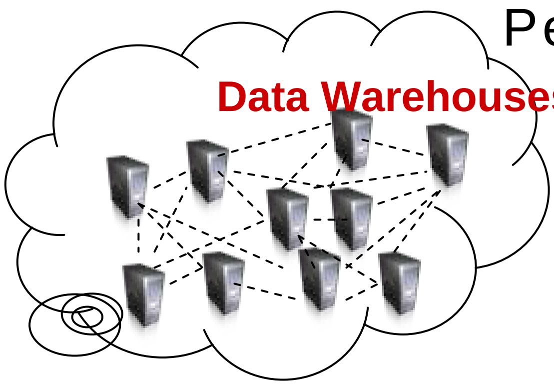
APWeb2012 Paper Presentation



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

Ziyu Lin(林子雨), *et al*

Department of Computer Science

Xiamen University

Homepage: http://www.cs.xmu.edu.cn/linziyu

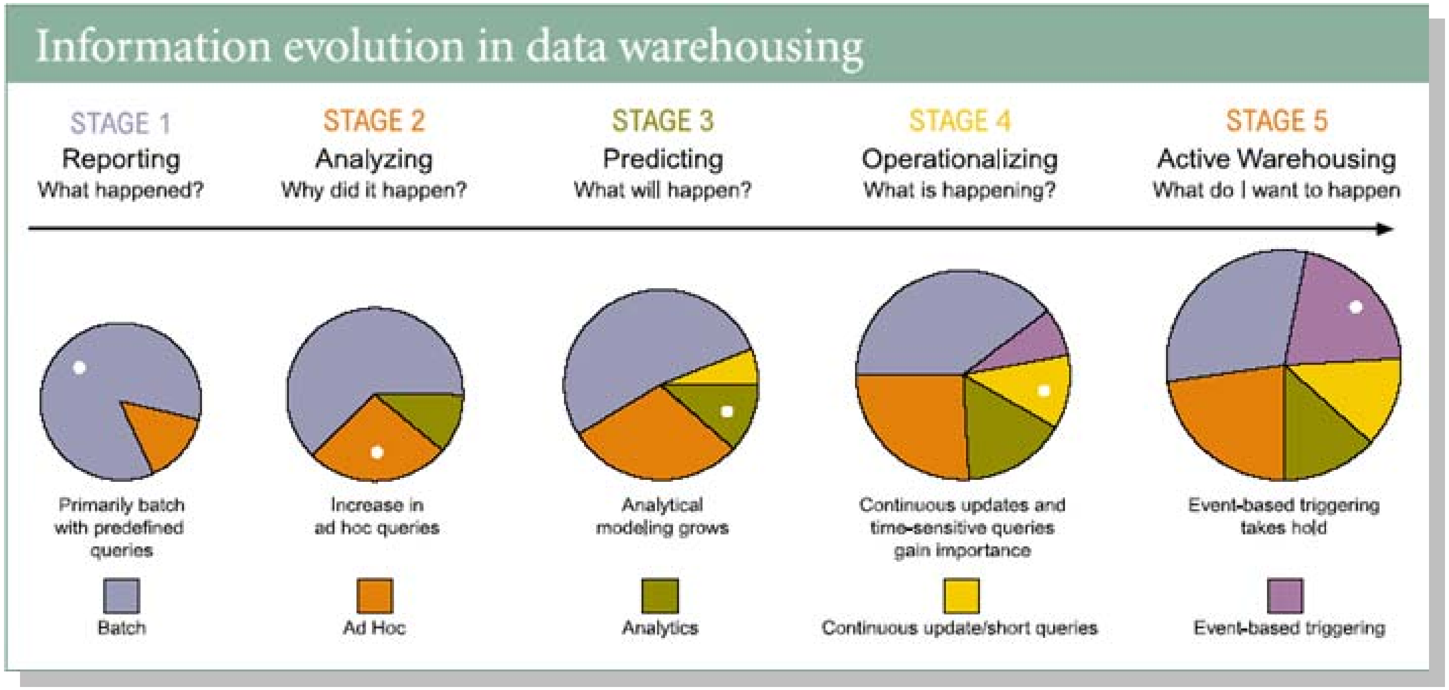
E-mail: ziyulin@xmu.edu.cn



Outline

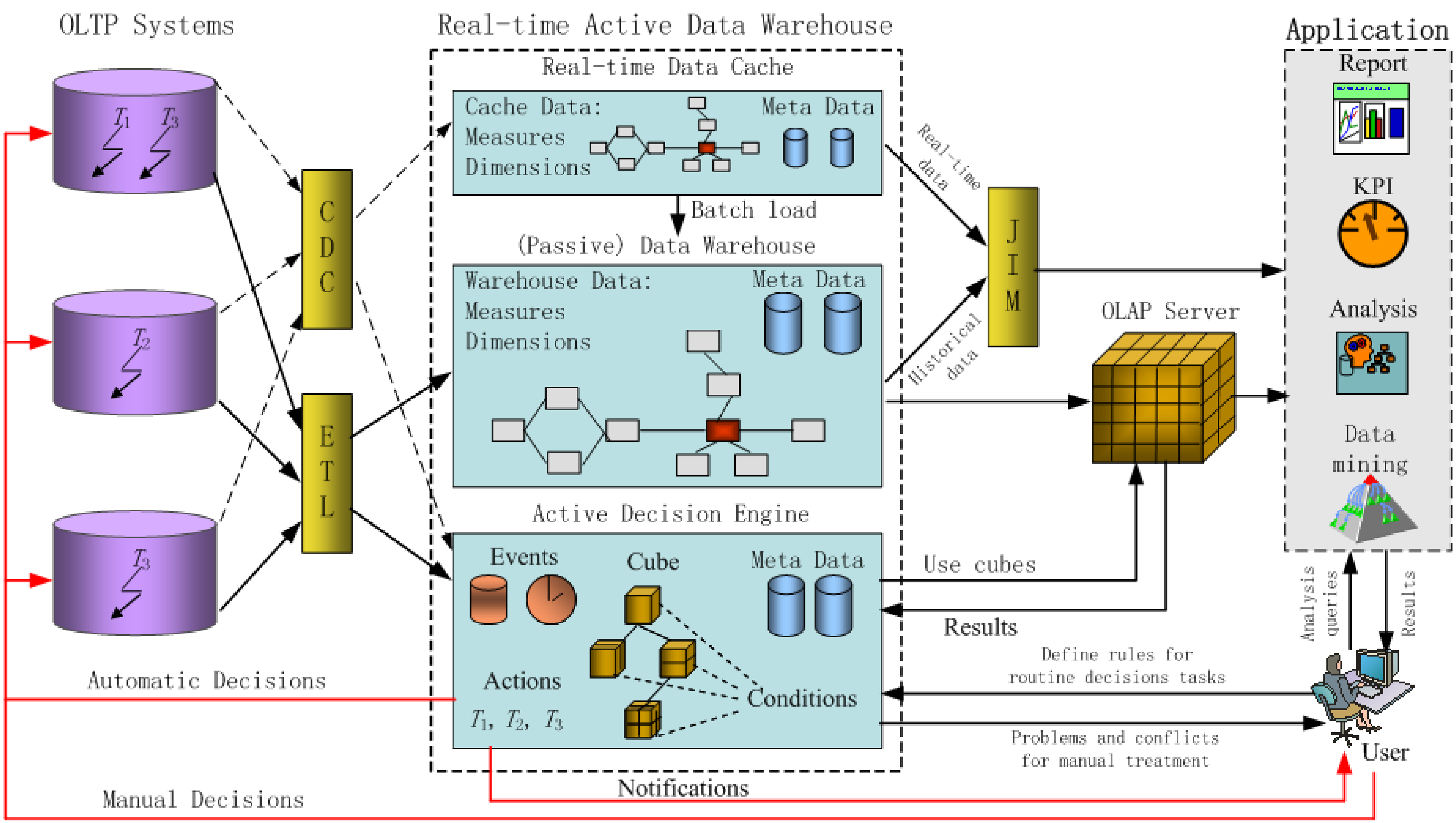
- Introduction
- Getting the Information for Performance Optimization
- Performance Optimization with ARPO Algorithm
- Empirical Study
- Conclusion
- References

Introduction



Data warehouses have been going through five different stages: reporting, analyzing, predicting, operationalizing and active warehousing.

Introduction



An example of real-time active data warehouse architecture

Introduction

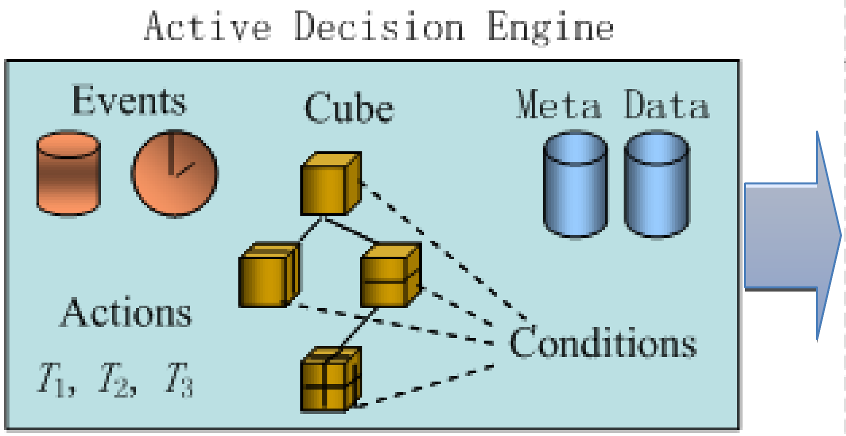
- Analysis rule[1] is a very important part of a real-time active data warehouse.
  - It detects the occurrence of events and initiates analysis process, during which multi-dimensional data will be used.
  - If certain condition evaluates to be TRUE, the corresponding action will be triggered, such as sending alerts to analysis workers.

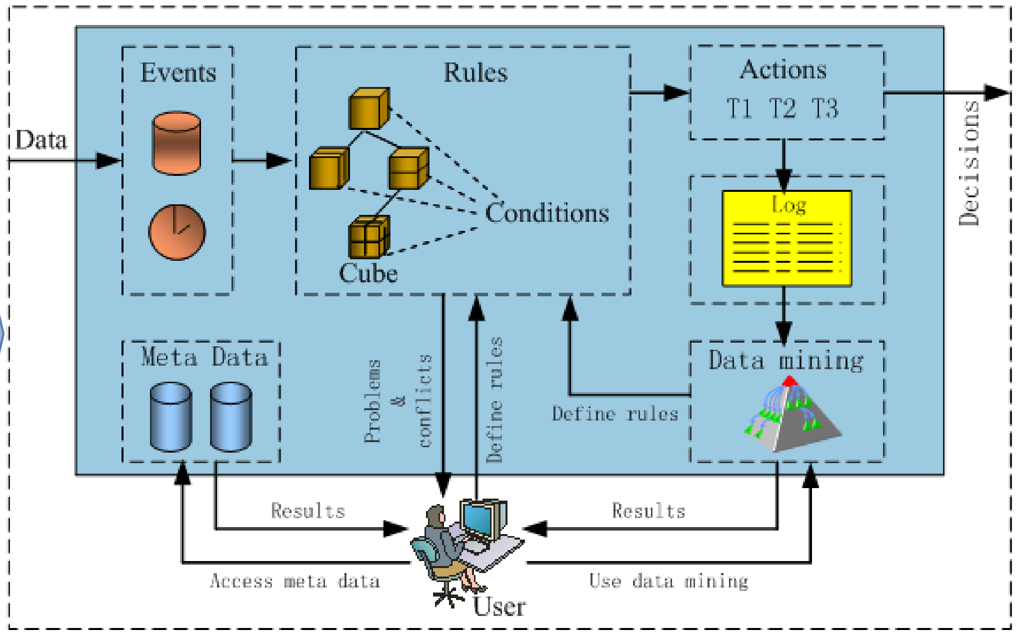
- Up to date, most of the research work on analysis rule is focused on its mechanism (e.g. [5,1]).In fact, performance optimization of analysis rules is also a critical aspect.
  - If more attention is paid to the optimization work, we can make full use of system resources, and
  - achieve better performance for analysis rules.

Introduction

- 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.

Getting the Information for Performance Optimization





•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.

Getting the Information for Performance Optimization

Definition 1. *Cube Using Pattern Matrix: Let* $C=\{c_0,c_1,\ldots,c_{m-1}\}$ *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.

Getting the Information for Performance Optimization

Definition 2. ***Frequent Cube Matrix:*** *Let* C=$\{c_0,c_1,\ldots,c_{m-1}\}$ *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.

Performance Optimization with ARPO Algorithm

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

begin
  get the time interval $[t_j, t_{j+1})$ to which the current time belongs;
  $i \leftarrow L.\ CubeID$;
  if $L.\ IsRealTime{=}TRUE$ **then**
    generate the cube c_i if not exist;
    execute L;
    if $A[i][j] = 1$ **then**
      materialize the cube c_i if it has not been materialized;
    else
      delete cube c_i;
    end
    return execution result of L;
  else
    if $([t_j, t_{j+1}) \in S)$ **or** $(U[i][j] \neq NULL)$ **then**
      initialize a waiting queue q_i if not exist;
      put $L.\ RuleInfoID$ into q_i;
      return q_i;
    end
  end
end

Empirical Study

- 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.

Empirical Study

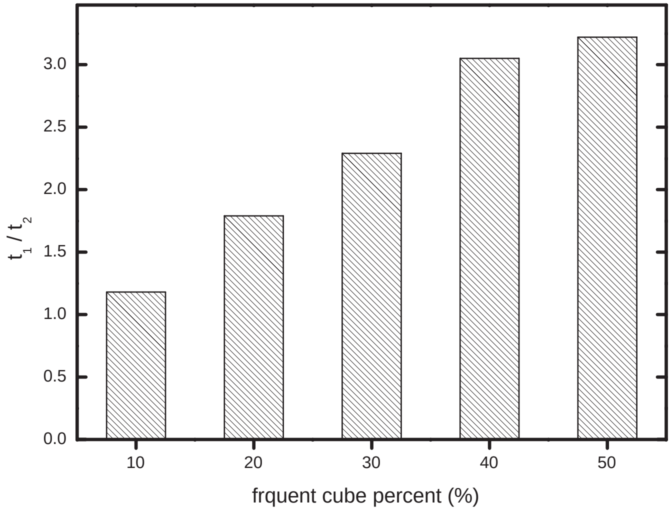


Fig. 2. Time cost ratio

- We can get from Fig.2 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 performance optimization.
- For example, when f=50%, time cost ratio t1/t2 can reach a high value of 3.22.

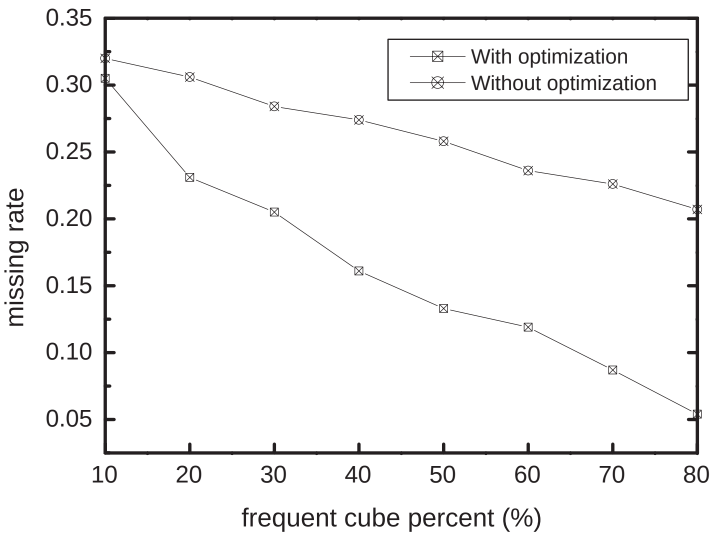


Fig. 3. Missing rate

- From 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.

Empirical Study

•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.

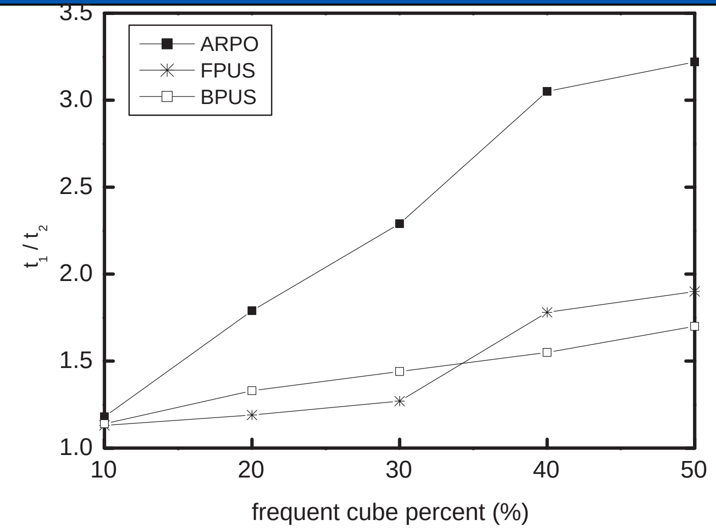


Fig. 4. Time cost ratio comparison

•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.

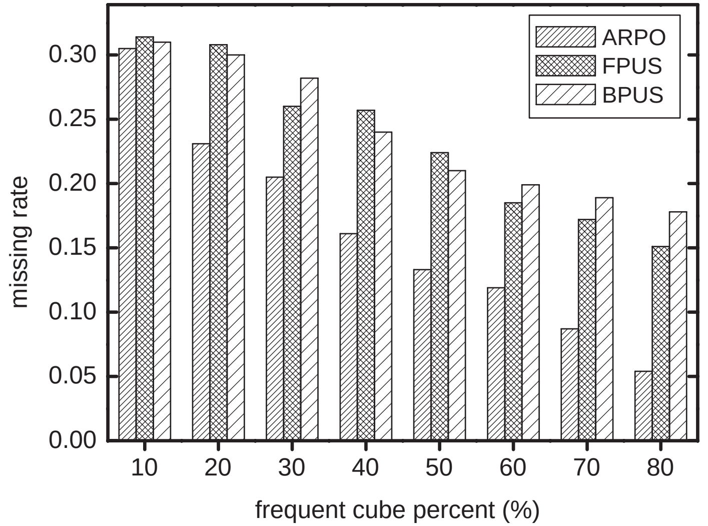


Fig. 5. Missing rate comparison

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.
- 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 DataWarehouses: 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 multidimensional 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 Integration and 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)

附件：报告人简介

林子雨

个人网页：http://www.cs.xmu.edu.cn/linziyu

男，1978年12月22日出生于浙江省温州市；小学、初中、高中就读于吉林省柳河县；1997年09月考入福州大学土木建筑工程系，2001年07月本科毕业并获得工学学士学位；2002年09月考入厦门大学信息科学技术学院计算机科学技术系攻读工学硕士学位；2005年09月考入北京大学信息科学技术学院计算机软件与理论专业攻读理学博士学位；2009年07月应聘于厦门大学计算机科学系，开始执教生涯……

2005.09~2009.07
博士

北京大学

厦门大学

福州大学

2002.09~2005.07 硕士

2009.07至今 讲师

1997.09~2001.07
本科

Thank You!