Untitled

December 15, 2023

1 Optimising with Raven

We need a development environment with the RavenDB.Client library and access to a running RavenDB server. Below are general steps to set up and run the code in a C# environment:

1. We can install the RavenDB.Client NuGet package using a package manager like NuGet Package Manager Console or Visual Studio Package Manager.

Install-Package RavenDB.Client!

2. Set Up RavenDB Server: Download and install the RavenDB server from the official RavenDB website. Follow the installation instructions for your operating system.
3. Create a RavenDB Database: After installing RavenDB, we need to create a database. We can do this using the RavenDB Studio, which is accessible through a web browser.
4. Configure Connection to RavenDB: In our C# code, configure the connection to the RavenDB server. Update the URL to match our RavenDB server instance and the name of the database we created.

[ ]:
```
var store = new DocumentStore
{
    Urls = new[] { "http://localhost:8080" }, // Update with your RavenDB␣
 ↪server URL
    Database = "YourDatabaseName" // Update with your RavenDB database name
};
store.Initialize();
```

[ ]:
```
Run the Code:
Make sure to adjust any placeholders (like database names or URLs) to match␣
 ↪your specific configuration.

The complete example using some of the provided code snippets:


using System;
using Raven.Client.Documents;
using Raven.Client.ServerWide;
using Raven.Client.ServerWide.Operations;
using RavenDB.Client;
```

```
class Program
{
    static void Main()
    {
        using (var store = new DocumentStore
        {
            Urls = new[] { "http://localhost:8080" }, // Update with your␣
 ↪RavenDB server URL
            Database = "YourDatabaseName" // Update with your RavenDB database␣
 ↪name
        })
        {
            store.Initialize();

            // Example of creating an index
            new Prediction_Index().Execute(store);

            // Example of querying with projection
            using (var session = store.OpenSession())
            {
                var predictions = session.Query<Prediction>()
                    .Where(p => p.ModelID == 1)
                    .SelectFields<PredictionProjection>()
                    .ToList();

                foreach (var prediction in predictions)
                {
                    Console.WriteLine($"Prediction ID: {prediction.
 ↪PredictionID}, Value: {prediction.PredictionValue}");
                }
            }

            // Additional code and queries can be added here
        }
    }
}

public class Prediction_Index : AbstractIndexCreationTask<Prediction>
{
    public Prediction_Index()
    {
        Map = predictions => from prediction in predictions
                             select new
                             {
                                 prediction.ModelID,
                                 prediction.FeatureID,
```

```
                                prediction.PredictionValue,
                                prediction.PredictionDate
                            };
    }
}

public class PredictionProjection
{
    public string PredictionID { get; set; }
    public string PredictionValue { get; set; }
}


Ensure that we replace placeholders such as "YourDatabaseName" and the server␣
 ↪URL with our actual RavenDB database name and server URL.
```

Optimizing raven queries:

1. RavenDB uses indexes for querying. Ensure that you have appropriate indexes for the fields used in your queries.

// Example index creation in RavenDB from prediction in docs.Predictions select new { prediction.ModelID, prediction.FeatureID, prediction.PredictionValue };

2. Querying with Linq: RavenDB uses LINQ for querying. Optimize your LINQ queries based on the data structure and indexing.

// Example LINQ query in RavenDB var results = session.Query() .Where(p => p.ModelID == 1 && p.FeatureID == 1) .ToList();

3. Use Projections: Only fetch the fields you need by using projections to reduce the amount of data transferred.

// Example projection in RavenDB var results = session.Query() .Where(p => p.ModelID == 1 && p.FeatureID == 1) .Select(p => new { p.PredictionID, p.PredictionValue }) .ToList();

4. Avoid SELECT N+1 Problem: Be mindful of the SELECT N+1 problem. Use the Include method to fetch related documents in a single query.

// Example of using Include in RavenDB var results = session.Query() .Include(p => p.ModelID) .Where(p => p.ModelID == 1) .ToList();

5. Query Execution Statistics: Use the RavenDB Management Studio to analyze query execution statistics and identify areas for improvement.
6. Denormalization: Consider denormalizing data where it makes sense to avoid complex joins and improve query performance.
7. Optimize Indexing Strategies: Review and optimize indexing strategies based on the types of queries your application performs.

[ ]: