Using the Table API for a Key-value Data Model



Leonard Lobel
CTO, SLEEK TECHNOLOGIES
lennilobel.wordpress.com

What Is the Table API?

Replaces
Azure Table storage

Implement a key-value data model

Key-value data model

Key = PartitionKey + RowKey

Value = key-value pairs!

Leverage Cosmos DB back end

Predictable throughput
Global distribution
Automatic indexing



Cosmos DB SQL API Table API



Cosmos DB	SQL API	Table API
Container	Collection	Table



Cosmos DB	SQL API	Table API
Container	Collection	Table
Item	Document	Row

```
Document (SQL API)
{
    "genre": "sci-fi",
    "id": "Star Trek II",
    "year": 1982,
    "length": "1h, 53m",
    "description": "Khan is back!"
}
```

Row (Table API)

PartitionKey	RowKey	Year	Length	Description
sci-fi	Star Trek II	1982	1h, 53m	Khan is back!



Cosmos DB	SQL API	Table API
Container	Collection	Table
Item	Document	Row
Partition Key	Any property	PartitionKey



Cosmos DB	SQL API	Table API
Container	Collection	Table
Item	Document	Row
Partition Key	Any property	PartitionKey
ID	id	RowKey



Why Use the Table API?

Migrate existing applications

Just change the connection string

Upgrade SDK

Cosmos DB Table SDK uses native protocol

No advantage for new applications

Table API is a layer over SQL API



Demo



Simple Azure Table Storage application



Demo



Migrating from Azure Table Storage to Cosmos DB



Demo



Upgrading to the Cosmos DB Table SDK



Summary



Table API

- Key-value store

Replaces Azure Table Storage

- Unpredictable throughput
- Limited geo-replication
- No automatic indexing

Migrating from Azure Table Storage

- Use azcopy, or the data migration tool
- Change the connection string
- Upgrade the SDK for the Table API

Not recommended for new applications

- SQL API is the way to go

