

Metadata Management and Harvesting System in Smart Open Data As a Service

Minh Chau Nguyen*, Heesun Won*, Siwoon Son*, Yerin Woo**, Hyeonjae Jeong**

* CybreBrain Research Section, Electronics and Telecommunication Research Institute, Daejeon, South Korea

**University of Science and Technology, Daejeon, South Korea

chau@etri.re.kr, hswon@etri.re.kr, siwoon.son@etri.re.kr, yrwoo@etri.re.kr, presentj94@etri.re.kr

Abstract— These days, with the available data platforms, the enormous growing information including data and metadata in big data and artificial intelligence fields can be stored, organized, and processed to solve different issues in different domains. However, there are still many limitations in the current platforms. Among these drawbacks, data and metadata exchanging interfaces between platforms, an essential process of information searching development, needs to be enhanced significantly. The main reason is that the emergence of too many platforms which follow totally different standards and structures has been making the process become more and more complicated and costly. Therefore, we propose here a system related to platform management, in particular to information harvesting management to let users obtain easily the desired metadata and data content collected from different systems. Moreover, it helps users manage effectively the harvesting process.

Keywords— Harvesting, Metadata Management, Open Data, Datahub, Database

I. INTRODUCTION

Nowadays, big data and artificial intelligence fields have been developed with many innovations coming simultaneously from numerous sources, such as data scientist, system builders, and application developers [1]. With the increasing amounts of data and metadata, there are more and more challenges in information exchange management [2]. In addition, there are various platforms for storing, managing, and sharing information such as CKAN [3], and DKAN [4] as open-sources, and Socrata [5] and Junar [6] are commercial software. Each of these platforms may follow totally different standards and structures, therefore causing a great concern for users who are interested in adopting any of the data of these platforms.

To solve the above problem, a general approach is to create a common module for collecting data and metadata from other platforms. Nevertheless, this approach may have some limitations as follows. First, the current solutions mainly focus on some specific platforms. It means that various metadata and data standards are not considered significantly, causing many difficulties in extending functions and collecting information [7]. Second, there is no clear model structure for storing and querying a set of data collecting processes. As a result, it is difficult to manage effectively the collected metadata and data.

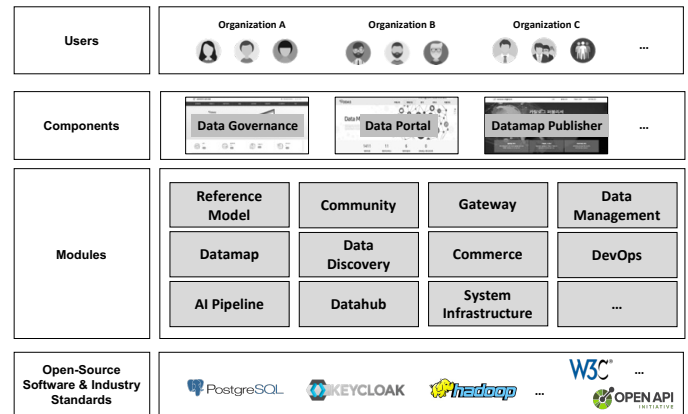


Figure 1. SODAS Platform

In this study, we propose Metadata Management and Harvesting System (MMHS) in smart open datahub that aims to solve the limitations of the approach described above. By improving these disadvantages, we intend to provide a flexible system with standard mechanisms for managing the harvesting process effectively. We first introduce our Smart Open Data As a Service (SODAS) platform and then explain in detail the architecture of MMHS.

II. SMART OPEN DATA AS A SERVICE (SODAS)

This section focuses on introducing the overview of SODAS platform shown in Figure 1. The purpose of SODAS is to become the all-in-one platform targeting both end-users and developers. Therefore, at the beginning, SODAS has been designed based on many industry standards such as W3C DCATv2 [8], OpenAPI Specification [9]. Moreover, by utilizing several popular open-source software, we have developed and organized the main functions of SODAS into different modules to guarantee the extendibility and availability of the platform. For developers, according to their specific demands, the plug-in architecture of SODAS lets the function and module development become much easier and faster. For end-users, due to multitenancy support, different users with different roles from different organizations can together use SODAS without any restrictions. Additionally, we have developed several components such as Data Governance, Data Portal, and Datamap Publisher to let users interact with the platform more conveniently. Note that, each component has

been created by connecting more than one module including Reference Model, Community, Gateway, Data Management, Datamap, Data Discovery, Commerce, Devops, AI Pipeline, Datahub, and System Infrastructure. Among these modules, our MMHS belongs to Datamap which is mainly used to support metadata management, harvesting, and publishing functions.

III. METADATA MANAGEMENT AND HARVESTING SYSTEM

In this section, we describe in detail the entire architecture of MMHS and explain how each function block works together in two specific scenarios. Figure 2 shows the overall design of MMHS, included four main blocks: API, Handler, Harvester and Data Queue. Note that, each block interconnects with other blocks mainly via queues in asynchronous mode to enhance security, performance, and connectivity.

Firstly, in API block, there are four function blocks: Authentication/Authorization, Config Manager, Source Manager, and Job Manager. More specifically, Config Manager is responsible for managing the configuration information such as default data storage, mapping rules from different standards to the datahub's standard. Source Manager is responsible for managing the source information such as name, access endpoint, and standard type. Job Manager is used for managing the job information such as started time, ended time, and current status.

Secondly, Handler block consists of four function blocks: Authorizer, Cleaner, Elastic Indexer, and Queue Manager. While the first three blocks are mainly responsible for improving security and performance, Queue Manager is used to manage the queue information in Data Queue for interconnection. Note that, there are different kinds of queues managed in Data Queue. For example, JOB_QUEUE queue contains the running jobs while DELETED_JOB_QUEUE queue contains the finished jobs. Additionally, for each running job ($J_1, \dots, J_{n-1}, J_n$), the corresponding queue JOB_METADATA_QUEUE_ J_x and JOB_DATA_QUEUE_ J_y are generated for metadata interexchange.

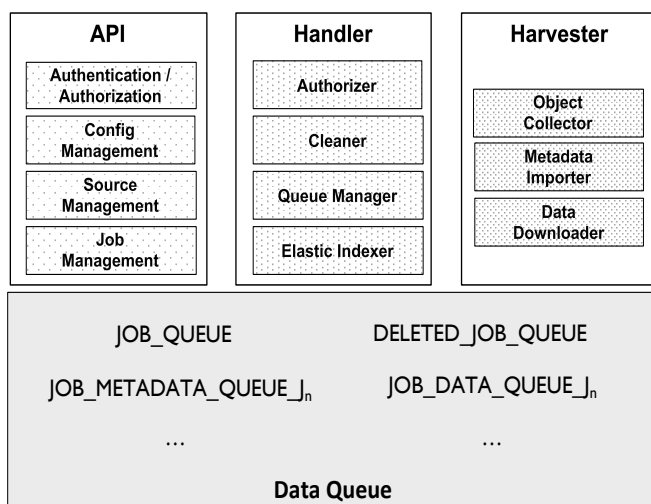


Figure 2. The overall architecture of MMHS

Thirdly, there are three function blocks in Harvester block: Object Collector, Metadata Importer, and Data Downloader. In

particular, Object Collecting takes charge of requesting the metadata from external systems based on the registered source information. Metadata Importer is responsible for mapping and storing the metadata into database while Download Importer is responsible for downloading and storing the data content. It means that database is not only responsible for storing the internal metadata such as source, jobm and config but also for storing the metadata collected from external systems. About the data storage, different kinds of systems such as Local File System, Hadoop File System, and Object Storage is used to store the data contents downloaded from external systems. Note that, the external systems are built based on different standards and structures such as CKAN [3], DKAN [4], and Socrata [5].

Figure 3 shows the scenario for starting the information harvesting process in our system. In the first step, user sends a request to Source Manager for obtaining a list of registered sources. User then selects one source and sends a request to Job Manager for starting the information harvesting process of that source. Based on the job information in database, the Job Manager checks whether there is any invalid information in the request or not. For example, the requested source cannot have more than one running job at the same time. If any issues are found, the user needs to verify the source and perform again the job selection step. In the next step, Job Manager asks Queue Manager to create a new JOB_METADATA_QUEUE_ J_n queue and a new JOB_DATA_QUEUE_ J_n queue for job J_n . Job Manager also asks Queue Manager to add job J_n to JOB_QUEUE queue and change the status of job J_n into “running” status in database. After that, Object Collector, Metadata Importer and Data Download which are continuously listening the queues start their process. Object Collector gets job J_n from JOB_QUEUE queue, asks Config Manager for configuration information and Source Manager for source information. Metadata Importer gets job J_n from JOB_QUEUE queue, asks Config Manager for configuration information, asks Source Manager for source information and then starts monitoring the JOB_METADATA_QUEUE_ J_n queue. Data Downloader gets job J_n from JOB_QUEUE queue, asks Config Manager for configuration information, asks Source Manager for source information and starts monitoring JOB_DATA_QUEUE_ J_n queue. Note that, job J_n may need to wait for being processed until the other jobs which were added before are processed. In the next step, Object Collector sends requests to the external system for obtaining a list of data's metadata. Note that, each item of this list contains many properties such as title, description. After that, Object Collector asks Queue Manager to add the items of this list ($m_1, m_2, \dots, m_n$) to JOB_METADATA_QUEUE_ J_n queue. Metadata Importer gets metadata m_n from JOB_METADATA_QUEUE_ J_n queue, performs mapping the properties of m_n with the Datahub's standard based on the configuration information and store the mapped m_n into database. For example, the “title” property can be mapped into the “name” property. Metadata Importer then asks Queue Manager to add the items of metadata list ($d_1, d_2, \dots, d_n$) which include direct data content download endpoints to JOB_DATA_QUEUE_ J_n queue if data download function is selected. Finally, Data Downloader gets metadata d_n from

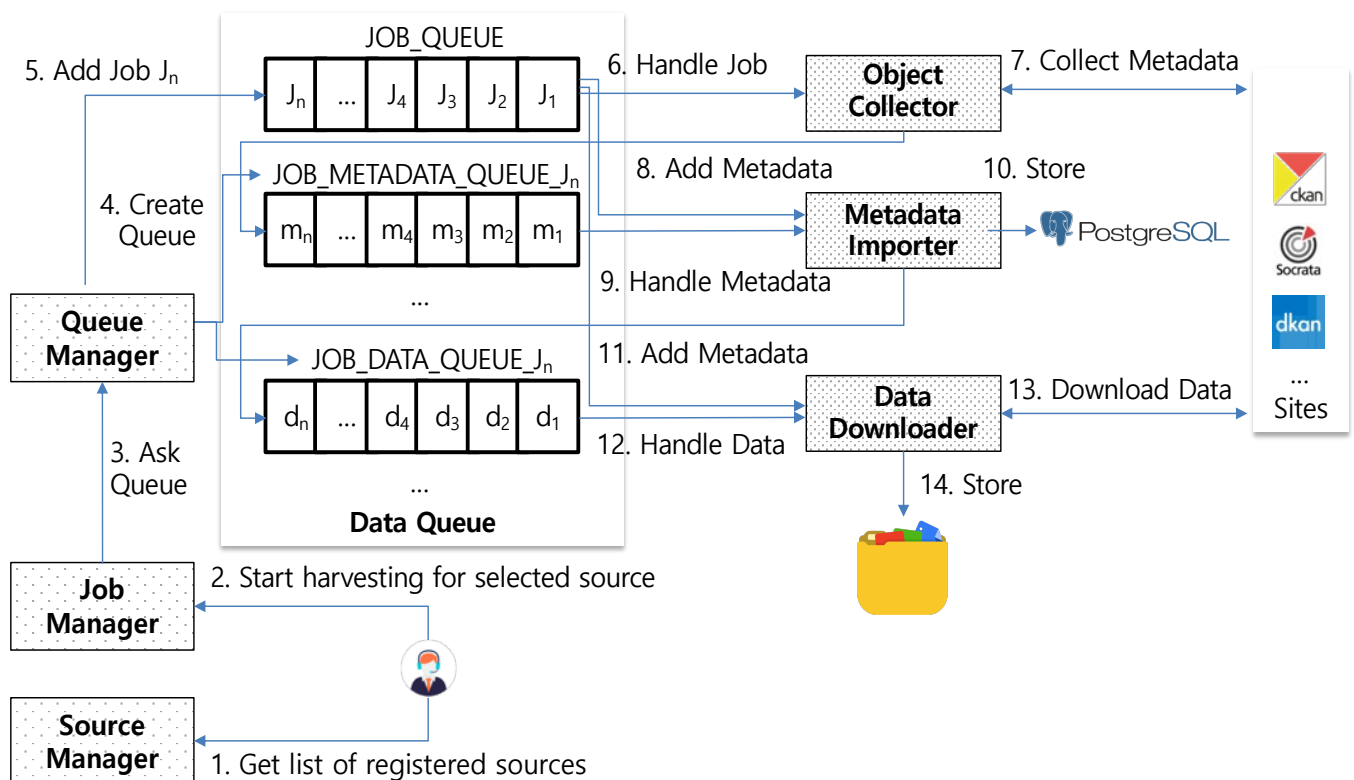


Figure 3. The procedure for starting the information harvesting process

JOB_DATA_QUEUE_Jn queue, performs downloading d_n and stores the result into the appointed storage system in Data Storage.

The procedure for stopping the information harvesting process is much simpler. In the first step, user sends a request to Job Manager for obtaining a list of running jobs. User then selects a job J_n and sends a request to Job Manager for stopping the information harvesting process. Based on the job information in database, the Job Manager checks whether there is any invalid information in the request or not. For example, the requested job already is in “stopped” status. After that, Job Manager asks Queue Manager to delete JOB_METADATA_QUEUE_Jn queue and JOB_DATA_QUEUE_Jn queue. Job Manager also asks Queue Manager to add job J_n to DELETED_JOB_QUEUE queue and change the status of job J_n into “stopped” status in database. After that, Object Collector gets job J_n from DELETED_JOB_QUEUE queue, stops processing job J_n and continues to process the other jobs in JOB_QUEUE. Metadata Importer gets job J_n from DELETED_JOB_QUEUE queue and stops processing the metadata related to job J_n and continues to process the metadata of other jobs. Data Downloader gets job J_n from DELETED_JOB_QUEUE queue and stops processing the data related to job J_n and continues to process the data of other jobs.

IV. CONCLUSIONS

In the paper, we explained the design of MMHS in SODAS platform for better data and metadata exchange. Firstly, we introduced briefly the overview of SODAS platform. Secondly,

we designed the three main function blocks of MMHS in detail. Thirdly, through two specific scenarios, we demonstrated that the described system can alleviate the consequences of different standards or structures for metadata and data management. Moreover, MMHS not only helps users to access easily their desired metadata and data collected from different external systems but also lets users manage effectively the harvesting process.

ACKNOWLEDGMENT

This work was supported by the Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. 2020-0-00077, Core Technology Development for Intelligently Searching and Utilizing Big Data based on DataMap).

REFERENCES

- [1] A. Oussous, F. Benjelloun, A. A. Lahcen and S. Belfkih, “Big Data technologies: A survey,” *Journal of King Saud University - Computer and Information Sciences*, vol. 30, pp. 431–448, Oct. 2018.
- [2] Suchanek, Fabian, and Gerhard Weikum. “Knowledge harvesting in the big-data era.” In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 933-938. 2013.
- [3] (2021) The CKAN website. [Online]. Available: <http://ckan.com/>.
- [4] (2021) The DCAN website. [Online]. Available: <http://getdcan.org/>.
- [5] (2021) The Socrata website. [Online]. Available: <http://socrata.org/>.
- [6] (2021) The Junar website. [Online]. Available: <http://junar.com/>.
- [7] D. Kim, M. S. Gil, M. C. Nguyen, H. S. Won, and Y. S. Moon, “CKAN Harvester Improvement for Efficient Open Data Collection and Management,” *ETRI Journal*, Vol. 43, No. 5, pp. 835-855, Oct. 2021.
- [8] R. Albertoni, D. Browning, S. Cox, A. G. Beltra, A. Perego and P. Winstanley, “Data Catalog Vocabulary (DCAT) - Version 2,” *The World Wide Web Consortium (W3C)*, Feb. 2020.

- [9] D. Miller, J. Whitlock, M. Gardiner, M. Ralphson, R. Ratovsky and U. Sarid, "OpenAPI Specification version 3.1.0," *OpenAPI Initiative*, Feb. 2021.



Minh Chau Nguyen received B.S. degree (2009) in Computer Science from the University of Sciences, Ho Chi Minh, Vietnam and M.S. degree (2013) in Computer Science from the Korea Advanced Institute of Science and Technology (KAIST). He is currently a senior researcher of the CybreBrain Research Section at Electronics and Telecommunications Research Institute (ETRI). His research interests include big data management, architecture, and distributed systems



Heesun Won received B.S.(1990) degree in Computer Science from Yonsei University and the M.S.(1992), Ph.D. (2016) degrees in Computer Science from Korea Advanced Institute of Science and Technology (KAIST). From 1992 to 1999, she was a researcher in Korean Broadcasting System. She is currently a principal researcher of CybreBrain Research Section at Electronics and Telecommunications Research Institute (ETRI). Her research interests include intelligent data hub, dataset curation, and *aaS platform.



Siwoon Son received B.S.(2014), M.S. (2016), and Ph.D. (2020) degrees in Computer Science from Kangwon National University. He is currently a postdoctoral researcher of the CybreBrain Research Section at Electronics and Telecommunications Research Institute (ETRI). His research interests include data mining, big data analysis and management, distributed storage systems, distributed processing systems, and distributed database systems.



Yerin Woo received B.S. degree (2019) in Engineering Information&Computer Engineering Computer Engineering from the Hongik University. She is currently a researcher of the CybreBrain Research Section at Electronics and Telecommunications Research Institute (ETRI). Her research interests include big data management, intelligent data hub and dataset curation.



Hyeonjae Jeong received B.S. degree (2020) in Computer Engineering from the Inje University. He is currently a researcher of the CybreBrain Research Section at Electronics and Telecommunications Research Institute (ETRI). His research interests include big data management, intelligent data hub and identity access management.