JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지 Vol. 23, No. 6: 645~651, Jun. 2019

기상 데이터를 활용한 CQRS 패턴의 조회 모델 구현

서보민[1] · 전철호[1] · 전현식[2] · 안세윤[3] · 박현주[4*]

Implementation of query model of CQRS pattern using weather data

Bomin Seo[1] · Cheolho Jeon[1] · Hyeonsig Jeon[2] · Seyun An[3] · Hyun-ju Park[4*]

[1]Graduate student, Department of Mobile Convergence Engineering, Hanbat National University, Daejeon, 34158 Korea
[2]Postdoc. Researcher, Department of Mobile Convergence Engineering, Hanbat National University, Daejeon, 34158 Korea
[3]Associate professor, Department of Industrial Design, Hanbat National University, Daejeon, 46958 Korea
[4*]Professor, Department of Information and Communication Engineering, Hanbat National University, Daejeon, 34158 Korea

요 약

대용량 데이터가 쏟아져 나오고 있는 지금, 대용량 데이터의 처리를 위해 데이터 처리 특성상 쓰기 작업보다 읽기 작업이 많아 소프트웨어 아키텍처나 데이터 저장 패턴에 많은 변화가 가해지고 있다. 이에 따라 본 논문에서는 명령(Command)과 쿼리(Query)의 책임을 분리하는 CQRS(Command Query Responsibility Segregation) 패턴의 쿼리 모델을 사용해 사용자의 요구사항에 효율적인 대용량 데이터 조회 시스템을 구현한다. 본 논문의 대용량 데이터는 기상청 Open API의 2018년 온도, 습도, 강수 데이터를 활용하여 약 23억 건의 데이터를 RDBMS(PostgreSQL)와 NoSQL(MongoDB)에 알맞게 저장한다. 또한 구현한 웹 서버 (Web Server) 입장에서의 CQRS패턴을 적용한 시스템과 CQRS 패턴을 적용하지 않은 시스템의 성능, 각 데이터베이스의 저장구조 성능, 데이터 처리 특징에 맞는 성능을 비교 및 분석한다.

ABSTRACT

At a time when large amounts of data are being poured out, there are many changes in software architecture or data storage patterns because of the nature of the data being written, rather more read-intensive than writing. Accordingly, in this paper, the query model of Command Query Responsibility Segmentation (CQRS) pattern separating the responsibilities of commands and queries is used to implement an efficient high-capacity data lookup system in users' requirements. This paper uses the 2018 temperature, humidity and precipitation data of the Korea Meteorological Administration Open API to store about 2.3 billion data suitable for RDBMS (PostgreSQL) and NoSQL (MongoDB). It also compares and analyzes the performance of systems with CQRS pattern applied from the perspective of the web server (Web Server) implemented and systems without CQRS pattern, the storage structure performance of each database, and the performance corresponding to the data processing characteristics.

키워드 : CQRS, RDBMS, NoSQL, Java, JSON

Keywords : CQRS, RDBMS, NoSQL, Java, JSON

Received 25 March 2019, **Revised** 10 April 2019, **Accepted** 5 May 2019

*** Corresponding Author** Hyun-Ju Park(E-mail:phj@hanbat.ac.kr)
Professor, Department of Information and Communication Engineering, Hanbat National University, Daejeon, 34158 Korea

Open Access http://doi.org/10.6109/jkiice.2019.23.6.645 print ISSN: 2234-4772 online ISSN: 2288-4165

ⓒThis is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License(http://creativecommons.org/li-censes/by-nc/3.0/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
Copyright Ⓒ The Korea Institute of Information and Communication Engineering.

Ⅰ. 서 론

정보기술의 발전으로 말미암아 대용량 데이터가 지속적으로 증가하고 있고 정부나 기업에서는 Open API를 통해 많은 데이터를 제공하고 있다. Open API는 누구나 사용할 수 있도록 공개된 공개 API를 말하며 대표적으로 공공데이터포털, 기상자료개방포털이 있다. 생활과 밀접한 관련이 있는 교통, 기상, 환율, 관광사진 등 다양한 정형/비정형 데이터를 JSON, XML과 같은 데이터 형식으로 제공하고 있다. 공공데이터법 제 1조, 제 3조에 따르면 누구나 이용가능하고, 영리 목적의 이용을 포함한 자유로운 활용이 보장된다. 이처럼 Open API에서 제공하는 다양하고 광대한 데이터를 활용해 개발자들은 다양한 분야에서 응용 시스템을 설계 구현하여 서비스하고 데이터를 분석한다[1] [2]. 또한 페이스북, 트위터, 인스타그램 등 소셜 네트워크의 발전으로 인해 스키마가 없는(Schema-less) 비정형 데이터가 많이 발생하고 있다. 비정형 데이터는 이미지, 동영상, 문서 등 특정 형태와 복잡한 구조를 가져 정형화되지 않은 데이터를 말한다[3]. 비정형 데이터를 RDBMS에 저장하기 위해서는 RDBMS에 맞는 구조 설계부터 저장 및 사용, 스케일업, 스케일아웃에 많은 비용이 발생한다. 이 밖에도 고정된 스키마, 구성과 확장 같은 제한사항이 많은 RDBMS는 대용량, 비정형 데이터에 대한 수집, 처리속도, 효율성의 문제로 비정형 데이터를 저장하기 부적합하다. 또한, 많은 양의 데이터에 대한 통계를 내기 위해서는 모든 데이터를 추출하여 통계를 내야함으로 데이터의 양이 많아질수록 속도가 저하되는 문제점이 존재한다.

본 논문에서는 앞서 설명한 데이터를 활용하고 기존 저장 방식의 문제점을 개선하기 위해 CQRS 패턴과 NoSQL의 문서지향 데이터베이스인 MongoDB를 사용하여 대용량 데이터를 요구사항에 맞게 효율적으로 처리하는 CQRS 패턴의 조회 모델이 적용된 시스템을 설계 구현한다.

본 논문의 구성은 다음과 같다. (관련연구)에서는 대규모 환경에서 확장이 용이한 아키텍처 패턴인 CQRS와 다양한 데이터 유형을 다루는 서비스가 증가하면서 데이터 유연성이 부각된 NoSQL에 대해 살펴본다. (본문)에서는 제안하는 시스템을 소개하고, 시스템의 구조와 특징, 그리고 기존 시스템과의 차별성을 설명한다. (성능 및 분석)에서는 본 논문에서 제안한 시스템의 성능 검증을 실시하고 (결론 및 향후 연구) 에서는 논문을 맺으며, 향후 연구에 대해 기술한다.

Ⅱ. 관련연구

2.1. JSON

JSON(JavaScript Object Notation)은 경량화된 데이터 교환 형식이다. XML(Extensible Markup Language)은 모든 데이터를 태그로 저장하여 사용하며 연속된 데이터 형태는 보안이 보장되지만, 데이터 크기에 비해 효율이 낮아지고 DOM(Document Object Mode)을 통한 객체를 파싱(Parsing)하는 데 오래 걸리는 문제점이 있다. 이를 대신할 데이터 교환 방식으로 JSON이 대두되었다[4]. JSON은 자바스크립트(JavaScript) 언어로부터 파생되어 자바스크립트와 비슷한 구문이나 프로그래밍 언어로부터 독립적이고 프로그래머들이 친숙하게 사용하는 텍스트형식이다[5].

2.2. CQRS

CQRS(Command Query Responsibility Segregation)는 2010년 'Greg Young'이 제안한 개념으로 별도의 인터페이스를 활용해 데이터를 등록/갱신하는 작업과 데이터를 조회하는 작업을 분리하는 아키텍처 패턴이다. CRUD(Create, Read, Update, Delete) 연산으로 비교하면, 데이터를 등록/갱신하는 작업은 Create, Update, Delete이고, 데이터를 조회하는 작업은 Read로 구분할 수 있다.

CQRS를 적용하는 방법으로는 최소 3가지가 알려져 있다. 첫 번째, 가장 간단한 적용 방법으로 응용 시스템에서 명령 모델과 쿼리 모델을 분리하고 하나의 RDBMS를 사용하는 것이다. 두 번째, 응용 시스템에서 명령 모델과 쿼리 모델을 분리하고 명령 모델과 쿼리 모델은 각 시스템에 알맞은 데이터베이스 선택 가능하다. 마지막으로는 응용시스템에서 발행하는 모든 활동들을 로그로 변환해 별도의 저장소에 저장하는 이벤트 소싱(Event Sourcing) 패턴을 적용한 것이다. 이벤트 소싱은 오직 저장만이 가능하고 수정 및 변경은 불가능하다.

2.3. NoSQL

NoSQL 시스템은 광범위한 데이터에 대해 높은 액세

스 가능성, 안정성 및 확장성을 제공하기 위해 고안된 비 관계형 데이터베이스이다. 데이터 모델에 기반해 집계 쿼리, 대용량 데이터의 압축, 분산처리 등에 뛰어난 컬럼 지향(column oriented), 키-값 하나의 묶음 형태로 저장되는 구조로 속도가 빠른 키-값 지향(key-value oriented), 데이터를 노드로 표현하며 유지보수가 간편한 그래프 지향(graphs oriented), 동적 스키마로 인해 유연한 문서 지향(document oriented)으로 분류된다[6]. 대표적으로 컬럼 지향에서는 Cassandra, 키-값 지향에서는 Redis, 그래프 지향에서는 Neo4J, 문서 지향에서는 MongoDB가 있다.

본 논문에서는 동적 스키마를 가지고 있어 데이터 형식에 다소 자유로운 MongoDB를 사용한다. MongoDB는 10gen이라는 미국 회사에서 개발한 오픈소스 DBMS이다. 데이터베이스 관리 시스템의 인기도 순위를 평가하는 DB-engines에 따르면 NoSQL의 문서 지향 데이터베이스 중 MongoDB가 1위를 기록했다[7]. MongoDB는 키와 필드 쌍으로 구성된 데이터 구조 문서로써 JSON 객체와 유사하다. 키-값, 범위, 검색 질의 등 다양하고 강력한 쿼리 기능이 특징이며 동적 스키마를 제공해 다형성을 지원하며 이베이, 페이스북 등 여러 웹 서비스에 널리 사용되고 있다.

Ⅲ. 본문

본 논문은 웹 프로토콜인 HTTP를 사용하여 요청을 처리하며, 문서 지향 데이터베이스인 MongoDB, 오픈소스 데이터베이스인 PostgreSQL와 Java Web Framework 중 하나인 Spring boot 이용하여 CQRS 패턴의 조회 모델을 구현한다. Open API인 기상자료개방포털에서 제공하는 기상 데이터를 활용하여 요구사항에 맞게 효율적으로 처리한다.

3.1. 전체 시스템 구조

CQRS 패턴에서 조회 모델에 대해 설계 구현한 아키텍처(CQRS-Query)는 그림 1과 같은 구조를 가진다. 첫째, Open API인 기상자료개발포털로부터 데이터를 수집해 전처리 과정을 하는 데이터 수집 및 전처리. 둘째, CQRS 패턴의 조회 모델을 구현한 CQRS-Query로 Controller 모듈은 본 시스템의 객체 생성 및 소멸, 제어 동작을 총괄한다. 바 차트, 파이 차트, 꺾은선 그래프에 맞는 동작을 수행하도록 명령을 내리는 모듈이다. Application 모듈은 요청에 대한 응답을 실질적으로 처리하며 요청으로부터 알맞은 데이터를 데이터베이스로부터 질의한다. 질의의 결과로 나온 객체들은 Domain 모듈 중 하나의 객체에 리스트 형식으로 저장된다. Domain 모듈은 각 차트마다 필요한 데이터를 가져오기 위한 스키마와 인터페이스를 정의한다.

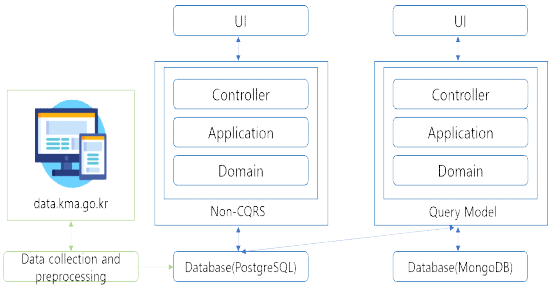

Fig. 1 Architecture

3.2. 데이터 수집 및 전처리

설계 구현한 시스템의 데이터 수집 시 데이터 저장과 저장 구조는 그림 2와 같다.

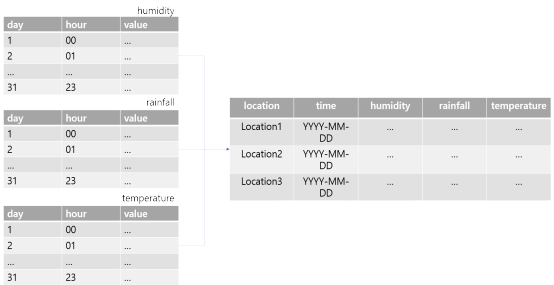

Fig. 2 Schema

데이터 수집은 기상자료개방포털에서 이루어지며 시간 - 습도/강수량/온도로 나뉘어져 있는 데이터를 지역, 시간, 습도, 강수량, 온도 데이터 구조로 데이터 처리를 거친 후 PostgreSQL에 저장된다. 데이터 조작 및 구조를 제공하는 파이썬 라이브러리인 Pandas를 사용하여 대전시의 총 79개 동마다 습도, 강수량, 온도로 나뉘어 있는 1년 치 데이터를 위치, 시간, 습도, 강수, 온도 하나의 데이터 구조로 합친 후 csv파일로 ',' 구분자로 만들어

PostgreSQL에 미리 생성한 (위치, 시간, 습도, 강수량, 온도) 스키마를 가진 daejeon 테이블에 COPY 명령에 의해 저장한다. 시간 별, 동 별로 질의가 많아 시간, 동 순으로 결합 인덱스를 생성하였다. CQRS-Query는 PostgreSQL에 질의를 통해 통계 데이터를 MongoDB에 JSON 형태로 저장한다. 저장 구조는 표 1과 같다.

Table. 1 NoSQL Documents

rainfall	humidity	temperature
{ "location":“내동", “totalRainfall":"3166230.42” }	{ "location":"신성동", "avgHumidity": "18.38" }	{ "location":"가수원동”, hour":0, "temperature" :"30.18” }

3.3. CQRS-Query

본 시스템은 웹 기반의 시스템을 구현하기 위해 Spring boot mvc 를 사용하였고, 데이터 형식은 JSON을 사용한다. 처리속도 향상을 위해 database server와 연결을 미리 저장해 두는 database connection pool인 HikeriCP를 사용한다. 첫 구동 시 CQRS-Query는 PostgreSQL에 모두 높은 값을 나타낼 것으로 예상되는 여름(6~8월)에 대해서 9개 동의 평균 습도 데이터, 대전 전 지역의 여름에 대한 총 강수량, 대전 전체 동의 여름에 대한 시간별 평균 온도를 질의해 NoSQL에 저장한다. 저장된 통계 데이터를 기반으로 사용자에게 다양한 UI(User Interface)를 제공하기 위해 습도, 강수량, 온도를 경향을 한눈에 알아보기 쉬운 바 차트, 지속적인 변화의 흐름을 알아보는 꺾은선 그래프, 전체의 비율을 알아보기 쉬운 파이 차트로 표현한다. 습도는 여름에 대한 9개 동의 평균 습도 데이터를 바 차트에 표현하고 강수량은 대전 전 지역의 여름에 대한 총 강수량을 꺾은선 그래프로 표현한다. 마지막으로 온도는 특정 동의 여름에 대한 시간별 평균 온도를 파이 차트로 표현한다.

3.4. 시스템 구성

전체 구성은 다음 표 2과 같이 크게 3가지로 나뉘어 진다. HTTP 요청을 위한 Client, 요청 및 데이터 처리를 위한 Server, 데이터저장을 위한 Database Server로 구성했다. 시스템이 구동되는 PC의 CPU는 Intel Core i5 6360U (2 GHz)이고, 메모리는 8 GB이다.

Table. 2 Operating environment

Composition	Software
Client	Chrome v72.0.3626.119
	postman v 6.7.3
Server	spring boot
	java 11
DBMS	MongoDB v3.6.3
	PostgreSQL v10.6

Ⅳ. 성능 분석

4.1. 테스트 환경 구축 및 시나리오 구상

시스템의 성능 분석을 위해 동일 환경, 동일 구조로 CQRS 패턴의 조회 모델을 구현한 시스템(CQRS-Query)과 CQRS 패턴이 적용되지 않은 시스템(Non-CQRS)을 구현 및 구동한다. 각 시스템의 성능은 다음 표 3과 같다. 시스템 구현 환경과 테스트 도구 구동 환경은 macOS를 사용한다. 테스트 도구는 성능분석을 위해 오픈소스 소프트웨어인 Apache JMeter를 사용한다. CQRS-Query와 Non-CQRS는 테스트를 위해 동일한 환경으로 구성했다.

Table. 3 Test environment

	CQRS-Qeury	Non-CQRS	macbook	Apache Jmeter
OS	Ubuntu 18.04.2 LTS		macOS Mojave v10.14.3	
CPU	Intel(R) Xeon(R) CPU E5-2680 2.70GHz (8C 16T) 4 vCPUs		Intel Core i5 6360U (2 GHz)	
Memory (GB)	6		6	

Non-CQRS와 CQRS-Query 시스템은 동일한 환경에서 구현 및 구동함에 따라 테스트는 각각 진행한다. 테스트 데이터는 기상자료개방포털에서 파이썬(Python) 언어로 데이터를 수집(Crawling)한다. 대전광역시(79개 동)의 2018년 온도, 습도, 강수량 데이터를 각 동당 약 3100만건의 데이터, 약 총 25억건의 데이터를 저장하여 활용한다. CRUD 요청 중 Read 요청에 대한 성능 테스트를 진행하여 macbook이 처리할 수 있는 동시 접속자의 수를 확인하고 평균 응답시간을 산출한다. 또한 데이

터 저장구조가 잘 설계되었는지 저장구조 성능에 대한 분석을 진행한다.

4.2. 성능 테스트 결과

이 절에서는 4.1절을 토대로 진행한 성능 테스트에 대하여 요약한다. 성능 테스트는 GET 요청에 대한 요약이며 CQRS 패턴의 조회 모델에 대한 설계 구현임으로 GET 요청만 진행한다. 각 동시 사용자가 0.3초 간격으로 요청하며 동시 사용자 수에서 100번 반복한다.

첫 번째 성능 테스트는 각 시스템에 대한 동시 사용자 1명의 대전 전체 동의 시간별 여름 평균 온도에 대한 요청이다. 대전 전체 동의 여름에 대한 시간별 여름 평균 온도에 결과는 다음 그림 3과 같다. 1명의 사용자가 사용할 동안 0%의 에러율을 보이며 안정적인 결과를 보이지만 응답 속도에서 대략 11배 정도 차이가 발생해 CQRS의 조회 모델의 성능이 뛰어난 것을 확인했다.

	Average (m/s)	Min(m/s)	Max(m/s)	Error(%)
CQRS-Query	463	463	463	0
Non-CQRS	5196	5196	5196	0

Fig. 3 Summer Average temperature result

이로 인해 동시 사용자 수가 늘어난다면 더 이상의 조회 성능 분석이 불가능해 Non-CQRS의 DBMS에 CQRS-Query의 DBMS와 같은 통계 데이터를 저장하는 구조를 생성해 동일 조건의 성능 테스트를 진행 후 비교분석한다.

9개 동('용전동', '은행선화동', '둔산2동', '원신흥동', '신탄진동', '노은3동', '가수원동', '신성동', '판암2동') 여름 평균 습도의 GET 요청의 결과는 다음 그림 4와 같다. 에러율은 동시 사용자가 100명, 500명, 1000명이 사용할 동안 0%의 에러율을 보이며 안정적인 결과를 보인다. 처리 성능의 경우 Non-CQRS와 CQRS-Query 둘 다 동시 접속자 1000명 사용 시 최대 처리량을 보여주고 있다. 평균 응답속도의 경우 두 시스템 모두 평균 응답속도가 1초가 넘지 않으며 Non-CQRS는 지속적으로 증가하며 CQRS-Query의 경우, 500명의 사용자부터는 동일한 응답속도를 보인다.

	Non-CQRS			CQRS-Query		
Thread	100	500	1000	100	500	1000
Roof	100	100	100	100	100	100
Average(ms)	12	27	30	12	15	15
Min(ms)	4	4	4	5	5	4
Max(ms)	238	1916	1044	232	1269	1837
Error(%)	0	0	0	0	0	0
Throughout	323.6	330.1	331.8	324.5	330.7	331.4

Fig. 4 Average humidity result

대전 전 지역의 여름 총 강수량의 GET 요청의 결과는 다음 그림 5와 같다. 에러율은 동시 사용자가 1000명까지 증가할 동안 에러율은 0%로 안정적인 결과를 보인다. 처리 성능의 경우 동시 접속자 수가 1000명일 때 최대 처리량을 보여준다. 응답 속도의 경우 사용자가 증가함에 따라 응답 속도가 증가하는 것을 확인할 수 있다. 동시 사용자 100명일 경우 Non-CQRS가 조금 빠른 경향을 보이지만 동시 사용자 500명 이상부터는 비슷한 결과를 보이고 있다.

	Non-CQRS			CQRS-Query		
Thread	100	500	1000	100	500	1000
Roof	100	100	100	100	100	100
Average(ms)	16	27	29	20	24	28
Min(ms)	6	6	5	7	6	5
Max(ms)	205	651	1250	587	1371	779
Error(%)	0	0	0	0	0	0
Throughout	319.5	328.3	331.0	317.4	328.9	331.2

Fig. 5 Total rainfall result

특정('내동') 동의 시간별 여름 평균 온도의 GET 요청의 결과는 다음 그림 6과 같다. 에러율은 동시 사용자가 1000명까지 증가할 동안 에러율은 0%로 안정적인 결과를 보인다. 처리 성능의 경우 동시 접속자 수가 1000명일 경우까지 증가해 최대 처리량을 보인다. 평균 응답 속도의 경우, Non-CQRS는 지속적으로 증가하고 CQRS-Query는 동시 접속자 수가 500명, 1000명일 때 동일한 평균 응답속도를 보인다. 평균 응답 속도가 큰 차이는 없으나 CQRS-Query가 빠른 결과를 보이고 있다.

	Non-CQRS			CQRS-Query		
Thread	100	500	1000	100	500	1000
Roof	100	100	100	100	100	100
Average(ms)	18	21	27	10	14	14
Min(ms)	7	5	6	4	4	4
Max(ms)	234	1505	1551	233	402	1147
Error(%)	0	0	0	0	0	0
Throughout	315.1	329.7	331.0	323.7	330.8	331.9

Fig. 6 Average temperature result

Ⅴ. 결론 및 향후 연구

4절. 성능 평가에서는 같은 통계 데이터를 조회함으로써 MongoDB와 PostgreSQL은 차이가 미세하게 나타나는 것으로 확인되었으나 Non-CQRS의 PostgreSQL이 통계 데이터가 아닌 원본 데이터를 가지고 동일한 결과를 반환하는 질의라면 차이는 급격히 커질 것이다. MongoDB는 JSON 형식으로 저장이 가능해 데이터를 가져와 바로 반환이 가능하다. 하지만 PostgreSQL은 데이터를 가져와 JSON으로 변환하는 연산이 추가로 필요하다. 많은 관계형 데이터베이스와 PostgreSQL에서 JSON 형식을 지원하고 있지만 MongoDB를 CQRS의 DBMS로 사용한 이유는 규모 확장이 용이하다는 것이다. 관계형 데이터베이스도 규모 확장이 가능하지만 ,규모 확장 시 많은 비용이 필요하고, 어느 정도의 확장을 해야 할지 가늠할 수 없다는 문제점이 있다.

본 논문에서는 정보 기술의 발전으로 인해 대용량 데이터가 쏟아져 나오는 환경에서 CQRS 패턴의 조회 모델을 적용한 시스템을 설계 구현 및 분석하였다. 본 논문에서 설계 구현한 CQRS 패턴의 조회 모델을 적용함으로써 일반적으로 시스템이 하는 변경 및 저장 연산, 조회 연산 중 조회 연산만을 담당해 대용량 데이터를 신속히 처리할 수 있는 시스템을 구축할 수 있다. 또한 CQRS 패턴의 명령 모델을 적용한다면 UI(User Interface)에 표현되는 방법에 따라 알맞은 데이터 형식으로 저장해 유지보수가 편리해진다.

향후 연구에서는 앞서 2절 관련 연구에서 간단히 소개한 시스템 상에 발생하는 모든 활동들을 이벤트로 변환해 별도의 저장소에 저장하고 변경이 불가하고 저장만 가능하다는 특징이 있는 이벤트 소싱 패턴과 CQRS 패턴의 저장, 수정, 삭제의 역할을 수행하는 명령 모델을 조합해 사용하면 데이터 분석 분야에서 큰 시너지 효과가 발생할 것으로 기대된다[8] [9]. CQRS 패턴을 적용함으로써 시스템의 복잡성이 증가할 수 있다는 단점이 있다. 이를 해결하기 위해 DDD(Domain Driven design) 소프트웨어 개발 방식을 적용해 복잡성을 낮출 수 있을 것으로 기대한다[10].

ACKNOWLEDGEMENT

This research was supported by a grant(13 AUDP-B070066-01) from Architecture & Urban Development Research Program funded by Ministry of Land, Infrastructure and Transport of Korean government.

This research was supported by Basic Science Research Program through National Research Foundation of Korea(NRF) funded by the Ministry of Education(2018R1D1A1B07043948)

REFERENCES

[1] Public Data Portal, www.data.go.kr [Internet], Available: https://www.data.go.kr.

[2] S. Y. Bang, H. D. Ha, C. J. Kim, “A Study on BigData-based Software Architecture Design for Utilizing Public Open Data,” *Journal of KIIT*, vol. 13, no. 10, pp. 99-10, 2015.

[3] B. H. Back, I. K. Ha, B. C. Ahn, “An Extraction Method of Sentiment Infromation from Unstructed Big Data on SNS,” *Journal of Korea Multimedia Society* vol. 17, no. 6, pp. 671-680, Jun. 2014.

[4] J. S. Oh, C. G. Song, “Transmission performance of improvements in mobile applications via XML and JSON data translation,” *Journal of Information Science Society*, vol.39, no. 1, pp. 129-131, 2012.

[5] json, www.json.org [Internet], Available : http://www.json.org/

[6] P. Gómez, R. Casallas, and C. Roncancio, “Data schema does matter, even in NoSQL systems!,” *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)*, Grenoble, pp. 1-6, 2016.

[7] DB-ENGINES, www.db-engines.com [Internet], Available : https://db-engines.com/en/ranking.

[8] K. HAKIM, “Correctness for CQRS Systems,” KTH Royal Institute of Technology, Stockholm, 2012.

[9] M. Overeem, M. Spoor, and S. Jansen, “The dark side of event sourcing: Managing data conversion,” *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, Klagenfurt, pp. 193-204, 2017.

[10] C. B. Kyun, DDD START! Learn domain-driven design implementation and key concepts, 1th ed. Republic of Korea, KR : Seoul, 2016.

서보민(Bomin Seo)

2018년 2월 : 한밭대학교 정보통신공학과 (공학사)
2018년 3월 ~ 현재 : 한밭대학교 모바일융합공학과 석사과정
※관심분야 : 데이터베이스, 웹 프로그래밍

전철호(Cheolho Jeon)

2018년 2월 : 한밭대학교 정보통신공학과 (공학사)
2018년 3월 ~ 현재 : 한밭대학교 모바일융합공학과 석사과정
※관심분야 : 데이터베이스, 웹 프로그래밍

전현식(Hyeonsig Jeon)

2005년 2월 : 한밭대학교 전파공학과 (석사과정)
2011년 2월 : 한밭대학교 전파공학과 (박사과정)
※관심분야 : 데이터베이스, 공간 데이터베이스, 위치 추정 알고리즘, ESL etc.

안세윤(Seyun An)

1998년 2월 : 이화여자대학교 산업디자인학과(학사)
2002년 2월 : 연세대학교 디자인경영(이학석사)
2011년 2월 : 연세대학교 실내건축학과(이학박사)
2013년 8월 ~ 현재 : 한밭대학교 산업디자인학과 부교수
※관심분야 : 실내 · 환경디자인, 감성마케팅

박현주(Hyun-ju Park)

1990년 2월 : 서울시립대학교 전산통계학화(공학사)
1992년 2월 : 서울대학교 전산과학과(공학석사)
1997년 2월 : 서울대학교 전산과학과(공학박사)
1998년 4월 ~ 2000년 3월 : 대전 산업대학교 정보통신학과 전임강사
2000년 4월 ~ 현재 : 국립 한밭대학교 정보통신공학과 교수
※관심분야 : 프로그래밍 언어, 운영체제, 데이터베이스