S-OLAP: an OLAP system for analyzing sequence data

Chun Kit Chui #1, Ben Kao #2, Eric Lo *, David Cheung #3

#Department of Computer Science, The University of Hong Kong
{1ckchui, 2kao, 3dcheung}@cs.hku.hk

*Department of Computing, The Hong Kong Polytechnic University
ericlo@comp.polyu.edu.hk

ABSTRACT

The Sequence OLAP (S-OLAP) system is a novel online analytical processing system for analyzing sequence data. S-OLAP supports "pattern-based" grouping and aggregation on sequence data — a very powerful concept and capability that is not supported by traditional OLAP systems. It also supports several new OLAP operations that are specific to sequence data analysis. The query processing techniques documented in [1] have been implemented in our S-OLAP engine for efficient query processing. The system also provides users with a friendly graphical interface for query construction and result visualization. Query parameters can be interactively refined and the results are updated in real-time so as to facilitate the exploratory analysis of sequence data.

1. INTRODUCTION

In this demonstration we present the Sequence OLAP (S-OLAP) system – a novel on-line analytical processing system for warehousing and analyzing sequence data. The S-OLAP system has several innovative and distinctive features. First, while traditional OLAP systems group data tuples based on their *attribute values*, S-OLAP system treats patterns as dimensions and it groups sequences based on the *patterns* they possess. Common aggregate functions such as `COUNT` can then be applied to each group. The resulting aggregate values form the cells of a so-called sequence data cuboid, or *s-cuboid*. Second, S-OLAP supports several high-level OLAP operations that are specific to sequence data analysis. These operations allow data analysts to explore and to navigate among different levels of summarization (s-cuboids) of sequence data. The collection of s-cuboids forms a *sequence data cube*.

2. DEMONSTRATION SCENARIO

The demonstration begins with a brief introduction to the S-OLAP system, including its major concepts, innovative features and architecture design. It then proceeds by highlighting the system features through an on-line click-stream data analysis application scenario. Online stores maintain web servers that record the click events performed by their users. Figure 1 presents a click events dataset from a web server log of an on-line retail store. We assume that a user registers an event/transaction into the system every time he browses a web page (the server receives a page request from a client browser). An event consists of a number of *dimension attributes*, such as *Time*, *Session ID*, and *Product* (the product that was displayed on the requested web page). Similar to conventional data, an attribute may be associated with a *concept hierarchy*. In this demo application, the *product* dimension is associated with a concept hierarchy of three abstraction levels individual-product→product-type→brand. The click events performed by a user can form logical sequences in many different ways. For example, a sequence can be formed by clustering a user's click events over 1-day, 1-week or 1-month periods.

time	*session ID*	*url*	*product*
090101T00:01	688	/product.jsp?id=128	Nike Shoes - Air Max 90iD
090101T00:08	688	/product.jsp?id=132	Adidas Sportswear - Adizero Singlet
090101T00:09	14230	/product.jsp?id=324	Puma Shoes - King FG
090101T00:31	688	/product.jsp?id=128	Nike Shoes - Air Max 90iD
...	...	...	...

Figure 1: An event database

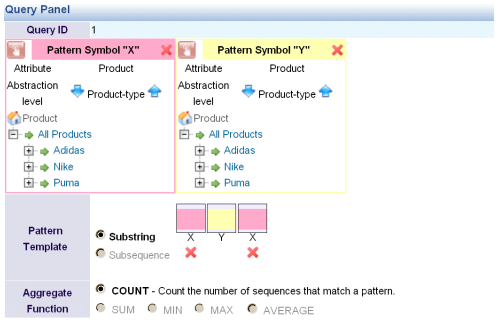

Figure 2: The query panel of the S-OLAP interface system

2.1 Constructing the first S-cuboid

With the enormous amount of sequence data available, the S-OLAP system that performs sequence summarizations would be of great value. For instance, if a marketing manager wants to learn about the shopping behavior of the website users, he may want to know "the number of users that have performed *comparison shopping* and their distributions over all product pairs". With the S-

This research is supported by Hong Kong Research Grants Council GRF grants HKU 713008E.

Figure 3: The s-cuboid panel of the S-OLAP interface

OLAP system, the participants can play as the role of the manager of the online-store and specify an S-OLAP query "(X, Y, X), COUNT" through the interface as shown in Figure 2. An S-OLAP query basically consists of two components [1] : A pattern template T (e.g., (X, Y, X)) and an aggregate function F (e.g., COUNT). For example, the pattern template (X, Y, X) defined on the *Product* attribute captures the *comparison shopping* semantics, which specifies that click sequences are grouped together if the user first browsed certain product X (e.g., a page displaying an Adidas shoes product), followed by certain product Y (e.g., a page displaying a Nike shoes product), and then went back to X (a page displaying an Adidas shoes product) again. Here, X and Y are called *pattern symbols* and they are instantiated with values of an attribute at a given *abstraction level* (*Product* AT the Product-type abstraction level in the example). Each instantiation of the pattern template (X, Y, X), such as (Adidas shoes, Nike shoes, Adidas shoes), gives a pattern. Data sequences that contain a given pattern are grouped into a cell. Each data sequence gives a value (or *measure*) to be aggregated. For example, a click sequence could be associated with the amount of purchase made, or the number of products the user has browsed, or simply the number of user sequences that contain the pattern. An aggregation function F is then applied to the values of the sequences of each cell to obtain an aggregate value of the cell. After query processing, the tabulated summary (i.e. an s-cuboid) as shown in Figure 3 will be displayed.

2.2 Interactive and exploratory data analysis

Supporting interactive and exploratory analysis of sequence data is another key feature of the S-OLAP system. It supports the following six high-level S-OLAP operations for a user to interactively change the s-cuboid specification. It thus enables a user to navigate from one s-cuboid to another to explore the big cube space with ease:

- APPEND / PREPEND (X) – add a symbol X to the end / front of the pattern template;
- DE-TAIL / DE-HEAD – remove the last / first symbol from the pattern template;
- P-ROLL-UP / P-DRILL-DOWN (X) – move the abstraction level of pattern dimension X one level up / down the concept hierarchy;

For example, after studying the comparison shopping distribution in Figure 3, one might observe that there is a particularly large amount of users browsing Adidas shoes, and then a Nike shoes, and then browse Adidas shoes again (the highlighted cell). He can further investigate whether those users would browse another product and the distribution of the browsing patterns. This distribution can

[1]A complete specification of an S-OLAP query is a lot more elaborate [1]. For example, one has to specify whether a pattern is a substring pattern or subsequence pattern. We simplify our description here so as to focus on the main issues.

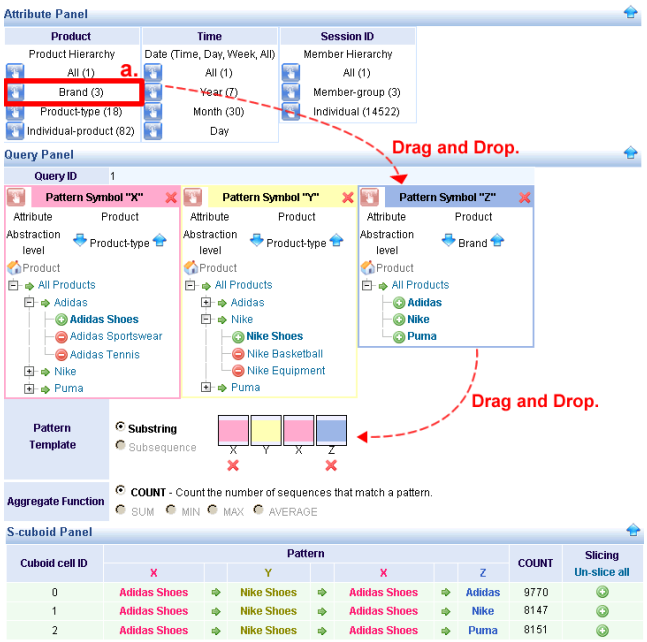

Figure 4: The interface after the slice, APPEND (Z) **and** P-ROLL-UP (Z) **operations**

be obtained by first performing a traditional *slice* OLAP operation on the cell (Adidas shoes, Nike shoes, Adidas shoes), followed by invoking the APPEND (Z) operation to modify the pattern template from (X, Y, X) to (X, Y, X, Z), where the newly appended symbol Z denotes the browsing from product X (Adidas shoes) to any product Z. Applying the above operations transforms the query specification to another and thus generates another s-cuboid. The S-OLAP GUI provides some drag-and-drop facilities such that the above operations can be done very easily. The results are updated in real-time so that the participants could experience the on-line analytical processing feature of the S-OLAP system.

Furthermore, after viewing the updated distributions, one might observe that there are too many cells, which makes the distribution reported by the s-cuboid too fragmented. In this case, a higher level summary of the browsing patterns can be obtained by invoking the P-ROLL-UP (Z) operation, which changes the abstraction level of the pattern symbol Z from the Product-type level to the Brand level. Again, the on-screen summary will be updated according to the changes of the query specifications. Figure 4 shows the updated query and s-cuboid.

From the above application examples, we can demonstrate that the biggest distinction of an S-OLAP system from a traditional OLAP system is that a sequence can be characterized not only by the attributes' values of its constituting events, but also by the subsequence/substring patterns it possesses. In other words, an S-OLAP system can support "pattern-based" grouping and aggregation – a very powerful concept and capability that is not supported by traditional OLAP systems.

2.3 S-OLAP for real business queries

S-OLAP systems have many more applications. Another application is the analysis of RFID logs. Today, many cities have implemented electronic transportation payment systems using RFID technology. Examples include Hong Kong's Octopus system and Washington DC's SmarTrip system. In those cities, every passen-

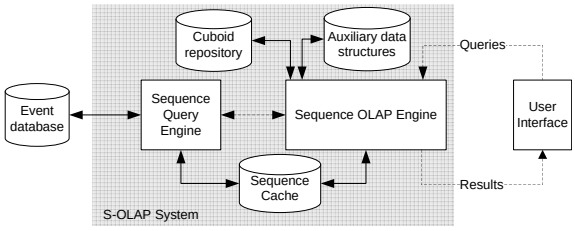

Figure 5: Architecture of S-OLAP System

ger carries a personal smart card, which can be used as a form of electronic money to pay for various kinds of transportation. The transactions performed by each card user can form a logical travel sequence according to the time attribute.

In the demonstration, we will show a number of the real business queries that we gathered from interviews with the managers of a metro company in Hong Kong. For instance, a marketing manager was once considering to organize a promotion campaign to offer round-trip discounts to passengers and she wanted to know "the number of passengers that performed *round-trip travel patterns* and their *distributions* over all origin-destination station pairs". In the demo, participants will see how the S-OLAP system is applied in those applications in supporting business decisions.

3. SYSTEM OVERVIEW

Unlike traditional OLAP systems, data in an S-OLAP is *non-summarizable*. That is, an s-cuboid at a higher level of abstraction (i.e., coarser aggregates) cannot be computed solely from a set of s-cuboids that are at a lower level of abstraction (i.e., finer aggregates) without accessing the base data. According to [1], summarizability only holds when the data is disjoint and complete during data partitioning. However for sequence data, since a sequence may contain multiple patterns, it may contribute values to multiple cells, which violates the disjointness requirement.

The non-summarizability of s-cuboids makes the implementation of an S-OLAP system very challenging. The main reason is that many existing OLAP optimization techniques are no longer applicable nor useful in implementing an S-OLAP system. In particular, it invalidates the power of *partial materialization* because an s-cuboid cannot be computed from other s-cuboids via simple aggregations. As a result, instead of precomputating s-cuboids, the system precomputes some other auxiliary data structures so that queries can be computed online using the pre-built data structures.

3.1 System Architecture

Figure 5 shows the architecture of the S-OLAP system. The current **S-OLAP Engine** $v2.0$ is implemented using C++, and the **User Interface** is developed with PHP and AJAX technology. Both the engine and the interface server run on Linux OS (kernel 2.6.24). All the data, definitions and indices are stored in MySQL databases.

The raw data of the system is a set of events that are deposited in an **Event Database**. Each event is modeled as an individual record/tuple in a way similar to those stored in a fact table of a traditional OLAP system. Each event record consists of a number of "dimension attributes" (such as *time*, *product*) as well as a few "measure attributes" (such as *amount*) and each dimension may be associated with a **concept hierarchy**. For instance, the *product* dimension is associated with a concept hierarchy of three abstraction levels individual-product→product-type→brand. Given a user requirement, the job of the **Sequence Query Engine** is to compose sets of event sequences out of the event database. The constructed sequence groups are cached in a **Sequence Cache** for efficiency.

Given an s-cuboid query, the S-OLAP engine consults a **Cuboid repository** to see if such an s-cuboid has been previously computed and stored. If not, the engine computes the s-cuboid with the help of certain **Auxiliary structures** for computational efficiency. S-OLAP Engine $v2.0$ has implemented the inverted indices approach (II) proposed in [1], which follows a semi-online computation strategy. The basic idea is similar to the idea of shell fragment cubes in [2], in which we partition the pattern dimensions into a set of low dimensional pattern fragments, and each fragment is represented by an inverted index. Using the precomputed inverted indices, we can dynamically assemble and compute the cells of the required s-cuboid online. The by-products of answering a query is the creation of new inverted indices, which are stored on disk. An in-memory caching system is also implemented in S-OLAP Engine $v2.0$ to reduce the I/O cost of fetching the disk-resident inverted indices. It has been shown that the performance advantage of II is particularly significant in answering iterative queries.

4. DEMONSTRATION DESCRIPTION

The objective of the S-OLAP system demonstration is to let the participants experience the process of interactive and exploratory analysis of sequence data, as well as to highlight the unique features and the usability of the S-OLAP system. The servers will be installed and hosted in our own labtop running Linux OS, and we will use Mozilla Firefox (version 5.0 or above, with cookies and javascripts enabled) as a browser of the web-based User Interface.

In the demonstration, we will first show that an S-OLAP query can be constructed in a user-friendly and efficient way through the GUI. We start with a general query Q_a to look for information about any two-step products browsing at the Brand abstraction level. To specify Q_a, one can first create two distinct pattern symbols "X" and "Y" by dragging the button "a" (see Figure 4) from the attribute panel and dropping it on the query panel area twice. Then one can invoke APPEND (X), followed by APPEND (Y) by dropping the corresponding pattern symbols to the pattern template area. The query construction is then finished and the resulting s-cuboid will be quickly generated and displayed on the s-cuboid panel. Since the shop is selling sportswear products of three brands only (i.e. Adidas, Nike and Puma), the s-cuboid consists of $3^3 = 9$ cells and captures a relatively high level summary statistics. With the first s-cuboid constructed, we are ready to pose some more follow-up queries to search for the statistics of our interest.

The comparison shopping patterns, as captured by pattern template (X, Y, X), can be obtained by simply dropping the pattern symbol X on the pattern template area (which invokes an APPEND (X) operation). Since this new query Q_b is a follow-up query, the interface system will request the query engine to execute Q_b immediately and refresh the result when the computation is done. With the help of the inverted indices that were built during the processing of the previous query Q_a, one will see that the computation of Q_b is reasonably fast. This contributes to an experience of on-line interactive and exploratory analysis of the sequence data.

Besides click-stream data, we will also show how S-OLAP is used in a number of application scenarios in answering real business queries. Furthermore, we will discuss and justify in more detail about the concepts behind the design of the S-OLAP system.

5. REFERENCES

[1] E. Lo, B. Kao, W.-S. Ho, S. D. Lee, C. K. Chui, and D. W. Cheung, "OLAP on sequence data," in *SIGMOD Conference*, 2008, pp. 649–660.

[2] X. Li, J. Han, and H. Gonzalez, "High-Dimensional OLAP: A Minimal Cubing Approach," in *VLDB*, 2004, pp. 528–539.