On-line Evaluation of a Data Cube over a Data Stream

Woo Sock Yang and Won Suk Lee
Department of Computer Science, Yonsei University
134 Shinchon-dong Seodaemun-gu
Seoul, 120-749, Korea
+82-2-2123-2716

Abstract: This paper proposes a dynamic data cube for applying a data cube to a data stream environment. The dynamic data cube specifies user-interesting areas with the support ratio of attribute value, and manages the attribute groups dynamically by grouping and dividing methods. With these methods, the memory usage and processing time are reduced. It also efficiently shows and emphasizes user-interesting areas by increasing the granularity for attributes that have higher support. We also propose an exception detecting method to quickly identify exception by using the reversed way of a multi-stage cluster sampling method. We perform experiments to verify how efficiently the dynamic data cube works in limited memory space.

Key-Words: Data stream, OLAP, Data cube, User-interesting area, Detecting exception, Cube tree

1 Introduction
The OLAP [1] has greatly evolved in the areas of database and data warehouse systems for multi-dimensional data analysis. The data cube, which has been a multi-dimensional data model of OLAP, shows many aspects of the data through two factors: dimension and measurement [2,3]. Now, OLAP is regarded as an essential tool for business decision makers and data analysts, and the data cube is successfully adopted on multi-dimensional data analyses [4]. In Currently, the amount of data and its generating speed are rapidly increased due to the fast growth of information technology and emerging ubiquitous eras. The data stream differs from other conventional data because it is generated continuously and enormously in a real-time manner, and its distribution characteristic is frequently changed. So, storing all data in a limited memory space is impossible. And it is hard to use a conventional data cube without modification for processing the data stream.

We propose the dynamic data cube, which uses a dynamic grouping mechanism for the dimension attributes. The dynamic data cube can analyze the entire data cube because it stores all cuboids placed in a single path. It also saves memory usage and reduces processing time by expanding and shrinking methods with coming tuples and managing them dynamically as groups. It somehow loses detailed information by grouping many attributes when they go out of interesting areas, but on the other hand it helps to give meaningful information to analysts because it divides the groups and gets detailed information when attributes come into interesting regions.

2 Related Work
The data cube models data in a multi-dimensional fashion and is defined by dimension and fact tables [5]. The dimension is an aspect which shows why analysts want to manage the data record, and the fact is a numerical value. The Stream Cube [6] is proposed to adopt a data cube over the data stream. It has the following three characteristics.

Firstly, it uses a tilted time frame to summarize time dimension. Analysts have shown interest mostly in recent data rather than old data, so the latest data should be stored in coarse granularity. Suppose that each tuple comes in every one minute. When the time reaches fifteen minutes, these data are summarized into a quarter span. This method saves more memory usage than storing every data into one-minute granularity. However, it becomes harder to discern detailed information from past data over time.

Secondly, the stream cube constructs the data cube only with cuboids inside of user-interesting areas [7, 8]. It defines minimal interesting layers and observation layers by applying the concept of iceberg query [9, 10], and saves memory usage by using cuboids that reside only between two layers. Therefore, once the stream cube is constructed, queries about the upper side of the observation layer or the lower side of the minimal interesting layer cannot be answered.

Thirdly, it uses the H-tree method [11] to store one popular path in which cuboids lie in between the minimal interesting layer and observation layer. The query about other paths not stored in the H-tree is answered by using given information. The popular path is fixed and is
decided by statistical results or the experience of analysts. As the fixed path cannot be changed during execution, the stream cube cannot show an accurate reaction to the radical changing of the data stream distribution.

3 Dynamic Data Cube

This paper proposes a dynamic data cube to make up for the weak points of the stream cube described above. The dynamic data cube maintains analysis granularity by grouping dimension attributes rather than storing the whole data cube. An important region, called the user-interesting area, lies between the minimum support ratio \( S_{min} \) and maximum support ratio \( S_{max} \) as shown in Figure 1. A support ratio of a group is defined as a ratio of the number of occurrence for the group to the number of occurrence for all groups.

![Fig.1 User-interesting areas](image)

In order to store analytic information, the dynamic cube uses statistical information as described in Definition 1.

**Definition 1. Statistical Information**
The dynamic data cube keeps the following data in its nodes to store information of the dimension attribute and statistical values of the measure attribute.
1. \( I \): \( I \) is a domain interval of the dimension attribute group.
2. \( C \): \( C \) is a summation of the occurrence of the dimension attribute group.
3. \( R \): \( R \) is a set of the values of the dimension attribute group.
4. \( M \): \( M \) is an average value of the measure attribute for the dimension attribute in a group.

**Definition 2. Sibling List**
1. Sibling list is a single-linked list that connects nodes containing statistical information as defined in Definition 1.
2. Sibling list maintains a pointer for connection between nodes.

**Definition 3. One-dimensional Tree**
The one-dimensional tree has values of dimension attribute and statistical information of the measure attribute related to one-dimensional cuboids.
1. It is composed of the sibling lists given in Definition 2.
2. Each tree level stores information of each dimension.
3. The upper- and lower-level dimensions are linked by a single-linked list.

**Definition 4. Cube Tree**
The cube tree stores dimension information and statistical values of the measure attribute in a single path from top cuboids to bottom cuboids.
1. The cube tree is composed of the sibling lists given in Definition 2.
2. Each level is composed of more than one sibling list, and all of the sibling lists in the same level are linked to each other by a double-linked list.
3. In order to connect sibling lists in the same level, it maintains the previous and next header pointers.
4. The upper and lower levels are connected by a single-linked list.
5. In order to connect different levels, it maintains the next level cuboids pointer.

3.1 The update of a dynamic data cube

When a new tuple \( t' \) is generated in time \( t \) of a data stream \( D_t \), the count value \( C_{t-1} \) of the dimension attribute and average value \( M_{t-1} \) of the measure attribute in time \( t-1 \) should also be updated. If the attribute value of a newly produced tuple lies in group range \( R \) and its measure attribute value is \( m_{t-1} \), Equation 1 is used to compute count value \( C \) and average measure value \( M \) in current time \( t \).

\[
C_t' = C_{t-1}' + 1, \quad M_t' = \frac{(M_{t-1}' \times C_{t-1}') + m_t}{C_t'} \tag{1}
\]

So that the dimension attribute group that lies outside of user-interesting areas is placed inside as best as possible, the dynamic data cube adjusts the range of the group using two phases: expanding phase and shrinking phase.

The expanding phase guides expanded groups to user-interesting areas by dividing the dimension attribute group whose support is larger than maximum support \( S_{max} \). This phase takes place as the following algorithm.
1. Divide a dimension attribute group in a one-dimensional tree by user-defined value \( \lambda \) in case its support is larger than user-defined maximum support \( S_{max} \).
2. Divide groups in the cube tree for the same level attribute group as in the one-dimensional tree.
3. If a dimension attribute group to be divided has a smaller interval than user-defined value \( \lambda \), then the group should not be divided.

If a dimension attribute group has larger support than user-defined support \( S_{max} \), the groups go out of user-interesting areas. Dividing a dimension attribute
group by $\lambda$ guides the group to user-interesting areas. Equation 2 is used to calculate the newly created group’s count and average measure value after the expanding phase.

$$C_{new} = C \times \frac{I_{new}}{I} = C \times \frac{1}{\lambda}, \quad M_{new} = M$$ (2)

In Equation 2, the new group’s interval $I_{new}$ is derived by dividing $I$ by $\lambda$. Its count $C_{new}$ is also derived by dividing $C$ by $\lambda$. The average measure value $M_{new}$ of the expanded dimension attribute group can be also used as an average measure value of the newly created group without further calculation because it is already an average value.

Figure 2 shows an example of the expanding phase as the groups with ranges of Red and Blue surrounded by the dotted line have surpassed support more than user-defined maximum support $S_{max} = 0.25$, divided each group into $\lambda = 2$. In this figure, the left side is a one-dimensional tree, and the right side is a cube tree given in Definitions 3 and 4, respectively.

The shrinking phase guides combined groups to user-interesting areas by shrinking the dimension attribute group whose support is larger than minimum support $S_{min}$. This phase takes place as the following algorithm.

1. Combine dimension attribute groups in a one-dimensional tree in case their supports are smaller than user-defined minimum support $S_{min}$.
2. Combine groups in the cube tree for the same level attribute group as in a one-dimensional tree.
3. If dimension attribute groups to be combined are not continuous in their attribute values, then the groups should not be combined.

![Fig.2 An example of expanding phase ($S_{min} = 0.1$, $S_{max} = 0.25$, $\lambda = 2$)](image-url)
The more continuous dimension attribute groups have smaller support than user-defined value $S_{min}$, the more the groups leave user-interesting areas. Combining dimension attribute groups guides the group placed in the user-interesting areas. Equation 3 is used to calculate the combined group’s count and average measure value.

$$C_{new} = \sum_{i=1}^{n} C_i, \quad M_{new} = \frac{\sum_{i=1}^{n} (M_i \times C_i)}{C_{new}} \tag{3}$$

The new count value of a group can be calculated by the summing up of those counts of $n$ groups. The average measure value $M_{new}$ of the combined dimension attribute group will be the average of the newly created group’s measure value.

4 Exception Detecting Method in the Dynamic Data Cube

Data is stored in a data cube in a summarized form and is explored by OLAP operations. OLAP operations don’t help analysts to reach a meaningful portion of the data cube, even though it makes it possible to explore the entire data cube. Analysts usually depend on hypothesis-driven exploration for the data cube through operations. Discovery-driven exploration was recently proposed to not go through a tedious exploring job [12]. Although the discovery-driven exploration supports detecting exception with pre-computed exceptions value at different levels, it is not applicable to the data stream as it must compute all coefficients between every dimension and hierarchy. Therefore, we propose an exception detecting method to quickly identify exceptions by using the reversed way of the multi-stage cluster sampling, a sampling method from statistics [13].

4.1 Exception detecting method with reversed way of multi-stage cluster sampling

The one-dimensional tree can be regarded as a population in statistics because it is made up of clusters of one-dimensional attribute and has stored average measure values since the beginning of the dynamic data cube. It always considers the cube tree as a set of samples. Thus, we can compare each sample with its own population, and if some of those samples which don’t follow the characteristics of the population are selected as exceptions. In order to tell a sample is an exception, there is a need for a precise way that could show the degree of exception.

The z-score makes it possible to compare two data no matter the difference in the amount of data, as its value lies between 0 and 1 [14]. Let $Z_T$ be the $z$-score of the population in the one-dimensional tree and $Z_P$ be the $z$-score of a multi-stage clustered sample in the cube tree respectively. Equation 4 is the formula to compute the exception $Z_{exp}$ in the dynamic data cube.

$$Z_{exp} = \max(|Z_T - Z_P| - \tau, 0) \tag{4}$$

If the distance between $Z_T$ and $Z_P$ is larger than user-defined value $\tau$, then the sample is detected as an exception.

4.2 Comparison with discovery-driven method

In order to find exceptions through discovery-driven exploration, it is necessary to compute all coefficients between every dimension and hierarchy [12]. For example, $SelfExp$ is computed as Equation 5.

$$SelfExp(y_{k_{d_{l}}-l_{a}}) = \max\left(\frac{|\hat{y}_{k_{d_{l}}-l_{a}} - \bar{y}_{k_{d_{l}}-l_{a}}|}{\sigma_{k_{d_{l}}-l_{a}}}, -\tau, 0\right) \tag{5}$$

where $y$ denotes a value of a cell in the data cube, and $\bar{y}$ is a value computed from the correlation between every dimension and hierarchy. In order to compute $\bar{y}$, all aggregated values through every dimension are needed.

The space complexity for computing those aggregated values is derived from Equation 6, assuming that $n$ is the number of dimensions and $m$ is the number of each dimension’s values.

$$S_{SE} = \sum_{r=1}^{n} nC_r \times m^r \times B = \sum_{r=1}^{n} \frac{n!}{r!(n-r)!} \times m^r \times B \tag{6}$$

In this equation, $B$ denotes the memory size for one cell. $\sum_{r=1}^{n} nC_r$ is the number of combinations for possible subsets of dimensions, and $m^r$ is the number of values in the $r^{th}$ dimension. Thus, the multiplication of these two values becomes the entire number of cells in the data cube. The time complexity $T_{SE}$ is for only updating without reading tables, so it is the same as time for sorting data in the order of specific attributes. Conventional database systems usually use the 2 Phase Multiway Sorting algorithm. Most time of this method is affected by time for disk read and write, and uses 64Kbytes for each block. Disk read and write time $T_{IO}$ is different among various systems, but is usually below milliseconds. Thus, the time complexity $T_{SE}$ depending on the number of cells is computed as Equation 7.

$$T_{SE} = \frac{S_{SE}}{64000} \times T_{IO} \tag{7}$$

On the other hand, exception detecting methods in the
dynamic data cube need to search the cube tree and compare values of samples in the cube tree with values of population in the one-dimensional tree. Thus, the space complexity $S_{DC}$ is obtained as follows.

$$S_{DC} = \sum_{r=1}^{m} \left( \frac{m}{k} \right)^{r} \times B$$

(8)

The space complexity of this exception detecting method is the same as the space complexity of the cube tree, because it uses the cube tree itself for exception detecting. Also, if the grouping phase is done in $1/k$-fold of its original memory size, then the number of attributes in the dimension becomes $m/k$, not $m$. Most of the time in the exception detecting method is spent getting samples for comparison; that is, searching time for the cube tree becomes time complexity, which can be estimated as Equation 9. Memory read and write time $T_{MIO}$ is also below milliseconds.

$$T_{DC} = \log_{\frac{m}{k}} S_{DC} \times T_{MIO}$$

(9)

As shown in the above equations, the space complexity and time complexity of discovery-driven exploration methods are far bigger than those of the dynamic data cube. In other words, the dynamic data cube is better than discovery-driven exploration in terms of space and time consumption.

5. Experiments

The data sets used in our experiments have distributed between 1.8 and 3.4 in zipfian distribution [15]. The number of dimensions varies from 5 to 8, and each data set consists of 100,000 tuples. The cardinality of the attribute value is 100. Each tuple is processed one by one to simulate the data stream environment.

The accuracy $A$ in our experiments is obtained as a relative error between the measure value $R.C$ of a conventional database and the measure value $G.C$ of the dynamic data cube as follows:

$$A = 1 - \frac{\sum_{i=1}^{\infty} \sum_{j=1}^{\infty} |R.C_{i}^{j} - G.C_{i}^{j}|}{R.C}$$

(10)

Figure 3 shows the memory usage of the data cube and dynamic data cube by varying the zipfian distribution. Figure 4 shows the memory usage along with the number of data tuples in zipfian distribution 1.8. The dynamic data cube uses less memory space than the data cube due to its expanding phase for the groups which have lower support than user-defined values. The memory usage of the data cube is continuously increased because it stores all newly produced attribute values.

Figure 5 shows the accuracy of the experiment in Figure 4. The dynamic data cube shows lower accuracy under the 200,000 tuples due to its expanding and shrinking phases as the dimension attribute support changed frequently. However once the distribution of the data stream becomes stable, the accuracy is increased since there are no more expanding and shrinking phases.

Figure 6 shows the memory usage for the different number of dimensions by varying the zipfian distribution. Figure 7 shows processing time taken for updating in the experiment of Figure 6. As the zipfian distribution has a higher value, most support values of the dimension attribute tend to be positioned in the same place.
Subsequently, the grouping phase is frequently happened so as the memory usage and processing time are reduced. As the number of dimensions is increased, memory usage and processing time are increased because the height of tree becomes higher.

Figure 8 shows the accuracy when zipfian distributions are varied. The difference in accuracy is not large among them. As the zipfian distribution has a higher value, the number of groups to be combined is enlarged. It makes the accuracy slightly higher by combining many groups which have unit ranges into larger single groups.

6. Conclusion
This paper proposed a dynamic data cube for efficiently adopting the data cube to a data stream environment. It is impossible to store all produced data in a limited memory space due to the characteristic of the data stream. Consequently, it is better to try to provide meaningful information by utilizing given memory space. The dynamic data cube reduces memory usage and processing time by employing user-interesting areas and by managing attribute values as groups and not as fine grain units. These methods can lose the accuracy for the outside of interesting areas, but it is possible to maintain precise accuracy for the interesting areas; that is, it tries to provide more memory space to interesting areas rather than to the other areas. At the same time, it provides recent, useful information by using its dynamical grouping mechanism.

We also proposed an exception detecting method to help analysts find exceptions faster and more easily. The proposed method employs the reversed way of multi-stage cluster sampling. The one-dimensional tree can be regarded as a population in statistics and the cube tree can be thought as a set of samples. Thus, we can compare each sample with its own population, and whether some of those samples which do not follow the characteristics of the population should be detected as an exception. We show that the proposed method needs less time and memory space than discovery-driven exploration methods.

ACKNOWLEDGEMENT
"This work was supported by the Korea Science and Engineering Foundation (KOSEF) NRL Program grant funded by the Korea government(MEST)"

References:
[6] Jiawei Han, Yixin Chen, Guozhu Dong, Jian Pei, Benjamin W. Wah, Jianyong Wang, Y. Dora Cai, Stream Cube: An Architecture for Multi-Dimensional Analysis of Data Streams, Distributed and Parallel Databases, Vol. 18, No. 2, 2005, pp.173-197.