# Web Data Management

(**Under Guidance:** Haim Levkowitz, *Acting Department Chair, Associate Professor Richard A. Miner School of Computer & Information Sciences, Kennedy College of Sciences*, Lowell, MA, haim@cs.uml.edu)

Naga Sai Tejaswi Gandu
*Computer Science*
*University of Massachusetts Lowell*
Lowell, MA
NagaSaiTejaswi_Gandu@student.uml.edu

***Abstract*—The Internet has evolved over the years and has become people's main information source. Every time they need to look up certain information, many individuals use the Web because it is so accessible and common. However, the widespread use of the Internet has created a number of fascinating difficulties. The ever-growing size of the Web in particular makes it more challenging to find, store, and retrieve content on the Web to aid users in finding what they're looking for. In this paper, I will briefly review some of the issues raised by this situation and discuss the research I have done to address them.**

## I. Introduction

The World Wide Web has experienced a tremendous expansion of content during the past years. The World Wide Web was initially created to speed up the exchange of new scientific findings by scientists, but after years of exponential growth, millions of people are utilizing it every day. The abundance of information on the World Wide Web is helpful to a lot of individuals, as seen by its success and spread. Users utilize the Web for a variety of reasons, from casual browsing on a broad subject to targeted studies on a specific issue. The three parts of Web Data Management are Storing, Searching, and Retrieval of Data.

### *A. Storing the Data*

A successful website must keep a number of things in mind. How you save the data on your website is one of the most crucial factors in the modern digital world. Understanding your storage needs is crucial for operational and security reasons. Options for data storage have an impact on analytics and user experience in different aspects [1]. Here are 4 different ways of storing data.

*1) Web Storage:* Web Storage is one of the preferred choices for client-side data. Web Storage allows for a more seamless user experience by storing data in the user's browser. You can save preference settings, localization information, and more with this program, up to 5 MB per storage region. Web storage comes in two flavors: localStorage and sessionStorage. SessionStorage stores information in a single tab and session, making it perfect for transient applications like shopping carts. A seamless cross-session experience is offered by LocalStorage, which distributes data between tabs and windows.

*2) Cloud Storage*: Even though Web Storage enhances user experience, there are situations when you need to store additional long-term data. One of the most popular methods for managing these bigger data repositories is cloud storage.

Private clouds employ on-site servers or specialized third-party data centers that are exclusively used by the company. Websites have more control and receive certain security benefits from this exclusivity, but it frequently results in higher costs.

Public clouds like Microsoft Azure and Amazon Web Services rely on external infrastructure and services. In comparison to private clouds, these solutions are often more accessible and less expensive. As a result, the pandemic might hasten the adoption of public clouds, which was already gaining ground.

Hybrid clouds, as the name suggests, combine resources and services that are owned by both third parties and the company. These options are more adaptable and can help you strike a balance between affordability and use and security and control.

*3) Cookies:* Cookies function similarly to Web Storage in that they temporarily save data in a user's browser. They have a comparable function, gathering information to improve user experiences, but they have a smaller capacity than Web Storage. You probably use cookies, either on purpose or through third-party programs, whether you're aware of it or not. Because cookie policies are often required to tell users about your use of them, privacy regulations cover them and you should be aware of them.

*4) Local Backups:* You can preserve local backups even if the majority of website data storage relies on the internet or browsers. You might want to maintain hard copies if you use website data for analytics.

### *B. Searching for Data*

Search engines can be used to obtain web data. Search engines act as answering machines. They are there to find, comprehend, and arrange the information on the internet so that the most appropriate answers to queries are provided to users. Search engines operate using three main functions.

*1) Crawlers:* Search engines use the search process known as crawling to send out a group of robots, also referred to as crawlers or spiders, to look for new and updated content. They are referred to as "web crawlers" because the technical word for automatically accessing a website and gathering data using software is crawling. Search engines nearly often

control these bots. Search engines can give relevant links in response to user search queries by applying a search algorithm to the data gathered by web crawlers, creating the list of websites that appear after a user types in any search engine [4].

*2) Indexing:* Organizing and storing web content in a large database is a process known as search engine indexing. In order to present readers with ranked lists on its Search Engine Results Pages (SERPs), the search engine must first analyze and comprehend the material. Using "crawlers," a search engine examines links and material on a website before indexing it. Following that, the search engine arranges the stuff it has crawled in its database.

*3) Ranking:* Ranking is the process of arranging search results according to relevance. Generally speaking, you can infer that a website's ranking reflects how pertinent the search engine believes it to be to the user's query. Search engine crawlers can be prevented from viewing portions or all of a website by telling search engines to keep particular pages off of their index.

### C. *Retrieving the Data*

Retrieving particular data items from the server to update portions of a webpage without having to load an entirely new page is another important task in modern websites and applications. Web scraping is a technique for obtaining massive amounts of data from websites and storing it in spreadsheet form in a database or a local file on your device. Using an effective online scraping application that can complete this operation quickly, web scraping minimizes the time required to copy and paste the results. The web scraping software may automatically load and grab data from numerous website pages based on a user's requirements. With only one click, it is straightforward to copy the data from the website to a file on the computer.

## II. Problems & Solutions

### A. *Problem 1: Fetching Text Content*

An HTML page and (typically) a number of extra files, including stylesheets, scripts, and pictures, make up a web page. The files needed to display the page are requested by your browser via one or more HTTP requests, and the server replies with the requested files. This is the underlying idea behind how a web page loads. The browser asks the server for additional files when you visit another page, and the server provides them as shown in Fig. 1 [2].


Fig. 1.

But look at a website that makes extensive use of statistics. Using the Vancouver Public Library website as an example. A website like this could serve as the user interface for a database, among other things. You might use it to look for books in a specific genre or it could give you suggestions for books based on ones you've already borrowed. When you do this, it must refresh the page to show the new collection of books.

Therefore, many websites use JavaScript APIs in place of the conventional paradigm to request data from the server and update the page's content without reloading the page as shown in Fig. 2. Therefore, when a user searches for a new product, the browser just asks for the information required to update the page, such as the list of new books to display.

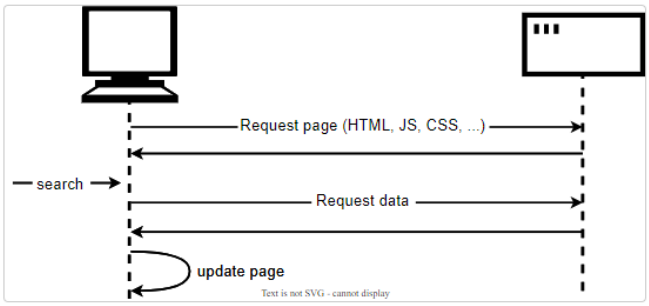


Fig. 2.

The main API used in this case is the Fetch API. This enables JavaScript to use an HTTP request to ask a server for specific resources while it is running on a page. When the server sends the data, JavaScript can use it to update the page, frequently by using DOM manipulation APIs. The requested data is frequently in the JSON format, which is useful for sending structured data, although it can also be in HTML or just plain text.

*Benefits of this type of model:*

*1)* The site feels faster and more responsive since page updates happen much more quickly and you don't have to wait for the page to reload.

*2)* Each update downloads less data, resulting in less bandwidth being squandered. On a PC with a broadband connection, this might not be a big deal, but on mobile devices and in nations without widely available fast internet service, it is a significant problem.

### B. *Problem 2:Hidden Web*

On the Web, a growing amount of information is only accessible through search interfaces. To access Web pages from specific Web sites, users must enter a collection of terms, or queries, into a search interface: The websites don't offer any fixed links that point to their pages. Search engines cannot download these pages, often known as the Hidden Web or Deep Web, because Web crawlers1 only use links on the Web to find pages. I am addressing two methods

concerning to Hidden Web: Hidden-Web meta searcher & Hidden-Web Crawler.

The metasearcher selects the websites that are most likely to contain relevant information, routes the user's query to those websites, and then gathers and returns the search results to the user. My focus is more on Web-crawlers.

*Hidden-Web Crawler:*

The pages from the Hidden-Web sites can be automatically downloaded by the Hidden-Web crawler, allowing us to use well-known indexing techniques to find the collection of pages that are pertinent to a user query. The key challenge for a Hidden-Web crawler is how to find pages in Hidden-Web sites because pages in the Hidden Web could be generated dynamically when a user performs a query. That is since there are no fixed links to Hidden Web pages, the crawler must autonomously build and send out queries to find them. Given that the crawler is unable to comprehend the semantics of a query interface, automatic query generation is undoubtedly challenging.

Time and network resources are frequently scarce for Hidden-Web crawlers. Because of this, a crawler must carefully choose and formulate keyword queries in order to download the most pages with the fewest number of resources. It may use up all of its resources if it issues utterly random queries that don't yield any matching pages and instead just returns a list of results. This query-selection issue can theoretically be described as a minimum-cover issue in a graph. To put it another way, we presumptively a crawler retrieves pages from a website that contains a set of pages S. Each Web page is represented in S as a graph vertex like in Fig. 3 [3]. The crawler's possible query qi is likewise represented by us as a hyperedge in the graph. When the crawler sends qi to the site, it returns vertices (pages), all of which are connected by a hyperedge qi. Additionally, each hyperedge has a weight that represents the expense of executing the query. In accordance with this formalization, our challenge is to choose the set of hyperedges that cover the greatest number of vertices while carrying the least overall weight.

This formalization is complicated by two key issues. First, the hyperedges of the graph are unknown since a Hidden-Web crawler does not know which Web pages will be returned for a query. The crawler cannot choose them without being aware of the hyperedges. In addition, it is unknown whether an effective method exists to tackle the minimum-cover problem in an optimal way given that it is known to be NP-hard [3].

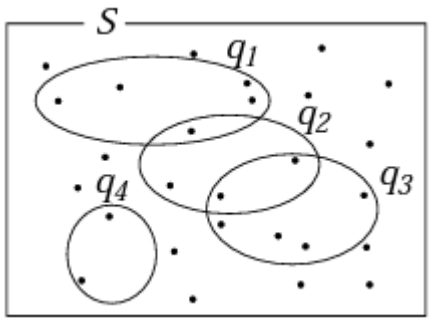


Fig. 3.

## III. Conclusion

In this paper, I briefly went over existing challenges that the web has brought us today. I also addressed two problems: Fetching text content and Hidden Web Crawlers.

## References

[1] AuthorDevin PartidaDevin Partida is the Editor-in-Chief of ReHack.com. She covers the internet. "5 Data Storage Concepts Serious Website Owners Should Understand." *Torque*, 4 Dec. 2020, https://torquemag.io/2020/12/5-data-storage-concepts-serious-website-owners-should-understand/.

[2] "Fetching Data from the Server - Learn Web Development: MDN." *Learn Web Development | MDN*, https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Fetching_data.

[3] *Frontiers in Web Data Management - University of California, Los Angeles*. http://oak.cs.ucla.edu/~cho/papers/cho-icts03.pdf.

[4] "How Search Engines Work: Crawling, Indexing, and Ranking - Beginner's Guide to Seo." *Moz*, https://moz.com/beginners-guide-to-seo/how-search-engines-operate.

[5] Morris, Will. "The Basics of How Search Engine Indexing Works." *Elegant Themes Blog*, 2 Oct. 2022, https://www.elegantthemes.com/blog/wordpress/how-search-engine-indexing-works#:~:text=Search%20engine%20indexing%20refers%20to,Engine%20Results%20Pages%20(SERPs).

[6] Suciu, Dan. "Learn Web Scraping: How Can You Extract Internet Data." *Medium*, Medium, 27 May 2021, https://dan-suciu.medium.com/learn-web-scraping-how-can-you-extract-internet-data-b08677c33aae.

[7] *What Is a Web Crawler? | How Web Spiders Work | Cloudflare*. https://www.cloudflare.com/learning/bots/what-is-a-web-crawler/.