The Hitchhiker's Guide to the Cosmos

Chad Green

Atlanta Code Camp
September 14, 2019

@chadgreen

Platinum Sponsors

Gold Sponsors

Special Thanks

KENNESAW STATE
UNIVERSITY

@chadgreen

Who is Chad Green

Director of Software Development
ScholarRx

chadgreen@chadgreen.com
chadgreen.com
ChadGreen
ChadwickEGreen

@chadgreen

What is Cosmos DB

@chadgreen

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

Azure Cosmos DB

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

Guaranteed low latency at the 99th percentile

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

Five well-defined consistency models

Elastic scale out of storage & throughput

Guaranteed low latency at the 99th percentile

Turnkey global distribution

Comprehensive SLAs

@chadgreen

Azure Cosmos DB

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

Five well-defined consistency models

Elastic scale out of storage & throughput

Guaranteed low latency at the 99th percentile

Turnkey global distribution

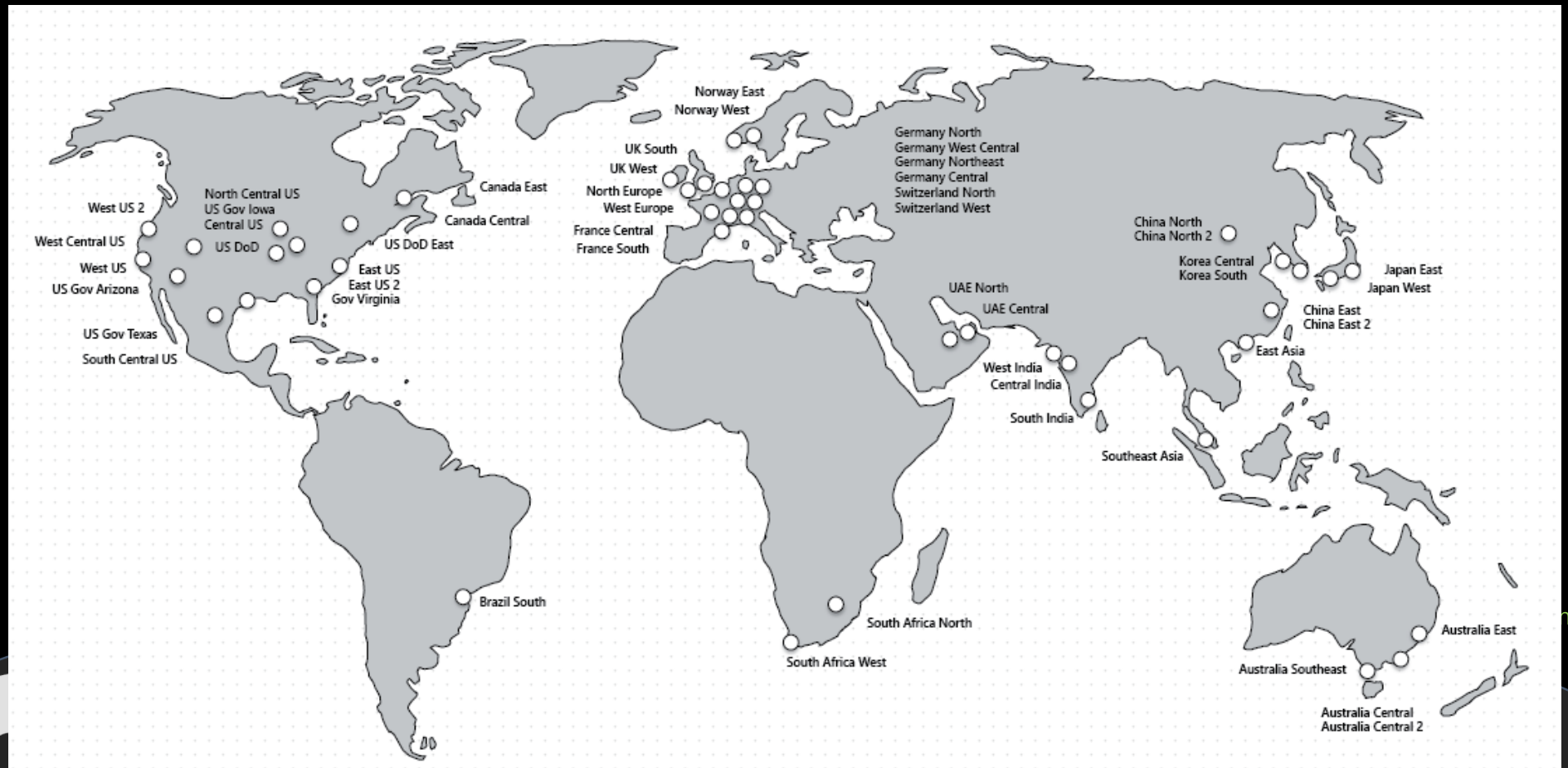
Comprehensive SLAs

@chadgreen

Azure Cosmos DB

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

Server-side programming



Regions where Azure Cosmos DB is available

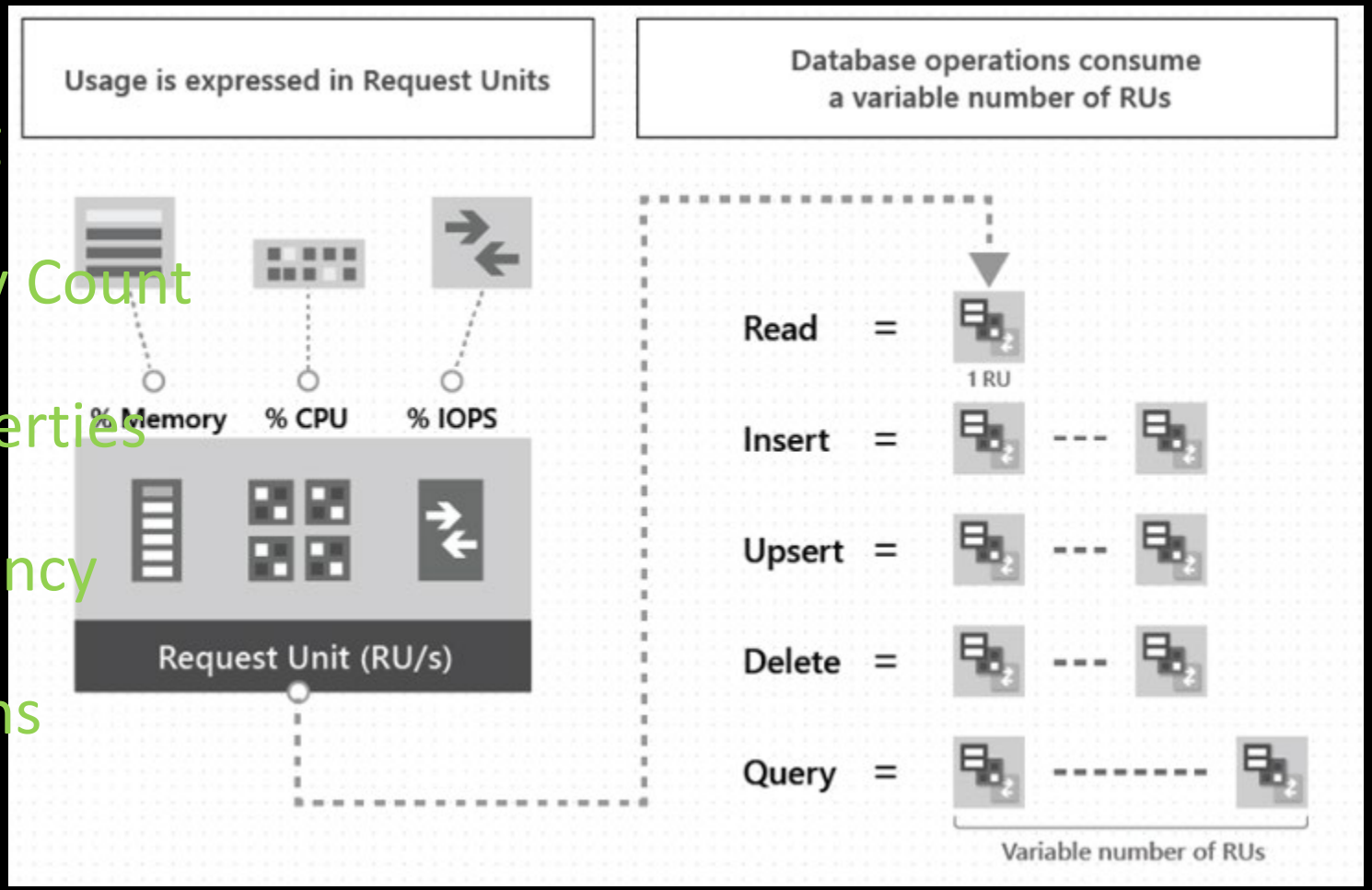
Turnkey global distribution

Comprehensive SLAs

@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?

XBOX LIVE

skype™

Office 365

Microsoft Azure

@chadgreen

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

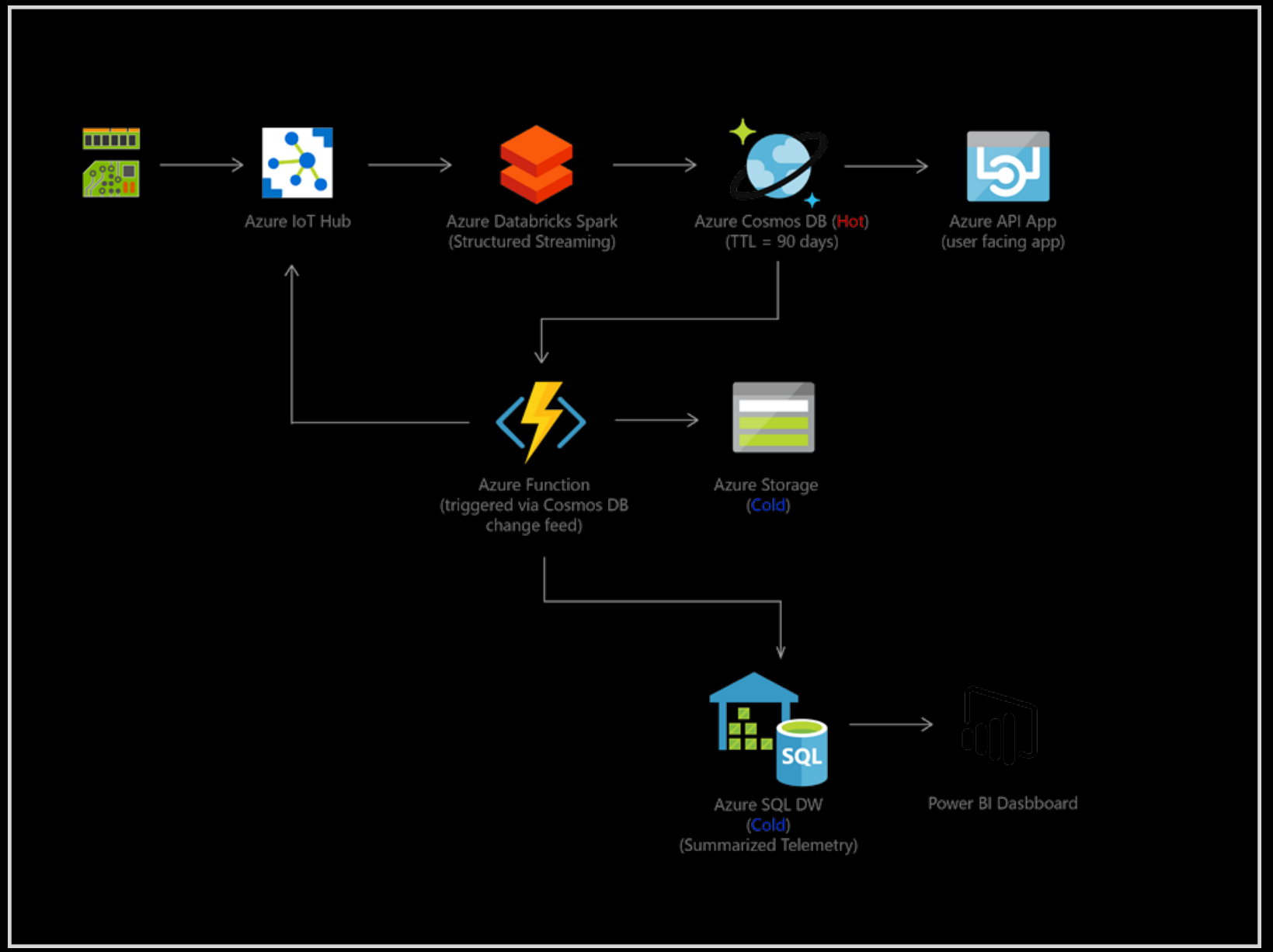
@chadgreen

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

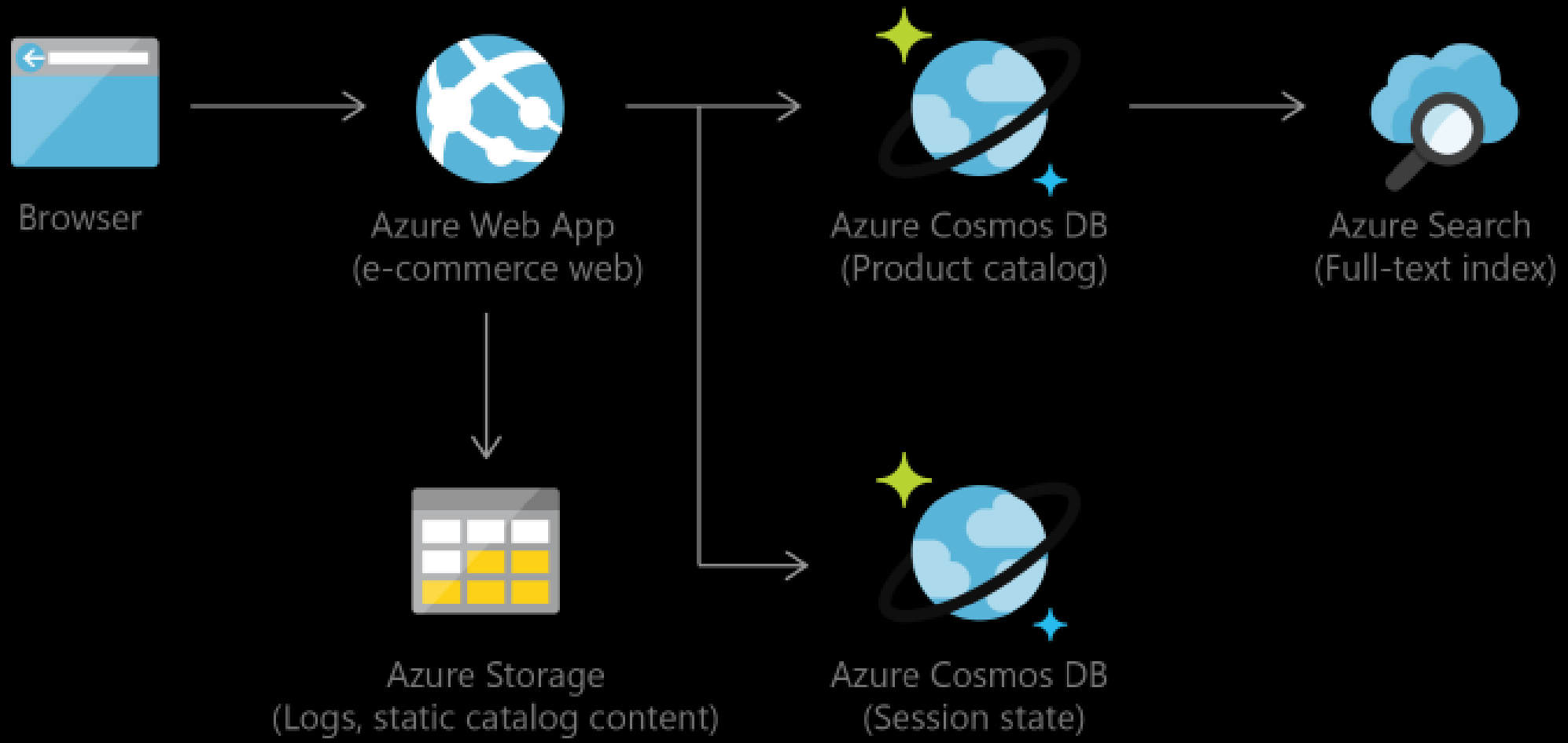
IoT and Telematics



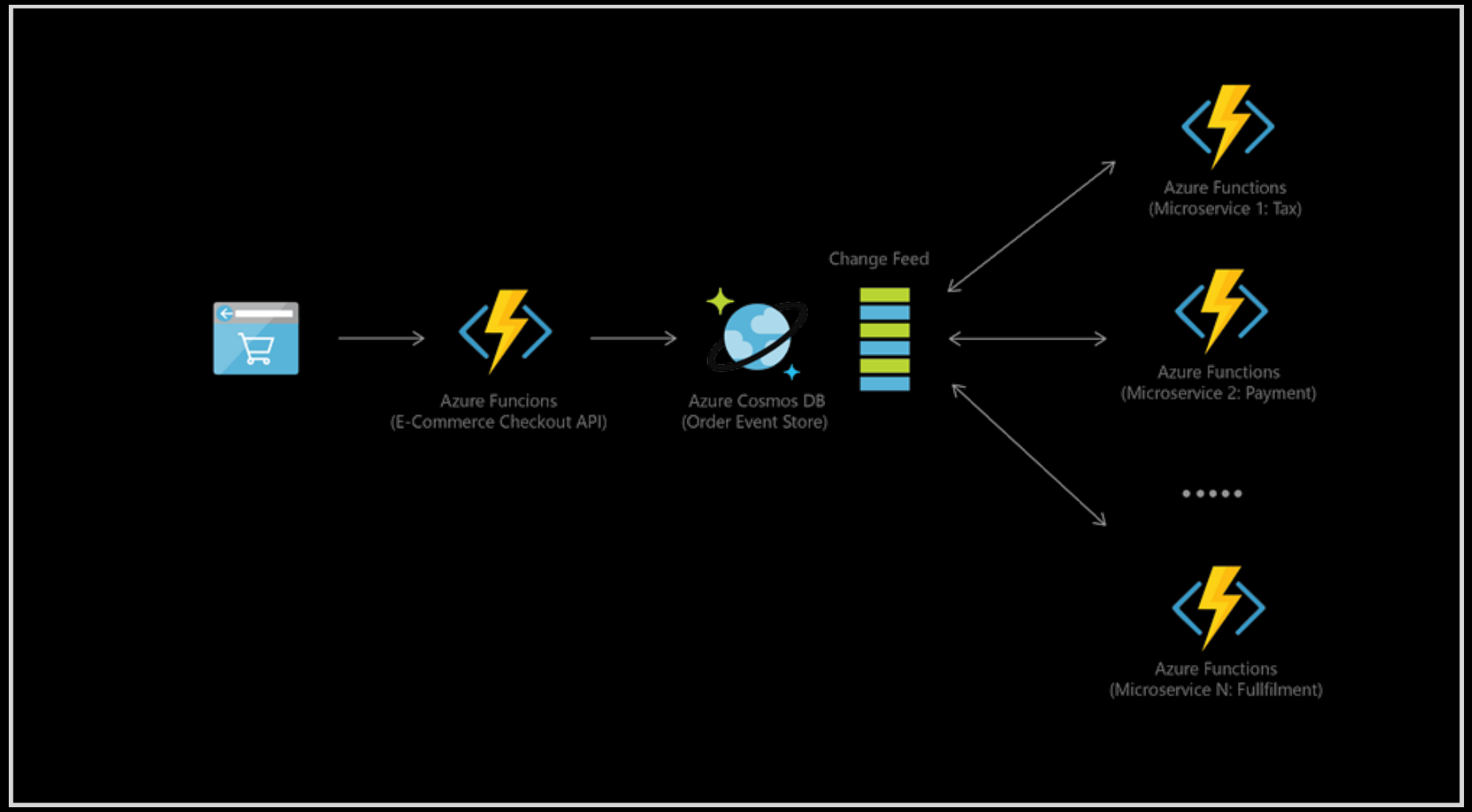
Retail and Marketing

- 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



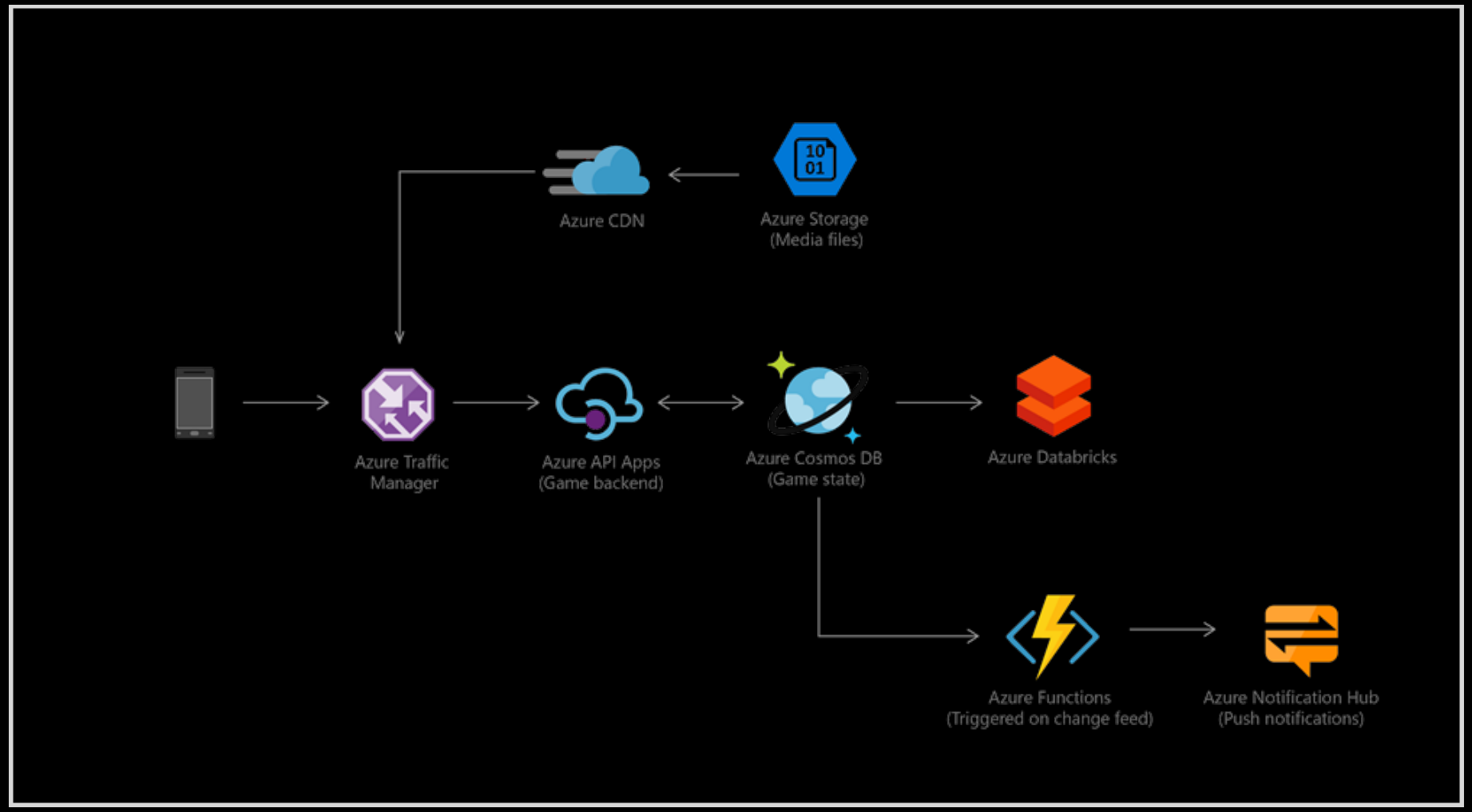
Retail and Marketing



Gaming

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

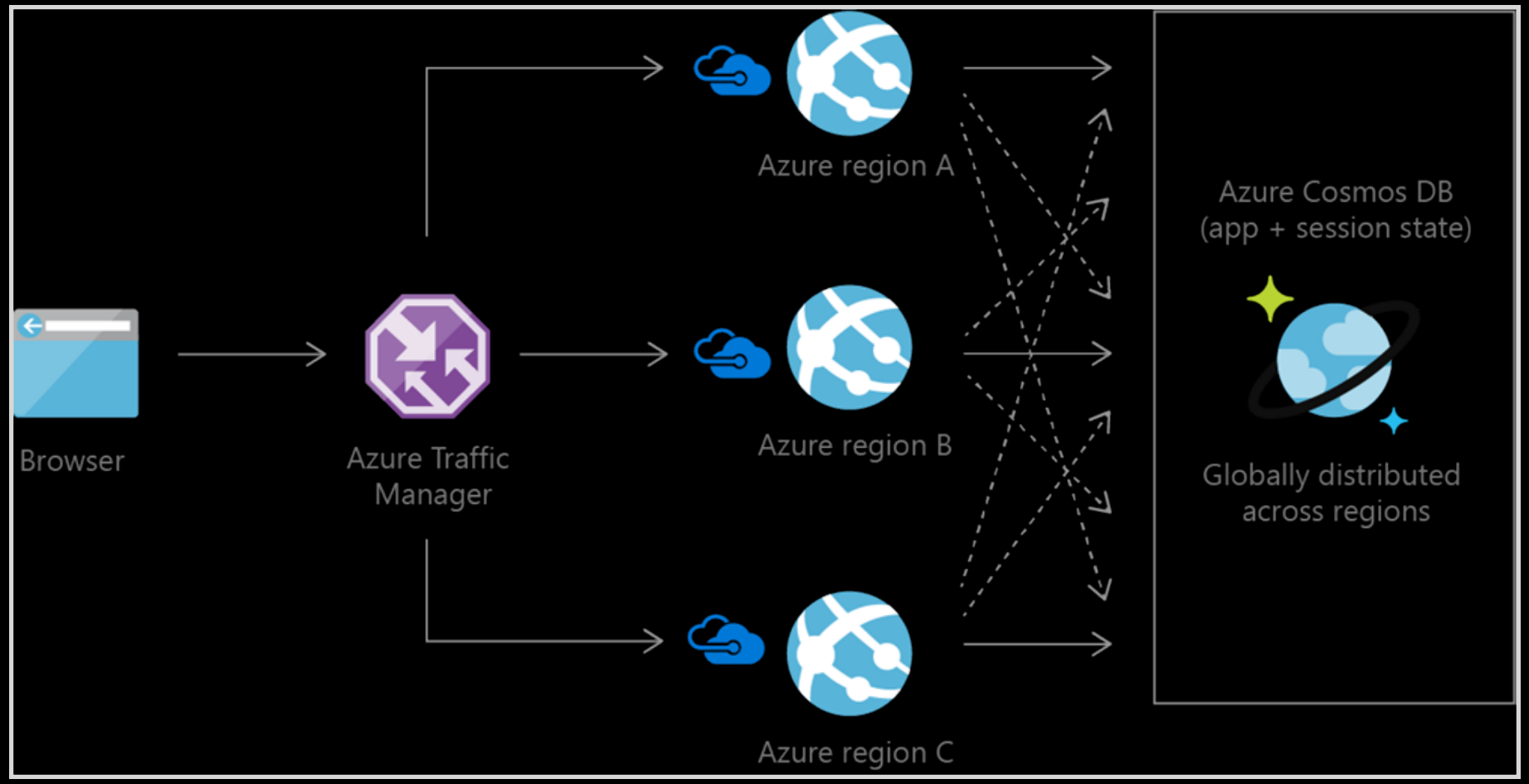
Gaming


Azure CDN
Azure Storage
(Media files)
Azure Traffic
Manager
Azure API Apps
(Game backend)
Azure Cosmos DB
(Game state)
Azure Databricks
Azure Functions
(Triggered on change feed)
Azure Notification Hub
(Push notifications)

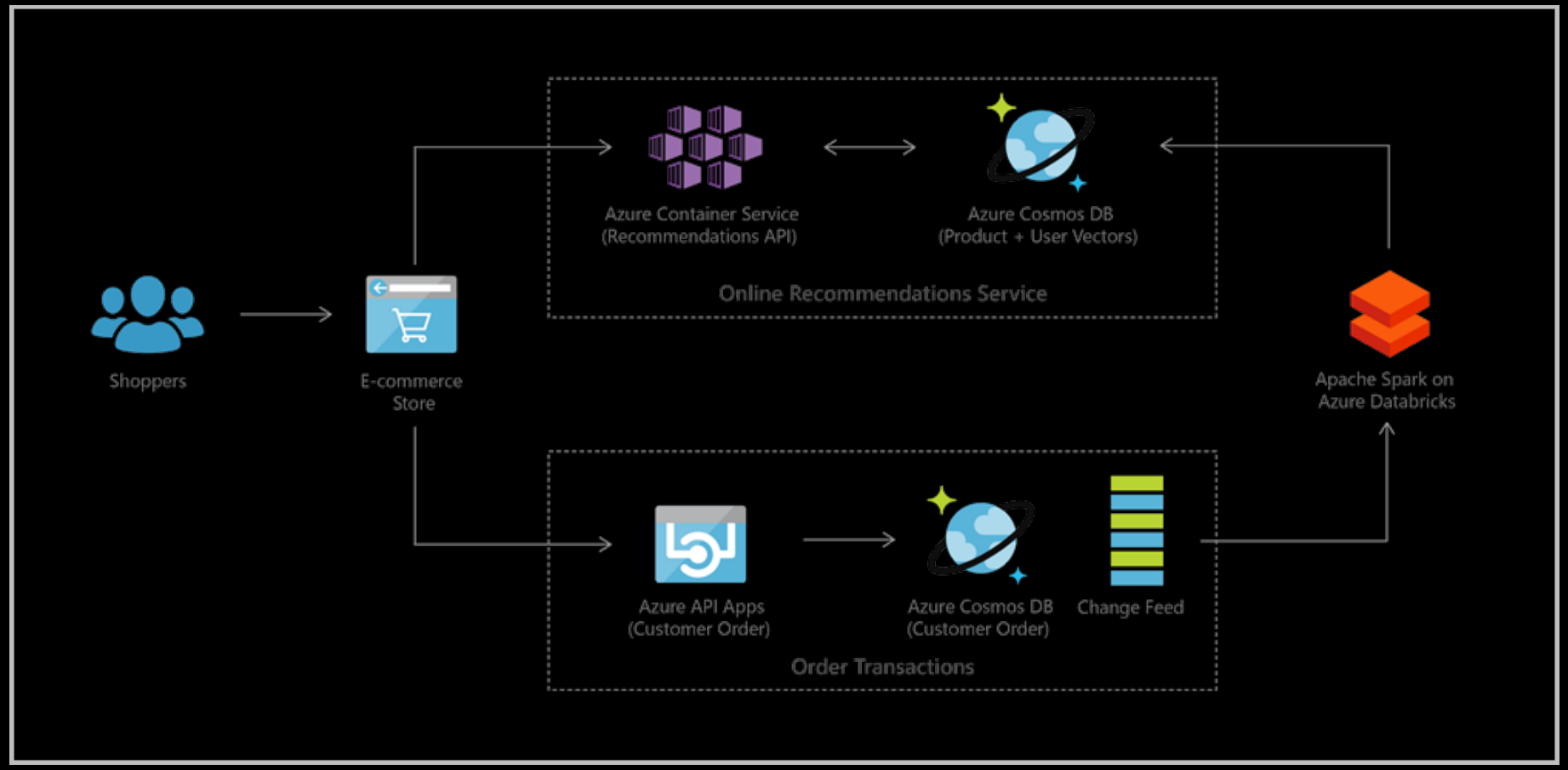
Web & Mobile Applications

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

Web & Mobile Applications – Social Applications



Web & Mobile Applications – Personalization



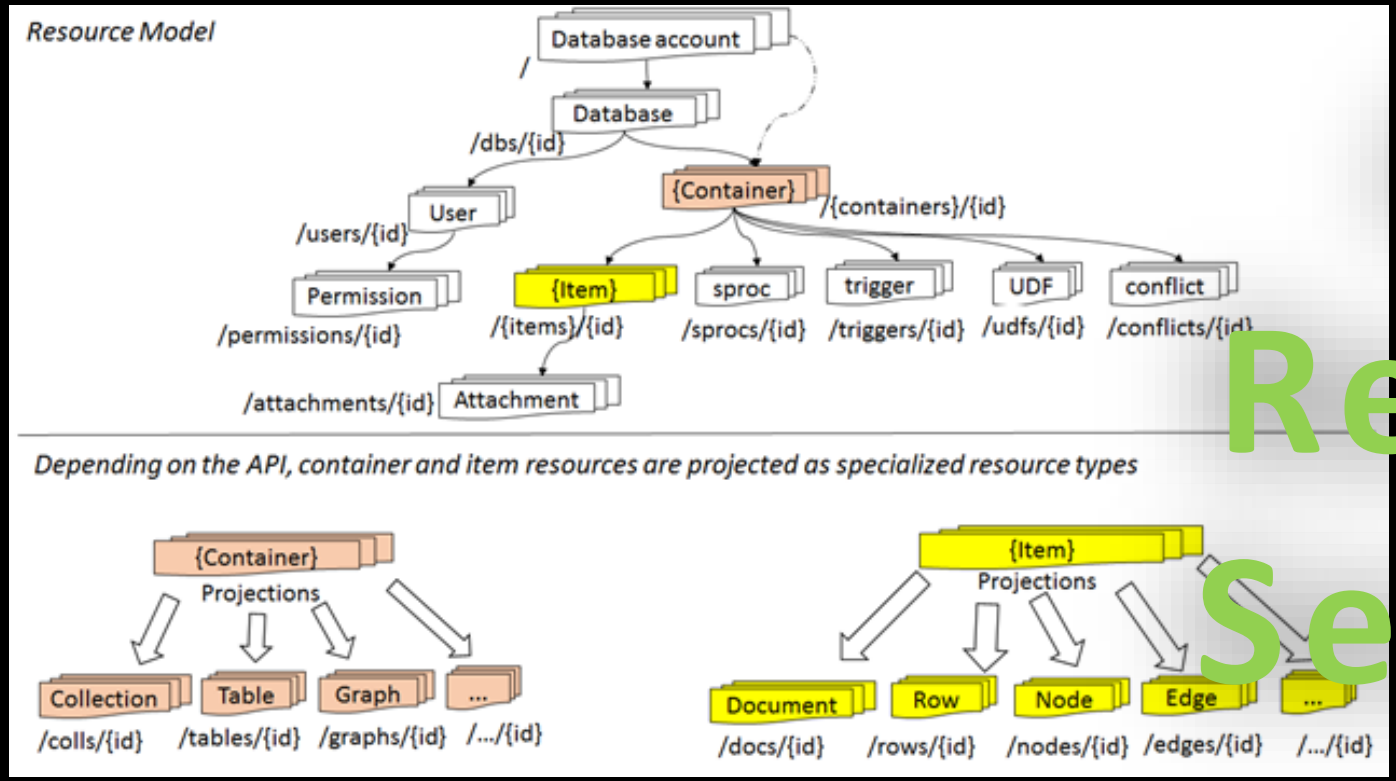
Integrations

@chadgreen

Cosmos DB Integrations

Navigating the 5 API Models

Resource Model and API Projections



Atom

Resource

Sequence

SQL API

Document Database

@chadgreen

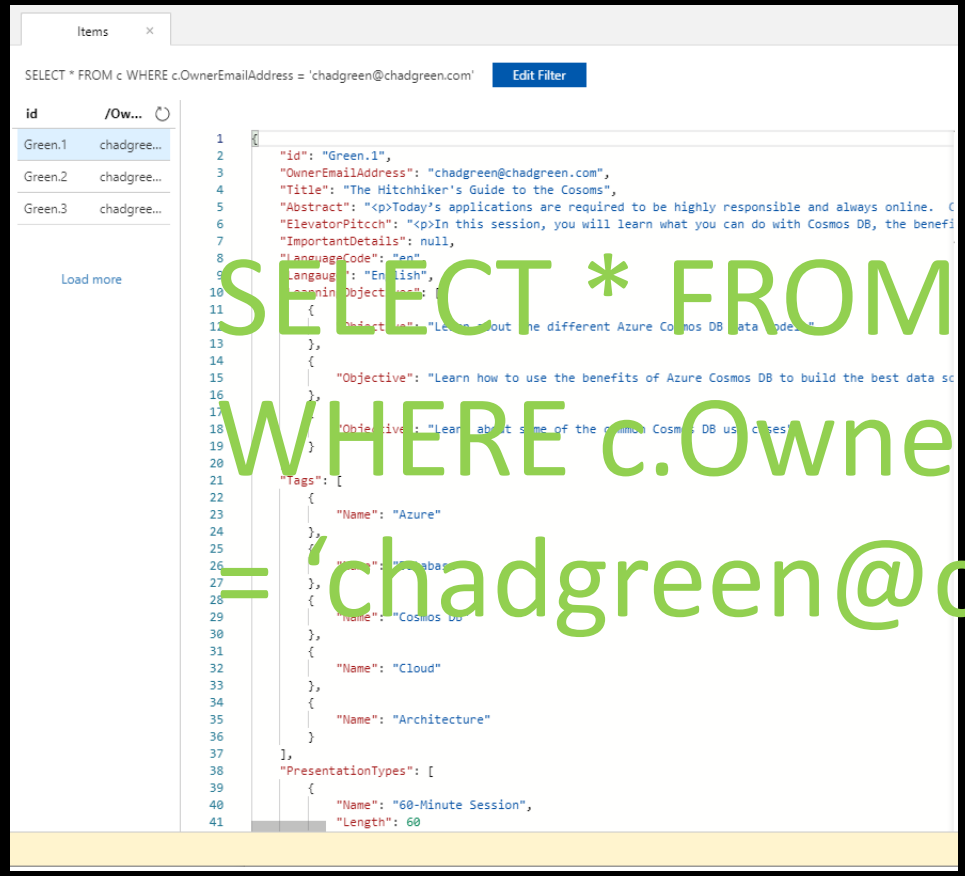
SQL API – What

- 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



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 ElevatorPitcch { 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 Langauge { 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

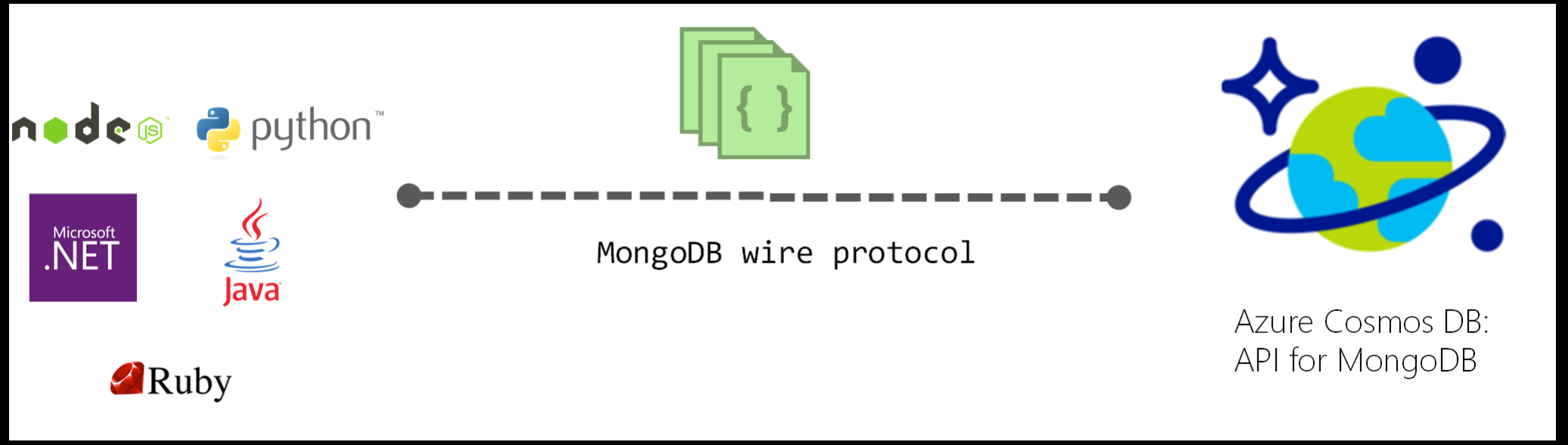
MongoDB

Document Database

@chadgreen

API for MongoDB – What

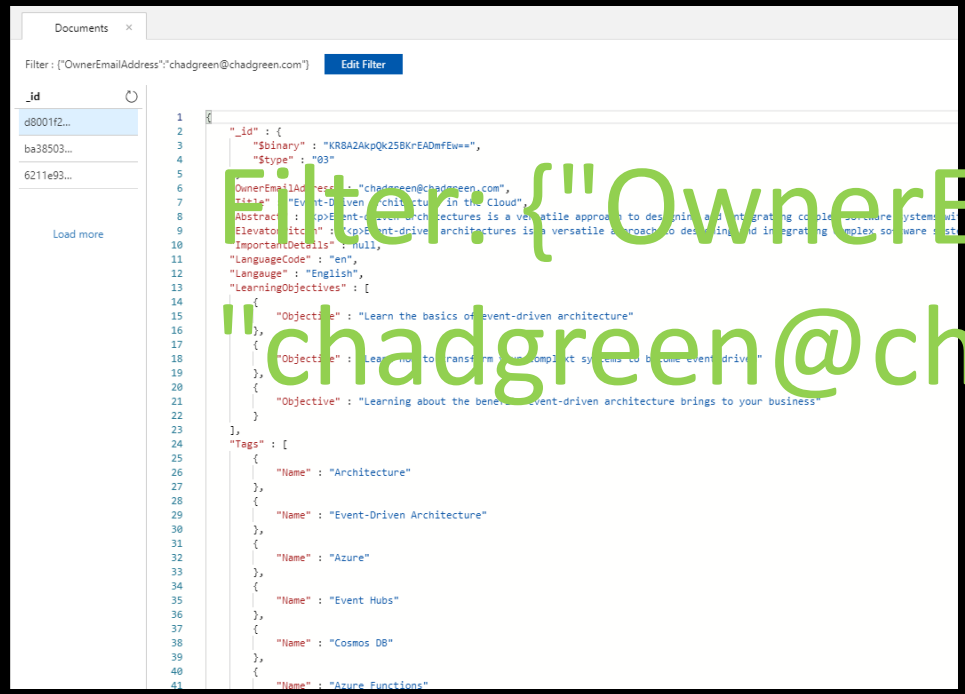
- Native MongoDB implementation
- Allows existing client SDKs, drivers, and tools to interact transparently
- Default is Mongo v3.2; v3.4 in preview



API for MongoDB – Why

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

API for MongoDB – How: Data Model



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

API for MongoDB – How: Insert

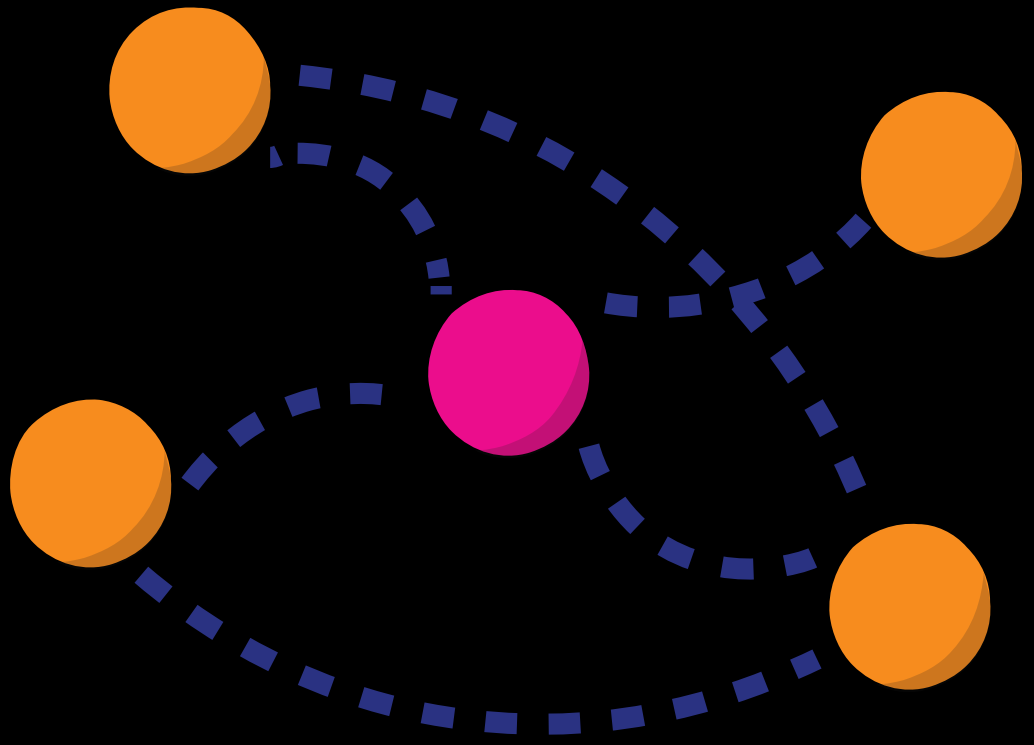
```
16 references | Chad Green, 1 day ago | 1 author, 1 change
public  class Presentation
{

    [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("ElevatorPitcch")]
    3 references | Chad Green, 1 day ago | 1 author, 1 change
    public string ElevatorPitcch { 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

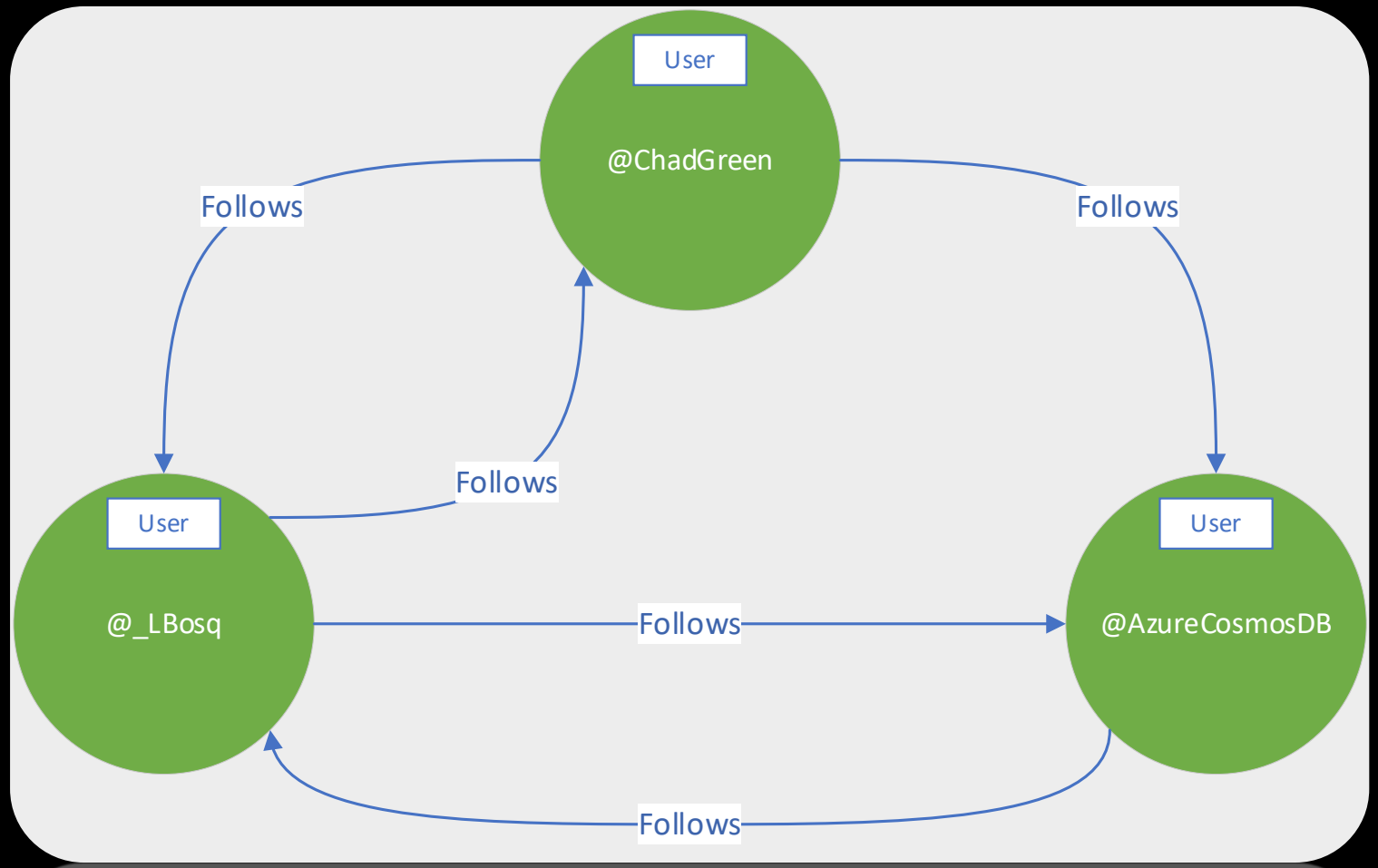
Graph Database

@chadgreen

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



Gremlin API – Why

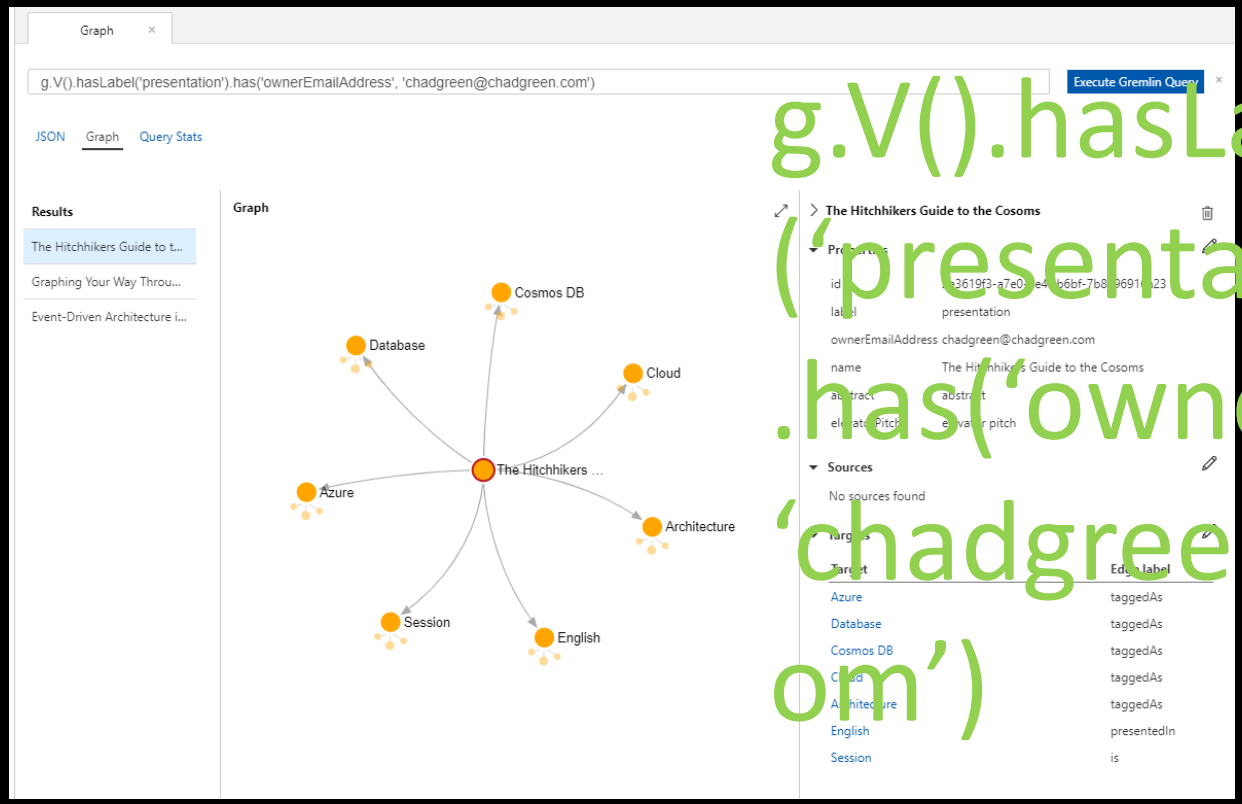
Building a graph database to model and traverse relationships among entities

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.c
om')
```

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
```

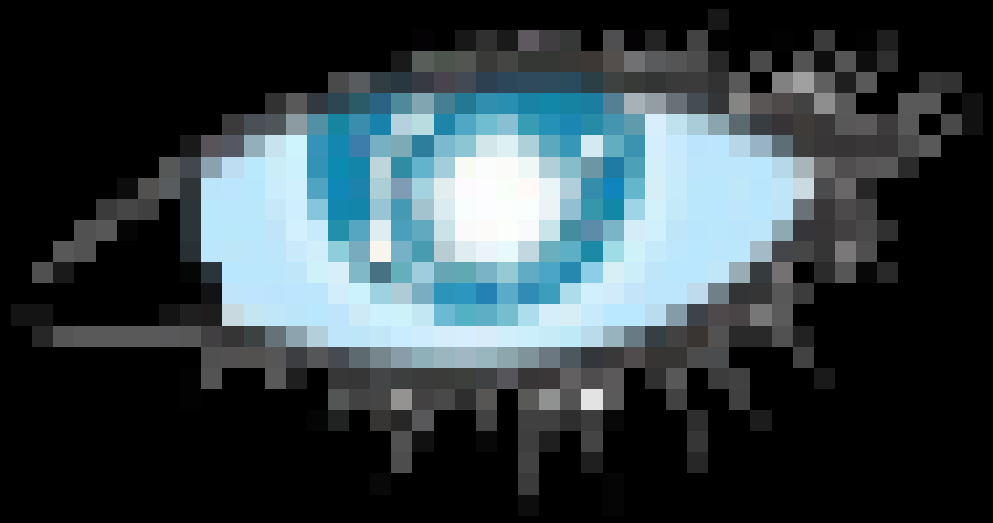
```
1 reference | Chad Green, 1 day ago | 1 author, 1 change
private static Task<ResultSet<dynamic>> SubmitRequest(GremlinClient gremlinClient, KeyValuePair<string, string> query)
{
  try
  {
    return gremlinClient.SubmitAsync<dynamic>(query.Value);
  }
  catch (ResponseException e)
  {
    Console.WriteLine("\tRequest Error!");

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

    // 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($"\t[\"x-ms-retry-after-ms\"] : { GetValueAsString(e.StatusAttributes, "x-ms-retry-after-ms")}");
    Console.WriteLine($"\t[\"x-ms-activity-id\"] : { GetValueAsString(e.StatusAttributes, "x-ms-activity-id")}");

    throw;
  }
}
```

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

@chadgreen

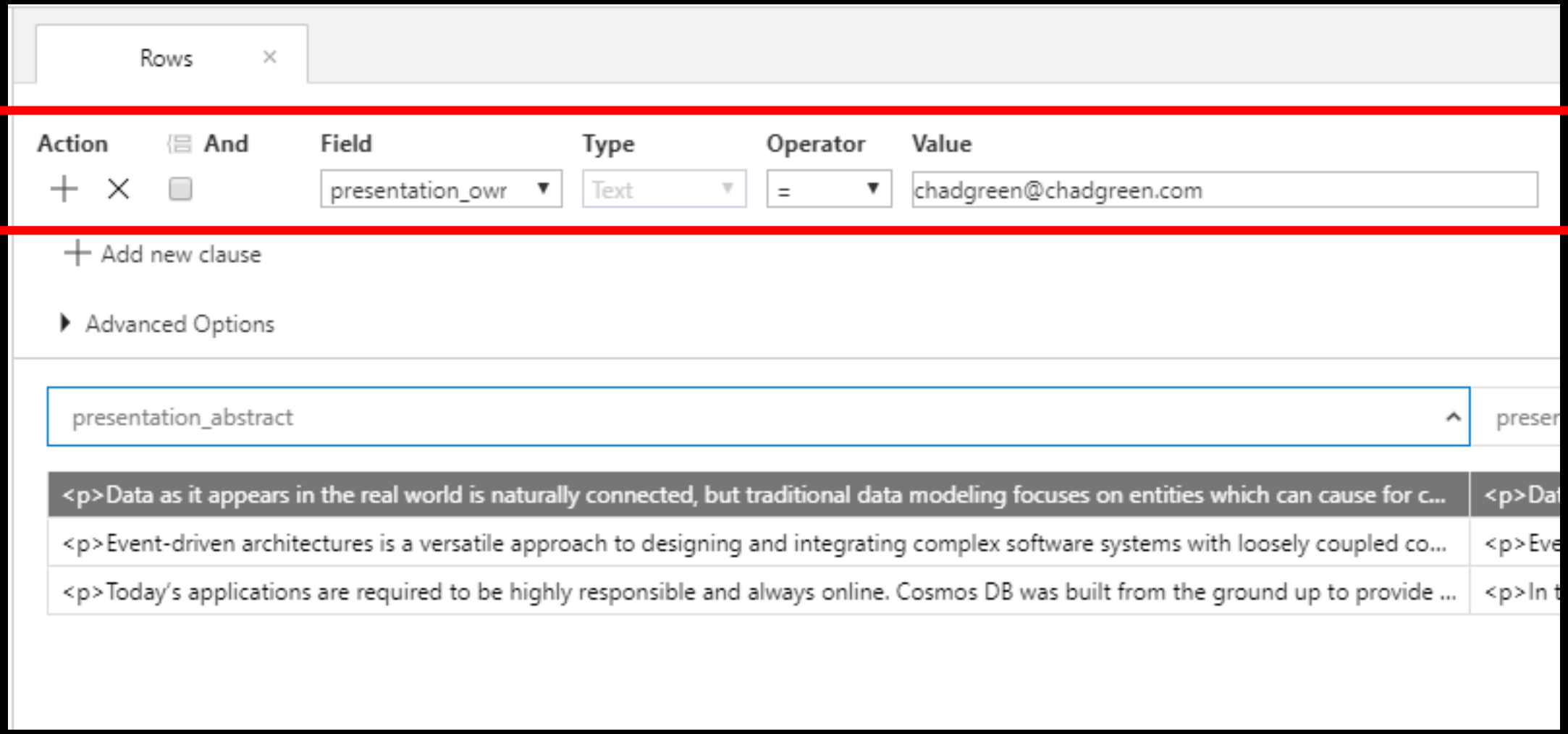
Cassandra API – What

- 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

Casandra API – Why

Migrating data from Cassandra to Azure Cosmos DB

Cassandra API – How: Data Model



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, presentatior
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.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)

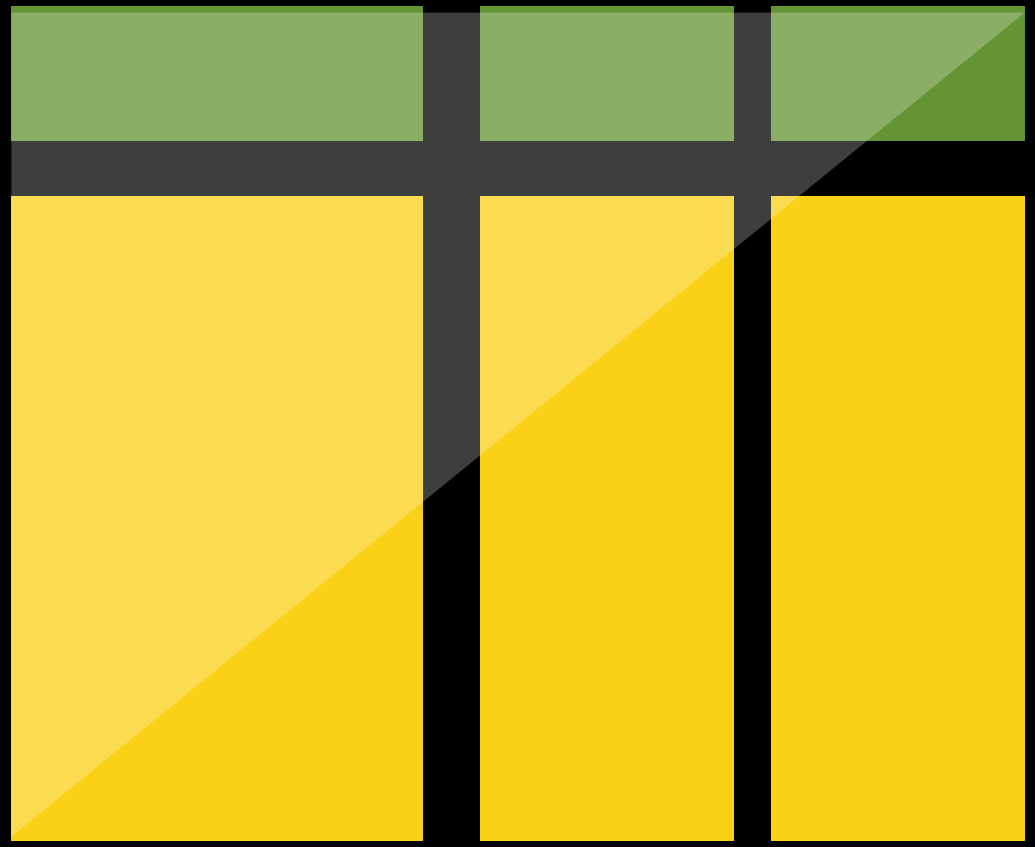


Table API

Table Storage

@chadgreen

Table API – What

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

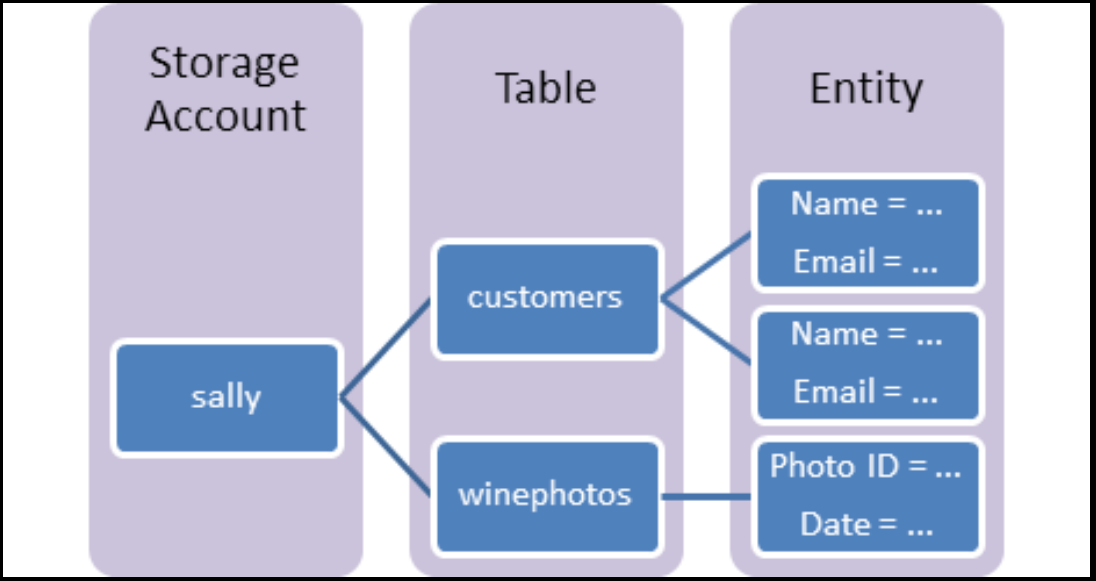


Table API – Why

Migrating data from Azure Table storage to Cosmos DB

Table API – How: Data Model

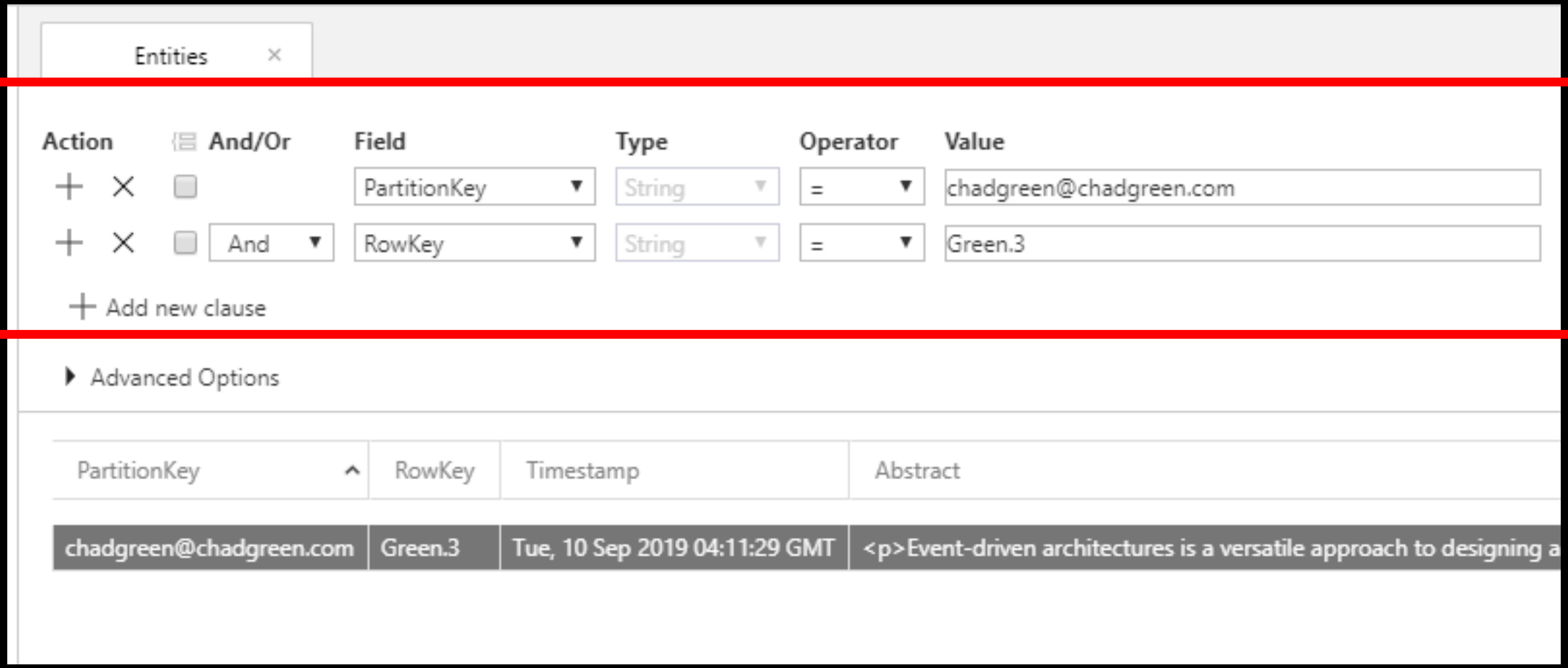


Table API – How: Insert

```
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 ElevatorPitcch { 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

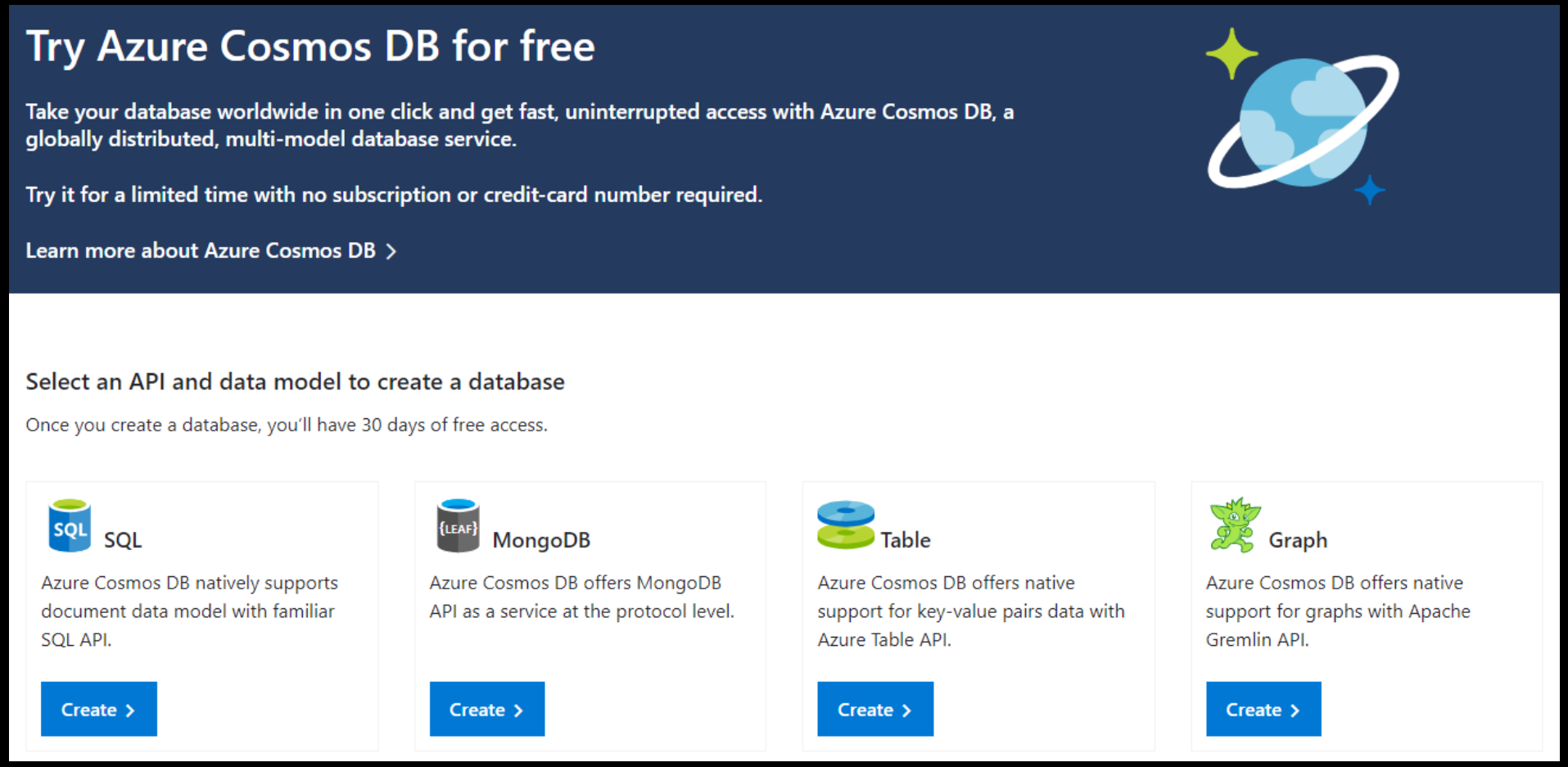
```
3 references | Chad Green, 2 days ago | 1 author, 1 change
public static async Task<Presentation> InsertOrMergeEntityAsync(CloudTable table, Presentation entity)
{
  if (entity == null)
  {
    throw new ArgumentNullException("entity");
  }

  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/



@chadgreen

Thank You

chadgreen@chadgreen.com
chadgreen.com
ChadGreen
ChadwickEGreen