Optimizing Data Storage in Handling Dynamic Input Fields with JSON String Compression

1st Irfan Darmawan
Department of Information System
Telkom University
Bandung, Indonesia
irfandarmawan@telkomuniversity.ac.id

2nd Alam Rahmatulloh
Departement of Informatics
Siliwangi University
Tasikmalaya, Indonesia
alam@unsil.ac.id

3rd Iqbal Muhammad Fajar Nuralam
Departement of Informatics
Siliwangi University
Tasikmalaya, Indonesia
iqbalmfn@gmail.ac.id

4th Rianto
Departement of Informatics
Siliwangi University
Tasikmalaya, Indonesia
rianto@unsil.ac.id

5th Rohmat Gunawan
Departement of Informatics
Siliwangi University
Tasikmalaya, Indonesia
rohmatgunawan@unsil.ac.id

***Abstract*—Dynamic input fields are a solution for managing multiple input values in a web-based application form. Dynamic multiple image upload is an implementation of the dynamic input field. Handling dynamic upload of multiple images by storing the image path will cause the existence of similar string data in one field in the table stored in the database. Creating a unique table in a database to store dynamic data is a workable solution. However, it is potentially a waste of tables and records, so that the database file size becomes larger and data access speeds are longer. To overcome this problem in this study, string data obtained from the dynamic input field are converted into JSON format and compressed with Zlib, before being saved into the database. The experimental results in this study indicate that the integration of JSON and Zlib can be applied to the handling of dynamic input field forms. The average speed of the data storage process by applying this technique is 50.36% faster than the conventional method. In comparison, the database file size decreased by about 37.58% smaller than using conventional techniques.**

***Keywords*— Dynamic Upload, JSON, Web, Zlib**

I. Introduction

The Web 2.0 paradigm is significant momentum in the last decade. The implementation of Web 2.0 technology has triggered an explosion of information and a spike in web-based media content. The adoption of individuals and companies from these technologies continues to increase [1], [2]. The complexity of web-based applications that continue to develop requires developers to continue to improve the quality of their products [3].

Modern web applications are sophisticated interactive programs with complex dynamic web-based user interfaces [3]. Input form fields are one element that is often found on web pages. The dynamic input field feature is a solution for handling multiple input values in a form [4]. This is very useful when you want to receive multiple inputs in the same field in a web page. The dynamic input field feature can be easily integrated using jQuery.

The file upload feature on web pages is one of the essential aspects of any project. Given the importance of this, developers are required to add file upload features to their projects. The feature of uploading multiple images is an implementation of the dynamic form field. Uploading many images at once can reduce the time to upload large numbers of images to the server, so there is no need to upload one at a time [5], [6]. The upload functionality of many images contained in a form is also handy for the dynamic gallery management section of web applications. Image files that have been uploaded via the form field input will then be stored in the database. In a database, image files can be stored using Binary Large Object (BLOB) data types [7], [8], or in the form of a path. Saving images with BLOB data types is done by storing the contents of these images in binary form into a table in the database [9]. While saving by the path method, the contents of the image are not stored in the table but only the path. In contrast, the uploaded image file is usually stored in a specific folder according to the conFiguration done before.

The dynamic image multiple upload features that are handled by storing data in the form of paths will cause a string of data from several similar paths in one field in the database table. Creating a unique table in a database to store dynamic data is one solution that can be done. Dynamic data is stored by using foreign keys from other tables. However, this has the potential for a waste of tables and records [10]. In addition to wasting the number of tables and records, this method results in a larger database file size and longer data access speeds. Other techniques to overcome this problem are delivered by formatting the data into a unique structure and then storing it in one column so that no unique tables are needed to store dynamic data. This method has been tried in research [10]. His research has succeeded in applying the json_encode function to convert array type data to JavaScript Object Notation (JSON) data format and store it in a field.

The conversion of arrays into JSON format is done because the JSON data format is more suitable for dynamic web applications and simple data transmission. JSON performance speed is greater than (eXtensible Markup Language (XML) because of its simple structure and easy access to data [11]. Besides, JSON is more effective in data size and response time of the Application Programming Interface (API) web than the XML data format [12], [13]. Programmers use JSON extensively to connect two servers that communicate via Web services, and in many other similar scenarios [14].

Storing data in JSON format into a field can be optimized by first performing the compression process so that the stored string is shorter. Therefore in this study, we will try to compress data from JSON format before it is stored in a field. The experiment will be applied to handle string data in dynamic upload multiple images on a web page. Zlib was chosen in this study because it can transparently read and write gzip-compressed files. With this compression, it is expected that the stored string length can be minimized and storage usage can be saved.

II. Related Work

Research conducted by [10], try using the json_encode function to convert data from an array type to a JSON data format and store it in a field in a database table. Handling dynamic data that previously had to be stored in two tables in a database, with the application of this method only needed one table. Data converted into JSON format only requires one field in storage. His research has successfully dealt with dynamic data storage by applying these techniques.

Other research conducted by [11], compare code, data model, accessing and extracting JSON, and XML formats. The results of his research revealed that JSON is more suitable for dynamic web applications and simple data transmission. JSON performance speed is greater than XML because of its simple structure and easy access to data. JSON is suitable for data exchange, whereas XML is useful for transmitting data between servers and web applications.

On research [12], the data sending method is compared using JSON and XML format. Data size and response time were measured in an experiment in this study. The experimental results in this study show that JSON is more effective in data size and web API response times than XML data formats.

Handling data strings on multiple image dynamic uploads on a web page by compressing data from the JSON format and compression with Zlib before being stored in a field is the main focus of this research. Application prototypes are made of two types. The first is an image upload application conventionally, while the second is by applying the concept of JSON and the use of Zlib. Experiments carried out by uploading images with varying amounts through applications that have been developed. Every upload image experiment that is carried out is calculated the resulting process time. The experimental results data are then inputted into a table and presented in graphical form. In the final stage, conclusions are drawn from the results of experiments that have been carried out.

III. Experimental Design

There are three stages carried out in this study, namely: the development of multiple image upload application prototypes, application prototype testing, analysis of test results.

A. Development of multiple image upload application prototype

This research will design a prototype of a web-based application that provides multiple image upload features. When uploading images, image content is not stored in the table but only the path. In contrast, the uploaded image file is stored in a particular folder that has been previously configured. So in this experiment, uploading image files in the BLOB data type was not done into the database table.

There are two types of applications that will be developed in this study. First, application with conventional techniques, second application by implementing JSON and Zlib. In applications in a conventional manner and the application of JSON Compression. In the first application, handling multiple image uploads is done by storing the uploaded image information into a string with an array data type into a database table. Two tables are needed to store information for each uploaded image. Whereas in the second application string with data array type that stores information, every uploaded image is converted into a JSON data format, then compressed using the Zlib library. After that, it is saved into a database table. In this second application, only one table is needed to store information for each uploaded image. The process flow of the prototype application developed by implementing JSON and Zlib is shown in Fig 1.

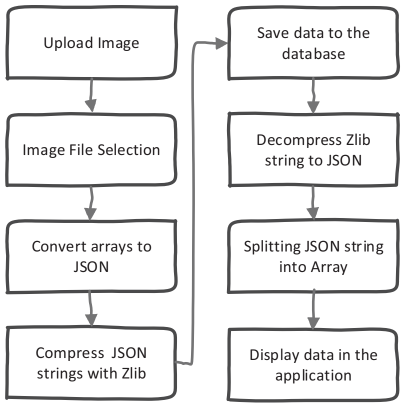

Upload Image
Image File Selection
Convert arrays to JSON
Compress JSON strings with Zlib
Save data to the database
Decompress Zlib string to JSON
Splitting JSON string into Array
Display data in the application

Fig. 1. JSON Conversion and Zlib Compression Proposed Method

In Fig 1, the flow of the process performed on the developed application is displayed. The stages of the process are as follows:

1) *Upload Image:* Uploading the image is the first step carried out in the application. The application is designed so that users can upload images with varying amounts.

2) *Image File Selection:* At this stage, a selection of uploaded image file types is made. Images that can be uploaded are limited to only types: * .jpg, * .jpeg, * .png, * .gif.

3) *Convert Arrays to JSON:* At this stage, a dynamic data string (image file) is converted to an array into the JSON format using the json_encode () function.

4) *Compress JSON String with Zlib:* Data that has been converted to JSON format is compressed using the zlib_encode () function.

5) *Save Data to the Database:* At this stage, the compressed JSON string is stored in the database.

6) *Decompress Zlib String to JSON:* At this stage, decompressed strings are compressed into JSON format. The decompression process is done using the zlib_decode () function.

7) *Splitting JSON String to Array:* At this stage, splitting the JSON string into an array using the json_decode () function so that it can be read by the system.

8) *Display data in the application:* At this stage, the JSON string conversion result data is displayed to the array in the application. To be able to break up every element of the array so that it can be read by the system, the repetition process uses the foreach () function.

B. Testing the application prototype

1) *Data preparation;* The data used as experiments in the form of image files * .jpg with a size of 100 KB can be downloaded from https://file-

examples.com/index.php/sample-images-download/sample-jpg-download/. The downloaded image file is then copied so that there are 20 similar image files with different names.

2) Experimental design: Experiments carried out by uploading images through applications that have been developed with a number of 1 to 20.

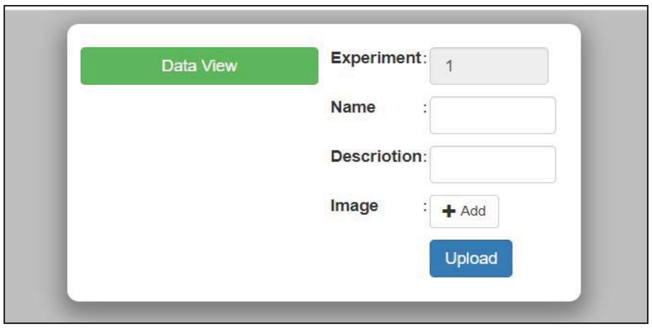
3) Measurement: Every upload image experiment that is carried out is calculated the resulting process time. The experimental results data are then inputted into a table and presented in graphical form.

4) Analysis of test results: At this stage, an analysis of the test result data is made, then conclusions are drawn from the results of the overall experiment that has been conducted.

IV. Result And Analysis

A. Implementation

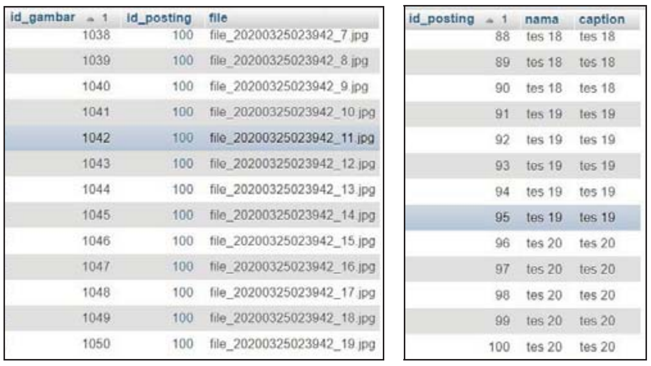
At this stage, web-based applications are developed that provide multiple image upload features, as shown in Fig 2.



Data View	Experiment:	1
	Name :	
	Descriotion:	
	Image :	+ Add
		Upload

Fig. 2. Display multiple upload image menu

Fig 2 shows an application that provides a dynamic multiple upload image feature. To be able to increase the number of images uploaded in one process, the user can press the "Add" button. The number of images that can be uploaded varies depending on the user. Image information that has been uploaded is stored in a table in a database. In applications that use conventional techniques, two tables are needed to store image information uploaded by the user, as shown in Fig 3.

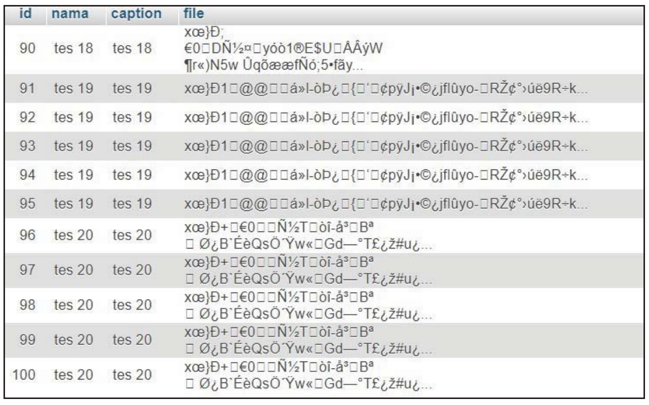


id_gambar ▲ 1	id_posting	file
1038	100	file_20200325023942_7.jpg
1039	100	file_20200325023942_8.jpg
1040	100	file_20200325023942_9.jpg
1041	100	file_20200325023942_10.jpg
1042	100	file_20200325023942_11.jpg
1043	100	file_20200325023942_12.jpg
1044	100	file_20200325023942_13.jpg
1045	100	file_20200325023942_14.jpg
1046	100	file_20200325023942_15.jpg
1047	100	file_20200325023942_16.jpg
1048	100	file_20200325023942_17.jpg
1049	100	file_20200325023942_18.jpg
1050	100	file_20200325023942_19.jpg

id_posting ▲ 1	nama	caption
88	tes 18	tes 18
89	tes 18	tes 18
90	tes 18	tes 18
91	tes 19	tes 19
92	tes 19	tes 19
93	tes 19	tes 19
94	tes 19	tes 19
95	tes 19	tes 19
96	tes 20	tes 20
97	tes 20	tes 20
98	tes 20	tes 20
99	tes 20	tes 20
100	tes 20	tes 20

Fig. 3. Display data in a table by applying conventional techniques

While for applications that use JSON conversion and Zlib compression techniques to store the uploaded image information, only one table is needed, as shown in Fig 4.



id	nama	caption	file
90	tes 18	tes 18	xœ}Ð; €0□DÑ½¤□yóò1®E$U□ÅÂýW ¶r«)N5w Ûqõææf Ñó;5•fãy...
91	tes 19	tes 19	xœ}Ð1□@@□□á»l-òÞ¿□{□'□¢pÿJ¡•©¿jflûyo-□RŽ¢°›úë9R÷k...
92	tes 19	tes 19	xœ}Ð1□@@□□á»l-òÞ¿□{□'□¢pÿJ¡•©¿jflûyo-□RŽ¢°›úë9R÷k...
93	tes 19	tes 19	xœ}Ð1□@@□□á»l-òÞ¿□{□'□¢pÿJ¡•©¿jflûyo-□RŽ¢°›úë9R÷k...
94	tes 19	tes 19	xœ}Ð1□@@□□á»l-òÞ¿□{□'□¢pÿJ¡•©¿jflûyo-□RŽ¢°›úë9R÷k...
95	tes 19	tes 19	xœ}Ð1□@@□□á»l-òÞ¿□{□'□¢pÿJ¡•©¿jflûyo-□RŽ¢°›úë9R÷k...
96	tes 20	tes 20	xœ}Ð+□€0□□Ñ½T□òî-å³□Bª □ Ø¿B`ÉèQsÖ´Ÿw«□Gd—°T£¿ž#u¿...
97	tes 20	tes 20	xœ}Ð+□€0□□Ñ½T□òî-å³□Bª □ Ø¿B`ÉèQsÖ´Ÿw«□Gd—°T£¿ž#u¿...
98	tes 20	tes 20	xœ}Ð+□€0□□Ñ½T□òî-å³□Bª □ Ø¿B`ÉèQsÖ´Ÿw«□Gd—°T£¿ž#u¿...
99	tes 20	tes 20	xœ}Ð+□€0□□Ñ½T□òî-å³□Bª □ Ø¿B`ÉèQsÖ´Ÿw«□Gd—°T£¿ž#u¿...
100	tes 20	tes 20	xœ}Ð+□€0□□Ñ½T□òî-å³□Bª □ Ø¿B`ÉèQsÖ´Ÿw«□Gd—°T£¿ž#u¿...

Fig. 4. Display data in a table by applying the concept of JSON and Zlib

B. Data Storage Speed Testing

The first stage of the experiment was carried out 20 times using applications that apply conventional methods. Whereas in the second stage, a similar experiment was carried out again using applications that applied JSON conversion and Zlib compression techniques. The results of each experiment are recorded in the table, as shown in table 1.

TABLE I. Experiment Of Upload Image

No	Experiment	Conventional (second)	JSON + Zlib (second)
1	Upload 1 File	0.0051	0.0056
2	Upload 2 Files	0.0060	0.0059
3	Upload 3 Files	0.0081	0.0068
4	Upload 4 Files	0.0106	0.0065
5	Upload 5 Files	0.0121	0.0065
6	Upload 6 Files	0.0150	0.0073
7	Upload 7 Files	0.0155	0.0079
8	Upload 8 Files	0.0174	0.0082
9	Upload 9 Files	0.0184	0.0090
10	Upload 10 Files	0.0187	0.0094
11	Upload 11 Files	0.0208	0.0106
12	Upload 12 Files	0.0230	0.0117
13	Upload 13 Files	0.0259	0.0120
14	Upload 14 Files	0.0269	0.0123
15	Upload 15 Files	0.0284	0.0128
16	Upload 16 Files	0.0306	0.0128
17	Upload 17 Files	0.0325	0.0138
18	Upload 18 Files	0.0338	0.0152
19	Upload 19 Files	0.0345	0.0164
20	Upload 20 Files	0.0357	0.0174
	Average	0.0209	0.0104

From the 20 experiments that have been carried out, as shown in table 1, it appears that the average time for uploading images using the conventional method is 0.0209 seconds. While in experiments using JSON Conversion and Zlib compression, the average upload image processing time is smaller, 0.0104 seconds. So the average time of the data storage process by applying JSON Conversion and Zlib compression techniques is 50.36% faster than the conventional method. The rate of increase in time for each experiment performed is shown in Fig. 5.

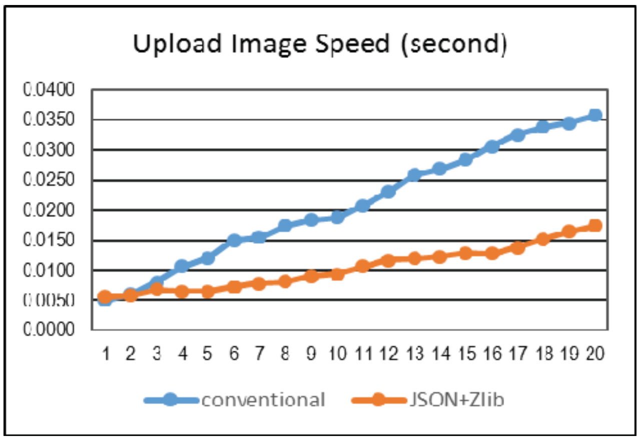


Fig. 5. Upload Image Speed

In Fig 5, it can be seen that in experiments using applications that apply conventional methods or by applying JSON conversion and Zlib compression techniques at the beginning of the experiment, the required processing time tends to be no different. But after 20 attempts, it appears that in a conventional way, there is a significant increase in image upload processing time compared to experiments on applications that use JSON conversion and Zlib compression techniques.

C. File Size Testing

At this stage, the size of the data in the database file used by the two dynamic upload multiple images applications is compared. Table 2 shows the size of the data in each database that was successfully obtained after the experiment.

TABLE II. DATABASE FILE SIZE

No	Database name	Size (bytes)
1	db_upload_conventional	66.549
2	db_upload_json	41.539

Table 2 displays the database file size information for both applications after the experiment. The first application to apply conventional techniques with the database name "db_upload_conventional" obtained data with a size of 66,549 bytes. Whereas in the second application that uses JSON conversion and Zlib compression techniques with the database name "db_upload_json" data obtained with a size of 41,539 bytes. Significantly the size ratio of the two database files is shown in Fig. 6.

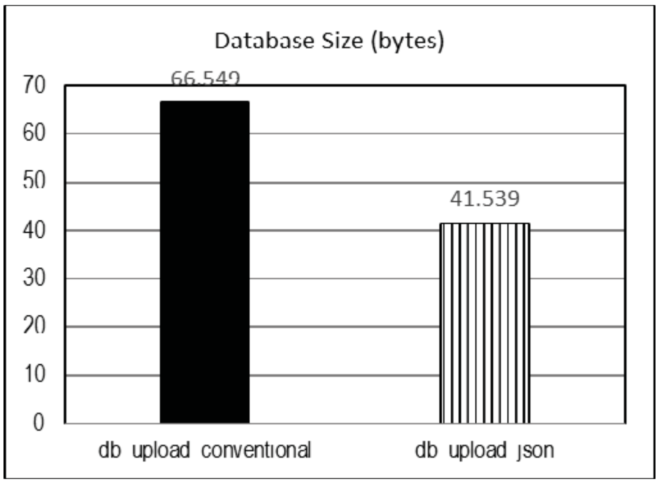


Fig. 6. Database File Size

From the chart view of the data size, as shown in Fig 6, the database file size of applications applying conventional techniques is 66,549 bytes. Whereas the application that uses JSON conversion and Zlib compressed the database file size to 41,539 bytes. So the database file size in applications that implement JSON conversion and Zlib compression is 37.58% smaller than the database file size in applications using conventional techniques.

V. CONCLUSION

JSON and Zlib integration can be applied to handling multiple images to upload dynamic forms. The average time of the data storage process by applying this technique is 50.36% faster compared to conventional methods. While the database file size decreased by about 37.58% smaller than the database file size in applications using conventional techniques. The selection of other compression algorithms, the addition of parameters in measurements in the experiment as well as testing the high concurrency handling are challenges that can be done in further research.

REFERENCES

[1] D. W. Wilson, X. Lin, P. Longstreet, and S. Sarker, “Web 2.0: A Definition, Literature Review, and Directions for Future Research,” *AMCIS 2011 Proc.*, p. Paper 368, 2011.

[2] D. R. Brake, “Are we all online content creators now? Web 2.0 and digital divides,” *J. Comput. Commun.*, vol. 19, no. 3, pp. 591–609, 2014, doi: 10.1111/jcc4.12042.

[3] S. Sabharwal, “Modeling the Navigation Behavior of Dynamic Web Applications,” *Int. J. Comput. Appl.*, vol. 65, no. 13, pp. 20–27, 2013.

[4] A. Galizia, G. Zereik, L. Roverelli, E. Danovaro, A. Clematis, and D. D’Agostino, “Json-GUI—A module for the dynamic generation of form-based web interfaces,” *SoftwareX*, vol. 9, pp. 28–34, 2019, doi: 10.1016/j.softx.2018.11.007.

[5] S. K and B. Dr.G.N.K. Suresh, “Automated Concealed Annotation,” pp. 3–7, 2014.

[6] H. Caple and J. S. Knox, “How to author a picture gallery,” *Journal. Theory, Pract. Crit.*, p. 146488491769198, 2017, doi: 10.1177/1464884917691988.

[7] Q. Huang, K. Birman, R. Van Renesse, W. Lloyd, S. Kumar, and H. C. Li, “An analysis of Facebook photo caching,” *SOSP 2013 - Proc. 24th ACM Symp. Oper. Syst. Princ.*, pp. 167–181, 2013, doi: 10.1145/2517349.2522722.

[8] S. Muralidhar *et al.*, “F4: Facebook’s warm blob storage system,” *Proc. 11th USENIX Symp. Oper. Syst. Des. Implementation, OSDI 2014*, pp. 383–398, 2014.

[9] Y. Ren, “A Novel Approach to Accessing Large Images in the

Database," no. Emcs, pp. 484–487, 2015, doi: 10.2991/emcs-15.2015.98.

[10] M. A. Rosid, "Implementasi JSON untuk Minimasi Penggunaan Jumlah Kolom Suatu Tabel Pada Database PostgreSQL," *JOINCS (Journal Informatics, Network, Comput. Sci.*, vol. 1, no. 1, p. 33, 2017, doi: 10.21070/joincs.v1i1.802.

[11] A. Šimec and M. Magličić, "Comparison of JSON and XML Data Formats," *Cent. Eur. Conf. Inf. Intell. Syst.*, pp. 272–275, 2014.

[12] A. R. Breje, R. Gyorödi, C. Gyorödi, D. Zmaranda, and G. Pecherle, "Comparative study of data sending methods for XML and JSON models," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 12, pp. 198–204, 2018, doi: 10.14569/IJACSA.2018.091229.

[13] B. Lin, Y. Chen, X. Chen, and Y. Yu, "Comparison between JSON and XML in Applications Based on AJAX," *Proc. - 2012 Int. Conf. Comput. Sci. Serv. Syst. CSSS 2012*, no. February 1998, pp. 1174–1177, 2012, doi: 10.1109/CSSS.2012.297.

[14] C. Severance, "Discovering JavaScript Object Notation," *Computer (Long. Beach. Calif).*, vol. 45, no. 4, pp. 6–8, Apr. 2012, doi: 10.1109/MC.2012.132.