Intermediate Data Format for the Elastic Data Conversion Framework

Tran Khanh Dang\textsuperscript{(c)}
\textit{Ho Chi Minh City University of Technology (HCMUT), VNU-HCM,}
268 Ly Thuong Kiet Street,
District 10, Ho Chi Minh City,
Vietnam
khanh@hcmut.edu.vn

Ta Manh Huy\textsuperscript{(c)}
\textit{Ho Chi Minh City University of Technology (HCMUT), VNU-HCM,}
268 Ly Thuong Kiet Street,
District 10, Ho Chi Minh City,
Vietnam
1870399@hcmut.edu.vn

Nguyen Le Hoang
\textit{Ho Chi Minh City University of Technology (HCMUT), VNU-HCM,}
268 Ly Thuong Kiet Street,
District 10, Ho Chi Minh City,
Vietnam
nlhoang@hcmut.edu.vn

Abstract—With the development of technology, data is becoming an extremely valuable resource. Data is being created, analyzed and used in a massive scale in every modern system. As a result, data analysis and data mining are very essential in each aspect of social applications. The value of data will be more useful if it can be linked and merged with other different data resources, especially for solving current social problems. In order to make a reality of this big challenge, data transformation is a crucial step that we have to overcome. Among data conversion methods, the intermediate method is to be a preferable one due to its expandable ability. However, for this type of model, finding the intermediate data type for the system is the crucial problem that needs to be solved firstly. In this paper, we will make comparisons among some popular data formats to find the appropriate data type for this elastic data conversion system.

Index Terms—data format, data type, data conversion, data integration system, data transformation, open data

I. INTRODUCTION

With the development of technology, data is becoming an extremely valuable resource. Data is being created, analyzed and used in a massive scale in every modern system. As a result, data analysis and data mining are very essential in each aspect of social applications. The value of data will be more useful if it can be linked and merged with other different data resources, especially for solving current social problems. In order to make a reality of this big challenge, data transformation is a crucial step that we have to overcome.

Data transformation can be described as a task that can flexibly convert data among different models and formats, thereby supporting the combination of data from various resources to a unified one, in another word, a unified dataset. This can bring many benefits to data analysis and management applications as well as can provide potential and optimal value such as in making predictions and supporting decision making.

Data now can be collected and integrated to store and manage in data centers for a variety of purposes.

However, this problem is not easy even when converting traditional data with few data sources in simple structure. Usually, this process requires the participation of human to understand and correct the meaning of the data in each source to solve the data ambiguity problems, including semantic and data representation ambiguity. The data conversion methods can be divided into two categories: direct conversion and intermediate conversion.

- Direct conversion: The data is converted directly from source to target format. This conversion method is the most popular conversion method due to its simplicity and ease of implementation. However, this method is only effective when the number of source and target formats is small. As this number increases, the complexity of the system increases rapidly (Fig.1).

- Intermediate conversion: Data will be converted to intermediate data format and this data will be converted to the format that the user wants. This method has the advantage that it will reduce the complexity of the system and make the system expandable (Fig.2).

Most studies and works mainly use the direct conversion model [5], [9]. The reason for the widely usage of this model is due to the natural factor of the goal or project: the need to convert data from one or several specific forms to one or several specific forms. However, with the goal of designing an extensible system having the framework can work with a variety of input and output formats, the direct transformation model creates a lot of complexity when adding more data types for the system. In the direct conversion model, if we add a new
TABLE I
RDBMS vs NoSQL

<table>
<thead>
<tr>
<th></th>
<th>Relational Databases</th>
<th>NoSQL Databases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal works</td>
<td>Designed for online transaction processing (OLTP) applications with high stability and suitable for online analytical processing (OLAP)</td>
<td>Designed for analysis of incomplete structured data and low-latency applications</td>
</tr>
<tr>
<td>Database model</td>
<td>The relational model normalizes data into tables of rows and columns. The diagram clearly specifies the relationship between tables and other database elements</td>
<td>NoSQL database provides diverse data models, including text, graph, key-value, etc.</td>
</tr>
<tr>
<td>ACID</td>
<td>Relational databases require Atomicity, Consistency, Isolation, Durability (ACID) properties</td>
<td>NoSQL databases often loosen some ACID attributes for a more flexible data model capable of horizontal scaling.</td>
</tr>
<tr>
<td>Performance</td>
<td>Depending on the system of the hard drive. Typically, optimizations will require query, index, and table structures.</td>
<td>Depending on hardware cluster size and network latency.</td>
</tr>
<tr>
<td>Expansion</td>
<td>Requiring to reconstruct tables, constraints and databases</td>
<td>High flexible and extensible ability</td>
</tr>
<tr>
<td>API</td>
<td>Data storage and retrieval requirements are communicated using Structured Query Language (SQL) queries. These queries are analyzed and executed by relational databases.</td>
<td>The object-based API allows application developers to easily store and retrieve structured data in memory. The fragment key looks for key-values, columns, or structured texts that contain the objects and properties arranged in series.</td>
</tr>
</tbody>
</table>

![Fig. 2. Intermediate Data Conversion](image)

input format to the system including \( n \) outputs of the system, it is required to manually programmatically extend \( n \) functions. However, if the intermediate conversion model is used, only one additional function is needed for the system. Therefore, the intermediate conversion model is the most suitable for the needs of a large and extensible data conversion system.

For this type of model, one of the most significant problems is finding the intermediate data type for the system. In this paper, we will make comparisons among some popular data formats to find the appropriate data type for the elastic data conversion system. This research direction is one of the research trends on Information Technology for the Ho Chi Minh City in the period of 2018-2023. The rest of this paper is organized as follows, we compare relational and NoSQL databases in section 2 to explain the reason of choosing NoSQL model; in section 3, we make some comparisons among some NoSQL data types to find the adaptable one. The summary and conclusion will be in section 4.

II. RELATIONAL AND NOSQL DATABASES

Since the 1960s, database management system is one of the core technologies in today’s information systems. During this development, there are many database models that are proposed based on different theoretical backgrounds; the most two common models are: relational databases RDBMS (such as Oracle, MySQL, SQL Server, etc.) and NoSQL databases (typically MongoDB, Postgre, Redis, etc.). According to DB-Engines [1], relational databases are the most popular with 74.5% of the market and NoSQL with nearly 20% (September 2020). The comparison between RDBMS and NoSQL is shown in Table I.

The intermediate data conversion system is designed to use an intermediate data type or standard data format in the transformation process. In fact, this standard format of data will also be stored in the storage block. For elastic and extensible needs and purposes, this standard data format must meet the following criteria: flexible structure, efficient storage and good scalability.

According to the above standards, NoSQL database is a better option since the RDBMS models will not meet the need for flexible structure.

III. COMPARISONS AMONG XML, JSON AND BSON

Among NoSQL databases, there are many standard data formats that can be used here; however, we only consider JSON, XML and BSON formats which can satisfy our needs for our data conversion framework in [16].

- **XML (Extensible Markup Language).** XML is one of the most commonly used data formats today. Designed with the aim of being easy to read, easy to understand for both humans and machines, XML has been widely used and practically become the communication standard.
- **JSON (Javascript Object Notation).** JSON was proposed in 2000, then was first synthesized and released as standard under the name ECMA-404 in 2013 [3]. First developed for the purpose of data exchange, JSON now is used for many different purposes: exchanging data, storing data, storing system settings, etc. Moreover, JSON has a very strong and growing community, especially on Github.
• BSON (Binary JSON). BSON is a binary representation of JSON. Designed by MongoDB based on the actual work for databases, BSON has many good properties such as closer to computer language and easier to find saved data. However, because the design purpose of BSON is to be applied to the MongoDB system, not many communities use and support this BSON format.

Choosing the a suitable format is an indispensable part of any system. The use of a suitable storage format affects not only system performance but also the structure (through the ecosystem of format support libraries), the maintainability (such as error detection capabilities) and the scalability (future system upgrades), etc. To find the suitable format, we mainly focus on these following properties:

• Size of the datasets when saving.
• The speed of loading and reading the datasets.
• Supported Libraries and Communities.

A. Comparison 1 - Size of datasets when saving

For a same dataset, the format that support smaller storage will be better since this can save a lot of resources and fasten the writing speed.

• Datasets: the data for comparisons are retrieved from www.data.gov as depicted in Table II. The chosen datasets are both available in XML and JSON formats.

• Method: each dataset will be converted to XML, JSON and BSON formats. Since the original datasets are already in XML and JSON formats, we only need to convert the datasets to BSON format from JSON format via bsonjs library in Python. The JSON-to-BSON conversion include loading the JSON-data, saving JSON-data to BSON data using bsonjs and then saving BSON-data to files. We then measure and compare the size of each dataset after converted.

• Results (Table III): XML data format has the biggest capacity while JSON has the smallest data storage. BSON datasets require just a little bigger than JSON.

B. Comparison 2 - Leading and Reading speed

For a same dataset, the format able to load and read data faster will be better. This factor is not related to above property; in fact, there are some formats able to compress the data to very small size but it takes long time to load and read.

• Datasets: the data for comparisons are retrieved from www.data.gov as depicted in Table II.

• Method: each converted dataset from comparison 1 will be stored in our system. Then it will be loaded and read. We measure and compare the process of loading and reading.

• Results (Table IV): XML takes the longest time to read and load data while JSON and BSON take nearly the same periods of time.
### TABLE IV Comparison Result 2

<table>
<thead>
<tr>
<th>Dataset</th>
<th>XML (s)</th>
<th>JSON (s)</th>
<th>BSON (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset 1</td>
<td>2.40</td>
<td>1.54</td>
<td>1.17</td>
</tr>
<tr>
<td>Dataset 2</td>
<td>11.03</td>
<td>4.11</td>
<td>5.19</td>
</tr>
<tr>
<td>Dataset 3</td>
<td>16.44</td>
<td>6.34</td>
<td>6.31</td>
</tr>
<tr>
<td>Dataset 4</td>
<td>0.56</td>
<td>0.34</td>
<td>0.29</td>
</tr>
<tr>
<td>Dataset 5</td>
<td>161.21</td>
<td>95.52</td>
<td>93.51</td>
</tr>
<tr>
<td>Dataset 6</td>
<td>849.02</td>
<td>345.41</td>
<td>346.42</td>
</tr>
<tr>
<td>Dataset 7</td>
<td>0.10</td>
<td>0.04</td>
<td>0.03</td>
</tr>
<tr>
<td>Dataset 8</td>
<td>106.81</td>
<td>15.84</td>
<td>18.65</td>
</tr>
<tr>
<td>Dataset 9</td>
<td>7.08</td>
<td>3.61</td>
<td>2.13</td>
</tr>
<tr>
<td>Dataset 10</td>
<td>42.38</td>
<td>21.54</td>
<td>24.34</td>
</tr>
<tr>
<td>Dataset 11</td>
<td>0.08</td>
<td>0.06</td>
<td>0.06</td>
</tr>
<tr>
<td>Dataset 12</td>
<td>4.49</td>
<td>0.46</td>
<td>0.21</td>
</tr>
<tr>
<td>Dataset 13</td>
<td>58.65</td>
<td>20.71</td>
<td>21.02</td>
</tr>
<tr>
<td>Dataset 14</td>
<td>2.79</td>
<td>0.80</td>
<td>1.19</td>
</tr>
<tr>
<td>Dataset 15</td>
<td>3.78</td>
<td>2.40</td>
<td>1.90</td>
</tr>
<tr>
<td>Dataset 16</td>
<td>172.70</td>
<td>72.12</td>
<td>71.81</td>
</tr>
<tr>
<td>Dataset 17</td>
<td>0.78</td>
<td>0.58</td>
<td>0.61</td>
</tr>
<tr>
<td>Dataset 18</td>
<td>0.58</td>
<td>0.43</td>
<td>0.38</td>
</tr>
<tr>
<td>Dataset 19</td>
<td>51.74</td>
<td>8.71</td>
<td>15.56</td>
</tr>
<tr>
<td>Dataset 20</td>
<td>3.11</td>
<td>0.12</td>
<td>0.05</td>
</tr>
</tbody>
</table>

C. Comparison 3 - Supported Libraries and Communities

The built-in libraries and the support from communities are also a very important aspect to consider when choosing a suitable data type. As a rule of thumb, the more popular the format is, the better option it is. There are many ways to compare the popularity of the aforementioned formats; however, in this paper, we will use the following criteria since they can give us some insights about the use of these formats in the developer community.

- Number of related repositories on GitHub.
- Number of questions on Stack Overflow.
- Data streaming supported.

Moreover, other than the above criteria, we also include some other ones which will provide a deeper understanding of these formats. These include:

- The format this format is based on.
- The formats based on this format.
- The standardization of the format.
- The standard APIs of the format.
- Notable libraries of the format.

It is always difficult to build anything from scratch; the more supported we have from existing libraries and communities, the more effectively and efficiently we can operate our system. The comparison in Table V has shown that:

- More people are using JSON than XML and BSON.
- JSON also has more supported libraries than others.
- BSON format is not yet matured and has limited support from the community.
- Both JSON and XML are full supported in data streaming.

IV. Conclusions

From the comparisons, we can conclude that both XML and JSON have the maturity in technology with good characteristics such as high scalability, flexibility and convertibility. However, we think that JSON format has a bigger advantage over the other two format and more suitable for our framework in [16] due to these reasons:

- Smaller storage capacity.
- Faster loading and reading process.
- Stronger communities of users and developers.

Recent studies have shown the important role of data conversion and its usefulness in data integration systems, especially the intermediate data conversion model which supported the diversity of input data sources and data formats or the challenging context of big data. To make this system performing effectively, finding the most appropriate intermediate data type is an essential problem and needs to be solved firstly.

In this paper, we make comparisons among some popular and mostly used data formats then choosing the suitable data type for this data conversion system. Even though relational databases are the most traditional and widely-used, NoSQL models will help us the attributes of flexibility and scalability. And among NoSQL data types, JSON seems to be the best option for the system due to its smaller storage capacity, faster loading/reading and bigger supported communities.

ACKNOWLEDGMENT

This work is supported by a project with the Department of Science and Technology, Ho Chi Minh City, Vietnam (contract with HCMUT No. 42/2019/HĐ-QPTKHCN, dated 11/7/2019).

REFERENCES

<table>
<thead>
<tr>
<th></th>
<th>XML</th>
<th>JSON</th>
<th>BSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of related repositories on GitHub</td>
<td>About 95,000 (October, 2020)</td>
<td>About 240,000 (October, 2020)</td>
<td>About 900 (October, 2020)</td>
</tr>
<tr>
<td>Number of questions on Stack Overflow</td>
<td>About 0.6% of questions on Stack Overflow in 2019 [2]</td>
<td>About 1.5% of questions on Stack Overflow in 2019 [2]</td>
<td>No data available</td>
</tr>
<tr>
<td>Data streaming supported</td>
<td>Full supported such as XMPP, QuiXSchematron, XLTL 3.0</td>
<td>Full supported such as Jackson (API), jq, logstash, ldjson-stream</td>
<td>Few</td>
</tr>
<tr>
<td>The format this format is based on</td>
<td>SGML</td>
<td>None (But based on JavaScript syntax)</td>
<td>JSON</td>
</tr>
<tr>
<td>The formats based on this format</td>
<td>YAML, Fast Infoset, SOAP, XML-RPC, Effi-cient XML Interchange (EXI)</td>
<td>YAML, BSON, Ion, Smile, CBOR, MessagePack, Extensible Data Notation (EDN)</td>
<td>None</td>
</tr>
<tr>
<td>The standardization of the format</td>
<td>Standardized</td>
<td>Standardized</td>
<td>Not Standardized</td>
</tr>
<tr>
<td>The standard APIs of the format</td>
<td>DOM, SAX, XQuery, XPath</td>
<td>Clarinet, JSONQuery, JSONPath, JSON-LD</td>
<td>No Standard APIs</td>
</tr>
<tr>
<td>Notable libraries of the format</td>
<td>JSON_checker, YAJL, json-c, json-parser, JSONKit, JSONUtil, json2.js</td>
<td>DOM4J, StAX, JDOM, xml2js, libxmljs, sax, fast-xml-parser, xml-stream</td>
<td>libbson, Mongo-GLib, bson4jackson, CookJson, PyMongo</td>
</tr>
</tbody>
</table>