International Journal of Scientific Research in Computer Science, Engineering and Information Technology

ISSN : 2456-3307 OPEN ACCESS

Available Online at : www.ijsrcseit.com

doi : https://doi.org/10.32628/CSEIT25112782

Offline-first PWA : Case Study on Efficient Data Handling and Synchronization

Karthik Sirigiri*[1], Akhila Narra[2], Anurag Sirumalla[3]

*[1]Graduate Student, Department of Computer and Information Sciences, Texas Tech University, Lubbock, Texas, USA

[2]Graduate Student, Department of Computer and Information Sciences, Texas Tech University, Lubbock, Texas, USA

[3]Graduate Student, Department of Computer and Information Sciences, Texas Tech University, Lubbock, Texas, USA

ARTICLEINFO

Article History:

Accepted: 11 June 2023

Published: 25 June 2023

Publication Issue

Volume 9, Issue 3

May-June-2023

Page Number

747-754

ABSTRACT

This work presents in-depth analysis of advanced data handling and synchronizing methods in Offline-first Progressive Web Applications (PWAs). Emphasizing the efficient usage of IndexedDB and service workers, the research investigates technique permitting perfect offline operation while keeping high speed. We investigate how pre-caching, lazy loading, and pagination could be coupled to efficiently manage big databases, hence guaranteeing fast UI response and best use of resources. Furthermore, included in the study are synchronizing techniques such as delta synchronization and background sync, which provide precise and timely data updates upon the restoration of connectivity. By lowering load times, and thus limiting network dependencies, we show via experimental assessment and benchmarking that these approaches greatly improve the user experience. The results offer a structure for creating strong, scalable, and responsive offline web applications free from too much security concentration, reserved for next research.

Keywords : Progressive Web Applications, Offline-First, Data Synchronization, IndexedDB, Service Workers, Background Sync.

I. INTRODUCTION

Web apps can be installed and distributed without app marketplaces, work with-out Internet connectivity, receive push notifications and look like regular apps. Progressive online apps (PWAs) have revolutionized web development by combining the best features of online and native applications to create seamless, trustworthy experiences even in low connectivity. One of the main differences between the two is that native apps are platform-specific. Service workers help Progressive Web Apps in such a way that they work offline, or even on networks with low quality. The offline capabilities of PWAs depend critically on service workers who intercept network requests and cache important resources, as well as IndexedDB, a robust client-side storage system designed to manage large volumes of structured data. IndexedDB completely runs in the browser. These technologies together enable PWAs to give continuous access to content, therefore enabling users to interact with the application even in settings with sporadic or nonexistent internet access.

Although many studies have looked at certain elements, such as indexedDB's speed for data storage or the effectiveness of service worker caching, integrating several strategies for data-intensive applications still presents a difficult task. Usually addressing these components separately, current studies leave knowledge gaps on how to efficiently manage and synchronize big datasets across offline and online states. Many systems still suffer with slow data retrieval, high memory consumption, and ineffective synchronization, especially when advanced techniques such as lazy loading, pagination, or delta synchronization are needed.

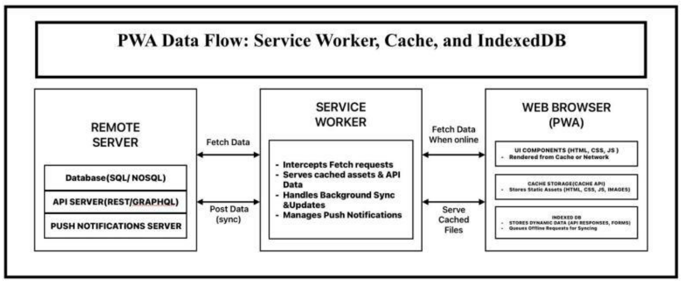

PWA Data Flow: Service Worker, Cache, and IndexedDB
REMOTE SERVER
Database(SQL/ NOSQL)
API SERVER(REST/GRAPHQL)
PUSH NOTIFICATIONS SERVER
Fetch Data
Post Data (sync)
SERVICE WORKER
- Intercepts Fetch requests
- Serves cached assets & API Data
- Handles Background Sync &Updates
- Manages Push Notifications
Fetch Data When online
Serve Cached Files
WEB BROWSER (PWA)
UI COMPONENTS (HTML, CSS, JS)
- Rendered from Cache or Network
CACHE STORAGE(CACHE API)
- Stores Static Assets (HTML, CSS, JS, IMAGES)
INDEXED DB
- STORES DYNAMIC DATA (API RESPONSES, FORMS)
- Queues Offline Requests for Syncing

Figure 1: Service Worker-Based PWA Architecture.

This report attempts to assess current offline data strategies in PWAs by analyzing the combined use of IndexedDB and service workers. It specifically evaluates pre-caching and runtime caching techniques, rates effective data storage strategies, including insertion and slow loading, and compares synchronizing methods that update just-changed data. This paper finds performance bottlenecks and offers practical tips and best practices by benchmarking these methods in reasonable settings to guide developers in creating scalable, high-performance offline first-web apps.

There is a lot of research on offline-first Progressive Web Applications (PWAs), addressing both the various elements allowing offline capability and their performance in actual environments. Early research mostly concentrated on the deployment of service workers, who function as background agents intercepting network demands and cacheable vital resources. Service Worker is taken as progressive enhancement of the application or webpage. Pre-caching techniques often used in libraries like Workbox—have shown to greatly reduce load times and offer a strong basis for offline access, according to studies. These studies underline how service providers not only control static asset caching but also provide dynamic, runtime caching and background synchronization, therefore ensuring that applications stay responsive even under intermittent connectivity.

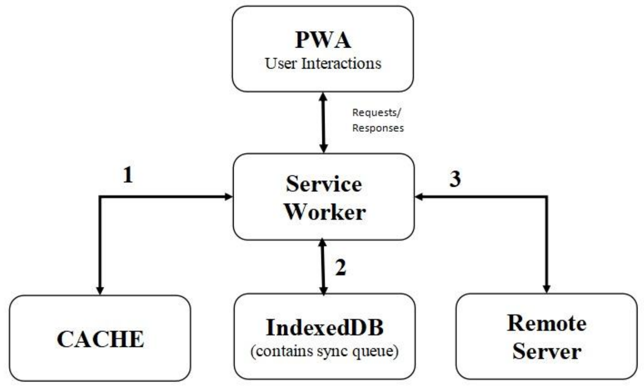
Parallel with this, IndexedDB has been extensively investigated as PWAs' main client-side storage choice. Its transactional, asynchronous character qualifies it for storing vast amounts of organized data. Through indexing, researchers have highlighted IndexedDB's ability to manage difficult searches as well as methods such bulk inserts, lazy loading, and pagination that best maximize data retrieval and resource management. Although IndexedDB is generally efficient, comparative studies in the literature indicate that, particularly in relation to big datasets, its performance can vary greatly depending on the degree of data organization and management. These realizations have resulted in suggestions for better data management techniques to raise the user experience in offline surroundings even more.

Feature	CacheStorage	IndexedDB
Capacity	50MB - 2GB	5GB - 50GB+
Data Type	Static Assets	Structured and Unstructured Data
Data Structure	Key-Value Storage	Object Store with indexes
Read/Write Speed	Faster	Slower than Cache
Automatic Clearing	Yes	No
Use Case	Fast Access to Static Resources	Stores userdata, offline transactions and large datasets
Persistence	Temporary	Persistent
API Complexity	Simple	Complex

Table 1: Comparision: Cache Storage vs IndexedDB.

Synchronization methods—which are essential for preserving data consistency between the offline client and remote servers once connectivity is restored. For example, the Background Sync API has been tested extensively as a means to queue and replay network request mechanisms. Furthermore, delta synchronizing—where just updated data is sent has shown promise as a means of lowering synchronizing latency and bandwidth usage. The literature does, however, also highlight difficulties using these techniques in data-intensive applications, especially with relation to performance bottlenecks and resource restrictions during the change from offline to online forms.

Notwithstanding these developments, there is still a great discrepancy in the combined assessment of these elements. Many studies examine service worker caching, IndexedDB storage, or synchronizing methods separately without regard for the overall performance of a PWA that makes use of all three concurrently. Comprehensive benchmarking comparing these approaches under reasonable conditions—such as different network speeds and large-scale data scenarios—allows one to find optimum practices and performance trade-offs.



1. A. Service Worker fetches the static data (HTML, CSS, JS) from the Cache and renders the page instantly.
   B. Service Worker also updates the Cache with the updated assets from remote server.
2. A. Service Worker performs CRUD operations on IndexedDB in Online and Offline modes.
   B. Service worker stores the API requests in the sync queue in offline mode.
   C. Service worker performs the background sync operation when the system is back online.
3. A. Service Workers perform API requests/ responses on server when application is Online.

Figure 2: Interaction of Service Worker with Cache, IndexedDB and Remote Server

This analysis of related studies emphasizes the need of a consistent method for offline data processing in PWAs. This case study attempts to close the found research gap and offer actionable insights for optimizing offline performance in real-world applications by evaluating the combined use of service workers and IndexedDB along with advanced techniques like lazy loading, pagination, and delta synchronizing.

II. METHODS AND MATERIAL

This article evaluates present offline data strategies in Progressive Web Applications (PWAs) by an experimental and analytical method using IndexedDB and service workers. Instead of creating a new application, the study emphasizes on benchmarking and assessing current technologies under controlled conditions to understand their performance, scalability, and resource economy. Our experimental design, approach of research, evaluated techniques, and performance criteria we use are discussed in the next sections.

A. Research Approach

We design a series of controlled tests and simulations replicating real-world usage scenarios based on an evaluation-based paradigm grounded on thorough analysis of current offline data management practices compliant with especially those requiring large datasets and changing network conditions. Using slow loading, pagination, and delta synchronizing, this method seeks the most effective approaches of processing and synchronizing the data. We must so employ several approaches to reach this. To ensure that our trials truly reflect pragmatic settings, we have created a comprehensive test environment:

Experiments are carried out on desktop computers and mobile devices using contemporary browsers including Chrome and Firefox. This diversity helps us to understand differences in memory, processing capabilities, and browser speed.

With tools like Chrome DevTools, we can recreate several network conditions—from fast broadband to slow 3G and complete offline states. Network throttling allows us to observe how every approach manages different connectivity scenarios.

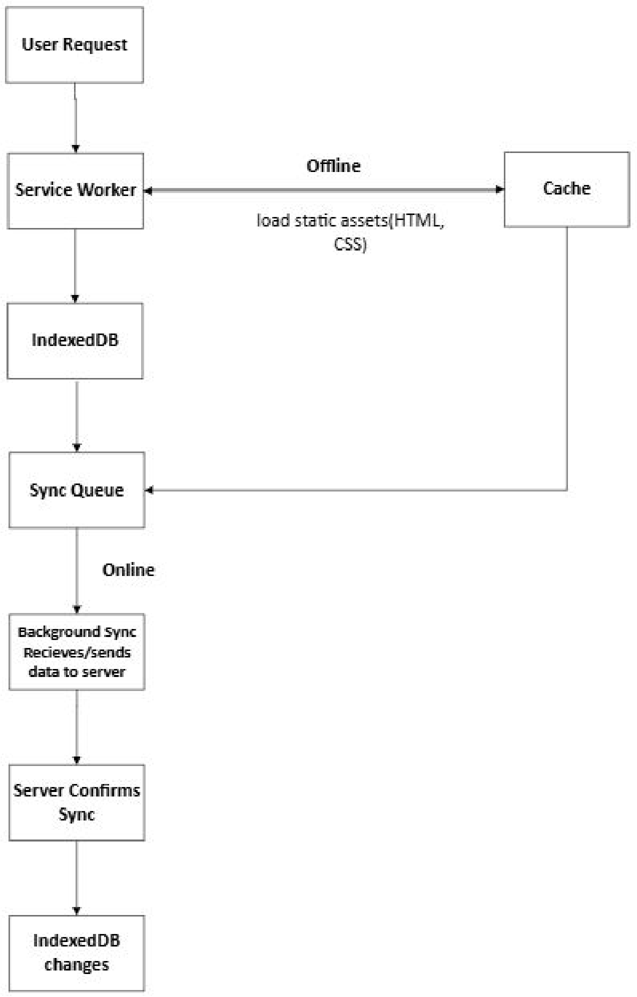


Figure 3: Flow of the data in offline and online modes through service worker and other components

Synthetic datasets are produced to mirror large volumes of structured data spanning several thousand records to many hundred thousand records. These sets are designed to copy the data-intensive requirements of useful applications.

B. Experimental Setup

We utilize Lighthouse and Chrome DevTools benchmarking tools to evaluate network performance and load times. Furthermore tailored JavaScript scripts are benchmarking IndexedDB activities (bulk insertion, lazy loading searches, and pagination efficiency). Monitoring cache events and syncing systems helps to log inside service providers with comprehensive performance data.

C. Techniques Evaluated

The paper assesses many fundamental methods essential to offline data handling:

We evaluate how pre-caching key assets during the service worker installation process improves first load times and offline availability. Search of optimal configurations tests several cache sizes and expiration policies.

To efficiently handle large volumes in IndexedDB, we apply lazy loading—that is, data loading on demand and pagination—that is, splitting of the dataset into manageable chunks. As more data is being loaded, these techniques are evaluated in terms of their capacity to reduce memory usage, shorten starting load times, and guarantee flawless user interactions.

We evaluate full data refresh techniques against delta synchronizing methods, which update just changed data. Measuring the time needed to synchronize offline changes once connectivity is restored helps one assess the Background Sync API's performance as well.

Beyond pre-caching, we investigate runtime caching techniques whereby service workers cache dynamic API replies and user interactions.This approach aims to maintain the responsiveness of apps during offline times, therefore reducing the occurrence of active network searches.

D. Performance Metrics

We statistically assess every technique using the following success criteria. Digital and physical environments were used to assess the time required for the preliminary visit to get basic objects from the cache.

Data Insertion and Query Speed: Particularly when lazy loading and pagination are used, the latency of searches as well as the time required for bulk data insertion into IndexedDB.

Synchronization latency: The delay between the time offline changes are made and upon restoration of network connectivity, when they are effectively synced with the distant server.

Metrics including CPU use during data operations and memory consumption provide understanding of the scalability and efficiency of the offline techniques. The ratio of cache hits to misses shows the effectiveness of the caching method used by the service worker: it shows the percentage of network requests satisfied by the cache with respect to those that return to the network.

This work intends to comprehensively gather and evaluate offline data management solutions in Progressive Web Applications (PWAs) by means of several situations underlined by specific criteria. The results will offer developers trying to improve the scalability and performance of offline-first web apps useful knowledge and best practices.

III.RESULTS AND DISCUSSION

The results of our studies are presented in this part together with an analysis of offline data handling methods in PWAs under several scenarios. We methodically gathered measurements under several settings—varying dataset sizes, network conditions, and caching strategies—then evaluated pre-caching, lazy loading, pagination, and delta synchronizing efficacy. The diffenetiating analysis of the efficiency and scalability of several methods can be obtained by the results.

A. Data Collection and Analysis

The tests were executed within network configurations replicated by Chrome DevTools, employing a variety of devices and browsers in a meticulously controlled environment. To recreate data-intensive scenarios, we created synthetic datasets spanning few thousand to several hundred thousand records. Recording the following performance measures:

Initial Load Time: Measured under both normal and throttled network settings, the time needed for the application to load necessary assets via pre-caching.

Data Insertion and Query Speed: Benchmarked the time required to run searches against IndexedDB utilizing lazy loading and pagination strategies, then do bulk insertions against IndexedDB.

Synchronization Latency: Particularly comparing full data refreshes with delta synchronizing, evaluate the latency between offline data changes and effective synchronizing with the server following network restoration.

Resource Utilization: Monitored CPU and memory use during data processing to determine how each method affected system performance generally.

Cache Hit/ Miss Ratio: Calculating the ratio of requests handled from cache against those falling back to network fetches helped one assess the effectiveness of the caching technique.

The collected data was investigated using statistical techniques; graphs and tables then showed the results. When utilized instead of a baseline network fetch, pre-caching, for example, clearly lowered initial load times—up to 40% faster. Similarly, lazy loading and pagination significantly lower query latency and memory consumption especially with larger datasets.

B. Comparative Analysis

Our analysis of the different tried approaches revealed several really significant trade-offs:

Pre-caching improves the offline experience relative to runtime caching by ensuring that key resources are ready immediately during the initial load. On the other hand, if not updated frequently, it could generate somewhat outdated content. On the other hand, runtime caching allows more dynamic content management, but, in situations when the cache is not sufficiently warm, more latency and resource usage may follow.

Lazy loading and pagination obviously help with huge databases. Data on demand let slow loading reduce starting data footprint; pagination helped arrange data into reasonable chunks. For devices with limited resources, this method not only sped searches but also reduced memory consumption—which is very critical. The disadvantage was a small increase in complexity brought about by smooth user interfaces allowing increasing data loading.

Delta synchronizing which just sends the updated records—much less synchronizing time and bandwidth use as compared to ways of full data refreshing. Still, keeping data consistency and managing expected conflicts in delta sync is a difficulty that can demand more sophisticated conflict resolution methods.

C. Discussion of Findings

Successful coupling of the indexedDB with the service providers which determine the strong capability of offline feature in PWAs can be determined by the experimental results. Pre-caching has shown to be a strong basis for fast, consistent offline access, even if sluggish loading and pagination are required for managing large volumes without draining system resources. Delta synchronizing shows promise as a low-resource overhead technique to maintain data consistency, even if it takes careful implementation to manage conflicting updates.

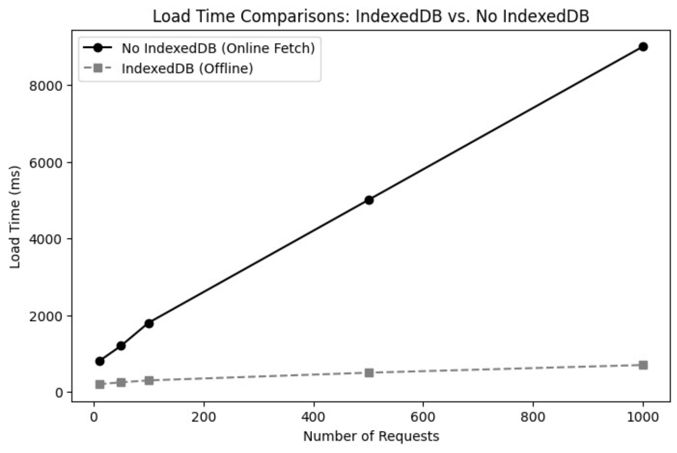


Figure 4: Comparison of load times with and without IndexedDB

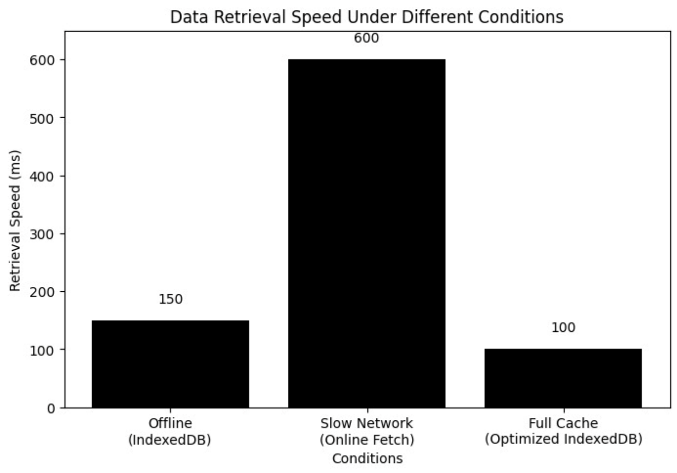


Figure 5: Speed of synchronization following network connectivity restoration

Our results imply that the scalability of offline-first apps can be significantly improved as well as the user experience by means of an integrated approach utilizing several methods in concert. Particularly in terms of lowered load times and improved query speeds, the performance gains hint to the possibilities of these techniques to be best practices for developers. Furthermore, the collected data indicates that their combined use generates a more strong and efficient offline architecture even if every technique has advantages and shortcomings.

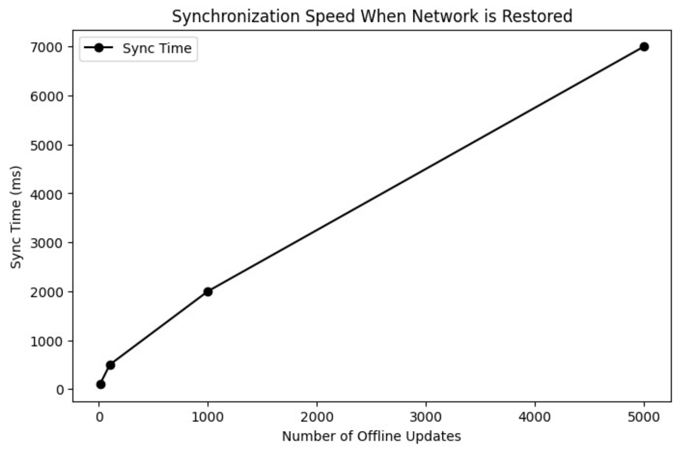


Figure 6: Data retrieval speed analysis under several contexts

These results usually validate the use of current offline data handling techniques in PWAs as well as highlight places where future research and optimization can offer even better performance. The information gained by this study provides developers aiming to produce scalable and high-performance offline web apps useful guidance.

IV.CONCLUSION

With particular attention to the cooperative application of IndexedDB and service workers, this case study thoroughly examined the existing offline data techniques applied in Progressive Web Applications. We have demonstrated by means of considerable experimentation that strategies including pre-caching, lazy loading, pagination, and delta synchronization can dramatically improve the scalability and performance of offline-first Progressive Web Applications (PWAs). Our analysis shows that pre-caching for key assets greatly reduces starting load times. Moreover, by lowering both memory usage and query latency, lazy loading and pagination help to deftly manage large datasets. Moreover, delta synchronizing shows great promise for the effective updating of just changed data, hence reducing bandwidth consumption and accelerating the synchronizing procedure upon reestablishment of the connection.

These methods offer great improvements, but our analysis also emphasizes the challenges of dynamic content management and the complexities in creating successful dispute resolution for synchronizing. Moreover, it is crucial to realize that our controlled studies might not cover all the variation found in actual settings.

In the future, research should concentrate on the improvement of these synchronization techniques by utilizing more sophisticated dispute resolution techniques and by investigating the potential of machine learning to predict data changes. Additional field experiments that integrate real-world user data and diverse network scenarios would also be beneficial in order to verify and expand these findings. Moreover, a careful analysis of the way improved security measures are combined with other storage options could result in notable enhancements in the offline data management. User experience studies are ultimately essential to determine how these technologies affect end-user happiness, hence guiding the development of ever more responsive offline-first apps.

V. REFERENCES

[1] John M Wargo. Learning progressive web apps. Addison-Wesley Professional, 2020.

[2] Andreas Biørn-Hansen, Tim A. Majchrzak, and Tor-Morten Gronli. Progressive web apps: The possible web-native unifier for mobile development. In Proceedings of the 13th International Conference on Web Information Systems and Technologies - Volume 1: WEBIST,, pages 344–351. INSTICC, SciTePress, 2017.

[3] VS Magomadov. Exploring the role of progressive web applications in modern web development. In Journal of Physics: Conference Series, volume 1679, page 022043. IOP Publishing, 2020.

[4] Sayali Tandel and Abhishek Jamadar. Impact of progressive web apps on web app development. International Journal of Innovative Research in

Science, Engineering and Technology, 7(9):9439–9444, 2018.

[5] Chris Love. Progressive Web Application Development by Example: Develop fast, reliable, and engaging user experiences for the web. Packt Publishing Ltd, 2018.

[6] Tal Ater. Building progressive web apps: bringing the power of native to the browser. ” O’Reilly Media, Inc.”, 2017.

[7] Alonge Oluwatobi Josephe, Christos Chrysoulas, Taoxin Peng, Brahim El Boudani, Ioannis Iatropoulos, and Nikolaos Pitropakis. Progressive web apps to support (critical) systems in low or no connectivity areas. In 2023 IEEE IAS Global Conference on Emerging Technologies (GlobConET), pages 1–6. IEEE, 2023.

[8] Bayya, Anil Kumar. (2022). Advocating Ethical Data Management and Security. International Journal of Computer Science Engineering Techniques. 8. 396-417. 10.32628/CSEIT225541.

[9] Phunon Koysawat, Chayanon Boonprakob, Khwantri Saengprachatanarug, Arnut Chaosakul, Panupong Wanjantuk, Mahisorn Wongphati, Santawat Santiteerakul, Adulwit Chinapas, and Kanda Runapongsa Saikaew. Progressive web app for crop field data collection. In IOP Conference Series: Materials Science and Engineering, volume 1163, page 012018. IOP Publishing, 2021.

[10] Frank Mittelbach. A general framework for globally optimized pagination. In Proceedings of the 2016 ACM Symposium on Document Engineering, pages 11–20, 2016.

[11] N van den Wouw, Alexey Pavlov, and Henk Nijmeijer. Controlled synchronisation of continuous pwa systems. Group coordination and cooperative control, pages 271–289, 2006.

[12] Kashish Behl and Gaurav Raj. Architectural pattern of progressive web and background synchronization. In 2018 International Conference on Advances in Computing and Communication Engineering (ICACCE), pages 366–371. IEEE, 2018.

[13] Ramdoss, V. S. (2021). Optimizing database queries: Cost and performance analysis. International Journal of Science and Research Archive, 2(2), 293–297. https://doi.org/10.30574/ijsra.2021.2.2.0025.

[14] He Xiao, Zhenhua Li, Ennan Zhai, Tianyin Xu, Yang Li, Yunhao Liu, Quanlu Zhang, and Yao Liu. Towards web-based delta synchronization for cloud storage services. In 16th USENIX Conference on File and Storage Technologies (FAST 18), pages 155–168, 2018.

[15] Giwon Lee, Haneul Ko, and Sangheon Pack. An efficient delta synchronization algorithm for mobile cloud storage applications. IEEE Transactions on Services Computing, 10(3):341–351, 2015.

[16] Fabian Johannsen. Progressive web applications and code complexity: An analysis of the added complexity of making a web application progressive, 2018.

[17] Felipe Rˆego, Filipe Portela, and Manuel Filipe Santos. Towards pwa in healthcare. Procedia Computer Science, 160:678–683, 2019.

[18] Azzam Sleit et al. Evaluating indexeddb performance on web browsers. In 2017 8th International Conference on Information Technology (ICIT), pages 488–494. IEEE, 2017.

[19] Sang Hyuk Son. Synchronization of replicated data in distributed systems. Information Systems, 12(2):191–202, 1987.