ISSN: 0974-5823 Vol. 7 No. 1 January, 2022

International Journal of Mechanical Engineering

Search performance evaluation of JSON datasets in a big data environment

Jung Kyu Park[1], Eun Young Park[*2]

[1]Professor, Department of Computer Software Engineering, Changshin University, 51352, Korea

[*2]Professor, Rinascita College of Liberal Arts and Sciences, Shinhan University,11644, Korea

Abstract.
BACKGROUND/OBJECTIVES: In recent years, IoT is rapidly increasing, and as smart devices become active in large quantities, the amount of data produced is rapidly increasing.
METHODS/STATISTICAL ANALYSIS: In addition, the generated data is a mixture of structured and unstructured data. Big data means processing and analyzing such large amounts of data. XML is widely used for data storage and retrieval. However, the XML method has a problem of inefficiency that takes a lot of time when searching for a large amount of data.
FINDINGS: In this paper, we analyzed the relational data model that is widely used in databases, the XML model and JSON model that are widely used for data management. In addition, a JSON model was presented to enable quick search in a big data environment. To analyze the performance of the proposed JSON modeling, a comparison experiment with XML was performed. By performing an experiment using data having 50,000 records, it was found that faster search was performed with the proposed JSON method. In addition, it was verified that the search speed linearly increases even when the data increases. In contrast, the XML model search was slow and unstable. Through experiments, it was found that JSON modeling performed faster search in a big data environment.
Improvements/Applications: In the future, we plan to improve the model proposed through database experiments in larger quantities.

Keywords: Big Data, Evaluation, JSON, Relational Database, XML

1. INTRODUCTION

Big data refers to extracting useful branches from large amounts of data structured or unstructured data that are difficult to process with existing database technology and analyzing the results [1,2]. Such big data is divided and stored in several storage devices and may also be located in several locations. Big data is currently being used not only in the field of computer systems, but also in various fields such as healthcare, machine industry, and finance [3]. In summary, big data refers to a platform that provides software and processing procedures for companies to create, process, and manage large amounts of data. Such big data must handle data storage, analysis, transmission, sharing, visualization, query, and data update.

In recent years, research related to big data is increasing rapidly, and the research predicted that big data technology could grow further [4,5]. When accessing large-scale data such as big data, the speed and efficiency of data access is important [6]. In general, to evaluate the efficiency of accessing large-scale data, the data patch time using a query is measured. In order to process a large amount of data, relational database and XML are mainly used. Relational databases are traditional data processing methods that store and manage big data even recently. In this method, data is managed as a two-dimensional table. DBMS (Database Management System) is a set of software that allows multiple users to easily access data in a database [7].

However, when applying the existing DBMS to big data and importing the data, the problem of time inefficiency occurs. As such, the way to solve the patch time problem is to use XML. XML (Extensible Markup Language) was developed by W3C and is used as a standard for transmitting data over the Internet. In particular, XML is widely used when storing and managing large amounts of data [8,9]. The XML-based method is used in restricted fields such as education, business, and manufacturing [10].

In this study, we propose a JSON (JavaScript Object Notation) method to store and manage large amounts of data. JSON deals with data consisting of attribute-value pairs and is an open standard in text format. Such JSON was chosen because it provides high throughput and low latency [11,12]. JSON originated from JavaScript but is now a language independent data format. For this reason, various languages such as C/C++ and Python can be used to create and manage JSON data. In addition, it shows very good performance compared to a relational database and XML. According to existing studies, it can be seen that web service applications show better performance than XML [10].

2. DATABASE MODELING

In this paper, we describe three database models for expressing published data. The first is the relational database method, the second is the XML method, and the last is the JSON method. Figure 1 shows the relationship between the publication data used in the paper as a block diagram. Publication data in the database are classified into journal articles, conference thesis, master's/doctoral dissertations, and web sites.

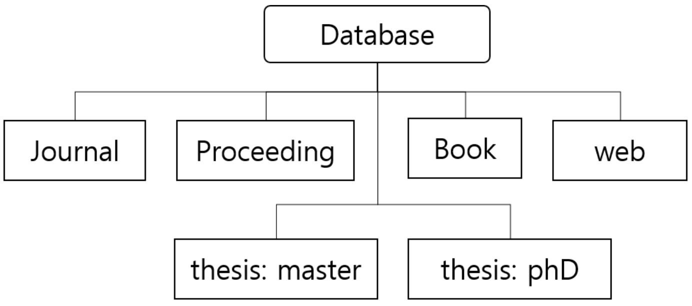

Figure 1. Structure of database

2.1. RELATIONAL MODEL

A relational model is the model most used from the initial stage of database development to the present. [4,5]. In the relational database model, attributes and user data are stored in tables. Relational data can have a relationship in which a row of a table is connected with a row of another table. Tables 1 to 3 below show publication information of thesis and books.

Definition 1: A tuple is {$A_1 = v_1$,... , $A_n = v_n$}, where A_1, ... , A_n means attribute and v_1, ... , v_n means the corresponding value.

Definition 2: The characteristics of the database table can be defined as a schema. Also, the tuple (T) is composed of several attributes. For example, assuming that U consists of the attribute set {C1, ..., Cn} and w is a tuple, the value of w.U can be specified as (w.C_1, w.C_2, ..., w.C_n).

Definition 3: The relationship of the characteristic T means a set of tuples having the T characteristic.

Table 1: Information of Journal

Key	Author	Title	Journal	Vol	Pages	Year	URL
journals/see/MurphyG08	Colleen Murphy, Paolo Gardoni	The Acceptability and the Tolerability of Societal Risks: A Capabilities-based Approach.	Sci. Eng. Ethics	14	77-92	2008	db/journals/see/see25.html#Davies19
journals/see/Dennis20	Louise A. Dennis	Computational Goals, Values and Decision-Making.	Sci. Eng. Ethics	28	2487-2495	2020	db/journals/see/see26.html#Dennis20
journals/spm/Trappe17a	Wade Trappe	No Need for Speed : More Signal Processing Innovation Is Required Before Adopting Automated Vehicles [In the Spotlight].	IEEE Signal Process. Mag.	234	124-122	2017	db/journals/spm/spm34.html#Trappe17a

Table 1 shows journal article information. Table 1 consists 8 attributes (Key, Author, Title, Journal name, Volume, Pages, Year, URL). In the table, the first row represents data attributes, and the remaining rows are tuples representing actual data. The first tuple is {Key=journals/see/MurphyG08, Author= Colleen Murphy, Paolo Gardoni, Title=The Acceptability and the Tolerability of Societal Risks: A Capabilities-based Approach, Vol=14, Pages=77-92, Year=2008, Journal= Sci. Eng. Ethics, URL=db/journals/see/see25.html#Davies19} do. As such, the signature is expressed as a set of {Key, Author, Title, Journal name, Volume, Pages, Year, URL}.

Table 2: Information of Conference

Key	Author	Title	Crossref	Pages	Year	ee
conf/zum/WinterS03	Kirsten Winter, Graeme Smith	Compositional Verification for Object-Z.	conf/zum/2003	280-299	2003	https://doi.org/10.1007/3-540-44880-2_18
conf/zum/Fischer98	Clemens Fischer	How to Combine Z with Process Algebra.	conf/zum/1998	5-23	1998	https://doi.org/10.1007/978-3-540-49676-2_2
conf/zum/HieronsHS03	Mark Harman, Harbhajan Singh	Automatically Generating Information from a Z Specification to Support the Classification Tree Method.	conf/zum/2003	388-407	2003	https://doi.org/10.1007/3-540-44880-2_23

Table 2 shows conference article information. Table 2 consists 7 attributes (Key, Author, Title, Crossref, Pages, Year, ee). In the table, the first row represents data attributes, and the remaining rows are tuples representing actual data. The first tuple is {Key=conf/zum/WinterS03, Author= Kirsten Winter, Graeme Smith, Title= Compositional Verification for Object-Z., Crossref=conf/zum/2003, Pages=280-299, Year=2003, ee=https://doi.org/10.1007/3-540-44880-2_18}. If we call this tuple t, then t.No=882267. The schema is consisting of set {Key, Author, Title, Crossref, Year, ee}. The actual data is showed in Table 2.

Table 3: Information of Book

Key	Title	Author	Year	Vol	Publisher	ISBN
books/sp/K2009	Introduction to Modern Traffic Flow Theory and Control: The Long Road to Three-Phase Traffic Theory	Boris S. Kerner	2009		Springer	978-3-642-02604-1
books/sp/BagchiC89	Interactive Relational Database design - A Logic Programming Implementation	Tapan P. Bagchi, Vinay K. Chaudhri	1989	402	Springer	3-540-51881-9

Table 3 shows book article information. Table 3 consists 7 attributes (Key, Title, Author, Year, Vol, Publisher, URL, ISBN). In the table, the first row represents data attributes, and the remaining rows are tuples representing actual data. A first tuple is {Key=books/sp/K2009, Title=Introduction to Modern Traffic Flow Theory and Control: The Long Road to Three-Phase Traffic Theory, AuthorBoris S. Kerner, Vol=, Publisher= Springer, Year=2090, ISBN=978-3-642-02604-1}. If we call this *t* as tuple, then **t**.Key=books/sp/K2009. The schema is consisting of set { Key, Title, Author, Year, Vol, Publisher, ISBN }. The actual data is showed in Table 3.

2.2. XML DATA MODELING

XML (Extensible Markup Language) was proposed as the W3C standard in 1998. XML is a text-based markup language very similar to HTML [8.9]. This feature makes it easy to read both humans and machines. Currently, XML is playing an important role in storing and delivering data in many applications. XML was created for the purpose of storing and transmitting data, and is a language for describing the structure of the stored data. Also, tags used in XML are not defined in advance like HTML, but are directly defined and used by the user. Additionally, XML aims to convey data without displaying the data. And the XML document is made up of Unicode characters. Figure 2 shows the contents of Tables 1, 2, and 3 reorganized and written in XML format.

```
<dblp>
        <article>
                <Key>journals/see/MurphyG08</Key>
                <Author>Colleen Murphy, Paolo Gardoni</author>
                <Title>The Acceptability and the Tolerability of Societal Risks: A Capabilities… </Title>
                <Vol>14</vol>
                <Pages>77-92</Pages>
                <Year>2008</Year>
                <Journal>Sci. Eng. Ethics</Journal>
                <URL>db/journals/see/see25.html#Davies19</URL>
        </article>

        <book>
                <Key>books/sp/K2009</Key>
                <Title>Introduction to Modern Traffic Flow Theory and Control: The Long Road to
                        Three-Phase Traffic Theory</Title>
                <Author>Boris S. Kerner</Author>
                <Year>2009</Year>
                <Vol> </Vol>
                <Publisher>Springer </Publisher>
                <ISBN> 978-3-642-02604-1</ ISBN >
        </book>

        <proceeding>
                <Key>conf/zum/WinterS03</Key>
                <Author> Kirsten Winter, Graeme Smith </Author>
                <Title> Compositional Verification for Object-Z. </Title>
                <Crossref>conf/zum/2003</Crossref >
                <Pages>280-299</Pages>
                <Year>2003</Year>
                <ee>https://doi.org/10.1007/3-540-44880-2_18</ee >
        </proceeding>


        ...
</ dblp >
```

Figure 2. Structure of XML format

2.3. JSON Data Modeling

JSON (JavaScript Object Notation) is composed of human-readable text and is a standard for data exchange [11,13]. JSON is an alternative to XML and was developed to make data exchange and storage easier. In this way, JSON is composed of text, and JSON data can be used in various programming languages. XML documents access documents using XML Document Object Model (DOM) [14]. However, JSON can be processed faster than XML because it is parsed and used immediately after receiving the string. However, JSON requires the user to verify the integrity of the transmitted data. Figure 3 shows the contents of Tables 1, 2, and 3 reorganized and written in JSON format.

```
{"article":[{"Key":"journals/see/MurphyG08","Author":" Colleen Murphy, Paolo Gardoni ","title":" The Acceptability and the Tolerability of Societal Risks: A Capabilities ","Vol":"3","Pages":"85-101","Year":"2008","Journal":" Sci. Eng. Ethics ","URL":" db/journals/see/see25.html#Davies19"}],

"book":[{"Key":" books/sp/K2009","Title":" Introduction to Modern Traffic Flow Theory and Control: The Long Road to Three-Phase Traffic Theory ", "Year":"2005", "Vol":"2009", "Publisher":" Springer ", ISBN":"978-0262201629",}],

"proceeding":[{"Key":"conf/zum/WinterS03 ","Author":" Kirsten Winter, Graeme Smith","Title":" Compositional Verification for Object-Z.","Crossref":" conf/zum/2003","Pages":"280-299","Year":"2003","ee":"https://doi.org/10.1007/3-540-44880-2_18"}] }
```

Figure 3. Structure of JSON format

3. Experimental Results

In this chapter, we performed a comparison experiment with XML to check the search performance of the proposed JSON. The experiment was largely performed based on four scenarios. The equipment used in the experiment was Linux Ubuntu 20.02 version. The computer hardware consists of a i7-6700K 4GHz CPU, 16 GB RAM, and a 500 GB SATA-6 SSD. For the experiment, benchmark data provided by DBLP was used, and this data file is composed of about 3GB XML single file [15]. In the experiment, the XML data file was queried using X-Path. In addition, the data converted to JSON was used in the experiment. By executing various queries on two databases, the query execution time and CPU usage were measured. In order to measure the query execution time, a query to search 1000, 5000, 10000, and 50000 data was executed. The execution time was measured by executing the same query 8 times, and the average time and standard deviation were calculated using this.

3.1. Test Data Set

For the data used in the experiment, we use the dataset provided by DBLP. The provided data is a dblp.xml file, which is composed in XML format, and the dataset converted to JSON format was used for experimentation. The original data consisted of about 50,000 records. For the experiment, four experimental queries were written as shown in Table 4.

Table 4: Information of Conference Article

Query	Description
Type 1	Displaying all content with the keyword "/journal" in URL
Type 2	Displaying the title of thesis with the keyword "Data" in the doctoral dissertation
Type 3	Displaying the title of conference paper with author "Kirsten Winter"
Type 4	Displaying the number of doctoral dissertations by year

3.2. QUERY PERFORMANCE

In this section, we conducted experiments to see the retrieval performance of XML and JSON format data. For the experiment, the results were derived by repeatedly performing the four queries specified in Table 4. In order to see the difference according to the change in the size of the data, the experiment was conducted by dividing the data into 4 groups (1000, 5000, 10000, 50000 records). Figure 4 shows the result of executing a Type 1 query. In this experiment, the JSON format improved performance by up to 46% (5000 records) and at least 26% (1000 records). Figure 5 shows the result of executing a type 4 query. In this experiment, the JSON format showed a performance improvement of up to 38% (50000 records) and at least 18% (1000 records).

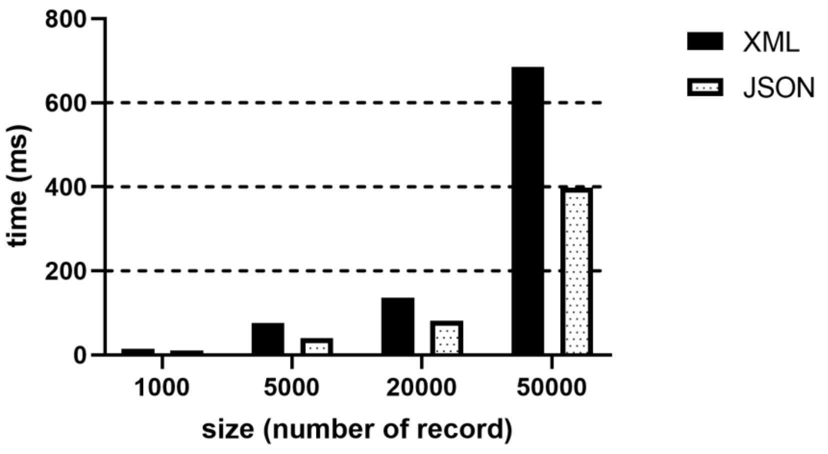

Figure 4. Query performance evaluation with query Type 1

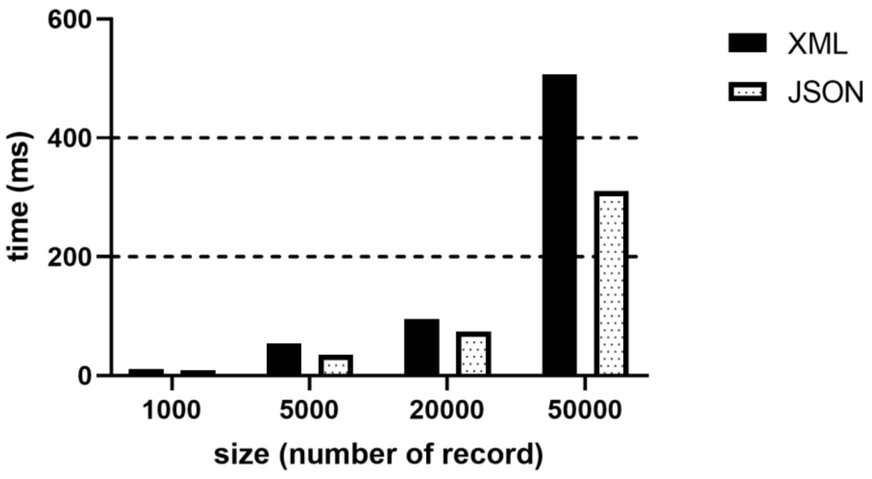

Figure 5. Query performance evaluation with query Type 4

3.3. CPU USAGE

To measure the CPU usage, the two datasets used in the previous query execution were used. For the experiment, the CPU usage rate was measured by repeatedly executing the four queries specified in Table 4. Figure 6 shows the CPU utilization result when executing a type 1 query. In this experiment, the JSON format improved performance by up to 26% (20000 records) and at least 8% (5000 records). Figure 7 shows the CPU utilization result when executing a type 4 query. In this experiment, the JSON format showed a maximum performance improvement of 30% (50000 records) and a minimum of 3% (20000 records). When all four queries were executed, the average CPU usage showed good results by about 10% or more in JSON search.

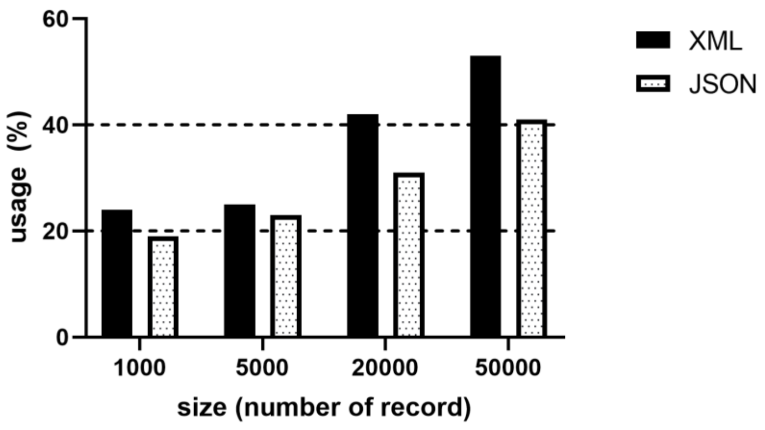

Figure 6. CPU usage evaluation with query Type 1

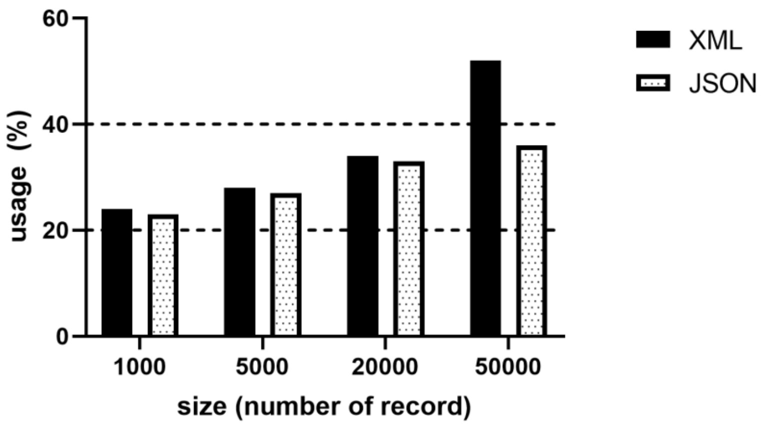

Figure 7. CPU usage evaluation with query Type 4

4. CONCLUSION

In this study, two datasets were used, XML and JSON, to measure data retrieval performance in big data. The basic dataset used in the experiment was a dataset of DBLP, which is widely used in other database experiments, and this dataset has about 50,000 records. The DBLP dataset was composed of two datasets, and to compare the performance, query execution performance and CPU usage were measured and compared.

In the query execution performance experiment, it was found that the JSON format was up to 46% faster than the XML format. In the experiment, it was shown that the search time was faster when the number of records was larger than when the number of records was small. When analyzing the results of the experiment, it was found that the JSON format is suitable for search as the amount of data increases. In particular, it was found that the speed increases especially when the amount of search results increases (Type1, 4). When all four queries were executed, the average CPU usage showed good results by about 10% or more in JSON search.

In this paper, we analyzed the search performance using JSON data format in a big data environment. The system developed in the study was used in the experiment to search up to 50,000 records. In the future, we plan to acquire more data and experiment to verify the search performance. Also, CPU usage and performance will be analyzed when performing queries by adding various databases.

5. REFERENCES

1. Salman S, Joshua ZH, Yulin H. Random Sample Partition: A Distributed Data Model for Big Data Analysis. IEEE Transactions on Industrial Informatics, 2019 Nov;15(11):5846-5854. DOI: 10.1109/TII.2019.2912723.
2. Rongheng L, Zezhou Y, Hao W, Budan W. Chronic Diseases and Health Monitoring Big Data: A Survey. IEEE Reviews in Biomedical Engineering, 2018 Apr;11:275-288. DOI: 10.1109/RBME.2018.2829704.
3. Peng Z, Xiang S, Samee UK. QuantCloud: Enabling Big Data Complex Event Processing for Quantitative Finance Through a Data-Driven Execution. IEEE Transactions on Big Data, 2019 Dec;5(4):564-575. DOI: 10.1109/TBDATA.2018.2847629.
4. Tamer ZE, Joshua ZH. Distributed Data Strategies to Support Large-Scale Data Analysis Across Geo-Distributed Data Centers. IEEE Access, 2020 Sep;8:178526-178538. DOI: 10.1109/ACCESS.2020.3027675.
5. Baeg J. Big Data Based Strawberry Growing Environment Decision Making System. Journal of Next-generation Convergence Technology Association, 2020 Aug;4(3):258-264. DOI:10.33097/JNCTA.2020.04.03.258.
6. Park JK, Kim J. Big data storage configuration and performance evaluation utilizing NDAS storage systems. AKCE International Journal of Graphs and Combinatorics, 2018 Aug;15(2):197-201. DOI: 10.1016/j.akcej.2017.09.003.
7. Yuanyuan Q, Yihang C, Jie Y, Jiajia L, Nei K. A Mobility Analytical Framework for Big Mobile Data in Densely Populated Area. IEEE Transactions on Vehicular Technology, 2017 Feb;66(2):1443-1455. DOI:10.1109/TVT.2016.2553182.
8. Jian L, Qiuru L, Lei Z, Shuhui S, Yongzhuang L. Enabling Massive XML-Based Biological Data Management in HBase. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2020 Nov;17(6):1994-2004. DOI:10.1109/TCBB.2019.2915811.
9. Sohail J, Kaleem RM, Mudassar A, Omar A, Muhammad A, Shehzad K, et al. A Methodology of Real-Time Data Fusion for Localized Big Data Analytics. IEEE Access, 2018, Apr;6: 24510-24520. DOI:10.1109/ACCESS.2018.2820176.
10. Xianghui W, Qian S, Jinlong L. JSON-LD Based Web API Semantic Annotation Considering Distributed Knowledge. IEEE Access, 2020 Oct; 8:197203-197221. DOI: 10.1109/ACCESS.2020.3034937.
11. Kim J. A Study on the JSON Compatible Serialization Standard Method of LPG Data. Journal of Next-generation Convergence Technology Association, 2020 Dec;4(6):581~588. DOI: 10.33097/JNCTA.2020.04.06.581.
12. Nico S, Sebastian M, JODA: A Vertically Scalable, Lightweight JSON Processor for Big Data Transformations. 2020 IEEE 36th International Conference on Data Engineering (ICDE), 2020 Apr:1726-1729. DOI:10.1109/ICDE48307.2020.00155.
13. Martin K, Alastair RB. A Conflict-Free Replicated JSON Datatype. IEEE Transactions on Parallel and Distributed Systems, 2017 Oct;10:2733-2746. DOI: 10.1109/TPDS.2017.2697382.
14. Chao L. Qingtian Z. Hua D, Nengfu X. A Mapping-Based Sharing Approach for Heterogeneous XML Data Supporting User Personalization. IEEE Access, 2019 Jun;7:76789-76805. DOI: 10.1109/ACCESS.2019.2922310.
15. DBLP dataset [Internet]. DBLP; 2021 [updated 2021 April 21; cited 2021 April 22]. Available from: http://dblp.uni-trier.de/xml/ (website)