Workshop 2 – Using Couchbase

Agenda

- What is NoSQL?
- What is a document database?
- Why NoSQL?
- Using the .NET SDK to interact with Couchbase

Couchbase
DEVELOPER COMMUNITY

Open source FTW

- https://github.com/couchbaselabs/aspnet-nosql-workshop

- If you find a typo, mistake, or spot an improvement, please send a pull request!

NoSQL: Document Database

- NoSQL is an umbrella term
- We'll be looking at a subset called "document databases"
- It's like a key/value database:
  - The key is some unique identifier
  - The value is in a known format (typically JSON)

Document

Key: Foo::123::456

Value:

```
{
    "name" : "Matt",
    "twitter" : "@mgroves",
    "favoriteMovies" : [
        "Star Wars",
        "Willy Wonka",
        "Glitter"
    ],
    "type" : "user"
}
```

Why ~~NoSQL~~ document databases?

- Architecture
- Performance
- Scaling
- Flexibility

Document database use cases

- Big data
- Profile management
- Content management
- Customer 360 view
- IoT
- Fraud detection
- Catalogs
- Personalization
- Digital communication
- Caching
- Mobile (with Couchbase Mobile)

https://www.couchbase.com/use-cases

NoSQL: Saving data

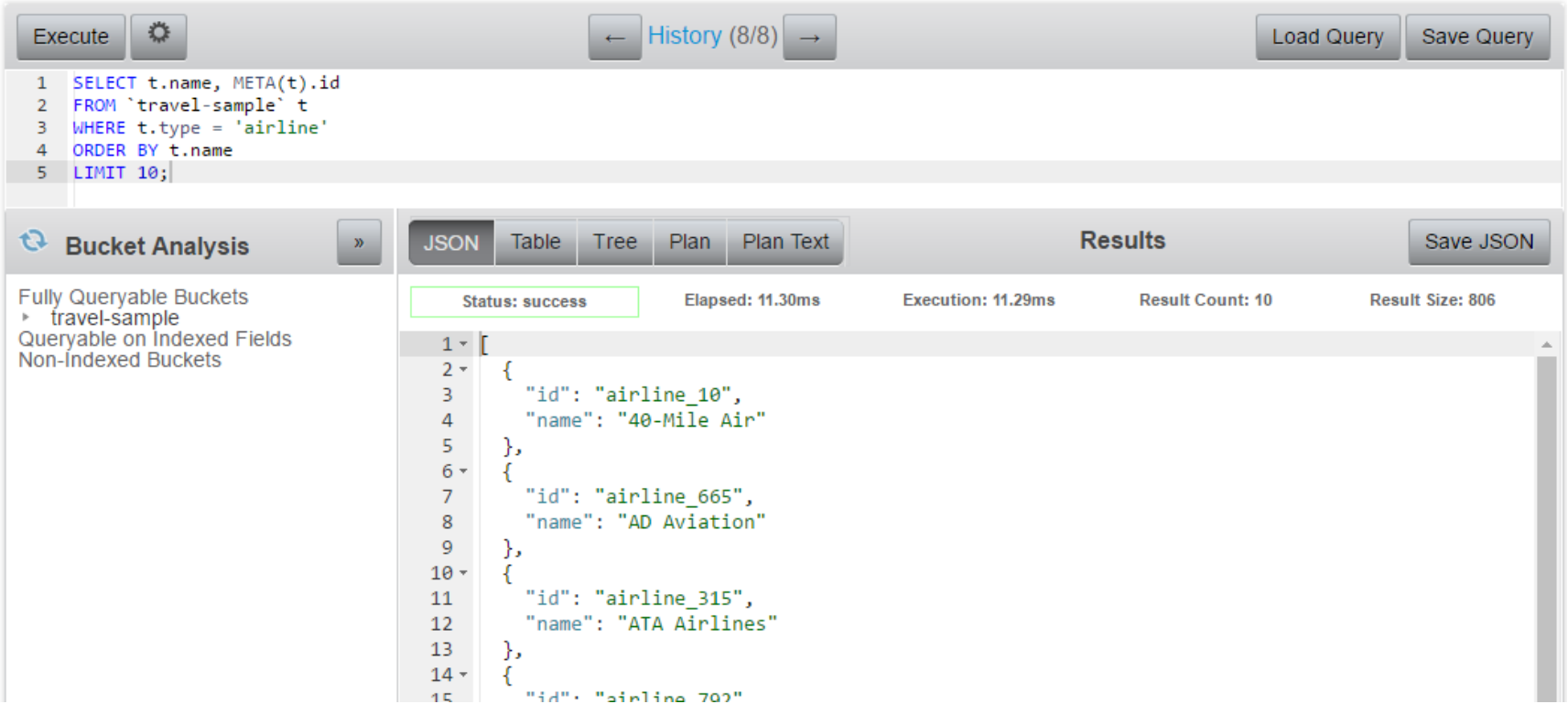
- NoSQL operations
  - Insert
  - Update
  - Upsert
- SQL (N1QL) options
  - INSERT
  - UPDATE
  - DELETE
  - MERGE
  - etc

NoSQL: Retrieving Data

- NoSQL operations
  - Get (by key)
- Views/indexes
  - Map/Reduce
- SQL (N1QL) options
  - SELECT

Query Workbench

http://localhost:8091



Exercise

How to install .NET SDK

NuGet: Install-Package CouchbaseNetClient

Connect to Couchbase

```
var config = new ClientConfiguration();
config.Servers = new List<Uri> {
    new Uri("couchbase://localhost")
};
ClusterHelper.Initialize(config);
```

```
IBucket _bucket = ClusterHelper.GetBucket("bucketname");
```

Insert a document

```
IDocument<dynamic> doc = new Document<dynamic>
{
    Id = Guid.NewGuid().ToString(),
    Content = new
    {
        firstName = "Connie",
        lastName = "James",
        city = "Columbus, Ohio",
        country = "USA",
        type = "person"
    }
};
_bucket.Insert(doc);
```

Insert, Replace, Upsert

```
IDocumentResult<dynamic> result1 = _bucket.Insert(document);
IDocumentResult<dynamic> result2 = _bucket.Upsert(document);
IDocumentResult<dynamic> result3 = _bucket.Replace(document);

Task<IDocumentResult<dynamic>> result4 = _bucket.InsertAsync(document);
Task<IDocumentResult<dynamic>> result5 = _bucket.UpsertAsync(document);
Task<IDocumentResult<dynamic>> result6 = _bucket.ReplaceAsync(document);

Task<IDocumentResult<dynamic>[]> result7 = _bucket.InsertAsync(
    new List<IDocument<dynamic>> { document, document2 });
Task<IDocumentResult<dynamic>[]> result8 = _bucket.UpsertAsync(
    new List<IDocument<dynamic>> { document, document2 });
Task<IDocumentResult<dynamic>[]> result9 = _bucket.ReplaceAsync(
    new List<IDocument<dynamic>> { document, document2 });
```

Getting document(s) by key(s)

```
IOperationResult<dynamic> aDocument = _bucket.Get<dynamic>("key");

dynamic result = await _bucket.GetAsync<dynamic>(id);
```

```
IDictionary<string, IOperationResult<dynamic>> multipleDocuments = _bucket.Get<dynamic>(
    new List<string> {"key1", "key2", "key3"}
);
```

Results

```
public interface IResult
{
    bool Success { get; }
    string Message { get; }
    Exception Exception { get; }
}
```

Insert, Replace, Upsert

```
public interface IDocumentResult : IResult
{
    ResponseStatus Status { get; }
    string Id { get; }
}
```

```
public interface IDocumentResult<T> : IDocumentResult
{
    Document<T> Document { get; }
    T Content { get; }
}
```

Get

```
public interface IOperationResult : IResult
{
    ulong Cas { get; }
    ResponseStatus Status { get; }
    string Id { get; }
}
```

```
public interface IOperationResult<out T> : IOperationResult
{
    T Value { get; }
}
```

Questions

Workshop 1: N1QL

- Create a primary index on default bucket

```
CREATE PRIMARY INDEX on `default`
```

- Add document(s) to default bucket
  - Use "Create Document" button in Documents view
- SELECT documents

```
SELECT d.* FROM `bucketname` d
```

Workshop 2: Using SDK in "Hello World"

- Write a console program that:
  - Insert document(s)
  - Select documents with N1QL

- Start with dotnet_workshop / dotnetcore_workshop folders
  - Fill in the blanks: look for “TODO”