行政院國家科學委員會專題研究計畫成果報告

在資料倉儲中,具體化視界的有效的遞增維護

An Efficient Incremental Maintenance for Materialized view for Data Warehouse Strategies in concurrent control in TBDD

計畫編號：NSC 90-2416-H-039-001

執行期限：90 年 8 月 1 日至 91 年 7 月 31 日

主持人：吳 帆 執行機構及單位名稱：中國醫藥學院醫務管理研究所

計畫參與人員：孫漢屏　執行機構及單位名稱：中國醫藥學院醫務管理研究所

一、中文摘要

傳統的資料庫設計都是為了解決企業線上交易所需。資料倉儲讓決策者快速從不同角度分析資料，隨時掌握企業營運狀況，以增強企業的競爭力。目前資料倉儲系統大部分未特別考慮共時控制與如果線上交易資料更新時，資料倉儲如何迅速同步更新的問題。在這個計畫，我們提出一個資料倉儲資料結構，它可從 snapshot 建構資料倉儲。我們設計了無死結的可共時修改或查詢此資料倉儲的操作。這些共時操作的正確性(序列性)與無死結性也都被證明操。

關鍵詞：資料倉儲、線上分析處理、共時控制、B-tree、線上交易處理

Abstract

The goal of the conventional database is aimed at the online transaction processing of the enterprises. Through the help of the data warehouse, the decision maker can survey the transaction data from different views, monitoring the operations of the corporation and then strengthening the competitive advantage. Most of the current data warehouses are constructed without special considerations for the concurrent control and synchronized update. In this project, we propose a data structure for the data warehouse at first. We construct the data warehouse from the data gotten from the snapshot. Designs are made of deadlock-free operations that concurrently update or query the data warehouse when a transaction is made. The correctness of these concurrent operations and the deadlock-free property are proved.

Keywords: Data warehouse, OLAP, Concurrent control, B-tree, OLTP

2、Background and Goal

Since the competition among the enterprises, the computer techniques for the enterprises move from supporting the online transactions toward supporting the decision-making system [1, 2]. Utilizing the data warehouse, the decision maker can monitor the operations and strengthen the competitive advantage [3].

A sound data warehouse should provide a multi-user environment that the queries can be quickly response and the synchronization with the OLTP can be achieved easily. Recently, a lot of the data structures are proposed to speed up the response of the data warehouse [4-8]. However, they made little discussion about the concurrent problems of their structures.

In this project, we propose a data structure for the data warehouse to solve the problems such as how to construct the data warehouse from the data gotten from the snapshot, how to concurrently update the data warehouse when a transaction is made, how to provide other concurrent operations, under the requirements of minimum number of locked nodes.

三、Method and Result

We evaluate the effect of indexing, clustering and re-computing, and decide to construct a tree-like structure for data warehouse. A B-tree-like structure, called TBDD (Tree Based on Data Dimension), is proposed. The leaf node contains the raw data, while each non-leaf node contains the

statistical data. We survey the B-tree-like access methods [9-16]; we append a *right-link* pointer field into each non-leaf node to increase the concurrent degree. The structure of TBDD for the data warehouse of a corporation that has three dimensions (time, area and unit) is shown in Fig. 1. Whenever aggregate queries arrive, the system will search the non-leaf node to compute the answer. The following is the algorithm for such queries.

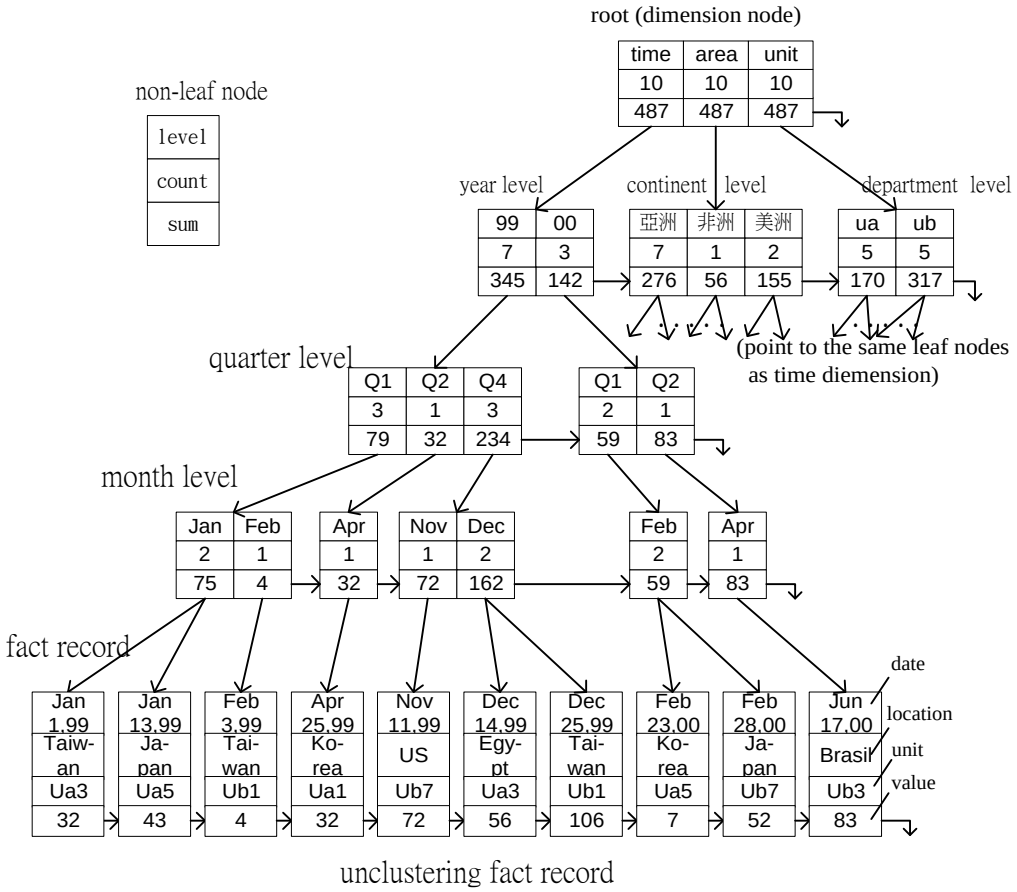

Fig. 1. An example of TBDD.

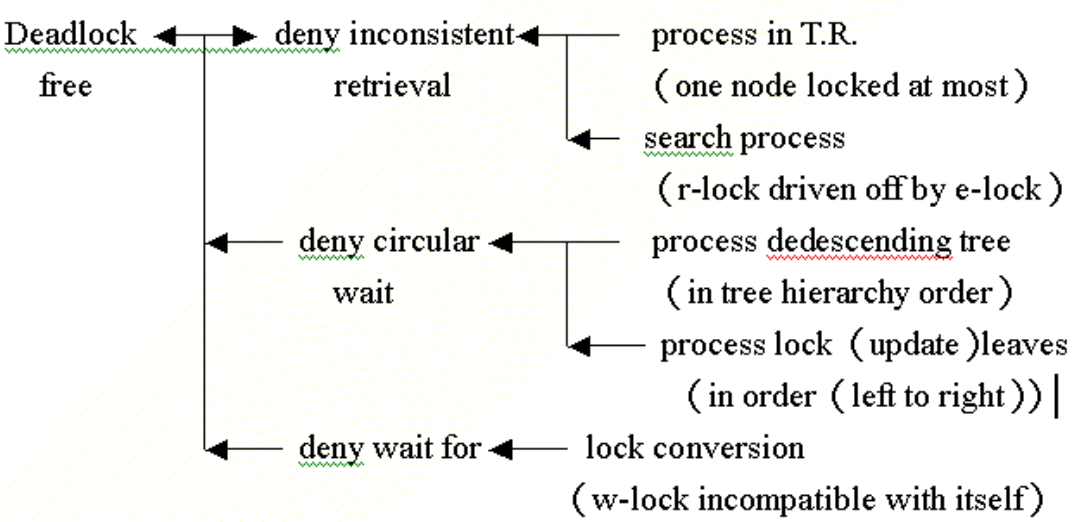

Fig. 2. Concept to prove deadlock free.

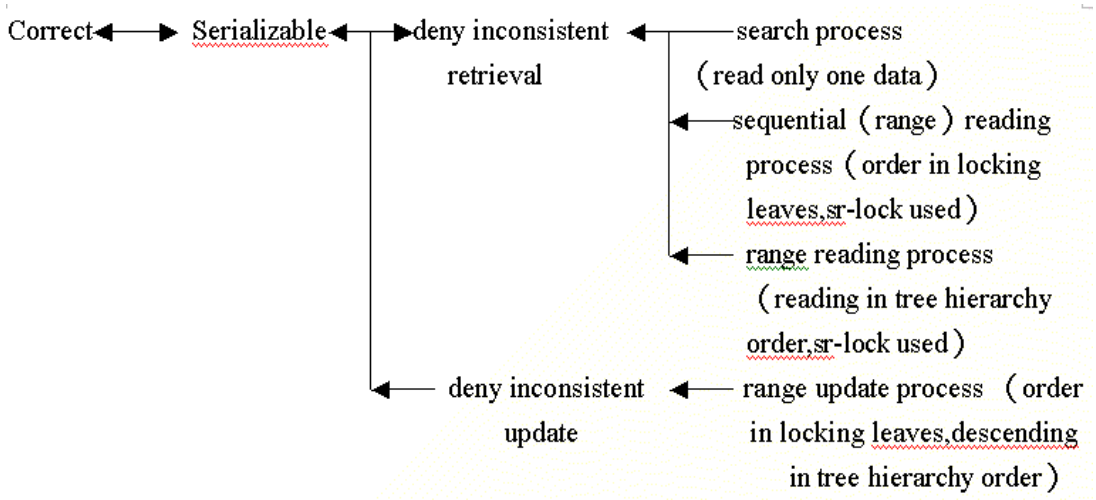

Fig. 3. Concept to prove serializability.

Query algorithm
S1 r-lock(root);
S2 descend from root to leaf without lock coupling;
S2.1 move right (if needed) without lock coupling to find correct node (lets it be N) that k may reside in;
S3 if k is in N then return(N);
else return(NULL).

The new data may insert into the data base. The data warehouse will be updated accordingly. An insertion operation in the TBDD can be divided into two parts: inserting data and reconstructing the tree if overflow occurs. f a data value is added into a data node, then the effect for this addition should also be included in the corresponding statistical fields in all the ancestors of this data node. When metadata are updated in non-leaf nodes, two directions, either top-down or bottom-up, can be chosen. With performance consideration, the insertion operation adopts top-down to update the metadata and releases the nodes already updated to allow other operations to access them.

Insertion algorithm
I1 initialize stack;
I2 search for leaf node N in which k may reside and w-lock(N);
I3 if k exists, w-unlock(N) and quit;
I4 descend from root to father of N in lock coupling (lock unit is a tuple of a node);
I4.1 move right (if needed) in lock coupling;
I4.2 add value to statistic data of all k's ancestors;
I4.3 push k's ancestors to stack;
I5 convert w-lock(N) to e-lock(N) and insert k to N;
I6 if overflowing in N, call reconstruction;
I7 clear stack.

Tree reconstruction is invoked if data overflows. The insertion operation in tree reconstruction ascends the tree level by level splitting and redistributing metadata in the nodes until it reaches a node that is non-full of data.

Reconstruction algorithm:
R1 split N into two nodes (N and N');
R2 e-unlock (N);
R2.1 pop N's original father from the stack;
R2.2 let old_N be N', and N be the father

of N;
R3 e-lock (N);
R3.1 move right without lock-coupling, if necessary;
R4 r-lock the *concerned* N's children and read their values;
R5 insert a key (the largest key in the leftmost brother of old_N) and a pointer (from N to old_N) to N, and redistribute the statistical data in N;
R6 if N overflows, repeat from R1 else e-unlock (N) and return.

In addition to the insertion, the users may modify the data node, say k, in the leaf node.

Modification algorithm
M1 search for leaf node N in which k resides and w-lock(N)；
M2 if k doesn't exist, w-unlock(N) and quit；
M3 given differences between new data and old one；
M4 descend from root to leaf in lock coupling for adding differences to statistic data of all k's ancestors.

When a deletion in the specified data node k in the deepest non-leaf node, a deletion operation needs to adjust the corresponding statistical data of all k's ancestors. The deletion operation is like the modification operation. It first searches the involved deepest non-leaf node containing the data node k. Then the operation marks the data instead of deleting it, since the overhead for merging is heavy.

Theorem 1. The interaction among n concurrent operations ($\{O_1, O_2, \ldots, O_n\}$, $n \geqq 1$) won't produce any deadlock .

Coffman *et al.* [17] stated that four necessary conditions must be in effect for a deadlock to exist. They are the (1) *non-preemption* condition, (2) *circular wait* condition, (3) *wait for* condition, and (4) *mutual exclusion* condition. To prove the deadlock–freedom for the concurrent operations, every operation will deny at least one condition of the above four conditions. The concept for the proof is shown in Fig. 2.

Theorem 2. Every concurrent operation correctly works (including terminating) in a multiprogramming environment as in a serial environment.

五、REFERENCES

1. G. Colliat. "*OLAP, relational, and multidimensional database systems*," SIGMOD Record, pp. 64-69, Vol. 25, No. 3, Sept. 1996.
2. S. Chaudhuri and U. Dayal. "*An overview of data warehouse and olap technology*," ACM SIGMOD Record, March 1997.
3. W.H. Inmon. *Building the Data Warehouse*. J. Wiley & Sons, Inc., second edition, 1996.
4. P. O'Neil and D. Quass. "*Improved query performance with variant indexes*," In Proceeding of the ACM SIGMOD international conference on management of data, pp. 38-49, Tucson, Arizona, May 1995.
5. Bayer, R. and McCreight, E., "Organization and Maintenance of

Large Ordered Indexes," *Acta Informatica*, Vol. 1, No. 3, 1972, pp. 173-189.

6. E. Baralis, S. Paraboschi and E. Teniente. "*Materialized view selection in a multidimensional database*," In Proc. of the 23th international conference on VLDB, pp. 156-165, Athens, Greece, Aug. 1997.
7. H. Gupta. "*Selections of views to materialize in a data warehouse*," In Proc. of ICDT, pp. 98-112, Jan. 1997.
8. N. Roussopoulos, Y. Kotidis, and M. Roussopoulos. "*Cubetree: organization of and bulk incremental updates on the data cube*," In Proceedings of the ACM SIGMOD Conference on Management of Data, pp. 89-99, Tucso, Arizona, May 1997.
9. Bayer, R. and Schkolnick, M., "Concurrency of Operations on B-trees," *Acta Informatica*, Vol. 9, 1977, pp. 1-21.
10. Bernstein, P.A., Hadzilacos, V., and Goodman, N., *Concurrency Control and Recovery in Database System*, Addison-Wesley, MA, 1987.
11. Guttman, A., "R-tree: a Dynamic Index Structure for Spatial Searching," in *Proceeding ACM SIGMOD Conference on Management of Data*, Vol. 14, No. 2, 1984, pp. 47-57.
12. Kwong, Y.S. and Wood, D., "A New Method for Concurrency in B-trees," *IEEE Trans. Soft. Eng.*, Vol. 8, No. 3, 1982, pp. 211-222.
13. Lehman, P.L. and Yao, S.B., "Efficient Locking for Concurrent Operations on B-trees," *ACM Trans. Database Syst.*, Vol. 6, No. 4, 1981, pp. 650-670.
14. Parr, J.R., "An Access Method for Concurrently Sharing a B-tree Based Indexed Sequential File," Technical Report, 36, Dep. Comput. Sci., Univ. of Western Ontario, London, Ont., Canada, April, 1977.
15. Srinivasan, V. and Carey, M.J., "Performance of B-tree Concurrency Control Algorithms," *ACM SIGMOD Conference on Management of Data,* 1991, pp. 416-425.
16. Srivastava, J., Tan, Jack S. Eddy, and Lum, V.Y., "TBSAM: An Access Method for Efficient Processing of Statistical Queries," *IEEE Trans. On Knowledge and Data Engineering*, Vol. 1, No. 4, 1989, pp. 414-423.
17. Coffman, E.G., E.G., Elphick, M.J., and Shoshani, A. "System deadlocks," *Computing Surveys*, Vol. 3, No. 2, 1971, p. 67-77.