ICIC Express Letters
Part B: Applications
Volume **11**, Number **10**, October **2020**

ICIC International ©2020 ISSN 2185-2766
pp. **953–959**

BUILDING METHODS OF INTELLIGENT DATA CATALOG BASED ON GRAPH DATABASE FOR DATA SHARING PLATFORM

Mi-Young Choi[1], Chang-Joo Moon[2] and Sung-Jae Jung[3]

[1]Research and Development Department
CnTech Systems Co., Ltd.
1205-ho, 130, Gwannaru-ro, Seongdong-gu, Seoul 04788, Korea
michelle@cntechsystems.com

[2]Department of Smart Vehicle Engineering
University of Konkuk
120 Neungdong-ro, Gwanggin-gu, Seoul 05029, Korea
cjmoon@konkuk.ac.kr

[3]Big Data Center
Seoul National University Bundang Hospital
172, Dolma-ro, Bundang-gu, Seongnam, Gyeonggi-do 13620, Korea
sungjae.jung@snubh.org

Received March 2020; accepted June 2020

Abstract. *Recently, the private and public sectors are constantly registering various types of data on a data sharing platform or an open data platform and creating value through linkage and expansion of data. Data catalogs are tools that can help you quickly find and understand the data accumulated on these data-sharing platforms. In this paper, we propose a method to build a data catalog based on a graph database. In the graph database, the joins are significantly reduced compared to the relational type due to the structural characteristics, while the query processing performance is improved. We show the process of transferring the proposed data catalog from the relational data model to the graph data model. We show that the same information retrieval is performed with a simpler query and higher performance in the graph database.*
Keywords: Data catalog, Metadata, Relational database, Graph database, Data sharing platform

1. **Introduction.** Globally, the importance of data has been dramatically increasing, paying attention to data values [1]. A data lake is a centralized repository that allows you to store all your structured and unstructured data at any scale [2]. Data sharing platforms, such as the open data platform, allow companies and institutions to create new value by linking data beyond individual data utilization [3]. In this paper, we will use the open data platform and data sharing platform in the same sense. The success of data sharing platforms depends on the quality (e.g., accuracy, completeness, timeliness, and consistency), the use of sharing data, and documenting the emerging impacts and benefits. Accessibility, interoperability and standards are critical factors to promote the reuse of sharing data [4].

The data catalog is a metadata management service that helps to quickly find and identify the data registered in the data sharing platform. The information search functions in the data catalog are different from general data dictionary [5].

The data catalog centralizes the metadata in one place, provides a complete view of the data stored in the database, and includes information about the location, profile, statistics, summary and other comments of the data. It also makes it easier for users to search and manage data sources, and to help organizations make appropriate decisions

DOI: 10.24507/icicelb.11.10.953

based on data usage [6]. To do this, the data catalog should provide information about the data input by the user when registering the data on the data sharing platform, the existing data dictionary information about the structure in which the data is stored, and information about internal characteristics of data, that is, data profiling information. Profiling techniques allow you to see data distribution and patterns through profile results for critical columns, and to ensure that data rules are compliant within the lowest possible level.

In general, queries for retrieving information from a data catalog require a significantly larger number of joins. In a relational database, joins are often a major factor in performance degradation, so there is also a denormalization to reduce joins and hence data inconsistencies. In addition, changes to the table structure in the relational database yield null data. Graph databases have often been mentioned in recent years as viable alternatives to relational models. The graph database is particularly well-suited for storing interconnected data to perform multiple levels of interconnectivity [7].

In this paper, we propose a new type of data catalog that uses meta information registered by users when registering data on data sharing platform, profiling information on some of data, and schema information on stored data. These three types of information will be stored in the graph database instead of the existing relational database. In the paper, the relational database is used only to manage the data registered by the user, and the data catalog will be implemented as a graph database rather than a relational database.

The remainder of the paper is organized as follows. Section 2 summarizes related research about data catalogs of data sharing platforms. Section 3 presents the building methodology steps for data catalog and explains each of its components. Section 4 presents the migration steps from Relational Data Model (RDM) to the Graph Data Model (GDM) while presents the evaluation and verification of proposed data catalog based on graph. Finally, in Section 5, we present our conclusions.

2. **Related Work.**

The Comprehensive Knowledge Archive Network (CKAN) is a public data open source platform developed by Open Knowledge [3]. CKAN supports key functions of data distribution, storage, visualization, linkage, retrieval and utilization, extension of functions, security and authentication, history and usage statistics, and communication. Drupal-based open data portal based on CKAN (DKAN) is a complimentary offering to CKAN in the effort to make data more open and accessible to create an open data catalog through creating an open data catalog [8]. Data Catalog Vocabulary (DCAT) is an RDF vocabulary designed to facilitate interoperability between data catalogs published on the Web [9]. RDF databases are very good at representing complex metadata, reference, and master data [10]. RDF is a graph data model that has been around since 1997. It is a W3C standard, and it is used to power schema.org and Open Graph, among other things. Plus, there is a bunch of RDF-based graph databases out there, some of which have been around for a while and can do things other databases cannot do [10].

The graph database stores data using vertices and lines (commonly referred to as nodes and relationships or lines) [11]. The main difference is that graph databases do not have a fixed schema as compared to relational databases. Most graph databases are inherently schema-less [11]. Since the graph database stores data as a single object, you can add relationships by adding lines, so you do not have to worry about null values or normalization violations. Predefined schemas are not required to import data types into the database [7]. This is because the data itself determines the structure and relationship of the nodes. Here the relationship is indicated by an arrow of graph. Existing relational databases must use joins to indicate the relationship of the data stored in the table. On the other hand, in the graph database, the relationship between data is directly generated and we use the method to inquire necessary data while traversing relations

between data generated in this way. A data catalog provided by commercial integrated data management products was built from scalable graph databases for rapid updating of metadata, smart searches, and queries, but they do not disclose how they were built into the graph database [12].

3. **Methodology to Building Data Catalog.** This paper proposes the architecture for constructing the data catalog of the data sharing platform using the graph database in Figure 1. The raw data registered by the user on the shared platform is stored in the relational database as it is. The data sharing platform data model used in this paper is CKAN and it manages raw data and metadata registered using PostgreSQL Relational Database Management System (RDBMS). A data catalog based on a graph database consists of the following three elements.

1) User-Described Metadata

User-described metadata is made by the user on the data sharing platform based on 5W1H (Who, What, Where, Why, When and How) of registered raw data and can be freely added to each item [13]. For example, 'When' can only manage the registration date, but it can also manage the last update date. For 'Who', you can register who created it, who uses it, and who is the administrator.

2) Data Profile

The data profile can be created by extracting the information only for the columns selected by the user. Since profiling requires the extraction of the distribution and statistical data of the columns whole data, it can be an expensive job. You can also define rules for values that are distributed in the columns. For example, if you define a rule that allows only 'M/F' in the gender column, data that violates this rule can be refused to register.

3) Data Schema

It is also called a data dictionary. When a two-dimensional table-like source data is registered in a relational database, the relational database generates table schema information such as table name, column name, data type, and data length.

When raw data is uploaded on the shared platform, the data schema information is automatically registered in the relational database, so the graph database reads this information to build the data catalog.

Figure 2 shows the steps to build a data catalog. When a user wants to upload data to a shared data platform, metadata in the metadata registration step is stored in the graph

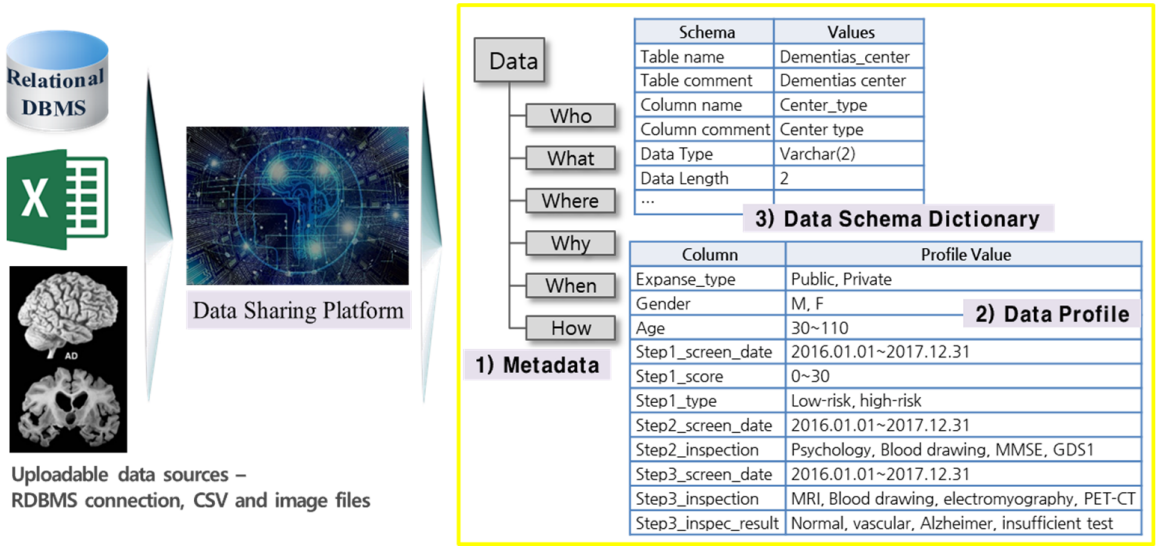


FIGURE 1. Main components of proposed data catalog

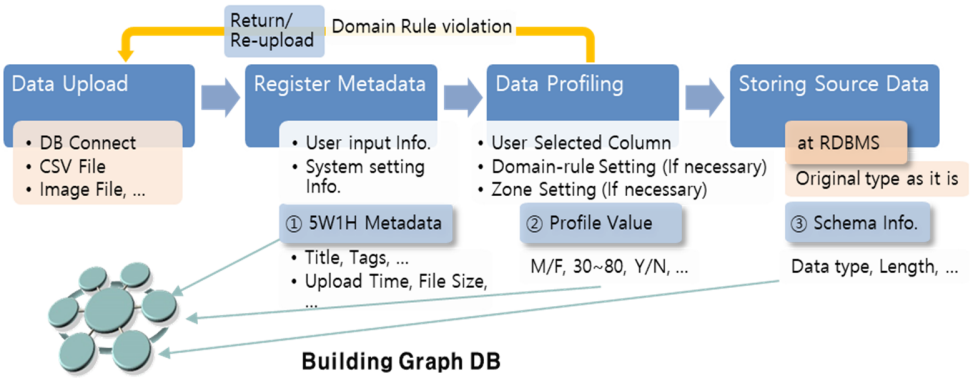
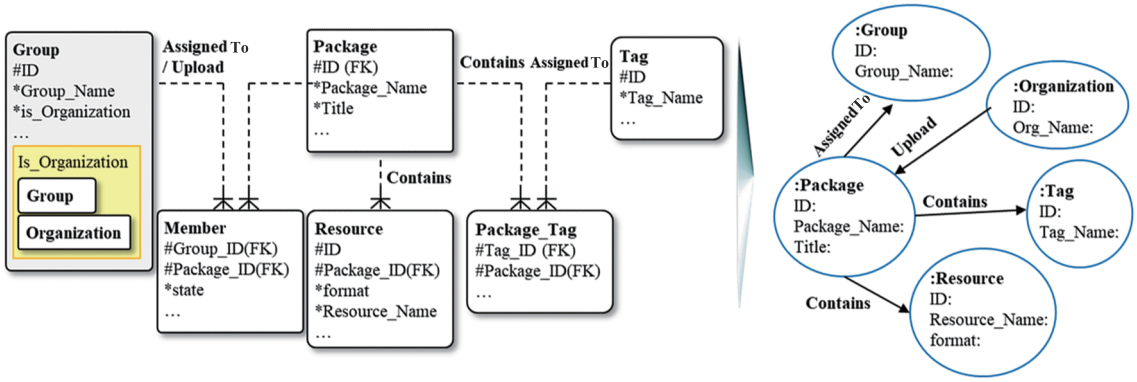


FIGURE 2. Data catalog building steps

database as information input by the user or automatically derived from the system. The next step is to store the data profile values, but profile results are not stored in the graph database immediately. The data schema information is first built into the graph database, and then only the column values that have been profiled are stored in the graph database. This is because the profile target column is a subset of the schema information.

To store information in the graph database, the graph database as well as the relational database must first define the graph data model. You can create a data node in the graph database according to the defined graph data model and connect the relationship line when the nodes are created. A representative data sharing platform, CKAN's metadata structure is specified using an Entity-Relationship Diagram (ERD) in Figure 3(a) at the conceptual data model level and Baker notation [14] is used. This ERD is used to describe the process of converting an existing relational database-based data catalog into a graph database.



(a) Relational data model (b) Graph data model

FIGURE 3. Relational data model and graph data model of data sharing platform's metadata

The key components of an ERD are entities, relationships, and attributes [15]. Entities can be divided into independent entities and dependent entities, and cross entities or join entities are one of the dependent entities. An entity can have a hierarchical structure and is called a super-type at the top and a sub-type at the bottom depending on where the entity is in the hierarchy [15]. The properties of an entity are specified by attributes. Some of the attributes are called Key or Primary Key (PK) to uniquely identify the entity instance or relational database table row. When this PK is referenced by another entity or table, the attributes inherited by the reference are called Foreign Key (FK).

In Figure 3(a), entities named 'Group', 'Package', 'Tag' are independent entities and 'Resource', 'Member', 'Package_Tag' are dependent entities and 'Member', 'Package_Tag'

is a join entity. 'Group' has sub-type entities called 'Group' and 'Organization' as subordinate. The PK of each entity is marked with '#' in front of the attribute name and FK is marked with '(FK)' after the attribute name.

Figure 3(b) shows an example of converting the ERD in Figure 3(a) to the graph model. To convert the relational data model into a graph data model, we use the methodology mentioned in [6] as shown in the following four steps. This methodology can be used independently of the type of graph database.

Step 1: Map independent and dependent entities that are not join entities to nodes.
Step 2: Map join entities to relationship lines.
Step 3: A column can be converted to a relational attribute or a node attribute first, but whether to keep it as an attribute or to convert an attribute to a node must be constantly considered in the modeling process.
Step 4: If necessary, you can also create an index on some nodes of the graph.

Figure 3(b) shows an example of changing the ERD in Figure 3(a) to the graph data model according to the above step. After mapping the independent entities 'Group', 'Package', 'Tag' and 'Resource' which is not a joining entity and a dependent entity in Figure 3(b) to a node first, the join entities 'Member' and 'Package_Tag' are mapped to 'Assigned To', 'Upload', 'Contains', etc. which are related to each characteristic. In this process, the sub-types of independent entities, 'Group' and 'Organization', can be regarded as independent entities and they are mapped to separate nodes in the graph data model.

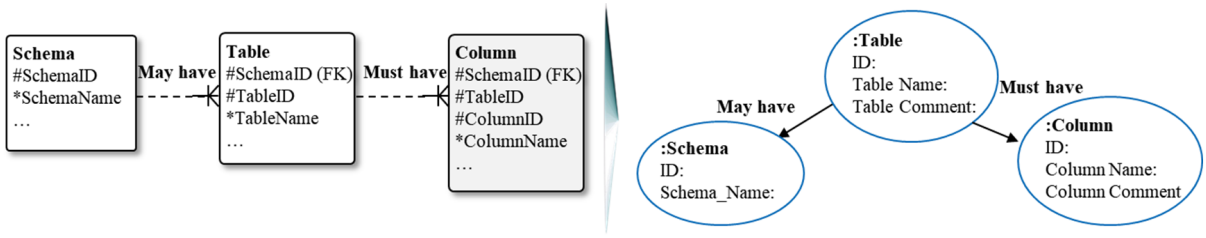
Table 1 shows a Cypher statement that sets nodes and relationship lines in the graph database according to the graph model. In this paper, Neo4j graph database is used. Cypher is a representative graph database query used in many graph databases besides Neo4j.

Table 1. Cypher query to create node and relationship

Cypher
Node Create Query: *Create (n:package {id:"3ef396b2-dbbc-46c5-85bc-ed9d050ee29f",* *type:"dataset", title"Handicapped Children Special Schools Status"})* *create (n:tag {id:1,name:"Handicapped"})* *create (n:tag {id:2,name:"Special Schools"})* *Relationship Create Query:* *match (n:package {id:"3ef396b2"}), (m:tag {id:1})* *create (n)-[:contains]->(m)*

4. **Transfer from Relational Data Model to Graph Data Model.** We present the data schema information which is one of the metadata is constructed using a graph model and then the relational schema information is added. We query on the data schema information using the Cypher. This step is preceded by the last step as shown in Figure 2, but the step is added front of the profile model in the graph data model construction as mentioned above.

Figure 4(a) shows the data schema information when raw data is stored in a relational database. 'Schema' is the subject that creates the table in the relational database, 'Table' is the table list that stores the raw data, 'Column' is the column information of each table, and it is designed with the Entity Attribute Value (EAV) [15] model mentioned above in the actual relational database. The proposed data catalog maps the data schema information operated in the relational database to the graph database as shown in Figure 4(b).



(a) Relational data model (b) Graph data model

FIGURE 4. Relational data model and graph data model for RDBMS schema

Figure 4(b) transformed the relational model of Figure 4(a) into a graph data model. 'Schema', 'Table', and 'Column' are not all join entities, so they are mapped to each node, and the role name of the relationship becomes the name of the relationship line.

Figure 5 shows a graph data model which connects metadata of data sharing platform Figure 3(b), schema information Figure 4(b), and profile information constituting the data catalog proposed in this paper. In Figure 5 rule information drawn in gray node is not detailed because it depends on how the user defines them. Profile value node stores profiling results for some or all the columns. Rules may map to the constraints in the relational database.

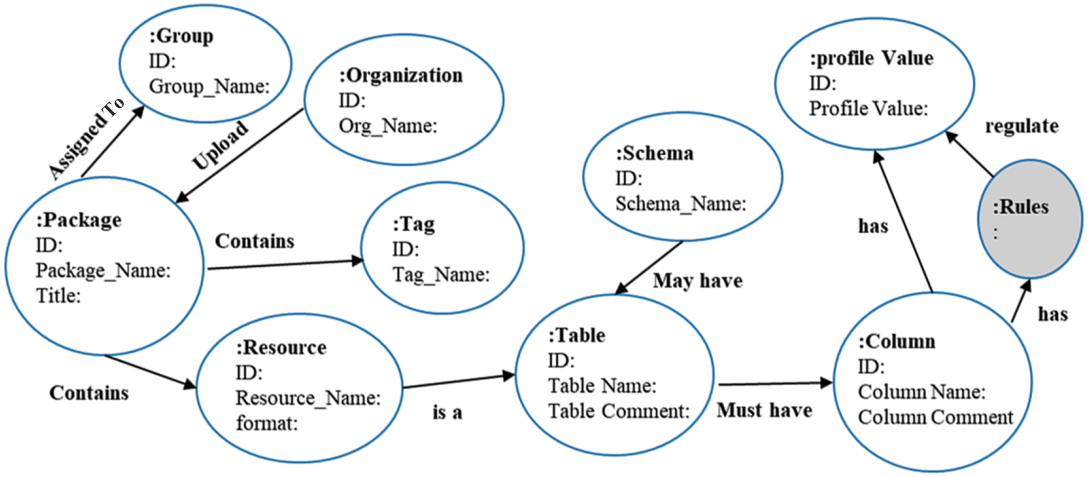


FIGURE 5. Graph data model for data catalog

Table 2 is comparing query performance and query contents obtained by querying the same metadata retrieval condition with relational database and graph database. It was confirmed that the query performance was about 10 times different in the general condition.

If you execute this condition in a relational database, you can get a list of desired result data by joining 'Tag', 'Package_Tag', 'Package', 'Member', and 'Group'. For decades relational databases have been a dominant choice. All joins in RDBMS are executed every time you query (traverse) the relationship. Executing a join means to search for a key in another table with B-Tree indices which needs O(log(n)) cost to look up a key. It means more joins make more lookups and make slower response. In some tests, Cypher is much faster than SQL queries in complex query search conditions [16].

5. **Conclusions.** In this paper, we propose a new data catalog for data sharing platform and suggest a method to construct the data catalog with graph database instead of relational database. We have confirmed that the graph database has better query performance than the relational database in relation-oriented search. DCAT is an RDF

TABLE 2. Query and execution performance comparison

SQL	*Cypher*
*select * from package* *where owner_org = (select g.id from "group" g* *where g.id = (select p.owner_org from package p* *where p.id = (select pt.package_id from* *package_tag pt inner join tag t on* *pt.tag_id = t.id and t."name" = 'Children'* *where pt.package_id = '4fc9c5d7')))* *and state = 'active'*	*match (t:tag {name:'Children'})–>* *(pt:package_tag)–(p:package)* *–>(m:member) <–(g:group2)* *where pt.package_id = p.id* *and p.id = m.table_id* *and g.is_organization = "True"* *and p.state = 'active'* *return t,p,m*
9ms ∼ 336ms	*1ms ∼ 44ms*

Search requirements: Other data registered by the organization that registered data lists with the tag 'Children'

vocabulary designed to facilitate interoperability between data catalogs published on the Web [9]. And RDF is a graph data model that has been around since 1997. DCAT is more suitable for graph databases with the inherent structural characteristics of RDF [17]. Future work tries to build DCAT as a graph database.

Like the relational data model, the graph data model does not have a single model for a single task. It is necessary to continuously study the construction of a graph data model and a graph database construction suitable for a data sharing platform.

Acknowledgment. This work is partially supported by CANE (CDM based intelligent Algorithm Network Environment Platform) project. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] LOD-based data management paradigm conversion strategy, *IT & Future Strategy*, no.1, 2014.
[2] *https://aws.amazon.com*, Accessed on 2019.08.10.
[3] Korea Data Agency, *A Study on the Improvement of Private Data Cooperation Using CKAN*, 2017.
[4] W. Carrara, S. Enzerink, F. Oudkerk, C. Radu and E. van Steenbergen, *Open Data Goldbook for Data Manangers and Data Holders*, European Commission, 2018.
[5] *https://www.dataversity.net/what-is-a-data-catalog/*, Accessed on 2019.05.27.
[6] *https://searchdatamanagement.techtarget.com/definition/data-catalog*, Accessed on 2019.05.27.
[7] *https://database.guide/whats-the-difference-between-graph-database-and-relational-database/*, Accessed on 2019.05.28.
[8] *https://dkan.readthedocs.io/en/latest/introduction/dkan-ckan.html*, Accessed on 2019.08.10.
[9] *https://www.w3.org/TR/vocab-dcat/*, Accessed on 2019.08.10.
[10] *https://www.zdnet.com/article/graph-databases-and-rdf-its-a-family-affair/*, Accessed on 2019.08.10.
[11] I. Robinson, J. Webber and E. Eifrem, *Graph Databases*, 2015.
[12] *https://www.informatica.com/products/big-data/enterprise-data-catalog.html*, Accessed on 2019.06.01.
[13] Donna Burbank, *Modern Metadata Strategies*, Global Data Strategy, Ltd., 2018.
[14] R. Barker, *CASE Method: Entity Relationship Modelling*, Addison-Wesley Professional, MA, 1990.
[15] T. J. Eggebraaten, J. W. Tenner and J. C. Dubbels, A health-care data model based on the HL7 Reference Information Model, *IBM Systems Journal*, vol.46, no.1, 2007.
[16] F. Holzschuher and R. Peinl, Performance of graph query languages: Comparison of cypher, gremlin and native access in Neo4j, *Proc. of the Joint EDBT/ICDT 2013 Workshops (EDBT'13)*, pp.195-204, 2013.
[17] T. Berners-Lee, *Relational Databases on the Semantic Web*, http://www.w3.org/DesignIssues/RDB-RDF.html, Accessed on 2019.06.01.