The LitOLAP Project: Data Warehousing with Literature

Owen Kaser, Steven Keith*
UNB Saint John
owen@unbsj.ca, n7sh4@unb.ca

Daniel Lemire
UQAM
lemire@acm.org

September 11, 2006

Keywords: on-line analytical processing, data warehouse, stylometrics, analogies

Abstract

The litOLAP project seeks to apply the Business Intelligence techniques of Data Warehousing and OLAP to the domain of text processing (specifically, computer-aided literary studies). A literary data warehouse is similar to a conventional corpus, but its data is stored and organized in multidimensional cubes, in order to promote efficient end-user queries. An initial implementation exists for litOLAP, and emphasis has been placed on cube-storage methods and caching intermediate results for reuse. Work continues on improving the query engine, the ETL process, and the user interfaces.

1 Introduction

Computer-aided tools can enhance literary studies [11, 12], and the litOLAP system aims to support such studies by adapting the well-developed techniques of data warehousing (DW) and on-line analytical processing (OLAP) [2]. These techniques are widely used to support business decision making. DW involves assembling data from various sources, putting it into a common format with consistent semantics, and storing it as multidimensional *cubes* to enable OLAP or data mining. OLAP techniques enable user-guided browsing of multidimensional and aggregate data. A business DW might integrate a merged company's inventory and sales transactional data over several decades, and its typical OLAP operations could include "rolling up" (i.e., "zooming out") from dates-by-week to dates-by-year or imposing range conditions (e.g., "omit 1983–1988 Manitoba data"). OLAP is for domain specialists, not IT professionals: the motto is "disks are cheap, analysts are expensive."

Building a DW over literary data (see Figure 1) differs from creating a corpus or digital library. The Extract-Transform-Load (ETL) stage retains only that information required to build the ordained DW cubes. Thus the DW schema constrains the possible analyses, but anticipated classes of queries can be fast. Dimensional hierarchies control allowable roll-ups, hence they are crucial. Our *Word* dimension allows roll-up by stem, suffix, and (several layers of) WordNet [9]

*current affiliation: Innovatia Inc., Saint John, New Brunswick

In CaSTA06, Fredericton, Canada, October 2006.

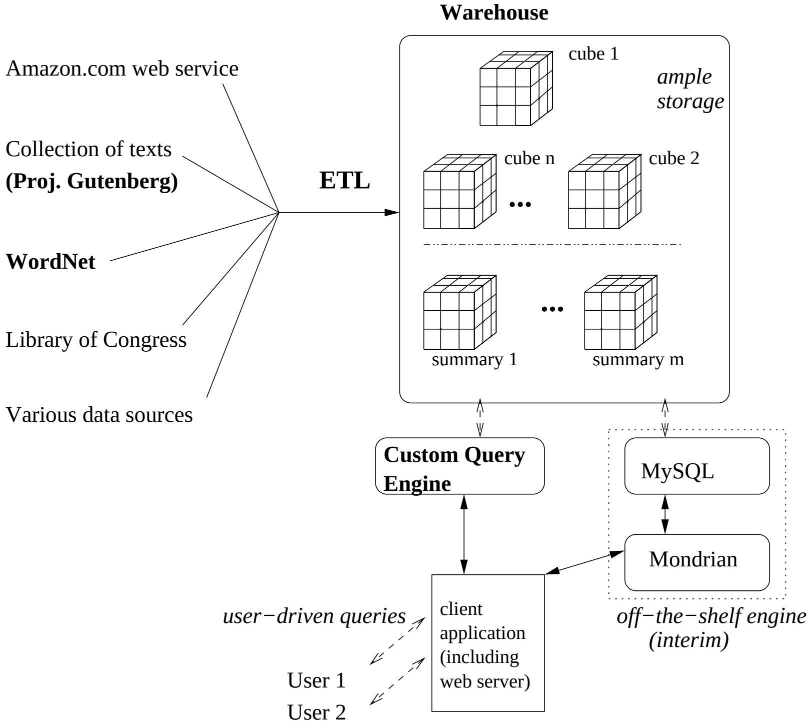

Figure 1: Literary DW, with existing portions in boldface.

hypernyms, for instance. (For instance, WordNet hypernyms for “whale” include the category “vertebrate, craniate”.)

As an example, our *AnalogyCube* was inspired by Turney and Littman [14]. It records how many times, when a short sliding window passes over a text, the subsequence $word_1$, joiningWord, $word_2$ is seen. For instance, the count for (whale, beneath, waves) is presumably higher in *Moby Dick* than in *Hamlet.* Turney and Littman have mined corpora to discover word-based analogies; via WordNet's hypernyms, litOLAP supports exploration of category-based analogies. A fuller motivation for litOLAP, our DW's schema and some example queries are given elsewhere [5, 6]; this poster primarily reports progress in implementing litOLAP.

OLAP using our DW can be contrasted with published text analyses that either required substantial custom programming or had to fit within the narrower range of queries supported by existing programs [13, 4] We do not believe that the software used for business OLAP can support several of the literary analogy queries planned. Therefore, litOLAP includes a custom engine (see Figure 2) that processes our kinds of queries. End users do not interact directly with the engine, so the custom query language (which supports the usual OLAP operations: slice, dice, roll-up as well as some specialized operations) need not be easy to use — it must be fast.

2 Technical Aspects

Figure 2 shows the engine's major components and technologies. After requesting that the engine open some “base cubes”, the application formulates queries (XML in Java strings), passes them

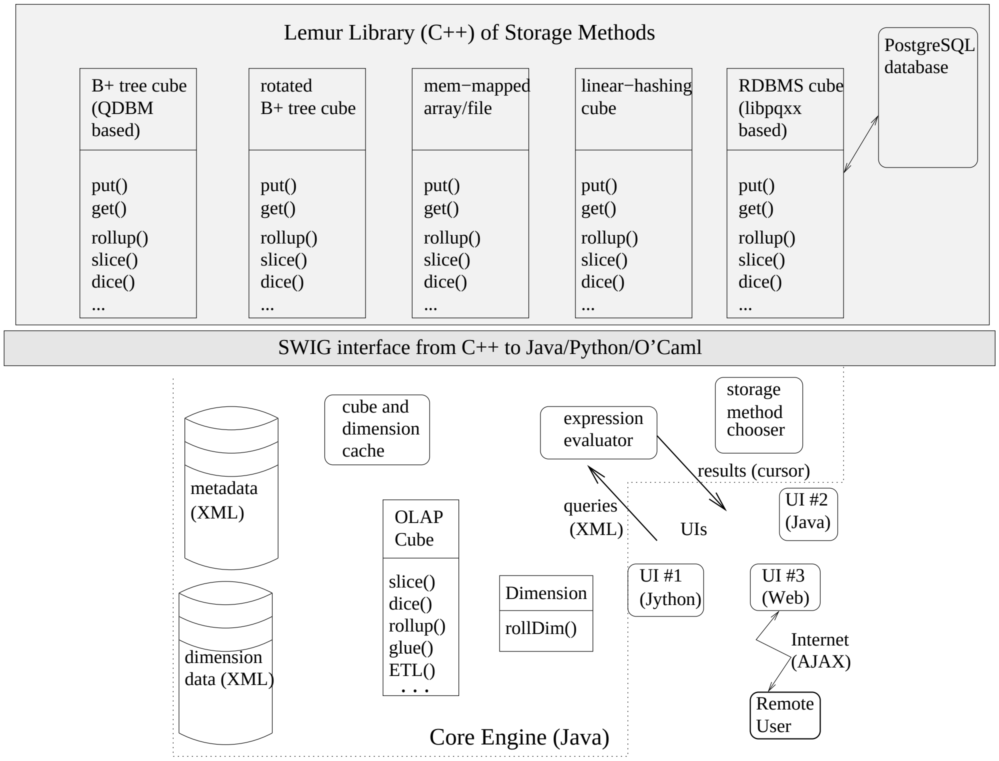

Figure 2: Custom OLAP engine.

to the engine, and eventually receives the results.

Development thus far has emphasized appropriate cube storage plus the pre-computation and caching of intermediate values. Keith's thesis [7] describes experiments with the storage methods shown in Figure 2. There was no overall "best storage method"; rather, many factors affected whether a particular operation was fast for a particular cube. For example, the B+ Tree sliced efficiently only on the first dimension, while the linear-hashing cube was able to slice equally well on all dimensions. The storage-method chooser evaluates the properties of a cube (size, density, number of dimensions, etc.) and will eventually use predictions of the expected operations' frequencies. (AnalogyCube is regularly sliced, for instance.) Its goal is selecting the most appropriate storage method when a cube is created.

Intermediate cubes and dimensions are calculated during query processing. Earlier, a particular subexpression may have been computed and the resultant intermediate cube may still be cached, speeding up the query. Deciding which intermediate values to save (and also which possibly useful intermediate values to precompute speculatively) is the well-studied *materialized-view problem* for databases [3, 1]. Automatic *rewriting* of queries to use cached intermediates is hard and remains to be added to the expression evaluator.

3 Status

Currently, the custom engine can answer a variety of "canned" test queries involving AnalogyCube or a *PhraseCube* (similar to word n-grams), and ETL processes the plain-text files on the Project Gutenberg [10] CD. Near-term enhancements include improving ETL quality, improving the custom engine's speed, and constructing a useful UI so that we can get end-user feedback. Longer term, the system needs to recognize when switching to specialized data structures(such as suffix arrays [8]) is advantageous. We believe that the overall idea of applying OLAP to literary data is promising. The initial custom engine is too slow for production use, but until more optimization is attempted, its promise is unclear.

References

[1] Rada Chirkova, Alon Y. Halevy, and Dan Suciu. A formal perspective on the view selection problem. *The VLDB Journal*, 11(3):216–237, 2002.

[2] E.F. Codd. Providing OLAP (on-line analytical processing) to user-analysis: an IT mandate. Technical report, E.F. Codd and Associates, 1993.

[3] Alon Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, 2001.

[4] Patrick Joula, John Sofko, and Patrick Brennan. A prototype for authorship attribution studies. *Literary and Linguistic Computing*, 21(2):169–178, 2006.

[5] Steven Keith, Owen Kaser, and Daniel Lemire. Analyzing large collections of electronic text using OLAP. In *APICS 2005*, October 2005.

[6] Steven Keith, Owen Kaser, and Daniel Lemire. Analyzing large collections of electronic text using OLAP. Technical Report TR-05-001, UNBSJ CSAS, June 2005.

[7] Steven W. Keith. Efficient storage methods for a literary data warehouse. Master's thesis, UNB, 2006.

[8] Udi Manber and Gene Myers. Suffix arrays: a new method for on-line string searches. *SIAM Journal on Computing*, 22(5):935–948, 1993.

[9] George A. Miller, Christiane Fellbaum, Randee Tengi, Susanne Wolff, Pamela Wakefield, Helen Langone, and Benjamin Haskell. Wordnet – a lexical database for the English language. `http://wordnet.princeton.edu/`.

[10] Project Gutenberg Literary Archive Foundation. Project Gutenberg. `http://www.gutenberg.org/`, 2006.

[11] Stephen Ramsay. Mining Shakespeare. In *ACH/ALLC 2005*. University of Victoria, June 2005.

[12] Michael Stubbs. Conrad in the computer: examples of quantitative stylistic methods. *Language and Literature*, 2005.

[13] TAPoR Project. TAPoR prototype of text analysis tools. online: `http://taporware.mcmaster.ca/~taporware/betaTools/index.shtml`, 2005. last accessed 27 June 2006.

[14] P. D. Turney and M. L. Littman. Corpus-based learning of analogies and semantic relations. *Machine Learning*, 60(1–3):251–278, 2005.