Performance Optimization of Analysis Rules in Real-time Active Data Warehouses*

Ziyu Lin[1], Dongzhan Zhang[1,**], Chen Lin[1], Yongxuan Lai[2], and Quan Zou[1]

[1] School of Information Science and Technology, Xiamen University, Xiamen, China
{ziyulin,zdz,chenlin,zouquan}@xmu.edu.cn
[2] School of Software, Xiamen University, Xiamen, China
laiyx@xmu.edu.cn

Abstract. Analysis rule is an important component of a real-time active data warehouse. Performance optimization of analysis rules may greatly improve the system response time when a new event occurs. In this paper, we carry out the optimization work through the following three ways: (1)initiating non-real-time analysis rules as less as possible during rush hour of real-time analysis rules; (2) executing non-real-time analysis rules using the same cube at the same time interval; and (3) preparing frequent cubes for the use of real-time analysis rules ahead of time. We design the LADE system to get all the reference information required by optimization work. A new algorithm, called ARPO, is proposed to carry out the optimization work. Empirical studies show that our methods can effectively improve the performance of analysis rules.

Keywords: analysis rules, real-time active data warehouses.

1 Introduction

In the past decades, data warehouses have been going through five different stages, i.e. reporting, analyzing, predicting, operationalizing and active warehousing. Now real-time active data warehouses [1,2,3,4] are attracting more and more attention due to the great benefits they bring to the organization.

Analysis rule[1] is a very important part of a real-time active data warehouse. It detects the occurrence of events and initiates analysis process, during which multi-dimensional data will be used. If certain condition evaluates to be TRUE, the corresponding action will be triggered, such as sending alerts to analysis workers. Up to date, most of the research work on analysis rule is focused on its mechanism (e.g. [5,1]). In fact, performance optimization of analysis rules is also a critical aspect, though, to the best of our knowledge, there is still no published work on it. If more attention is paid to the optimization work, we can on one

* Supported by the Fundamental Research Funds for the Central Universities under Grant No. 2011121049, the Natural Science Foundation of Fujian Province of China under Grant No. 2011J05158 and 2011J05156, and the Natural Science Foundation of China under Grant No. 61001013 and 61102136.

** Corresponding author.

Q.Z. Sheng et al. (Eds.): APWeb 2012, LNCS 7235, pp. 669–676, 2012.

hand make full use of system resources, and on the other hand, achieve better performance for analysis rules.

In this paper, we propose the issue of performance optimization of analysis rules in real-time active data warehouses. Here analysis rules are divided into two types, namely, *real-time analysis rules* and *non-real-time analysis rules*. We define rush hour and frequent cubes for real-time analysis rules, and cube using pattern for non-real-time analysis rules. Our optimization work is focused on three aspects: (1)initiating non-real-time analysis rules as less as possible during rush hour of real-time analysis rules; (2) executing non-real-time analysis rules using the same cube at the same time interval; and (3) preparing frequent cubes for the use of real-time analysis rules ahead of time. The LADE(Log data mining-based Active Decision Engine) system is designed to help get all the reference information required by optimization work, such as rush hour, cube using pattern matrix and frequent cube matrix. Then we give a new algorithm, called ARPO (Analysis Rule Performance Optimization), to carry out the optimization work. We also conduct experiments in LADE system, and the results show that our method can effectively improve the performance of analysis rules in real-time active data ware houses.

The remainder of this paper is organized as follows: Sect. 2 gives problem statement. In Sect. 3, we introduce the LADE system first, followed by the detailed description of getting reference information for optimization work. In Sect. 4, we will show how to carry out the optimization work. The experiment results are reported in Sect. 5. Sect. 6 discusses the related work. Finally, we give the discussion and conclusion in Sect. 7.

2 Problem Statement

Compared to the traditional data warehouse, a real-time active data warehouse usually has an additional component called "real-time data cache", which stores all the real-time data. Through CDC(Change Data Capture), data change occurring in the OLTP system can be captured and propagated to the real-time data cache and active decision engine. The active engine is composed of event model, rule model, action model and meta-data model. Event model detects the occurrence of events, and rule model runs analysis rules after the occurrence of certain event. If a rule evaluates to be TRUE, the corresponding action will be triggered, such as notifying the analysis workers. Such process is called *active decision making*, during which the active decision engine may access the OLAP server for the required cubes. User may define the analysis rules for the active decision engine, and deal with the problems and conflicts that need to be treated manually.

Analysis rules can be classified into two types: real-time analysis rules and non-real-time analysis rules. The former, denoted $\mathcal{R}$, after being triggered, makes real-time decision, which will be fed back to operational system to satisfy real-time business requirements. The latter, denoted $\mathcal{N}$, after being initiated, makes decision usually not for the purpose of real-time applications.

Analysis rules use cubes to carry out multi-dimensional analysis. With the help of *cube using pattern* (see Definition 1), we can initiate, during the same period as much as possible, those non-real-time analysis rules that access the same cube. The detailed information about how a cube is defined can be found in [1].

Also, by generating frequent cubes those are most accessed in certain period ahead of time for a real-time analysis rule $\mathcal{R}$, we can greatly improve the performance of $\mathcal{R}$ in some cases. As far as non-real-time analysis rules are concerned, the concept of frequent cube is not so useful, since they do not make real-time decision.

Problem Statement. Given a set of real-time analysis rules $S_1 = \{\mathcal{R}_1, \mathcal{R}_2, ..., \mathcal{R}_m\}$ and a set of non-real-time analysis rules $S_2 = \{\mathcal{N}_1, \mathcal{N}_2, ..., \mathcal{N}_n\}$, performance optimization of analysis rules works as follows:

- find the rush hour $\bigcup[t_a, t_b)$, and during these time intervals, initiate the rules in S_2 as less as possible;
- find those rules in S_2 that use the same cubes in the multi-dimensional analysis process, and initiate them at the same period as much as possible;
- find the frequent cubes for certain period $[t_1, t_2)$, and prepare these cubes for the rules in S_1 before t_1.

3 Getting the Information for Performance Optimization

3.1 The LADE System

The LADE system designed by us is used to perform data mining based on the log of analysis rules. As is shown in Fig.1, in LADE, we extend the traditional architecture of active decision engine [5] by adding the logging component, called *action log*, to record all the necessary information about analysis rules, such as *ID*, *IsRealTime*, *RuleInfoID*, *CubeID* and *Time*.

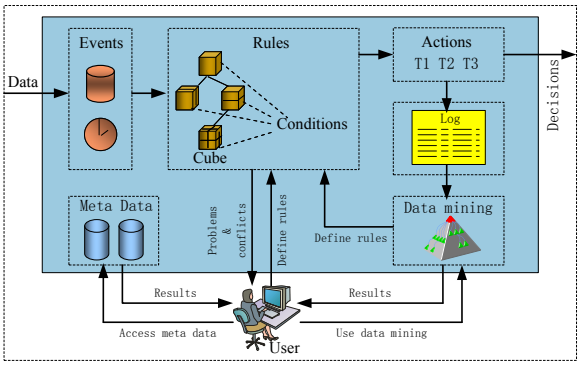

Fig. 1. The architecture of active decision engine in the LADE system

3.2 Cube Using Pattern

Definition 1. ***Cube Using Pattern Matrix:*** *Let* $C=\{c_0,c_1,...,c_{m-1}\}$ *and* $I = \bigcup_{j=0}^{n-1} [t_j,\ t_{j+1})$*, where* c_i *is a cube,* m *is the number of cubes used by all non-real-time analysis rules,* $[t_j,\ t_{j+1})$ *is a unit interval, and* n *is the number of unit intervals that a day is divided into. Frequent cube matrix is an* $m \times n$ *matrix* $U = (u_{ij})$ *such that* $u_{ij} = p$*, where* p *is null or a pointer pointing to a link list.*

In Definition 1, the link list, pointed by u_{ij}, is used to store the $RuleInfoID$ of all those non-real-time analysis rules that use cube c_i during time interval $[t_j,\ t_{j+1})$. Cube using pattern matrix, U, can be stored in memory for the use of performance optimization algorithm, and we can get it from the action log.

3.3 Frequent Cube

Definition 2. ***Frequent Cube Matrix:*** *Let* $C=\{c_0,c_1,...,c_{m-1}\}$ *and* $I = \bigcup_{j=0}^{n-1} [t_j,\ t_{j+1})$*, where* c_i *is a cube,* m *is the number of cubes used by real-time analysis rules,* $[t_j,\ t_{j+1})$ *is a unit interval, and* n *is the number of unit intervals that a day is divided into. Frequent cube matrix is an* $m \times n$ *matrix* $A = (a_{ij})$ *such that*

$$a_{ij} = \begin{cases} 1 & if\ c_i\ is\ a\ frequent\ cube\ for\ [t_j, t_{j+1}) \\ 0 & otherwise \end{cases}$$

Frequent cube matrix A can be easily maintained in memory to enhance the performance of optimization algorithm. Also it can be easily extended according to our requirements.

4 Performance Optimization with ARPO Algorithm

Performance optimization work is based on the necessary reference information such as rush hour, cube using pattern matrix and frequent cube matrix. System will do the optimization work whenever an analysis rule is initiated. Algorithm 1 shows the process of performance optimization, in which, q_i in the 15th line is used to store the $RuleInfoID$ of all those rules using c_i. When a rule is initiated, if it is a real-time analysis rule, system will generate the required cube for it. Also, if the cube is a frequent cube for the current time interval, it will be materialized for the use of other coming real-time analysis rules, which will be a great help for improving the performance of those rules. If a non-real-time analysis rule L is initiated, system will first judge if it is rush hour now, or if there are other rules using the same cube as L. If either one of the two conditions evaluates to be TRUE, L will be enqueued into a waiting queue q, which accommodates all those rules sharing the same cube. At an appropriate time, the rules in the waiting queue will be dequeued and continue their analysis process. In this way, system may just generate the required cube one time to satisfy all those rules from the same waiting queue, which also greatly improves the system performance.

Algorithm 1. ARPO(A, U, L, S)

Input : 1: frequent cube matrix A
2: cube using pattern matrix U
3: analysis rule L
4: rush hour set S
Output: 1: execution result

```
1  begin
2    get the time interval [t_j, t_{j+1}) to which the current time belongs;
3    i ← L. CubeID;
4    if L. IsRealTime=TRUE then
5      generate the cube c_i if not exist;
6      execute L;
7      if A[i][j] = 1 then
8        materialize the cube c_i if it has not been materialized;
9      else
10       delete cube c_i;
11     end
12     return execution result of L;
13   else
14     if ([t_j, t_{j+1}) ∈ S) or (U[i][j] ≠ NULL) then
15       initialize a waiting queue q_i if not exist;
16       put L. RuleInfoID into q_i;
17       return q_i;
18     end
19   end
20 end
```

5 Empirical Study

In this section, we report the performance evaluation of our method. The algorithms are implemented with C++. All the experiments were conducted on Intel i7-2600 3.40GHz CPU, 16.0GB memory DELL PC running Windows 7 and Oracle 11g.

In the LADE system, we use the TPC benchmark TPC-H to get the required datasets. We have been running the LADE system for several months. The action log contains three month of data, from which we can get the rush hour set S, cube using pattern matrix U and frequent cube matrix A.

Experiment 1. We design 50 real-time analysis rules which will use 30 cubes all together in the analysis process. We change the value of f, the percent of frequent cubes to the overall 30 cubes, from 10% to 50%. Fig.2 shows the change of t_1/t_2 during this process, where t_1 is the total time cost for the execution of all these 50 rules without optimization work, and t_2 is the time cost with optimization work. We can get from the experiment result that frequent cube plays an important role in decreasing the execution time of real-time analysis rules. ARPO algorithm can make full use of frequent cubes in the process of

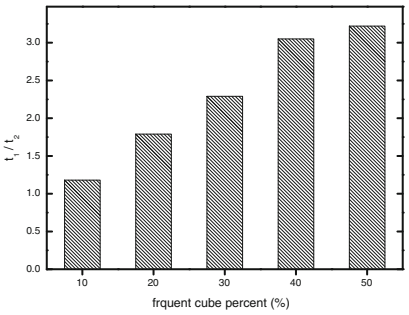

Fig. 2. Time cost ratio

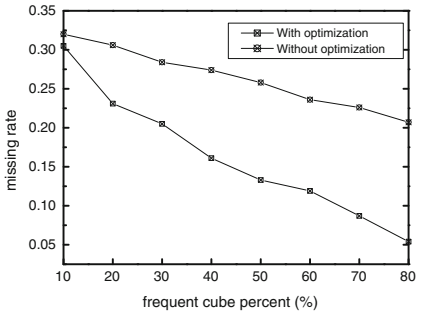

Fig. 3. Missing rate

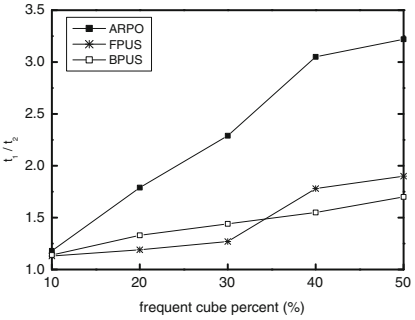

Fig. 4. Time cost ratio comparison

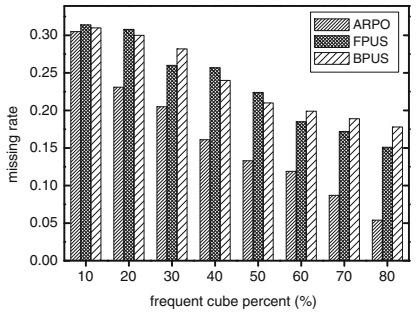

Fig. 5. Missing rate comparison

performance optimization. For example, when f=50%, time cost ratio t_1/t_2 can reach a high value of 3.22.

Experiment 2. If a cube required by a real-time analysis rule is not available right now, which means that it needs to be generated upon the time of requirement, we say that the real-time analysis rule misses this cube. Here we define a new variable $r = a_1/a_2$, where a_1 is the total times that real-time analysis rules miss cubes during a period of time T, and a_2 is the total times that real-time analysis rules require cubes during T. Just as Experiment 1, we change the percent of frequent cubes to the overall 30 cubes from 10% to 80%. From the experiment result in Fig.3, we can get that ARPO algorithm can reduce missing rate greatly. When $f = 10\%$, the values of r before and after optimization are 0.32 and 0.305 respectively. When $f = 80\%$, they are 0.207 and 0.054 respectively.

Experiment 3. Some desirable methods used to deal with the issue of materialized view selection, such as FPUS [6] and BPUS [7], may also help to improve the performance of real-time analysis rules, because they maintain those frequently-

used cubes in the system. Fig.4 shows the time cost ratio of ARPO compared with those of FPUS and BPUS, from which we can get that, ARPO may achieve much better performance than both FPUS and BPUS. As far as ARPO is concerned, the larger the value of f is, the greater the performance improvement is. However, for FPUS and BPUS, the value of t_1/t_2 only change a little when the value of f changes between 10% and 50%. For example, when f=50%, the value of t_1/t_2 for ARPO is 3.22, and for FPUS and BPUS, however, they are only 1.9 and 1.7 respectively. Fig.5 shows the missing rate of ARPO compared with those of FPUS and BPUS. We can observe that, the value of f has much more influence on ARPO than on FPUS and BPUS. In another word, ARPO may take better use of frequent cubes than both FPUS and BPUS.

6 Related Work

Real-time active data warehouse is attracting more and more attention due to the great benefits it may bring to organizations. It can support both strategic and tactic decisions. Analysis rule is a very important part of the real-time data warehouse, which is based on the knowledge developed in the field of active database systems. Paton *et al.* in [8] do a lot of survey work in active database system. This survey presents the fundamental characteristics of active database systems, describes a collection of representative systems within a common framework, considers the consequences for implementations of certain design decisions, and discusses tools for developing active applications.

In [5], Schrefl *et al.* introduce the conception of ECA rules from active database system into the field of data warehouses, and applies the idea of event-condition-action rules (ECA rules) to automate repetitive analysis and decision tasks in data warehouses. The work of an analyst is mimicked by analysis rules, which extend the capabilities of conventional ECA rules in order to support multidimensional analysis and decision making. In [5], the authors also present the architecture of active data warehouse, and describe in detail the knowledge of event model, action model, analysis graph and analysis rules.

In [9], Tho *et al.* combine (1) an existing solution for the continuous data integration and (2) the known approach of active data warehousing by introducing protocols that enable the correct collaboration between the two. Therefore, analysis rules can operate on real-time data.

Up to date, the research work in the field of real-time active data warehouses mostly focuses on the aspects such as the architecture of real-time active data warehouse [9], real-time data integration, real-time data modeling, time consistency [10], query optimization, scalability, real-time alerting, and so on.

7 Discussion and Conclusion

In this paper, we focus on the performance optimization of analysis rules in real-time active data warehouses. The LADE system is designed to get all the reference information required by optimization work, and a new algorithm, called

ARPO, is proposed to carry out the optimization work based on the reference information. Extensive experiments show that our method can effectively improve the system performance of analysis rules.

References

1. Thalhammer, T., Schrefl, M., Mohania, M.: Active Data Warehouses: complementing OLAP with Analysis Rules. Data and Knowledge Engineering 39, 241–269 (2001)
2. Chen, L., Rahayu, J.W., Taniar, D.: Towards Near Real-Time Data Warehousing. In: 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 1150–1157. IEEE press, New York (2010)
3. Polyzotis, N., Skiadopoulos, S., Vassiliadis, P., Simitsis, A., Frantzell, N.: Supporting Streaming Updates in an Active Data Warehouse. In: 23rd International Conference on Data Engineering, pp. 476–485. IEEE Press, New York (2007)
4. Lin, Z.Y., Lai, Y.X., Lin, C., Xie, Y., Zou, Q.: Maintaining Internal Consistency of Report for Real-Time OLAP with Layer-Based View. In: Du, X., Fan, W., Wang, J., Peng, Z., Sharaf, M.A. (eds.) APWeb 2011. LNCS, vol. 6612, pp. 143–154. Springer, Heidelberg (2011)
5. Schrefl, M., Thalhammer, T.: On Making Data Warehouses Active. In: Kambayashi, Y., Mohania, M., Tjoa, A.M. (eds.) DaWaK 2000. LNCS, vol. 1874, pp. 34–46. Springer, Heidelberg (2000)
6. Tan, H.X., Zhou, L.X.: Dynamic selection of materialized views of multi-dimensional data. Journal of Software 13(6), 1090–1096 (2002)
7. Harinarayan, V., Rajaraman, A., Ullman, J.D.: Implementing data cubes efficiently. In: ACM SIGMOD 1996 International Conference on Management of Data, pp. 205–216. ACM Press, New York (1996)
8. Paton, N.W., Diaz, O.: Active Database Systems. ACM Computing Surveys 31(1), 63–103 (1999)
9. Tho, M.N., Tjoa, A.M.: Zero-Latency Data Warehousing for Heterogeneous Data Sources and Continuous Data Streams. In: 5th International Conference on Information Integrationand Web-based Applications Services, pp. 55–64. Austrian Computer Society, Vienna (2003)
10. Bruckner, R.M., Tjoa, A.M.: Capturing Delays and Valid Times in Data Warehouses-Towards Timely Consistent Analyses. Journal of Intelligent Information Systems 19(2), 169–190 (2002)