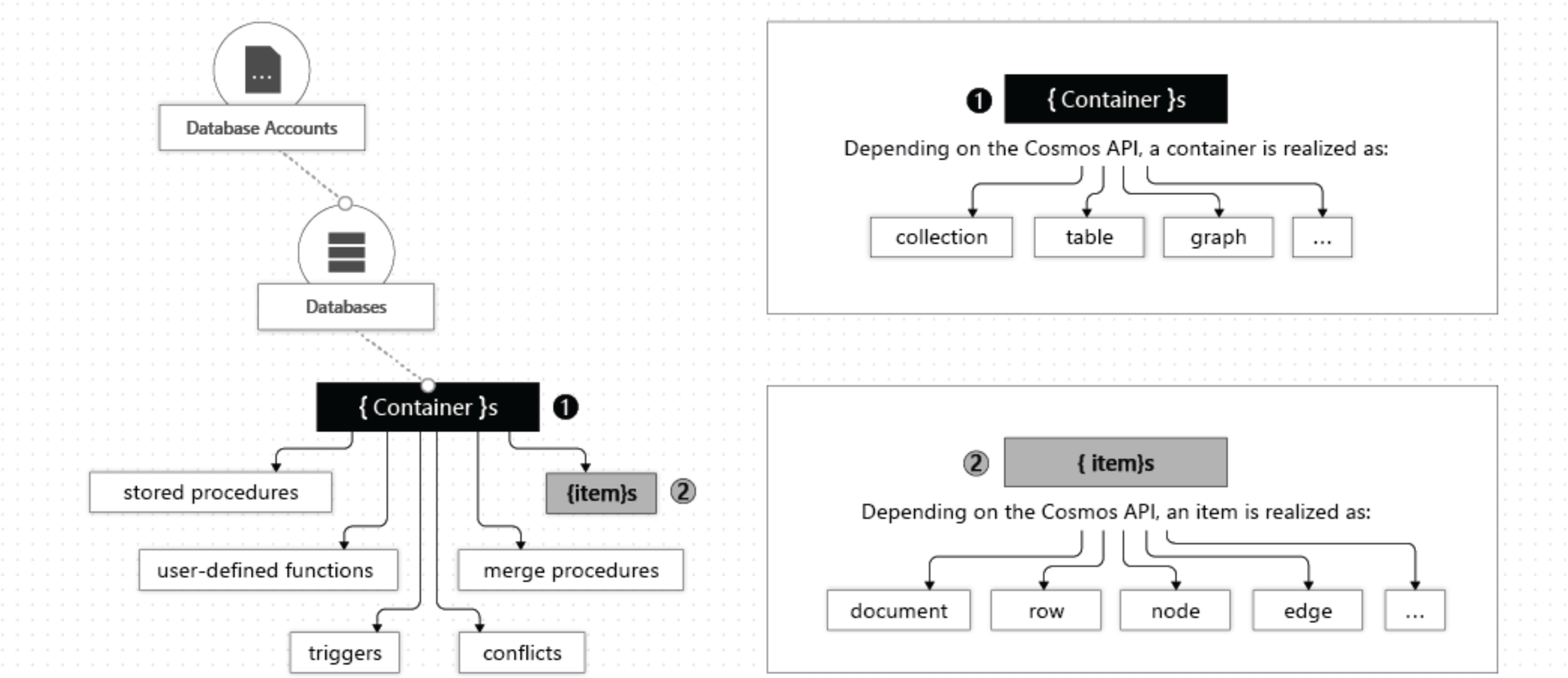


Work With Databases, Containers and Items in Azure Cosmos DB

edureka!

Databases, Containers and Items in Azure Cosmos DB

- After you create an ***Azure Cosmos DB account*** under your Azure subscription, you can manage data in your account by creating databases, containers, and items
- The following image shows the hierarchy of different entities in an Azure Cosmos DB account:



Azure Cosmos – Databases

- You can create one or multiple Azure Cosmos databases under your account
- A database is analogous to a ***namespace***
- A database is the ***unit of management*** for a set of Azure Cosmos containers
- The following table shows how an Azure Cosmos database is *mapped* to various API-specific entities:

Azure Cosmos entity	SQL API	Cassandra API	Azure Cosmos DB API for MongoDB	Gremlin API	Table API
Azure Cosmos database	Database	Keyspace	Database	Database	NA

Operations on Azure Cosmos Database

You can interact with an Azure Cosmos database with Azure Cosmos APIs as described in the following table:

Operation	Azure CLI	SQL API	Cassandra API	Azure Cosmos DB API for MongoDB	Gremlin API	Table API
Enumerate all databases	Yes	Yes	Yes (database is mapped to a keyspace)	Yes	NA	NA
Read database	Yes	Yes	Yes (database is mapped to a keyspace)	Yes	NA	NA
Create new database	Yes	Yes	Yes (database is mapped to a keyspace)	Yes	NA	NA
Update database	Yes	Yes	Yes (database is mapped to a keyspace)	Yes	NA	NA

Azure Cosmos – Containers

- An Azure Cosmos container is the ***unit of scalability*** both for provisioned throughput and storage
- A container is ***horizontally partitioned*** and then ***replicated*** across multiple regions
- The items that you add to the container and the throughput that you provision on it are ***automatically*** distributed across a set of logical partitions based on the partition key
- An Azure Cosmos container is specialized into API-specific entities as shown in the following table:

Azure Cosmos entity	SQL API	Cassandra API	Azure Cosmos DB API for MongoDB	Gremlin API	Table API
Azure Cosmos container	Collection	Table	Collection	Graph	Table

Operations on Azure Cosmos Container

An Azure Cosmos container supports the following operations when you use any of the Azure Cosmos APIs:

Operation	Azure CLI	SQL API	Cassandra API	Azure Cosmos DB API for MongoDB	Gremlin API	Table API
Enumerate containers in a database	Yes	Yes	Yes	Yes	NA	NA
Read a container	Yes	Yes	Yes	Yes	NA	NA
Create a new container	Yes	Yes	Yes	Yes	NA	NA
Update a container	Yes	Yes	Yes	Yes	NA	NA
Delete a container	Yes	Yes	Yes	Yes	NA	NA

Azure Cosmos – Items

- Depending on which API you use, an Azure Cosmos item can represent *either* :
  - a **document** in a ***collection***
  - a **row** in a ***table***
  - *or* a **node** *or* **edge** in a ***graph***
- The following table shows the mapping of API-specific entities to an Azure Cosmos item:

Cosmos entity	SQL API	Cassandra API	Azure Cosmos DB API for MongoDB	Gremlin API	Table API
Azure Cosmos item	Document	Row	Document	Node or edge	Item

Operations on Items

- Azure Cosmos items support the following operations and you can use any of the Azure Cosmos APIs to perform the operations:

Operation	Azure CLI	SQL API	Cassandra API	Azure Cosmos DB API for MongoDB	Gremlin API	Table API
Insert, Replace, Delete, Upsert, Read	No	Yes	Yes	Yes	Yes	Yes

Creating and Updating Documents Using C# Code

- The upcoming demo shows how to:
    1. Use the Azure portal to create an Azure Cosmos DB SQL API account
    2. Create a document database and collection
    3. Add data to the collection
    4. Then use a SQL .NET SDK web app to add more data to the collection

Demo 2 – Create, Read, Update and Delete Data Using Appropriate APIs

edureka!

Implement Scaling in Azure Cosmos DB

edureka!

Azure Cosmos DB and its Usage

Most modern apps require responsive and highly available online database systems. Azure Cosmos DB fulfills this requirement for various businesses by providing a full-fledged, globally-distributed, multi-model database as a service.

- Azure Cosmos DB allows businesses to scale across various geographies including various Azure Region worldwide
- It supports various APIs like SQL, Cassandra, Mongo DB, Gremlin and Table storage
- It can be used by various businesses that require faster data throughput with high availability and low latency such as in Retail, Marketing, IoT, Web and Mobile platforms

Note: We will discuss **Partitions** & **Containers** in more detail.

Azure Cosmos DB

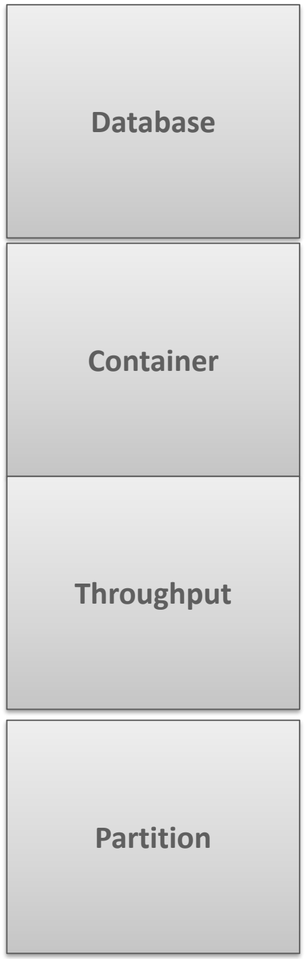
Important Terminologies

Database in Azure Cosmos DB is defined as a namespace and a unit for managing the database accounts. We need to provide a unique Azure Cosmos DB Account name to create a Database account.

Containers in Azure Cosmos DB are defined as a unit of scalability for storage. A container is generally horizontally partitioned and then replicated across multiple azure data centers.

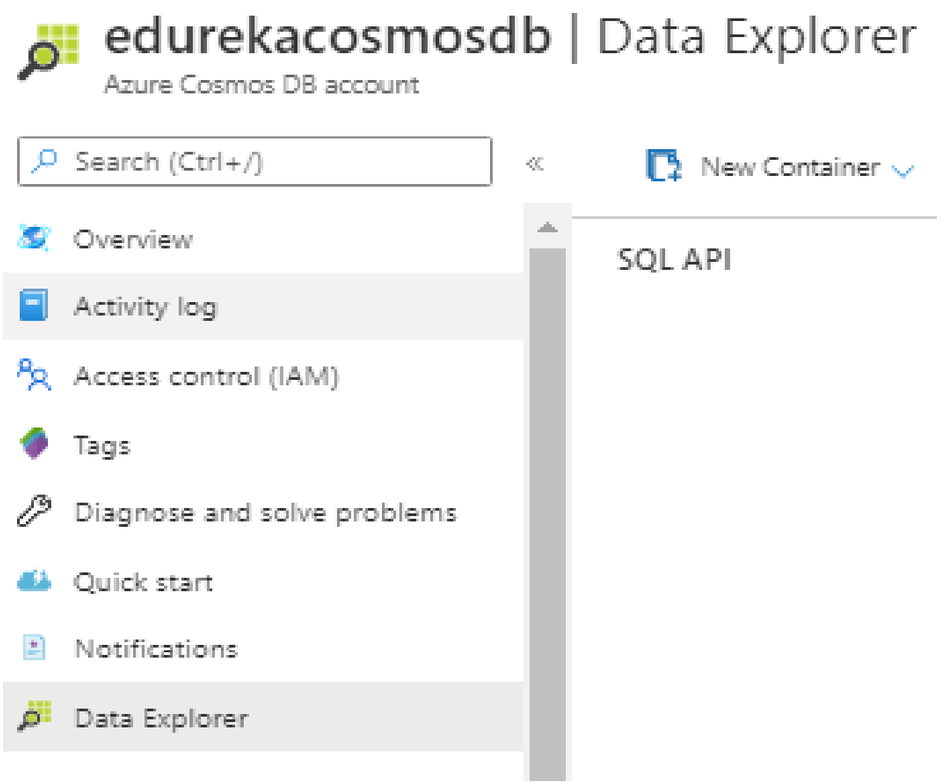
Throughput in Azure Cosmos DB is defined by Azure to offer a predictable performance scale. We can start with a minimum of 400 RU/sec and scale up to millions of requests per second. With Azure Cosmos DB free tier, we will get 400 RU/s and 5 GB of storage for free in an account.

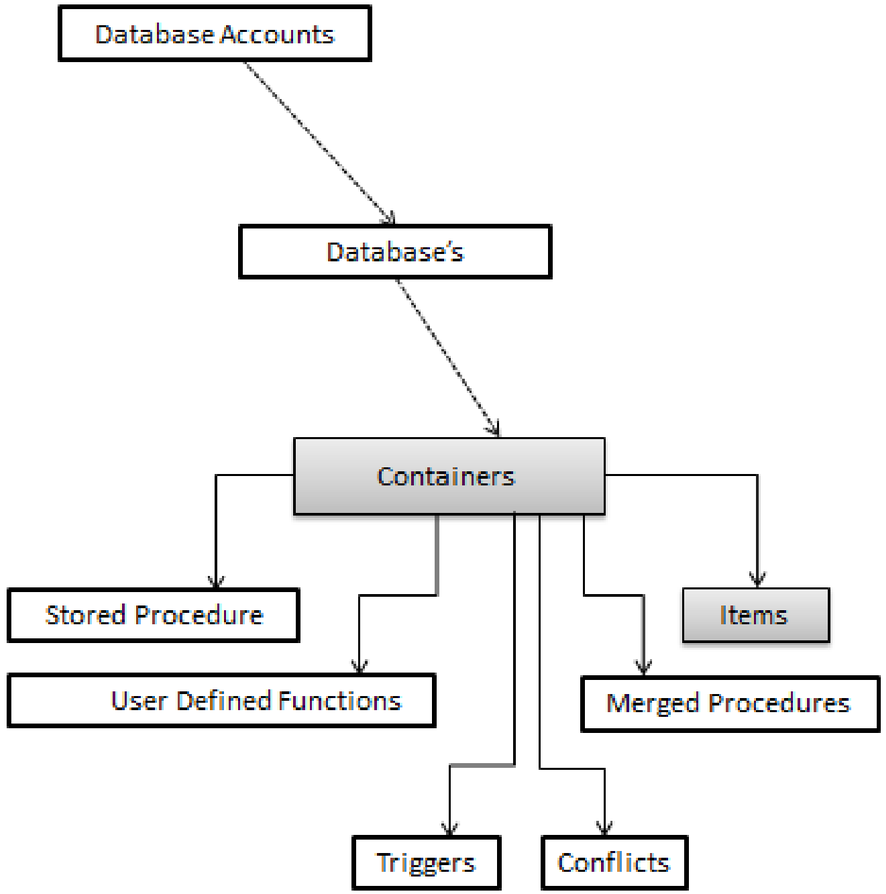
Partition in Azure Cosmos DB is mainly used to scale containers created in the above database to align with the performance of applications. Items in Partition are divided into Logical Partitions and choosing the right partition key is very important.



Managing Containers

Containers are mostly used for storage and throughput and can be created as below from the Cosmos DB Data Explorer with a unique container id and can be used to define stored procedures, functions, Items, Triggers etc.





Cosmos DB Hierarchy

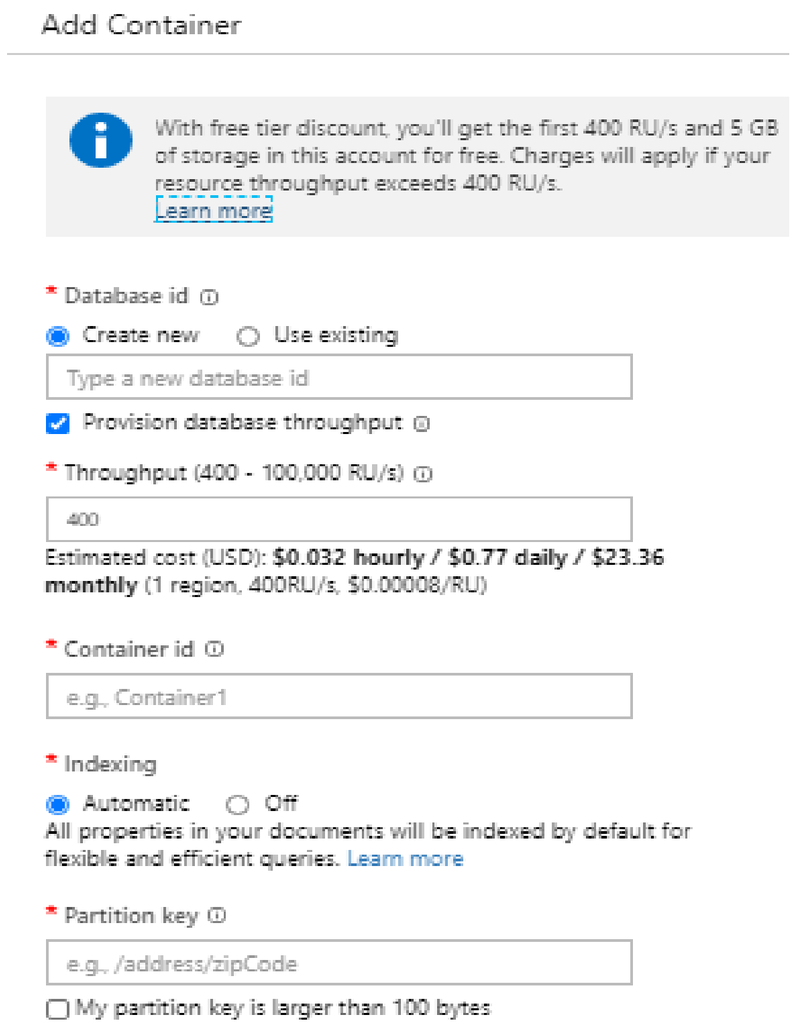
edureka!

Manage Partitions and Key

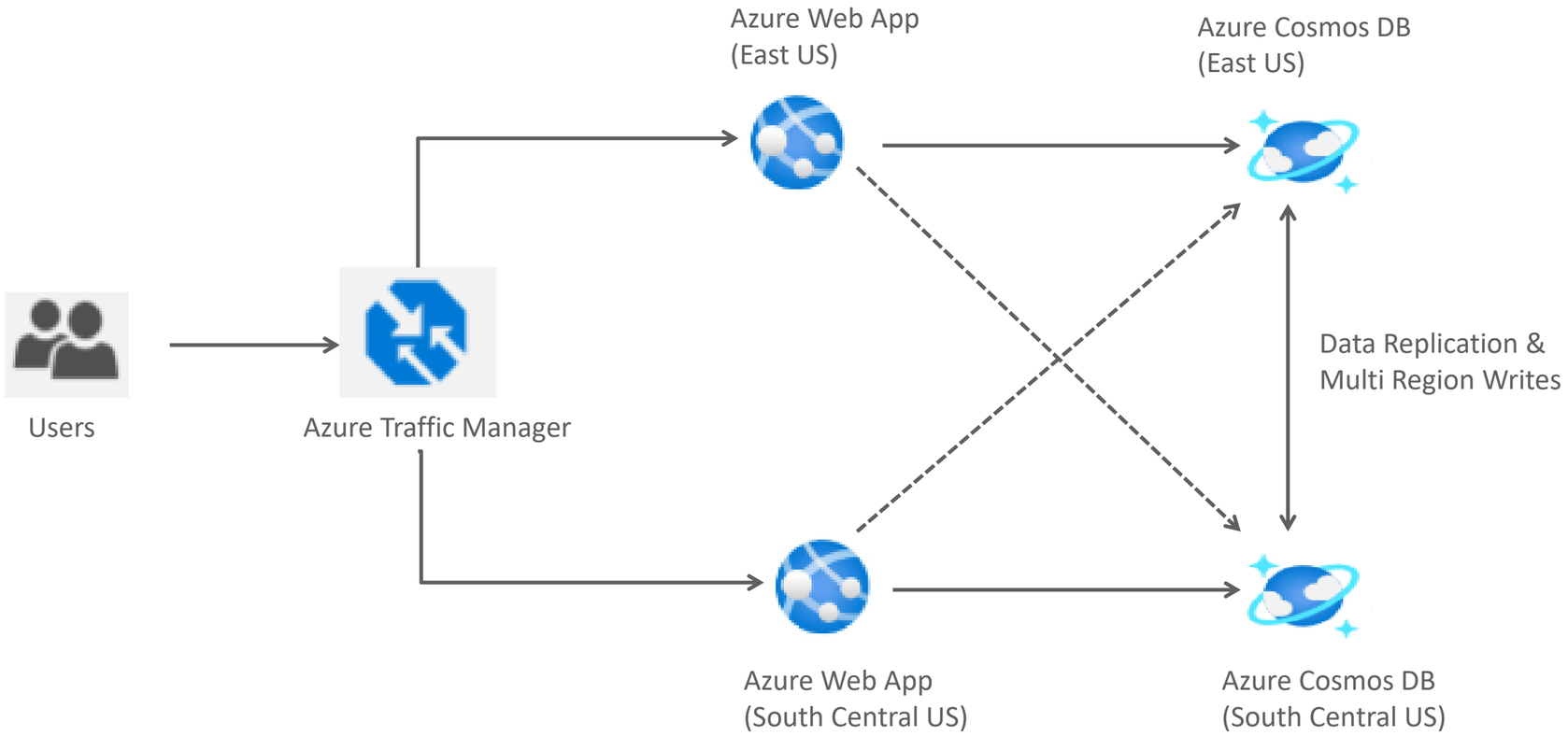
Partitions are used by Azure Cosmos DB to scale individual Containers that are created with a unique container id. Partitions are categorized into two – Logical and Physical Partitions.

The data and throughput are partitioned based on the partition key that has been specified.

Select a Right Partition Key: The Partition key is used to partition data among multiple servers for scalability automatically. Choose a JSON property name that has a wide range of values and is likely to have evenly distributed access patterns.



Cosmos DB Use Case – Multi Region Writes



Demo 3 – Implement Scaling Using Partitioning and Containers

edureka!

Implement Server-side Programming

edureka!

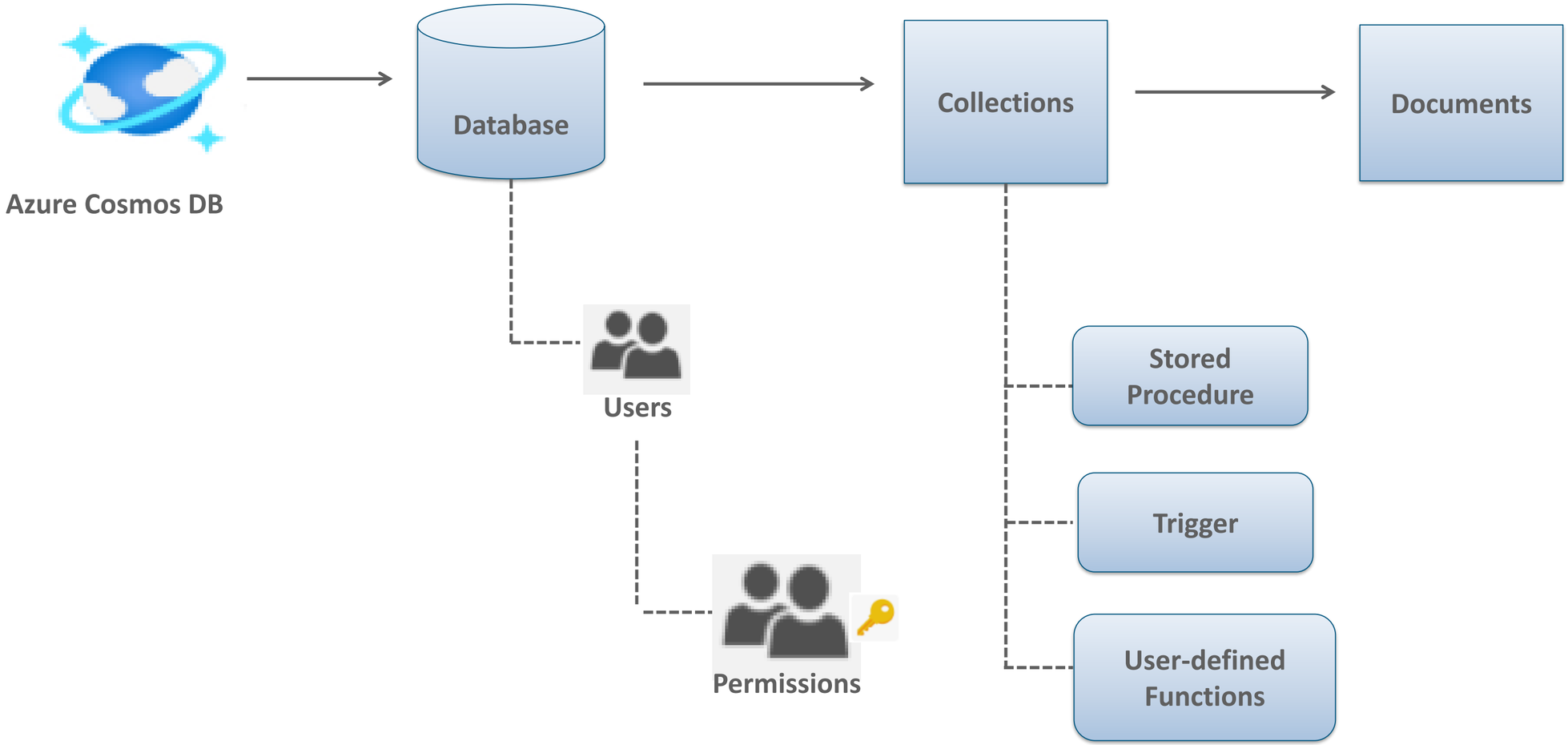
Server-side Programming in Azure Cosmos DB

Server-side program is the term given to various types of programming languages that run on the server directly. This program takes the user inputs, communicates with the databases, and renders the content to be displayed in client systems or the web browser.

- Azure Cosmos DB has inbuilt capabilities to perform server-side programming, and it uses pre-defined API like SQL API along with JavaScript capabilities that can write stored procedures, define triggers and user-defined functions, and can be executed directly from Azure Cosmos DB Engine
- Transactions can be performed using the API as a function in JavaScripts and can run complex operations with the dataset. Cosmos DB transactions are defined to meet all the ACID (Atomicity, Consistency, Isolation, and Durability) property

Note: We will discuss **Stored Procedures**, **Triggers,** and **User-defined Functions** in detail.

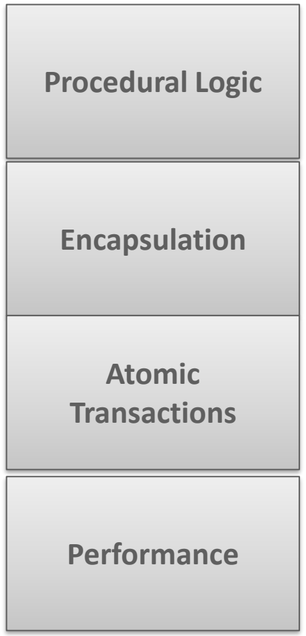
Understand Azure Cosmos DB Key Features



Benefits of Server-side Operations in Azure Cosmos DB

Cosmos DB server-side programs are defined in a JavaScript functions, which are finally executed by JavaScript engine running inside a cosmos database engine. All complex transactional operations can be executed with the help of a JavaScript engine.

Some key benefits of server-side operations in Cosmos DB include as below:



Procedural Logic:

Custom rules or business logic that include complex operations that can be performed using JavaScript.

Encapsulation:

Encapsulations add an extra layer of abstraction which is useful when the data is schema-less and you don't have to manage adding/updating extra business logic or rules.

Atomic Transactions:

Azure Cosmos DB guarantees that the transaction via stored procedure or trigger is complete and fulfills ACID property.

Performance:

On-demand Execution, Batching, Pre-Compilation are some performance benefits of using JavaScript Server-side Programs.

Important Terminologies

Stored Procedures

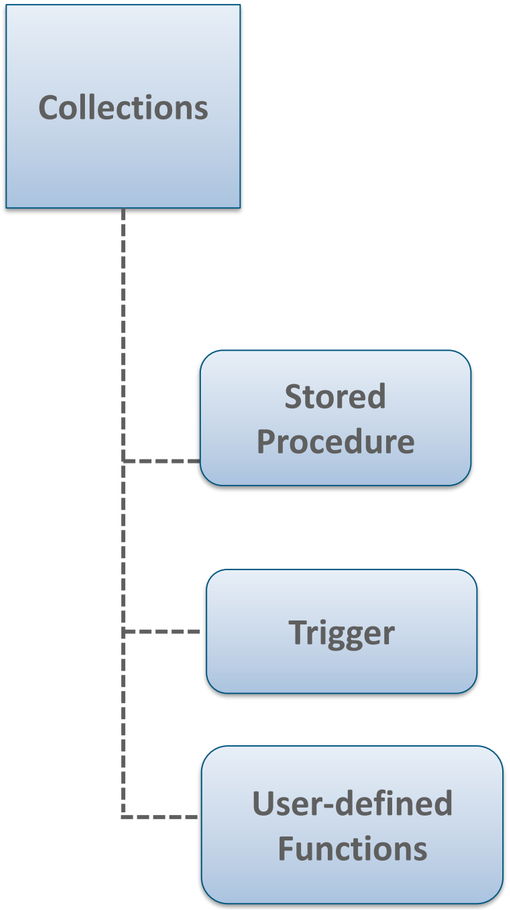
- All CRUD operations like INSERT, UPDATE, DELETE, DROP, etc. are defined in a stored procedure, and a query is executed to perform the above operations. Stored Procedures can be used to manipulate one or more documents in Azure Cosmos DB. This can be executed in Cosmos DB using SQL API.

Trigger

- Trigger is used as an action to be performed based on certain events and can be used to validate or modify the data when any document is added or modified by the user or apps. A trigger must be specified in Cosmos DB in a SQL API method.

User-defined Functions (UDF)

- UDF operates just like any other built-in SQL functions and always returns a JavaScript object. UDF is used to implement custom business logic and can be called within queries.



Steps to Create Stored Procedure Using Cosmos

Below are some of the steps :

- Open Microsoft Visual Studio IDE and Click on Create a new solution
- Navigate & Open NuGet Package Manager Console and install Microsoft.Azure.DocumentDB Package
- Add App.Config Settings with Cosmos DB URI & Primary Key
- Login to Azure portal -> Cosmos DB -> Note URI & Primary Key
- Right Click on Project and Add New Item -> JavaScript File named ‘sTestFile’
- Add JavaScript logic to get the context and response
- Create a new stored procedure definition and execute the stored procedure within the code logic

  - `client.ExecuteStoredProcedureAsync<string>`

- Test and Validate the stored procedure in the program until it is successful

Summary

Azure Storage Overview

Cloud storage solution for modern applications that rely on durability, availability, and scalability to meet the needs of their customers

Massively scalable, so you can store and process hundreds of terabytes of data

Elastic, so you can design and scale applications according to the amount of data stored and the number of requests made against it

Uses an auto-partitioning system that automatically load-balances your data based on traffic

Azure Storage

edureka!

Azure Table Storage

Azure Table Storage is a NoSQL key-value store for rapid development using massive semi-structured datasets

Azure Table Storage

It embraces a strong consistency model and is most opt for Enterprise-level data stores

edureka!

Authorizing Requests Using Shared Key

Shared Key authorization relies on your **account access keys** and other parameters to produce an ***encrypted signature string*** that is passed on the request in the *Authorization* header

Every request made against a storage service must be authorized, unless the request is for a blob or container resource that has been made available for public or signed access

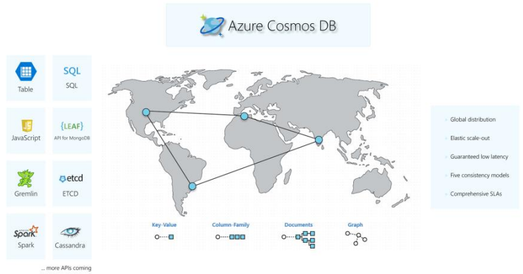
edureka!

Table Service REST API

- The Table service offers structured storage in the form of tables
- The Table service API is a REST API for working with tables and the data that they contain

edureka!

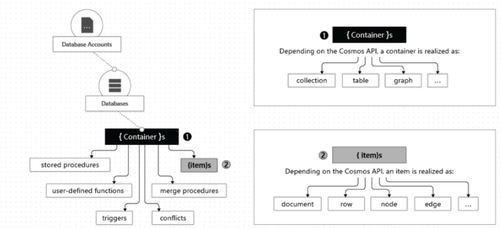
Azure Cosmos DB – Features And Support



edureka!

Databases, Containers And Items In Azure Cosmos DB

- After you create an ***Azure Cosmos DB account*** under your Azure subscription, you can manage data in your account by creating databases, containers, and items
- The following image shows the hierarchy of different entities in an Azure Cosmos DB account:



edureka!

edureka!

edureka!

edureka!

Thank You

For more information please visit our website
www.edureka.co