2N-5

JavaScript処理系におけるLINQを用いたNoSQLストレージに対する問合せ処理

中挾晃介† 天笠俊之‡ 北川博之‡

†筑波大学情報学群情報科学類 ‡筑波大学システム情報系情報工学域

1 はじめに

近年，インターネット上には様々なWebアプリケーションが溢れており，これらのWebアプリケーションが扱うデータは，急激に大規模化している．[1] この膨大なデータを扱うにあたり，NoSQLを選択するWebアプリケーションが増えてきている．これは，一般にNoSQLストレージがRDBMSに比べてスケールアウトにより比較的性能を上げやすいという性質を持つためである．このNoSQLストレージには，Cassandra[2]，HBase[3]，CouchDB[4]，MongoDB[5] などがある．

一方で，昨今では，サーバサイドでもJavaScriptによるWebアプリケーション開発が増えている．このサーバサイドJavaScriptの一つにNode.js[6] がある．このようなサーバサイドJavaScriptを用いて，NoSQLストレージにアクセスする際，そのアクセスには透過的で，効率的な処理が望まれている．

Webアプリケーション開発においては，データベースに加えて，XMLやリスト等のメモリ上のデータ構造に対するアクセスを行う必要がある．そこで，様々なタイプのデータソースに対する操作を，SQLライクに共通の構文で操作できる，LINQ[7] という技術が開発された．このLINQの一つに，LINQ to Objectsというクラスがあり，これを用いると，配列のようなIEnumerableインターフェースを実装する全てのコレクションクラスに対してクエリ処理をすることができる．このクラスをJavaScriptのライブラリに移植したものに，JSINQ[8] がある．ただし，現状ではJavaScriptからNoSQLへの問合せをサポートするLINQのクラスは存在しない．

そこで，本研究では，サーバサイドJavaScriptから，JSINQを通して，NoSQLストレージへの問合せを可能にする．具体的には，サーバサイドJavaScriptとしてNode.jsを，NoSQLストレージとしてMongoDBを用いる．本システムでは，JSINQに対して拡張を行うことにより，MongoDB上のデータに対してシームレスなアクセスを提供する．これにより，JavaScript上においてMongoDBを利用したアプリケーションのコーディングと，大規模データの処理が容易になることが期待される．

本稿では，第2節で本研究に関連する既存の技術について，第3節で提案手法を説明し，最後に第4節でまとめをそれぞれ述べる．

```
from n in numbers
where n >= 2 && n <= 4
orderby n
select n;
```

図 1: LINQの問合せ例

2 LINQ

LINQは，配列やリレーショナルデータ，XMLなどに対する操作をC#やVBなどの.NET Framework対応言語に統合するものである．LINQの問合せ例を図1に示す．この例は，整数の配列numbersから，2以上4以下の数字を選択する例である．通常は，foreach文などのイテレータによってデータを操作する必要があるが，SQLライクなクエリによって，簡単にデータ操作が実現できる．

3 提案手法

3.1 提案手法の処理の流れ

提案するシステムにおいてJSINQを用いてMongoDBにアクセスし、データを処理する流れを説明する．

今，Customer（顧客）とOrder（注文）というコレクションがあり，顧客の年齢が30歳以上の人が注文した商品を取得するという簡単な問合せを考える．このクエリは以下のように書くことができる．

Query Processing using LINQ for NoSQL Storage on JavaScript Implementation

Kousuke NAKABASAMI†(nakabasami@kde.cs.tsukuba.ac.jp), Toshiyuki AMAGASA‡(amagasa@cs.tsukuba.ac.jp) and Hiroyuki KITAGAWA‡(kitagawa@cs.tsukuba.ac.jp)
†College of Information Sciences, University of Tsukuba
‡Faculty of Engineering, Information and Systems, University of Tsukuba

```
from    customer, order
where   customer.id == order.customerId
        && customer.age >= 30
select  order.item
```

このクエリの具体的な処理の流れは，以下のようになる．

1) クエリの文字列を，一語ずつ分解し配列に格納する．格納した配列を順に走査し，Where 句を見つけた場合，その Where 句の条件式の中で MongoDB のクエリ構文に変換できるものを配列から取り出し，別の配列に格納する．ここでは，顧客の年齢が 30 歳以上という条件が MongoDB で処理できるので，この条件式を取り出す．もう一つの条件は，Customer と Order の Join であるが，MongoDB は Join をサポートしていない．このような，MongoDB のクエリ構文に変換できない条件式は，取り出さない．

2) 1) で取り出した条件式を MongoDB のクエリ構文に変換する．以下に，変換したクエリを示す．

```
customer.find({ age : { $gte : 30 }});
```

3) 1) における，クエリの文字列が入っている配列から要素を順に一つずつ取り出し，文字列連結をして，MongoDB のクエリ構文に変換できる Where 条件を除いた，新しいクエリの文字列を作る．

4) 3) で作成した新しいクエリの文字列をパースし，配列を用いた構文木を作る．

5) 4) で作成した構文木をコンパイルし，JavaScript で実行可能な関数を作る．

6) 5) で作成した関数を実行する．実行する関数の中で，MongoDB にアクセスする別の関数を呼び出す．このとき，2) において，MongoDB のクエリ構文に変換した Where 句（顧客の年齢が 30 歳以上）と，アクセスする MongoDB の URL をこの関数の引数として渡す．

7) 6) において呼び出した，MongoDB にアクセスする関数の中で，Node.js 用の MongoDB のドライバである，node-mongodb-native を用いて MongoDB に接続し，データを受け取る．このとき同時に，MongoDB のクエリ構文に変換した Where 句（顧客の年齢が 30 歳以上）も適用する．

8) 最後に JSINQ 側で，7) で MongoDB から受け取ったデータを 3) で作成した新しいクエリに適用し，最終的に望んだ結果を受け取る．Join の処理はここで行う．

以上が，JSINQ を拡張して実装した提案手法の処理の流れである．

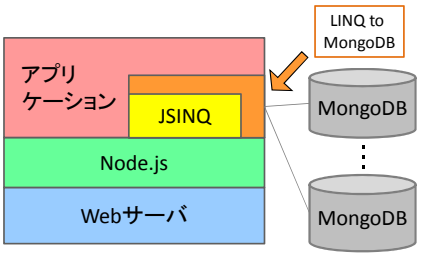

図 2: システムの全体像

3.2 システムの実装

実装したシステムの全体像を図 2 に示す．Node.js からの MongoDB に対する問合せ処理において，JSINQ を用いる．JSINQ は，jsinq.js，jsinq-query2.js，jsinq-mongo.js からなり，それぞれ，jsinq.js で配列などの IEnumerable を実装するオブジェクトの操作を行い，jsinq-query2.js でクエリの解析と実行を行い，jsinq-mongo.js で MongoDB のアクセスを行っている．

4 まとめ

本稿では，Node.js から，JSINQ を通して，MongoDB への問合せを可能にする一連の手法を説明した．これにより，LINQ による複数の MongoDB を横断した問合せが可能となる．

今後の課題として，実装した LINQ to MongoDB を，ベンチマークである TPC-H[10] を用いて，システムの構成に応じた性能の比較評価を行う予定である．

参考文献

[1] Sattam Alsubaiee, Alexander Behm, Raman Grover, Rares Vernica, Vinayak Borkar, Michael J. Carey, Chen Li, ASTERIX: Scalable Warehouse-Style Web Data Integration, II-Web'12 May 20 2012, Scottsdale, AZ, USA, Copyright 2012 ACM 978-1-4503-1239-4/12/05

[2] Cassandra, http://cassandra.apache.org/

[3] HBase, http://hbase.apache.org/

[4] CouchDB, http://couchdb.apache.org/

[5] MongoDB, http://www.mongodb.org/

[6] node.js, http://nodejs.org/

[7] C#での LINQ の使用, http://msdn.microsoft.com/ja-jp/library/bb655883(v=vs.90).aspx

[8] JSINQ - LINQ to Objects for JavaScript, http://jsinq.codeplex.com/

[9] linq.js - LINQ for JavaScript, http://linqjs.codeplex.com/

[10] TPC-H, http://www.tpc.org/tpch/