Winter 12-10-2002

The Design of a Web Snapshot Management System for Decision Support Applications

David Chao

The Design of a Web Snapshot Management System for Decision Support Applications

David Chao
College of Business
San Francisco State University
San Francisco, CA
dchao@sfsu.edu

Abstract

Database snapshots that are defined and/or delivered via the Web are called web snapshots. This paper addresses the requirements for web snapshot management. A web snapshot management system is proposed; its architecture and functions of the major components are described; new commands are defined to perform web snapshot management activities.

1. Introduction

Providing information to support decision-making is a major database application on the World-Wide Web. Many web sites let external users query their databases. Frequently accessed queries can be predefined and retrieved on personalized pages. Internally, Intranets are being used as a platform within an organization for improving communication, information sharing, and for developing business applications to support managerial decision-making. Many companies develop Enterprise Information Portal (EIP) to integrate Intranet applications. Decision support databases and tools such as web-enabled Online Analytical Processing (OLAP) tools are major components of EIP [7]. With the development of decision support activities on Intranet, an easy access to the decision support databases via Intranet is essential to the success of such development.

Database snapshots are an effective way to support predefined and recurrent queries for decision support. They are read-only copies of a selected portion of the database representing the state of the database, and a user's view of the operational data at a fixed point in time (snaptime) [1]. A DEFINE SNAPSHOT statement that defines the snapshot contents initiates a snapshot. In Structure Query Language, SQL, this command might look like:

DEFINE SNAPSHOT <snapshot-name>
AS <query>
AS OF <snaptime>

where <query> can be any valid SQL SELECT command. Database records that satisfy the query are said to be relevant or qualified to the snapshot. Relevant records are materialized and stored under the snapshot-name. A snapshot is read-only from the user's point of view to maintain its consistency with the database at the snaptime. As updates occur to the database, the snapshot will eventually become "stale" and its value for decision-making will diminish. To bring a snapshot to a new state consistent with the database, the user issues a REFRESH command. In SQL this command is:

REFRESH SNAPSHOT <snapshot-name>
AS OF <new snaptime>

Thus, snapshots are similar to "materialized views", however, with the following major differences: 1. Materialized views are created to efficiently answer queries with real time data [4]. Unlike materialized views, snapshots are not designed to provide real time data. 2. User determines the snaptime of a snapshot. In decision support systems users may either prefer not to use, or should not be allowed to use real time data. Many such applications call for historical or end-of-period data. Thus, snapshots are created mainly for decision support applications. 3. Materialized views are updated automatically either by an immediate or deferred update scheme deemed optimal by the system. Snapshots are "refreshed" only when the user explicitly issues a refresh request.

A decision support database (DSDB), such as a data warehouse, is essentially a generalization of the snapshot concept. A DSDB contains data from various sources including internal and external data. Data from these sources are first extracted, cleaned, and then integrated before loaded to the DSDB [5]. A DSDB is read-only, and holds information that is consistent with the various data sources as of a specific point in time. Updates to the DSDB will be done according to a predefined schedule. Thus, a DSDB is a snapshot of the source databases. Its content remains static between two refreshes so that users involved in decision support and analysis activities can work with a relatively static copy of the database.

Database snapshots that are defined and/or delivered via the Web for decision support applications are called *web snapshots*. Figure 1 depicts the environment and major components involve with web snapshots. Web snapshots are defined on a DSDB. The DSDM maintenance algorithm takes updates from various data sources to

Table 1: Possible combinations of web snapshot materialization and delivery options

Delivery Options	Materialization Options: Database File	HTML File	XML File	OtherFile Formats	Virtual Snapshot
HTMLFile	Conversion	Ready	Conversion	Conversion	Querying & Formatting
XML File	Conversion	Conversion	Ready	Conversion	Querying & Formatting
Other File Formats	Conversion	Conversion	Conversion	Ready/ Conversion	Querying & Formatting

update the DSDB and web snapshots. A web snapshot manager is in charge of the snapshot management activities, such as defining and refreshing snapshots, and delivering web snapshots to the web server. The management of web snapshots is quite different from that in a traditional database system. Some major differences and new requirements are discussed in the following.

Virtual And Materialized Snapshots Snapshots defined on a traditional database must be materialized to maintain the consistency at the snaptime. A DSDB is updated by a periodical maintenance schedule, and is static between two updates. Hence, it is consistent with its source databases at the time of the last maintenance. Therefore, snapshots that require the same consistent point as the DSDB do not have to be materialized to maintain their consistency. Hence it is possible to have *virtual* snapshots defined on a DSDB.

Snapshot Location A web snapshot can be easily downloaded to a user's computer as a client-site snapshot. A web page can be saved as a local page. File formats supported by browsers can be opened and/or saved as a local file. File formats not supported by browsers can be downloaded without opening. A client-site snapshot is locally available and saves communication costs if accessed frequently. In a wireless environment client-site snapshots can be implemented as a mobile data warehouse to support mobile agents [6].

Materialization And Delivery Options In a traditional database system, snapshots are typically materialized at a centralized location and are materialized as a regular database file. A web snapshot may be delivered as an HTML web page, an XML page, or other file formats such as a spreadsheet. Table 1 illustrates the possible combinations of snapshot materialization and delivery options. A materialized snapshot may be *ready* to be delivered, or requires *conversion* to the target format. Virtual snapshots need to be generated by querying the database, and format the results to the target format. In practice, a snapshot's delivery format may be different from its materialization format. A user may view the snapshot in an HTML format with a browser and download it as a spreadsheet.

Personalization Web sites today typically provide personalized web pages to users. A snapshot may be embedded or appear as link in a personalized web page. A web snapshot manager must keep track of users and their snapshots in order to create personalized pages.

Snapshot Refresh Options A snapshot is usually refreshed in response to a user's refresh request. Such a refresh is called *pull refresh* [2]. Alternatively, a snapshot can be refreshed by a *push refresh* where the snapshot manager refreshes a snapshot automatically and still maintains the user's requirement for consistency. This can be done if a snapshot's snaptime is implicitly changing with time. For example, a user may require the snapshot to be one-day old; the server is able to update the snapshot even without the refresh request.

Large Number of Users and Small Number of Snapshot Definitions Web sites typically present forms with interface controls to users to define queries by clicking pre-entered values in the controls. This implies that web sites can predetermine the kind of snapshots that will be defined. The number of unique values in those controls is small when compared with the number of web site users. Consequently the number of unique snapshot definitions that can be constructed is small. Therefore, the ratio of the number of snapshot users to the number of unique snapshot definitions tends to be large. It is very likely that each snapshot may associate with many users. Hence a materialized snapshot may be used to support many users.

This paper proposes a web snapshot management system. This system implements new snapshot management commands that incorporate requirements discussed above. The architecture of the web snapshot manager and the functions of its components will be described. From a user's perspective, managing snapshots involves

defining, accessing, refreshing and deleting snapshots. These commands will be defined in the next section.

2. Defining Web Snapshot Management Commands

A web snapshot is the realization of the function S(SC(Q, T, D(T)), SM(O, N, L, {F}, R)) where SC(Q, T, D(T)) is a set of parameters specifying the contents of the snapshot, and SM(O, N, L, {F}, R) is a set of parameters specifying the management of the snapshot. The meaning of these parameters is explained below:

SC(Snapshot Contents)
Q: Query operations and logical conditions
T: Snaptime
D(T): Database state at T

SM(Snapshot Management)
O: Snapshot owner
N: Snapshot name
L: Snapshot location
{F}: Snapshot formats where {} indicates repetition
R: Snapshot refresh option

The following DEFINE SNAPSHOT command lets users to enter parameters:

DEFINE SNAPSHOT <*snapshot-name*>
AS <*query*>
[AS OF *snaptime*]
[OWNER IS *system* | *username*]
[MATERIALIZE AT *client-site*| *server-site*]
[{DELIVER AS *file format*}]
[REFRESH BY *push* | *pull*]

In the syntax, clauses enclosed in the brackets are optional; clauses enclosed in the braces may be repeated; words in italic separated by the vertical bars are possible choices for the parameters.

The AS OF clause lets users specify the snaptime. Since the DSDB is basically a snapshot of the source databases, the DSDB is consistent with the source databases at the time the DSDB was last updated in a scheduled maintenance. Without the AS OF clause, the DSDB's consistent point is assumed.

The OWNER IS clause specifies the user of the snapshot. Without it, the snapshot is system-owned. Only registered users can issue DEFINE SNAPSHOT command.

The MATERIALIZE AT *client-site* clause specifies user's intention to download the snapshot to local site. Note that the snapshot may still be materialized at the server. Without it, the server-site is assumed.

The DELIVER AS lets users specify the format of delivery such as an HTML or XML page. Although users may specify a delivery format, the actual materialization may be different. A snapshot manager may consider other factors (discussed in the next section) in choosing a format for materialization. This clause may be repeated which indicates users may request multiple delivery formats. Hence, it is possible to enter the following clauses in a DEFINE SNAPSHOT command:

DELIVER AS xml
DELIVER AS spreadsheet

The REFRESH BY clause specifies the snapshot to be refreshed by a *push* or *pull* refresh. Without it, the *pull* refresh is assumed.

Refreshing Snapshots: The REFRESH command refreshes the snapshot to the new snaptime specified in the AS OF clause.

REFRESH <*snapshot-name*>
[AS OF *snaptime*]

A forward refresh is performed when the new snaptime is later than the old snaptime; otherwise a backward refresh is performed. Without the AS OF clause, the DSDB's consistent point is assumed. Note that the REFRESH command applies only to snapshots on the server. To refresh a client-site snapshot, the following RETRIEVE command is proposed.

Retrieving Snapshots The RETRIEVE command delivers a snapshot with the specified format which may be different from its original materialized format.

RETRIEVE <*snapshot-name*>
[{AS *file format*}]

The AS clause may be repeated. Without it, the web page is assumed. Hence, it is possible to enter the following clauses in a RETRIEVE command:

AS xml
AS spreadsheet

Deleting Snapshots: The delete command is simply:

DELETE <*snapshot-name*>

3. Components of a Web Snapshot Manager

This section describes the architecture of the web snapshot manager. Figure 2 depicts the major components of the web snapshot manager and the interface among them. It takes snapshot management commands as inputs and delivers requested snapshots as outputs to the web server. The functions of these components are discussed in the following.

Authorization Control and User Registry This module maintains user directory and checks that the user has the authorization to issue commands. Only registered users can issue snapshot management commands. Validated commands are passed to the Command Processor.

Command Processor This module analyzes the commands, determines their action, and passes them to appropriate components for execution.

Snapshot Catalog This module maintains information about web snapshots found in the DEFINE SNAPSHOT command including parameters Q, T, O, N, L, and F, R described earlier. It also maintains a link to the snapshot once it is materialized. This information is needed to support the RETRIEVE and REFRESH SNAPSHOT commands. It updates a snapshot's snaptime once it is refreshed. Information of the deleted snapshots will be removed from the catalog.

Materialization Optimizer This module takes snapshot definitions from the catalog and determines the optimal materialization plan. Managing snapshots incurs the cost of generating, refreshing, and delivering snapshots. Generating a snapshot may incur the cost of querying the database and formatting the results to a target format. Refreshing a snapshot may be done by performing a differential refresh or by regenerating the snapshot. Delivering a snapshot may involve the cost of reading and formatting the snapshot to a target format. Without describing the optimizer in details major issues involved in deciding an optimal plan are discussed in the following.

. Choosing a criterion for optimization: A typical criterion in selecting an optimal snapshot materialization is to minimize the total snapshot management costs. For web snapshots, however, reducing the response time of delivery is important in maintaining the quality of web site service.

. Choosing between materialization and virtual implementation: Not all web snapshots require materialization. The frequency of accessing a snapshot is an important parameter in making this decision. A frequently accessed snapshot typically requires materialization.

. Choosing between one materialized format and multiple formats: The DEFINE SNAPSHOT and the RETRIEVE SNAPSHOT commands let users specify multiple delivery formats. Materializing in multiple formats will reduce the response time but may require more refresh costs and vice versa.

. Choosing between using one materialized copy to support one snapshot and multiple snapshots: Snapshots with identical (or significantly overlapping) definitions and with the same consistency requirement can be supported more efficiently with a single materialization.

Refresh Optimizer This module processes the REFRESH commands. The major functions of this module are:

. Choosing an optimal refresh method: A snapshot can be refreshed by a full regeneration or by a differential refresh. Typically, a frequently refreshed snapshot or a snapshot with only a few updates can be refreshed more efficiently with a differential refresh.

. Maintaining update log: The REFRESH command lets users to specify a new snaptime, which may be before or after the current snaptime. In order to support this refresh capability, the optimizer must maintain a log of updates. Updates to a base table are either deletions or insertions if modifications are treated as the deletion of the before-image followed by the insertion of the after-image. A typical log design is: (TimeStamp, UpdateFlag, UpdatedRecord) where the TimeStamp records the update time and the UpdateFlag is a flag indicating deletion or insertion [3]. This type of log keeps updates in chronological order. Since snapshots are defined on a DSDB, the updates gathered by the DSDB maintenance algorithm can be used to refresh snapshots.

. Generating refresh messages: Updates between the old snaptime and the new snaptime can be generated from the log. These refresh messages are sent to the materialization optimizer to refresh the materialized copy.

Once the refresh is done, the refresh optimizer will notify the snapshot catalog to update the snaptime.

Retrieve and Delivery Module This module processes the RETRIEVE commands. It retrieves the snapshot from the materialization optimizer, converts it to the format requested by the user, and delivers it to the web server.

4. Summary

This paper addresses issues and requirements for the management of web snapshots. Database snapshots that are defined and/or delivered via the Web are called web snapshots. They are used mainly for decision support applications in a Web environment. New commands are defined to perform web snapshot management activities. A web snapshot management system is proposed; the functions of the major components in this system are described.

References

[1] Adiba, M. & Lindsay, B. (1980). Database snapshots. Proceedings of the 6th International Conference on Very Large Data Bases, pp. 86-91.

[2] Aksoy, D., Franklin, M., Zdonik, S. (2001). Data Staging for On-Demand Broadcast. Proceedings of the 27th VLDB Conference, Roma, Italy, 2001

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

[5] Gray, P., Watson, H. (1998). Decision Support in the Data Warehouse. Prentice Hall, 1998

[6] Lee, K., Si, A., and Leong, H. (1998). Incremental View Update for a Mobile Data Warehouse. ACM Symposium on Applied Computing, Atlanta, 1998

[7] Murray, G. (1999). Making Connections with Enterprise Knowledge Portals. White Paper, Computerworld, Sept. 6, 1999

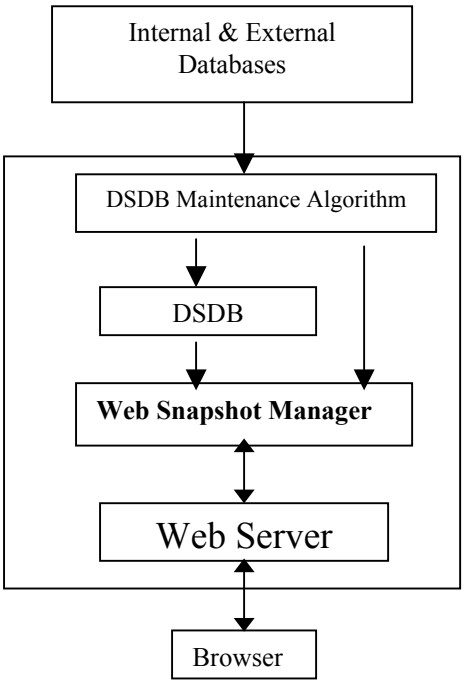

Figure 1: Components of web snapshot management for decision support.

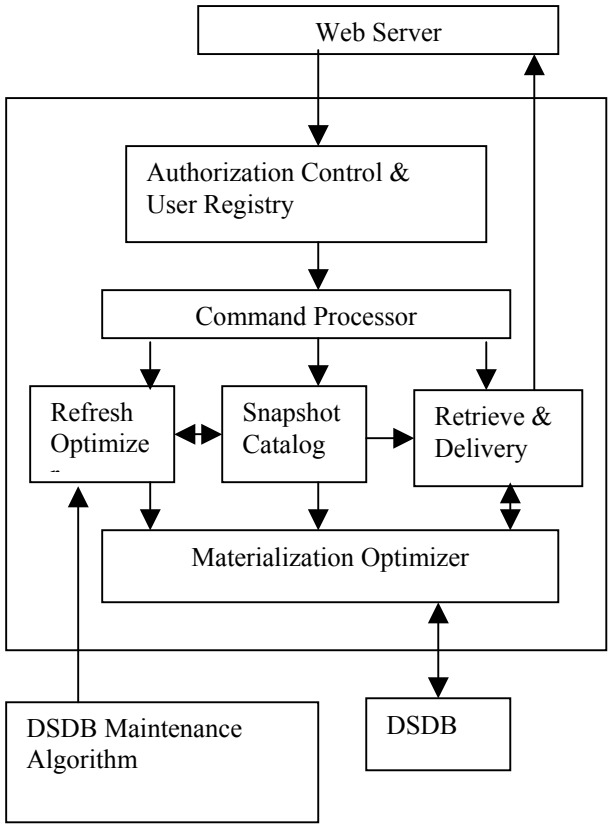

Figure 2: Components of a web snapshot manager