The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 21, No. 5, pp.127-134, Oct. 31, 2021. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2021.21.5.127
JIIBC 2021-5-17

마이크로 서비스 아키텍처를 지원하는 데이터 프로파일링 소프트웨어의 개발

Development of Data Profiling Software Supporting a Microservice Architecture

장재영*, 김지훈**, 지서우**

Jae-Young Chang*, Jihoon Kim**, Seowoo Jee**

요 약 최근 빅데이터 산업의 확대로 고품질의 데이터를 확보하는 것이 중요한 이슈로 떠오르고 있다. 고품질의 데이터를 확보하기 위해서는 데이터에 품질에 대한 정확한 평가가 선행되어야 한다. 데이터의 품질은 데이터에 대한 통계와 같은 메타정보를 통해 평가할 수 있는데 이러한 메타정보를 자동으로 추출하는 기능을 데이터 프로파일링이라고 하다. 지금까지 데이터 프로파일링 소프트웨어는 기존의 데이터 품질 또는 시각화 관련 소프트웨어의 부품이나 추가적인 서비스로 제공되는 것이 일반적이었다. 따라서 프로파일링이 요구되는 다양한 환경에서 직접적으로 사용하기에는 적합하지 않았다. 본 논문에서는 이를 해결하기 위해 마이크로 서비스 아키텍처를 적용하여 다양한 환경에서 서비스가 가능한 데이터 프로파일링 소프트웨어의 개발 결과를 제시한다. 개발된 데이터 프로파일러는 restful API를 통해 데이터의 메타정보에 대한 요청과 응답을 제공하여 사용하기 쉬운 서비스를 제공한다. 또한, 특정 환경에 종속되지 않고 다양한 빅데이터 플랫폼이나 데이터 분석 도구들과 원활한 연계가 가능하다는 장점이 있다.

Abstract Recently, acquisition of high quality data has become an important issue as the expansion of the big data industry. In order to acquiring high quality data, accurate evaluation of data quality should be preceded first. The quality of data can be evaluated through meta-information such as statistics on data, and the task to extract such meta-information is called data profiling. Until now, data profiling software has typically been provided as a component or an additional service of traditional data quality or visualization tools. Hence, it was not suitable for utilizing directly in various environments. To address this problem, this paper presents the development result of data profiling software based on a microservice architecture that can be serviced in various environments. The presented data profiler provides an easy-to-use interface that requests of meta-information can be serviced through the restful API. Also, a proposed data profiler is independent of a specific environment, thus can be integrated efficiently with the various big data platforms or data analysis tools.

Key Words : Data Profiling, Data Lake, Big Data, MicroService Architecture, Restful API, Data Quality, Metadata

*정회원, 한성대학교 컴퓨터공학부
**학생회원, 한성대학교 컴퓨터공학부
접수일자 2021년 9월 8일, 수정완료 2021년 10월 3일
게재확정일자 2021년 10월 8일

Received: 8 September, 2021 / Revised: 3 October, 2021 / Accepted: 8 October, 2021
*Corresponding Author: jychang@hansung.ac.kr
Dept. of Computer Engineering, Hansung University, Korea

Ⅰ. 서 론

데이터 산업은 4차 산업혁명 시대의 핵심으로 인공지능, 사물 인터넷, 로봇기술, 가상현실 등 최신 IT 기술에 중요한 기반 정보를 제공한다. 이들은 과거로부터 축적되거나 실시간으로 생성되는 대용량의 데이터로부터 수집, 가공, 분석, 추론 등의 기술을 적용하여 각 분야에서 필요한 고급 정보를 생성하여 활용한다. 이러한 업무들을 지원하기 위해서는 하둡(hadoop)과 같은 대용량의 데이터 저장 및 처리 기술이 지원되어야 하며, 고품질의 데이터를 확보하기 위한 노력도 병행되어야 한다. 특히 대용량 데이터 처리 문제는 현재 비약적인 발전을 이루어져 다양한 솔루션들이 활용되고 있지만, 데이터의 품질 문제는 단순 솔루션만으로는 해결할 수 없는 다양한 정성적인 이슈들이 존재한다[1][2]. 그럼에도 불구하고 데이터 품질은 반드시 해결해야 하는 문제로 아무리 최신의 고급분석 기술을 적용한다고 하더라도 저품질의 데이터로는 올바른 분석결과를 기대할 수 없다.

고품질의 데이터를 확보하기 위해서는 데이터 생산 및 수집단계부터 다양한 노력이 병행되어야 하는데, 무엇보다도 데이터에 품질에 대한 정확한 평가가 선행되어야 한다. 데이터의 품질은 데이터의 구조나 저장된 값들의 통계 정보를 통해 평가할 수 있는데 이러한 정보들을 메타 데이터(metadata)라고도 한다. 하지만 메타 데이터라고 하면 데이터 자체의 구조나 값에 대한 정보뿐만 아니라 데이터의 생성날짜, 소유자 등과 같은 외적인 정보를 의미하기도 한다. 따라서 이러한 데이터 자체의 구조나 값들에 대한 메타 데이터 정보를 특별히 데이터 프로파일(data profile)이라고 하며, 데이터 프로파일 정보를 자동으로 추출하거나 계산하는 소프트웨어나 업무를 데이터 프로파일링(data profiling)이라고 한다[3].

현재까지의 데이터 프로파일링은 대부분 관계형 데이터베이스와 같은 테이블 형태의 정형 데이터(formatted data)에 대해서 적용된다. 대표적인 프로파일 항목으로는 컬럼 수, 레코드 수, 컬럼명 리스트와 같은 테이블에 대한 기본 정보부터 각 컬럼의 값들에 대한 통계 정보(결측치 수, 최대/최소, 평균, 중앙값 등)가 있다. 프로파일 항목에 관해서는 정해진 규정은 없지만 저장된 데이터 전반적인 상황을 파악할 수 있는 값들의 분포나 통계 정보를 포괄한다. 또한 데이터 품질의 평가하는데 필요한 부가적인 정보도 프로파일링의 대상 정보가 될 수 있다.

앞서 언급한 바와 같이 데이터 프로파일 정보는 데이터 품질의 평가하는 기본정보 활용된다. 따라서 현재까지 대부분의 데이터 프로파일링 정보를 제공하는 소프트웨어는 프로파일링만을 단독으로 수행하는 독립적인 형태는 거의 없고 품질평가/개선 소프트웨어나 데이터 시각화 소프트웨어 등에 일부 기능으로 제공되는 경우가 대부분이다[5]. 또한, 일부 빅데이터 플랫폼이나 데이터 레이크(data lake)[6] 환경에서 데이터 프로파일 정보를 독자적으로 제공하기도 한다. 따라서 데이터 프로파일링이 요구되는 다양한 환경에서 직접적으로 사용하기에 적합한 형태로 제공되는 데이터 프로파일링 소프트웨어는 거의 없는 실정이다.

본 논문에서는 이러한 한계를 극복하기 위해 특정 환경에 구애받지 않은 서비스가 가능한 데이터 프로파일링 시스템의 개발 결과를 제시한다. 제안된 시스템은 외부 환경과의 원활한 연동을 지원하기 위해 독립적인 서비스가 가능하도록 구성하였다. 이를 위해 마이크로 서비스 아키텍처(microservice architecture)[7][8]를 지원하도록 설계하였으며, 특히 restful API로 프로파일링 서비스의 요청과 응답이 가능하도록 개발하였다. 따라서 특정 프로그래밍 언어나 플랫폼에 독립적인 서비스가 가능하여, 기존의 시스템과의 통합이나 연계를 쉽게 실현할 수 있다. 기존의 대부분의 데이터 프로파일링은 관계형 데이터베이스의 테이블을 대상으로 서비스를 제공하여 일부 CSV와 같은 파일 형태의 데이터에 대해서는 제한된 서비스를 제공하였다. 그러나 최근 데이터 레이크(data lake)를 비롯한 대부분의 빅데이터 플랫폼에서는 CSV와 같은 파일 형태의 원본 데이터를 제공하고 있다. 따라서 제안된 시스템에서는 관계형 데이터베이스의 테이블뿐만 아니라 CSV와 같은 구분자로 표현된 파일(delimited file, sequential file with column delimiters)에 대해서도 원활한 프로파일링을 지원한다. 따라서 원천 데이터가 어떠한 위치 또는 어떠한 형태로 구성되어 있어도 프로파일링 서비스가 가능하도록 개발하였다.

본 논문의 구성은 다음과 같다. 2장에서는 데이터 프로파일링의 기본 기능과 기존의 프로파일링 도구들을 소개한다. 3장에서는 마이크로 서비스 아키텍처의 장점과 사례들을 제시한다. 4장에서는 본 논문에서 제시하는 데이터 프로파일링 도구 개발 결과를 제시한다. 마지막으로 5장에서는 결론을 맺는다.

Ⅱ. 데이터 프로파일링 기술 현황

데이터 프로파일링은 데이터로부터 통계, 패턴 등의

정보를 수집 또는 계산하여 데이터 현황, 데이터 품질개선을 위한 정보를 획득하는 과정을 말한다. 이렇게 확보된 정보는 데이터의 형태에 대한 직관적인 정보를 제공할 수 있을 뿐만 아니라, 데이터 오류 탐지, 정제, 가공, 연계를 지원하는 기초정보로 활용된다. 데이터 프로파일링에서 제공하는 정보의 종류는 정의에 따라 다양한 분류체계를 갖는다. 예를 들어 [9]와 [10]에서는 데이터 프로파일을 구조정보(structural information), 내용정보(content information), 관계정보(relationship information) 등으로 세분화하였는데, 구조정보는 저장된 데이터들의 포맷 등에 대한 정보와 각 컬럼에 대한 합, 평균, 최대/최소 등의 정보를 의미하며, 내용정보는 데이터 개별요소에 대한 오류, 널값 또는 부정확하거나 모호한 값 등의 정보를 나타낸다. 마지막으로 관계정보는 데이터 간의 관련 정보나 참조 관계에 대한 정보를 의미한다. [3]에서는 프로파일링 업무를 대상 데이터의 형태에 따라 단일 컬럼(single column) 정보, 다중 컬럼(multiple column) 정보, 의존성(dependencies) 정보 등으로 분류하였다. 단일 컬럼 정보는 특정 컬럼의 레코드 수나 값들의 분포, 결측치 비율 등에 대한 정보를 나타내며, 다중 컬럼 정보는 컬럼 간의 연관 관계(correlation 또는 association rules)나 클러스터(clusters)와 같이 2개 이상의 컬럼을 결합하여 생성되는 프로파일 정보를 의미한다. 마지막으로 의존성 정보는 후보 키(candidate key)나 외래 키(foreign key) 탐색, 함수적 종속성(functional dependencies)이나 컬럼 간의 포함 관계(inclusion dependencies) 등에 대한 정보를 나타낸다. 이처럼 프로파일 정보는 다양한 형태로 나누어 분류할 수 있지만, 세부항목들에 대한 정의와 내용은 큰 차이를 보이지 않는다는 것을 알 수 있다.

데이터 프로파일링은 데이터의 규모가 커짐에 따라 데이터의 형태나 분포를 빠른 시간 내에 파악할 수 있는 유용한 수단을 제공해준다. 이를 통해 사용자가 데이터를 직접 획득하지 않고도 데이터의 현황을 파악할 수 있도록 지원하여 사용자의 편의성을 증대시킬 수 있을 뿐만 아니라, 데이터 품질에 대한 평가나 개선 과정에 고급 정보를 제공해준다. 특히 빅데이터 분석과정 중 데이터 품질개선을 위한 가공에 최대 80%의 시간과 비용을 소모하고 있는 사실에 비추어볼 때, 데이터 프로파일링 빅데이터 분야에서 차지하는 중요도는 매우 크다고 볼 수 있다[11].

이러한 상황에도 불구하고 현재까지 데이터 프로파일링은 상용이나 오픈소스 소프트웨어를 막론하고 ETL 도구, 데이터 품질 평가/개선 도구 또는 시각화 도구들에서의 일부 기능으로만 제공되는 경우가 대부분이다. 또는 캐글(Kaggle; https://www.kaggle.com/)이나 마이크로소프트 애저(Microsoft Azure)[12]와 같은 빅데이터 플랫폼 또는 저장소 일부에서 자체적으로 제공되고 있다. 따라서 데이터 프로파일 정보가 필요한 상황에서 이러한 소프트웨어들을 활용하는 것 외에는 방법이 없으며, 데이터 프로파일링만을 위한 전문적인 서비스는 거의 없는 실정이다. 그 이유는 데이터 프로파일링이 아직 전문적인 데이터에 대한 분석 서비스의 일종으로 인식되지 않고 단순히 데이터 품질이나 시각화를 위한 보조 수단으로만 인식되기 때문이다. 하지만 최근 들어 빅데이터 플랫폼이나 데이터 레이크 구축이 활발히 높아짐에 따라 저장된 데이터들의 정보를 포함하는 메타 데이터에 관심도 높아지고 있다. 따라서 데이터 자체에 대한 프로파링에 대한 요구도 점차 증가하는 추세이다. 따라서 기존의 다른 목적을 개발된 소프트웨어의 일부 기능으로 제공되는 데이터 프로파일링 서비스로는 이러한 요구사항을 만족시킬 수 없으며 특정 환경에 종속되지 않고 쉬운 접근성을 보장하는 데이터 프로파일링 서비스에 대한 수요가 점차 증가할 것으로 예상된다.

Ⅲ. 마이크로 서비스 아키텍처

기존의 웹 애플리케이션들의 개발과정에서는 각 모듈이나 컴포넌트들을 각자 개발한 후 하나의 패키지로 통합하여 배포(deploy)하고, 마지막으로 이를 하나의 애플리케이션 프로세스로 구동하는 방식을 사용하였다. 그림 1(a)와 같은 이러한 방식을 모놀리식 아키텍처(monoithic architecture)라고 하는데 이러한 방식은 빠른 시간내에 개발과 배포가 용이하지만, 시간이 지남에 따라 각 컴포넌트 간의 의존성이 강해 확장 및 유지보수가 어렵다는 단점을 갖고 있다[4][7][8]. 이러한 문제를 극복하기 위해 최근 많은 관심을 받는 구조가 그림 1 (b)와 같은 마이크로 서비스 아키텍처이다. 마이크로 서비스 아키텍처란 소프트웨어 구성 시 필요한 핵심 기능을 독립적인 서비스들로 세분화하여 개발하는 서비스 지향 관점의 개발 구조(SOA: Service-Oriented Architecture)이다[4][13].

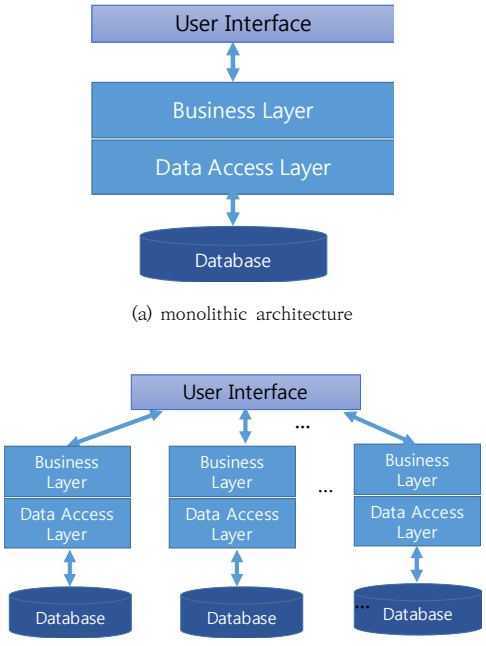

그림 1. 모놀리식 아키텍처와 마이크로 서비스 아키텍처[4]
Fig. 1. monolithic vs. microservices architecture

마이크로 서비스 아키텍처는 모놀리식 아키텍처에 비해 많은 장점을 갖고 있다[4]. 우선 기존의 각 서비스에 대해 새로운 기능을 추가하거나 수정 및 유지보수가 용이하다. 또한 특정 서비스의 트래픽 처리량이나 데이터 보관량을 유동적으로 조절할 수 있는 규모 확장성도 보장한다. 배포 과정에서도 큰 차이가 있는데 마이크로 서비스 아키텍처는 각 서비스의 독립적인 개발 이후 서비스 업데이트 시 전체 소프트웨어의 중단이 없이 지속적인 배포가 가능하다. 장애 대응에 있어서도 단일 서비스의 오류가 나머지 서비스에 대해 독립적이기 때문에 한 서비스에 장애가 있더라도 전체 소프트웨어에는 큰 영향을 끼치지 않는다. 또한, 장애가 있는 특정 서비스에 대해서만 복구과정이 필요하므로 오류 파악 및 유지보수에 유리하여 전체 소프트웨어의 재기동시간이 단축된다. 마지막으로 마이크로 서비스 아키텍처에서는 개발자들이 각 서비스에 최적화된 애플리케이션 개발 툴 선택에 대한 자유도가 높다. 다시 말해, 각 개발자들은 다양한 언어를 사용해 서비스를 개발할 수 있다. 이처럼 마이크로 서비스 아키텍처는 기존의 모놀로식 아키텍처에 비해서 많은 장점을 갖고 있으며, 최근 기존의 아키텍처에서 마이크로 서비스 아키텍처로의 전환에 관한 방법론 연구가 활발히 진행되고 있다[8].

현재 많은 온라인 서비스 업체들이 마이크로 서비스 아키텍처를 도입하였는데, 대표적으로 네플릭스(Netflix)는 시스템의 수평적 확장과 높은 신뢰성을 목적으로 7개년에 걸쳐 데이터베이스를 포함한 모든 인프라를 클라우드로 이전했는데, 이 과정에서 마이크로 서비스 아키텍처 구성을 위한 자체 라이브러리들을 개발해 적용했고, 2012년부터 해당 오픈소스들을 Netflix OSS(Open Source Software)라는 이름으로 공개했다[14]. 아마존(Amazon)은 2000년대 초반 서비스의 지속적 개발과 배포, 개발 주기 단축, 새로운 기술 도입 등의 목적으로 마이크로 서비스 아키텍처로 전환을 선택했다. 이후에도 AWS S3를 포함한 여러 제품들을 개발했고, 현재는 기존의 아키텍처에 서버리스(serverless) 개념을 도입한 AWS 람다(Lambda)를 출시했다[15]. 이외에도 많은 국내외 IT 서비스 업체들이 마이크로 서비스 아키텍처를 적용하는 추세이다. 본 논문에서도 이와 같은 추세에 맞춰 데이터 프로파일링 서비스를 마이크로 아키텍처를 적용하는 애플리케이션에 통합이 용이하도록 개발한 결과를 제시한다.

Ⅳ. 데이터 프로파일러 구조 및 기능

1. 데이터 프로파일러 구조

마이크로 서비스 아키텍처를 지원하는 데이터 프로파일러의 전체적인 구조는 그림 2와 같다. 이 그림에서 오른쪽의 응용 서비스는 데이터 프로파일링 기능이 요구되는 애플리케이션들이다. 데이터 품질관리 도구나 ETL 도구 등이 그 예이며, 기타 데이터 제공 서비스를 담당하는 데이터 포털 등이 잠재적인 사용자가 될 수 있다. 이러한 애플리케이션들을 구축할 때 마이크로 서비스 아키텍처를 채용할 경우 API 게이트웨이(gateway)를 통해 서비스를 제공받을 수 있다. 뿐만 아니라 마이크로 서비스 아키텍처를 채용하지 않더라도 데이터 프로파일링 기능이 필요한 어떠한 환경에서도 쉽게 접근 가능한 API를 이용하여 서비스를 받을 수 있다.

데이터 프로파일링 서버는 앞단에서 작업 큐(job queue)와 작업 컨트롤러(job controller)를 활용하여 API를 통해 요청되는 프로파일링 서비스의 부하와 작업 순서를 제어하게 된다. 프로파일링 대상이 되는 원천 데이터가 대용량이거나 프로파일링 작업시간이 많이 소요되는 요청에 대해서는 백그라운드로 서비스를 할 수 있으며, 그 이외에 데이터의 크기가 작거나 단순한 프로파

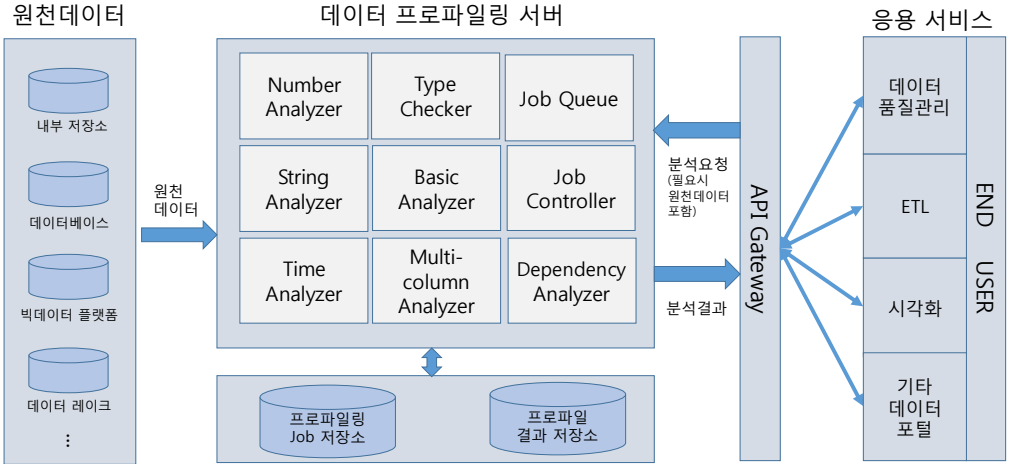

그림 2. 데이터 프로파일러 아키텍처
Fig. 2. Data Profilier Archirtecture

일링 요청일 경우에는 즉시 처리할 수 있도록 제어한다. 타입 검사기(type checker)는 원천 데이터의 각 컬럼에 대한 타입을 자동으로 결정하는 역할을 한다. CSV와 같은 텍스트 형식의 원천 데이터는 데이터베이스 테이블과 같은 데이터 사전(data dictionary)이 미리 정의되어 있지 않으므로 프로파일링을 위해서는 각 컬럼의 타입을 자동을 결정해야 하는 일이 자주 발생한다. 본격적인 프로파일링 서비스는 그 이외의 모듈에서 수행하게 되는데 기본 분석기(basic analyzer)는 원천 데이터의 기본적인 정보, 예를 들어 레코드나 컬럼의 수, 크기, 각 컬럼의 값 분포 등의 요약 정보를 생성하게 된다. 수치 분석기(number analyzer), 문자열 분석기(string analyzer), 시간 분석기(time analyzer)는 단일 컬럼에 대한 프로파일을 생성하는 모듈들로 각각 수치형, 문자형, 시간형 타입의 컬럼에 대한 프로파일 정보를 생성한다. 다중 컬럼 분석기(multi-column analyzer)와 의존성 분석기(dependency analyzer)는 2장에서 정의한 대로 각각 다중 컬럼에 연관된 분석과 컬럼 간의 의존 또는 포함 관계에 대한 분석정보를 생성한다[3]. 그 이외에 요청된 작업 정보와 프로파일링을 정보를 저장하는 저장소가 있다. 이들은 작업 히스토리와 분석결과를 관리하여 재요청에 대응하는 역할을 한다.

마지막으로 그림 2의 왼편에 있는 원천 데이터는 프로파일링 대상이 되는 원천 데이터가 저장된 저장소 정보를 나타낸다. 원천 데이터는 가장 기본적으로 프로파일링 서버의 내부 저장소에 있는 데이터일 수도 있고, 내/외부의 데이터베이스 테이블일 수도 있다. 또한, 데이터를 전문적으로 제공하는 빅데이터 플랫폼이나 데이터 레이크 환경과 같은 원격에 저장된 데이터에 대해서도 접근할 수 있도록 지원한다. 데이터의 접근이 곤란한 환경에도 대응하기 위해 API 요청시에 원천 데이터를 직접 업로드할 수 있는 환경도 지원된다.

2. API 설계 결과

본 논문에서 제시하는 데이터 프로파일링은 스프링부트(Spring Boot) 환경에서 개발되었으며, restful API 형태로 제공된다. 프로파일링 요청은 원천 데이터에 대한 접근 정보뿐만 아니라 상세한 분석명세에 대한 정보를 포함한다. 따라서 GET 방식을 사용할 경우 요청 URL의 길이와 형식이 복잡해질 수 있어 POST 방식으로 요청하고 요청 양식은 JSON을 사용하였다. 또한, 응답으로 제공되는 프로파일링 결과도 역시 JSON 형식으로 제공된다.

우선 프로파일링 요청 정보로 전달되는 요청 API의 JSON 형식은 표 1과 같다. 이 표에서 dataset은 프로파일링 대상이 되는 데이터에 대한 정보로 데이터 타입과 접근 방법에 대한 정보를 제공한다. 접근 방법은 크게 4가지로 구분되는데 로컬 파일 경로(path), URL로 접근 가능한 원격 파일, 데이터베이스의 테이블, 마지막으로 직접 데이터를 업로드하는 방식 중 하나를 지정할 수 있다. 헤더는 CSV와 같은 파일의 경우 첫째 행에 컬럼 이름의 존재 여부를 나타낸다. profiles는 각 컬럼에 대해서 어떠한 프로파일링을 수행할 것인가를 나타낸다. basic은 기본 프로파일링을 수행할 컬럼들의 리스트이

며, number, string, time은 각각 수치, 문자열, 시간 분석의 대상이 되는 컬럼들의 이름이나 컬럼 번호에 대한 리스트이다. 다중 컬럼 및 의존 정보에 대한 명세는 생략하였다.

표 1. 요청 API의 JSON 스키마
Table 1. JSON Schema of Request API

항목명	설명
dataset	프로파일링 대상 데이터셋
source	데이터셋의 경로 (path, url, db, upload)
type	데이터 타입
path	로컬 경로
url	URL 경로
db	데이터베이스 경로
database	DBMS 이름
conn	데이터베이스 connection string
table	테이블 이름
header	헤더(컬럼명 리스트) 존재유무
profiles	프로파일링 대상 컬럼 리스트
basic	기본 프로파일링 대상 컬럼 리스트
number	수치 프로파일링 대상 컬럼 리스트
string	문자열 프로파일링 대상 컬럼 리스트
time	시간/날짜 프로파일링 대상 컬럼 리스트

그림 3은 요청 API로 전달되는 JSON 양식의 예를 보여준다. 이 예에서는 데이터가 원격 파일에 있는 경우이며 이때는 URL로 데이터를 접근하게 된다. 또한 프로파일링 항목 중에 기본, 수치, 문자열 분석은 각각에 명시된 컬럼 리스트에 대해서 수행하도록 지정하였다. profiles 항목이 생략되면 모든 항목에 대해 프로파일링을 수행하고 컬럼 타입도 자동으로 추정하여 해당 정보를 생성하도록 개발하였다.

표 2는 응답으로 전달되는 프로파일링 결과에 대한 JSON 형식이다. 실제 제공되는 프로파일링 항목은 매우 많아 본 논문에서 모두 표현할 수 없어 중요한 일부 항목만을 제시하였다. 또한, 다중 컬럼 정보와 의존 정보에 대한 명세도 생략하였다. 응답 API는 우선 데이터에 대한 이름, 타입, 크기, 컬럼 수와 같은 기본적인 메타 정보를 포함한다. 다음으로 results 항목은 각 컬럼에 대한 프로파일 정보를 배열 형태로 저장한다. 각 컬럼에 대한 프로파일은 표 1에서 설명한 바와 같은 기본 프로파일(basic)외에, 타입별로 number, string, date 프로파일에 대한 분석 결과가 저장된다.

```
{
  "dataset": {
               "source": "url",
               "type": "csv",
               "url": "http://qufa.com/sample.csv"
             },
  "header": true,
  "profiles": {
                 "basic": ["name", "age"],
                 "number": ["age"],
                 "string": ["name"]
               }
}
```

그림 3. 요청 API의 예
Fig. 3. An Example of Request API

그림 4는 그림 3에서 요청한 API에 대해서 분석결과로 전달된 프로파일 예를 보여준다. 이 예는 sample.csv라는 이름의 2개의 컬럼을 갖는 CSV 파일에 대한 분석 결과이다. 첫째 컬럼에 대해서는 기본 및 문자열 분석의 결과를 나타내며, 둘째 컬럼은 기본과 수치 분석에 대한

표 2. 응답 API의 JSON 스키마
Table 2. JSON Schema of Response API

항목명	설명
dataset_name	데이터셋 이름
dataset_type	데이터셋 타입
dataset_size	데이터셋 크기
dataset_column_cnt	데이터셋 열 수
dataset_row_cnt	데이터셋 행 수
results	profiling 결과
column_name	컬럼 이름
column_type	컬럼 타입
profiles	프로파일 결과 목록
basic_profile	basic 프로파일
null_cnt	결측치 수
value_distributuion	값들의 분포정보
...	...
number_profile	number 프로파일
min	최소값
max	최대값
...	...
string_profile	string 프로파일
blank_cnt	빈칸의 수
min_len	최소 길이
...	...
date_profile	date 프로파일
highest_date	가장 최신 시간
lowest_date	가장 과거 시간
...	...

```
{
  "dataset_name": "sample.csv
  "dataset_type": "csv",
  "dataset_size": 300,
  "dataset_column_cnt": 2,
  "dataset_row_cnt": 20,
  "results": [
   {   "column_name": "name",
       "column_type": "string",
       "profiles": {
              "basic_profile": {
                "row_cnt": 20,
                "distinct_cnt": 6,
                "null_cnt": 2,
                "value_distribution": {
                  "Jason": 3,
                  "Paul": 4},
              ...},
              "string_profile": {
                "blank_cnt": 0,
                "min_len": 3
                ... }}},
    { "column_name": "age
      "column_type": "number",
      "profiles": {
              "basic_profile": {
                "row_cnt": 20,
                "value_distribution": {
                  "50": 3,
                  "51": 4 },
                .....},
              "number_profile": {
                "min": 25
                "max": 75
                ... }}}]
}
```


그림 4. 응답 API의 예
Fig. 4. An Example of Response API

분석 결과를 표현한다.

지금까지 본 논문에서 제시하는 프로파일링 요청과 응답 API를 소개하였다. 이 예들에서 보는 바와 같이 JSON 형태로 필요한 프로파일링 명세를 작성하여 restful API로 POST 방식으로 요청하면 프로파일링 서버는 요청된 프로파일 정보를 분석 및 생성하여 JSON 형태로 전달하게 된다. 이와 같이 개발된 프로파일링 서버는 특정 환경에 독립적이고 다른 모듈들과의 관련성도 적어 타 시스템과 통합이 매우 단순해진다. 따라서 마이크로 서비스 아키텍처를 지향하는 통합 환경에 쉽게 연계될 수 있는 장점이 있다.

V. 결 론

본 논문에서는 마이크로 서비스 아키텍처를 지원하는 데이터 프로파일러의 개발 결과를 제시하였다. 마이크로 서비스 아키텍처는 독립적으로 수행되는 컴포넌트들이 결합하여 하나의 큰 서비스가 이루어지도록 설계하는 형태로 최근 많은 관심을 받는 소프트웨어 아키텍처 설계 방안 중의 하나이다. 마이크로 서비스 아키텍처는 기존의 모놀로식 아키텍처와 비교하여 전체 구조가 직관적이고 안정적인 실행을 보장하며, 유지보수도 간편해지는 등 많은 장점을 지닌다. 이러한 아키텍처를 지원하기 위해 본 논문에서 제안한 시스템은 restful API를 이용하여 프로파일링 서비스에 대한 손쉬운 요청과 응답이 가능하며, 외부 컴포넌트들과의 연계 없이 독립적인 수행이 가능하도록 개발하였다. 또한, 원본 데이터셋의 다양한 접근 방법을 지원하여 특정 환경에 의존적이지 않고 통합이 가능하도록 지원하고 있다. 따라서 제안된 시스템은 ETL, 데이터 품질관리/분석 도구, 데이터 포털 등 데이터 처리/분석/저장 등이 요구되는 다양한 응용 분야에 활용될 수 있다. 향후에는 실제 데이터 품질도구이나 데이터 포털과 같은 환경에 개발된 데이터 프로파일러를 연계하여 서비스할 예정이며, 실무적으로 적용했을 때의 성능이나 기능들을 재평가한 후에 개선에 반영할 계획이다.

References

[1] L. Cai and Y. Zhu, "The Challenges of Data Quality and Data Quality Assessment in the Big Data Era", Data science journal, Vol. 14, 2015. DOI: http://doi.org/10.5334/dsj-2015-002

[2] F. Sidi, Fatimah, et al. "Data quality: A Survey of Data Quality Dimensions", International Conference on Information Retrieval & Knowledge Management. IEEE, 2012. DOI: 10.1109/ InfRKM.2012.6204995

[3] Z. Abedjan, L. Golab, and F. Naumann. "Data Profiling: A Tutorial" Proceedings of the 2017 ACM International Conference on Management of Data. 2017. DOI: https://doi.org/10.1145/3035918.3054772

[4] M. Kalske, N. Mäkitalo, and T. Mikkonen. "Challenges When Moving from Monolith to Microservice Architecture." International Conference on Web Engineering. Springer, Cham, 2017. DOI: 10.1007/978-3-319-74433-9_3

[5] Panoply, Modern Data Management, Panoply Ltd., https://panoply.io/

[6] M. Miloslavskaya, and A. Tolstoy. "Big data, Ffast Data and Data Lake Concepts", Procedia Computer Science 88, 2016. DOI: https://doi.org/10.1016/j.procs.2016.07.439

[7] W. Hasselbring, and G. Steinacker. "Microservice

Aarchitectures for Scalability, Agility and Reliability in E-commerce" IEEE International Conference on Software Architecture Workshops (ICSAW). IEEE, 2017. DOI: 10.1109/ICSAW.2017.11

[8] P. Di Francesco, P. Lago, and I. Malavolta. "Migrating Towards Microservice Architectures: an Industrial Survey" IEEE International Conference on Software Architecture (ICSA). IEEE, 2018. DOI: 10.1109/ICSA.2018.00012

[9] H.-Y. Hsueh, C.-N. Chen, and K.-F. Huang, "Generating Metadata from Web Documents: a Systematic Approach", Human-centric Computing and Information Sciences, Vol. 3, No. 1, 2013, DOI: https://doi.org/10.1186/2192-1962-3-7

[10] R. Kimball, et al. The Data Warehouse Lifecycle Toolkit (Second ed.). Wiley, 2008,

[11] G. Fletcher,, P. Groth, and J. Sequeda. "Knowledge Scientists: Unlocking the Data-Driven Organization" arXiv preprint arXiv:2004.07917, 2020.

[12] Copeland, Marshall, et al. "Microsoft Azure" New York, NY, USA:: Apress, 2015.

[13] T. Erl, Service-oriented architecture. Pearson Education Incorporated, 1900.

[14] A. Cockcroft. "NetflixOSS-A Cloud Native Architecture" LASER Sessions 2&3—Overview, 10th Laser Summer School on Software Engineering, XP055276156, 2013.

[15] P. Sbarski, and S. Kroonenburg, Serverless Architectures on Aws: with Examples Using Aws Lambda, Simon and Schuster, 2017.

[16] Y. Han, "Two-dimensional Automatic Transformation Template Matching for Image Recognition", Journal of the Korea Academia- Industrial cooperation Society, Vol. 20, No. 9, 2019, DOI: https://doi.org/10.5762/KAIS.2019.20.9.1

[17] G. Jo, H.-W. Cho and Y. J. Son, "Implementation of 360-degree Image Recognition System Using Lidar Sensors and Yolo v3 Libraries in Cloud Server Environment", The Journal of Korean Institute of Information Technology, Vol. 18, No. 2, 2020, DOI: https://doi.org/10.14801/jkiit.2020.18.2.1

저 자 소 개

장 재 영(정회원)

- 1992년 : 서울대학교 계산통계학과 (이학사)
- 1994년 : 서울대학교 계산통계학과 (이학석사)
- 1999년 : 서울대학교 계산통계학과 (이학박사)
- 2000년 ~ 현재 : 한성대학교 컴퓨터공학부 교수
- 관심분야 : 데이터베이스, 데이터마이닝

김 지 훈(학생회원)

- 2015년 ~ : 한성대학교 컴퓨터공학부
- 관심분야 : 빅데이터분석, 데이터베이스

지 서 우(학생회원)

- 2015년 ~: 한성대학교 컴퓨터공학부
- 관심분야 : 빅데이터분석, 데이터베이스

※ 본 연구는 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(No. 2020-0-00121, 데이터 품질 평가기반 데이터 고도화 및 데이터셋 보정 기술 개발)을 받아 수행된 연구임