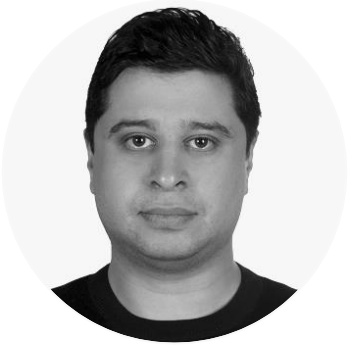
Working with Azure Cosmos DB - Table API



Reza Salehi
CLOUD CONSULTANT

@zaalion linkedin.com/in/rezasalehi2008

Overview

Why would you migrate to the Cosmos DB Table API?

Multiple methods to query your data

Migrating your data from Azure Table Storage

Demo: Migrating to Cosmos DB Table API

The Azure Cosmos DB Table API is a premium offering for table storage.

A Premium Offering

Storage Accounts / Table Storage

Table Storage

A NoSQL key-value store for rapid development using massive semi-structured datasets

- ✔ Store semi-structured data that's highly available
- ✔ Create massively-scalable apps
- ✔ Create apps that require a flexible data schema
- ✔ Use JSON to serialize data
- ✔ Perform OData-based queries

Get started with Table storage >

Create your free Azure account today >

Use Azure Cosmos DB's support for Tables API to take advantage of global distribution, automatic indexing and rich query, dedicated throughput, and single digit millisecond latencies. >

A Premium Offering

- Turnkey global distribution
- Dedicated throughput
- Single-digit ms latencies
- Guaranteed high availability
- Automatic secondary indexing

Table API Offerings

	Azure Table storage	Azure Cosmos DB Table API
Latency	Fast, but no upper bounds on latency.	Single-digit millisecond latency for reads and writes, backed with <10 ms latency for reads and writes at the 99th percentile, at any scale, anywhere in the world.
Throughput	Variable throughput model. Tables have a scalability limit of 20,000 operations/s.	Highly scalable with dedicated reserved throughput per table that's backed by SLAs. Accounts have no upper limit on throughput and support >10 million operations/s per table.
Global distribution	Single region with one optional readable secondary read region for high availability. You can't initiate failover.	Turnkey global distribution from one to any number of regions. Support for automatic and manual failovers at any time, anywhere in the world. Multi-master capability to let any region accept write operations.

Table API Offerings

	Azure Table storage	Azure Cosmos DB Table API
Indexing	Only primary index on PartitionKey and RowKey. No secondary indexes.	Automatic and complete indexing on all properties by default, with no index management.
Query	Query execution uses index for primary key, and scans otherwise.	Queries can take advantage of automatic indexing on properties for fast query times.
Consistency	Strong within primary region. Eventual within secondary region.	Five well-defined consistency levels to trade off availability, latency, throughput, and consistency based on your application needs.
Pricing	Storage-optimized.	Throughput-optimized.
SLAs	99.9% to 99.99% availability, depending on the replication strategy.	99.999% read availability, 99.99% write availability on a single-region account and 99.999% write availability on multi-region accounts. Comprehensive SLAs covering availability, latency, throughput and consistency.

Containers in Each API

Azure Cosmos API	Specialized Entity
SQL API	Collection
Table API	**Table**
MongoDB API	Collection
Cassandra API	Table
Gremlin API	Graph

Query Azure Cosmos DB Using the Table API

Query on PartitionKey and RowKey

Query by using an OData filter

Query by using LINQ (.NET SDK)

```
https://<table-endpoint>/People(PartitionKey='Boston',RowKey='1000')
```

Query on PartitionKey and Rowkey

The PartitionKey and RowKey properties form an entity's primary key, you can use them to identify the entity

```
https://<endpoint>/People()?$filter=PartitionKey%20eq%20’Boston'%20and%20Email
%20eq%20’Reza@test.com'
```

Query by Using an OData Filter

Query by Using an OData Filter

- Use the logical operators defined by the OData Protocol Specification to compare a property to a value
- The property name, operator, and constant value must be separated by URL-encoded spaces (%20)
- All parts of the filter string are case-sensitive
- The constant value used in filters must be of the same data type as the property for the query to return a valid result

Query by Using LINQ (.NET SDK)

```
C#

CloudTableClient tableClient = account.CreateCloudTableClient();
CloudTable table = tableClient.GetTableReference("People");

TableQuery<CustomerEntity> query = new TableQuery<CustomerEntity>()
    .Where(
        TableQuery.CombineFilters(
            TableQuery.GenerateFilterCondition("PartitionKey", QueryComparisons.Equal, "Smith"),
            TableOperators.And,
            TableQuery.GenerateFilterCondition("Email", QueryComparisons.Equal,"Ben@contoso.com")
    ));

await table.ExecuteQuerySegmentedAsync<CustomerEntity>(query, null);
```

.NET Framework SDK

Using the **Microsoft.Azure.Cosmos.Table** library. The Microsoft.Azure.CosmosDB.Table will be deprecated.

Provisioning a Cosmos DB Table API Instance

Create Azure Cosmos DB Account

Try Cosmos DB for free, up to 20K RU/s, for 30 days with unlimited renewals.

Basics | Network | Tags | Review + create

Azure Cosmos DB is a globally distributed, multi-model, fully managed database service. Try it for free, for 30 days with unlimited renewals. Go to production starting at $24/month per database, multiple containers included. Learn more

Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

* Subscription	Pay-As-You-Go
* Resource Group	Select existing...
	Create new

Instance Details

* Account Name	Enter account name
* API	Azure Table
Jupyter Notebooks or Apache Spark	Sign up for Jupyter Notebooks or Apache Spark preview
* Location	(US) East US
Geo-Redundancy	Enable / Disable
Multi-region Writes	Enable / Disable

Use Data Migration Tool to migrate your data to Azure Cosmos DB.

AzCopy can also be used
to migrate to
Azure Cosmos DB.

Demo

Provisioning a Cosmos DB Table API instance

Using Data Migration Tool to import an Azure Table Storage table to Cosmos DB

Summary

Benefits of migrating to the Cosmos DB Table API

There are multiple methods to query your data

Cosmos DB Table API .NET SDK

Migrating your data from Azure Table Storage

Demo: Migrating to Cosmos DB Table API