Journal of Telematics and Informatics (JTI)
Vol.7, No.3, September 2019, pp. 166~173
ISSN: 2303-3703

JSON Implementation with Zlib Compression for Database Efficiency in Handling Dynamic Upload Multiple Images

[1]Rianto, [2]Alam Rahmatulloh, [3]Iqbal Muhammad Fajar Nuralam,
Departement of Informatics, Universitas Siliwangi
Jalan Siliwangi No. 24 Kota Tasikmalaya, (0265) 330634
e-mail : [1]rianto@unsil.ac.id; [2]alam@unsil.ac.id; [3]iqbal.fajar.nuralam14@student.unsil.ac.id

Abstract

In this era, in an application it is often found that a dynamic input form is a form that can be filled in a lot of data, the application user is free to fill the data according to his wishes. To handle this dynamic input form, the developer usually adds a new table in a database specifically for storing dynamic data, it has the potential to waste tables and records in a database. Applying the JSON conversion technique is a solution to overcome this, dynamic data is stored in a special field so that the use of tables and records can be minimized, as well as to compress the JSON string length applied by the Zlib algorithm. In this study, dynamic data is a images file uploaded on an input form. The results of this study in the case of handling multiple dynamic form upload images shows that the JSON conversion technique is better than conventional techniques in terms of faster data storage speeds of 72.6%, in terms of simpler database structures, and in terms of 22.7% more data size small, so database management becomes more efficient.

Keywords *: Database, Data Donamics, JSON, Upload, Zlib*

1. Introduction

Today the database is a very important need for a company or government agency. Databases are commonly used to manage internal records or records, present data to corporate consumers on the internet, and are widely used to support other commercial processes [1]. Over time, technological developments in application development are increasing rapidly. In web-based application technology, for example, starting from web 1.0 (static web), web 2.0 (dynamic web), and now entering the era of web 3.0 (semantic web) [2].

Often found in an application there is a dynamic input form which is a form that can be filled with a lot of data, the application user is free to fill the data according to his wishes. To handle this dynamic input form, developers usually add special tables in the database to store dynamic data. This has the potential to waste tables and records, dynamic data is stored by utilizing foreign keys from other tables [3]. In addition to wasting the number of tables and records, this method causes the database file size to be larger and the data access speed will be longer. Therefore, it is very important to do efficiency in the database, because the database is the heart of the information system. Data must be available when the user wants to use, data must also be accurate and consistent. Apart from these requirements, the purpose of database design is the efficiency of data storage and the efficiency of reading and updating data [4].

JSON has a higher level of parsing and is easier to implement than XML [5] and the JSON flexibility is superior to XML [6]. In addition JSON is superior to XML in terms of execution time and CPU usage [7]. Therefore, based on these data, research is carried out to overcome inefficient database management in dynamic data storage by utilizing JSON. JSON is an object that was originally designed and developed with a lightweight, text-based data exchange format, the independent language of data exchange comes from the ECMAScript (javascript) literal object in standard language programming [8]. JSON objects are analyzed as string arrays, with higher parsing, efficiency and an easier format than transportation formats such as XML. JSON is made of two data structures namely a collection of name-value pairs and an ordered list of values and has a data format that can be exchanged with the built-in data structure in the programming language and reduces the complexity and processing time [5].

Zlib is one of the algorithms for compressing data. Using the Zlib algorithm aims to compress the JSON string length in the database. The use of this Zlib Algorithm is because it is

lossless, where compression data compression techniques can be decompressed again and the results are the same as the data before the compression process, so the file transfer process will be maximized and the stored string length can be minimized [9].

Database Efficiency

A database is a collection of data that is logically related and describes the integration between a table and other tables, which are designed to meet the information needs of an organization [10]. The purpose of database design is the efficiency of data storage and the efficiency of reading and updating data [4].

JSON

JSON (JavaScript Object Notation) is a data exchange format that is lightweight, easy to read and written by humans, and easily translated and generated by a computer. This format is based on part of the JavaScript Programming Language, ECMA-262 Standard Edition 3rd - December 1999. JSON is a text format that does not depend on any programming language because it uses the style of language commonly used by C family programmers including C, C ++, C # , Java, JavaScript, Perl, Python etc. Because of these properties, make JSON ideal as a data exchange language [8].

Image

Image (digital image) is a discrete image that can be processed by a computer. This image can be generated through a digital camera and scanner or images that have undergone a digitization process [11].

Algoritma *Zlib*

The Zlib algorithm is a derivative of the Deflate compression algorithm. This algorithm was created by Jean-Loup Gailly who created the data compression process and Mark Adler that created the data decompression process. The Zlib algorithm performs compression by compressing data consisting of a series of blocks, according to the data input block. Each block in the data is compressed using the Deflate data compression algorithm as a compressor which is a variation of the LZ77 algorithm combined with Huffman Coding [12].

Related Work

The researcher [5] in his research explained that JSON is a lightweight data exchange format, where JSON makes time efficient for data translation, reduces complexity, and data processing time. Explaining that JSON has a higher parsing efficiency, and in conclusion the researcher recommends encryption by using the JSON format as an alternative to encryption using the XML format. The researcher [6] analyzes the data exchange format comparison on JSON and XML. The result in terms of the performance of the data exchange format in the form of JSON is better between the two and in the data parsing speed and JSON flexibility is superior to XML. Even when compression is enabled, XML generates more overhead on the data stream compared to JSON.

The researcher [7] in his research shows that JSON is the best choice for storage and query speed. While XML and JSON technology are still relatively new to date compared to conventional databases. JSON technology does show a greater potential for database technology to handle huge data due to increased data every year. Results of previous research shows that JSON is better than XML, so it becomes a reference for using JSON as a medium for handling dynamic data in the case

2. Research Method

The method proposed in this study can be seen in Figure 1.

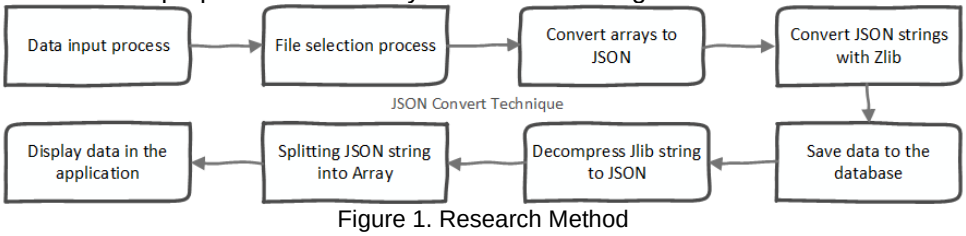

Figure 1. Research Method

Data / File Input Process

Data / file input is the first step to implement the JSON conversion technique in handling dynamic upload multiple images. In the application prototype that has implemented this JSON conversion the user is asked to enter several input forms including name, description, and image files to be uploaded. Users can upload images files dynamically as many times as users want, for illustrations can be seen in Figure 2.

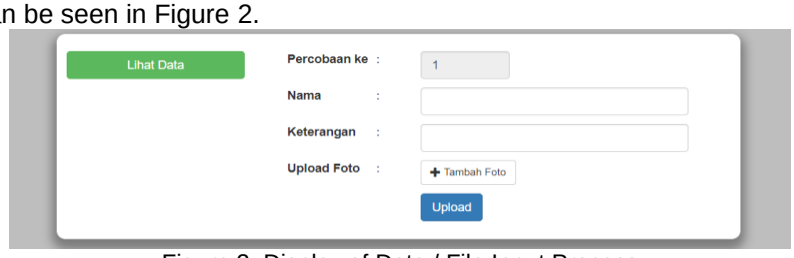

Figure 2. Display of Data / File Input Process

If the choose file button is clicked, a file explorer pop-up window will appear on Windows, for an illustration, see Figure 3.

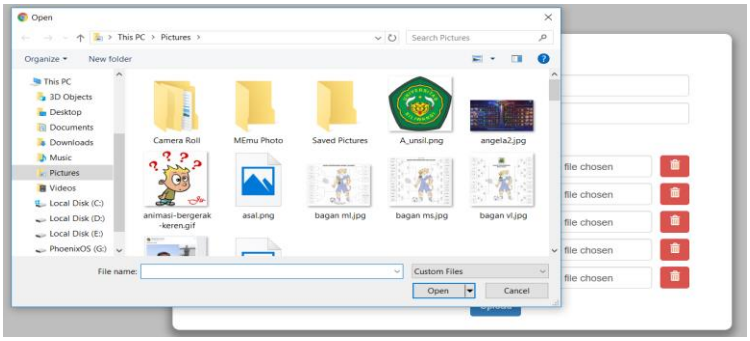

Figure 3. Pop-Up File Explorer window on Windows

File Selection Process

This stage is done to filter what types of files are allowed to be uploaded. In this study the types of files that are allowed are those included in the image category including * .jpeg / *. Jpg, * .png, and * .gif. This file selection process is not done based on file extensions but based on the original file type to prevent the ImageTragick attack, which is a file that contains a special script, and the file is saved with an extension image. The way to attack is to upload the file to a form that has the file upload feature.

The file selection process stage is carried out in two stages: the front-end and the back-end. The first stage is done on the front-end as shown in Figure 3, the file selection on the front-end is done when the user will select the images file to be uploaded, the file explorer windows pop-up window only raises the file types allowed to upload.

The second stage is done on the back-end, which is when processing files after the user selects the file to be uploaded. As an illustration when the file selection process shows that the uploaded file does not include the permissible file type shown in Figure 4, an illustration shows that the file uploaded is one that is allowed for the file type shown in Figure 5.

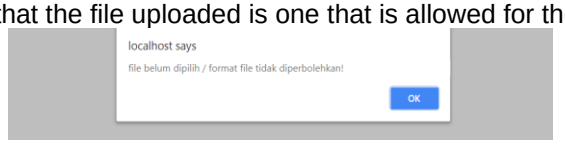

Figure 4. *Upload Unauthorized File Types*

Figure 5. *Upload Allowable File Types*

Convert Array to JSON

The image file uploaded is actually processed in the form of an array, if it is stored in the database it must be repeated several times with the aim of each array element being broken back into an ordinary string and can be read in the database. As a result the storage process to the database is carried out many times as much as the repetition is done, besides the number of files uploaded also affects the number of records stored. For that purpose, the conversion from array to JSON form with the aim of the row of data stored in the array is converted into one string line.

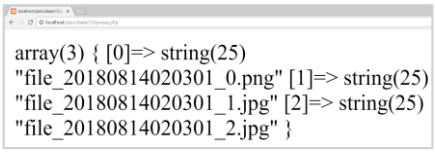

Figure 6. Files in Array type

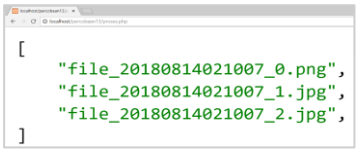

Figure 7. Files in JSON type

Figure 6 shows the file upload process, when the user selects 3 images files to upload and shows the results of the file upload process that is still in array form. And after being converted into JSON, the result is shown in Figure 7. The process of converting from array to JSON is done by using a json_enccode function in the PHP programming language which functions to convert the array to JSON.

JSON String Compression with Zlib

Data that has been converted to JSON will be compressed with the aim of the length of the JSON string to be trimmed. The JSON string compression process with this Zlib algorithm uses the zlib_encode () function with the Deflate algorithm as its compressor, the function has become a library in the PHP programming language. The JSON String compression result with the Zlib algorithm is shown in Figure 8, the compression string will be unique characters and the length of the string will be shorter.

x��VJ��I�720�0�40702�46�7�+�KW�□v�□�

Figure 8. JSON String Compression Results with Zlib Algorithm

Save the Zlib String to the Database

The JSON string that has been compressed into a Zlib string is then saved to the database.

id	nama	caption	file
1	tes	tes	x??VJËÌI?720´0°4070262±?7Ð+ÈKWÒÁ*g?G,▯▯ Ãi▯P

Figure 9. Save the Zlib String to the Database

Figure 9 shows that the Zlib string that is compressed from the JSON string is successfully saved to the database. By compressing the JSON string, it will further trim the length of the string stored in the database. When using conventional handling techniques the data storage process is carried out 3 times, and produces 3 records. By applying the JSON conversion technique, the storage process is only done once because 3 uploaded files are converted into one JSON string line and compressed with the Zlib algorithm so that it can save the use of tables, number of records, and time needed to store data to the database.

Decompress the Zlib String to JSON

To return the compression string with the Zlib algorithm to JSON form, the decompression process is performed. The decompression process is carried out using the Zlib library in the PHP programming language by calling the zlib_decode () function.

Split the JSON String into an Array

The compressed JSON string is split back into array form using the json_decode syntax in the PHP programming language.

Display Data in Application

The last step is to display the data converted from the JSON string to the array in the application. To be able to split each array element so that it can be read by the system, the repetition process is performed using the foreach function in the PHP programming language, the foreach function is a special repetition for reading array values.

3. Result and Analysis

Testing

Testing is done by comparing the application of dynamic upload multiple images by using the JSON conversion technique and dynamic upload multiple images application that uses conventional handling techniques. The parameters used for this test are in terms of data storage speed, database structure, and data size.

Data Storage Speed Testing

The test was carried out as many as 10 types of experiments, where each type of experiment was carried out 5 times in applications that applied the JSON conversion technique and in conventional applications. Table 1 describes the test history in the application that applies the JSON conversion technique and Table 2 describes the testing history in conventional applications.

Table 1. Testing JSON version data storage speed

No	Jenis Pengujian	Pengujian Ke -					Rata – Rata (detik)
		1 (detik)	2 (detik)	3 (detik)	4 (detik)	5 (detik)	
1	Upload 1 File	0,0792	0,0544	0,0864	0,1110	0,0491	0,0760
2	Upload 2 File	0,0861	0,0724	0,1010	0,0919	0,0719	0,0847
3	Upload 3 File	0,1200	0,0849	0,0847	0,0659	0,1236	0,0958
4	Upload 4 File	0,1103	0,0757	0,0967	0,1427	0,0849	0,1021
5	Upload 5 File	0,0951	0,0849	0,2701	0,0407	0,1170	0,1216
6	Upload 6 File	0,1046	0,0937	0,0836	0,0606	0,1216	0,0928
7	Upload 7 File	0,0811	0,0593	0,0797	0,0783	0,0853	0,0767
8	Upload 8 File	0,0770	0,0693	0,0673	0,0583	0,0653	0,0674
9	Upload 9 File	0,0746	0,0908	0,0906	0,0986	0,0975	0,0904
10	Upload 10 File	0,1261	0,0600	0,0702	0,0682	0,0663	0,0782

Table 2. Testing conventional data storage speed

No	Jenis Pengujian	Pengujian Ke -					Rata – Rata (detik)
		1 (detik)	2 (detik)	3 (detik)	4 (detik)	5 (detik)	
1	Upload 1 File	0,1201	0,2202	0,1225	0,2465	0,1066	0,1632
2	Upload 2 File	0,3271	0,1664	0,1941	0,1218	0,2697	0,2158
3	Upload 3 File	0,3353	0,2222	0,1732	0,2701	0,2652	0,2532
4	Upload 4 File	0,2818	0,2825	0,3101	0,3037	0,213	0,2782
5	Upload 5 File	0,3049	0,273	0,3013	0,2603	0,2701	0,2819
6	Upload 6 File	0,2953	0,3045	0,3013	0,3337	0,3206	0,3111
7	Upload 7 File	0,3144	0,3166	0,3379	0,3258	0,3181	0,3226
8	Upload 8 File	0,2799	0,3896	0,2599	0,5152	0,3533	0,3596
9	Upload 9 File	0,5197	0,426	0,4076	0,6162	0,6147	0,5168
10	Upload 10 File	0,4539	0,4323	0,5619	0,6106	0,6197	0,5357

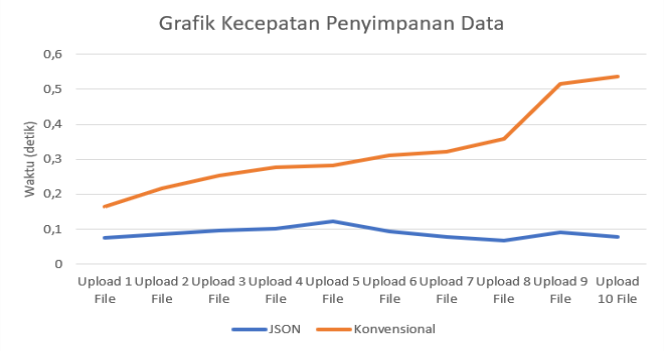

Figure 10. Comparison of data storage speed graphs

Figure 10 shows a graph of data storage speed in dynamic upload multiple images applications using the JSON conversion technique and dynamic upload multiple images application using conventional handling techniques.

Database Structure Testing

This test is done by comparing the database structure used in dynamic upload multiple images applications by using JSON conversion handling techniques and multiple dynamic images upload applications that use conventional handling techniques.

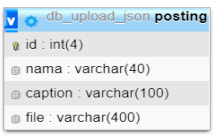

Figure 11. JSON version database structure

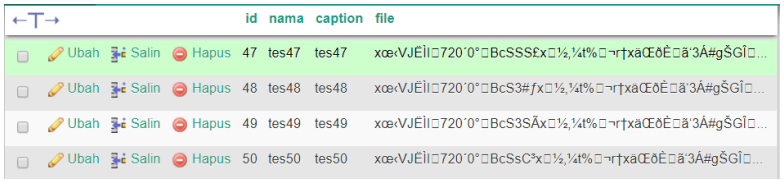

Figure 12. Number of post table records JSON version

Figure 11 shows the database structure for dynamic upload multiple images applications using the JSON conversion handling technique. Database with the name db_upload_json only requires 1 table with the name of the post because dynamic image file data in the form of a JSON string can be stored in a field named file. In testing the data storage speed, the data is inputted 50 times, then the record stored in the database is 50 records, shown in Figure 12.

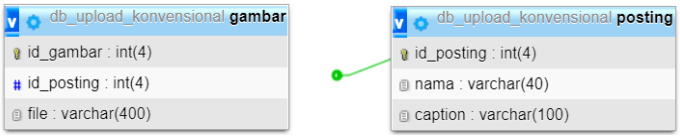

Figure 13. Conventional version database structure

Figure 13 shows the database structure in dynamic upload multiple images applications with conventional handling techniques. The database is named db_upload_conventional, to be able to store dynamic image file data on conventional handling techniques it takes 1 special table. The first table is a post table whose function is to accommodate non-dynamic data such as names and captions, while the second table is an image table whose function is to accommodate dynamic image file data. Figure 14 shows the number of records in the post table and Figure 15 shows the number of records in the picture table.

Figure 14. Number of conventional post table records

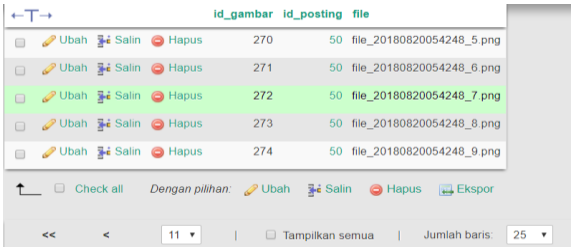

Figure 15. Number of conventional image table records

Large Data Size Test

This test is done by comparing the size of the data in the application database dynamic upload multiple images by using JSON conversion handling techniques with applications with conventional handling techniques. The number of uploaded image files is 1000 files with the number of characters in the file name and the same file type where every 1 time the data input is uploaded as many as 10 files. Table 3 shows each database.

Table 3. Large size of data in the database

No	Database Name	Size (bytes)	Size on Disk (bytes
1	db_upload_json	35.351	36.864
2	db_upload_konvensional	45.731	49.152

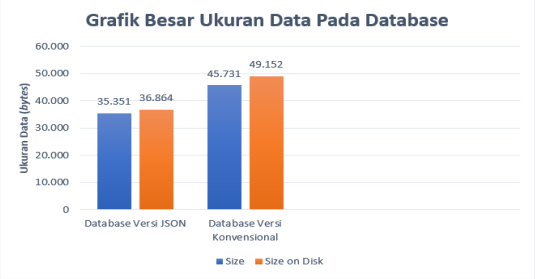

Figure 16. Graph comparison of the size of the data in the database

Figure 16 shows a large graph of the size of the data in the database used in dynamic upload multiple images applications by using JSON conversion handling techniques and databases in applications with conventional handling techniques.

4. Conclusion

The JSON conversion technique can handle form dynamic upload multiple images to be more efficient, and shows the JSON function not only as a data exchange format. In testing in terms of speed of data storage results in applications that use JSON conversion handling techniques the data storage speed is 72.6% faster than applications that use conventional handling techniques.

In the tests carried out by comparing the database structure in applications that use JSON conversion handling techniques only need 1 table and to accommodate dynamic image file data simply stored in a special field to accommodate the data so that the database structure becomes simpler and the number of records stored less and less. In the tests carried out by comparing the size of the data in the database, the results of applications that use JSON conversion handling techniques have a data size of 22.7% smaller than applications that use conventional handling techniques.

Suggestions for further research are to recompile the JSON conversion technique with a special library to handle dynamic input forms, using other test parameters with the hope that the results of the compilation will be more accurate.

References

[1] Molina, Ulman dan Widom, Database Systems : The Complete Book, New Jersey: Prentice Hall, 2001.

[2] M. R. Arief, Pemrograman Web Dinamis menggunakan PHP dan MySQL, Yogyakarta: C.V ANDI OFFSET, 2011.

[3] M. A. Rosid, "Implementasi JSON untuk Minimasi Penggunaan Jumlah Kolom Suatu Tabel Pada Database PostgreSQL," *Journal Of Informatics, Network, and Computer Science,* vol. 1, pp. 33-42, 2016.

[4] A. Noertjahyana, S. Rostianingsih dan A. Handojo, "Pengaruh Desain Terhadap Penerapan Efektifitas Database Melalui Beberapa Contoh Kasus," *Jurnal Informatika,* vol. 6, pp. 1-6, 2015.

[5] A. El-Aziz dan A. Kannan, "JSON Encryption," *International Conference on Computer Communication and Informatics (ICCCI -2014),* pp. 1-6, 2014.

[6] S. Zunke dan V. D'Souza, "JSON vs XML: A Comparative Performance Analysis of Data Exchange Formats," *IJCSN International Journal of Computer Science and Network,* vol. 3, no. 4, pp. 257-261, 2014.

[7] M. K. Yusof dan M. Man, "Efficiency of JSON approach for Data Extraction and Query Retrieval," *Indonesian Journal of Electrical Engineering and Computer Science Vol. 4, No. 1,* pp. 203-214, 2016.

[8] JSON, "Pengenalan JSON," 2 Juni 2018. [Online]. Available: https://www.json.org/json-id.html. [Diakses 2 Juni 2018].

[9] P. Deutsch dan J.-L. Gailly, ZLIB Compressed Data Format Specification version 3.3, NetworkWorking Group, 1996.

[10] T. Connoly dan C. Begg, Database Systems : A Practical Approach to Design, Implementation, and Management, University of the west of Scotland: Addison Wesley, 2002.

[11] P. Fahzuanta, “Analisis Perbandingan Pendeteksi Garis Tepi pada Citra Digital Antara Metode Edge Linking dan Operator Sobe,” *Universitas Sumatera Utara,* 2010.

[12] ZLIB, “Zlib Algorithm,” 15 December 2017. [Online]. Available: http://www.zlib.net. [Diakses 7 September 2018].