Journal of Korea Multimedia Society Vol. 23, No. 12, December 2020(pp. 1496-1506)
https://doi.org/10.9717/kmms.2020.23.12.1496

개인 건강 정보 처리를 위한 배치 어플리케이션에서 데이터 질의 속도 향상을 위한 PHDItemReader 설계 및 구현

전동철†, 황희정††

Desgin and Implementation of PHDItemReader to Speed up Data Query in Batch Application for Processing Personal Health Record

Dong-cheol Jeon†, Heejoung Hwang††

ABSTRACT

With the progress of miniaturization and high performance of various sensors, a lot of data is generated in various fields and being collected in real-time, but the use of such large-capacity data is often unable to keep up with the collection technology. In the medical field, health data is collected and managed by platform, which causes inconvenience to users in searching their own health data and receiving medical services. In this paper, in order to solve these problems, we designed and implemented PHDItemReader to improve the speed of data query in a batch application environment that can integrate and process health data having various data expression formats. The experiment compared and analyzed 3 types of query speed based on 1,000,000 hypothetical health data, and as a result of the experiment, it was verified that the PHDItemReader implemented in this paper improved up to about 21% compared to the existing one.

Key words: PHR, Spring, Batch, MongoDB

1. 서 론

4차 산업 혁명으로 스마트폰, 웨어러블 디바이스, IoT 센서 등 다양한 경로를 통하여 데이터가 수집되고 있다. 이로 인하여 처리해야 하는 데이터의 양이 과거에 비하여 급격하게 증가하고, 이를 웹 혹은 모바일을 통하여 사람들에게 보여주기 위한 데이터 처리 기술의 중요성이 증가하게 되었다[1]. 일반적으로 대용량의 데이터를 효율적으로 처리하는 방법으로는 일괄 처리(Batch process) 시스템이 있는데, 일괄 처리란 대용량의 데이터 작업을 정해진 시간에 사용자와의 상호작용을 최소화하고, 자동화된 형태로 수행되는 데이터 처리를 말한다. 이러한 일괄 처리 시스템의 특징은 다음과 같다[2].

- 사용자의 간섭이 최소화된 상태로 작업이 수행되어야 한다.
- 정해진 시간 제약 내에 실행이 완료되어야 한다.
- 많은 자원이 소모되는 대용량 작업이어야 한다.

과거 대부분의 일괄 처리는 쉘 스크립트에서 일괄

※ Corresponding Author : Heejoung Hwang, Address: (13120) Seongnamdaero 1342, Seongnam-si, Gyeongi-do, Korea, TEL : +82-31-750-4758, E-mail : hwanghj@gachon.ac.kr
Receipt date : Sep. 7, 2020, Revision date : Oct. 28, 2020
Approval date : Nov. 23, 2020
† Dept. of IT Convergence Engineering, Gachon University (E-mail : aseah@naver.com)
†† Dept. of Computer Engineering, Gachon University (E-mail : hwanghj@gachon.ac.kr)
※ This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2020-2017-0-06130) supervised by the IITP(Institute for Information & Communications Technology Promotion)

처리해야하는 작업을 작성하고, 레거시 코드부분에는 해당 작업을 수행하기 위한 스케줄을 설정하는 방식으로 수행되었다. 이는 고도의 쉘 스크립트에 대한 이해가 필요했고, 복잡한 일괄 처리 시스템을 구현하는 것에는 한계가 있다. 이를 해결하기 위하여 node.js에서 Kue, Agenda, Node-schedule 등 다양한 라이브러리를 통하여 일괄 처리 시스템을 해당 어플리케이션 안에서 구현할 수 있도록 하였다[3]. 하지만, node.js를 통한 일괄 처리 어플리케이션은 단순 일괄 처리 작업에는 유용했지만, 작업의 규모가 커지고 복잡해지면 구현하는 것에 한계가 있다. 또한, 기존의 배치 어플리케이션에서 발생하는 공통적인 문제는 다음과 같다[4].

- 배치 처리 과정에서 사용되는 로깅, 트랜잭션, 데이터 소스의 관리 등 부가적인 부분도 신경써야한다.
- 배치 작업에 대한 테스트와 에러 추적에 대한 어려움이 존재한다.
- 배치의 세부적인 작업 분할이 이루어지지 않고, 대부분 하나의 클래스 안에서 수행되어 배치 처리 어플리케이션의 소스 관리에 어려움을 유발한다.

이러한 문제점들을 해결하기 위하여 자바 기반의 프레임워크인 스프링 프레임워크에서는 기존의 스프링 라이브러리들을 사용하면서 배치 처리 작업을 지원할 수 있는 스프링 배치(spring batch)라는 모듈을 개발했다. 이를 통하여 배치 처리 어플리케이션을 개발하는 과정에서 발생하는 문제점들을 보완하고 기존의 스프링에서 지원하는 기능들을 사용하여 순수 배치 작업 부분에만 집중할 수 있도록 했다.

의료 분야에서는 과거에 병원 및 약국 등 의료 기관에서만 건강 데이터가 수집되었지만, 최근 IT 융복합 기술의 발전으로 의료 기관뿐만 아니라 일상 생활에서도 다양한 센서를 통해 개인 건강 기록(Personal Health Record, PHR)이 수집되고 있다[5]. 하지만, 이렇게 다양한 경로를 통해서 수집된 개인 건강 기록은 건강 관리 플랫폼별로 관리가 되고, 서로 다른 데이터 표현 형식을 갖고 있다. 이로 인해 개인 건강 데이터를 통합하기 위한 여러 연구들이 진행되었으며 법 제도적인 틀 안에서 다양한 시스템들이 시범적으로 개발되거나 부분 활용되고 있다[6]. 하지만 건강 데이터를 통합하는 과정에서 발생하는 대량의 데이터를 일반적인 방식으로 처리하면, 서버의 과부하를 유발하거나 데이터 표현 형식이 맞지 않는 문제가 발생한다.

본 논문에서는 이러한 개인 건강 정보 통합 플랫폼에서 다양한 경로로 수집된 대용량의 건강 데이터의 일괄 처리 시 발생하는 성능 문제 해결을 목표로 한다. 문제 해결을 위해 MongoDB와 스프링 배치 기반의 배치 어플리케이션 구현의 핵심 모듈인 Item Reader를 개선해 건강 데이터 처리에 최적화한 PHDItemReader를 설계하고 구현하였다.

2. 관련 연구

2.1 Spring batch

스프링 배치(spring batch)는 SpringSource와 Accenture사가 2007년에 개발한 대용량 환경에서 실행하는 배치 어플리케이션에서 빠른 대처와 유연한 시스템 개발 및 유지보수를 위한 스프링의 배치 어플리케이션 개발 모듈이다. 스프링 배치의 특징은 다음과 같다.

- 스프링 프레임워크에서 배치 작업에 필요한 로깅, 트랜잭션, 데이터 소스 관리 등을 지원한다.
- 스프링 기반의 기본적인 인프라를 통하여 배치 어플리케이션의 안정화가 용이하다.
- 작업을 여러 단계로 나누어서 처리하거나, 병렬 처리하는 등 배치 처리의 흐름을 설정할 수 있다.
- 배치 작업 내에서 세부적인 작업 분할로 에러 추적과 재시작점 파악이 용이하다.

스프링 배치의 기본 구조는 Fig. 1과 같다[7].

- JobRepository : 배치 전체 작업(Job) 및 배치 세부 작업(Step)의 상태를 관리하는 저장소로, 스프링 배치에서 지정한 테이블 스키마를 기반으로 데이터베이스에 유지된다.
- JobLauncher : 배치 전체 작업을 실행하기 위한 인터페이스
- Job : 배치 어플리케이션에 대한 일련의 처리 흐름을 요약한 단일 실행 단위
- Step : 배치 전체 작업을 구성하는 하나의 처리 단위로, Job 1개는 N개의 Step으로 구성된다.

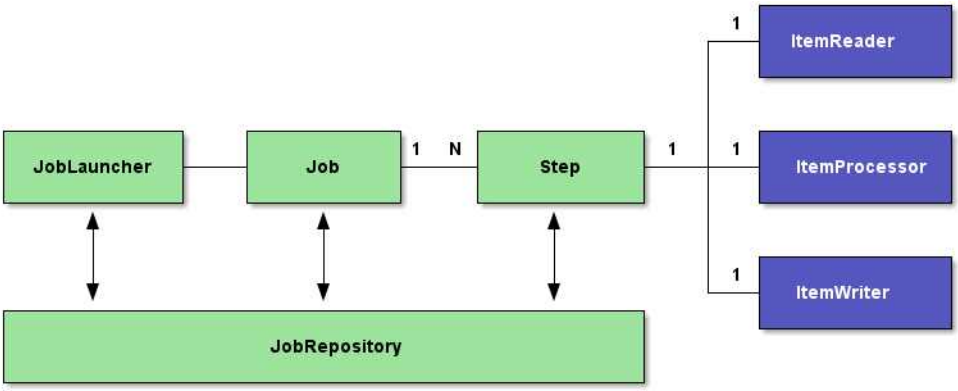

Fig. 1. Basic structure of spring batch.

- Reader/Processor/Writer : 데이터 묶음 기반 처리(Chunk-oriented processing)에서 데이터를 읽고, 가공하고, 처리하기 위한 인터페이스이다.

데이터 묶음 기반 처리란 데이터를 데이터의 묶음(Chunk) 단위로 일괄 처리하는 방식으로, 배치 어플리케이션은 대용량의 데이터 처리를 목표로 하기 때문에, 데이터베이스의 접근 빈도, 자원 사용, 처리 속도 측면에서 효율적으로 하기 위한 스프링 배치에서 사용하는 처리 방식이다. 데이터 묶음 기반 처리의 프로세스는 Fig. 2와 같다[8].

데이터 묶음 기반 처리의 데이터 처리 과정은 다음과 같다.

- 데이터 소스에서부터 ItemReader를 통하여 단일 데이터(Item)을 읽는다.

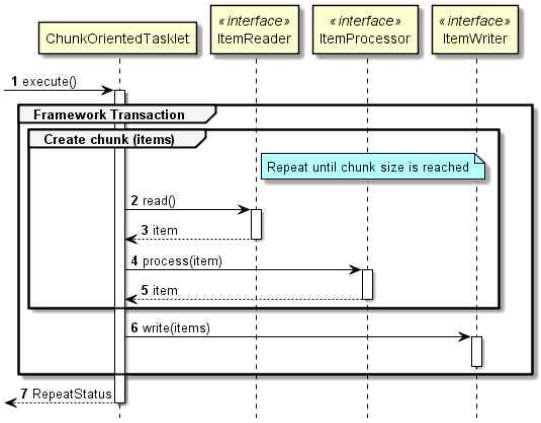

Fig. 2. Process of chunk-oriented processinig.

- 읽은 데이터를 ItemProcessor에서 받아 변환 및 검증 작업을 진행한다.
- 데이터 묶음의 크기만큼 데이터가 처리될 때까지 위의 작업을 반복 수행한다.
- ItemWriter는 데이터의 묶음(items)에 대하여 수정/삭제/출력 작업을 수행한다.
- 요청한 모든 데이터에 대하여 전체 과정을 반복 수행한다.

2.2 PHR

PHR(Personal Health Record)은 개인으로부터 발생하는 개인 고유의 건강 기록으로, 생체 정보 등의 라이프로그, 진단 기록 및 유전체 정보 등 개인의 건강과 관련된 모든 것을 포함하고 있는 정보를 말한다[9]. PHR 시스템의 활용을 통하여 중복 처치나 진료 과정의 감소 혹은 제거, 비용과 시간의 절감이 가능할 것으로 기대되고 있다[10]. 최근 PHR을 활용한 서비스는 진료 기록, 운동량 정보, 라이프로그 기록, 유전체 분석 결과 등을 모두 포함하는 영역으로 확장되고 있으며, 웨어러블 디바이스 및 센서 기술 등 IT 융복합 기술의 발전에 따라 다양한 헬스케어 디바이스와 연계 및 통합되어 다양한 서비스를 만들어내고 있다[11].

최근 고령화 및 만성 질환의 문제가 화제가 되면서 병원과 같은 의료 기관에서만 건강 관리를 받는 것이 아닌 일상 생활에서의 건강을 신경써야하는 필요성이 증가하게 되었고, 그에 따라 사람들이 개인 건강 기록에 대하여 더 관심을 갖게 되었다[12]. 또한 4차 산업혁명으로 인하여 센서 기술이 발전하게 되

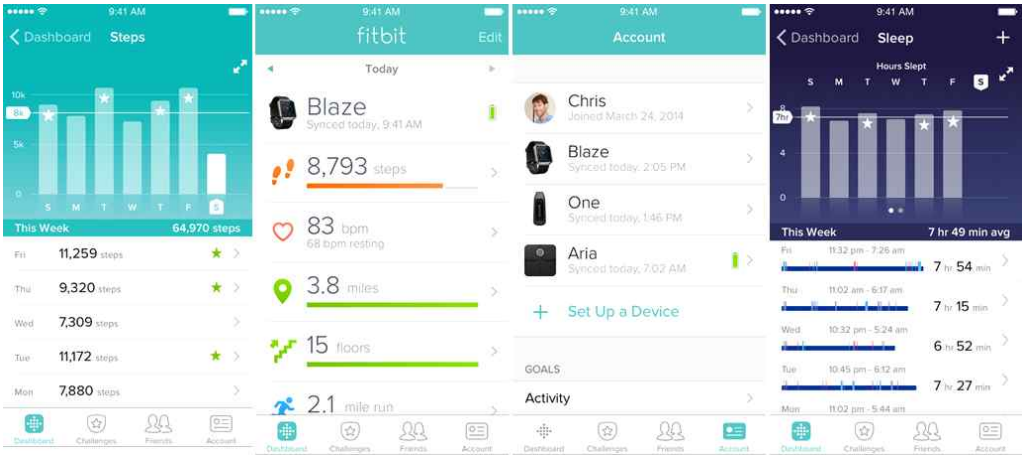

Fig. 3. Mobile dashboard of Fitbit's personal healthcare service.

고, 이에 따라 웨어러블 디바이스를 통하여 각종 생체 데이터를 수집할 수 있게 되었고, 이를 해당 개인 건강 관리 플랫폼 전용 모바일앱 혹은 웹을 통하여 조회할 수 있게 되었다[13]. 대표적으로 미국의 Fitbit은 스마트 워치 등 IoT 센서 기능이 탑재된 웨어러블 디바이스를 통하여 사용자의 건강 혹은 운동 정보를 수집하고, 수집된 건강 데이터를 전용 모바일 앱을 통하여 조회할 수 있고 이를 기반으로 건강 관리에 도움을 준다. Fig. 3은 Fitbit의 모바일 앱 화면이다[14].

3. PHR 스키마 및 ItemReader 설계

건강 데이터는 PHR 플랫폼별로 관리되어 한 사람의 건강 정보가 분산 저장되고 있다. 사람들은 이로 인하여 자신의 건강 데이터를 조회하는 것에 어려움을 겪고 있으며, 이를 위해 건강 데이터를 통합 관리할 수 있는 플랫폼이 필요하게 되었다. 건강 데이터를 통합 관리하는 플랫폼을 구현하는 과정에서 건강 데이터의 표현 형식이 다양해지고 데이터의 규모가 커져, 일반적인 방식으로 처리하게 되면 서버의 과부하를 유발 할 수 있다. 이를 해결하기 위하여 다양한 데이터 표현 형식에 대응할 수 있고, 대량의 건강 데이터를 일괄 처리할 수 있는 배치 어플리케이션이 필요하게 되었다.

3.1 PHR 데이터 스키마

기존의 RDBMS(Relational Database Management System)를 기반으로 다양한 데이터 표현 형식이 존재하는 건강 데이터를 처리하기 위해서는 데이터의 구조가 너무 정형화되어 있고, 비용과 효율적인 측면에서 한계를 가지고 있다. 건강 데이터를 통합 관리하기 위해서는 다양한 경로에서 수집된 건강 데이터를 관리할 수 있어야 하는데, 이를 해결하기 위하여 본 논문에서는 비정형 데이터 구조를 관리하는 NoSQL 중 하나인 MongoDB를 사용하여 다양한 데이터 표현 형식의 건강 정보에 대응할 수 있도록 하였다.

플랫폼별로 수집된 건강 데이터를 통합하여 처리하는 배치 어플리케이션에서 사용하고 처리 결과를 사람들에게 보여주기 위해서는 기존의 건강 데이터를 그대로 보존하면서 일정한 규칙에 따라 규격을 가지고 있어야 한다. ID 값은 데이터에 대한 고유키 값을 제외하고, 건강 데이터를 통합하는 과정에서 동일 사용자가 다양한 건강 관리 플랫폼을 이용하는 환경에서 동일한 사용자인지 여부를 판단하고 해당 사용자가 어떤 건강 관리 플랫폼에서 데이터를 발생시켰는지 구분하기 위한 각각의 ID 값이 필요하다. 또한 개인 건강 데이터는 다양한 경로를 통해서 수집되는데, 이는 대용량으로 누적이 되며, 이 중에서 특정 데이터를 찾기 위해서는 검색 조건을 통하여 조회할 수 있어야 한다. 이를 위하여 특정 날짜 기간 동안의 데이터를 검색하기 위한 날짜 데이터, 세부 건강 데이터를 모르더라도 키워드를 통하여 검색이 용이할 수 있도록 하는 키워드 데이터가 필요하다. 마지막으로 각 건강 관리 플랫폼에서 발생한 실제 건강

Table 1. Health data structure for unified repository.

Name	Type	Description
_id	String	Primary key of the health data
user_id	String	User id to classify the people and save the health data from various paths to a single person.
dept_id	String	Department id for classifying healthcare platform that health data came from it
date	Date	Date that is saved to repository
type	String	Health record's type which is divided to 4 types (Med_Record, Training, Life_log, Diag_predict)
tags	Array	Keywords to response the various types of health data. Tags can be used to find easy the health data even people don't know about the detail.
data	Object (JSON)	Real health data. It contains original platform's health data and data format is free JSON type.

데이터를 담는 데이터가 필요하다. Table 1은 요구사항에 따라 설계된 스키마 정보이다.

건강 관리 플랫폼별로 데이터가 상이하게 표현되기 때문에 비슷한 유형끼리 데이터를 묶어 해당 건강 데이터가 어떤 데이터를 표현하고 있는지 파악하기 쉽게 할 수 있어야하며, 조회가 용이할 수 있게 해야 한다. 본 논문에서는 MED_RECORD, DIAG_PREDICT, TRAINING, LIFE_LOG 총 4가지로 수집된 건강 데이터를 분류하였다. 각 데이터 유형에 대한 설명은 다음과 같으며, Fig. 4는 각 건강 데이터 유형에 대한 샘플 데이터이다.

- MED_RECORD : 병원과 같은 의료 기관에서 측정한 건강 데이터.
- DIAG_PREDICT : 건강 상태의 모니터링을 기반으로 예측한 질병 진단 예측 데이터.
- TRAINING : 건강 재활 과정에서 수행한 재활용 게임(트레이닝)에 관한 데이터.
- LIFE_LOG : 일상 생활에서 스마트폰의 센서 혹은 웨어러블 디바이스 등을 통해서 수집된 건강 데이터.

3.2 PHDItemReader 설계

건강 데이터는 의료 기관뿐만 아니라 다양한 센서들을 통해서도 수집된다. 이러한 다양한 경로에서 수집된 건강 데이터를 통합하는 과정에서 데이터의 규모가 커지고, 이를 일반적인 방법으로 처리하게 되면 서버의 과부하를 유발할 수 있다. 이를 해결하기 위하여 대량의 데이터를 일괄 처리하는 배치 어플리케


```
{
 "_id" : "5fas35gkbdtot",
 "user_id" : "aseah",
 "dept_id" : "ahospital",
 "type" : "Med_record",
 "date" : "2020.06.28T11:38:58",
 "tags" : ["hospitalA", "HL7", "weight"],
 "data" : {
   "resourceType" : "Observation",
     "code" : {
       "coding" : [{
         "system" : http://loic.org,
         "code" : "29463-7",
         "display" : "Body weight",
     }]
   },
   "effectiveDateTime" : "2020-06-28",
   "valueQuantity" : {
     "value" : 185,
     "unit" : "lbs",
     "system" : http://unitsofmeasure.org,
     "code" : "[lb_av]"
   }
 }
}
```

```
{
 "_id" : "31bj32blkdp39",
 "user_id" : "aseah",
 "dept_id" : "ahospital",
 "type" : "Diag_predict",
 "date" : "2020.06.30T17:25:36",
 "tags" : ["예측", "예방", "당뇨"],
 "data" : {
   "category" : "predict",
   "disease" : diabetes",
   "predictDate" : "2020-06-29",
   "detail" : [{
       "sex" : "male",
       "age" : 53,
       "sbp" : 123,
       "dbp" : 83,
       "fbp" : 352,
       "unit" : "mmHg",
       "bmi" : 26.5
   }],
   "description" : "당뇨위험군으로 지속
          적인 병원 치료, 식단조절 희망"
 }
}
```

```
{
 "_id" : "9pg2gb51jbd",
 "user_id" : "aseah",
 "dept_id" : "trainingCenter",
 "type" : "Training",
 "date" : "2020.07.03T09:51:11",
 "tags" : ["game", "재활", "반응훈련"],
 "data" : {
   "trainingCode" : "9728_R",
   "trainingDate" : "2020-07-03",
   "course" : [
     {
       "game" : "catching a mole",
       "treatment" : "response",
       "score" : 67,
       "unit" : "pt",
       "normalRangeMin" : 70,
       "normalRangeMax" : 90,
       "result" : "under"
     }
   ],
   "prescription" : "물리치료 약 1주일 후
                 재검사"
 }
}
```

```
{
 "_id" : "29lgibjgyq3kg",
 "user_id" : "aseah",
 "dept_id" : "adminfitbit",
 "type" : "Life_log",
 "date" : "2020.05.19T18:09:22",
 "tags" : ["심장박동", "fitbit", "activity"],
 "data" : {
   "activities-heart" : [
     {
       "dateTime" : "2020-05-19",
       "value" : {
         "customHeartRateZones" : [],
         "heartRateZones" : [{
           "caloriesOut" : 740.15264,
           "max" : 94,
           "min" : 30,
           "minutes" : 593,
           "name" : Out of Range"
         }],
         "restingHeartRate" : 68
       }
     }
   ]
}
```


Fig. 4. Health data samples.

이션이 필요하게 되었다. 배치 어플리케이션을 구현하는 방법은 다양하지만 대부분 단순 혹은 단일 배치 작업에는 유용하지만 많은 작업이 필요한 경우에는 구현 과정에서 어려움이 존재했다. 또한 에러 추적이 어려워서, 건강 데이터가 배치 작업 중에서 에러가 발생하여 잘못 처리되거나, 로그가 남아있지 않으면 사람들에게 잘못된 건강 정보를 전달할 수 있고, 이는 건강 관리하는 과정에서 문제를 유발할 수 있다. 따라서 기존의 스프링 모듈을 통하여 안정화된 인프라를 지원받고, 배치 처리 과정에서 세부적인 작업 분할로 로그의 추적이 용이한 스프링 배치 기반으로 배치 어플리케이션을 구현해야한다. Fig. 5는 헬스케어 배치 어플리케이션 중에서 한 달 주기의 건강 데이터를 유형 별로 집계하는 배치 어플리케이션의 설계이다.

Healthcare Job Scheduler에서 정해진 한 달의 주기에 따라 배치 작업을 실행하는 메소드를 호출한다. Job Launcher는 스케줄러에서 배치 작업을 실행하라는 명령을 받아서 MongoHealthcareRepository 안에서 배치 작업에 대한 메타 데이터를 불러온 후 HealthcareJob이라는 배치 작업을 실행한다. 세부 배치 작업을 수행하는 MonthlyStatisticsStep 안에서는 데이터 묶음 기반의 처리를 기반으로 Custom MongoItemReadeer에서 한 달 기간 내의 단일 건강 데이터를 읽고, MonthlyItemProcessor에서는 읽은 단일 건강 데이터에 대하여 데이터의 유형에 따라서 분류 및 필터를 하는 작업을 수행한다. 해당 작업을 배치 세부 작업에서 설정한 데이터 묶음의 크기만큼 수행한 뒤, MonthlyItemWriter에서 데이터 묶음의 크기만큼의 가공된 데이터를 MongoHealthcare Repository로 저장한다. 한 달 기간 내의 전체 건강 데이터에 대하여 해당 작업을 수행할 때까지 위의 작업을 반복 수행한다.

배치 어플리케이션에서 데이터베이스의 건강 데이터를 읽는 부분인 ItemReader는 배치 작업을 수행하는 과정에서 데이터를 데이터베이스에서 낱개로 읽기 때문에 가장 많은 시간이 소요된다. ItemReader의 데이터 질의 속도가 배치 어플리케이션의 수행 시간을 결정짓는 가장 중요한 역할을 한다. 따라서 본 논문에서는 스프링 배치 기반의 통합 개인 건강 데이터 처리 시스템에서 데이터를 읽는 ItemReader의 질의 속도를 단축시키기 위하여 기존에 존재하던 MongoDB 기반의 ItemReader인 MongoItemReader를 개선하였다. Fig. 6은 구현한 ItemReader의 프로세스 다이어그램이다.

Fig. 7은 프로세스 다이어그램을 기반으로 한 Item Reader에서 건강 데이터를 읽는 부분인 doRead 함수에 대한 가상 코드이다. 건강 데이터를 읽기 전에 먼저 MongoDB와 배치 어플리케이션 사이의 연결을 수행한다. 그리고 ItemReader에서 필수로 사용되는

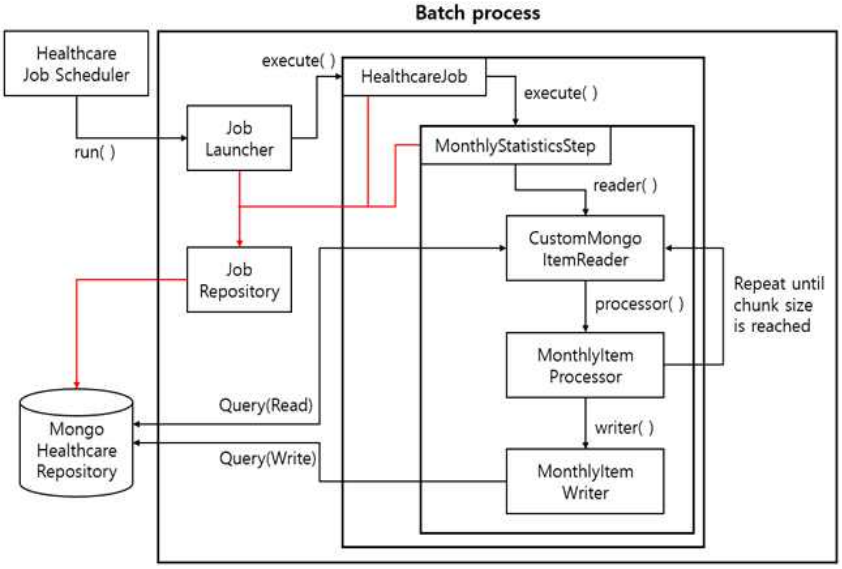

Fig. 5. Design of healthcare batch application.

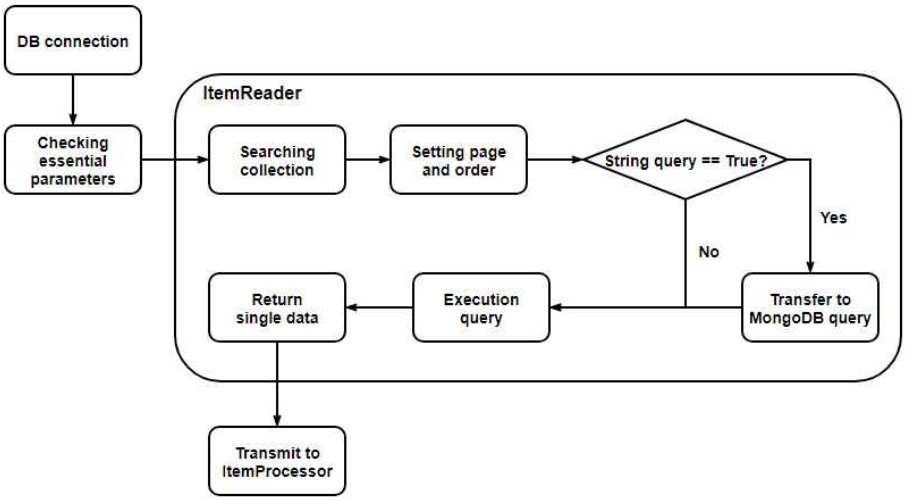

Fig. 6. Process diagram of ItemReader.

파라미터인 쿼리, 테이블(Collection), 페이지 및 정렬 정보에 대하여 검사한다. 모든 파라미터가 정상적으로 포함이 되었으면 ItemReader에서 쿼리를 통해 건강 데이터를 읽는 작업을 수행한다. 외부에서 받은 파라미터를 토대로 테이블을 탐색하고 페이지 및 정렬 정보에 대한 변수 설정을 해준다. 쿼리는 JSON 타입의 쿼리를 작성하는 문자열 형태의 쿼리와 MongoDB에서 스프링에게 제공하는 MongoDB 쿼리 두 가지로 나뉘는데, 최종 수행되는 쿼리는 스프링의 MongoDB 쿼리이므로, 문자열 형태의 JSON 쿼리를 MongoDB 쿼리로 변환해준다. 그 후에 해당 쿼리를 실행하고 데이터 묶음 기반의 데이터 처리의 흐름에 따라 단일 건강 데이터를 반환해주고 데이터를 필터 및 검증하는 ItemProcessor로 전달해준다.

```
function doReadPage
  if String-based query is existing then
      set paging and sorting
      if fields param is existing then
          set mongodb query based on fields param
    else
        set mongodb query
    integrate mongodb query and setting of paging and
    sorting
    transfer String-based query to mongo query(JSON
    formatted)
    if collection param is existing then
        for k = 1 to page size
            return execution result of query
    else
        for k = 1 to page size
            return execution result of query
  else
      set paging
      if collection param is existing then
          for k = 1 to page size
              return execution result of JSON formatted query
      else
          for k = 1 to page size
              return execution result of JSON formatted query
end
```

Fig. 7. Pseudo code of doReadPage.

4. 구현 및 실험

4.1 PHDItemReader 구현

본 논문에서는 서로 다른 데이터 표현 형식의 한계를 극복하기 위하여 MongoDB를 사용하여 비정형의 건강 데이터를 관리할 수 있도록 하였다. 또한 3장의 설계를 바탕으로 대용량의 데이터를 처리하기 위하여 스프링 배치 기반의 배치 어플리케이션을 구현하는 과정에서 건강 데이터를 읽기 위한 ItemReader를 구현하여 건강 데이터의 질의 속도를 높일 수 있도록 하였다. Fig. 8은 본 논문에서 구현한 PHDItemReader에 대한 클래스 다이어그램이다.

기본적으로 InitializingBean 인터페이스 구현을 통하여 PHDItemReader에서 반드시 포함되어야 하는 필수 파라미터 값에 대한 체크를 진행할 수 있도록 하였으며, AbstractPaginationDataItemReader 클래스를 상속 받음으로써 ItemReader의 실질적인 기능인 데이터를 데이터베이스에서 읽는 기능과 PHDItemReader에서 데이터를 읽는 과정에서 상태

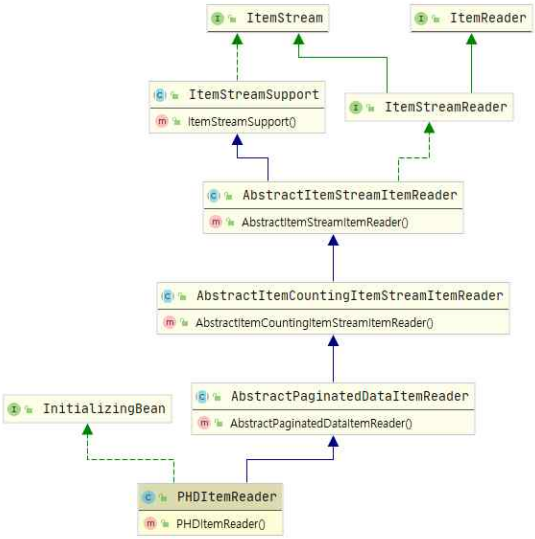

Fig. 8. Class diagram of PHDItemReader.

저장 및 복원을 위한 마커(Maker) 기능을 수행할 수 있도록 하였으며, PHDItemReader에서는 ItemReader 클래스에서 상속받은 doReadPage 메소드를 재정의 하였다.

설계된 클래스 아키텍처에 따라 프로그램을 구현하였으며, Fig. 9는 배치 어플리케이션의 일괄 처리 작업 과정에서 본 논문에서 구현한 ItemReader를 활용하여 6개월 사이의 LIFE_LOG 유형의 건강 데이터를 질의하는 소스코드이다. 먼저 PHDItemReader에서 건강 데이터를 읽기 위하여 해당 데이터의 도메인 클래스인 HealthData 객체를 타겟 클래스로 선언해준다. 그 후 MongoDB에 대한 설정 값, 쿼리 결과를 나타내는 페이지 정보, 결과 값의 정렬 방식 등 필수 파라미터 값들을 설정해준다. 파라미터에 대한 설정이 끝나면, ItemReader에서 어떤 건강 데이터를 찾을 것인지에 대한 쿼리 정보를 전달하는데, Fig. 9에서는 2020년 1월 1일부터 2020년 6월 31일까지 LIFE_LOG 유형의 건강 데이터를 질의한다. 마지막으로 필수 파라미터 값에 대한 유효성 검사를 진행한 뒤, 쿼리 결과로 나온 건강 데이터를 낱개로 Item Processor로 전달해준다.

4.2 질의 비교 분석 실험

4.2.1 실험 환경

RDBMS를 기반으로 하는 ItemReader는 다양한

```
@Bean
@StepScope
protected PHDItemReader<HealthData> phdItemReader()
    throws Exception {

  //Define PHDItemReader binded with HealthData
  phdItemReader<HealthData> itemReader = new
PHDItemReader<HealthData>();

  itemReader.setTemplate(mongoOperation); //Template
  configuration
  itemReader.setTargetType(healthdata.class);  //Set Target
  Class
  itemReader.setPageSize(100);

  //Sorting
  Map<String, Direction> sorts = new HashMap<String,
  Sort.Direction>(1);
  sorts.put("date", Direction.ASC);
  itemReader.setSort(sorts);

  //Query method based on MongoDB query
  itemReader.setQuery("{$and:[{type:{$eq:'LIFE_LOG'}},
    {date:{$gte:2020.01.01.T00:00:00,
    $lte:2020.06.31T23:59:59}}]}");

  //Query to find based on the condition (Experiment 3)
  itemReader.afterPropertiesSet(); //Check the essential
  param

  return itemReader;
}
```

Fig. 9. Source code about ItemReader in batch application.

데이터 표현 형식을 갖는 통합 건강 데이터에 대응하기 위해서는 각각의 데이터 표현 형식에 대한 구조가 스키마에 정의가 되어 있어야 하며, 각각에 대한 쿼리가 추가되어야 한다. 하지만, 이러한 비정형의 데이터를 유연하게 처리할 수 있는 NoSQL과 비교했을 때 극심한 성능 차이를 보인다. 따라서 본 논문에서는 스프링 배치에서 MongoDB를 기반으로 데이터를 질의하기 위하여 기본으로 제공해주는 MongoItem Reader와 본 논문에서 구현한 PHDItemReader 사이의 성능 차이를 비교 하였다.

성능 분석 과정에서 비교 대상 외에 영향을 끼칠 수 있는 변수가 개입되는 것을 방지하기 위하여 동일한 환경에서 실험 환경을 구축하였다. 1대의 Window 10 2004 버전의 운영체제 기반의 컴퓨터에서 실험을 진행했으며, 실험에 사용한 데이터베이스는 비정형 구조의 데이터를 다루는 MongoDB를 사용하였다. 배치 어플리케이션 구현 및 실험은 IntelliJ IDEA 2020.2를 사용하여 진행하였다. 실험 환경에 대한 구

Table 2. System environment of performance

Development tool	IntelliJ IDEA 2020.2
Database	NoSQL
	MongoDB 4.2.3
CPU	RAM 8GB, Intel i5-7200
OS	Window 10 ver.2004

Fig. 10. Result of experiments.

조는 Table 2와 같다.

실제 사용자들이 자신들의 건강 상태를 웹 혹은 모바일을 통해 조회하는 상황을 고려하여 실험을 3가지로 나누어 진행하였다. 가상의 사용자 5명과 가상의 건강 관리 플랫폼 5개를 기반으로 1,000,000 건의 건강 데이터를 데이터베이스에 저장하고, 사용자가 자신의 건강 데이터를 조회할 때 필요한 조건들을 기반으로 쿼리문 3개를 작성하여 각각에 대한 MongoItemReader와 본 논문에서 구현한 PHDItem Reader의 데이터 질의 속도를 비교하였다. Table 3는 실험 조건에 대한 설명이다.

4.2.2 실험 결과

Fig. 10과 Table 4는 4.2.1에서 설계한 실험 환경을 바탕으로 실험을 진행한 결과이다.

Table 4. Result of experiments

	MongoItem Reader	PHDItem Reader	Improvement rate
Experiment 1	196s	168s	14.2%
Experiment 2	271s	232s	14.3%
Experiment 3	481s	398s	20.8%

그래프의 수치가 낮을수록 동일한 질의를 처리할 때 소요되는 시간이 더 짧았다는 것을 의미한다. 그래프에서 좌측 막대 그래프는 기존에 존재하던 Item Reader인 MongoItemReader를 기반으로 질의를 처리한 결과이며, 우측 막대 그래프는 본 논문에서 구현한 PHDItemReader의 질의 처리 결과이다. 전체 실험 결과는 본 논문에서 구현한 PHDItemReader의 질의 처리 속도가 더 우수했다. 실험 1에서 구현한 PHDItemReader는 기존의 MongoItemReader보다 28초 더 빠른 질의 처리 속도를 보여주었으며, 약 14.2%의 처리 속도의 차이를 보였다. 실험 2에서는 구현한 PHDItemReader가 39초 더 빠른 질의 처리 속도를 보였으며, 약 14.3%의 처리 속도 차이를 보였다. 마지막으로 실험 3에서는 본 논문에서 구현한 PHD ItemReader가 83초 더 빠른 질의 처리 속도를 보였으며, 약 20.8%의 처리 속도 차이를 보였다. 실험1은 단순한 질의 형태를 가지고 있어, 처리 속도가 3개의 실험 중에서 가장 빠르게 나왔다. 실험2는 단순한 질의 형태를 가지고 있지만, 1년의 범위 값을 질의하는 과정에서 실험1보다 더 시간이 소요되었다. 하지만, 실험1, 2는 비교적 간단한 질의를 처리하기 때문에 처리 속도의 차이가 크지 않고, MongoItemReader와의 질의 처리 속도 차이도 작았다. 실험 3은 질의가 실험 1, 2에 비하여 더 복잡하게 구성이 되어있어 이를 처리하는 과정에서 시간이 더 소요되는 것으로 나타났으며, MongoItemReader와의 처리 속도 차이

Table 3. Configuration of experiments

Total data	Type of data	People	Healthcare platform	Period	Experiment	
1,000,000	4	5	5	1 year	Simple query	1. Total health data of 'A'
						2. Total health data in 1 year
					Complex query	3. Health data of 'LIFE_LOG' in 6 months

도 실험 1, 2에 비하여 격차가 더 크게 나타났다.

5. 결 론

4차 산업혁명으로 인하여 센서 기술이 발전하게 되었고, 다양한 센서들을 기반으로 대량의 데이터가 수집되고 있다. 의료분야에서도 의료 ICT 기술의 발전으로 의료 기관뿐만 아니라, 일상생활에서도 다양한 헬스케어 플랫폼을 기반으로 건강 데이터가 수집되고 있다. 하지만, 이러한 다양한 플랫폼을 통해서 수집된 개인 건강 정보는 플랫폼별로 관리되어 사람들이 자신들의 건강 정보를 조회할 때 불편함을 겪고 있다. 이를 해결하기 위하여 개인 건강 정보를 통합하여 관리할 수 있는 서비스의 필요성이 증가하게 되었고, 이 과정에서 다양한 데이터 표현 형식을 갖고 있는 대량의 건강 정보를 일괄 처리하는 배치 어플리케이션이 필요하게 되었다. 본 논문에서는 스프링 배치 기반의 건강 관리 배치 어플리케이션에서 건강 데이터를 읽는 작업을 수행하는 ItemReader를 설계 및 구현하였다. 데이터베이스는 다양한 건강 데이터 표현 형식에 대응하기 위하여 비정형 데이터를 저장하고 관리할 수 있는 MongoDB 사용하였으며, JSON 형식의 데이터 스키마를 구축하였다. 논문에서 설계 및 구현한 ItemReader의 성능을 검증하기 위하여 스프링 배치에서 기존에 MongoDB의 데이터를 읽어오기 위하여 사용된 MongoItemReader와 질의 처리 속도를 비교분석하였다. 실험은 가상의 건강 데이터 1,000,000건을 기반으로 3가지의 질의를 각각 수행하였으며, 질의는 단순 질의2개, 복합 질의1개로 구성했다. 실험 결과 본 논문에서 구현한 PHDItem Reader의 건강 데이터 질의 속도가 약 20% 빠른 것으로 나타났으며, 대량의 건강 데이터를 일괄 처리하는 과정에서 유용할 것으로 기대된다.

향후 연구 계획으로는 본 논문에서 구현한 Item Reader를 기반으로 건강 데이터 통합 관리 배치 어플리케이션을 구현하는 것이며, 이를 바탕으로 대량의 건강 데이터를 저장하고 가공하는 과정에서 시스템의 과부하를 줄이고 서버의 성능을 향상시켜 사람들의 건강 관리를 보조할 수 있을 것을 기대한다.

REFERENCE

[1] S.H. Jung, J.C. Kim, and C.B. Sim, "A Novel Data Prediction Model Using Data Weights and Neural Network Based on R for Meaning Analysis between Data," *Journal of Korea Multimedia Society*, Vol. 18, No. 4, pp. 524-532, 2015.

[2] J.G. Son and J.G. Kim, "Squall: A Real-time Big Data Processing Framework Based on TMO Model for Real-time Events and Micro-batch Processing," *Journal of Korean Institute of Information Scientists and Engineerings*, Vol. 44, No. 1, pp. 84-94, 2017.

[3] Node.js Batch NPM(2011), https://www.npmjs.com (accessed June 8, 2020).

[4] D. Cheng, Y.X. Zhou, Y. Wang, and C. Jiang, "Adaptive Scheduling Parallel Jobs with Dynamic Batching in Spark Streaming," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 29, No. 12, pp. 2672-2685, 2018.

[5] D. Jeon, B.M. Lee, and H. Hwang, "Performance Analysis of RDBMS and MongoDB through YCSB in Medical Data Processing System Based HL7 FHIR," *Journal of Korea Multimedia Society*, Vol. 21, No. 8, pp. 934-941, 2018.

[6] M. Yi and H. Hwang, "Design of Secure Personal Health Record Management Systems," *Journal of Korea Institute of Information Technology*, Vol. 13, No. 8, pp. 71-80, 2015.

[7] Spring Batch(2004), https://spring.io/batch (accessed June 12, 2020).

[8] Chunk-oriented Processing(2006), https://docs.spring.io/spring-batch/docs/2.0.x/reference/html/ whatsNew.html (accessed June 12, 2020).

[9] N. Archer, U.F. Thomas, C. Lokker, K.A. McKibbon, and S.E. Straus, "Personal Health Records: A Scoping Review," *Journal of the American Medical Informatics Association*, Vol. 18, No. 4, pp. 515-522, 2011.

[10] T. Chen, C. Liu, T. Chen, C. Chen, J. Bau, T. Lin, et al., "Secure Dynamic Access Control Schema of PHR in Cloud Computing," *Journal of Medical Systems*, Vol. 36, No. 6, pp. 4005-4020, 2012.

[11] R. Brooks, “e-Health Part 1: Current State of Play,” *Australian Nursing Journal*, Vol. 20, No. 2, pp. 20, 2012.

[12] J. Lee, G. Go, H. Kang, and H. Jung, “Web Based Body Change Monitoring System,” *Journal of the Korea Institute of Information and Communication Engineering*, Vol. 20, No. 3, pp. 615-620, 2016.

[13] J.S. Seo, A.N. Kim, S.H. Kim, S.H. Lee, B.R. Nam, M.K. Lee, et al., “Study on Korean Medicine Personal Health Record Platform,” *Journal of Physiology and Pathology in Korea Medicine*, Vol. 30, No. 6, pp. 458-465, 2016.

[14] Fitbit Dashboard(2007), https://www.fitbit.com/kr/ app (accessed October 21).

전 동 철

2019년 가천대학교 컴퓨터공학과 (공학사)

2019년~현재 가천대학교 일반대학원 IT융합공학과 컴퓨터공학 석사과정

관심분야 : Database, u-Health, Mobile, Web

황 희 정

2000년 인하대학교 컴퓨터공학과 (공학석사)

2008년 인천대학교 컴퓨터공학과 (공학박사)

2000년~현재 가천대학교 IT대학 컴퓨터공학과

관심분야 : Software Engineering, u-Health, Big Data, Medical Informatics, Ubiquitous Computing