An Initial Design of a Website Snapshot Management System

David Chao
College of Business
San Francisco State University
San Francisco, CA 94132
E-mail: dchao@sfsu.edu

Abstract: A website snapshot is the state of a web site at a specific point in time. It supports applications that require historical data. Most website snapshots are created by making a copy of the website. These date-time stamped physical snapshots are unable to satisfy users' need for snapshots of different snapshot times. This research proposes a scheme that is able to create website snapshots that meet any snapshot time requirements by recording changes to a website in a log. For web pages producing dynamic content from a database, this scheme will allow the pages to access a database snapshot at the website snapshot time.

I. Introduction

A website snapshot is the state of a web site at a specific point in time. It supports applications that require historical data and preserves website content for future reference. In practice, some website snapshots are created to answer questions about website content at the snapshot time for audit and compliance purposes [5]. Some are created due to government policies for preserving government-supported Web content [3]. Some are created in an attempt to preserve the web content as an archive of civilization for future generations to study. Creating and preserving web site snapshots have been ongoing undertakings of many web projects. An example is the Internet Archive which uses web crawler to collect all open access web pages since 1966. Its Wayback Machine lets users view snapshots of Web sites as they appeared at various points in the past [2]. Some are created for research and analyses that use snapshots to study certain trends in the study subject. Website snapshots enable researchers to analyze data with new tools that haven't existed before [2].

In addition to providing historical data, website snapshots are also used in web server's cache engines. The goal is to save bandwidth and improve download time by delivering data from the cache that otherwise would make repeated trips across the Internet. Similarly, popular internet search engines such as Google also create a cache of website snapshots in order to speed up searching [11]. Website snapshots can also be used as a backup of the operating website.

Website snapshots may exist in many forms. They may be accessible online as a server-side resource, downloaded to a user's machine, or created on a disk for offline access. A typical way of creating a server-side website snapshot is using a program that copies files from the Web server to a computer at an archive. The snapshot may contain all the files at the site or a subset. Snapshots are taken periodically, so that the archive has a sequence of snapshots for each site [9]. A client-side website snapshot can be created by using a website downloading software; an example is Web Site Downloader [11]. This type of software lets users download an entire website, or a selective part of it, for offline viewing.

Although the process of creating a website snapshot is simple, it may still face some practical problems [1]. The state of a website changes whenever a change occurs to the website. Most website snapshots created today exist physically as a date-time stamped copy of the website. A major problem with the physical snapshot is that it only represents the website's state at one specific time. It is unable to satisfy a user's request for a snapshot of a different time. Dynamic web pages pose another problem because they use server-side and/or client-side script code to create dynamic content. These pages may also retrieve information stored in databases. Thus, retrieving the page alone may not yield the desirable result.

This research proposes an initial design for a website snapshot management system. The system is able to: (1) generate a website snapshot with any snapshot time specified by a user, (2) generate a website snapshot only when requested, and (3) provide database snapshot that is consistent with the website snapshot time to create dynamic content. The system is designed to manage the website snapshots of only one website.

II. Tracking Changes to Website State

II. 1 The Needs for Tracking Changes

A web site is a system of integrated web documents where each document is identified by a URL. Therefore, the state of a website represents the state of all web documents of the website, and it changes whenever the website changes its content by inserting a new document, deleting an existing document, modifying a current document, relocating a document to a new directory, or reorganizing its directory structure.

A website as currently published is the result of past changes; it represents the current state of the website. Tracking all changes to a website to its current state enables a website to rollback its current state to any state in the past. In so doing, many non-current documents need to be kept; these include deleted documents and old versions of a

Proceedings of the Fifth International Conference on Electronic Business, Hong Kong, December 5-9, 2005, pp. 105 - 108.

modified document. All old versions of a web document are its snapshots at various times in the past. The URL of a current document may also go through many changes due to a renaming or relocating of the document.

Figure 1 summarizes the effects of web site reorganization and change to a document on its URL. When a web site reorganizes its directory structure, all affected documents will have a new URL. Similarly, when a web document is renamed or relocated to another directory, it will have a new URL as well. When a document is modified, its URL will not change and the old version before the modification becomes the document's snapshot between the previous and the current modification. In order to retrieve the snapshot the old version is archived. A deleted document is archived as well for later retrieval.

Web site actions	**Effects on current and historical links**
Adding a new document	Adds a new URL to current links
Modifying a document	No change to URL; the old document becomes a snapshot and is archived
Deleting a document	Deletes a current link; adds a historical link and document is archived
Renaming a document	Adds a new URL to current links; the old URL becomes historical link
Relocating a document	Adds a new URL to current links; the old URL becomes historical link
Reorganization	Adds all affected documents' URLs to current links; all affected documents' old URLs become historical

Figure 1: Effects of eb site reorganization and web document changes.

Together, the URLs invalidated by a web site due to reorganization, document removal, renaming, or relocation, plus the links to document snapshots are a web site's *historical links*. From this perspective, a website consists of two parts: links to current documents and links to historical documents..

II. 2 Factors Affecting Website State

A website state captures not only the code that defines the web documents but also the presentation of the web documents as a results of running the code [2]. Thus, factors affecting the rendering of web documents will affect website state. These factors include:

a. Web document code: This includes the code that creates web page's static and dynamic contents including server-side and/or client-side scripts. Frequently, code creating dynamic content is stored separately in a different file as in the case of Server-Side-Include, Code-Behind technologies, and XML document style sheet.

b. The state of internal resources it references: Internal resources are files managed by a web site and are available in creating the web site's contents. Typical examples of internal resources referenced by a web document that affect its rendering are style sheets, files embedded by a Server-Site include directive, image files, script files, and databases. These internal resources may create dynamic contents that change every time the document is opened, and they are subject to change that consequently changes the web document's rendering. Many of these files are *internal supporting files* that are created for supporting a web document and are not for publishing individually.

c. The state of external resources it references: External resources are files not managed by the web site but can be referenced in creating the web site's contents. A web document may reference external style sheets, components, web services, or databases. These external resources are also subject to change.

d. Web site host environment variables: Script code may reference a host's system variables in creating dynamic content. A typical example is using the system clock to get the current date and time. A web document that displays the current date and time is always in a new state.

e. Web technologies implemented on the server-side as well as on the client-side: A web page is often created by using many technologies and may reference documents that are created by a different set of technologies. The complete rendering of a web document requires that all technologies, client-side and server-side, are properly implemented.

The first three factors affect the consistency of the data of a web document and are called *consistency* factors. The last two factors are *environment* factors. Figure 2 shows the relation between a web document and factors affecting its rendition. Note that a web page may be a composite document of several tightly-coupled documents, such as style sheets and code-behind files.

Based on the consistency factors' effects, three levels of a web document's snapshot can be defined. (1) Level 1 snapshot: A web document snapshot is the state of web document code at snapshot time. Creating a level 1 snapshot enables a web site to trace the changes to the web document code over time. (2) Level 2 snapshot: A level 2 snapshot is a level 1 snapshot with the additional requirement that all the internal resources it references are at least level 1 snapshot at the same snapshot time. These internal resources can be categorized into two groups: database and non-database files. For a database file to be a level 1 snapshot, it must be consistent with the state of the database at the snapshot time. (3) Level 3 snapshot: A level 3 snapshot is a level 2 snapshot with the additional requirement that all the external resources it references are at least level 2 snapshots at the snapshot time.

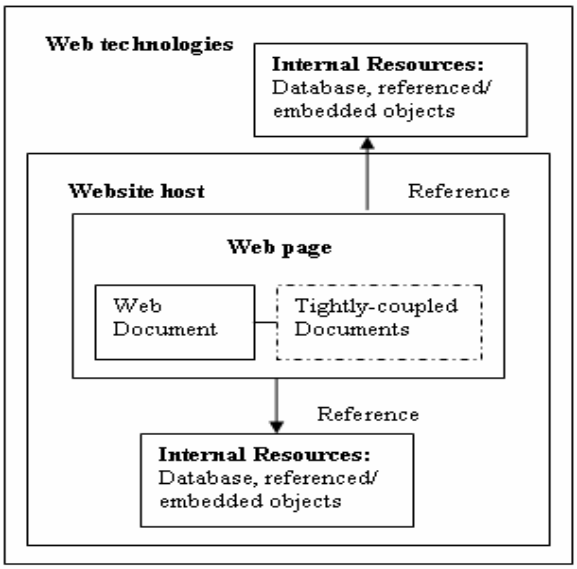

Figure 2: Relations between a web page and factors affecting its rendition.

The two environment factors will improve the consistency of a web document snapshot if they are enforced; that is, the host variables are reset to snapshot time and web technologies implemented are compatible with the technologies at the snapshot time. Four levels of consistency improvement are defined depending on whether the environment factors are enforced: (1) Plus 0: If both environment factors are not enforced. (2) Plus 1: If the host variables are reset to the snapshot time. (3) Plus 2: If web technologies are compatible with the technologies at the snapshot time. (4) Plus 4: If both factors are enforced.

Consistency factor

Environment Factor	Level 1	Level 2	Level 3
Plus 0	Level 1 + 0	Level 2 + 0	Level 3 + 0
Plus 1	Level 1 + 1	Level 2 + 1	Level 3 + 1
Plus 2	Level 1 + 2	Level 2 + 2	Level 3 + 2
Plus 3	Level 1 + 3	Level 2 + 3	Level 3 + 3

Table 1: Levels of website snapshot

Table 1 summarizes all possible levels of snapshot states. The level 1 and level 2 snapshots are related to files that are managed by the website and changes to these files can be recorded by the website; hence, they are relatively easy to achieve. The level 3 snapshots involve resources that are not managed by the web site and it's difficult, if not impossible, for the web site to keep track of changes to these resources. This research develops a web document snapshot management system to deliver web documents' level 1 and level 2 snapshots.

III. Initial Design of Website Snapshot Management System

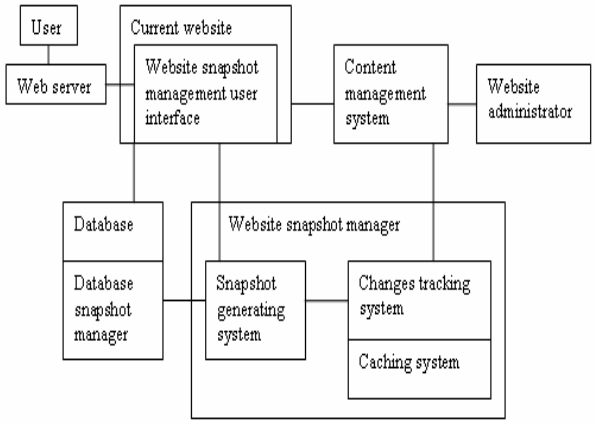

Figure 3: Components of a website snapshot management system.

This system consists of two modules: Database Snapshot Manager for maintaining database snapshots and Website Snapshot Manager for maintaining web document snapshots. Figure 3 shows the major components of this system in a typical website environment. This system assumes that the website administrator uses a content management system in maintaining the website. Changes to the website as discussed in the previous section are passed from the content management system to the Website Snapshot Manager.

The Website Snapshot Manager contains 3 components: (1) Changes Tracking System: This system uses a log to record all website changes including insertions, deletions, modifications of web documents, and changes to web document's URL. (2) Caching System: This system is responsible for storing deleted documents and old versions of current web documents. (3) Website Snapshot Generating System: This system generates a website snapshot based on a user's requirement for snapshot time.

The objective of the Database Snapshot Manager is to provide a database snapshot at any snapshot time requested by users. This requires recording all updates in a log. The log uses time stamp to record update times and uses flags to indicate deletions and insertions where a modification is treated as the deletion of the old version followed by an insertion of the new version. With such an update log available, rolling back the current database using updates with time stamp later than the snapshot time can generate a snapshot at any snapshot time. Database snapshot management and related materialized view management have been the topics of much research [3][4] [8] [9].

The current website contains a user-accessible module of snapshot management user interface. This interface lets users submit a request for a website snapshot. This request translates to a command: Create Website Snapshot As Of *snapshot time*.

IV. Summary

Website snapshots have been used in many applications. Most website snapshots are created by making a copy of the websites. These date-time stamped physical snapshots are unable to satisfy users' need for snapshots of different snapshot times. This research proposes a scheme that is able to create website snapshots that meet any snapshot time requirements by recording changes to a website in a log. For web pages producing dynamic content from database, this scheme will allow the pages to access database snapshots at the website snapshot time.

References

[1] Arms, W., , R., Adkins, Ammen, C., and Hayes, Allen, 'Collecting and Preserving the Web: The Minerva Prototype' , RLG DigiNews, Vol. 5, Number 2, 2001, http://www.rlg.org/preserv/diginews/diginews5-2.html#feature1

[2] Bentley Historical Library, University of Michigan, University Archives and Records Program, http://www.umich.edu/~bhl/bhl/uarphome/digipres.htm

[3] Chao, D., Diehr, G., & Saharia, A. (1996) Maintaining Join-based Remote Snapshots Using Relevant Logging. Proceedings of the Workshop on Materialized Views, ACM SIGMOD, Montreal, Canada, 1996.

[4] Chien, S., Tsotras, V., & Zaniolo, C. (2001) Efficient Management of Multiversion Documents by Object Referencing. Proceedings of 13th International Conference on Very large Data Bases, 2001.

[5] EBusinessWare Case Study, 'Record Retention: Website Snapshot Spider', http://www.ebusinessware.com/ebw-CaseStudy-RecordsRetention.pdf

[6] Google Answers: Q Cache Engine Questions, http://answers.google.com/answers/main?cmd=threadview&id=237111

[7] Internet Archive, 'WayBack machine', **http://www**. archive.org/

[8] Labrinidis, A. & Roussopoulos, N. (2000). Webview Materialization. ACM SIGMOD International Conference on Management of Data, May 14-19, 2000.

[9] Marian, A., Gregory Cobena, S., & Mignet, L (2001) Change-Centric management of Versions in an XML Warehouse. Proceedings of the 27th VLDB Conference, Rome, Italy, 2001.

[10] The National Archives and Records Administration, 'Web Harvest and Web Snapshot Information', http://www.archives.gov/records-mgmt/policy/web-harvest-snapshot.html

[11] Web Site Downloader, http://www.web-site-downloader.com/entire/?int=GGL