A General Model for Online Analytical Processing of Complex Data

Jian Pei

Department of Computer Science and Engineering
State University of New York at Buffalo
Buffalo, NY 14260-2000, USA
jianpei@cse.buffalo.edu
http://www.cse.buffalo.edu/faculty/jianpei

Abstract. It has been well recognized that online analytical processing (OLAP) can provide important insights into huge archives of data. While the conventional OLAP model is capable of analyzing relational business data, it often cannot fit many kinds of complex data in emerging applications, such as bio-medical data, time series and semi-structured data.
In this paper, we propose *GOLAP*, a general OLAP model. We show that GOLAP is consistent with the conventional OLAP model on multi-dimensional databases. Moreover, we show that the model can be applied to complex data as well. As an example, we illustrate a research prototype system, GeneXplorer, which enables OLAP over gene expression data.

1 Introduction

It has been well recognized that *online analytical processing* (OLAP) is an essential data analysis service and can provide critical insights into huge archives of application data. In contrast to online transactional processing (OLTP), OLAP supports queries about multi-dimensional, multi-level aggregates and summarizations of the data. Moreover, the users are enabled to browse the summarizations of the data at various granularities to identify trends, exceptions and interesting regions.

While many previous studies on efficient and effective OLAP over relational business data (e.g., [8,18,27,26,19,2,4,7,12,9,28,25,23,20]), few of them systematically studies how to extend the OLAP model to handle complex data, such as bio-medical data, time series and semi-structured data. To motivate the study and appreciate the challenges, let us look at the problems in OLAP of gene expression data.

Recently, the DNA microarray technology enables simultaneously monitoring the expression levels of thousands of genes during important biological processes and cross collections of related samples. Figure 1 shows the typical structure of microarray gene expression data. Usually, a row in the matrices represents a gene and a column represents a sample or condition. The numeric value in

I.-Y. Song et al. (Eds.): ER 2003, LNCS 2813, pp. 321–334, 2003.

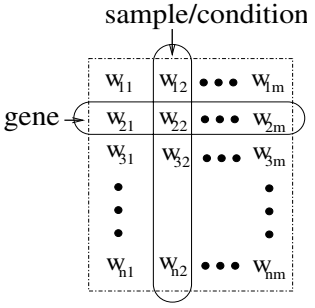


Fig. 1. A gene expression matrix.

each cell characterizes the expression level of a specific gene in a particular sample/condition.

Many previous studies have highlighted the importance of online analysis of gene expression data in bio-medical applications. As more and more gene expression data are accumulated, in addition to analyzing individual genes and samples/conditions, it is important to answer analytical queries about various "summarizations" over the gene expression data, e.g., the patterns and the trends. For example, in a gene expression database, with a new gene expression data sample, an analyst may want an online answer to the query "*with respect to the samples in the database whose gene expressions are similar to the new sample, what are the major patterns?*" This is a typical online analytical query.

Most of the previous studies on analysis of gene expression data focus on techniques of answering some specific queries, such as similarity search, alignments and clustering. However, there does not exist a general model in which the summarization-oriented OLAP queries can be specified and answered effectively.

Can we extend OLAP to handle complex data? While many algorithms for various analyses are available, the core problem is to *develop a general conceptual model* such that OLAP of complex data can be specified and evaluated effectively.

In this paper, we study the general model for OLAP of complex data. We make the following contributions.

- We illustrate that the conventional OLAP model does not work well for complex data. A typical example, the processing of gene expression data, is shown. It motivates our proposal of a new model.
- A general OLAP model, GOLAP, is proposed. We show that the model is a generalization of the conventional OLAP model.
- We elaborate how the GOLAP model can handle complex data.
- As an application, we demonstrate an OLAP system for gene expression data based on the GOLAP model. It shows that our model is effective.

The remainder of the paper is organized as follows. Section 2 introduces the preliminaries and motivates our study by showing the problems of the conventional OLAP model on complex data. Our GOLAP model is developed in

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

Fig. 2. Base table `sales` for a data warehouse.

Section 3. It is shown that GOLAP is compatible to the conventional OLAP model on multidimensional databases. In Section 4, we present how to apply GOLAP to complex data and use time series gene expression data as an example. GeneXplorer, a research prototype OLAP system for gene expression data based on the GOLAP model, is also demonstrated. Section 5 discusses related work. The paper is concluded in Section 6.

2 Preliminaries and Motivations

In this section, we first revisit the conventional OLAP model briefly. Then, using gene expression data as an example, we illustrate why the conventional model cannot support effective OLAP over complex data.

2.1 OLAP Operations in the Multidimensional Data Model

For the sake of simplicity, we illustrate the ideas of OLAP using the following example.

Suppose that, in a marketing management department, data are collected under the schema `sales(Store, Product, Season, Sale)`. The **base table**, which holds the sales records, is shown in Figure 2. Attributes `Store`, `Product` and `Season` are called **dimension attributes** (or **dimensions** in short), while attribute `Sale` is called a **measure attribute** (or a **measure** in short).

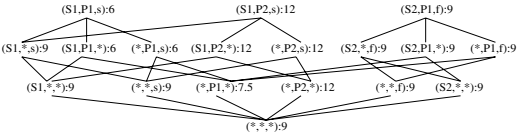
A **data cube** [10,11] grouped by dimensions `Store`, `Product` and `Season` using an **aggregate function** (`AVG(Sale)` in this example) is the set of results returned from the 8 group-by queries with each dimension subset of {`Store, Product, Season`} forming the group-by. Each group-by corresponds to a set of **cells**, described as tuples over the group-by dimensions, identifying those tuples in the base table `sales` that match the conditions. The tuples in the data cube $Cube_{Sales}$ is shown in Figure 3(a). Here, symbol "$*$" in a dimension means that the dimension is generalized such that it matches any value in the domain of this dimension.

As shown, a data cube is the n-dimensional generalization of the `group-by` operator. It computes group-bys corresponding to all possible combinations of a set of dimensions. A record in a data cube is also called an **aggregate cell**.

Two basic OLAP operations are *roll up* and its dual, *drill down*. A cell c_1 is **rolled up** from cell c_2, and c_2 is **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 "$*$". In other words, cell c_2 is a contributor to the aggregate of cell

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.

Fig. 3. Data cube $Cube_{Sales}$

c_1. For example, in the data cube in Figure 3(a), cell $(S_1, *, Spring)$ is a roll-up from cell $(S_1, P_1, Spring)$, and the latter cell is a drill-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).

All cells in a data cube form a lattice according to the roll-up/drill-down relation. Figure 3(b) shows the lattice for the data cube cells in Figure 3(a), while the top element, *false*, is not shown. Conceptually, OLAP provides a set of operations in the data cube space.

In addition to roll up and drill down, some other OLAP operations can be defined. For example, the **slice** operation performs a selection on one dimension of the data cube, and thus returns a sub-cube. The **dice** operation defines a sub-cube by selections on multiple dimensions, i.e., a composition of a set of slice operations. **Pivot** (also known as **rotate**) is a visualization operation that rotates the data axes in the presentation.

2.2 Challenges in OLAP of Complex Data

To examine the challenges in OLAP on complex data, let us consider an example: OLAP on gene expression data. As shown in Figure 1, gene expression data are in the form of matrices. *Can we treat a gene expression matrix with n genes and m conditions as a base table with n records and m dimensions or vice versa so that the conventional OLAP operations can be applied?* Unfortunately, such a naïve extension is unacceptable in both syntax and semantics.

In syntax, it is hard to define the measure and the aggregate function for gene expression matrices if the columns and the rows are treated as dimensions and tuples, respectively. Unlike the analysis of business data, where the common numeric aggregate functions (e.g., `SUM`, `MAX`, `MIN` and `AVG`) work well, the analysis of gene expression data often looks at the patterns, i.e., the common features approximately shared by similar genes or samples/conditions. There is no appro-

priate measure existing in the matrix. The numerical aggregate functions may not make sense in practice.

In semantics, the brute-force extension of roll up and drill down operation is meaningless in analyzing gene expression data. For example, suppose we take the samples/conditions as dimensions. Then, a roll up operation removes one or more samples/conditions from our analysis. The samples/conditions cannot be generalized. Similar problems exist if genes are treated as dimensions. That is, semantically, treating genes or samples/conditions as dimensions cannot achieve meaningful summarization of the data.

Similar problems exist in OLAP over other kinds of complex data, such as sequences, time series and semi-structured data. To make the situation even more challenging, in some kinds of data, such as semi-structured data, there can be even no explicitly existing dimension at all.

To conduct effective OLAP over complex data, we need to design a meaningful model. Based on the above analysis, we can obtain the following observations.

- There are two major issues in defining an OLAP model. On the one hand, it is essential to *define how to partition the data into summarization units at various levels*. On the other hand, it is critical to *define how to summarize the data*.
- *The summarization units for OLAP should yield to some nice hierarchical structure*, such as a lattice. That may facilitate the generation of concept hierarchies from OLAP analysis, and also support the users to browse the semantic summarizations of the data at various levels.

3 GOLAP: A General OLAP Model

In this section, we develop GOLAP, a general model for OLAP. We show that the model is a generalization of the conventional OLAP model.

3.1 The GOLAP Model

Let $\mathcal{D}$ be the space of data objects. A **base database** $B \subseteq \mathcal{D}$ is a set of data objects on which the OLAP is conducted. The concept of base database is the generalization of base table in the conventional model. In OLAP, we partition the objects in a base database into groups such that each group is a unit for summarization and all the groups form a hierarchy. The idea is formulated as follows.

Definition 1 (Grouping function). A **grouping function** is a function $g : 2^{\mathcal{D}} \times 2^{\mathcal{D}} \to 2^{\mathcal{D}}$. Given a base database B and a set of **query objects** Q (i.e., the set of objects need to be summarized) such that $Q \subseteq B$, $g(Q, B)$ is the subset of objects in B such that $g(Q, B)$ is the smallest summarization unit containing Q. $g(Q, B)$ is undefined for $Q \not\subseteq B$. A grouping function g should satisfy the following requirements.

1. **Containment**. For any sets of objects B and Q, $Q \subseteq g(B,Q) \subseteq B$. In words, the summarization unit should contain all query objects, and be in the base database B.
2. **Monotonicity**. For any sets of objects B, Q_1 and Q_2 such that $Q_1 \subseteq Q_2 \subseteq B$, $g(B,Q_1) \subseteq g(B,Q_2)$. In words, a larger query set needs a larger summarization unit.
3. **Closure**. For any sets of objects B and Q such that $Q \subseteq B$, $g(B,Q) = g(B,g(B,Q))$. In words, a summarization unit is self-closed. ∎

A grouping function defines how data objects should be partitioned. Based on a grouping function, we can partition a base database into classes.

Definition 2 (Class). Given a grouping function g and a base database B. A subset of objects $S \subseteq B$ is said a **class** if $g(B,S) = S$. ∎

In words, we use the closure of g (with respect to a given base database B) to define classes, the summarization units. We have the following result.

Theorem 1. *Given a base database B and a grouping function g. Let $\mathcal{C}$ be the set of classes with respect to B and g. Then, $(\mathcal{C}, \subseteq)$ is a lattice.*
Proof sketch. The theorem follows the facts (1) $g(B,B) = B$ and $g(B,\emptyset)$ are the top and the bottom elements of the lattice, respectively[1]; and (2) lattice $(\mathcal{C}, \subseteq)$ is a quotient lattice of $(2^B, \subseteq)$. ∎

Moreover, we define a **member function** $member : \mathcal{C} \to 2^{\mathcal{D}}$ with respect to a lattice of classes as follows. Given a class $c \in \mathcal{C}$, where $\mathcal{C}$ is the set of classes with respect to a base database B and a grouping function g, $member(c)$ returns the complete set of objects of class c in the base database.

So far, we define how to partition the data objects into summarization units (i.e., classes) and the summarization units yield to a nice hierarchical structure, a lattice. Now, we formalize the summarization of a set of data objects.

Definition 3 (Summarization function). Let M be the domain of summary over sets of objects. A **summarization function** is $f : \mathcal{D} \to M$. In words, a summarization function returns a summary for a set of data objects. ∎

Now, we are ready to define the OLAP operations in our model.

Definition 4 (OLAP operations). Given a base database B, a grouping function g and a summarization function f. Let $\mathcal{C}$ be the set of classes. There are three **basic OLAP operations** in the GOLAP model.

– **Summarize**. A `summarize` operation takes a set of objects Q as input and returns a duple

$$(g(B,Q), f(g(B,Q))).$$

That is, it returns the smallest class containing Q and the summary of the class.

[1] Please note that, in general, it is unnecessary $g(B,\emptyset) = \emptyset$.

– **Roll up**. A `roll up` operation takes a class c and a set of objects Q as parameters, and returns a duple

$$(g(B, member(c) \cup Q), f(g(B, member(c) \cup Q))).$$

That is, a roll up operation starts from a class c and returns the smallest class summarizing both the objects in class c and the objects in Q.

– **Drill down**. A `drill down` operation takes a class c and a set of objects Q as parameters, and returns

$$(g(B, member(c) - Q), f(g(B, member(c) - Q))).$$

That is, a drill down operation starts from a class c and returns the smallest class summarizing only the object in c but not in Q. ∎

Based on the above, we have the GOLAP model and the corresponding data warehouse.

Definition 5 (GOLAP and G-warehouse model). Given a data object space $\mathcal{D}$, (g, f) is a **GOLAP model** in $\mathcal{D}$ if g and f are a grouping function and a summarization function, respectively.

In a data object space $\mathcal{D}$, given a GOLAP model (g, f), $\mathcal{W} = \{(c, f(c))|c \in \mathcal{C}\}$ is said the **general data warehouse**, or **G-warehouse** in short, with respect to a base database B, where $\mathcal{C}$ is the complete set of classes with respect to g and B. ∎

As can be seen, the essentials for a GOLAP model is the grouping function and the summarization function. Moreover, the classes with respect to a grouping function and a base database form a lattice. These properties are intuitive and consistent to our observations in Section 2.2.

Moreover, the GOLAP model has the following nice property: the product of two GOLAP models is still a GOLAP model, as stated below.

Theorem 2 (Product of GOLAP models). *Let (g_1, f_1) and (g_2, f_2) be two GOLAP models on data space $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively. Then, $((g_1, g_2), (f_1, f_2))$ is a GOLAP model on data space $\mathcal{D}_1 \times \mathcal{D}_2$.*

Proof sketch. The proof follows the corresponding definitions. To illustrate the correctness, let us consider joining two base tables and the corresponding data cube lattices. Clearly, the product of the lattices is still a cube lattice on the joined table. ∎

Theorem 2 is essential in GOLAP model. It indicates that we can construct advanced OLAP model from simple ones. Moreover, it provides an approach to achieve integration of multiple OLAP models and data warehouses.

3.2 Applying the GOLAP Model to Multi-dimensional Databases

In this section, we apply the GOLAP model to multi-dimensional databases to check whether it is consistent with the conventional OLAP model.

Let $\mathcal{D} = (D_1, \ldots, D_n, M)$ be the schema of a base table as defined in Section 2.1. $\mathcal{D}$ is also the space of the tuples.

First, we define the summarization function f_r as follows. Given an aggregate function *aggr* over domain M. For any set of tuples T, let $V_T = \{t.M | t \in T\}$ be the multiple set (i.e., a bag) containing all the measure values in T. We define $f_r(T) = aggr(V_T)$.

Then, the grouping function can be defined as follows. Given any tuples $t_x = (x_1, \ldots, x_n, m_x)$ and $t_y = (y_1, \ldots, y_n, m_y)$. We define $t_x \wedge t_y = (z_1, \ldots, z_n, m_z)$ such that $m_z = aggr(m_x, m_y)$ and, for $(1 \leq i \leq n)$, $z_i = x_i$ if $x_i = y_i$, otherwise, $z_i = *$.

Given a base database (i.e., the base table) B and a set of query tuples Q. Let $t = (x_1, \ldots, x_n, m) = \bigwedge_{t_j \in Q} t_j$, and $x_{i_1}, \ldots, x_{i_k}$ are those dimension values not equaling to $*$. Then, $g_r(B, Q)$ is defined as the result of the following query in SQL.

```
SELECT D_1, ..., D_n, M
FROM   B
WHERE  D_{i_1} = x_{i_1} AND ... AND D_{i_k} = x_{i_k}
```

It is easy to show that g_r is a grouping function. That is, g_r satisfies the requirements of containment, monotonicity and closure in Definition 1.

Clearly, the GOLAP model (g_r, f_r) is consistent with the conventional OLAP model for multi-dimensional databases, as illustrated in Section 2.1. A data warehouse is the materialization of the classes descriptions and their summary (i.e., aggregate measures).

4 Applying GOLAP on Complex Data

In Section 3, we developed GOLAP, a general OLAP model. It is a generalization of the conventional OLAP model. Now, the problem becomes *how the GOLAP model can be applied to complex data.*

4.1 GOLAP Based on Hierarchical Clustering

The key of applying GOLAP to complex data is to find appropriate grouping functions and summarization functions. The general idea is that we can define such functions based on hierarchical clustering of the data objects. Here, the term **clustering** refers to the methods partitioning the data objects into clusters as well as to the set of all clusters, while the term **cluster** refers to a specific group of data objects.

Given $\mathcal{D}$, the space of the data objects. Let $\mathcal{CL}$ be a **hierarchical clustering** of data objects. For any base database B containing a set of objects, $\mathcal{CL}(B)$ is a hierarchy of clusters such that

1. each cluster is a subset of objects in B;
2. the hierarchy covers every object in B, i.e., each object appears in at least one cluster;
3. the base database B itself is a cluster in the hierarchy;
4. the ancestor/descendant relation in the hierarchy is based on the containment of the sets of objects in the clusters; and
5. for any two clusters c_1 and c_2, if $c_1 \neq c_2$ and $c_1 \cap c_2 \neq \emptyset$, then $c_1 \cap c_2$ is a cluster.

Then, a grouping function g_c can be defined as follows. Given a base database B and a set of query objects Q, $g_c(B, Q)$ returns the smallest cluster c in $\mathcal{CL}(B)$ such that $Q \in c$. It can be shown that g_c is a grouping function as defined in Definition 1.

Please note that not every clustering satisfies the above condition. For example, some clustering may allow overlaps among clusters. If the non-empty intersection of two clusters is not a cluster, then such a clustering fails the above requirements. However, given a clustering, it can always be fixed to meet the above requirements by inserting some "intermediate clusters" into the hierarchy. The general idea is that we can always make the non-empty intersections of clusters as "intermediate clusters". Limited by space, we omit the details here.

The definition of summarization function f_c is application-oriented. In general, given a class c of objects, $f_c(c)$ returns the summary of the class. The summary can be the pattern(s) in the class, the regression, etc.

According to the above analysis, (g_c, f_c) is a GOLAP model over data objects in $\mathcal{D}$. Moreover, based on Theorem 2, we can conduct OLAP on a database containing multi-dimensional data and multiple kinds of complex data.

4.2 An Example

While GOLAP is general for complex data looks elegant, *what is the effect of applying the GOLAP model to complex data?*

To examine the effect of the GOLAP model, we apply it to time-series gene expression data. In [15], we developed a hierarchical clustering method for time-series gene expression data. The clustering satisfies the requirements in Section 4.1. For example, Figure 4 shows the hierarchical clustering in the Iyer's data set [14].

We define the grouping function g_{gene} according to the hierarchical clustering. Given a set Q of objects, $g_{gene}(B, Q)$ returns the smallest cluster in the hierarchy that is a superset of Q.

To summarize the objects, we use the *coherent patterns* in the clusters. A **coherent pattern** characterizes the common trend of expression levels for a group of co-expressed genes in a cluster. In other words, a coherent pattern is a "template", while the genes in the cluster yield to the pattern with small divergence. Figure 5 shows some examples. We define the summarization function f_{gene} as mapping a cluster of objects into the coherent pattern in the cluster.

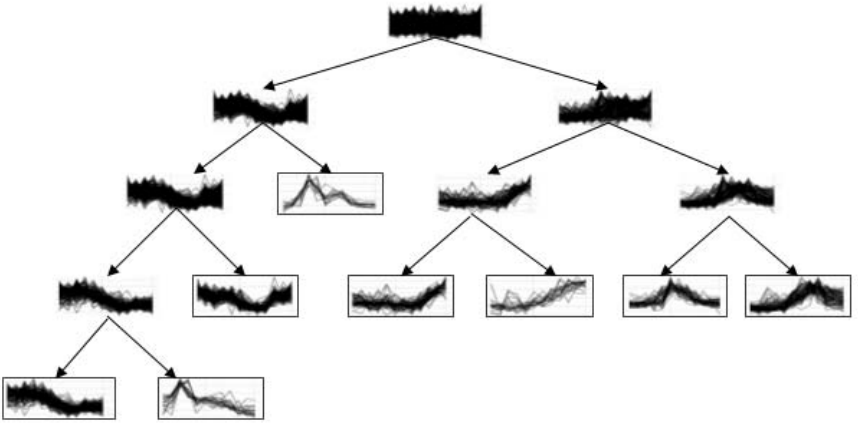


Fig. 4. The hierarchy of co-expressed gene groups in the Iyer's data set

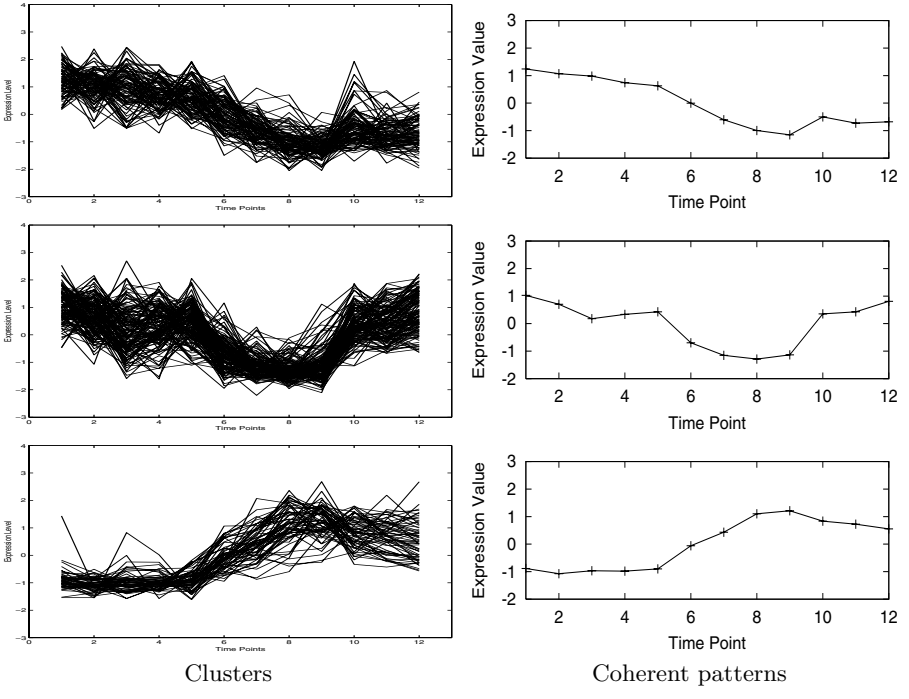


Fig. 5. Examples of clusters and corresponding coherent patterns

(g_{gene}, f_{gene}) forms a GOLAP model for time series gene expression data. Based on this model, we develop **GeneXplorer**, an interactive OLAP system

for time series gene expression data. The major OLAP operations in GeneXplorer are described as follows, where B is the base database (e.g., the Iyer's data set).

- **Summarization**. A user can submit a subset of genes Q. GeneXplorer returns $g_{gene}(B, Q)$ as well as $f_{gene}(g_{gene}(B, Q))$.
- **Roll up**. Starting from a current cluster c, a user can move to c's parent, and browse the genes and the coherent pattern in c's parent.
- **Drill down**. Starting from a current cluster c, a user can select one child of c, and browse the genes and the coherent pattern in that child.
- **Slice**. A user can compare the coherent patterns of the current cluster and its siblings.
- **Dice**. A user can select a subset of genes and apply the clustering. A hierarchy of clusters within the subset will be presented and the user can conduct the OLAP on the subset.
- **Pattern search**. A user can specify a coherent pattern, or some features of a pattern, the system returns the clusters and patterns that are similar to the specification within a user-specified similarity threshold.

As can be seen, the above OLAP operations are based on the GOLAP model (g_{gene}, f_{gene}). Some extensions are provided to facilitate the user's interactions.

To facilitate the above OLAP operations, GeneXplorer maintains a data structure (attraction tree) as the materialization of the clusters and the patterns. That can be regarded as a data warehouse.[2]

In summary, we illustrate how to use the GOLAP model to conduct OLAP on time series gene expression data. By carrying the similar idea, GOLAP can be extended to handle many other kinds of complex data, including sequences and semi-structured data (e.g., XML documents). Therefore, the GOLAP model based on hierarchical clustering as presented in Section 4.1 is general enough for those complex data.

Moreover, we also integrate the multi-dimensional data and the time series gene expression data in GeneXplorer. That is, each gene may have some attributes. As supported by Theorem 2, the attributes and the time series data are treated consistently in a uniform framework.

The idea of applying GOLAP model to complex data based on hierarchical clustering is general for many other applications. For example, hierarchical clustering is often available in document archives. Thus, with appropriate organization, GOLAP model can be applied. Moreover, when more than one hierarchical clustering theme presents, Theorem 2 indicates that we can construct a OLAP system containing more than one "*dimensions*". Here, one dimension corresponds to one hierarchical clustering theme.

[2] Precisely, an attraction tree is not exactly the materialization of the clusters and the patterns. Instead, it stores the objects and their relations (similarity) so that the clusters and patterns based on users' queries can be derived quickly on the fly. To this extent, an attraction tree is an index structure supporting the data warehouse.

5 Related Work

OLAP and data warehousing started from managerial practices. Some classical readings include [5,13,16,17,22]. It has been well recognized that OLAP is more efficient if a data warehouse is used. In [6], Colliat discusses how to support OLAP by relational and multidimensional databases. [4] is an excellent overview of the major technical progresses and research problems in 90's. In [29], Widom discusses some interesting research problems in data warehousing.

The data cube operator was firstly proposed by Gray et al. in [10,11]. Since it has been proposed, it became one of the most influential operator in OLAP. Several researches have been dedicated to the foundation and modelling of multidimensional databases and OLAP operations. [1,3,19,21] are some typical examples. However, most of them focus on multidimensional data. There is no systematic study on general OLAP model on complex data, like sequences, time series and semi-structured data.

Some recent studies aim at supporting advanced analysis in data warehouses and data cubes, such as discovery-driven exploration for online analysis of the exceptions in a data cube [25], online explanation of the differences in multidimensional aggregates [24], user-adaptive exploration in a data cube [26], finding the most general contexts under which the observed patterns occur [27], and answering hypothetical queries [2]. However, all these studies focus on specific analytical queries. There is no a general model. Again, they are based on multidimensional data.

6 Conclusions

In this paper, we study the problem of modelling for OLAP on complex data. We propose GOLAP, a general OLAP model. We show that GOLAP is consistent with the conventional OLAP model and, at the same time, general enough to support effective OLAP over complex data.

This study sheds light on the OLAP over complex data. Instead of conducting ad hoc, specific query oriented OLAP operations, we can have a uniform model for OLAP and develop the meaningful operations systematically.

Furthermore, the study opens the doors to some interesting future studies. For example, it is interesting to study how to model advanced OLAP operations specific to complex data. How to handle the changes in the base table, such as in the OLAP of XML document streams, is a challenging problem. Moreover, how to develop general techniques to implement a GOLAP model is important for warehousing complex data.

Acknowledgements. The author is grateful to Mr. Daxin Jiang and Ms. Chun Tang for their helps in the implementation of GeneXplorer. The author also thanks the anonymous reviewers for their invaluable comments.

References

1. R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional databases. In *Proc. 1997 Int. Conf. Data Engineering (ICDE'97)*, pages 232–243, Birmingham, England, April 1997.
2. Andrey Balmin, Thanos Papadimitriou, and Yannis Papakonstantinou. Hypothetical queries in an olap environment. In Amr El Abbadi, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal, Nabil Kamel, Gunter Schlageter, and Kyu-Young Whang, editors, *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10–14, 2000, Cairo, Egypt*, pages 220–231. Morgan Kaufmann, 2000.
3. Luca Cabibbo and Riccardo Torlone. A logical approach to multidimensional databases. In Hans-Jörg Schek, Fèlix Saltor, Isidro Ramos, and Gustavo Alonso, editors, *Advances in Database Technology – EDBT'98, 6th International Conference on Extending Database Technology, Valencia, Spain, March 23–27, 1998, Proceedings*, volume 1377 of *Lecture Notes in Computer Science*, pages 183–197. Springer, 1998.
4. S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. *SIGMOD Record*, 26:65–74, 1997.
5. E.F. Codd. Providing olap (on-line analytical processing) to user-analysis: An it mandate. In *Technical Report, E.F. Codd and Associates*, 1993.
6. George Colliat. Olap, relational, and multidimensional database systems. *SIGMOD Record*, 25(3):64–69, 1996.
7. P. Deshpande, J. Naughton, K. Ramasamy, A. Shukla, K. Tufte, and Y. Zhao. Cubing algorithms, storage estimation, and storage and processing alternatives for OLAP. *Data Engineering Bulletin*, 20:3–11, 1997.
8. S. Geffner, D. Agrawal, A. El Abbadi, and T. R. Smith. Relative prefix sums: An efficient approach for querying dynamic OLAP data cubes. In *Proc. 1999 Int. Conf. Data Engineering (ICDE'99)*, pages 328–335, Sydney, Australia, Mar. 1999.
9. F. Gingras and L.V.S. Lakshmanan. nD-SQL: A multi-dimensional language for interoperability and OLAP. In *Proc. 1998 Int. Conf. Very Large Data Bases (VLDB'98)*, pages 134–145, New York, NY, Aug. 1998.
10. J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data cube: A relational operator generalizing group-by, cross-tab and sub-totals. In *Proc. 1996 Int. Conf. Data Engineering (ICDE'96)*, pages 152–159, New Orleans, Louisiana, Feb. 1996.
11. J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals. *Data Mining and Knowledge Discovery*, 1:29–54, 1997.
12. H. Gupta, V. Harinarayan, A. Rajaraman, and J. D. Ullman. Index selection for OLAP. In *Technical Note 1996 available at http://db.stanford.edu/ ullman/ullman-papers.html#dc*, Stanford University, Computer Science, 1996.
13. W. H. Inmon. *Building the Data Warehouse.* John Wiley & Sons, 1996.
14. Iyer V.R., Eisen M.B., Ross D.T., Schuler G., Moore T., Lee J.C.F., Trent J.M., Staudt L.M., Hudson Jr. J., Boguski M.S., Lashkari D., Shalon D., Botstein D. and Brown P.O. The transcriptional program in the response of human fibroblasts to serum. *Science*, 283:83–87, 1999.
15. D. Jiang, J. Pei, and A. Zhang. Interactive exploration of coherent patterns in time-series gene expression data. In *Submitted to the Nineth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*, 2003.

16. Ralph Kimball. *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses*. John Wiley & Sons, 1996.
17. Ralph Kimball and Kevin Strehlo. Why decision support fails and how to fix it. *SIGMOD Record*, 24(3):92–97, 1995.
18. 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.
19. Wolfgang Lehner. Modelling large scale olap scenarios. In Hans-Jörg Schek, Fèlix Saltor, Isidro Ramos, and Gustavo Alonso, editors, *Advances in Database Technology – EDBT'98, 6th International Conference on Extending Database Technology, Valencia, Spain, March 23–27, 1998, Proceedings*, volume 1377 of *Lecture Notes in Computer Science*, pages 153–167. Springer, 1998.
20. Alberto O. Mendelzon and Alejandro A. Vaisman. Temporal queries in olap. In Amr El Abbadi, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal, Nabil Kamel, Gunter Schlageter, and Kyu-Young Whang, editors, *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10–14, 2000, Cairo, Egypt*, pages 242–253. Morgan Kaufmann, 2000.
21. Torben Bach Pedersen and Christian S. Jensen. Multidimensional data modeling for complex data. In *Proceedings of the 15th International Conference on Data Engineering, 23–26 March 1999, Sydney, Austrialia*, pages 336–345. IEEE Computer Society, 1999.
22. N. Pendse and R. Creeth. The olap report. In *Technical Report, Business Intelligence*, 1995.
23. S. Sarawagi. Indexing OLAP data. *Bulletin of the Technical Committee on Data Engineering*, 20:36–43, 1997.
24. S. Sarawagi. Explaining differences in multidimensional aggregates. In *Proc. 1999 Int. Conf. Very Large Data Bases (VLDB'99)*, pages 42–53, Edinburgh, UK, Sept. 1999.
25. S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of OLAP data cubes. In *Proc. Int. Conf. of Extending Database Technology (EDBT'98)*, pages 168–182, Valencia, Spain, Mar. 1998.
26. S. Sarawagi and G. Sathe. Intelligent, interactive investigaton of OLAP data cubes. In *Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'00)*, page 589, Dallas, TX, May 2000.
27. G. Sathe and S. Sarawagi. Intelligent rollups in multidimensional OLAP data. In *Proc. 2001 Int. Conf. on Very Large Data Bases (VLDB'01)*, pages 531–540, Rome, Italy, Sept. 2001.
28. Jayavel Shanmugasundaram, Usama Fayyad, and P. S. Bradley. Compressed data cubes for olap aggregate query approximation on continuous dimensions. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 223–232. ACM Press, 1999.
29. J. Widom. Research problems in data warehousing. In *Proc. 4th Int. Conf. Information and Knowledge Management*, pages 25–30, Baltimore, Maryland, Nov. 1995.