Journal of Computer Science, Information Technology and Telecommunication Engineering (JCoSITTE)
Vol. 6, No. 2, September 2026, pp. 908~915
ISSN: 2721-3838, DOI: 10.30596/jcositte.v6i2.26015

Adaptive Categorical Dictionary Implementation for Payload Reduction in AJAX Server-side DataTables Communication

Rosyidah Siregar[1], Husni Lubis[2], Ihsan Lubis[3]
[1]Department of Information Technology, Universitas Harapan Medan, Sumatera Utara, Indonesia
[2,3]Department of Information System, Universitas Harapan Medan, Sumatera Utara, Indonesia

ABSTRACT

Efficient data transmission is a critical aspect of modern web applications, particularly in scenarios involving large tabular datasets rendered through server-side DataTables. This study proposes an adaptive categorical dictionary approach to reduce the payload size transmitted between the server and client. The strategy leverages the high frequency of categorical values within datasets by encoding them into shorter symbolic representations stored in a dynamically generated dictionary. The dictionary is constructed on the server during the initial request and maintained throughout the session, while the client retains a synchronized copy in memory. The research utilizes a publicly available college student dataset containing 1,545 records, focusing on columns with repetitive categorical values such as major, gender, and enrollment status. Experimental simulations were conducted under varying DataTables page lengths (10, 25, 50, and 100) to evaluate the impact of dictionary encoding on request and response payload sizes. Results demonstrate consistent payload reductions across all configurations, with significant improvements observed in larger page lengths—exceeding 12% in some cases. These findings confirm the effectiveness of the adaptive dictionary in minimizing response payloads, thereby improving communication efficiency in AJAX-based data-driven applications. The approach maintains compatibility with native PHP and JavaScript implementations and introduces minimal overhead, making it suitable for integration into existing server-side processing architectures.

Keyword : AJAX communication; DataTables; Payload Optimization; Adaptive Dictionary; Server-Side Processing.

Corresponding Author:
Husni Lubis,
Department of Information System
Universitas Universitas Harapan Medan
Email : husnilubis@unhar.ac.id

Article history:
Received Jul 23, 2025
Revised Aug 20, 2025
Accepted Aug 26, 2025

1. INTRODUCTION

The demand for responsive and efficient web applications has significantly increased in the era of real-time data visualization and interaction (Shethiya, 2025). One of the most widely adopted components for managing tabular data in web interfaces is DataTables, a jQuery-based plugin that supports features such as pagination, search, and sorting (Sahid & Nama, 2022). While DataTables offers a server-side processing mode to handle large datasets more efficiently (Gat et al., 2024), its default JSON-based data exchange can become a bottleneck when transmitting redundant categorical data across requests.

In applications that involve repetitive categorical fields such as student enrollment status, gender, or academic major, the same string values are often transmitted repeatedly with every AJAX request. This redundancy introduces unnecessary overhead in terms of payload size, bandwidth consumption, and response latency, especially when navigating across multiple pages of data or executing frequent filtering and sorting operations.

Several compression strategies have been proposed in the context of web communication, including general-purpose techniques such as GZIP compression (Anand et al., 2023) and custom approaches such as Delta Encoding (Spindler et al., 2024) (Jiang et al., 2021) or dictionary encoding (Cohen et al., 2024) (Sun et al., 2023) for structured data. However, these techniques are either applied at the transport level or tailored for highly structured datasets like logs or sensor streams. There remains a lack of adaptive, client-aware optimization strategies specifically designed for reducing payloads in AJAX-based tabular data interactions.

To address this gap, this research proposes an adaptive categorical dictionary mechanism that can be integrated into the AJAX server-side processing of DataTables. The system automatically identifies columns with low cardinality based on a configurable uniqueness threshold, constructs a per-

session dictionary for those fields, and transmits the dictionary once during the initial request. Subsequent AJAX responses transmit only encoded values, with the dictionary version managed to ensure synchronization between the server and client. This adaptive dictionary model introduces two communication phases: a base synchronization phase, where the client receives the full dictionary and its version metadata, and an optimized phase, where only encoded data is transmitted as long as the dictionary remains unchanged. If a change occurs in the categorical values, the server triggers a dictionary update, maintaining the consistency of decoded data on the client side. This approach significantly reduces the size of transmitted JSON payloads without affecting the rendering logic of DataTables on the frontend.

The experiment was conducted using a dataset of student records sourced from Kaggle, with a simulation of AJAX server-side pagination, search, and sort operations. A baseline system without dictionary encoding was also implemented to compare the raw payload sizes with and without the adaptive dictionary mechanism across multiple pages. The results of this research demonstrate that adaptive categorical dictionaries can achieve substantial payload reduction, especially in datasets with several low-cardinality fields. Furthermore, the approach is fully compatible with existing DataTables infrastructure and can be applied to various PHP-native web applications without requiring major frontend modifications. This study contributes a novel and lightweight optimization method to the field of web data communication and sets the stage for future enhancements such as persistent client-side caching and delta dictionary updates.

2. RESEARCH METHOD

To address the inefficiencies in data payload transmission within server-side DataTables implementations, this study proposes an adaptive categorical dictionary approach tailored for AJAX-based web communication. The research methodology is structured to provide a systematic framework from problem identification to solution evaluation. The stages include analyzing the core communication problem, reviewing relevant literature on dictionary-based compression, preparing the experimental dataset, designing the adaptive system architecture, conducting experiments through simulated user interactions, and finally evaluating the results through quantitative metrics. Each stage is carefully designed to ensure that the proposed solution can be objectively assessed in terms of its efficiency in reducing payload without altering user-visible functionality.

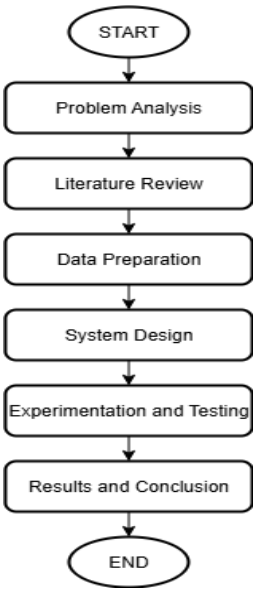

Figure 1. Research Methods

2.1. Problem Analysis

In web-based data systems that utilize server-side DataTables (Praba et al., 2021), especially those with frequent pagination and filtering, repeated transmission of categorical string values significantly contributes to communication payload overhead (Setiyadi & Setiawan, 2025). This becomes particularly inefficient when dealing with columns that contain low-cardinality values (e.g., gender,

major, status). Although general-purpose compression techniques exist, they are not always effective due to short payload sizes and JSON serialization overhead. Therefore, a specialized, context-aware approach is required to address this redundancy and optimize payload delivery.

2.2. Literature Review

Previous studies have explored various payload optimization techniques for web communication, including string tokenization, delta encoding, and dictionary-based compression. In the context of AJAX and JSON communication, methods like Delta Encoding (Zhou et al., 2022) (Wollmer et al., 2023) and Dictionary-Based Compression (DBC) (Zizi & Turquais, 2021) (Fira et al., 2022) have shown promising results. However, most approaches rely on static dictionaries or pre-defined schemas that lack adaptability to evolving datasets. Adaptive techniques, as proposed in this research, offer a dynamic mechanism to adjust to unique values encountered during runtime, ensuring both flexibility and efficiency.

2.3. Data Preparation

The dataset used for experimentation was obtained from Kaggle (Uddin, 2022), titled College Student Management Dataset, consisting of 1,545 records. The relevant columns used in this study include:

a. id (identifier),
b. major (categorical),
c. age (numeric),
d. gender (categorical),
e. gpa (numeric),
f. course_load (numeric),
g. avg_course_load (numeric),
h. attendance_rate (numeric),
i. enrollment_status (categorical).

The data is considered static for experimental purposes and is stored in a MySQL database. The dataset is accessed via a PHP native backend system that simulates real-world server-side processing in DataTables.

2.4. System Design

The system is designed to operate in two modes: baseline (non-compressed) and adaptive dictionary-based. In the adaptive mode, the following components are implemented:

2.4.1. Adaptive Dictionary Construction

Upon the initial data request, the server evaluates each column to determine its eligibility for dictionary encoding. A threshold of 0.5% unique value ratio is applied to identify low-cardinality categorical columns. If eligible, a dictionary is generated mapping each unique value to an index. These dictionaries are stored in the server-side session for reuse.

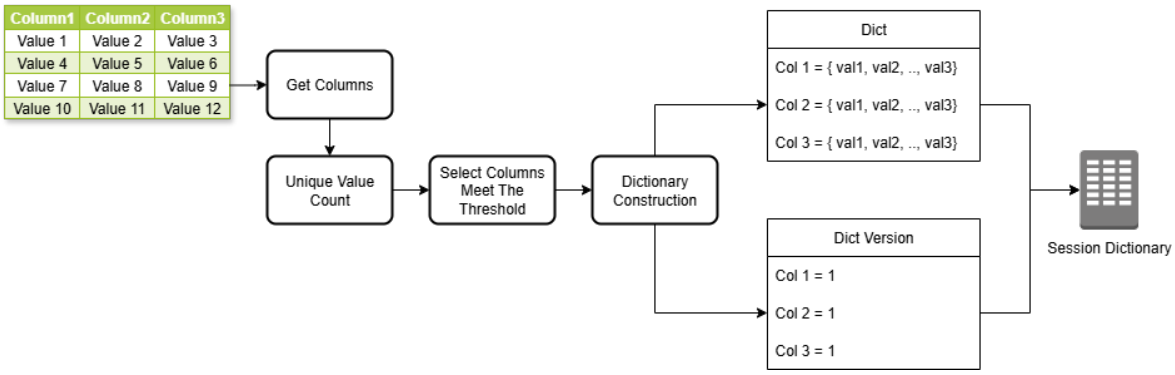

Figure 2. Dictionary Construction Process

Each dictionary is also versioned. When the dataset changes or new values appear, the dictionary is updated and its version incremented. The client-side system maintains a local copy of the dictionary and its current version.

2.4.2. Client-Server Communication

Communication between the client (browser) and server follows a structured AJAX pattern with DataTables parameters. The flow is divided into two stages:

a. Initial Request Phase: The server builds and returns the initial dictionary (if any), its version, and encoded data. The client stores this dictionary locally.

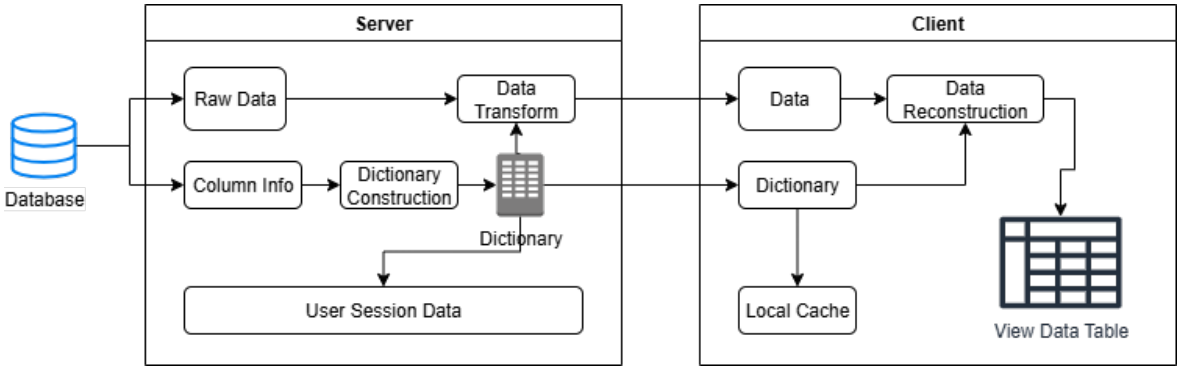

Figure 3. Initial Request Phase

b. Subsequent Request Phase: The client includes its known dictionary versions in the request. If there is no version change, the server returns only encoded data. If versions have changed, updated dictionaries and their new versions are transmitted alongside data.

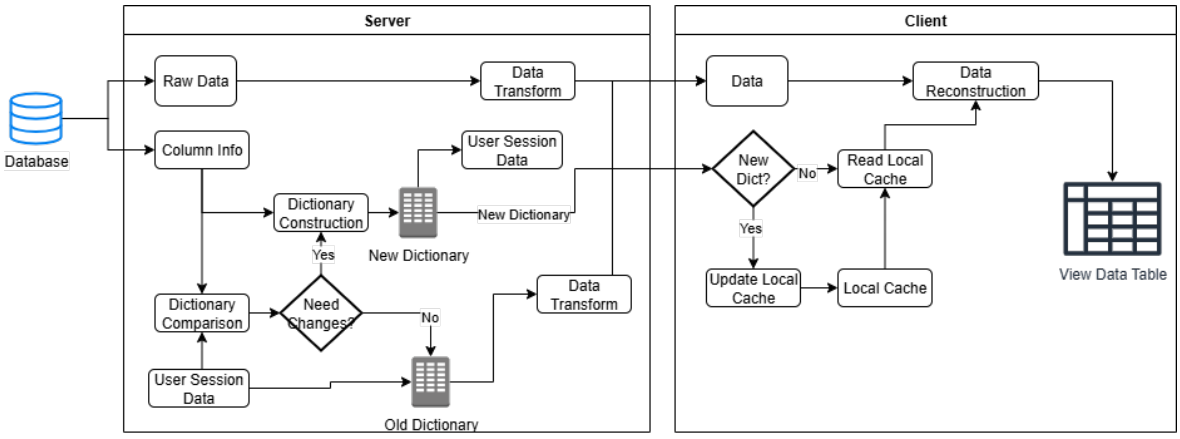

Figure 4. Subsequent Request Phase

This mechanism ensures that categorical values are replaced by compact integer representations, reducing repeated transmission of string literals.

2.5. Experimentation and Testing

The experiment simulates a single-user session navigating all pages of the dataset using both the baseline and adaptive systems. The following steps are performed:

a. Pagination Simulation: The client is scripted to request each page sequentially.
b. Payload Measurement: For each page, the size of the server response is recorded.
c. Cost Accumulation: Total payload size (in bytes) across all pages is computed.
d. Page Length Variation: Tests are repeated using different page lengths to observe performance across granularities (e.g., 10, 25, 50, 100 rows per page).

All experiments are performed in a controlled environment using static data and one client instance to eliminate external network variations.

2.6. Results and Conclusions

The evaluation phase in this research focuses solely on analyzing payload consumption as the primary performance metric. To measure the impact of the adaptive categorical dictionary implementation, the system was tested under controlled conditions simulating user navigation through paginated data views. Both the baseline system (without compression) and the proposed adaptive version were subjected to identical user interaction sequences. Payload size was recorded at each page request during the simulation. These measurements were collected for both configurations using varying page lengths (i.e., the number of records displayed per page). For every page visit, the total size of the server response (in bytes) was logged. This allowed the construction of a comparative dataset of payload sizes across different conditions.

The primary variable of interest is the cumulative payload, representing the sum of all response sizes throughout the complete navigation session. This metric was chosen because it directly reflects the total amount of data transmitted between server and client during a typical session. To support interpretation, response size deltas between the baseline and adaptive versions were computed for each page, enabling the identification of efficiency trends. The analysis does not directly assert superiority of one system over another but instead focuses on the empirical observation of payload differences. Conclusions regarding performance are drawn based on the consistency and magnitude of these differences over the simulation runs.

To quantify the observed results, several key metrics were defined:

a. Payload Size per Page. Each page visit yields a response from the server, whose size in bytes is denoted as:

$$P_i = \text{Size of response at page i} \quad (1)$$

b. Total Payload per Session. The total payload across all n pages in a browsing session is computed as:

$$TP = \sum_{i=1}^{n} P_i \quad (2)$$

c. Average Payload per Page. To examine payload behavior on a per-page basis, the average payload is calculated as:

$$\bar{P} = \frac{TP}{n} \quad (3)$$

These metrics form the basis for interpreting the experimental outcome. The primary focus remains on assessing how the adaptive dictionary impacts communication payload, particularly in relation to varying page sizes and categorical column compositions. Visual comparisons and efficiency trends are further elaborated in the next section.

3. RESULTS AND DISCUSSION

This section presents the outcomes of the experimental evaluation of the proposed adaptive categorical dictionary strategy. The objective is to assess the extent to which the technique reduces communication payload in server-side DataTables AJAX processing, compared to the conventional (uncompressed) approach. The experiment was conducted by simulating sequential page navigation in a DataTable rendered using full server-side processing. Two system configurations were compared: the baseline version, which returns raw uncompressed JSON responses, and the adaptive version, which applies the dictionary-based transformation to categorical columns identified during the first request.

To observe the effect of different page sizes on payload consumption, the simulation was repeated using four different page length values: 10, 25, 50, and 100. For each configuration, the complete traversal of all pages was performed and the server's response payload size was recorded at every request. This comparative approach enables the identification of patterns and variations in data efficiency, both in terms of per-page payload and cumulative transmission cost. The analysis focuses on how the volume of data transferred changes as page size increases, and how the dictionary-based compression technique adapts to different data granularities.

3.1. Small Page Length Evaluation (10 and 25)

In this section, the experiment focuses on evaluating the impact of the adaptive categorical dictionary strategy when used with smaller page lengths, specifically 10 and 25 entries per page. These configurations represent common default values used in many real-world DataTables applications and are typically associated with more frequent AJAX interactions due to the limited number of records shown per request. The aim of this evaluation is to observe how the overhead introduced by the dictionary mechanism interacts with the relatively high frequency of data requests. At smaller page lengths, each server response contains fewer rows, which can make the size of the dictionary (transmitted during the first request or when updates occur) more noticeable in terms of total payload. However, the repeated reuse of the dictionary in subsequent pages may offer benefits in total cost reduction over the course of full pagination.

Table 1. Small Page Length Evaluation (Standard Payload)

	Page Length 10			**Page Length 25**		
	Request Payload	**Response Payload**	**Total Payload**	**Request Payload**	**Response Payload**	**Total Payload**
Total	14351	325672	340023	5712	319571	325283
Average	92.58709677	2101.109677	2193.696774	92.12903226	5154.370968	5246.5

Table 2. Small Page Length Evaluation (Enhanced Payload)

	Page Length 10			Page Length 25		
	Request Payload	**Response Payload**	**Total Payload**	**Request Payload**	**Response Payload**	**Total Payload**
Total	14351	325672	340023	5712	278806	284518
Average	92.58709677	2101.109677	2193.696774	92.12903226	4496.870968	4589

At a page length of 10, the standard approach resulted in an average total payload of 3,557.19 bytes, consisting of 92.59 bytes of request payload and 3,464.60 bytes of response payload. In contrast, the adaptive approach achieved a reduced average total payload of 3,123.52 bytes, with the same request size and a compressed response payload of 3,030.93 bytes. This reflects a notable reduction of approximately 12.17% in the average total payload.

When the page length was increased to 25, the standard method exhibited an average total payload of 5,246.50 bytes (with 92.13 bytes for request and 5,154.37 bytes for response). The adaptive method, on the other hand, yielded a lower average total of 4,589.00 bytes, reducing the response payload to 4,496.87 bytes. This corresponds to an efficiency gain of around 12.53%.

3.2. Large Page Length Evaluation (50 and 100)

To further evaluate the scalability and performance of the adaptive categorical dictionary strategy, additional experiments were conducted using larger pagination settings, specifically with page lengths of 50 and 100. These settings simulate use cases in which users prefer to view more data per page, often at the cost of larger payload sizes due to increased data volume in each request-response cycle.

Table 3. Large Page Length Evaluation (Standard Payload)

	Page Length 50			**Page Length 100**		
	Request Payload	**Response Payload**	**Total Payload**	**Request Payload**	**Response Payload**	**Total Payload**
Total	2851	317556	320407	1483	316581	318064
Average	91.96774194	10243.74194	10335.70968	92.6875	19786.3125	19879

Table 4. Large Page Length Evaluation (Enhanced Payload)

	Page Length 50			**Page Length 100**		
	Request Payload	**Response Payload**	**Total Payload**	**Request Payload**	**Response Payload**	**Total Payload**
Total	2851	276791	279642	1483	275816	277299

Average	91.96774194	8928.741935	9020.709677	92.6875	17238.5	17331.1875

For a page length of 50, the standard server-side processing approach recorded an average response payload of 10,243.74 bytes and an average total payload of 10,335.71 bytes, with request sizes remaining relatively small at 91.97 bytes. In comparison, the adaptive method demonstrated a reduction in response payload to 8,928.74 bytes, resulting in a lower average total payload of 9,020.71 bytes. This marks an approximate 12.73% reduction in total payload size.

When the pagination was increased further to 100 rows per page, the standard approach exhibited an average total payload of 19,879.00 bytes, composed of 92.69 bytes for the request and 19,786.31 bytes for the response. The adaptive method again achieved a more efficient transmission, with the response payload dropping to 17,238.50 bytes, and a corresponding average total payload of 17,331.19 bytes. This translates to a 12.83% efficiency gain compared to the standard method.

3.3. Comparative Insights and Observations

The experimental evaluation across varying page lengths highlights several key insights regarding the performance impact of the adaptive categorical dictionary strategy on payload consumption. Firstly, the most significant gains were consistently observed in the response payloads rather than the request payloads. This is expected, as the primary optimization targets categorical values transmitted repetitively in each response row. Since the request structure remains relatively minimal and static in server-side DataTables communication, improvements in this area were marginal.

Secondly, while the adaptive method shows consistent advantages across all scenarios, the magnitude of efficiency gain is influenced by the page length. In small page length settings (10 and 25), the reduction in total payload ranged from moderate to noticeable. However, in larger page lengths (50 and 100), the gains became more prominent—reaching over 12% in payload savings. This suggests that the strategy scales effectively with data volume, which is crucial for applications dealing with large tabular datasets. Additionally, the experimental data reveals that the overhead of dictionary handling during the initial request is negligible in terms of payload cost. The dictionary is transmitted only when necessary—upon the first request or when a column's unique value set changes—ensuring that compression efficiency outweighs the additional transmission cost.

Moreover, the consistency in request payload sizes across all methods and scenarios confirms that the optimization does not interfere with standard client-server interactions and is transparent to front-end implementations. This makes the approach particularly attractive for integration into existing systems with minimal modifications. In summary, the comparative analysis underscores the practical efficiency and adaptability of the proposed method. Its ability to reduce payload size without altering the structure of existing protocols supports its potential for real-world application in bandwidth-sensitive or large-scale DataTables deployments.

4. CONCLUSION

This research introduced and evaluated an adaptive categorical dictionary strategy aimed at reducing payload size in server-side DataTables communication. The method was designed to address inefficiencies in transmitting repetitive categorical values, which commonly inflate response payloads during paginated data browsing. By implementing a server-managed dictionary that maps frequent categorical values to compact representations—synchronized with a client-held counterpart—the system achieved payload compression without altering the fundamental request-response flow or requiring significant client-side modifications. The dictionary is initialized during the first page request and updated only when necessary, minimizing transmission overhead.

Experimental evaluation focused solely on payload consumption, comparing the standard server-side implementation with the adaptive dictionary-enhanced variant. Tests were conducted across varying page lengths (10, 25, 50, and 100) to understand the scalability and effectiveness of the approach. The results revealed consistent reductions in response and total payload sizes, particularly in scenarios with larger page lengths, where improvements exceeded 12%. The study concludes that the proposed strategy is a viable and effective enhancement to server-side DataTables implementations, especially for datasets with high categorical redundancy. It offers meaningful bandwidth savings with minimal architectural disruption, making it suitable for resource-constrained environments or large-scale web applications.

Future research may expand the scope by incorporating dynamic datasets, evaluating dictionary management over prolonged sessions, or integrating the approach with additional compression or encoding techniques to further enhance communication efficiency.

REFERENCES

Anand, K., Priyadharshini, M., & Priyadharshini, K. (2023). Compression and Decompression of Files Without Loss of Quality. *Proceedings of the 1st IEEE International Conference on Networking and Communications 2023, ICNWC 2023*, 1–6. https://doi.org/10.1109/ICNWC57852.2023.10127236

Cohen, D., Cohen, S., Naor, D., Waddington, D., & Hershcovitch, M. (2024). Dictionary Based Cache Line Compression. *HOTSTORAGE 2024 - Proceedings of the 2024 16th ACM Workshop on Hot Topics in Storage and File Systems*, 8–14. https://doi.org/10.1145/3655038.3665941

Fira, M., Costin, H. N., & Goraş, L. (2022). A Study on Dictionary Selection in Compressive Sensing for ECG Signals Compression and Classification. *Biosensors*, *12*(3), 146. https://doi.org/10.3390/bios12030146

Gat, Jamil, M., Wingdes, I., Widayanti, T., Wijaya, T., & Kusrini. (2024). Using Server-side Processing Techniques to Optimize Data Presentation Responsiveness. *2024 6th International Conference on Cybernetics and Intelligent System, ICORIS 2024*, 1–6. https://doi.org/10.1109/ICORIS63540.2024.10903755

Jiang, H., Liu, C., Paparrizos, J., Chien, A. A., Ma, J., & Elmore, A. J. (2021). Good to the Last Bit: Data-Driven Encoding with CodecDB. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 843–856. https://doi.org/10.1145/3448016.3457283

Praba, A. D., Safitri, M., & Faridi, F. (2021). Implementasi Datatables Server-Side Untuk Mempercepat Load Halaman Pada Aplikasi E-Commerce. *JIKA (Jurnal Informatika)*, *5*(2), 139. https://doi.org/10.31000/jika.v5i2.4339

Sahid, A., & Nama, G. F. (2022). Design and Development of Management Information Systems at the University of Lampung Library Repository Using the Laravel Framework. *Journal of Engineering and Scientific Research*, *4*(2), 74–83. https://doi.org/10.23960/jesr.v4i2.110

Setiyadi, A., & Setiawan, E. B. (2025). Analysis data loading of new entrepreneur in West Java using client server-side method. *AIP Conference Proceedings*, *3200*(1), 40011. https://doi.org/10.1063/5.0255261

Shethiya, A. S. (2025). Scalability and Performance Optimization in Web Application Development. *Journal of Science and Technology Computer Science & Information Technology*, *2*(1). https://creativecommons.org/licenses/by/4.0/deed.en

Spindler, J., Fent, P., Riedl, A., & Neumann, T. (2024). Can Delta Compete with Frame-of-Reference for Lightweight Integer Compression? *Proceedings of the VLDB Endowment. ISSN*, *2150*, 8097.

Sun, X., Mo, D., Wu, D., Ye, C., Yu, Q., Cui, J., & Zhong, H. (2023). Efficient regular expression matching over hybrid dictionary-based compressed data. *Journal of Network and Computer Applications*, *215*, 103635. https://doi.org/10.1016/j.jnca.2023.103635

Uddin, Z. (2022). *College Student Management Dataset*. https://www.kaggle.com/datasets/ziya07/college-student-management-dataset

Wollmer, B., Wingerath, W., Ferrlein, S., Panse, F., Gessert, F., & Ritter, N. (2023). The Case for Cross-entity Delta Encoding in Web Compression (Extended). *Journal of Web Engineering*, *22*(1), 131–146. https://doi.org/10.13052/jwe1540-9589.2217

Zhou, X., Qi, C. R., Zhou, Y., & Anguelov, D. (2022). RIDDLE: Lidar Data Compression with Range Image Deep Delta Encoding. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2022-June*, 17191–17200. https://doi.org/10.1109/CVPR52688.2022.01670

Zizi, M. O. F., & Turquais, P. (2021). A dictionary learning method for seismic data compression. *Geophysics*, *87*(2), 1–83. https://doi.org/10.1190/geo2020-0948.1