

Module 09: Design and implement a replication strategy for Azure Cosmos DB SQL API

In [1]:

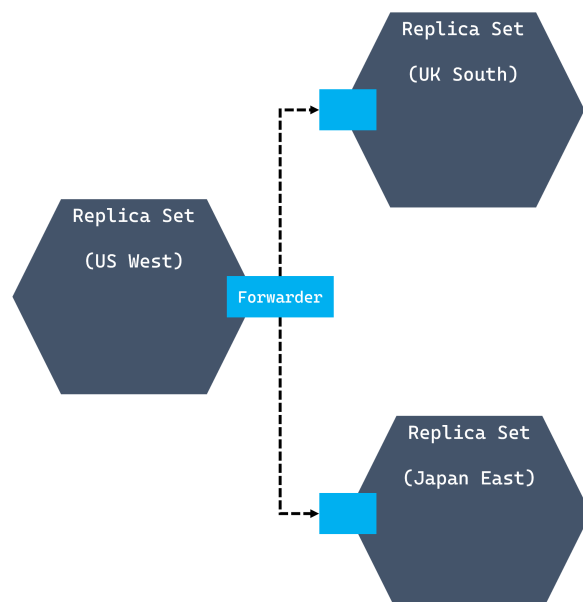
```
public class Product
{
    public string id { get; set; }
    public string categoryId { get; set; }
    public string categoryName { get; set; }
    public string sku { get; set; }
    public string name { get; set; }
    public string description { get; set; }
    public double price { get; set; }
}
```

Configure replication and manage failovers in Azure Cosmos DB

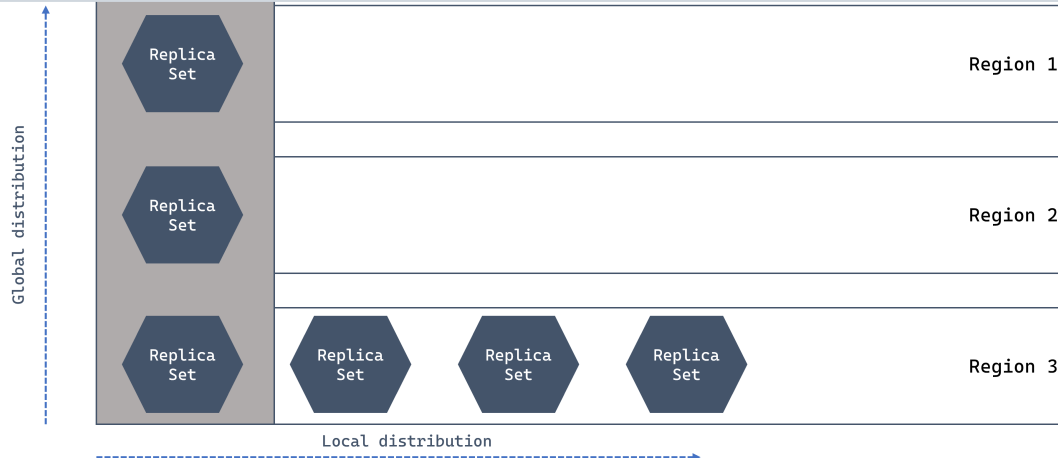
Understand replication

A replica set is a group of replicas that can dynamically grow and shrink to meet the needs of the database platform.

Each replica set will have other geographically distant replica sets that manage the same partition keys if data is distributed globally. These replica sets can then forward data to other replica sets in different regions to create replica copies of the data.



An Azure Cosmos DB account replicates data within a region (local distribution) among different replica sets servicing various partition key values. Replica sets that manage the same partition key value are referred to as a partition set as they will forward data between each other (global distribution).



Distribute data across regions

Configuring global distribution in Azure Cosmos DB is a turnkey operation that is performed when an account is created or afterward.

Configuring geo-redundancy for a new account:

[Home](#) > [Create a resource](#) > [Select API option](#) >

Create Azure Cosmos DB Account - Azure Cosmos DB for NoSQL ...

Basics Global Distribution Networking Backup Policy Encryption Tags Review + create

Global Distribution

Configure global distribution and regional settings for your account. You can also change these settings after the account is created.

Geo-Redundancy ⓘ ☐ Enable ☒ Disable

Multi-region Writes ⓘ ☐ Enable ☒ Disable

Availability Zones ⓘ ☐ Enable ☒ Disable

Configuring geo-redundancy for an existing account:

oisdaufoisadoi | Replicate data globally ...

Azure Cosmos DB account

Search (Ctrl+/) << Save Discard Manual Failover Automatic Failover

Overview Activity log Access control (IAM) Tags Diagnose and solve problems Cost Management Quick start Notifications Data Explorer

Settings

Features

Replicate data globally

Default consistency Backup & Restore

Click on a location to add or remove regions from your Azure Cosmos DB account.
* Each region is billable based on the throughput and storage for the account. [Learn more](#)

Configure regions
Multi-region writes ⓘ
☒ Disable ☐ Enable

Configure the regions for reads, \ be configured when a new region

Write Region

North Europe

Read Regions

Japan East

Australia East

East US 2

South Africa North

The cost of distributing data globally is **RU/s x # of regions**.

For example, consider a solution that uses approximately 1,000 RU/s per hour; data is written to one Azure region and replicated to five more regions. The formula for this would be:

$1,000 \times (1+5) = 6,000$

The account would be billed for **6,000 RU/s** used at a per-hour rate.

Azure Cosmos DB failovers

An automatic failover plan can transfer the write region to one of the read regions in the case of an outage.

Define automatic failover policies:


Automatic Failover ...

Enable Automatic Failover ⓘ

ON

OFF

Drag-and-drop read regions items to reorder the failover priorities.

Tip: Drag  on the left of the hovered row to reorder the list.

Write Region

North Europe

Read Regions	Priorities
Japan East	1
East US 2	2
South Africa North	3
Australia East	4

Perform manual failovers:

Manual Failover ...

Select a Read Region to become the new Write Region.

Tip: Identify all dependent services leveraging this account and ensure that triggering a failover will not jeopardize your production application.

Write Region

North Europe

Read Regions

Japan East

East US 2

South Africa North

Australia East



I understand and agree to trigger a failover on my current Write Region.

Configure SDK region

Use the ApplicationRegion or ApplicationPreferredRegions properties to configure preferred regions.

```
In [ ]: // Setting an application region

using Microsoft.Azure.Cosmos;

CosmosClientOptions options = new () { ApplicationRegion = Regions.UKSouth };
CosmosClient client = new (connectionString, options);
```

```
In [ ]: // Or using the CosmosClientBuilder class

using Microsoft.Azure.Cosmos.Fluent;

CosmosClient client = new CosmosClientBuilder(connectionString)
    .WithApplicationRegion(Regions.UKSouth)
    .Build();
```

```
In [ ]: // Setting a List of preferred application regions

List<string> regions = new() { "East Asia", "South Africa North", "West US" };
CosmosClientOptions options = new () { ApplicationPreferredRegions = regions };
CosmosClient client = new (connectionString, options);
```

```
In [ ]: // Or using the CosmosClientBuilder class

CosmosClient client = new CosmosClientBuilder(connectionString)
    .WithApplicationPreferredRegions( new List<string>
    {
        Regions.EastAsia,
        Regions.SouthAfricaNorth,
        Regions.WestUS
    } ) .Build();
```

Use consistency models in Azure Cosmos DB SQL API

Understand consistency models

In a distributed database system, tradeoffs are often made between highly consistent data with extended latency and speedy data operations that may not be consistent immediately.



Each of the five consistency levels is well-defined with clear tradeoffs when compared with each other:

Consistency Level	Description
Strong	Linear consistency. Data is replicated and committed in all configured regions before acknowledged as committed and visible to all clients.
Bounded Staleness	Reads lag behind writes by a configured threshold in time or items.
Session	Within a specific session (SDK instance), users can read their own writes.
Consistent Prefix	Reads may lag behind writes, but reads will never appear out of order.
Eventual	Reads will eventually be consistent with writes.

Configure default consistency model in the portal

In the Azure portal, the Default consistency pane is used to configure a new default consistency level for the entire account.

The screenshot shows the Azure portal interface for configuring the default consistency level of an Azure Cosmos DB account. The account name is 'oisdaufaisadoi'. The 'Default consistency' pane is active, showing the 'Session' consistency level selected. A sidebar on the left contains navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Cost Management, Quick start, Notifications, Data Explorer, Settings, Features, Replicate data globally, Backup & Restore, Firewall and virtual networks, Private Endpoint Connections, and CORS. The main content area displays a world map and a diagram illustrating session consistency across different regions (North Europe, West Europe, North Central US) with colored dots representing data points.

Change consistency model with the SDK

Using the **ItemRequestOptions** class, you can relax the current default consistency level to a weaker one.

In []:

```
// Set the item request Consistency Level option
ItemRequestOptions options = new()
{
    ConsistencyLevel = ConsistencyLevel.Eventual
};

// Assign the option to the create item operation
var item = new Product
{
```

```

id = $"{Guid.NewGuid()}",
categoryId = $"{Guid.NewGuid()}",
name = "Reflective Handlebar"
};

CosmosClient client = new (connectionString);
Database database = await client.CreateDatabaseIfNotExistsAsync("cosmicworks");
Container container = await database.CreateContainerAsync("cosmicworks", "/categoryId".

await container.CreateItemAsync<Product>(item, requestOptions: options);

```

Use session tokens

Using the .NET SDK classes, the session token can be manually extracted and passed back to the Azure Cosmos DB resource.

In []:

```

// Create an item with an Item Response
var id = $"{Guid.NewGuid()}";
var categoryId = $"{Guid.NewGuid()}";
var item = new Product
{
    id = id,
    categoryId = categoryId,
    name = "Reflective Handlebar 2"
};
ItemResponse<Product> response = await container.CreateItemAsync<Product>(item);

// Get the session token from the item response
string token = response.Headers.Session;

// Set the item request option session token to the previous token
ItemRequestOptions options = new()
{
    SessionToken = token
};

// Use the token on the new request
ItemResponse<Product> readResponse = await container.ReadItemAsync<Product>(
    id, new PartitionKey(categoryId), requestOptions: options);

Console.WriteLine($"Session token: {token}");

```

Configure multi-region write in Azure Cosmos DB SQL API

Understand multi-region write

With Azure Cosmos DB, every region supports both writes and reads.

oisdaufoisadoi | Replicate data globally ...
Azure Cosmos DB account

Search (Ctrl+/) Save Discard Manual Failover Automatic Failover

Overview Activity log Access control (IAM) Tags Diagnose and solve problems Cost Management Quick start Notifications Data Explorer

Settings Features Replicate data globally Default consistency Backup & Restore

Click on a location to add or remove regions from your Azure Cosmos DB account.
* Each region is billable based on the throughput and storage for the account. [Learn more](#)

Configure regions
Multi-region writes

Disable **Enable**

Configure the regions for reads, writes and availability zone (supported in configured when a new region is added). [+ Add region](#)

Regions	Reads Enabled	Writes Enable
North Europe	✓	✓
Japan East	✓	✓
Australia East	✓	✓
East US 2	✓	✓
South Africa North	✓	✓

Understand conflict resolution policies

Azure Cosmos DB's multi-region write feature has automatic conflict management built in. The default policy is known as Last Write Wins that uses the `_ts` property by default.

Replace the default `_ts` property configuring any numeric property as a conflict resolution path on the .NET SDK