Programming against NoSql Databases

John F. Gnazzo

Introduction

A NoSQL database defines a type of database that is non-relational.

Advantages of NoSQL over Relational Databases include:

- Low latency
- High performance
- Highly scalable

NoSQL databases such as MongoDB, can be easily accessed through programming languages such as C#.

An Example of using C# to perform the four (4) basic database CRUD (Create, Read, Update, Delete) Operations, is illustrated.

Discussion

NoSQL databases provide a mechanism for storage and retrieval of data which is modeled in means other than the tabular relations used in relational databases. Such databases have existed since the late 1960s, but did not obtain the "NoSQL" moniker until a surge of popularity in the early twenty-first century, triggered by the needs of Web 2.0 companies such as Facebook, Google and Amazon.com.

Motivations for this approach include: simplicity of design, simpler "horizontal" scaling to clusters of machines, which is a problem for relational databases, and finer control over availability. The data structures used by NoSQL databases (e.g. key-value, wide column, graph, or document) are different from those used by default in relational databases, making some operations faster in NoSQL. The particular suitability of a given NoSQL database depends on the problem it must solve. Sometimes the data structures used by NoSQL databases are also viewed as "more flexible" than relational database tables.

NoSQL databases are increasingly used in big data and real-time web applications. NoSQL systems are also sometimes called "Not only SQL" to emphasize that they may support SQL-like query languages.[1]

MongoDb is a popular commercial NoSql database. MongoDB stores data using a flexible document data model that is similar to JSON. Documents contain one or more fields, including arrays, binary data and sub-documents. Fields can vary from document to document. This flexibility allows development teams to evolve the data model rapidly as their application requirements change. When you need to lock down your data model, optional document validation enforces the rules you choose.

[1] Wikipedia, https://en.wikipedia.org/wiki/NoSQL

Developers access documents through rich, idiomatic drivers available in all popular programming languages. Documents map naturally to the objects in modern languages, which allows developers to be extremely productive. Typically, there's no need for an ORM layer.[2]

Given a Mongo Db instance of **test**, the following class demonstrates the basic CRUD operations.

```
using System.Collections.Generic;
using System.Threading.Tasks;
using MongoDB.Bson;
using MongoDB.Driver;

namespace MongoDbProjectForBlog
{
  public class MongoClass
  {
    protected static IMongoClient Client;
    protected static IMongoDatabase Database;

    // Constructor
    public MongoClass()
    {
      Client = new MongoClient();
      Database = Client.GetDatabase("test");
    }

    // Add data to document  (Create)
    public async void Add(string document, BsonDocument jsonData)
    {
      var collection = Database.GetCollection<BsonDocument>(document);
      await collection.InsertOneAsync(jsonData);
    }

    // Select Data in document (Read)
    public async Task<List<BsonDocument>> Select(string document, string field, string data)
    {
      var collection = Database.GetCollection<BsonDocument>(document);
      var filter = Builders<BsonDocument>.Filter.Eq(field, data);
      var result = await collection.Find(filter).ToListAsync();
      return result;
    }

    // Update data to document (Update)
    public async Task<UpdateResult> Update(string document, BsonDocument jsonData)
    {
      var collection = Database.GetCollection<BsonDocument>(document);
      var filter = Builders<BsonDocument>.Filter.Eq("name", "Juni");
      var update = Builders<BsonDocument>.Update
        .Set("cuisine", "American (New)")
        .CurrentDate("lastModified");
      var result = await collection.UpdateOneAsync(filter, update);
      return result;
    }

    // Delete data from document (Delete)
    public async Task<DeleteResult> Delete(string document, string field, string data)
```

[2] MongoDb: https://www.mongodb.com/what-is-mongodb

```
    {
        var collection = Database.GetCollection<BsonDocument>(document);
        var filter = Builders<BsonDocument>.Filter.Eq(field, data);
        var result = await collection.DeleteManyAsync(filter);
        return result;
    }
  }
}
```

Conclusion

NoSql databases such as MongoDb can provide value to businesses as an alternative to a relational database by providing greater performance, lower latency, higher scalability, and a simpler approach to storing large volumes of data. NoSql Databases can also be easily accessed using the C# programming language.