



# The Hitchhiker's Guide to the Cosmos

Chad Green

Atlanta Code Camp  
September 14, 2019

@chadgreen



## Platinum Sponsors

---



# Magenic®

CODINGBLOCKS.NET



greater sum

## Gold Sponsors

# Special Thanks

---



KENNESAW STATE  
UNIVERSITY

# Who is Chad Green

Director of Software Development  
ScholarRx



[chadgreen@chadgreen.com](mailto:chadgreen@chadgreen.com)

 [chadgreen.com](http://chadgreen.com)

ChadGreen

ChadwickEGreen



@chadgreen

A portrait of a young man with long brown hair and a beard, looking off to the side. He is wearing a black button-down shirt and a green backpack. The background is black.

# What is Cosmos DB

@chadgreen

# Azure Cosmos DB

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

Turnkey global  
distribution



@chadgreen

# Azure Cosmos DB

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

Turnkey global  
distribution

Turnkey global  
distribution

@chadgreen

# Azure Cosmos DB

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

Elastic scalability  
of storage & throughput

Turnkey global distribution

Comprehensive SLAs

@chadgreen

# Azure Cosmos DB

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

Guaranteed low latency  
of fast storage & throughput

Turnkey global distribution

Elastic scale out  
of storage & throughput

Comprehensive SLAs

@chadgreen

# Azure Cosmos DB

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

~~Guaranteed low latency  
at the 99<sup>th</sup> percentile~~

Turnkey global distribution

Elastic scale out  
of storage & throughput

Guaranteed low latency  
at the 99<sup>th</sup> percentile

Comprehensive SLAs

@chadgreen

# Azure Cosmos DB

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

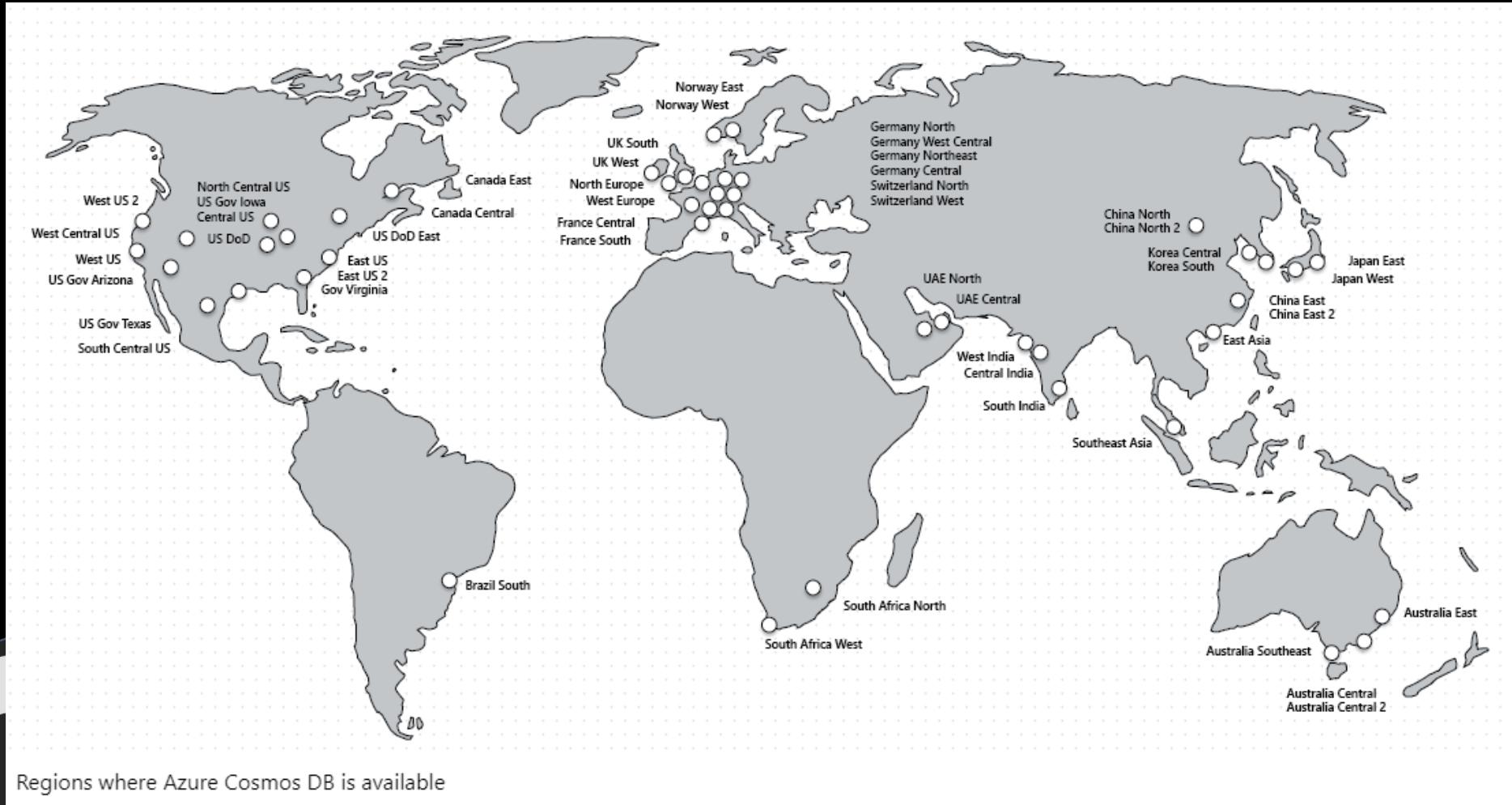
## Benefits & Features



# Azure Cosmos DB

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

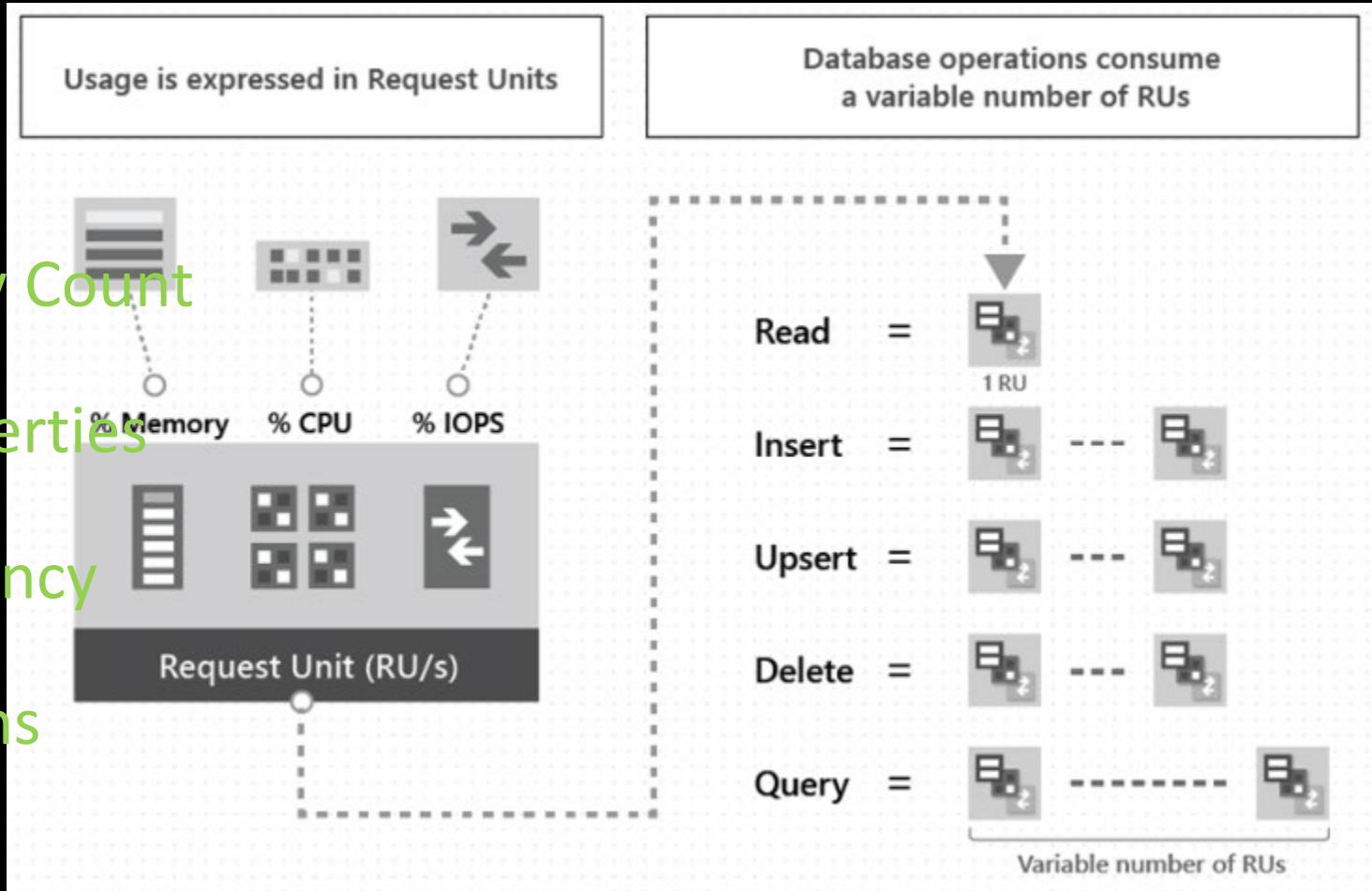
Setup multiple failover regions ready



@chadgreen

# Azure Cosmos DB Request Units

- Item Size
- Item Indexing
- Item Property Count
- Indexed Properties
- Data Consistency
- Query Patterns
- Script Usage



# Azure Cosmos DB Pricing

Unit	Price
Provisioned Throughput (multiple region writes) per 100 RU/s	\$0.016/hour
Provisioned Throughput (single region writes) per 100 RU/s	\$0.008/hour
SSD Storage (per GB)	\$0.25 GB/month

Starts at approximately \$23.61/month

Save 15-65% with Reserved Pricing

# Azure Cosmos Capabilities

what if we have  
REALY large data  
requirements?



# Azure Cosmos Capabilities

Resource	Default Limit
Maximum RUs per container	1,000,000
Maximum RUs per database	1,000,000
Maximum RUs per (logical) partition key	10,000
Maximum storage across all items per (logical) partition key	10 GB
Maximum number of distinct (logical) partition keys	Unlimited
Maximum storage per container	Unlimited
Maximum storage per database	Unlimited

# Cosmos Use Cases



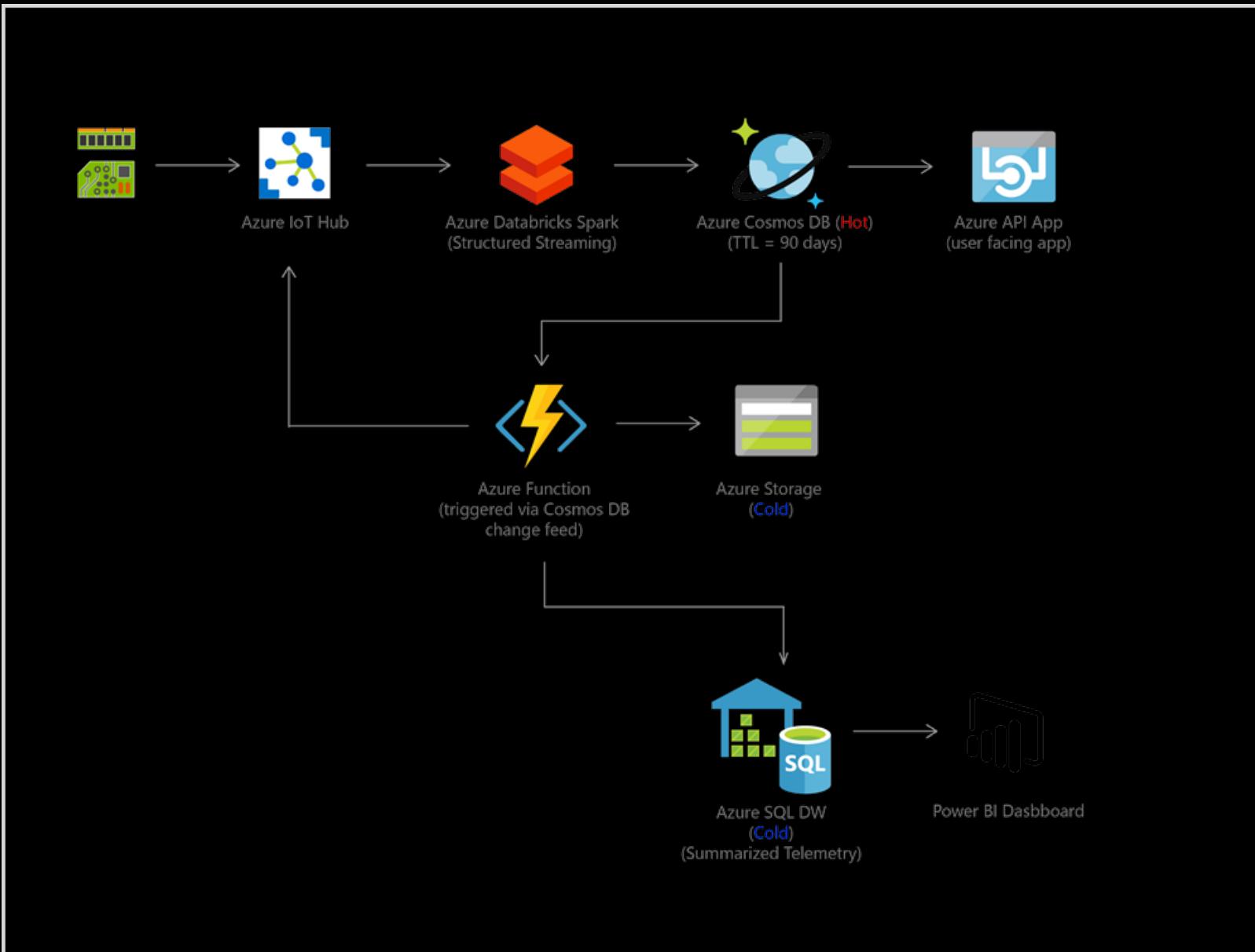
@chadgreen

# IoT and Telematics

Common Pattern in IoT use cases

- Ingest bursts of data from devices and sensors of various locales
- Process and analyze streaming data to derive real-time insights
- Archive data to cold storage for batch analytics

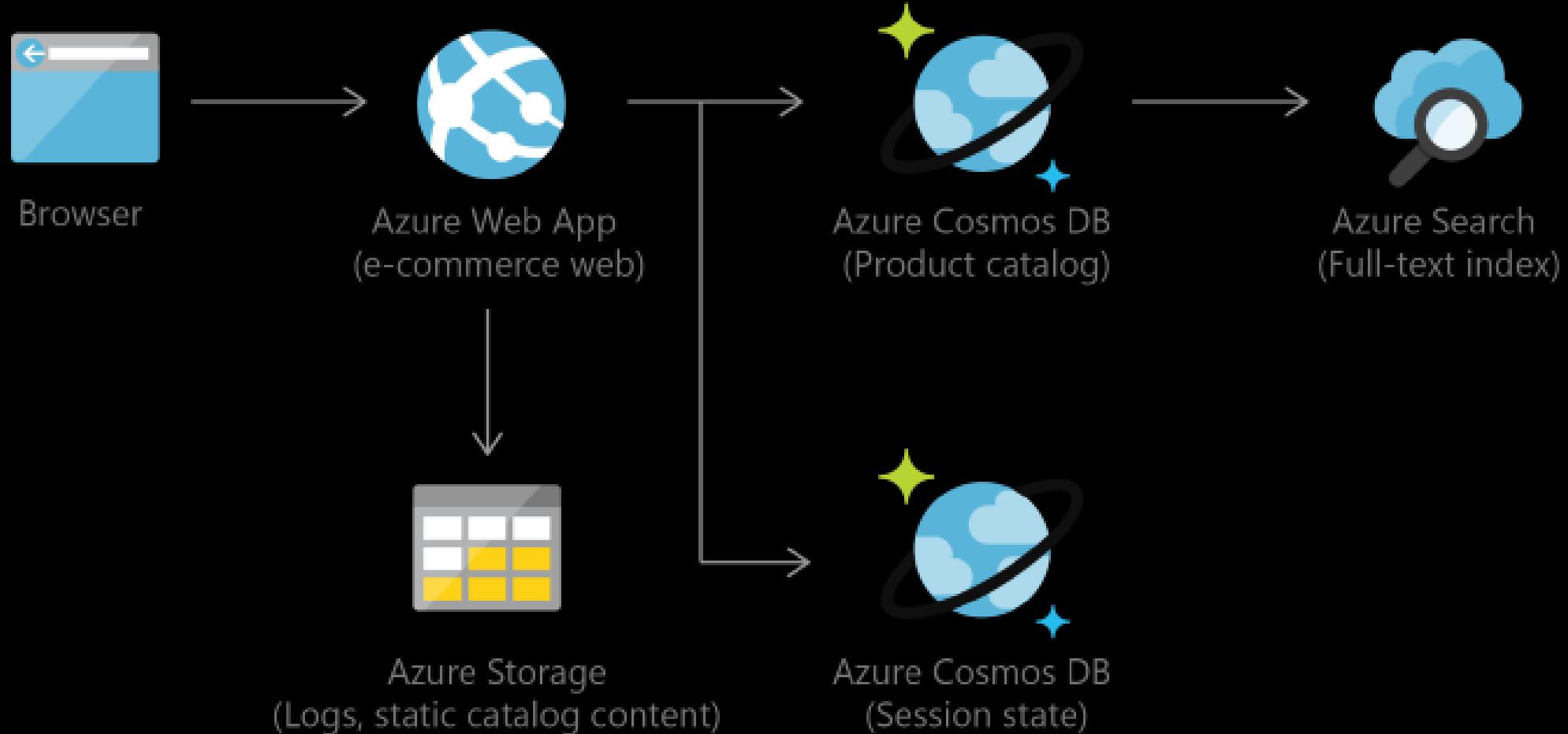
# IoT and Telematics



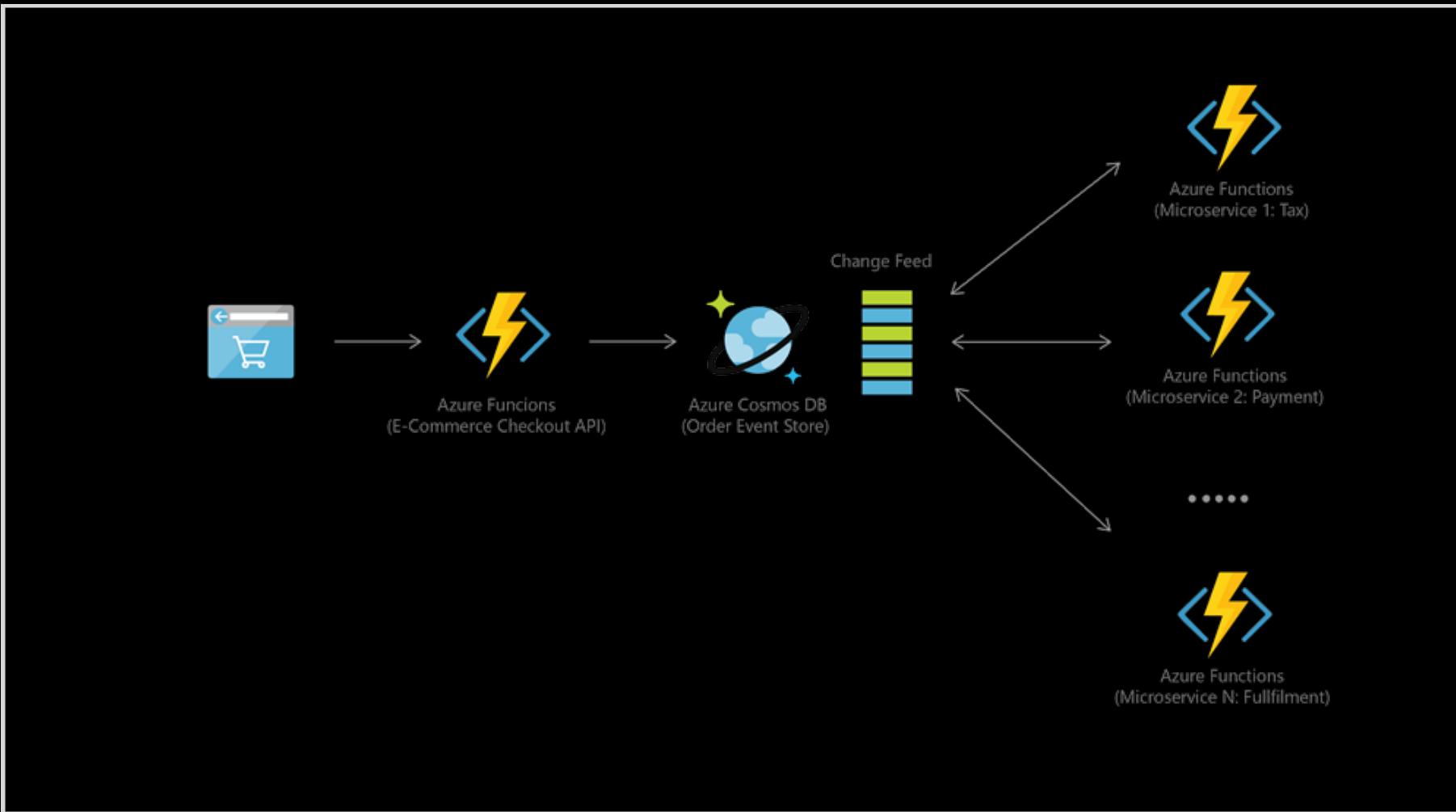
# Retail and Marketing

- Used extensively by Microsoft's own e-commerce platforms
- Storing and querying a set of attributes for entities
- Examples of catalog data
  - User Accounts
  - Product Catalogs
  - IoT Device Registries

# Retail and Marketing



# Retail and Marketing

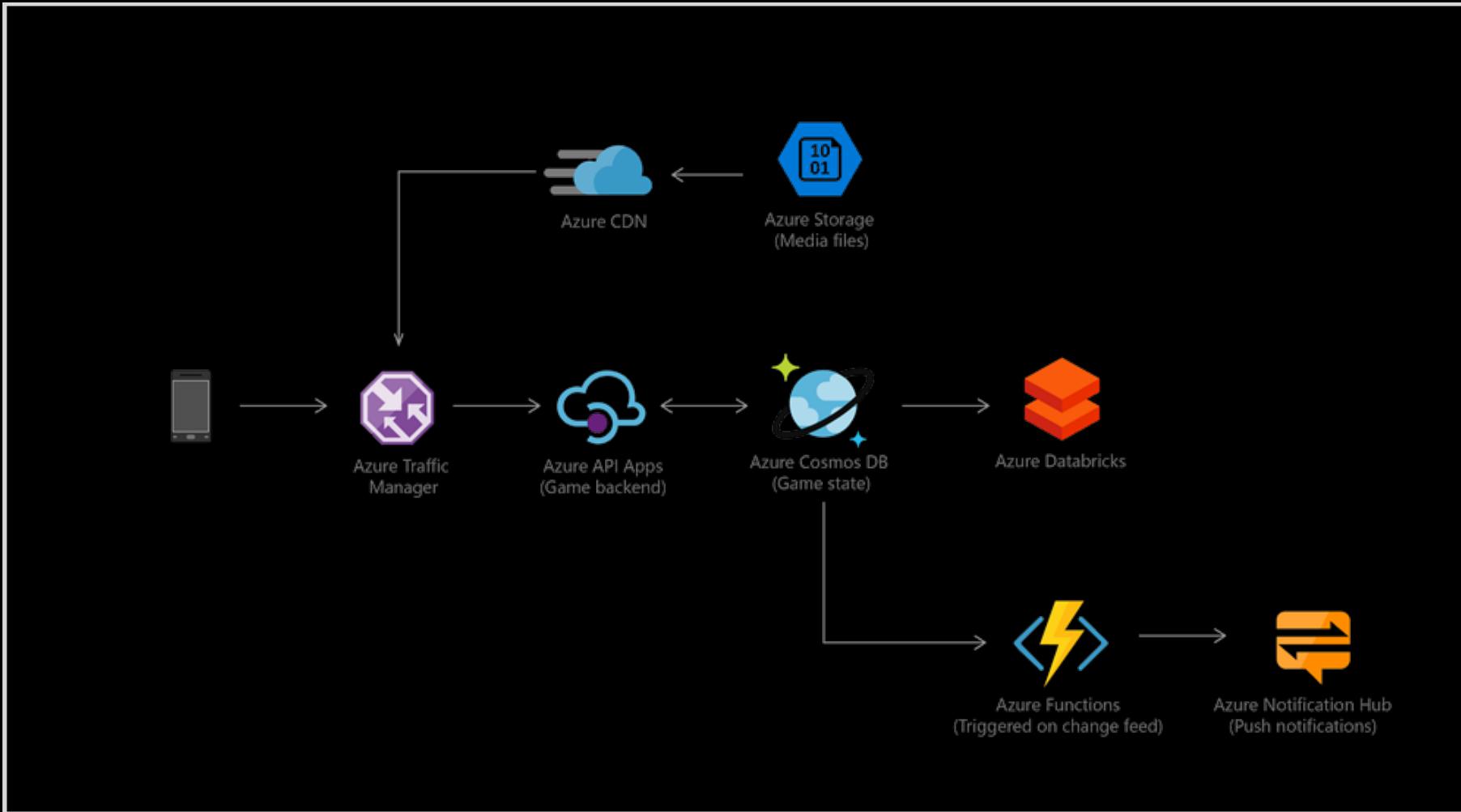


# Gaming

- Database tier is crucial
- Often require single-millisecond latencies for reads and writes
- Needs to be fast and able to handle massive spikes



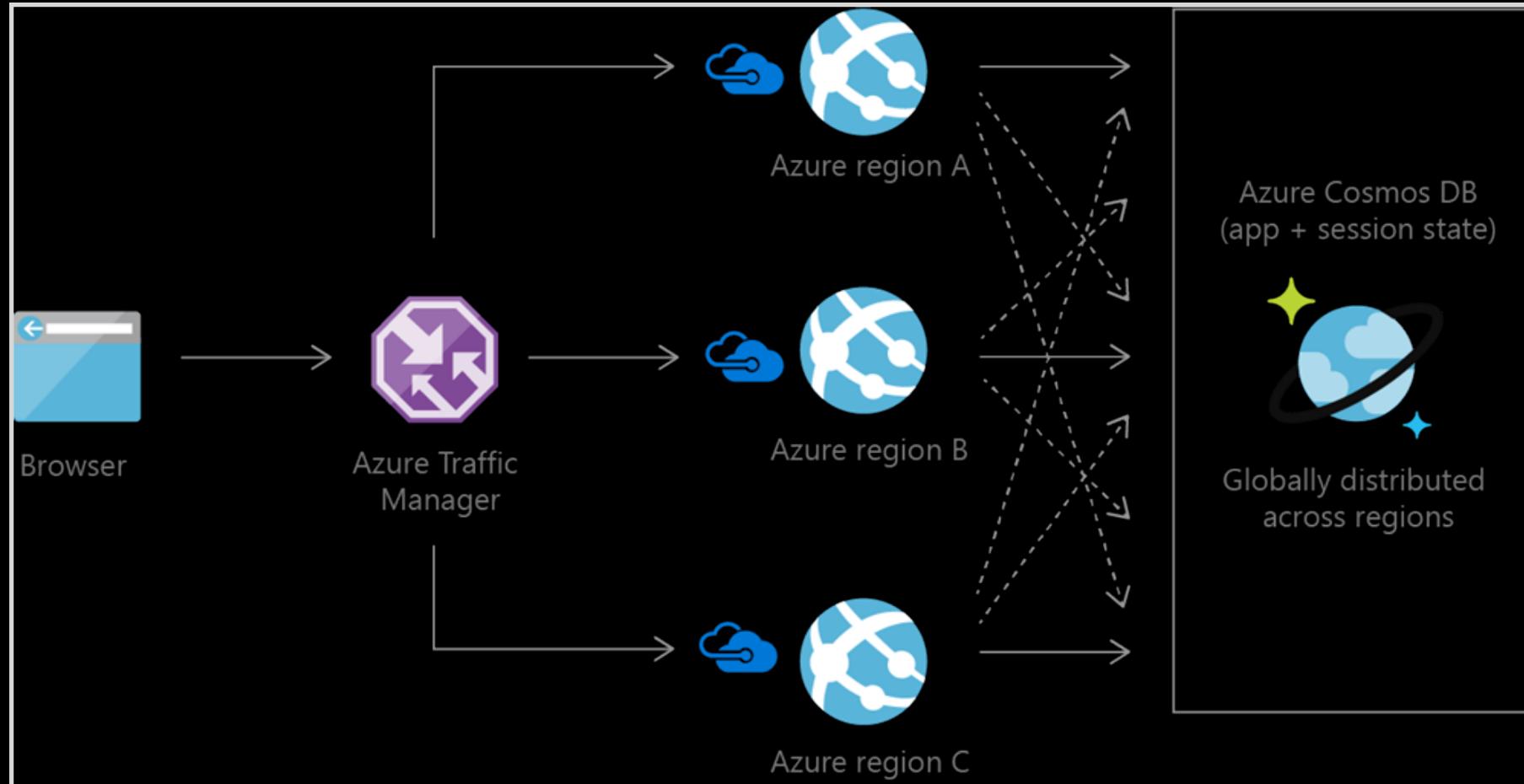
# Gaming



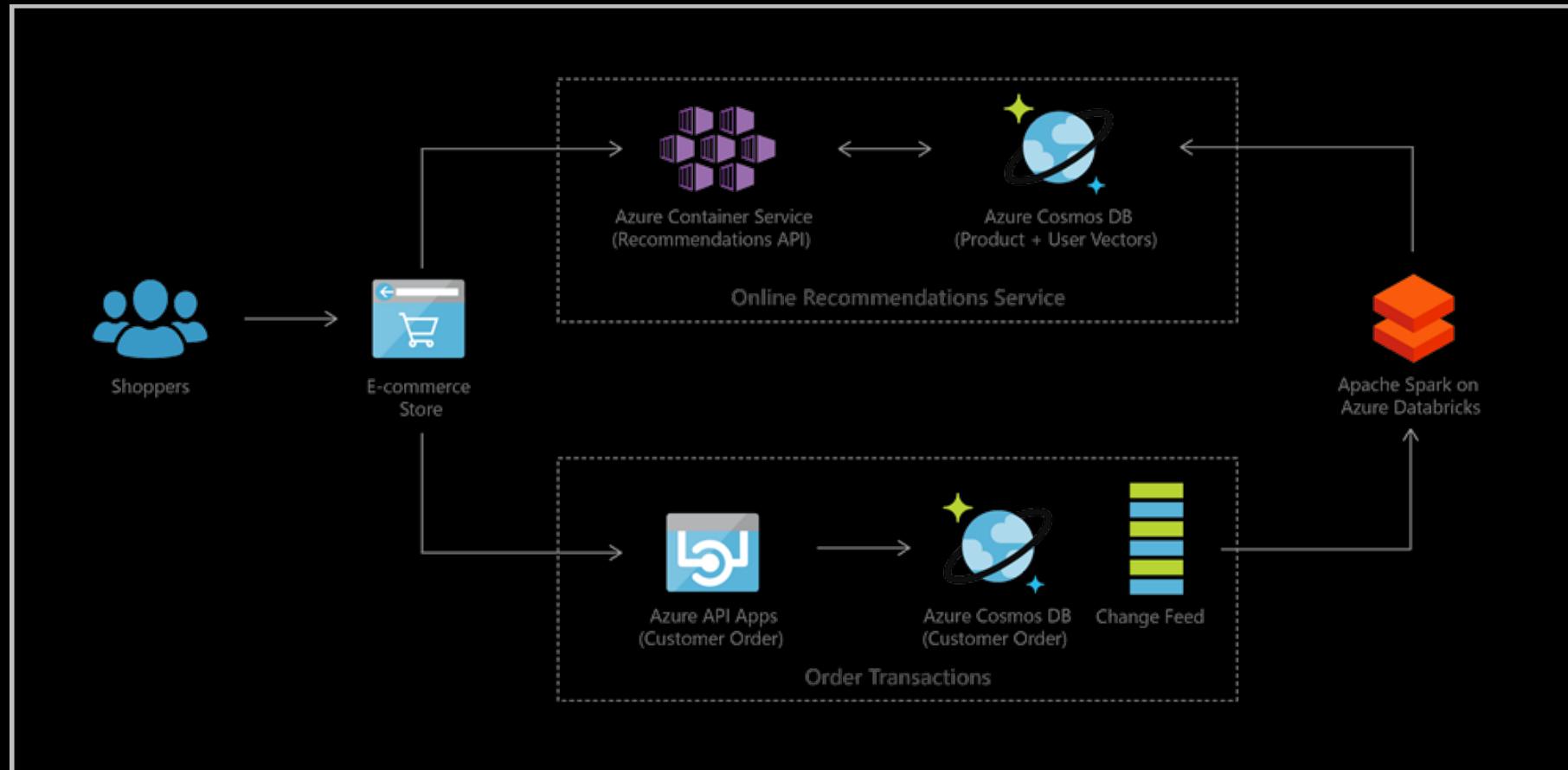
# Web & Mobile Applications

- Modeling social interactions
- Integrating with third-party services
- Building rich personalized experiences

# Web & Mobile Applications – Social Applications



# Web & Mobile Applications – Personalization

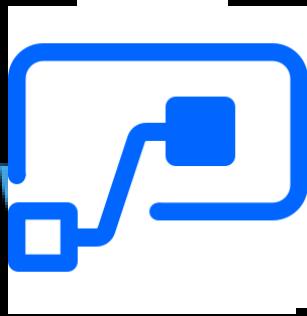




# Integrations

@chadgreen

# Cosmos DB Integrations

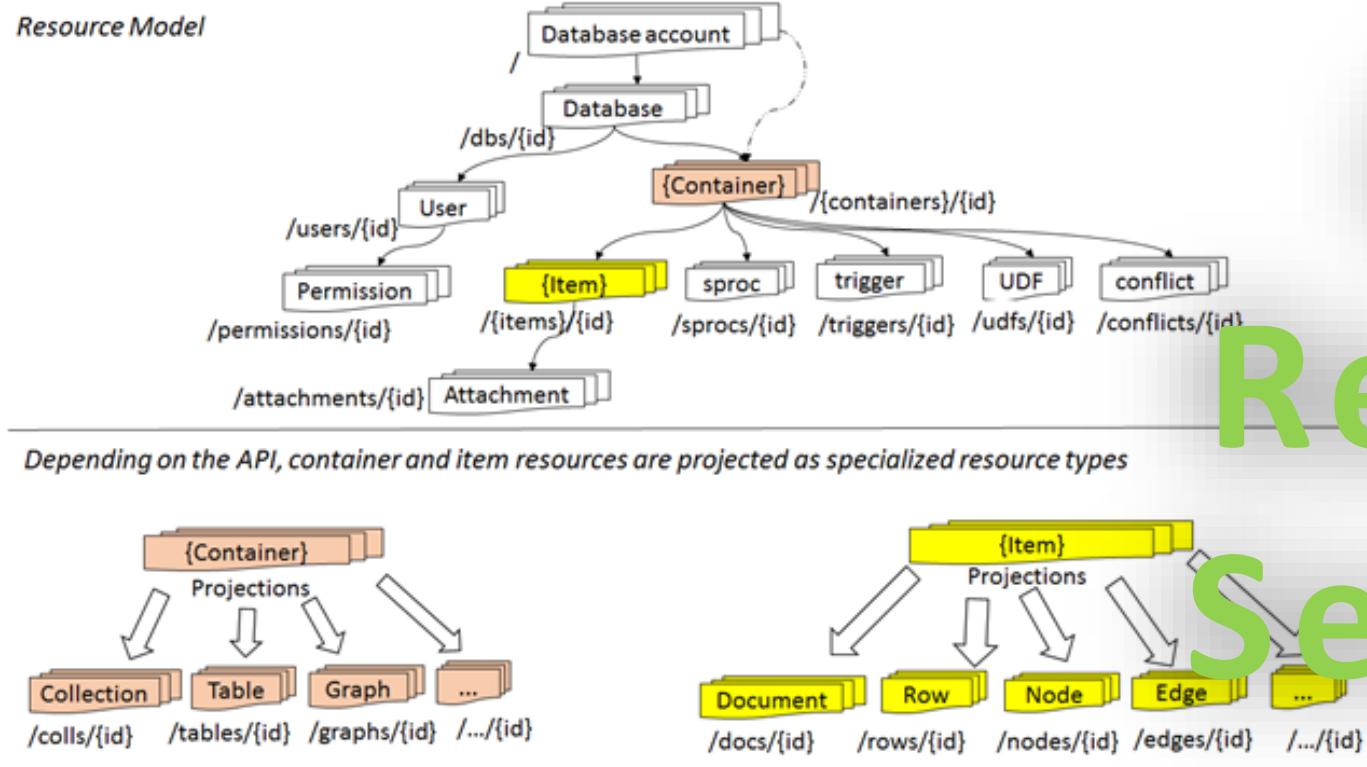


Abstractions for Apps

# Navigating the 5 API Models



# Resource Model and API Projections



Atom  
Resource  
Sequence



# SQL API Document Database

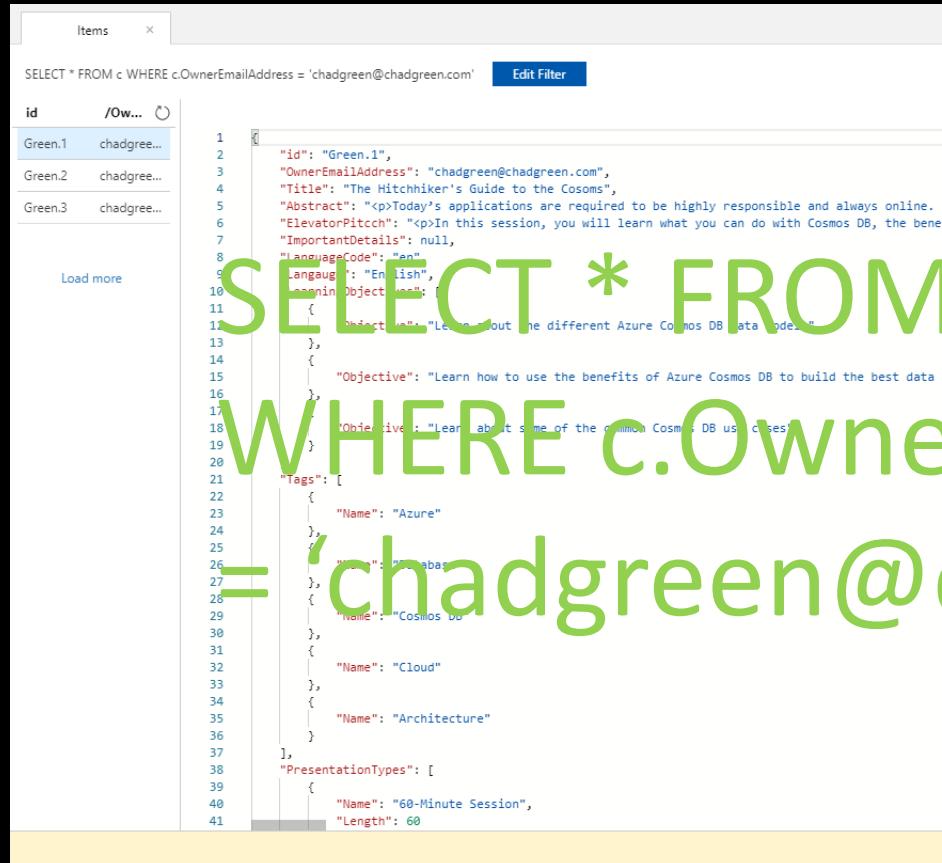
# SQL API – What

- Document Database
- Originally Microsoft's DocumentDB implementation
- Supports using SQL as a JSON query language
- Uses JavaScript's programming model as foundation for query language

## SQL API – Why

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

# SQL API – How: Data Model



The screenshot shows the Azure Cosmos DB SQL API results interface. At the top, there's a search bar with the query: "SELECT \* FROM c WHERE c.OwnerEmailAddress = 'chadgreen@chadgreen.com'". Below the search bar, there's a "Edit Filter" button. The results table has columns for "id" and "OwnerEmailAddress". Three items are listed: Green.1, Green.2, and Green.3. Each item row contains a "Load more" link. To the right of the table, the raw JSON response is displayed, showing the details of each item.

```
1  [
2    {
3      "id": "Green.1",
4      "OwnerEmailAddress": "chadgreen@chadgreen.com",
5      "Title": "The Hitchhiker's Guide to the Cosmos",
6      "Abstract": "<p>Today's applications are required to be highly responsible and always online. (",
7      "ElevatorPitch": "<p>In this session, you will learn what you can do with Cosmos DB, the benefits of using it, and how to build your own data models using the SQL API.</p>",
8      "ImportantDetails": null,
9      "LanguageCode": "en",
10     "Language": "English",
11     "minObject": [
12       {
13         "Objective": "Learn about the different Azure Cosmos DB data models"
14       },
15       {
16         "Objective": "Learn how to use the benefits of Azure Cosmos DB to build the best data structures"
17       }
18     ],
19     "Objective": "Learn about some of the common Cosmos DB use cases"
20   },
21   {
22     "Tags": [
23       {
24         "Name": "Azure"
25       },
26       {
27         "Name": "Database"
28       },
29       {
29         "Name": "Cosmos DB"
30       },
31       {
32         "Name": "Cloud"
33       },
34       {
35         "Name": "Architecture"
36       }
37     ],
38     "PresentationTypes": [
39       {
40         "Name": "60-Minute Session",
41         "Length": 60
42       }
43     ]
44   }
45 ]
```

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

# SQL API – How: Insert

```
18 references | Chad Green, 1 day ago | 1 author, 1 change
public class Presentation
{
    [JsonProperty(PropertyName = "id")]
    6 references | Chad Green, 1 day ago | 1 author, 1 change
    public string Id { get; set; }

    5 references | Chad Green, 1 day ago | 1 author, 1 change
    public string OwnerEmailAddress { get; set; }

    4 references | Chad Green, 1 day ago | 1 author, 1 change
    public string Title { get; set; }

    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public string Abstract { get; set; }

    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public string ElevatorPitch { get; set; }

    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public string ImportantDetails { get; set; }

    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public string LanguageCode { get; set; }

    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public string Language { get; set; }

    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public LearningObjective[] LearningObjectives { get; set; }

    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public Tag[] Tags { get; set; }

    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public PresentationType[] PresentationTypes { get; set; }

}
```

# SQL API – How: Insert

```
/// <summary> Add the specified existing presentation to the database (if it doe ...
3 references | Chad Green, 1 day ago | 1 author, 1 change
private async Task AddPresentation(ExistingPresentation existingPresentation)
{
    Presentation presentation = Generate.Presentation(existingPresentation);

    try
    {
        // Read the item to see if it exists. ReadItemAsync will throw an exception if the item does not exist and return status code 404 (Not found).
        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)
    {
        // Create an item in the container representing the presentation. Note we provide the value of the partition key for this item, which is "Andersen"
        ItemResponse<Presentation> presentationResponse = await this.container.CreateItemAsync<Presentation>(presentation, new PartitionKey(presentation.OwnerEmailAddress));

        // Note that after creating the item, we can access the body of the item with the Resource property off the ItemResponse.
        // We can also access the RequestCharge property to see the amount of RUs consumed on this request.
        Console.WriteLine("Created item in database with id: {0} Operation consumed {1} RUs.\n", presentationResponse.Resource.Id, presentationResponse.RequestCharge);
    }
}
```

# SQL API – How: Query

```
/// <summary> Query the database to get the details about the presentation
1 reference | Chad Green, 1 day ago | 1 author, 1 change
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}");
        }
    }
    Console.WriteLine();
}
```

# SQL API – Query

- API (using Microsoft.Azure.Cosmos)

- LINQ to SQL API

- JavaScript

- Stored Procedures

- Triggers

- User Defined Functions

- Entity Framework

{LEAF}

MongoDB  
Document Database

# API for MongoDB – What

- Native MongoDB implementation
- Allows existing client SDKs, drivers, and tools to interact transparently
- Default is Mongo v3.2; v3.4 in preview



API for MongoDB – Why

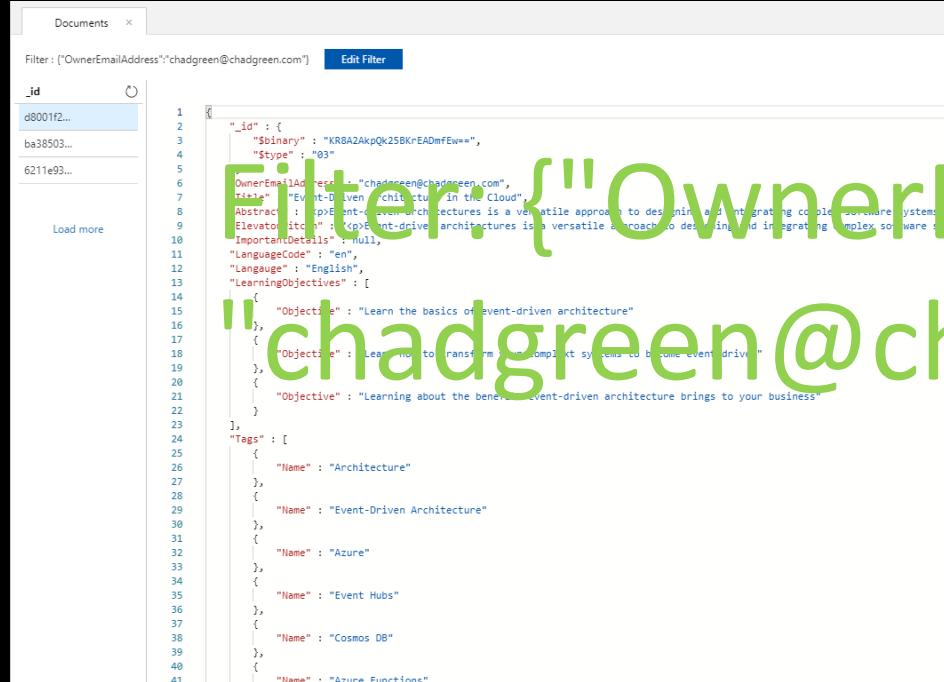
Migrating data from a

MongoDB database to Azure

Cosmos DB's fully managed

service

# API for MongoDB – How: Data Model



The screenshot shows the MongoDB Compass interface. On the left, a list of document IDs is visible: d8001f2..., ba38503..., and 6211e93... A 'Load more' button is present below the list. On the right, a detailed view of a single document is shown. The document has the following structure:

```
_id : {
  "$binary" : "KR8A2AkpQk25BK+EDmFEw==",
  "$tstype" : "03"
},
OwnerEmailAddress : "chadgreen@chadgreen.com",
Title : "Event-Driven Architecture in the Cloud",
Abstract : "Event-driven architecture is a versatile approach to designing and integrating complex software systems. It allows for loose coupling between components, making it easier to scale and maintain. This article explores the basics of event-driven architecture and how it can be applied to modern cloud-based systems.",
ElevateItcID : "(p)Event-driven architectures is a versatile approach to designing and integrating complex software systems. It allows for loose coupling between components, making it easier to scale and maintain. This article explores the basics of event-driven architecture and how it can be applied to modern cloud-based systems.",
ImportDetails : null,
LanguageCode : "en",
Language : "English",
LearningObjectives : [
  {
    "Objective": "Learn the basics of event-driven architecture"
  },
  {
    "Objective": "Learn how to transform complex systems to become event-driven"
  },
  {
    "Objective": "Learning about the benefits event-driven architecture brings to your business"
  }
],
Tags : [
  {
    "Name" : "Architecture"
  },
  {
    "Name" : "Event-Driven Architecture"
  },
  {
    "Name" : "Azure"
  },
  {
    "Name" : "Event Hubs"
  },
  {
    "Name" : "Cosmos DB"
  },
  {
    "Name" : "Azure Functions"
  }
]
```

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

# API for MongoDB – How: Insert

```
16 references | Chad Green, 1 day ago | 1 author, 1 change
public class Presentation
{
    [BsonId(IdGenerator =typeof(CombGuidGenerator))]
    0 references | Chad Green, 1 day ago | 1 author, 1 change
    public Guid Id { get; set; }

    [BsonElement("OwnerEmailAddress")]
    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public string OwnerEmailAddress { get; set; }

    [BsonElement("Title")]
    4 references | Chad Green, 1 day ago | 1 author, 1 change
    public string Title { get; set; }

    [BsonElement("Abstract")]
    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public string Abstract { get; set; }

    [BsonElement("ElevatorPitch")]
    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public string ElevatorPitch { get; set; }

    [BsonElement("ImportantDetails")]
    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public string ImportantDetails { get; set; }

    [BsonElement("LanguageCode")]
    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public string LanguageCode { get; set; }

    [BsonElement("Langauge")]
    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public string Langauge { get; set; }

    [BsonElement("LearningObjectives")]
    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public LearningObjective[] LearningObjectives { get; set; }

    [BsonElement("Tags")]
    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public Tag[] Tags { get; set; }

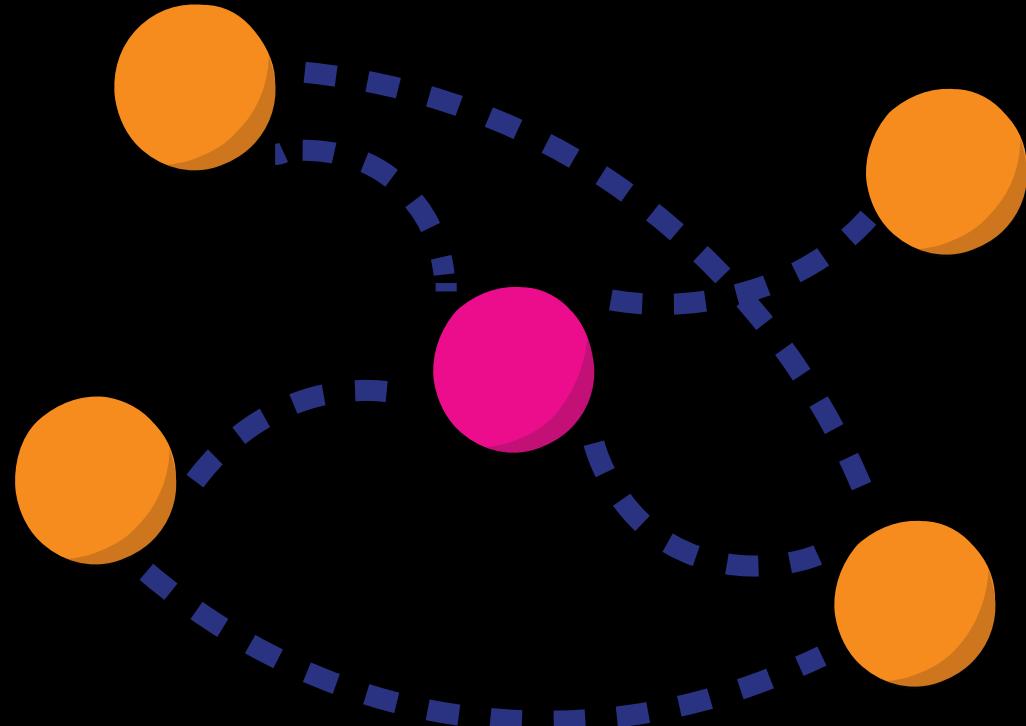
    [BsonElement("PresentationTypes")]
    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public PresentationType[] PresentationTypes { get; set; }
}
```

# API for MongoDB – How: Insert

```
3 references | Chad Green, 1 day ago | 1 author, 1 change
public void CreatePresentation(Presentation presentation)
{
    var collection = GetPresentationsCollectionForEdit();
    try
    {
        collection.InsertOne(presentation);
    }
    catch (MongoCommandException ex)
    {
        string msg = ex.Message;
    }
}
```

# API for MongoDB – How: Query

```
1 reference | Chad Green, 1 day ago | 1 author, 1 change
public List<Presentation> GetAllPresentations()
{
    try
    {
        var collection = GetPresentationsCollection();
        return collection.Find(new BsonDocument()).ToList();
    }
    catch (MongoConnectionException)
    {
        return new List<Presentation>();
    }
}
```



# Gremlin

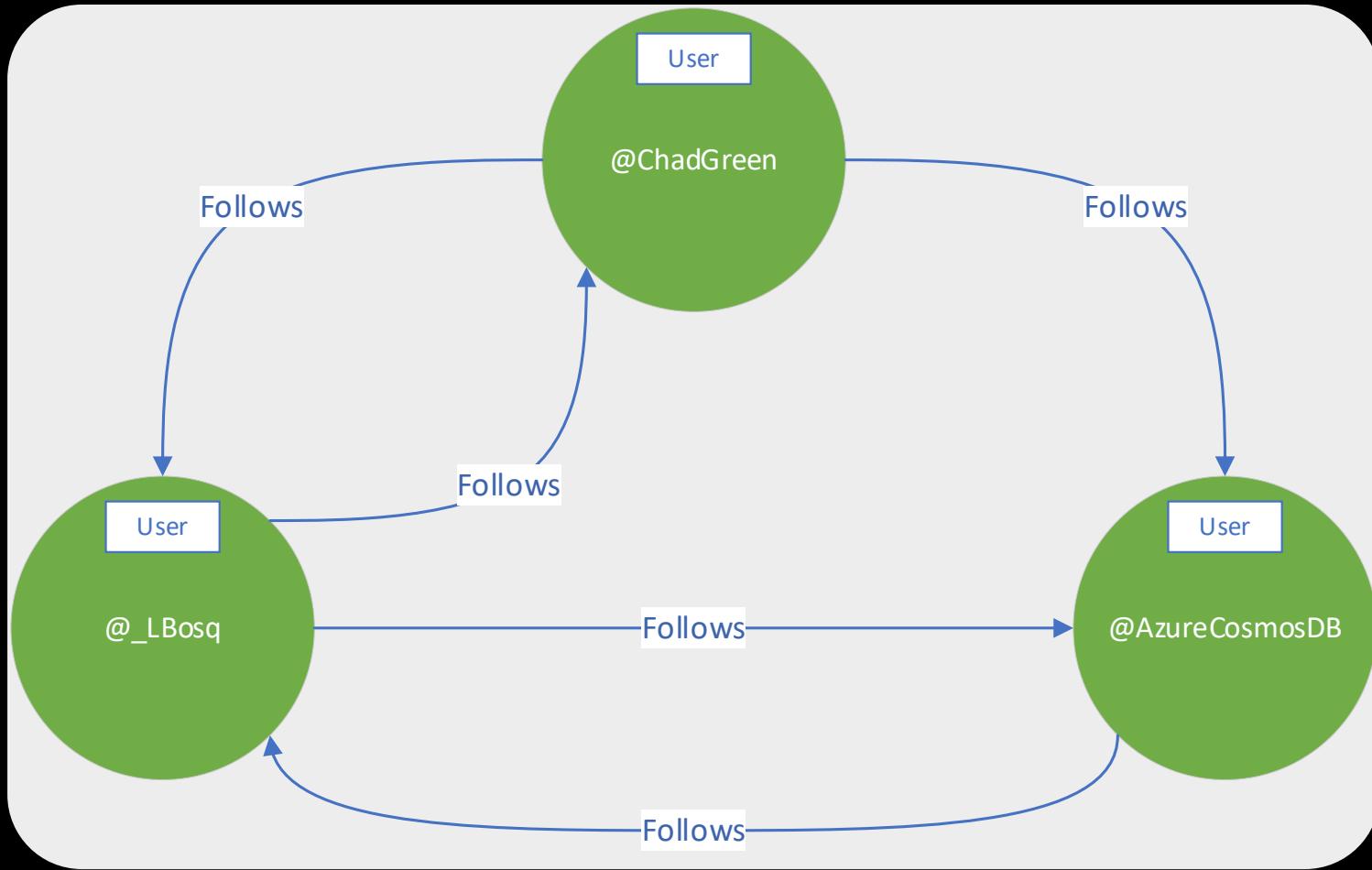
# API

# Graph Database

# Gremlin API – What

- Collection of *vertices* and *edges*
- Represents entities as vertices and the ways in which those entities relate to the world as relationships
- Allows us to model all kinds of scenarios

# Gremlin API – What



Follows

@chadgreen

# Gremlin API – Why

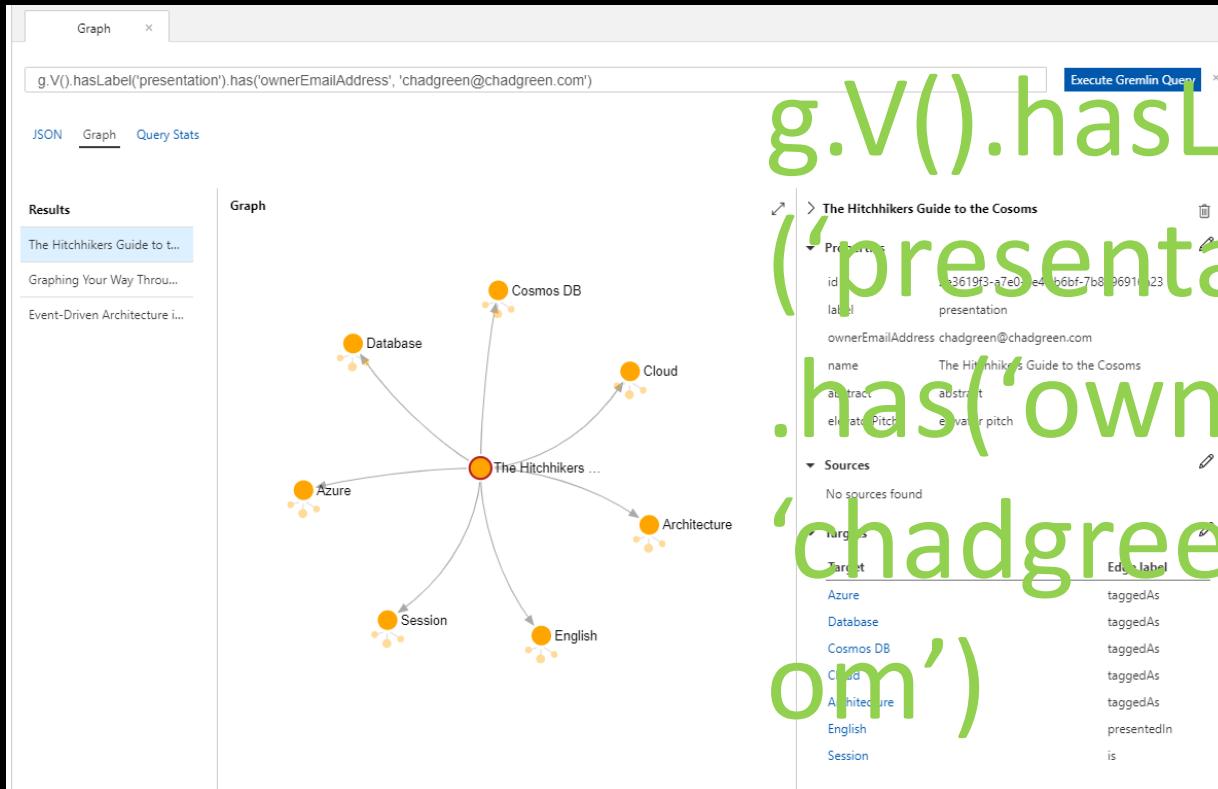
Building a graph database to  
model and traverse  
relationships among entities

# Gremlin API – Why

- Social Networks
- Search
- Recommendations
- Communication networks
- Identity and access management
- Fraud detection

Represent  
data as it  
is found in  
nature

# Gremlin API – How: Data Model



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

# SQL API – How: Insert

```
"g.addV('presentation').property('ownerEmailAddress', 'chadgreen@chadgreen.com').property('name', 'The Hitchhikers Guide to the Cosoms').property('abstract', 'abstract').property('elevato

1 reference | Chad Green, 1 day ago | 1 author, 1 change
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!");

        // Print the Gremlin status code.
        Console.WriteLine($"\"{e.StatusCode}\"\n");

        // On error, ResponseException.StatusAttributes will include the common StatusAttributes for successful requests, as well as
        // additional attributes for retry handling and diagnostics.
        // These include:
        //   x-ms-retry-after-ms      : The number of milliseconds to wait to retry the operation after an initial operation was throttled. This will be populated when
        //                               : attribute 'x-ms-status-code' returns 429.
        //   x-ms-activity-id        : Represents a unique identifier for the operation. Commonly used for troubleshooting purposes.
        PrintStatusAttributes(e.StatusAttributes);
        Console.WriteLine($"\"{e.StatusAttributes["x-ms-retry-after-ms"]}\": { GetValueAsString(e.StatusAttributes, "x-ms-retry-after-ms")}\n");
        Console.WriteLine($"\"{e.StatusAttributes["x-ms-activity-id"]}\": { GetValueAsString(e.StatusAttributes, "x-ms-activity-id")}\n");

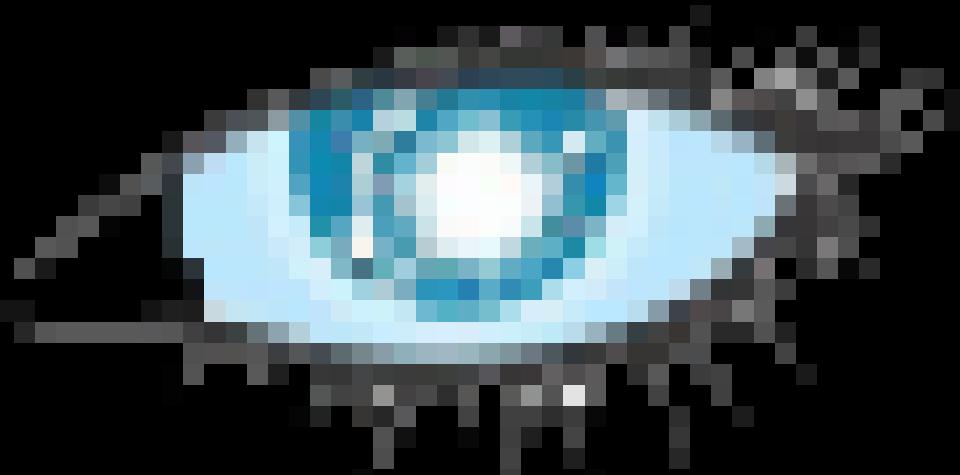
        throw;
    }
}
```

# SQL API – How: Query

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

# Gremlin API – Query

- API (using Gremlin.Net)
- Community API (Gremlin.Net.CosmosDb)
- JavaScript
  - Stored Procedures
  - Triggers
  - User Defined Functions
- SQL API



# Cassandra API Wide Column Store

# Cassandra API – What

- Open-source, distributed, wide column store, NoSQL database
- Designed to handle large amounts of data
- Uses many commodity servers, providing high availability
- Developed to power the Facebook inbox search feature

Cassandra API – Why

Migrating data from Cassandra  
to Azure Cosmos DB

# Cassandra API – How: Data Model

The screenshot shows a search interface for a Cassandra database. At the top, there is a header with the word "Rows" and a close button "X". Below this is a search bar. The main area has a red box highlighting a search query builder. The builder includes fields for "Action" (with a plus sign "+"), "And" (with a checkbox), "Field" (set to "presentation\_owr"), "Type" (set to "Text"), "Operator" (set to "="), and "Value" (set to "chadgreen@chadgreen.com"). Below the builder are two buttons: "+ Add new clause" and "► Advanced Options". Further down, there is a search input field containing "presentation\_abstract" with a dropdown arrow. The results section displays three rows of search results, each starting with "<p>". The first result discusses data connectivity, the second mentions event-driven architectures, and the third speaks about today's highly responsible applications.

<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>Da
<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

```
14 references | Chad Green, 1 day ago | 1 author, 1 change
public class Presentation
{
    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public string presentation_id { get; set; }

    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public string presentation_owneremailaddress { get; set; }

    4 references | Chad Green, 1 day ago | 1 author, 1 change
    public string presentation_title { get; set; }

    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public string presentation_abstract { get; set; }

    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public string presentation_elevatorpitch { get; set; }

    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public string presentation_importantinformation { get; set; }

    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public string presentation_languagecode { get; set; }

    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public string presentation_language { get; set; }

}
```

# Cassandra API – How: Insert

```
// Connect to cassandra cluster (Cassandra API on Azure Cosmos DB supports only TLSv1.2)
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();

// Creating KeySpace and table
session.Execute("DROP KEYSPACE IF EXISTS speakingengagements");
session.Execute("CREATE KEYSPACE speakingengagements WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'datacenter1' : 1 };");
Console.WriteLine(String.Format("created keyspace speakingengagements"));
session.Execute("CREATE TABLE IF NOT EXISTS speakingengagements.presentation (presentation_id text PRIMARY KEY, presentation_owneremailaddress text, presentation_name text)");
Console.WriteLine(String.Format("created table Presentation"));

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

// Inserting Data into presentation table
mapper.Insert<Presentation>(Generate.Presentation.ExistingPresentation.EventDrivenArchitectureInTheCloud));
//mapper.Insert<Presentation>(Generate.Presentation.ExistingPresentation.GraphingYourWayThroughTheCosoms));
mapper.Insert<Presentation>(Generate.Presentation.ExistingPresentation.TheHitchhikersGuideToTheCosoms));
Console.WriteLine("Inserted data into presetnation table");

Console.WriteLine("Select ALL");
Console.WriteLine("-----");
foreach (Presentation presentation in mapper.Fetch<Presentation>("Select * from presentation"))
{
    Console.WriteLine(presentation);
}
```

# Cassandra API – How: Query

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

# Cassandra API – Query

- API (using CassandraCSharpDriver)
- Cassandra Query Language (CQL)
- Cassandra-based tools (like cqlsh)

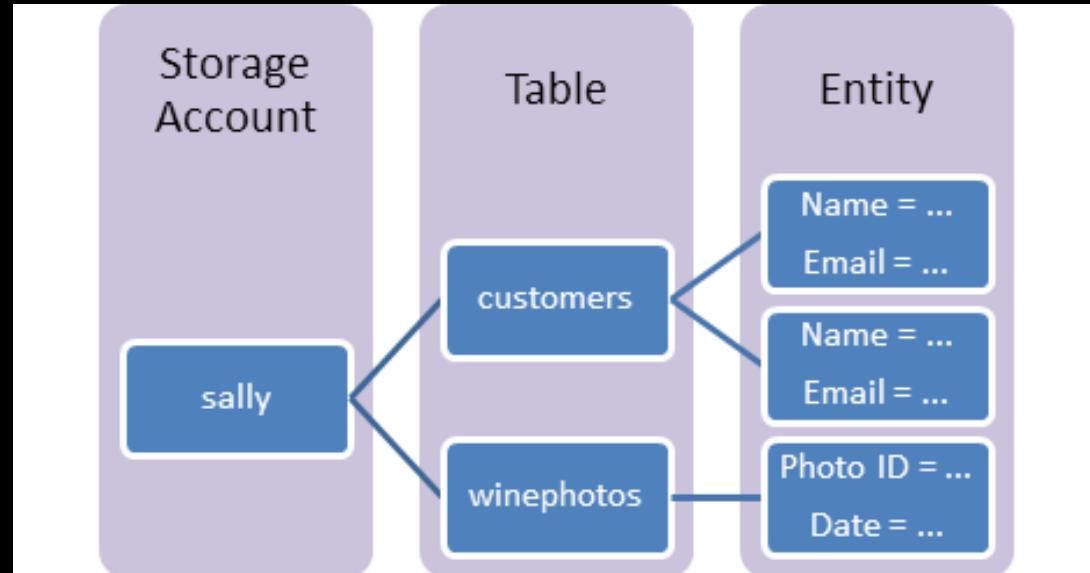


# Table API

## Table Storage

# Table API – What

- Stores large amounts of structured data
- Ideal for storing structured, non-relational data



## Table API – Why

Migrating data from Azure

Table storage to Cosmos DB

# Table API – How: Data Model

Entities X

Action	And/Or	Field	Type	Operator	Value
+	<input type="checkbox"/>	PartitionKey	String	=	chadgreen@chadgreen.com
+	<input checked="" type="checkbox"/> And	RowKey	String	=	Green.3
+ Add new clause					
<span style="font-size: small;">▶ Advanced Options</span>					
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

```
19 references | Chad Green, 2 days ago | 1 author, 1 change
public class Presentation : TableEntity
{

    0 references | Chad Green, 2 days ago | 1 author, 1 change
    public Presentation() { }

    3 references | Chad Green, 2 days ago | 1 author, 1 change
    public Presentation(string id, string ownerEmailAddress)
    {
        PartitionKey = ownerEmailAddress;
        RowKey = id;
    }

    5 references | Chad Green, 2 days ago | 1 author, 1 change
    public string Title { get; set; }

    3 references | Chad Green, 2 days ago | 1 author, 1 change
    public string Abstract { get; set; }

    3 references | Chad Green, 2 days ago | 1 author, 1 change
    public string ElevatorPitch { get; set; }

    3 references | Chad Green, 2 days ago | 1 author, 1 change
    public string ImportantDetails { get; set; }

    3 references | Chad Green, 2 days ago | 1 author, 1 change
    public string LanguageCode { get; set; }

    3 references | Chad Green, 2 days ago | 1 author, 1 change
    public string Langauge { get; set; }

}
```

# Table API – How: Insert

3 references | Chad Green, 2 days ago | 1 author, 1 change

```
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

```
1 reference | Chad Green, 2 days ago | 1 author, 1 change
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;
    }
}
```

# Call to Action



[azure.microsoft.com/en-us/try/cosmosdb/](https://azure.microsoft.com/en-us/try/cosmosdb/)

## Try Azure Cosmos DB for free

Take your database worldwide in one click and get fast, uninterrupted access with Azure Cosmos DB, a globally distributed, multi-model database service.

Try it for a limited time with no subscription or credit-card number required.

[Learn more about Azure Cosmos DB >](#)



### Select an API and data model to create a database

Once you create a database, you'll have 30 days of free access.



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 >](#)

@chadgreen

# Thank You

chadgreen@chadgreen.com

 chadgreen.com

ChadGreen

ChadwickEGreen



@chadgreen