

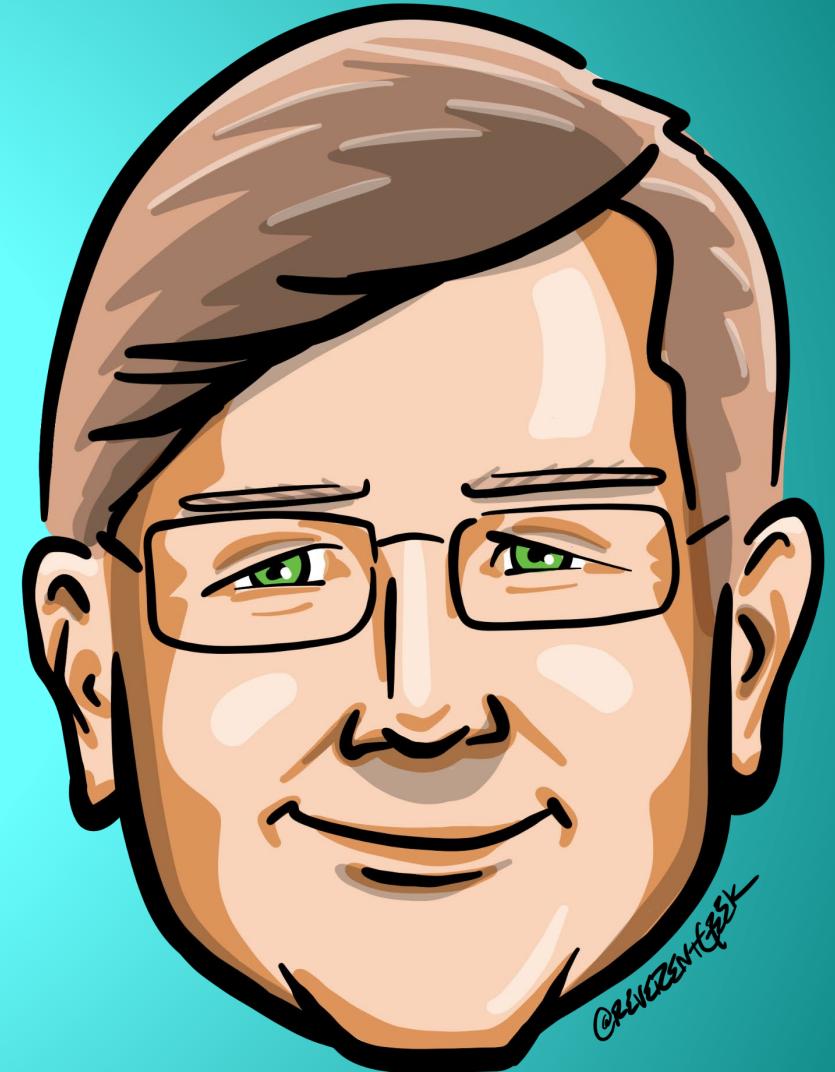
THE HITCHHIKER'S GUIDE TO THE COSMOS

Chad Green



Who is Chad Green

- ✉️ chadgreen@chadgreen.com
- .twitch TaleLearnCode
- 🌐 TaleLearnCode.com
- 🐦 ChadGreen & TaleLearnCode
- linkedin ChadwickEGreen





What is Cosmos DB

The Hitchhiker's Guide to the Cosmos

Azure Cosmos DB

A globally distributed, massively scalable, multi-model database service

Turnkey global distribution



Azure Cosmos DB

A globally distributed, massively scalable, multi-model database service

Turnkey global distribution



Azure Cosmos DB

A globally distributed, massively scalable, multi-model database service

Elastic scale out of
storage & throughput

Turnkey global distribution



Azure Cosmos DB

A globally distributed, massively scalable, multi-model database service

Elastic scale out of
storage & throughput

Turnkey global distribution



Azure Cosmos DB

A globally distributed, massively scalable, multi-model database service

**Guaranteed low latency
at the 99th percentile**

Turnkey global distribution

Elastic scale out
of storage & throughput



Azure Cosmos DB

A globally distributed, massively scalable, multi-model database service

**Guaranteed low latency
at the 99th percentile**

Turnkey global distribution

Elastic scale out
of storage & throughput



Azure Cosmos DB

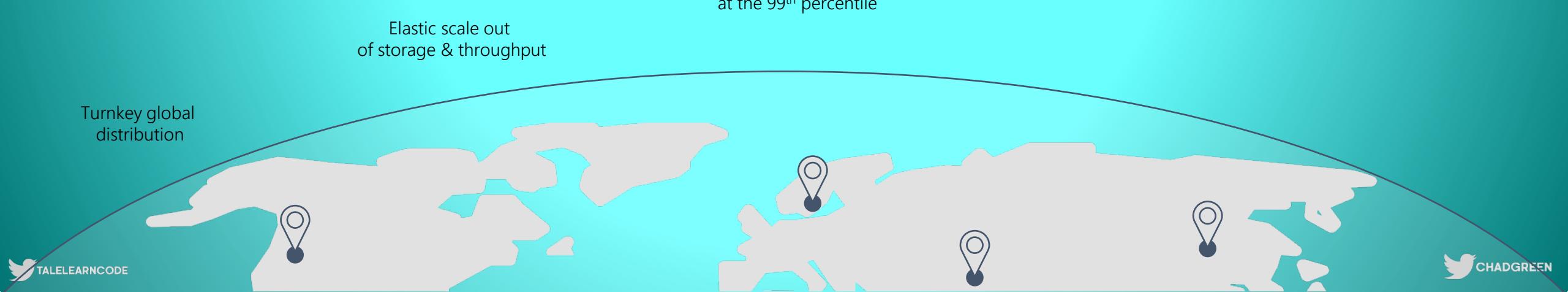
A globally distributed, massively scalable, multi-model database service

Five well-defined consistency models

Turnkey global distribution

Elastic scale out
of storage & throughput

Guaranteed low latency
at the 99th percentile



Azure Cosmos DB

A globally distributed, massively scalable, multi-model database service

Strong

Turnkey global distribution

Elastic scale out
of storage & throughput

Guaranteed low latency
at the 99th percentile

Azure Cosmos DB

A globally distributed, massively scalable, multi-model database service

Strong

Turnkey global distribution

Elastic scale out
of storage & throughput

Guaranteed low latency
at the 99th percentile

Azure Cosmos DB

A globally distributed, massively scalable, multi-model database service

Strong

Bounded Staleness

Turnkey global distribution

Elastic scale out
of storage & throughput

Guaranteed low latency
at the 99th percentile

Azure Cosmos DB

A globally distributed, massively scalable, multi-model database service

Strong

Bounded Staleness

Turnkey global distribution

Elastic scale out
of storage & throughput

Guaranteed low latency
at the 99th percentile

Azure Cosmos DB

A globally distributed, massively scalable, multi-model database service

Strong

Session

Bounded Staleness

Turnkey global distribution

Elastic scale out
of storage & throughput

Guaranteed low latency
at the 99th percentile

Azure Cosmos DB

A globally distributed, massively scalable, multi-model database service

Strong

Session

Bounded Staleness

Turnkey global distribution

Elastic scale out
of storage & throughput

Guaranteed low latency
at the 99th percentile

Azure Cosmos DB

A globally distributed, massively scalable, multi-model database service

Strong

Consistent Prefix

Bounded Staleness

Session

Turnkey global distribution

Elastic scale out
of storage & throughput

Guaranteed low latency
at the 99th percentile

Azure Cosmos DB

A globally distributed, massively scalable, multi-model database service

Strong

Consistent Prefix

Bounded Staleness

Session

Turnkey global distribution

Elastic scale out
of storage & throughput

Guaranteed low latency
at the 99th percentile

Azure Cosmos DB

A globally distributed, massively scalable, multi-model database service

Strong

Eventual

Bounded Staleness

Session

Turnkey global distribution

Elastic scale out
of storage & throughput

Guaranteed low latency
at the 99th percentile

Consistent Prefix

Azure Cosmos DB

A globally distributed, massively scalable, multi-model database service

Comprehensive SLAs

Turnkey global distribution

Elastic scale out
of storage & throughput

Guaranteed low latency
at the 99th percentile

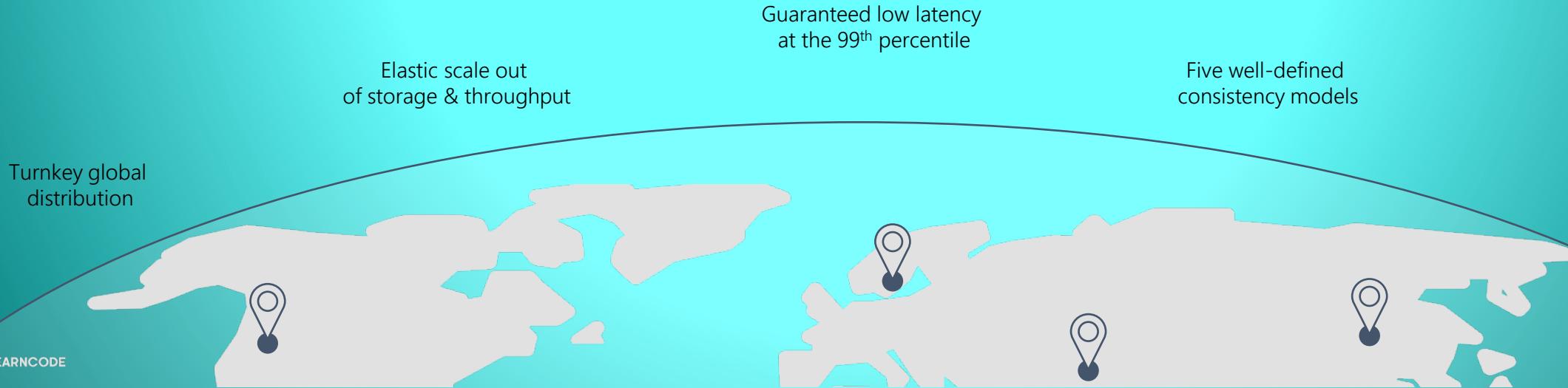
Five well-defined
consistency models



Azure Cosmos DB

A globally distributed, massively scalable, multi-model database service

Comprehensive SLAs



Azure Cosmos DB

A globally distributed, massively scalable, multi-model database service



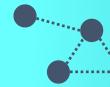
Key-value



Column-family



Document



Graph

Turnkey global distribution

Elastic scale out
of storage & throughput

Guaranteed low latency
at the 99th percentile

Five well-defined
consistency models

Comprehensive
SLAs

Azure Cosmos DB

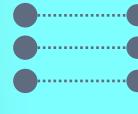
A globally distributed, massively scalable, multi-model database service



Table API



cassandra

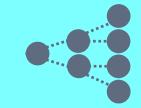


Key-value



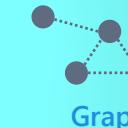
Column-family

Core
(SQL)
API



Document

MongoDB



Graph

Guaranteed low latency
at the 99th percentile

Five well-defined
consistency models

Turnkey global
distribution

Elastic scale out
of storage & throughput

Comprehensive
SLAs

Azure Cosmos DB

A globally distributed, massively scalable, multi-model database service

What if we have **REALLY** large data requirements?



XBOX 360™



Azure Cosmos DB Capabilities

Resource	Default Limit
Maximum RUs per container	1,000,000
Maximum RUs per database	1,000,000
Maximum RUs per (logical) partition key	20-Gb
Maximum number of distinct (logical) partition keys	Unlimited
Maximum storage per container	Unlimited
Maximum storage per database	Unlimited
Maximum size of an item	2-Mb



Cosmos Use Cases

The Hitchhiker's Guide to the Cosmos

IoT and Telematics

Azure Cosmos DB Use Cases

**Ingest Bursts of
Data**

**Process and
Analyze
Streaming Data**

**Archive Data to
Cold Storage**

IoT and Telematics

Azure Cosmos DB Use Cases

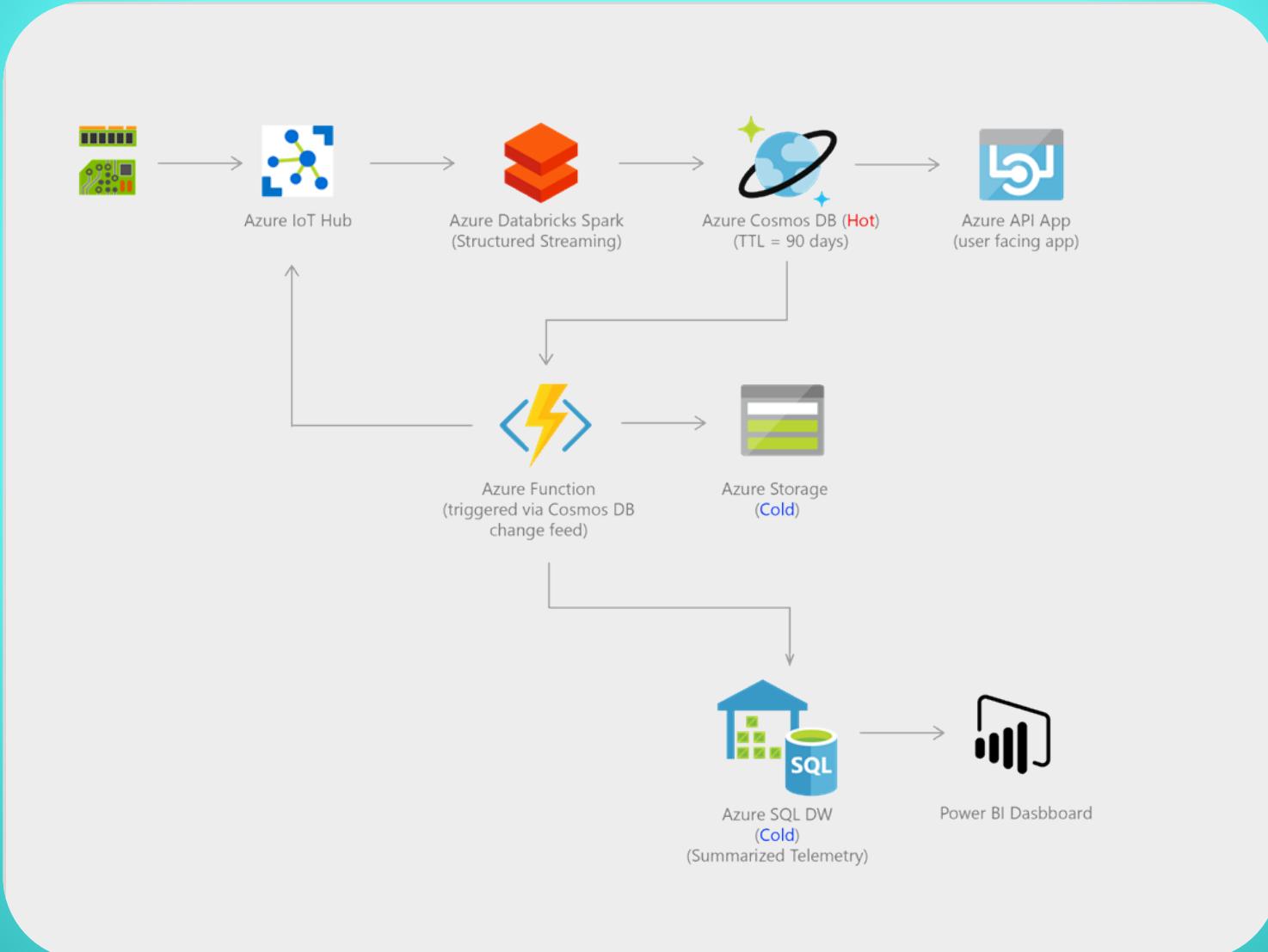
**Ingest Bursts of
Data**

**Process and
Analyze
Streaming Data**

**Archive Data to
Cold Storage**

IoT and Telematics

Azure Cosmos DB Use Cases



Retail Marketing

Azure Cosmos DB Use Cases

Storing and querying sets of attributes

User Accounts

**Product
Catalogs**

**IoT Device
Registers**

Retail Marketing

Azure Cosmos DB Use Cases

Storing and querying sets of attributes

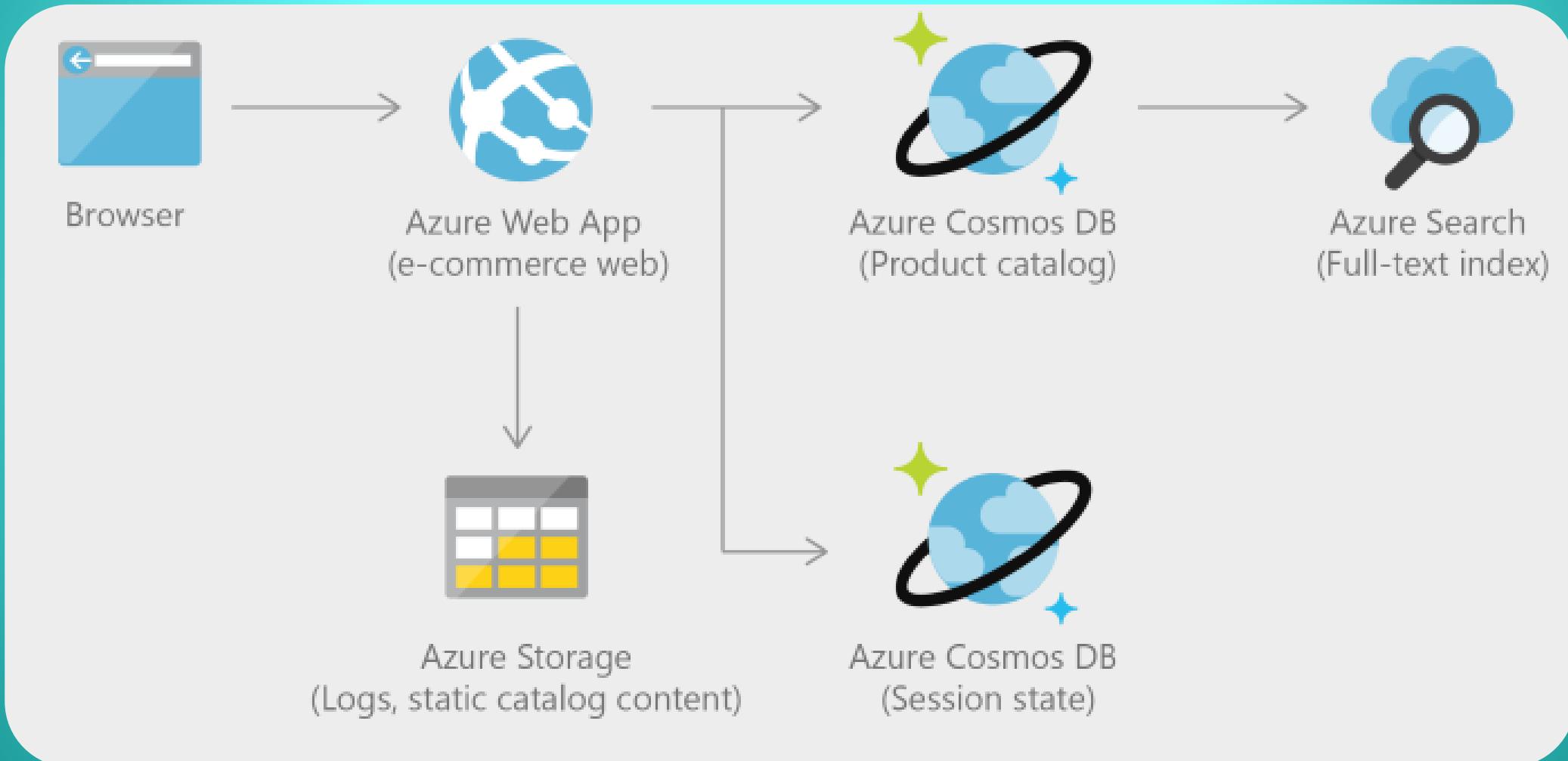
User Accounts

**Product
Catalogs**

**IoT Device
Registers**

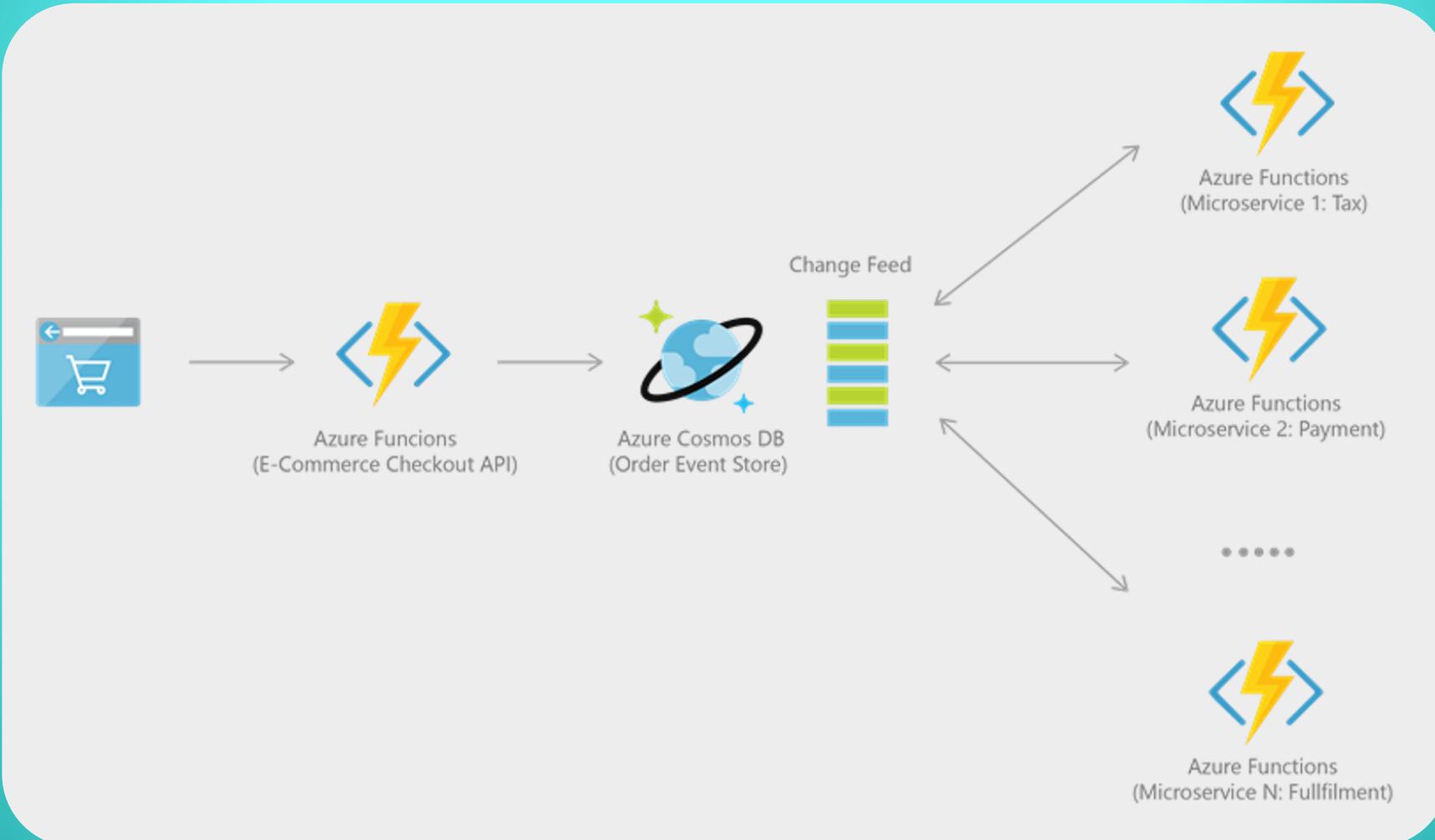
Retail and Marketing

Azure Cosmos DB Use Cases



Retail and Marketing

Azure Cosmos DB Use Cases



Gaming

Azure Cosmos DB Use Cases

**Single-Millisecond
Latencies**



Handle Massive Spikes



Gaming

Azure Cosmos DB Use Cases

**Single-Millisecond
Latencies**

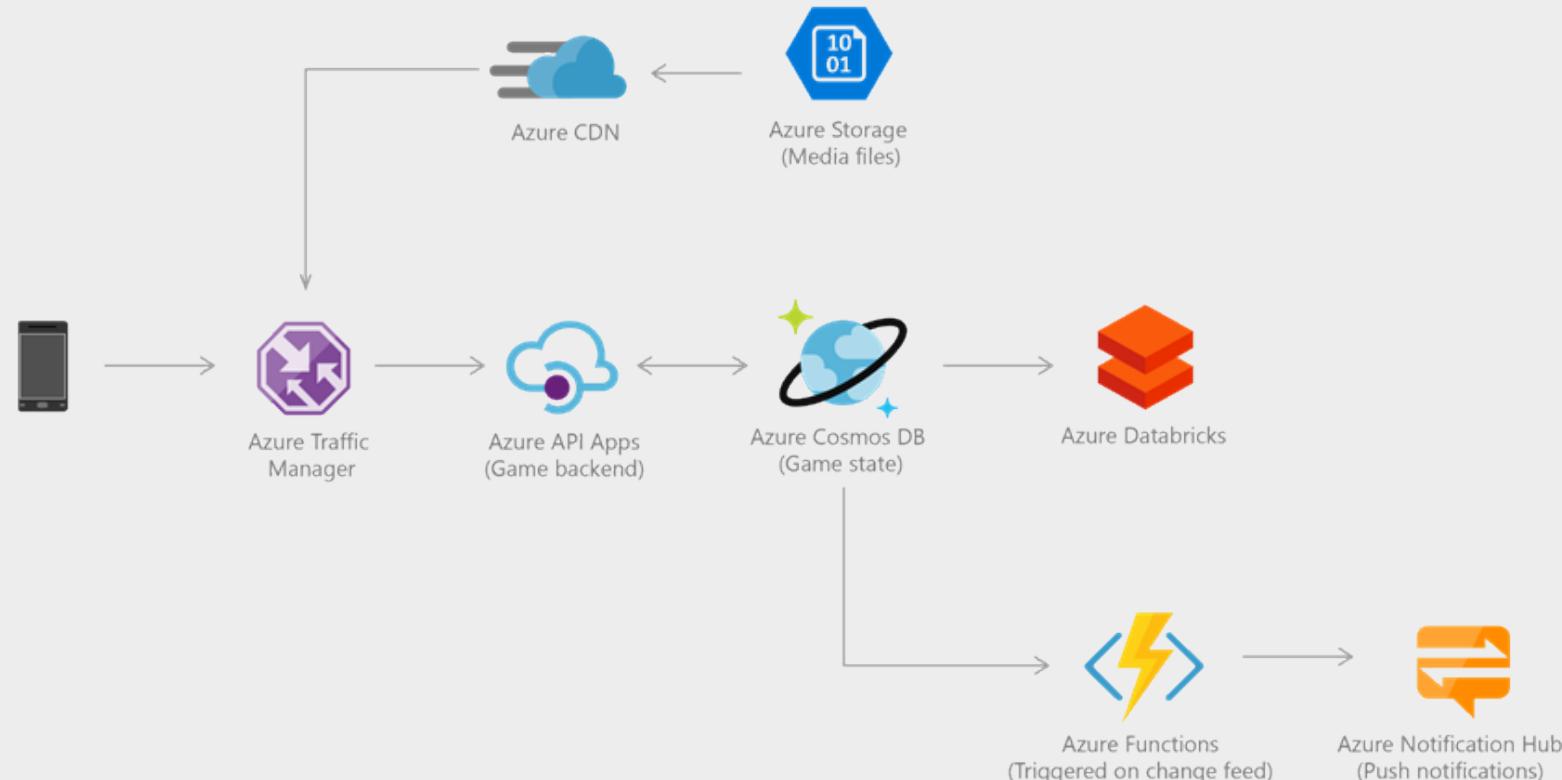


Handle Massive Spikes



Gaming

Azure Cosmos DB Use Cases



Web & Mobile Applications

Azure Cosmos DB Use Cases

Modeling Social Interactions

Integrating with Third-Party Services

Building Rich Personalized Experiences

Web & Mobile Applications

Azure Cosmos DB Use Cases

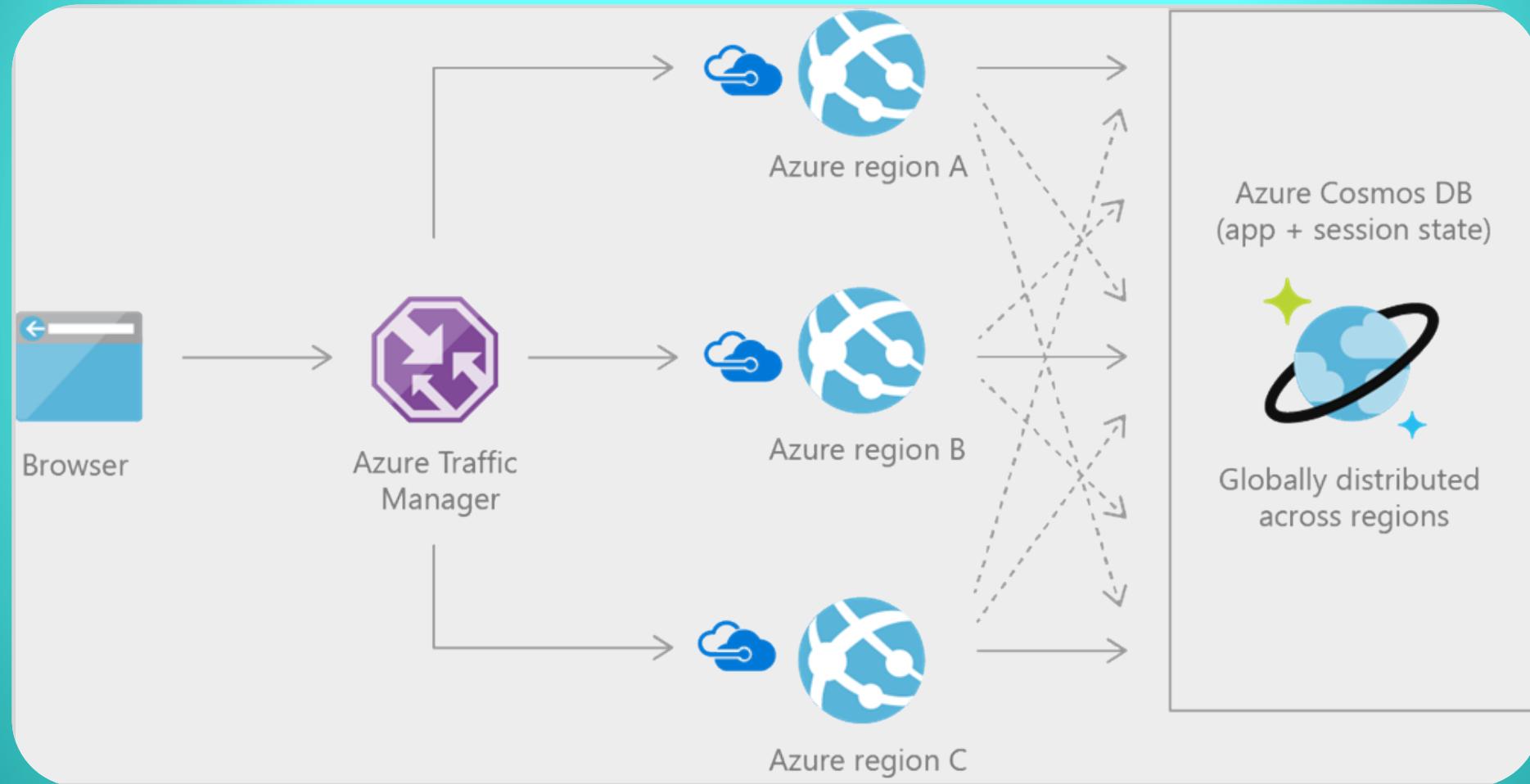
Modeling Social Interactions

Integrating with Third-Party Services

Building Rich Personalized Experiences

Gaming

Azure Cosmos DB Use Cases





Integrations

The Hitchhiker's Guide to the Cosmos

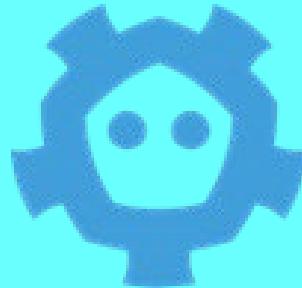
Cosmos DB Integrations



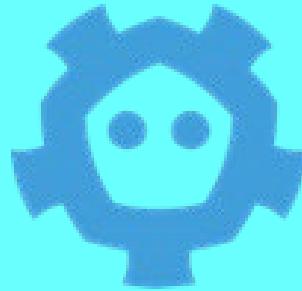
Cosmos DB Integrations



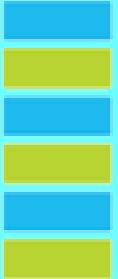
Cosmos DB Integrations



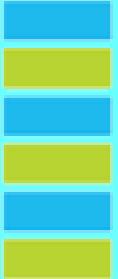
Cosmos DB Integrations



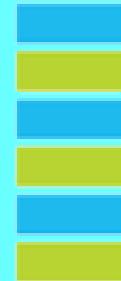
Cosmos DB Integrations



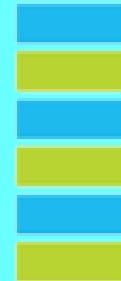
Cosmos DB Integrations



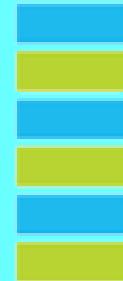
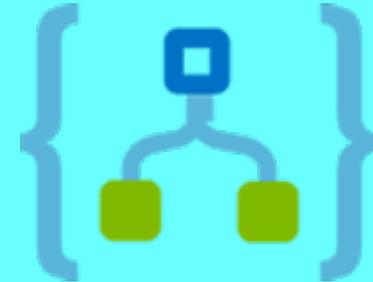
Cosmos DB Integrations



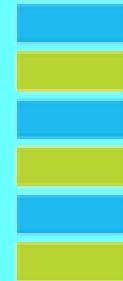
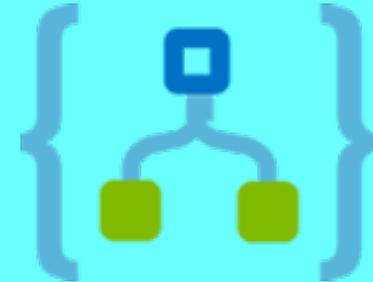
Cosmos DB Integrations



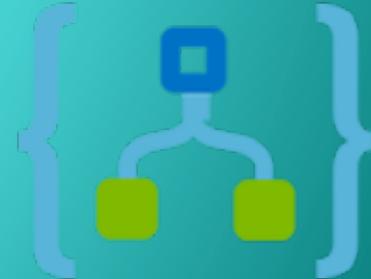
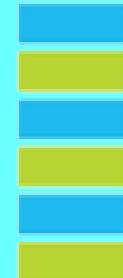
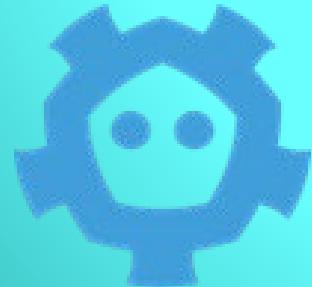
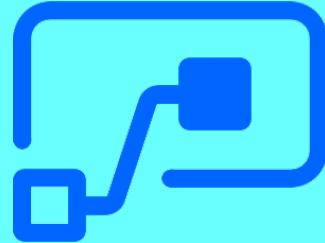
Cosmos DB Integrations



Cosmos DB Integrations



Cosmos DB Integrations



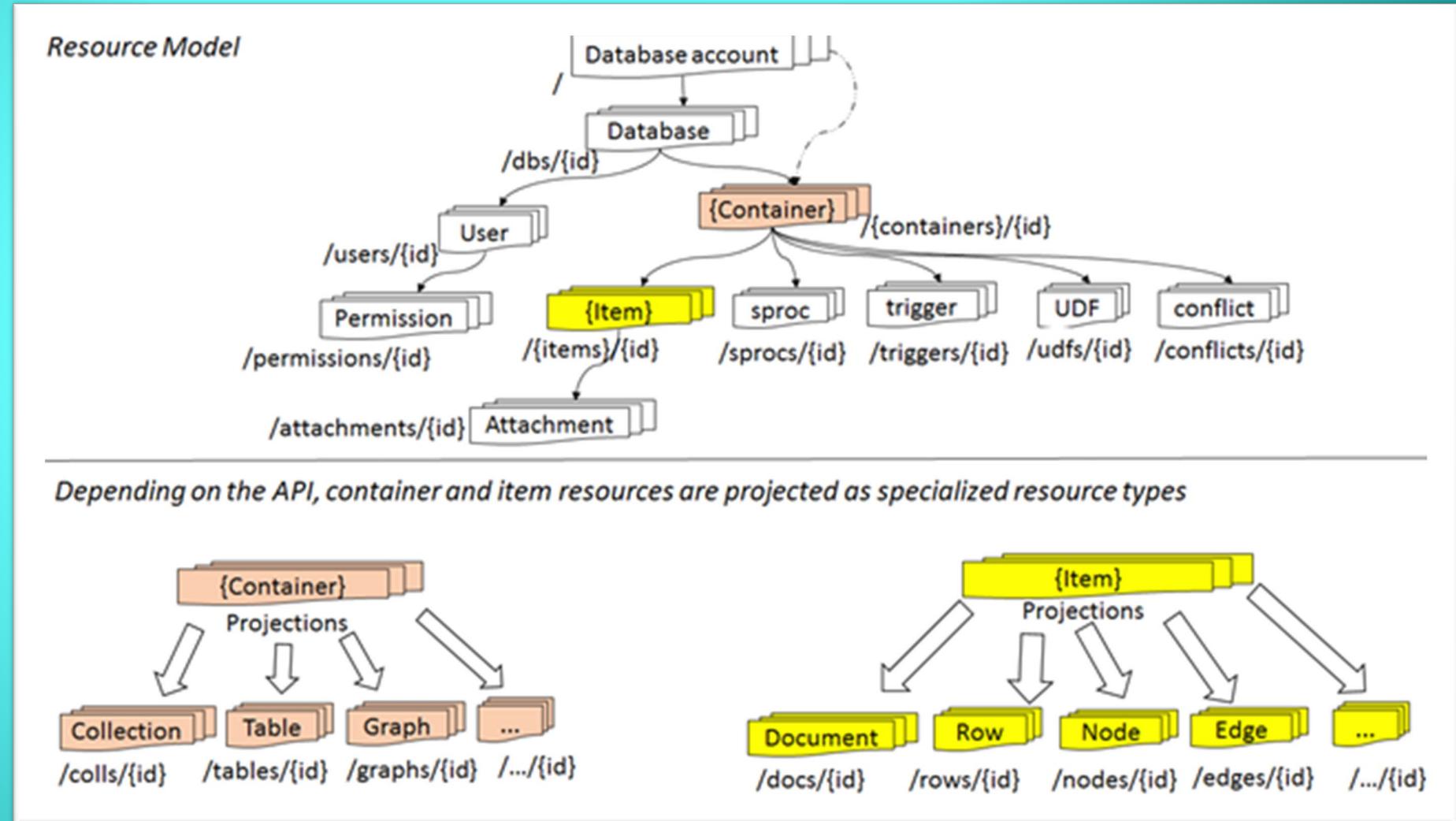
Wide- Range of Data APIs

The Hitchhiker's Guide to the Cosmos



Atom Resource Sequence

Wide-Range of Data APIs



SQL API

Document Database



SQL API – What

Wide-Range of Data APIs

**Document
Database**

**SQL to Query
JSON
Documents**

**JavaScript
Programming
Model**

SQL API – When

Wide-Range of Data APIs

Building a new non-relational
document database and want to
query using SQL

SQL API – How: Data Model

Wide-Range of Data APIs

The screenshot shows the Azure portal interface for a SQL API. At the top, there's a search bar with 'Items' and a close button. Below it is a query editor with the following SQL statement:

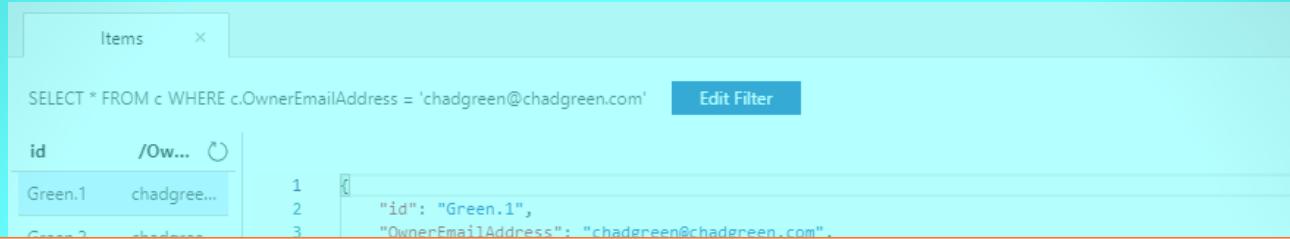
```
SELECT * FROM c WHERE c.OwnerEmailAddress = 'chadgreen@chadgreen.com'
```

Next to the query is a blue 'Edit Filter' button. On the right side of the interface, there's a large code editor window displaying JSON data. The JSON structure represents the results of the SQL query. It includes fields like 'id', 'OwnerEmailAddress', 'Title', 'Abstract', 'ElevatorPitch', 'ImportantDetails', 'LanguageCode', 'Langauge', 'LearningObjectives', 'Tags', and 'PresentationTypes'. The 'LearningObjectives' array contains three items, each with an 'Objective' field. The 'Tags' array contains five items, each with a 'Name' field. The 'PresentationTypes' array contains one item, also with a 'Name' field.

```
1  {
2    "id": "Green.1",
3    "OwnerEmailAddress": "chadgreen@chadgreen.com",
4    "Title": "The Hitchhiker's Guide to the Cosoms",
5    "Abstract": "<p>Today's applications are required to be highly responsible and always online. Cosmos DB is designed to handle the challenges of modern distributed systems and provide reliable data storage and processing capabilities. In this session, we will explore the various data models supported by Cosmos DB, including document, graph, and key-value models, and how they can be used to build efficient and scalable applications. We will also discuss the benefits of using Cosmos DB for different use cases, such as real-time analytics, machine learning, and IoT. By the end of the session, you will have a better understanding of the different data models available in Cosmos DB and how to choose the right one for your specific needs.", "ElevatorPitch": "<p>In this session, you will learn what you can do with Cosmos DB, the benefits of using it, and how to build the best data solution for your application.", "ImportantDetails": null,
6    "LanguageCode": "en",
7    "Langauge": "English",
8    "LearningObjectives": [
9      {
10        "Objective": "Learn about the different Azure Cosmos DB data models"
11      },
12      {
13        "Objective": "Learn how to use the benefits of Azure Cosmos DB to build the best data solution"
14      },
15      {
16        "Objective": "Learn about some of the common Cosmos DB use cases"
17      }
18    ],
19    "Tags": [
20      {
21        "Name": "Azure"
22      },
23      {
24        "Name": "Database"
25      },
26      {
27        "Name": "Cosmos DB"
28      },
29      {
30        "Name": "Cloud"
31      },
32      {
33        "Name": "Architecture"
34      }
35    ],
36    "PresentationTypes": [
37      {
38        "Name": "60-Minute Session",
39        "Length": 60
40      }
41    ]
42  }
```

SQL API – How: Data Model

Wide-Range of Data APIs



A screenshot of a SQL API interface titled "Items". It shows a query: "SELECT * FROM c WHERE c.OwnerEmailAddress = 'chadgreen@chadgreen.com'". Below the query is a table with columns "id" and "/Ow...". There are two rows: "Green.1" and "Green.2". The row "Green.1" is selected. To the right of the table is a JSON representation of the data:

```
1  {
2   "id": "Green.1",
3   "OwnerEmailAddress": "chadgreen@chadgreen.com",
```

```
SELECT *
FROM c
WHERE c.OwnerEmailAddress =
'chadgreen@chadgreen.com'
```

```
34
35   {
36     "Name": "Architecture"
37   }
38 ],
39 "PresentationTypes": [
40   {
41     "Name": "60-Minute Session",
42     "Length": 60
```

SQL API – How: Insert

Wide-Range of Data APIs

```
public class Presentation
{
    [JsonProperty(PropertyName = "id")]
    public string Id { get; set; }

    public string OwnerEmailAddress { get; set; }
    public string Title { get; set; }
    public string Abstract { get; set; }
    public string ElevatorPitch { get; set; }
    public string ImportantDetails { get; set; }
    public string LanguageCode { get; set; }
    public string Langauge { get; set; }
    public LearningObjective[] LearningObjectives { get; set; }
    public Tag[] Tags { get; set; }
    public PresentationType[] PresentationTypes { get; set; }
}
```

SQL API – How: Insert

Wide-Range of Data APIs

```
public class Presentation
{
    [JsonProperty(PropertyName = "id")]
    public string Id { get; set; }

    public string OwnerEmailAddress { get; set; }
    public string Title { get; set; }
    public string Abstract { get; set; }
    public string ElevatorPitch { get; set; }
    public string ImportantDetails { get; set; }
    public string LanguageCode { get; set; }
    public string Langauge { get; set; }
    public LearningObjective[] LearningObjectives { get; set; }
    public Tag[] Tags { get; set; }
    public PresentationType[] PresentationTypes { get; set; }
}
```

SQL API – How: Insert

Wide-Range of Data APIs

```
private async Task AddPresentation(ExistingPresentation existingPresentation)
{
    Presentation presentation = Generate.Presentation(existingPresentation);
    try
    {
        ItemResponse<Presentation> presentationResponse =
            await this.container.ReadItemAsync<Presentation>(presentation.Id,
            new PartitionKey(presentation.OwnerEmailAddress));
        Console.WriteLine("Item in database with id: {0} already exists\n",
            presentationResponse.Resource.Id);
    }
    catch
    (
        CosmosException ex) when (ex.StatusCode == HttpStatusCode.NotFound)
    {
        ItemResponse<Presentation> presentationResponse =
            await this.container.CreateItemAsync<Presentation>(presentation,
            new PartitionKey(presentation.OwnerEmailAddress));
        Console.WriteLine("Created item with id: {0} Using {1} RUs.\n",
            presentationResponse.Resource.Id, presentationResponse.RequestCharge);
    }
}
```

SQL API – How: Insert

Wide-Range of Data APIs

```
private async Task AddPresentation(ExistingPresentation existingPresentation)
{
    Presentation presentation = Generate.Presentation(existingPresentation);
    try
    {
        ItemResponse<Presentation> presentationResponse =
            await this.container.ReadItemAsync<Presentation>(presentation.Id,
            new PartitionKey(presentation.OwnerEmailAddress));
        Console.WriteLine("Item in database with id: {0} already exists\n",
            presentationResponse.Resource.Id);
    }
    catch
    (
        CosmosException ex) when (ex.StatusCode == HttpStatusCode.NotFound)
    {
        ItemResponse<Presentation> presentationResponse =
            await this.container.CreateItemAsync<Presentation>(presentation,
            new PartitionKey(presentation.OwnerEmailAddress));
        Console.WriteLine("Created item with id: {0} Using {1} RUs.\n",
            presentationResponse.Resource.Id, presentationResponse.RequestCharge);
    }
}
```

SQL API – How: Insert

Wide-Range of Data APIs

```
private async Task AddPresentation(ExistingPresentation existingPresentation)
{
    Presentation presentation = Generate.Presentation(existingPresentation);
    try
    {
        ItemResponse<Presentation> presentationResponse =
            await this.container.ReadItemAsync<Presentation>(presentation.Id,
                new PartitionKey(presentation.OwnerEmailAddress));
        Console.WriteLine("Item in database with id: {0} already exists\n",
            presentationResponse.Resource.Id);
    }
    catch
    (
        CosmosException ex) when (ex.StatusCode == HttpStatusCode.NotFound)
    {
        ItemResponse<Presentation> presentationResponse =
            await this.container.CreateItemAsync<Presentation>(presentation,
                new PartitionKey(presentation.OwnerEmailAddress));
        Console.WriteLine("Created item with id: {0} Using {1} RUs.\n",
            presentationResponse.Resource.Id, presentationResponse.RequestCharge);
    }
}
```

SQL API – How: Query

Wide-Range of Data APIs

```
private async Task QueryPresentationsAsync()
{
    var sqlQueryText = "SELECT * FROM c WHERE c.OwnerEmailAddress = 'chadgreen@chadgreen.com'";
    Console.WriteLine("Running query: {0}\n", sqlQueryText);

    QueryDefinition queryDefinition = new QueryDefinition(sqlQueryText);
    FeedIterator<Presentation> queryResultSetIterator
        = this.container.GetItemQueryIterator<Presentation>(queryDefinition);

    List<Presentation> presentations = new List<Presentation>();
    while (queryResultSetIterator.HasMoreResults)
    {
        FeedResponse<Presentation> currentResultSet = await queryResultSetIterator.ReadNextAsync();
        foreach (Presentation presentation in currentResultSet)
        {
            presentations.Add(presentation);
            Console.WriteLine($"{presentation.Title}");
        }
    }
}
```

SQL API – How: Query

Wide-Range of Data APIs

```
private async Task QueryPresentationsAsync()
{
    var sqlQueryText = "SELECT * FROM c WHERE c.OwnerEmailAddress = 'chadgreen@chadgreen.com'";
    Console.WriteLine("Running query: {0}\n", sqlQueryText);

    QueryDefinition queryDefinition = new QueryDefinition(sqlQueryText);
    FeedIterator<Presentation> queryResultSetIterator
        = this.container.GetItemQueryIterator<Presentation>(queryDefinition);

    List<Presentation> presentations = new List<Presentation>();
    while (queryResultSetIterator.HasMoreResults)
    {
        FeedResponse<Presentation> currentResultSet = await queryResultSetIterator.ReadNextAsync();
        foreach (Presentation presentation in currentResultSet)
        {
            presentations.Add(presentation);
            Console.WriteLine($"\\t{presentation.Title}");
        }
    }
}
```

SQL API – How: Query

Wide-Range of Data APIs

```
private async Task QueryPresentationsAsync()
{
    var sqlQueryText = "SELECT * FROM c WHERE c.OwnerEmailAddress = 'chadgreen@chadgreen.com'";
    Console.WriteLine("Running query: {0}\n", sqlQueryText);

    QueryDefinition queryDefinition = new QueryDefinition(sqlQueryText);
    FeedIterator<Presentation> queryResultSetIterator
        = this.container.GetItemQueryIterator<Presentation>(queryDefinition);

    List<Presentation> presentations = new List<Presentation>();
    while (queryResultSetIterator.HasMoreResults)
    {
        FeedResponse<Presentation> currentResultSet = await queryResultSetIterator.ReadNextAsync();
        foreach (Presentation presentation in currentResultSet)
        {
            presentations.Add(presentation);
            Console.WriteLine($"\\t{presentation.Title}");
        }
    }
}
```

SQL API – How: Query

Wide-Range of Data APIs

```
private async Task QueryPresentationsAsync()
{
    var sqlQueryText = "SELECT * FROM c WHERE c.OwnerEmailAddress = 'chadgreen@chadgreen.com'";
    Console.WriteLine("Running query: {0}\n", sqlQueryText);

    QueryDefinition queryDefinition = new QueryDefinition(sqlQueryText);
    FeedIterator<Presentation> queryResultSetIterator
        = this.container.GetItemQueryIterator<Presentation>(queryDefinition);

    List<Presentation> presentations = new List<Presentation>();
    while (queryResultSetIterator.HasMoreResults)
    {
        FeedResponse<Presentation> currentResultSet = await queryResultSetIterator.ReadNextAsync();
        foreach (Presentation presentation in currentResultSet)
        {
            presentations.Add(presentation);
            Console.WriteLine($"\\t{presentation.Title}");
        }
    }
}
```

SQL API – Querying Options

Wide-Range of Data APIs

API

LINQ to SQL

JavaScript

**Entity
Framework**

MongoDB
Document Database

{ LEAF }

MongoDB – What

Wide-Range of Data APIs

**Native
Implementation**

**Interact
Transparently**

**V3.6
Compatibility**



MongoDB – When

Wide-Range of Data APIs

Migrating data from a MongoDB
database to Azure Cosmos DB's
fully managed service

MongoDB – How: Data Model

Wide-Range of Data APIs

The screenshot shows the MongoDB Compass interface. On the left, a sidebar lists documents with IDs d8001f2..., ba38503..., and 6211e93... A "Load more" button is visible below the list. On the right, a large panel displays a JSON document with the following structure:

```
1  {
2   "_id" : {
3     "$binary" : "K8A2AkQk25BKrEADmfEw==",
4     "$type" : "03"
5   },
6   "OwnerEmailAddress" : "chadgreen@chadgreen.com",
7   "Title" : "Event-Driven Architecture in the Cloud",
8   "Abstract" : "<p>Event-driven architectures is a versatile approach to designing and integrating complex software systems wi",
9   "ElevatorPitch" : "<p>Event-driven architectures is a versatile approach to designing and integrating complex software syst",
10  "ImportantDetails" : null,
11  "LanguageCode" : "en",
12  "Langauge" : "English",
13  "LearningObjectives" : [
14    {
15      "Objective" : "Learn the basics of event-driven architecture"
16    },
17    {
18      "Objective" : "Learn how to transform your complext systems to become event driven"
19    },
20    {
21      "Objective" : "Learning about the benefits event-driven architecture brings to your business"
22    }
23  ],
24  "Tags" : [
25    {
26      "Name" : "Architecture"
27    },
28    {
29      "Name" : "Event-Driven Architecture"
30    },
31    {
32      "Name" : "Azure"
33    },
34    {
35      "Name" : "Event Hubs"
36    },
37    {
38      "Name" : "Cosmos DB"
39    },
40    {
41      "Name" : "Azure Functions"
42    }
43  ]
```

MongoDB – How: Data Model

Wide-Range of Data APIs



A screenshot of the MongoDB Compass interface. The title bar says "Documents". Below it is a filter bar with the query "Filter : {"OwnerEmailAddress": "chadgreen@chadgreen.com"}" and a "Edit Filter" button. The main area shows a table with two columns: "_id" and a document preview. The _id column lists three document IDs: "d8001f2...", "ba38503...", and "6211e93...". The document preview shows the following JSON structure:

```
1  {
2    "_id" : {
3      "$binary" : "K8A2AkQk25BKrEAdmFEw==",
4      "$type" : "03"
5    },
6    "OwnerEmailAddress" : "chadgreen@chadgreen.com",
7    "Title" : "Event-Driven Architecture in the Cloud",
8    "Abstract" : "<p>Event-driven architectures is a versatile approach to designing and integrating complex software systems wi",
9    "ElevatorPitch" : "<p>Event-driven architectures is a versatile approach to designing and integrating complex software syst"
```

Filter: {"OwnerEmailAddress": "chadgreen@chadgreen.com"}

```
26   },
27   {
28     "Name" : "Architecture"
29   },
30   {
31     "Name" : "Event-Driven Architecture"
32   },
33   {
34     "Name" : "Azure"
35   },
36   {
37     "Name" : "Event Hubs"
38   },
39   {
40     "Name" : "Cosmos DB"
41   },
42   {
43     "Name" : "Azure Functions"
```

MongoDB – How: Insert

Wide-Range of Data APIs

```
public class Presentation
{
    [BsonId(IdGenerator =typeof(CombGuidGenerator))] public Guid Id { get; set; }

    [BsonElement("OwnerEmailAddress")] public string OwnerEmailAddress { get; set; }

    [BsonElement("Title")] public string Title { get; set; }

    [BsonElement("Abstract")] public string Abstract { get; set; }

    [BsonElement("ElevatorPitch")] public string ElevatorPitch { get; set; }

    [BsonElement("ImportantDetails")] public string ImportantDetails { get; set; }

    [BsonElement("LanguageCode")] public string LanguageCode { get; set; }

    [BsonElement("Langauge")] public string Langauge { get; set; }

    [BsonElement("LearningObjectives")] public LearningObjective[] LearningObjectives { get; set; }

    [BsonElement("Tags")] public Tag[] Tags { get; set; }

    [BsonElement("PresentationTypes")] public PresentationType[] PresentationTypes { get; set; }
}
```

MongoDB – How: Insert

Wide-Range of Data APIs

```
public void CreatePresentation(Presentation presentation)
{
    var collection = GetPresentationsCollection();
    try
    {
        collection.InsertOne(presentation);
    }
    catch (MongoCommandException ex)
    {
        string msg = ex.Message;
    }
}
```

MongoDB – How: Insert

Wide-Range of Data APIs

```
private IMongoCollection<Presentation> GetPresentationsCollection()
{
    MongoClientSettings settings = new MongoClientSettings();
    settings.Server = new MongoServerAddress(host, 10255);
    settings.UseTls = true;
    settings.SslSettings = new SslSettings();
    settings.SslSettings.EnabledSslProtocols = SslProtocols.Tls12;

    MongoIdentity identity = new MongoInternalIdentity(dbName, userName);
    MongoIdentityEvidence evidence = new PasswordEvidence(password);

    settings.Credential = new MongoCredential("SCRAM-SHA-1", identity, evidence);

    MongoClient client = new MongoClient(settings);
    var database = client.GetDatabase(dbName);
    var presentationCollection = database.GetCollection<Presentation>(collectionName);
    return presentationCollection;
}
```

MongoDB – How: Query

Wide-Range of Data APIs

```
public List<Presentation> GetAllPresentations()
{
    try { var collection = GetPresentationsCollection();
    return collection.Find(new BsonDocument()).ToList();
}
    catch (MongoConnectionException)
    {
        return new List<Presentation>();
    }
}
```

MongoDB – Cosmos or Atlas

Wide-Range of Data APIs

Cosmos DB

- Small documents – the smaller the cheaper
- Read more often than write
- Like idea to start small and pay-as-you-go
- Support included in Azure subscription
- Need a guaranteed latency despite usage

MongoDB Atlas

- Any size documents (only choice for documents over 2-Mb)
- Like to have a fixed budget for storage
- Use Mongo-API features that are not covered by Cosmos DB
- Freedom to create unique indexes
- Write data more often than read
- Want to decide backup policy

Graph API

Graph Database



Graph API – What

Wide-Range of Data APIs

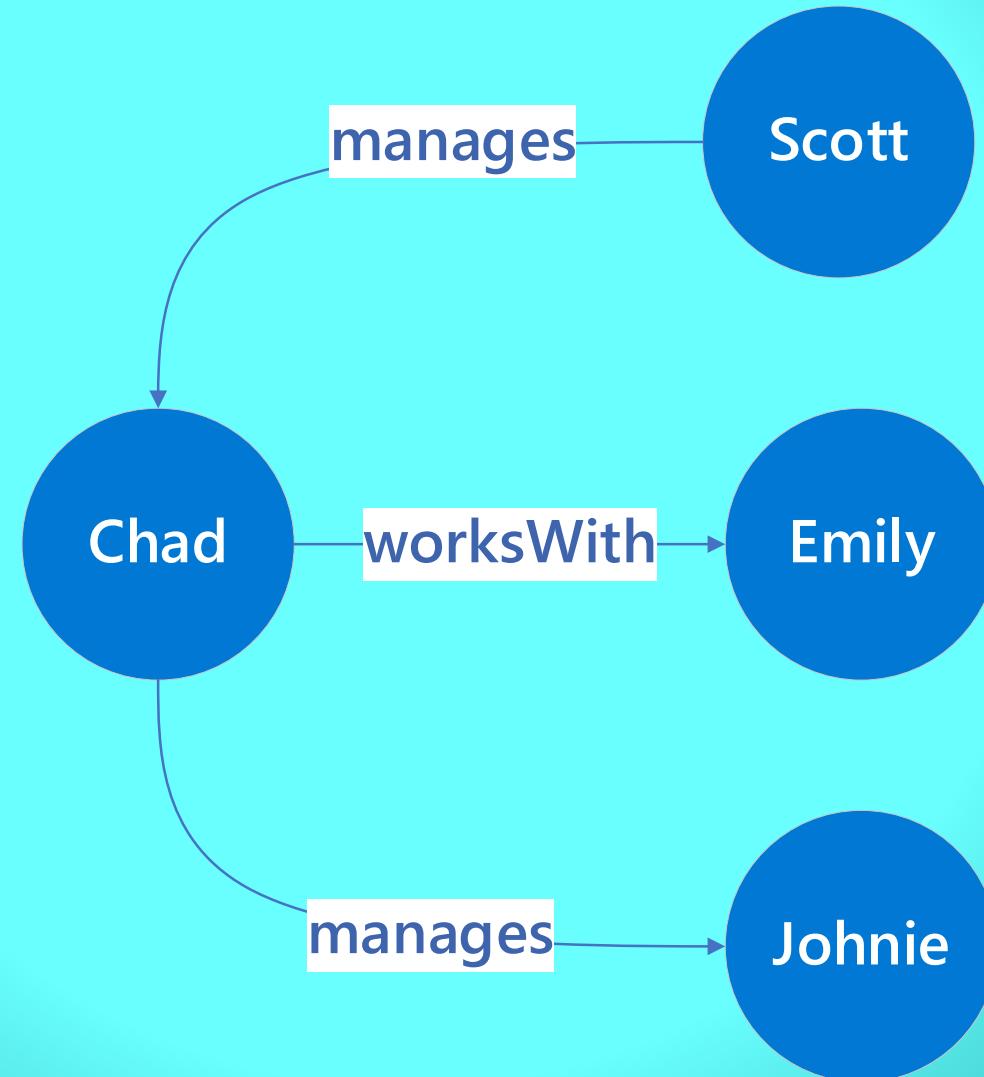
**Vertices and
Edges**

**Natural Data
Representation**

**Model All Kinds
of Scenarios**

What is a Graph

Wide-Range of Data APIs



Property Graph Model

Wide-Range of Data APIs

Contains **nodes** (vertices) and **relationships** (edges)

Nodes and relationships contain **properties**

Relationships are **named** and **directed** with a **start** and **end** node



The Power of Graph Databases

Wide-Range of Data APIs

Performance

The Power of Graph Databases

Wide-Range of Data APIs

Performance

Flexibility

The Power of Graph Databases

Wide-Range of Data APIs

Performance

Flexibility

Agility

Graph API – When

Wide-Range of Data APIs

Building a graph database to model
and traverse relationships among
entities

Graph API – How: Data Model

Wide-Range of Data APIs

Graph x

```
g.V().hasLabel('presentation').has('ownerEmailAddress', 'chadgreen@chadgreen.com')
```

Execute Gremlin Query x

JSON Graph Query Stats

Results

- The Hitchhikers Guide to t...
- Graphing Your Way Throu...
- Event-Driven Architecture i...

Graph

> The Hitchhikers Guide to the Cosoms

Properties

id	2e3619f3-a7e0-4e4f-b6bf-7b8196916a23
label	presentation
ownerEmailAddress	chadgreen@chadgreen.com
name	The Hitchhikers Guide to the Cosoms
abstract	abstract
elevatorPitch	elevator pitch

Sources

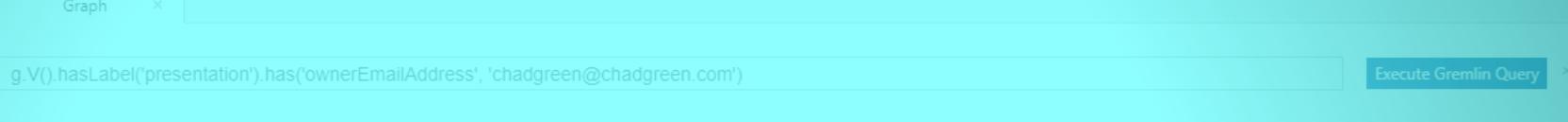
No sources found

Targets

Target	Edge label
Azure	taggedAs
Database	taggedAs
Cosmos DB	taggedAs
Cloud	taggedAs
Architecture	taggedAs
English	presentedIn
Session	is

Graph API – How: Data Model

Wide-Range of Data APIs



A screenshot of a web-based Gremlin query interface. At the top, there's a header with tabs for "Graph" and "x". Below the header is a text input field containing the Gremlin query: "g.V().hasLabel('presentation').has('ownerEmailAddress', 'chadgreen@chadgreen.com')". To the right of the query input is a blue button labeled "Execute Gremlin Query" with a small "x" icon next to it. At the bottom of the interface, there are three tabs: "JSON", "Graph", and "Query Stats".

```
g.V().hasLabel('presentation').has('ownerEmailAddress', 'chadgreen@chadgreen.com')
```

```
g.V().hasLabel  
(‘presentation’)  
.has(‘ownerEmailAddress’,  
‘chadgreen@chadgreen.com’)
```

Cloud
Architecture
English
Session

taggedAs
taggedAs
presentedIn
is

Graph API – How: Insert

Wide-Range of Data APIs

```
g.addV('presentation')
.property('ownerEmailAddress', 'chadgreen@chadgreen.com')
.property('name', 'The Hitchhikers Guide to the Cosmos')
.property('abstract', 'This is a really cool talk!')
.property('elevatorPitch', 'Cool talk!')
```

Graph API – How: Insert

Wide-Range of Data APIs

```
private static Task<ResultSet<dynamic>> SubmitRequest(GremlinClient gremlinClient,
    KeyValuePair<string, string> query)
{
    try
    {
        return gremlinClient.SubmitAsync<dynamic>(query.Value);
    }
    catch (ResponseException e)
    {
        Console.WriteLine("\tRequest Error!");
        Console.WriteLine($" \tStatusCode: {e.StatusCode}");
        PrintStatusAttributes(e.StatusAttributes);
        Console.WriteLine($" \t[\"x-ms-retry-after-ms\"] :
            { GetValueAsString(e.StatusAttributes, "x-ms-retry-after-ms") }");
        Console.WriteLine($" \t[\"x-ms-activity-id\"] :
            { GetValueAsString(e.StatusAttributes, "x-ms-activity-id") }");

        throw;
    }
}
```

Graph API – How: Insert

Wide-Range of Data APIs

```
private static Task<ResultSet<dynamic>> SubmitRequest(GremlinClient gremlinClient,
    KeyValuePair<string, string> query)
{
    try
    {
        return gremlinClient.SubmitAsync<dynamic>(query.Value);
    }
    catch (ResponseException e)
    {
        Console.WriteLine("\tRequest Error!");
        Console.WriteLine($" \tStatusCode: {e.StatusCode}");
        PrintStatusAttributes(e.StatusAttributes);
        Console.WriteLine($" \t[\"x-ms-retry-after-ms\"] :
            { GetValueAsString(e.StatusAttributes, "x-ms-retry-after-ms") });
        Console.WriteLine($" \t[\"x-ms-activity-id\"] :
            { GetValueAsString(e.StatusAttributes, "x-ms-activity-id") });

        throw;
    }
}
```

Graph API – How: Insert

Wide-Range of Data APIs

```
private static Task<ResultSet<dynamic>> SubmitRequest(GremlinClient gremlinClient,
    KeyValuePair<string, string> query)
{
    try
    {
        return gremlinClient.SubmitAsync<dynamic>(query.Value);
    }
    catch (ResponseException e)
    {
        Console.WriteLine("\tRequest Error!");
        Console.WriteLine($" \tStatusCode: {e.StatusCode}");
        PrintStatusAttributes(e.StatusAttributes);
        Console.WriteLine($" \t[\"x-ms-retry-after-ms\"] :
            { GetValueAsString(e.StatusAttributes, "x-ms-retry-after-ms") }");
        Console.WriteLine($" \t[\"x-ms-activity-id\"] :
            { GetValueAsString(e.StatusAttributes, "x-ms-activity-id") }");

        throw;
    }
}
```

Graph API – How: Query

Wide-Range of Data APIs

```
g.V()  
.hasLabel('tag')  
.has('name', 'Azure')  
.in('taggedAs')  
.hasLabel('presentation')
```

Gremlin API – Querying Options

Wide-Range of Data APIs

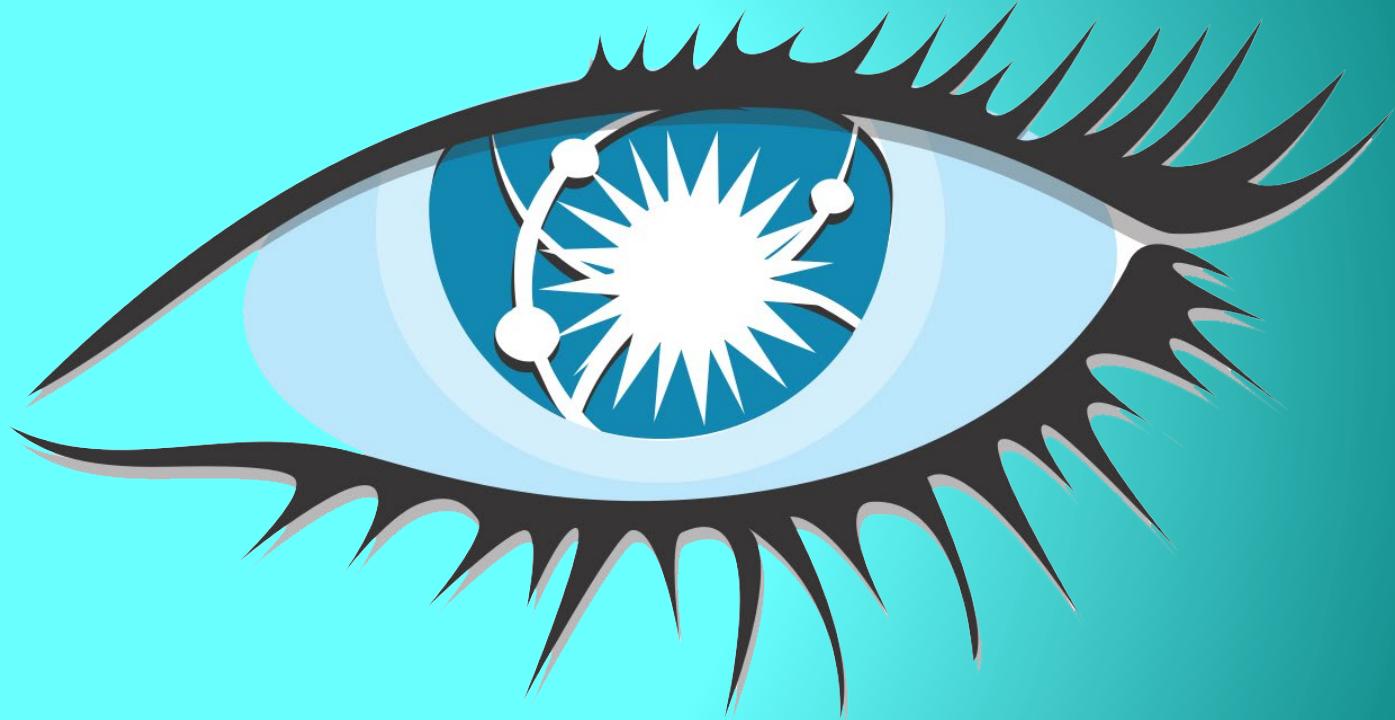
API

JavaScript

SQL API

Cassandra API

Wide Column Store



Cassandra API – What

Wide-Range of Data APIs

**Handle Large
Amounts of
Data**

**Uses
Commodity
Servers**

**Developed by
Facebook**

Graph API – When

Wide-Range of Data APIs

Migrating data from Cassandra to
Azure Cosmos DB

Cassandra API – How: Data Model

Wide-Range of Data APIs

The screenshot shows a search interface for a wide-range of data APIs. At the top, there is a header "Rows" with an "X" button. Below it is a query builder section with the following fields:

Action	And	Field	Type	Operator	Value
+	X	presentation_owr	Text	=	chadgreen@chadgreen.com

Below the query builder are two buttons: "+ Add new clause" and "► Advanced Options".

Underneath the query builder, there is a search bar containing the text "presentation_abstract".

Finally, there is a list of search results, each consisting of a snippet of text and a truncated URL. The first result is:

<p>Data as it appears in the real world is naturally connected, but traditional data modeling focuses on entities which can cause for c... <p>Dat

The other results are partially visible:

<p>Event-driven architectures is a versatile approach to designing and integrating complex software systems with loosely coupled co... <p>Eve

<p>Today's applications are required to be highly responsible and always online. Cosmos DB was built from the ground up to provide ... <p>In t

Cassandra API – How: Insert

Wide-Range of Data APIs

```
public class Presentation
{
    public string presentation_id { get; set; }
    public string presentation_owneremailaddress { get; set; }
    public string presentation_title { get; set; }
    public string presentation_abstract { get; set; }
    public string presentation_elevatorpitch { get; set; }
    public string presentation_importantinformation { get; set; }
    public string presentation_languagecode { get; set; }
    public string presentation_language { get; set; }
}
```

Cassandra API – How: Insert

Wide-Range of Data APIs

```
var options = new Cassandra.SSLOptions(SslProtocols.Tls12,
    true, ValidateServerCertificate);

options.SetHostNameResolver((ipAddress) => CassandraContactPoint);

Cluster cluster = Cluster.Builder()
    .WithCredentials(UserName, Password)
    .WithPort(CassandraPort)
    .AddContactPoint(CassandraContactPoint).WithSSL(options).Build();

ISession session = cluster.Connect();

session = cluster.Connect("speakingengagements");
IMapper mapper = new Mapper(session);

// Inserting Data into presentation table
mapper.Insert<Presentation>(Generate.Presentation(
    ExistingPresentation.EventDrivenArchitectureInTheCloud));
```

Cassandra API – How: Query

Wide-Range of Data APIs

```
Presentation presentationId3 = mapper.FirstOrDefault<Presentation>
("Select * from presentation where presentation_owneremailaddress = ?",
 "chadgreen@chadgreen.com");
Console.WriteLine(presentationId3.presentation_title);
```

Cassandra API – Query Options

Wide-Range of Data APIs

API

CQL

**Cassandra-
Based Tools**

Table API

Table Storage

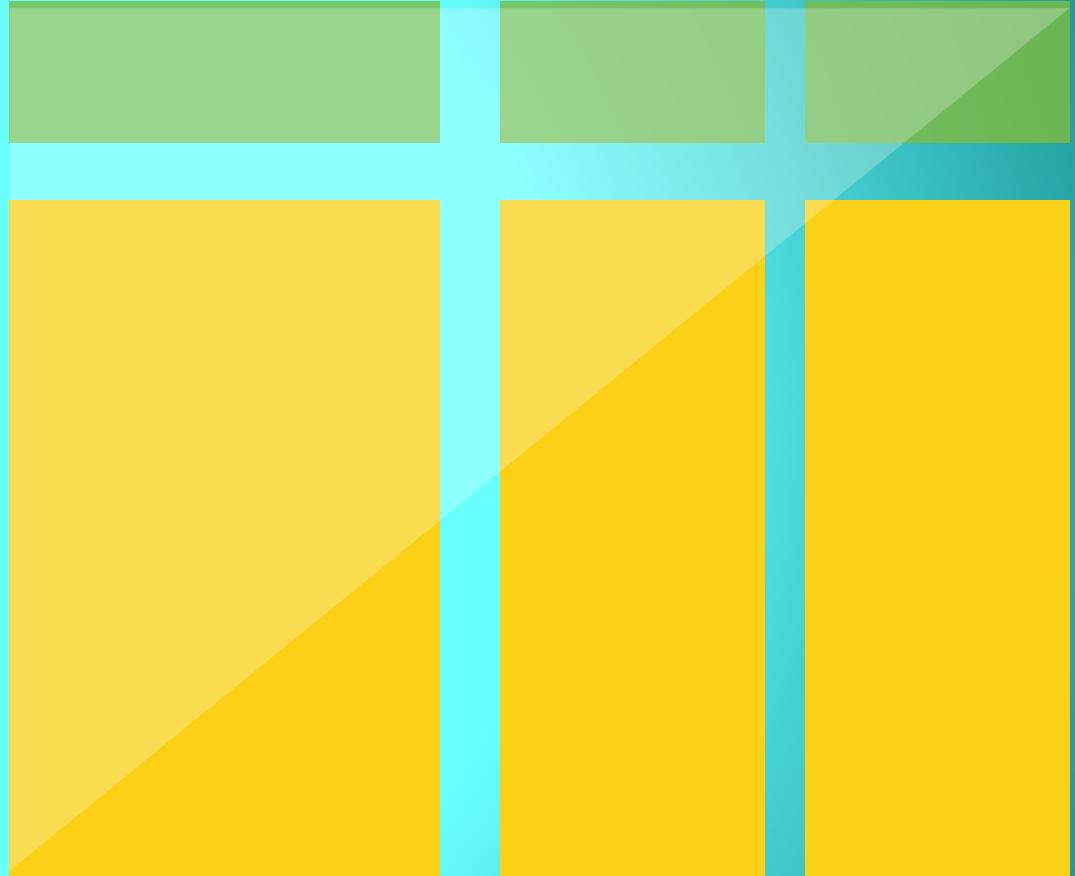


Table API – What

Wide-Range of Data APIs

**Store Large
Amounts of
Data**

**Storing
Structured Non-
Related Data**

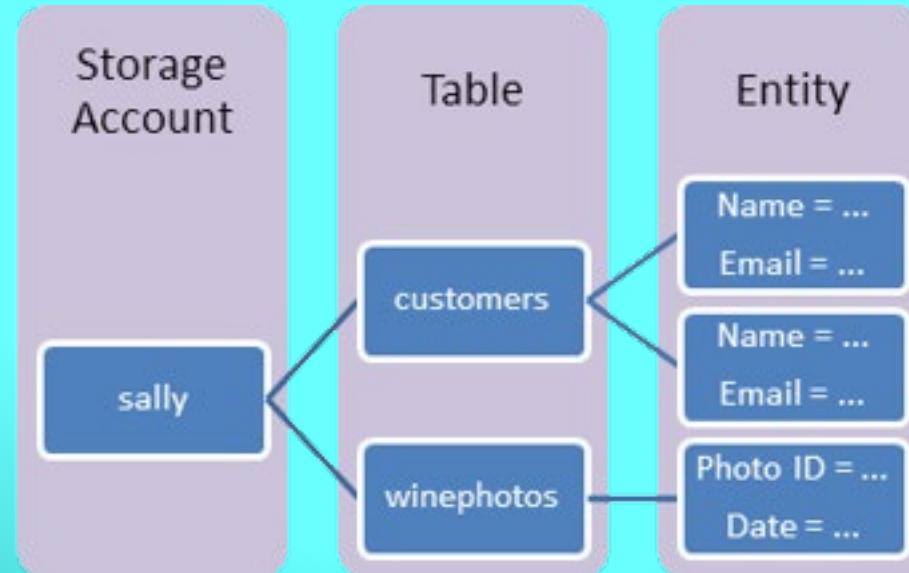


Table API – When

Wide-Range of Data APIs

Migrating data from Azure Table
storage to Cosmos DB

Table API – How: Data Model

Wide-Range of Data APIs

Entities X

Action	And/Or	Field	Type	Operator	Value
+ X	<input type="checkbox"/>	PartitionKey	String	=	chadgreen@chadgreen.com
+ X	<input type="checkbox"/> And	RowKey	String	=	Green.3

[+ Add new clause](#)

[Advanced Options](#)

PartitionKey	RowKey	Timestamp	Abstract
chadgreen@chadgreen.com	Green.3	Tue, 10 Sep 2019 04:11:29 GMT	<p>Event-driven architectures is a versatile approach to designing a

Table API – How: Insert

Wide-Range of Data APIs

```
public class Presentation : TableEntity
{
    public Presentation() { }

    public Presentation(string id, string ownerEmailAddress)
    {
        PartitionKey = ownerEmailAddress;
        RowKey = id;
    }

    public string Title { get; set; }
    public string Abstract { get; set; }
    public string ElevatorPitch { get; set; }
    public string ImportantDetails { get; set; }
    public string LanguageCode { get; set; }
    public string Langauge { get; set; }
}
```

Table API – How: Insert

Wide-Range of Data APIs

```
public class Presentation : TableEntity
{
    public Presentation() { }

    public Presentation(string id, string ownerEmailAddress)
    {
        PartitionKey = ownerEmailAddress;
        RowKey = id;
    }

    public string Title { get; set; }
    public string Abstract { get; set; }
    public string ElevatorPitch { get; set; }
    public string ImportantDetails { get; set; }
    public string LanguageCode { get; set; }
    public string Langauge { get; set; }
}
```

Table API – How: Insert

```
Wic public static async Task<Presentation> InsertOrMergeEntityAsync(CloudTable table, Presentation entity)
{
    if (entity == null)
    {
        throw new ArgumentNullException("entity");
    }

    try
    {
        // Create the InsertOrReplace table operation
        TableOperation insertOrMergeOperation = TableOperation.InsertOrMerge(entity);

        // Execute the operation.
        TableResult result = await table.ExecuteAsync(insertOrMergeOperation);
        Presentation insertedCustomer = result.Result as Presentation;

        if (result.RequestCharge.HasValue)
        {
            Console.WriteLine("Request Charge of InsertOrMerge Operation: " + result.RequestCharge);
        }

        return insertedCustomer;
    }
    catch (StorageException e)
    {
        Console.WriteLine(e.Message);
        Console.ReadLine();
        throw;
    }
}
```

Table API – How: Insert

Wic

```
public static async Task<Presentation> InsertOrMergeEntityAsync(CloudTable table, Presentation entity)
{
    if (entity == null)
    {
        throw new ArgumentNullException("entity");
    }

    try
    {
        // Create the InsertOrReplace table operation
        TableOperation insertOrMergeOperation = TableOperation.InsertOrMerge(entity);

        // Execute the operation.
        TableResult result = await table.ExecuteAsync(insertOrMergeOperation);
        Presentation insertedCustomer = result.Result as Presentation;

        if (result.RequestCharge.HasValue)
        {
            Console.WriteLine("Request Charge of InsertOrMerge Operation: " + result.RequestCharge);
        }

        return insertedCustomer;
    }
    catch (StorageException e)
    {
        Console.WriteLine(e.Message);
        Console.ReadLine();
        throw;
    }
}
```

Table API – How: Query

```
Wid public static async Task<Presentation> RetrieveEntityUsingPointQueryAsync(CloudTable table,
    string partitionKey, string rowKey)
{
    try
    {
        TableOperation retrieveOperation
            = TableOperation.Retrieve<Presentation>(partitionKey, rowKey);
        TableResult result = await table.ExecuteAsync(retrieveOperation);
        Presentation presentation = result.Result as Presentation;
        if (presentation != null)
        {
            Console.WriteLine("\t{0}\t{1}\t{2}", presentation.PartitionKey,
                presentation.RowKey, presentation.Title);
        }

        if (result.RequestCharge.HasValue)
        {
            Console.WriteLine("Request Charge of Retrieve Operation: " + result.RequestCharge);
        }

        return presentation;
    }
    catch (StorageException e)
    {
        Console.WriteLine(e.Message);
        Console.ReadLine();
        throw;
    }
}
```

Table API – How: Query

Wide Range of Data APIs

```
public static async Task<Presentation> RetrieveEntityUsingPointQueryAsync(CloudTable table,
    string partitionKey, string rowKey)
{
    try
    {
        TableOperation retrieveOperation
            = TableOperation.Retrieve<Presentation>(partitionKey, rowKey);
        TableResult result = await table.ExecuteAsync(retrieveOperation);
        Presentation presentation = result.Result as Presentation;
        if (presentation != null)
        {
            Console.WriteLine("\t{0}\t{1}\t{2}", presentation.PartitionKey,
                presentation.RowKey, presentation.Title);
        }

        if (result.RequestCharge.HasValue)
        {
            Console.WriteLine("Request Charge of Retrieve Operation: " + result.RequestCharge);
        }

        return presentation;
    }
    catch (StorageException e)
    {
        Console.WriteLine(e.Message);
        Console.ReadLine();
        throw;
    }
}
```

etcd API

Key-Value Storage



etcd

etcd API – What

Wide-Range of Data APIs

Simple

Secure



kubernetes

Fast

Reliable

etcd API – Why

Wide-Range of Data APIs

No etcd
Operations
Management

etcd API – Why

Wide-Range of Data APIs

No etcd
Operations
Management

etcd API – Why

Wide-Range of Data APIs

No etcd
Operations
Management

Global
distribution &
high availability

etcd API – Why

Wide-Range of Data APIs

No etcd
Operations
Management

Global
distribution &
high availability

etcd API – Why

Wide-Range of Data APIs

No etcd
Operations
Management

Global
distribution &
high availability

Elastic
scalability

etcd API – Why

Wide-Range of Data APIs

No etcd
Operations
Management

Global
distribution &
high availability

Elastic
scalability

etcd API – Why

Wide-Range of Data APIs

No etcd
Operations
Management

Global
distribution &
high availability

Security &
enterprise
readiness

Elastic
scalability

etcd API – When

Wide-Range of Data APIs

Allows developers to scale

Kubernetes state management on a

fully managed cloud native PaaS

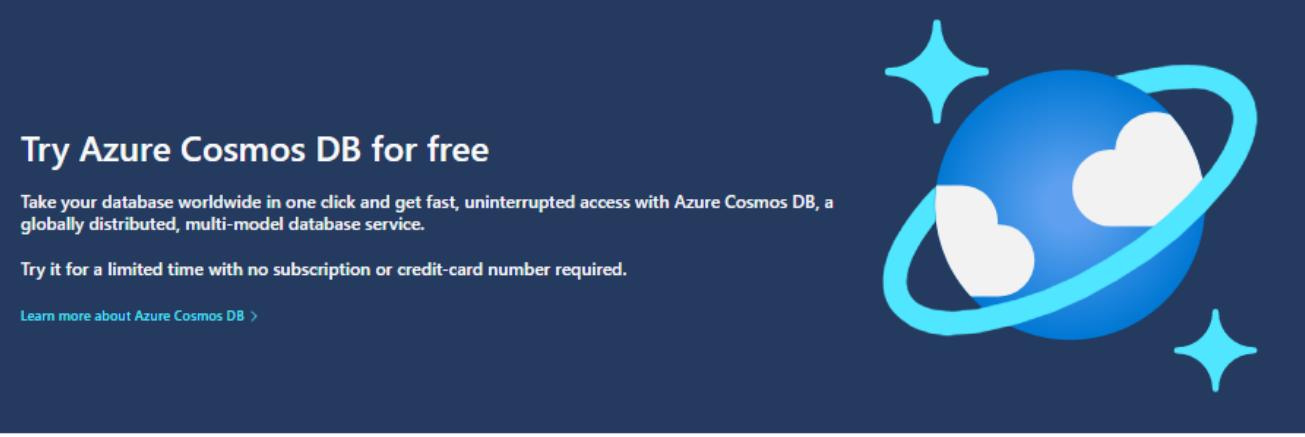
service

Call to Action

The Hitchhiker's Guide to the Cosmos



azure.Microsoft.com/en-us/try/cosmosdb



The landing page for Azure Cosmos DB features a dark blue header with the text "Try Azure Cosmos DB for free". Below this, a subtext encourages users to "Take your database worldwide in one click and get fast, uninterrupted access with Azure Cosmos DB, a globally distributed, multi-model database service." A large, stylized blue planet with white heart-shaped continents and a ring is centered on the right. Below the header, a call-to-action button says "Try it for a limited time with no subscription or credit-card number required." A link to "Learn more about Azure Cosmos DB" is also present. The main body of the page is white and contains five sections, each with a "Create >" button:

- SQL**: Azure Cosmos DB natively supports document data model with familiar SQL API. [Create >](#)
- MongoDB**: Azure Cosmos DB offers MongoDB API as a service at the protocol level. [Create >](#)
- Table**: Azure Cosmos DB offers native support for key-value pairs data with Azure Table API. [Create >](#)
- Graph**: Azure Cosmos DB offers native support for graphs with Apache Gremlin API. [Create >](#)
- Cassandra**: Azure Cosmos DB offers Cassandra API as a service at the protocol level. [Create >](#)

Free Database

Free Database

Free Database

1000 RU/s

25-Gb

Serverless

Thank You!

✉️ chadgreen@chadgreen.com

.twitch TaleLearnCode

🌐 TaleLearnCode.com

🐦 ChadGreen & TaleLearnCode

linkedin ChadwickEGreen

