Efficacious Data Cube Exploration by Semantic Summarization and Compression

Laks V.S. Lakshmanan[†] Jian Pei[‡] Yan Zhao[†]

[†] University of British Columbia, Canada. {laks, yzhao}@cs.ubc.ca

[‡] State University of New York at Buffalo, USA. jianpei@cse.buffalo.edu

Abstract

Data cube is the core operator in data warehousing and OLAP. Its efficient computation, maintenance, and utilization for query answering and advanced analysis have been the subjects of numerous studies. However, for many applications, the huge size of the data cube limits its applicability as a means for semantic exploration by the user.

Recently, we have developed a systematic approach to achieve efficacious data cube construction and exploration by semantic summarization and compression. Our approach is pivoted on a notion of *quotient cube* that groups together structurally related data cube cells with common (aggregate) measure values into equivalence classes. The equivalence relation used to partition the cube lattice preserves the roll-up/drill-down semantics of the data cube, in that the same kind of explorations can be conducted in the quotient cube as in the original cube, between classes instead of between cells. We have also developed compact data structures for representing a quotient cube and efficient algorithms for answering queries using a quotient cube for its incremental maintenance against updates.

We have implemented SOCQET, a prototype data warehousing system making use of our results on quotient cube. In this demo, we will demonstrate (1) the critical techniques of building a quotient cube; (2) use of a quotient cube to answer various queries and to support advanced OLAP; (3) an empirical study on the effectiveness and efficiency of quotient cube-based data warehouses and OLAP; (4) a user interface for visual and interactive OLAP; and (5) SOCQET, a research prototype data warehousing system integrating all the techniques. The demo reflects our latest research results and may stimulate some interesting future studies.

Proceedings of the 29th VLDB Conference, Berlin, Germany, 2003

1 Introduction

Data warehouses form the essential infrastructure for many data analysis tasks. A *data warehouse* is a *subject-oriented*, *integrated*, *time-variant*, and *non-volatile* collection of data in support of managemental decision making processes [1]. A core operation in a data warehouse is the construction of a *data cube*, which can be viewed as a multi-level, multi-dimensional database with aggregate data at multiple granularities.

Let us consider an example. In a marketing management data warehouse, data are collected under the schema sales(Store, Product, Season, Sale). A *base table*, which holds the sales records, is shown in Figure 1. Attributes Store, Product and Season are called *dimensions*, while attribute Sale is called a *measure*.

Store	Product	Season	Sales
S_1	P_1	Spring	6
S_1	P_2	Spring	12
S_2	P_1	Fall	9

Figure 1: Base table sales for a data warehouse.

A *data cube* grouped by Store, Product, Season using an *aggregate function* (AVG(Sale) in this example) is the set of results returned from the 8 group-by queries with each subset of {Store, Product, Season}

forming the group-by. Each group-by corresponds to a set of *cells*, described as tuples over the group-by dimensions, identifying those tuples in the base table sales that agree with the cell on the group-by dimensions. The cells in the data cube $Cube_{Sales}$ are shown in Figure 2(a). Here, symbol "$*$" in a dimension means that the dimension is generalized such that it matches any value in the domain of this dimension.

In a data cube, two basic semantic relations among cells are *roll-up* and *drill-down*. A cell c_1 can be *rolled up* from cell c_2, and c_2 can be *drilled down* from cell c_1, if c_1 generalizes c_2 in some dimensions, that is, in all dimensions where c_1 and c_2 have different values, c_1 has values "$*$".

For example, in the data cube in Figure 2(a), cell $(S_1, P_1, Spring)$ can be rolled up to cell $(S_1, *, Spring)$, and the latter cell can be drilled down to the former one. Cell $(S_1, *, Spring)$ represents a higher level aggregate (i.e., the sales of *ALL products* in store S_1 and in the spring) than cell $(S_1, P_1, Spring)$ does (i.e, the sales of product P_1 in store S_1 and in the spring). Cells in a data cube form a lattice according to the roll-up/drill-down relation. Figure 2(b) shows the lattice for the data cube cells in Figure 2(a), while the top element, *false*, is not shown.

How does a data cube facilitate the online analytical processing (OLAP)? First, a data cube materializes (i.e., pre-computes) aggregates over various dimension combinations. With proper indexes, OLAP queries about aggregates can be answered promptly. Second, management is often concerned about relations and changes among various aggregates. A data cube materializing various aggregates facilitates answering such queries efficiently. Third, the user may navigate the data cube by exploring neighborhoods of cells via roll-up and drill-down operations, allowing them to detect interesting trends.

As data cubes are very useful for data warehouses and OLAP, are the current data cube techniques good enough? Unfortunately, there are some inherent problems that the current techniques cannot handle well.

- *Problem 1: Weak semantic relations among aggregate cells in data cubes.* While the essential roll-up/drill-down semantics are usually kept in a data cube, many kinds of critical semantics are not denoted. For example, as shown in Figure 2, tuple $(S_2, P_1, Fall)$ is the only contributor to the aggregates in cells $(S_2, *, Fall)$, $(S_2, P_1, *)$, $(*, P_1, Fall)$, $(*, *, Fall)$, and $(S_2, *, *)$. A one-step roll-up from cell $(S_2, P_1, Fall)$ along any dimension will not give the user any fruitful aggregate information. This kind of semantic relations among aggregate cells is critical for providing effective OLAP services.

- *Problem 2: No support of semantic navigation of data in data cubes.* In practice, a data cube lattice could be huge. For example, even without any hierarchy in any dimension, a 10-dimension data cube with a cardinality of 100 in each dimension leads to a lattice with $101^{10} \approx 1.1 \times 10^{20}$ cells. Assuming a high sparsity of one in a million cells being non-empty, we still have a huge lattice with 1.1×10^{14} non-empty cells! Hierarchies and denser cubes can make matters much worse.

  Exploring in a huge lattice is far from trivial. Suppose that a manager wants to identify exceptions by browsing the data. Without a proper navigation, the manager has no idea on which dimensions should be used to roll up or drill down. Many steps in her exploration may be just fruitless. Again, to provide an effective navigation service, we need to figure out semantics of the data in a data cube more than just roll-up and drill-down.

- *Problem 3: No semantic compression of data cubes.* It is well recognized that data cubes in many applications tend to be huge. Thus, many studies have focused on compressing data cubes. However, almost all approaches proposed previously are *syntactic*. That is, they do not consider the semantic relation among cells in a cube. This raises two major concerns. On the one hand, many syntactic compression methods make use of approximation, thus losing some information. On the other hand, most of them do not support direct (exact) query answering or browsing without uncompressing the compressed cube.

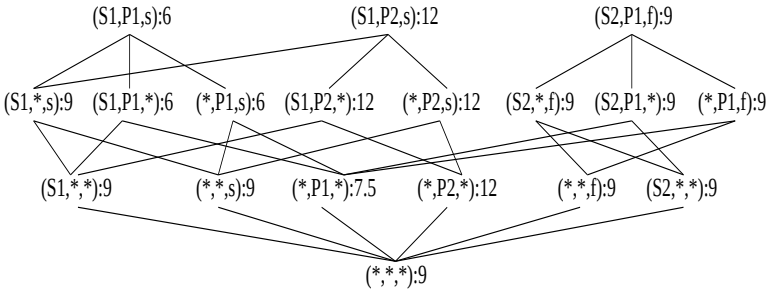
In summary, for the user to understand and effectively use the information in a data cube, we need compressed representations that do not lose information and preserves the cube semantics.

How do we construct semantic summaries of a data cube? For many applications, the semantics of an aggregate cell c in a data cube can be defined as the set of tuples in the base table that can be rolled-up to c. Let us take the data cube in Figure 2(a) as an example. The six cells, namely $(S_2, P_1, Fall)$, $(S_2, *, Fall)$, $(S_2, P_1, *)$, $(*, P_1, Fall)$, $(*, *, Fall)$, and $(S_2, *, *)$, have the same semantics, since they cover the identical set of tuples in the base table, i.e., $\{(S_2, P_1, Fall)\}$. In other words, $(S_2, P_1, Fall)$ is the only tuple in the base table contributing to the aggregates in each of the six cells.

Intuitively, we can "*summarize*" the six cells above into a "*class*", since they carry identical measures. With semantic summarization, we can derive classes from the data cube cell lattice in Figure 2(b)

Store	Product	Season	AVG(Sales)
S_1	P_1	Spring	6
S_1	P_2	Spring	12
S_2	P_1	Fall	9
S_1	$*$	Spring	9
S_1	P_1	$*$	6
$*$	P_1	Spring	6
...	...	...	...
$*$	$*$	Fall	9
S_2	$*$	$*$	9
$*$	$*$	$*$	9

(a) Cells in data cube $Cube_{Sales}$.



(b) The lattice of cells.

Figure 2: Data cube $Cube_{Sales}$

and get a *quotient lattice* in Figure 3. In the quotient lattice, a class is a set of structurally related cells carrying the same semantics. A quotient lattice is also called a *quotient cube*.[1] Clearly, the lattice in Figure 3 is substantially smaller than the one in Figure 2(b).

Now, let us examine how such a semantic summarization approach can solve the problems identified above.

- *Semantic relations in data cubes.* In a quotient cube, we store not only the roll-up and drill-down semantics about cells, but also the summarization of semantics of cells and the relation among classes. A user can conduct OLAP operations on semantic classes, which is more effective than those on cells.
- *Semantic navigation.* Now, a user can navigate the data cube by classes. Moreover, she can drill down *into* classes and investigate the internal structure of a class. Figure 4 illustrates a drill-down into class C_3 in the quotient cube of Figure 3.

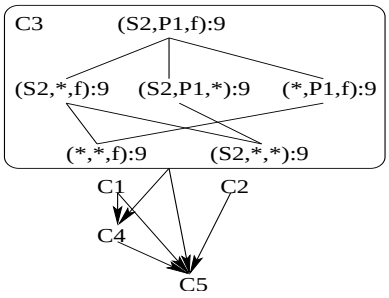


Figure 4: Drill-down to the internal structure of a class.

- *Semantic compression.* Clearly, since all cells in a class carry the same semantics, a semantic compression can be achieved. For each class, we only need to record the borders of the class. Furthermore, we can answer various queries and conduct various browsing and exploration operations using the semantic compression directly, without ever having to uncompress it.

[1] More precisely, a quotient cube is a quotient lattice preserving the roll-up and drill-down semantics on classes.

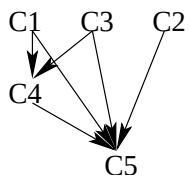
As shown above, a semantic approach brings significant improvements to the effectiveness and efficiency to data warehousing and OLAP. We are developing SOCQET, a systematic approach for effective and efficient semantic summarization for data warehousing and OLAP. We have made good progress in the following aspects.

First, different applications may require different semantic summarizations. In general, the semantics of cells in a data cube can be defined by an aggregate function $aggr()$. All cells having the same values for $aggr()$ can be regarded as having the same semantics. An arbitrary partition of the cells into classes merely based on their aggregate values may not be preferable, since such a partition may destroy the rolling-up/drilling-down semantics.

We have worked out methods to partition cells in a data cube into classes such that the resulting quotient cube lattice retains the roll-up/drill-down semantics [2]. Furthermore, we can answer various questions about quotient cubes, such as "can we make the quotient cube lattices as small as possible w.r.t. a given aggregate function?", "Can we construct a quotient cube useful for general-purpose OLAP?", and "What are the effects on quotient cubes if hierarchies appear in some dimensions?"

Second, we have developed a systematic method for effective semantic summarization in data cubes for various applications. Moreover, we have developed a comprehensive methodology for building a quotient cube-based data warehouse supporting both conventional and semantic OLAP operations [3].

Third, we have also developed efficient algorithms to construct quotient cubes for data warehousing and OLAP [3]. In particular, we devise efficient



Class	Cells
C_1	$(S_1, P_1, Spring), (S_1, P_1, *), (*, P_1, Spring)$
C_2	$(S_1, P_2, s), (S_1, P_2, *), (*, P_2, Spring), (*, P_2, *)$
C_3	$(S_2, P_1, Fall), (S_2, *, Fall), (S_2, P_1, *), (*, P_1, Fall), (*, *, Fall), (S_2, *, *)$
C_4	$(*, P_1, *)$
C_5	$(S_1, *, Spring), (S_1, *, *), (*, *, Spring), (*, *, *)$

Figure 3: A quotient cube.

data structures and algorithms to store, index and incrementally maintain quotient cubes, and answer various queries and conduct advanced analysis over quotient cubes.

Fourth, we have construct an effective user interface to enable the users to conduct semantic navigation and exploration over the quotient cube-based data warehouse. The users can investigate both the relations among the classes and the internal structures of the selected classes.

In this demo, we integrate the algorithms and implement SOCQET, a research prototype data warehousing and OLAP system. The system will fully support the conventional data warehousing and OLAP operations. The expected performance of the prototype system will be substantially higher than a system constructed using previously proposed techniques.

Why would the demo be interesting?

The proposed demo will expand and deepen our understanding on effective data warehousing and OLAP. The research will also bring benefits to many other related research works on advanced data analysis, such as data mining, data visualization, and interactive data exploration. For example, we can apply semantic summarization to data mining results, making the results easier to comprehend. Furthermore, before mining, we can first apply semantic summarization. Thus, the mining is conducted on a much smaller and more meaningful summarized data. That may make the mining more efficacious. As another example, data can be visualized based on semantic summarization. Moreover, a semantics-based interactive data exploration can be achieved. That is valuable in many applications.

Our prototype system will demonstrate the value of semantics-based summarization in conducting advanced analysis and visualization in several such settings and for very large data sets, including real-life ones. We expect the prototype system and the new techniques would be of interest to audience from both industry and academia.

2 About the Demo

Our demo consists of four major parts.

First, we will present the techniques to materialize quotient cubes using examples. We will analyze why such a materialization method is effective and efficient. We will also illustrate the storage efficiency of the storage techniques using real data sets.

Second, we will demonstrate how various queries can be answered using a materialized quotient cube. Examples and experiments will be used to illustrate the costs of query-answering.

Third, we will present a set of extensive performance studies on the proposed techniques and related methods proposed previously. We will examine the differences between our method and the previously proposed approaches. The experimental results on both real and synthetic data sets will indicate the benefits of the new techniques.

Last, we will showcase a prototype quotient cube based data warehousing and OLAP system, including a quotient cube engine and an interactive user interface. The audience will be encouraged to play with the demo and experience the exciting tour using semantic navigation services.

The major algorithms and experiments of the proposed demo have been implemented and conducted. We are now integrating the components into a prototype system.

References

[1] W. H. Inmon. *Building the Data Warehouse*. John Wiley & Sons, 1996.

[2] L. Lakshmanan, J. Pei, and J. Han. Quotient cube: How to summarize the semantics of a data cube. In *Proc. 2002 Int. Conf. Very Large Data Bases (VLDB'02)*, Hong Kong, China, Aug. 2002.

[3] L.V.S. Lashmanan, J. Pei, and Y. Zhao. Qc-trees: An efficient summary structure for semantic OLAP. In *Proc. 2003 ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'03)*, June 2003.