A Set Refreshing Algorithm for Data Warehouse On-Line Maintenance*

LI Zi-mu[1], LI Lei[2], ZHOU Xing-ming[3], WU Jian-ping[1]

[1](China Education and Research Network Center, Tsinghua University, Beijing 100084, China);
[2](School of Public Affairs, Tsinghua University, Beijing 100084, China);
[3](Computer School, National University of Defense Technology, Changsha 410073, China)
E-mail: zmli@263.net; zmli@cernet.edu.cn
http://www.edu.cn
Received May 28, 1999; accepted October 13, 1999

Abstract: In this paper, a Version-control Set Refreshing Algorithm (VSRA) is proposed, which uses incremental maintenance, version-control and batch processing mechanism to guarantee on-line maintenance and data consistency. VSRA not only reduces communication between database and data warehouse, but also promotes the refreshing efficiency of materialized views. Users can perform on-line analytical process at any time and get correct results with VSRA.
Key words: data warehouse; on-line maintenance; materialized view; set; version control

On-line maintenance of data warehouse has been attracting more attention recently. Reference [1] gives a three-tier data warehouse architecture as shown in Fig. 1. It has many advantages compared with the existing model showed in Fig. 2[2]. However, it often aggravates the traffic between DWbase and data warehouse. We propose a new algorithm called VSRA to solve this problem. It works with the three-tier architecture we proposed, collects data changes in databases and packs them together as a set to send to data warehouse. Data warehouse batch processes this set and applies the result to the materialized view for refreshing.

1 VSRA in Detail

VSRA is made up of three parts: view maintenance, version-control and OLAP query.

1.1 OLAP query

Definition 1. Message types in OLAP query of VSRA:

olap_start(q_i): message sent to data warehouse when query q_i starts.

olap_end(q_i): message for data warehouse when q_i finishes.

* This project is supported by the Ninth Five-Year Pre-Research Project of China ("九五"国防预研基金). **LI Zi-mu** was born in 1971. He got Ph. D. degree in National University of Defense Technology in 1999. Now, he works in China Education Research Network Center, Tsinghua University, as a post-doctor. His research interests include advanced computer networking architecture and distributed database. **LI Lei** was born in 1971. He received Ph. D. degree in National University of Defense Technology in 1999. He is now taking part in post doctor program in School of Public Affairs, Tsinghua University. His current research interests include computer network architecture, E-Government and E-Commerce. **ZHOU Xing-ming** was born in 1938. He is a member of Chinese Academy of Sciences. His research interests include advance computer architecture, distributed and parallel database system. **WU Jian-ping** was born in 1953. He is a professor of Tsinghua University. His current research interests are advanced computer networking architecture, network management and network security.

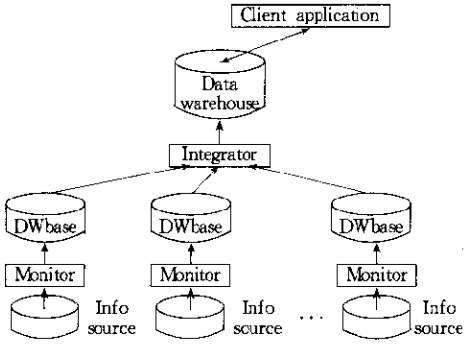

Fig. 1

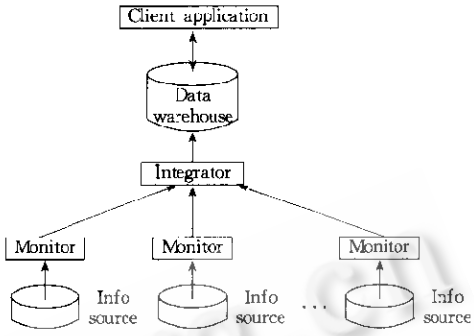

Fig. 2

dw_olap(q_i^*): message from data warehouse to DWbase where q_i^* is the drill-down part of q_i.

dwb_olap(a_i^*): message from DWbase to data warehouse where a_i^* is the answer to q_i^*.

sys_VA: message of version advancing for data warehouse and DWbase.

◀At data warehouse

Initialize: UQS←∅; //UQS is the abbreviation of Unanswered Query Set

—receive olap_start(q_i)

UQS←UQS+{q_i};

if q_i needs drill-down then {decompose the drill-down part q_i^* from q_i;

send dw_olap(q_i^*) to DWbase;

repeat { } until receive dwb_olap(a_i^*); }

execute q_i in data warehouse;

send olap_end(q_i) to data warehouse;

—receive dwb_olap(a_i^*)

commit a_i^* to the transaction started q_i;

—receive olap_end(q_i) UQS←UQS−{q_i};

if UQS=∅ then {send sys_VA to DWbase and data warehouse;}

◀at DWbase

—receive dw_olap(q_i^*)

calculate q_i^* with R_k^0 where R_k^0 is the tables of version 0 related with q_i^*;

a_i^*←answer of q_i^*;

send dwb_olap(a_i^*) to data warehouse;

1.2 View maintenance

Definition 2. For tables $R_1, R_2, \ldots, R_n$ in database and materialized view $V=Q(R_1, R_2, \ldots, R_n)$ in data warehouse where Q is a query of PSJ, there is a directed graph G in which V, $R_1, R_2, \ldots, R_n$ are nodes and R_i ($i=1,2,\ldots,n$) points to V by an arrow-line.

Definition 3. In directed graph G, if R_i points to V then R_i is the parent of V and is represented as $R_i=P(V)$.

Definition 4. ΔR_i^m is the changing set of table R_i of version m. U_i is the record of insertion/deletion of R_i in database. Key(U_i) is the attribute reflected in V according to its definition.

Definition 5. Message types and global variant in view maintenance of VSRA:

VN: the maximum version number in DWbase.

db_up(U_i): data changing message about R_i from Monitor to DWbase. DWbase will change U_i to its own format u_i when it receives this message.

dwb_rfsh(ΔV): view refreshing message from DWbase to data warehouse where ΔV is the part which should be applied to V.

sys_delta: message for stopping update ΔR_i^1 from system. The system sends this message when $VN=1$.

The algorithm first calculates ΔV based on $R_i^0, R_i^2, \Delta R_i^1$ and then sends ΔV to data warehouse. Data warehouse applies ΔV to the corresponding materialized view. In the algorithm, $\eta(Q, R_i)=Q(R_1^1,\ldots,R_{i-1}^1,\ldots,\Delta R_i^1,R_{i+1}^0,\ldots,R_n^0)$.

◀at DWbase

initialize: $\Delta R_i^1 \leftarrow \varnothing$, $VN \leftarrow 1$;

—receive db_up(U_i)

if $VN=2$ then {$R_i^2 \leftarrow R_i^2 \uplus u_i$; //if R_i^2 exists, then update R_i^2 and record $\Delta R_i^2 \leftarrow \Delta R_i^2 \uplus u_i$;} //$u_i$ in ΔR_i^2

else {$R_i^1 \leftarrow R_i^1 \uplus u_i$; //update R_i^1 because $VN=1$

$\Delta R_i^1 \leftarrow \Delta R_i^1 \uplus u_i$;} //record u_i in ΔR_i^1

—receive sys_delta

$R_i^2 \leftarrow R_i^1$; $\Delta R_i^2 \leftarrow \varnothing$; $VN \leftarrow 2$; $\Delta V \leftarrow \varnothing$;

while $R_i = P(V)$ do

{for $\forall U_j, U_k \in \Delta R_i$, if U_j is a insertion and U_k is a deletion and $j<k$, key(U_j)=key(U_k) then delete u_j and u_k from ΔR_i^1; //make ΔR_i^1 minimum

$\Delta V \leftarrow \Delta V \uplus \eta(Q, R_i)$; }

until V has found all of its parents;

send dwb_rfsh(ΔV) to data warehouse;

◀at data warehouse

—receive *dwb_rfsh*(ΔV)

$V^1 \leftarrow V^0 \uplus \Delta V$;

repeat { } until UQS=$\varnothing$;

send sys_VA;

1.3 Version advance

VSRA uses version control to guarantee data consistency between view maintenance and query. Version will advance when V^1 has been created and queries on V^0 are all finished. Following is the version advancing part of VSRA:

◀at data warehouse

—receive sys_VA

$V^0 \leftarrow V^1$; /* replace old version with new version */

◀at DWbase

—receive sys_VA

$R_i^0 \leftarrow R_i^1$; $R_i^1 \leftarrow R_i^2$; $\Delta R_i^1 \leftarrow \Delta R_i^2$ //replace old version with new version to keep consistency $VN \leftarrow 1$;

2 Extension of VSRA

VSRA improves the efficiency of data warehouse's on-line maintenance. But it may cause version advancing to be "starved". If the queries' time is too long and other queries started during this time, version cannot

advance until all of these queries finish. If this situation occurs circularly, version advancing will be "starved". We extend VSRA to three-version to solve this problem. In DWbase, R_i^1 is consistent with V^1, R_i^2 is the result of R_i^1 and ΔR_i^2. So we create version V^2 in data warehouse when version advances after all queries based on V^0 finished. V^2 is the result of V^1 and ΔR_i^2 and it is consistent with R_i^2. Three-version VSRA not only has all advantages of the two-version VSRA, but also does not need to extend to n version[3] to eliminate transaction restart and "starvation" in version advancing.

References:

[1] Li Zi-mu, Zhou Xing-ming. On-line maintenance and drill-down in data warehouse. Chinese Journal of Computers, 1999, 22(9):988～992 (in Chinese).

[2] Hammer, J., Garcia-Molina, H., Widom, J., *et al*. The Stanford data warehousing project. IEEE Data Engineering Bulletin, 1995,18(2):41～48.

[3] Quass, D. Materialized views in data warehouses [Ph. D. Thesis]. Stanford University, 1997.

附中文参考文献：

[1] 李子木,周兴铭.数据仓库的联机维护与下查.计算机学报,1999,22(9):988～992.

一种数据仓库联机维护的集合刷新算法

李子木[1]，李 磊[2]，周兴铭[3]，吴建平[1]

[1](清华大学 中国教育和科研计算机网络中心,北京 100084)

[2](清华大学 公共管理学院,北京 100084)

[3](国防科学技术大学 计算机学院,湖南 长沙 410073)

摘要：提出了一种版本控制集合刷新算法(VSRA).它采用增量维护、版本控制和批处理机制保证数据仓库的联机维护和数据一致性.VSRA 不仅减少了数据库和数据仓库之间的通信流量,而且提高了实体化视图的刷新效率.用户可以随时使用 VSRA 进行联机分析处理,并能得到正确的结果.

关键词：数据仓库;联机维护;实体化视图;集合;版本控制

中图法分类号：TP311 **文献标识码**：A