Microsoft

AZ-203.3 Module 01: Develop solutions that use Azure Table storage

Prashanth Kumar

Microsoft

Topics

- Azure Table storage
- Authorization in Azure Storage
- Table service REST API

Lesson 01: Azure Table storage

Table storage in Azure

- NoSQL data in Azure Storage
  - Schemaless design
  - Flexible data structures as your application evolves
  - Take advantage of the scale of Azure Storage

Tables hierarchy

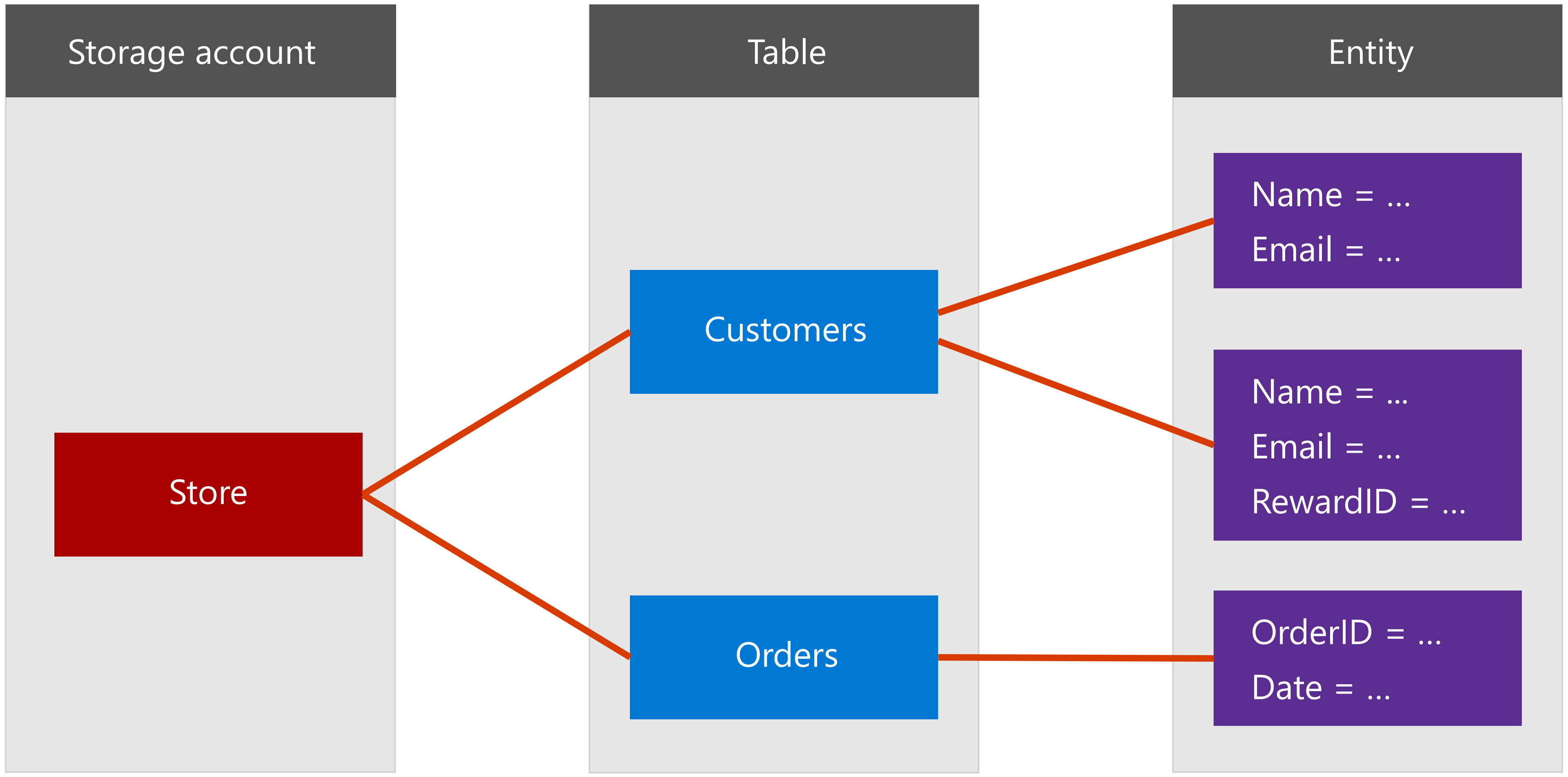
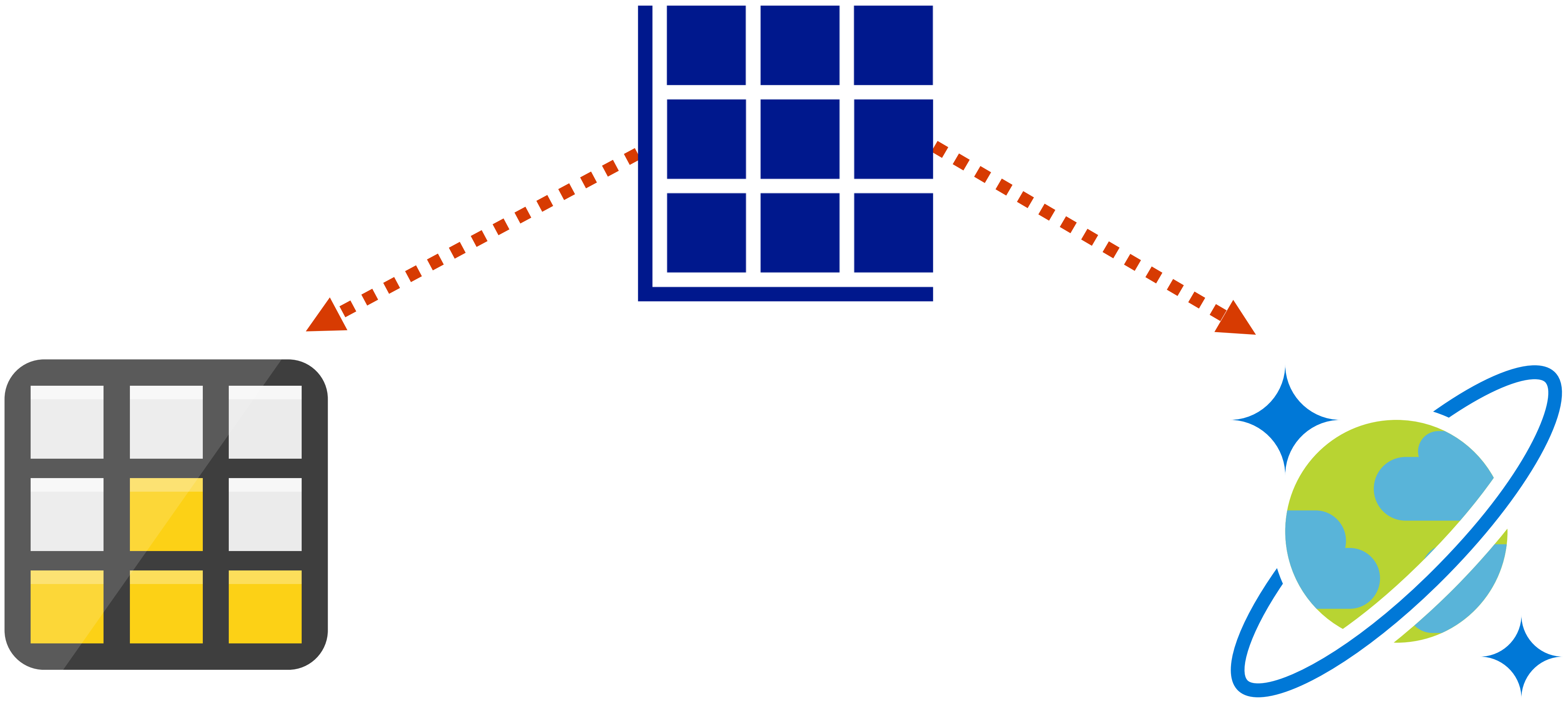


Table storage structure

Row key	Partition key	First name	Last name	Grade
aschmid2957389	teems_elementary	None	Schmid	5
tbright2874395	teems_elementary	Tina	None	v
gpeeler3458738	macon_middle	Gregory	Peeler	8

Storage data choices in Azure



Choosing Azure Storage or Cosmos DB

Attribute	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 reads and <15-ms latency 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 30+ regions. Support for automatic and manual failovers at any time, anywhere in the world.

Choosing Azure Storage or Cosmos DB (cont.)

Attribute	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, 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.

Read-efficient table design

- Design for querying
  - Consider query priorities when designing partition and row keys
- Specify partition and row keys in queries
  - Avoid table scans in your queries
  - Avoid cross-partition queries
- Denormalize data
  - Storage can be cost optimized, making multiple copies of data ideal sometimes
- Use compound keys
- Use query projection
  - Reduce the transferred amount of data

Write-efficient table design

- Avoid hot partitions
  - Keys should spread requests across partitions
- Avoid traffic spikes
  - Smooth traffic over time
- Don’t create a separate table per entity
  - Atomic transactions across entity types are more efficient in a single table
- Consider maximum throughput
  - Be aware of scalability targets for Azure Storage

Normalization vs. de-normalization

Normalized: Optimized for writes over reads

```
{
  "id": "08259",
  "pilot": [{ "name": "Hailey Nelson" }]
},
{
  "id": "08259",
  "ticketPrice": 255.00, "flightCode":
"3754"
},
{
  "id": "08259",
  "origin": { "airport": "SEA", "gate":
"A13" },
  "destination": { "airport": "JFK",
"gate": "D4" }
}
```

De-normalized: Optimized for reads over writes

```
{
"id": "08259",
  "ticketPrice": 255.00,
  "flightCode": "3754",
  "origin": {
    "airport": "SEA", "gate": "A13"
  },
  "destination": {
    "airport": "JFK", "gate": "D4"
  },
  "pilot": [{
    "name": "Hailey Nelson"
  }]
}
```

Lesson 02: Authorization in Azure Storage

Container permissions

- There are three levels of container access that are available:
  - **Full public read access:**
    - Enumerate container blobs
    - Read individual blobs
    - Cannot enumerate containers
  - **Public read access for blobs only:**
    - Read individual blobs
  - **No public read access:**
    - No access to blobs, containers, or enumerating contents

Authorization

- Every request must be authorized
  - Exception - blob or container resources that have been made publicly available (opt-in)
- REST API requests can use Shared Key authorization scheme
  - Requires two headers
    - Date (or x-ms-date)
    - Authorization

Shared Access Signatures

- A Shared Access Signature (SAS Token) is a URI that grants access to a protected container, blob, queue, or table for a specific time interval
  - Allows client application to access a resource without using the storage account key
  - Should only be used with secure (HTTPS) requests
  - Can be generated with the following components:
    - Start Time
    - Expiry Time
    - Permission Levels (Read, Write, Delete, List, None)

Stored Access Policies

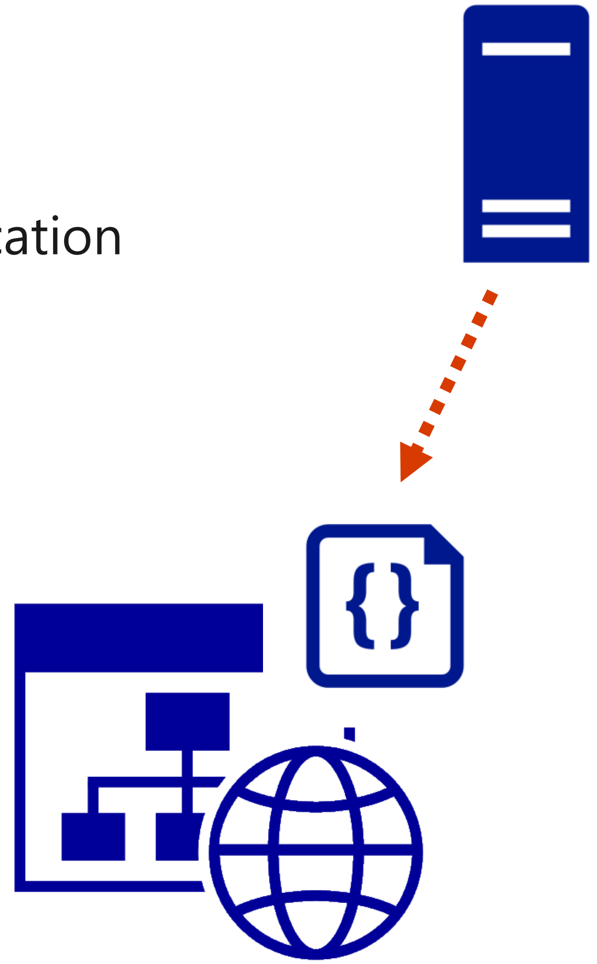
- Granular control over a set of shared access signatures
  - Signature lifetime and permissions are stored in the policy as opposed to the URL
- Container, Queue, or Table can have up to five stored access policies

Establishing a stored access policy

- Policy that can generate short-lifetime signatures to access resources
  - Signatures are concatenated to the end of the resource URI
  - Signatures are verified on the server for validity
- Signatures generated from a single policy share characteristics:
  - Permission (read, write, read-write, delete)
  - Start time
  - Expiry time
  - Resource scope (blob, table, etc.)
- All signatures generated by a single policy can be revoked as a group

CORS support for the Azure Storage services

- CORS is an HTTP feature that enables requests from one domain to another
  - This is mostly required to issue API calls from a JavaScript application
- Azure Storage supports enabling CORS at the service level
  - Can be scoped to specific domains and specific permissions
  - Can be scoped to storage services
    - Blob
    - File
    - Queue
    - Table



Lesson 03: Table service REST API

Table service resources

- http://account.table.core.windows.net/table(<partition key>,<row key>)

Resource	Name or value
Storage account	crmdata
Table	customerprofiles
Partition key	centraleurope
Row key	contoso

- http://crmdata.table.core.windows.net/customerprofiles
- http://crmdata.table.core.windows.net/customerprofiles(centraleurope, contoso)

Table service resources API

https://[account].table.core.windows.net/[table]

Method	Endpoint
GET	https://[account].table.core.windows.net/table
PUT	https://[account].table.core.windows.net/table
POST	https://[account].table.core.windows.net/[table]
DELETE	https://[account].table.core.windows.net/table
MERGE	https://[account].table.core.windows.net/table

Table services modifying resources

Partition key	middleschoolstudents
Row key	237548902
First name	Emilia
Last name	McCarty
Age	11

PUT: https://cornfielddistrict.table.core.windows.net/students(appleorchardmiddle, 237548902)

```
{
        "Age": 12,
        "Sport": "Tennis"
}
```

Partition key	middleschoolstudents
Row key	237548902
First name	Emilia
Last name	McCarty
Age	12
Sport	Tennis

Table services resource queries using OData

https://[account].table.core.windows.net/table

Operation	OData Query
Get an entity by a partition and row key	[GET] /table
Query a table for entities that match an expression	[GET] /table?$filter=[query expression]
Delete an entity	[DELETE] /table
Insert or replace an entity	[PUT] /table

Table services resource queries using OData (cont.)

Example 1: Retrieve entity

https://[account].table.core.windows.net/[table]([partitionKey], [rowKey])

Example 2: Filtering

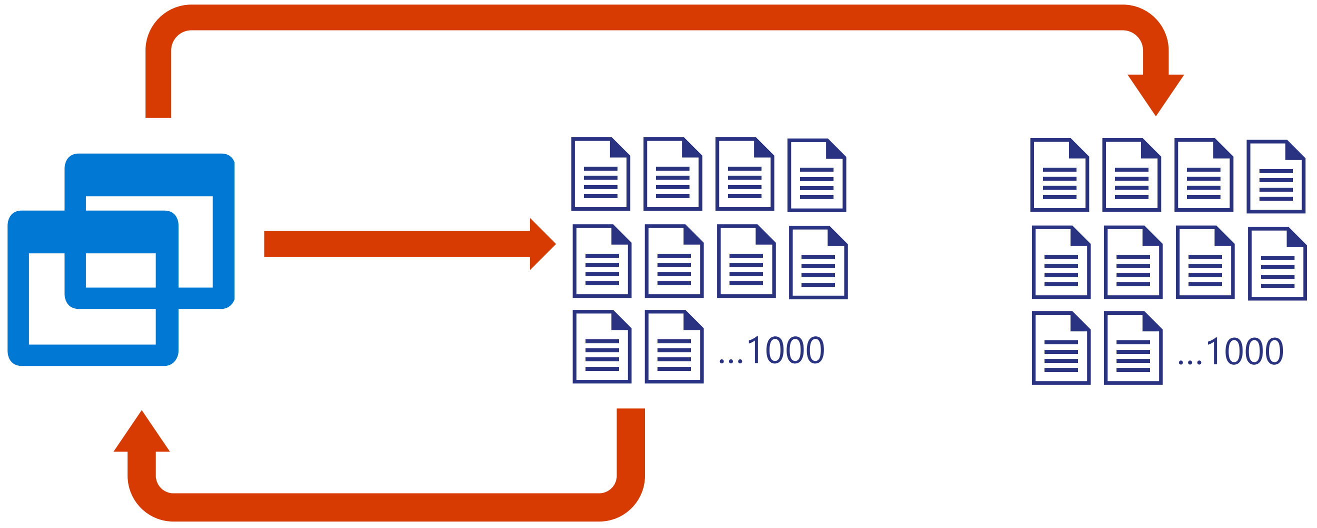
https://[account].table.core.windows.net/[table]()?filter=Grade ge 3

Example 3: Projection

https://[account].table.core.windows.net/[table]()?select=FirstName,LastName

Query time out and pagination

- Queries for tables or entities can return a maximum of 1,000 items at a time
  - If you need more, you will need to reissue the query with a pointer indicating where to start the next batch of results
- Table service provides headers to reissue queries



Query time out and pagination headers

Continuation token header	Description
x-ms-continuation-NextTableName	This header is returned in the context of a Query Tables operation. If the list of tables returned is not complete, a hash of the name of the next table in the list is included in the continuation token header.
x-ms-continuation-NextPartitionKey	This header is returned in the context of a Query Entities operation. The header contains a hash of the next partition key to be returned in a subsequent query against the table.
x-ms-continuation-NextRowKey	This header is returned in the context of a Query Entities operation. The header contains a hash of the next row key to be returned in a subsequent query against the table. Note that in some instances, x-ms-continuation-NextRowKey may be null.

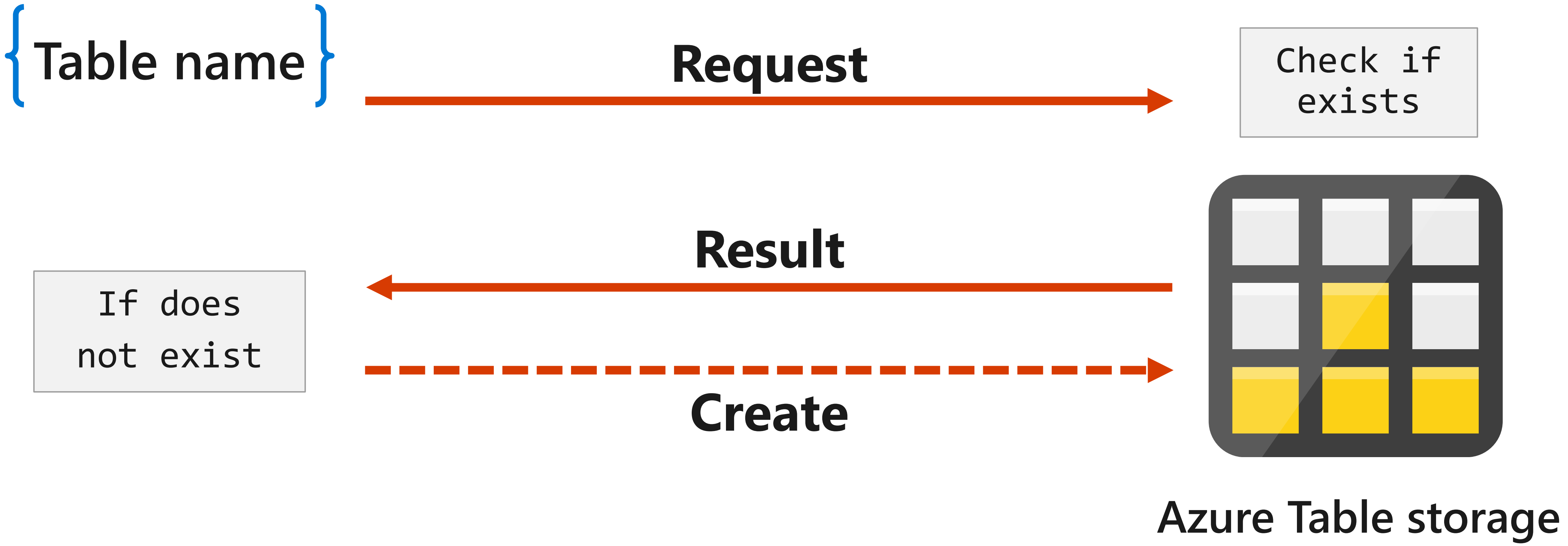
Accessing Storage tables with the .NET SDK

```
var client = account.CreateCloudTableClient();

var table = client.GetTableReference("people");

await table.CreateIfNotExistsAsync();
```

CreateIfNotExists() member



Implementing TableEntity class

```csharp
public class CustomerEntity : TableEntity
{
    public CustomerEntity(string lastName, string firstName)
    {
        this.PartitionKey = lastName;
        this.RowKey = firstName;
    }

    public CustomerEntity() { }
    public string Email { get; set; }
    public string PhoneNumber { get; set; }
}
```

Querying a table

```
var condition = TableQuery.GenerateFilterCondition(
    "PartitionKey", QueryComparisons.Equal, "Doe"
);
var query = new TableQuery<PersonEntity>().Where(condition);

foreach (CustomerEntity entity in table.ExecuteQuery(query))
{
    Console.WriteLine($"{entity.RowKey} {entity.PartitionKey} [Age: 
{entity.Age} | IsRetired: {entity.IsRetired} | Hometown: {entity.Hometown}]");
}
```

Inserting an entity

```csharp
CustomerEntity customer1 = new CustomerEntity("Haynes", "Jodie");
customer1.Email = "jodie@contoso.com";
customer1.PhoneNumber = "425-555-0101";

TableOperation insertOperation = TableOperation.Insert(customer1);

table.Execute(insertOperation);
```

Demo: Managing Azure Table storage by using .NET

Microsoft

Review

- Azure Table Storage
- Authorization in Azure Storage
- Table service REST API

Microsoft