# Efficient Incremental View Maintenance in Data Warehouses

Ki Yong Lee, Jin Hyun Son, Myoung Ho Kim
Korea Advanced Institute of Science and Technology
373-1, Guseong-Dong, Yuseong-Gu
Daejeon, South Korea
{kylee, jhson, mhkim}@dbserver.kaist.ac.kr

## ABSTRACT

In the data warehouse environment, the concept of a materialized view is nowadays common and important in an objective of efficiently supporting OLAP query processing. Materialized views are generally derived from select-project-join of several base relations. These materialized views need to be updated when the base relations change. Since the propagation of updates to the views may impose a significant overhead, it is very important to update the warehouse views efficiently. Though various view maintenance strategies have been discussed so far, they typically require too much access to base relations, resulting in the performance degradation.

In this paper we propose an efficient incremental view maintenance strategy called *delta* propagation that can minimize the total size of base relations accessed by analyzing the properties of base relations. We first define the delta expression and a delta propagation tree which are core concepts of the strategy. Then, a dynamic programming algorithm that can find the optimal delta expression are proposed. We also present various experimental results that show the usefulness and efficiency of the strategy.

## Keywords

View Maintenance, Data Warehouse, Materialized View

## 1. INTRODUCTION

The concept of a data warehouse as a common technology has been utilized to provide analysts and managers with strategic information about the key figures of the underlying business. Data warehouses, therefore, periodically extract and store the data needed for analytical purposes from remote information sources. Since each specific data content is of no interest at this level, almost all queries on data warehouses are related to statistics involving aggregates in support of on-line analytical processing (OLAP). These kinds of query patterns generally include entire table scans and/or relation joins which incur the high cost of query processing. In this regard, materialized views should be very effective in speeding up the OLAP queries as well as update processing, and are increasingly supported by commercial database systems.


*CIKM'01*, November 5-10, 2001, Atlanta, Georgia, USA.


The main objective of a materialized view is to improve query performance. However, when a warehouse is updated especially due to the changes of remote information sources, the materialized views must also be updated. While queries calling for up-to-date information are growing and the amount of data reflected to data warehouses has been increasing, the time window available for making the warehouse up-to-date has been shrinking. Hence, an efficient view maintenance strategy is one of the outstanding issues in the data warehouse environment. This can improve the performance of query processing by minimizing OLAP queries' down time and interference.

There can be roughly two different methods in reflecting data changes to materialized views: recomputation and incremental maintenance. Here, incrementally maintaining a materialized view includes computing and propagating only its changes. Compared to the sizes of base relations and views, their changes are generally very small. Hence, it is cheaper to compute only the changes of a view rather than to recompute it from scratch. In line with this, many possible strategies that allow incremental view maintenance have been proposed [1][2][3][4][5][6][7][8][9].

### 1.1 Related Work

Various kinds of materialized views can be formed over the base and/or view relations, i.e., projection views, select-project-join (SPJ) views, aggregation views, and so on. [1][2][10] proposed formal expressions through which we can incrementally maintain the SPJ views. The incremental view maintenance expressions which can additionally support the aggregation views were proposed in [3][4][7][9]. How to maintain views correctly using the previous maintenance expressions were discussed in [11][12][13] when multiple data sources are distributed and their changes occur concurrently. However, all these works did not mention how to select an efficient maintenance expression of many candidate expressions being able to be applied to the view maintenance.

There are two noticeable and comparable methods to incre-

mentally maintain a SPJ materialized view, the major form of a view in a data warehouse. For a n-way join view $V$ over base relations $R_i$ $(1 \leq i \leq n)$, i.e., $V = R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n$, [1] proposed the following expression to compute the change of view $V$, which is denoted by AV, and proved its correctness.

$$\begin{aligned} AV = & (\Delta R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n) \\ & \cup (R_1 \bowtie \Delta R_2 \bowtie \cdots \bowtie R_n) \\ & \cup \cdots \\ & \cup (\Delta R_1 \bowtie \Delta R_2 \bowtie \cdots \bowtie \Delta R_n) \end{aligned} \quad (1)$$

Here, $\Delta R_i$ denotes the change of base relation $R_i$. For updating view $V$, it is sufficient to compute its changes AV such as expression (1) and propagate AV to $V$. Expression (1) consists of $(2^n - 1)$ terms, each of which is a n-way join form of base and/or delta relations. For example, the changes AV of a 3-way join view $V$ over base relation $R_1$, $R_2$ and $R_3$ can be obtained by the expression:

$$\begin{aligned} AV = & (AR1 \bowtie R_2 \bowtie R_3) \cup (R_1 \bowtie \Delta R_2 \bowtie R_3) \cup \\ & (R_1 \bowtie R_2 \bowtie \Delta R_3) \cup (AR1 \bowtie \Delta R_2 \bowtie R_3) \cup \\ & (\Delta R_1 \bowtie R_2 \bowtie \Delta R_3) \cup (R_1 \bowtie \Delta R_2 \bowtie \Delta R_3) \cup \\ & (\Delta R_1 \bowtie \Delta R_2 \bowtie \Delta R_3) \end{aligned}$$

Since expression (1) requires too many terms in computing the change of a view, [3] proposed a new maintenance expression with only n terms in achieving the same goal as [1]. Namely,

$$\begin{aligned} AV = & (\Delta R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n) \\ & \cup (R_1' \bowtie \Delta R_2 \bowtie \cdots \bowtie R_n) \\ & \cup \cdots \\ & \cup (R_1' \bowtie R_2' \bowtie \ldots \bowtie R_{n-1}' \bowtie \Delta R_n) \end{aligned} \quad (2)$$

Here, $R_i'$ means the relation $R_i$ to which the change of $R_i$, $\Delta R_i$, is reflected, that is, $R_i' = R_i \cup \Delta R_i$. This propagation strategy also computes AV correctly using only n terms [3].

Recently, [14] presented an incremental maintenance algorithm that computes the change of view as a series of asynchronous steps. And, [15] proposed a maintenance algorithm that exploit common subexpressions between different view maintenance expressions. The algorithms developed in [14] and [15] are both based on the expression (2).

In general, there can be many possible expressions which can compute the change of a view according to the incremental view maintenance strategies such as [1] and [3]. Thus, it is important that we decide an efficient maintenance expression in order to improve the performance of view maintenance. For this point, [16] showed that the maintenance expression proposed in [3] has built the most efficient maintenance strategy so far.

## 1.2 Motivation

In the area of an incremental view maintenance, one of the most important factors as the performance measure is closely related to the total amount of access to base relations. As we mentioned in Section 1.1, a general vew maintenance expression is composed of base relations and their changes occurred by the changes of the remote information sources. Because the size of a base relation is commonly much bigger than that of its changes, it is critical to select an efficient maintenance expression which accesses as small amount of base relations as possible. With a view comprised of n base relations, the expression proposed in [1] has $(2^n - 1)$ terms as described in expression (1) and hence each base relation is accessed $(2^{n-1} - 1)$ times. On the other hand, [3] presented an expression with n terms in which each base relation is accessed $(n - 1)$ times. This expression has been known to be the most efficient among all strategies developed until now [16].

Note that the above two expressions consist of the terms identically formed to the view definition. In other words, when a *view* is defined as a join of n base relations $R_i$ $(1 \leq i \leq n)$, each term in the expressions is also a join form of corresponding n relations, i.e., $R_i$, $\Delta R_i$, or $R_i'$, as you can see in expression (1) and (2). In this point, previous view maintenance strategies have inherently a serious limitation that each base relation in the expressions must be accessed at least $(n - 1)$ times. If we can efficiently utilize the characteristics of a propagation order and the size of base relations, we can minimize the total amount of access to base relations in an objective of high performance view maintenance.

The following example motivates the need for different view maintenance policies. For a view $V$ over base relation $R_1$, $R_2$ and $R_3$, an expression to compute the changes of the view by [3] may be formed as

$$\begin{aligned} AV = & (AR1 \bowtie R_2 \bowtie R_3) \cup (R_1' \bowtie \Delta R_2 \bowtie R_3) \\ & \cup (R_1' \bowtie R_2' \bowtie \Delta R_3) \end{aligned} \quad (3)$$

If base relation $R_3$ is very large compared to $R_1$ and $R_2$, we can reduce the cost of computing AV considerably by minimizing the accesses of base relation $R_3$ using the following form:

$$\begin{aligned} AV = & (((AR1 \bowtie R_2) \cup (R_1' \bowtie \Delta R_2)) \bowtie R_3) \\ & \cup (R_1' \bowtie R_2' \bowtie \Delta R_3) \end{aligned} \quad (4)$$

In the comparison of expression (3) and (4), expression (4) makes base relation $R_3$ accessed only once by adjusting expression (3) with reflecting the size of base relations. This is because the propagation order of the changes of base relations is altered according to the view maintenance expression. The propagation order greatly affects the total amount of relations accessed, which will be substantially explained in Section 3. However, the choice of an optimal propagation order is not as straightforward as the above example may indicate. In this paper we propose an efficient incremental view maintenance strategy which guarantees high performance view maintenance. A algorithm that can find out an optimal maintenance expression are also presented.

In Section 2, we present our materialized view model and its cost model under which the performance of various view maintenance strategies can be evaluated. Our proposed view maintenance method called *delta* propagation is described with in-depth analyses in Section 3. The algorithm for an optimal view maintenance expression is proposed in Section 4. Section 5 gives results from performance experiments. Finally, we conclude our work by summarizing it with further work in Section 6.

## 2. PRELIMINARIES

In this section we first specify the view definition model in a data warehouse considered in the paper. And then, we discuss the cost model which is used for the performance estimation of view maintenance expressions.

### 2.1 View Definition Model

In data warehouse systems, the operational data are stored as fact tables or dimension tables through data extraction and integration from remote data sources [17][18]. In order to speed up query processing, data warehouses usually introduce the concept of a summary table defined over fact tables and dimension tables. Some summary tables can also be defined over other summary tables. With these summary tables, we can form some useful analytical information in advance and efficiently support relevant OLAP queries. In this paper the fact tables and dimension tables are called as base relations, and the summary tables as materialized views. If a summary table is used to define other summary table, the summary table can also be called a base relation. When remote information sources are changed, the changes are propagated to the warehouse periodically. The period has been usually daily or weekly so that updates are performed in a batch manner. But, nowadays the cycle of the change propagation is getting shorter and the time window available for making the warehouse up-to-date is required to be shrunken more and more. Hence, the study on the efficient view maintenance mechanisms is being vigorously performed. In the procedure of the change propagation, the base relations in a data warehouse are first updated and then the materialized views have to be maintained to reflect the updates done against the base relations.

A materialized view can be variously defined over base relations and/or other materialized views. In this paper we consider materialized views defined by the select-project-join (SPJ) expression. The SPJ views as a general form of a view definition can cover the most part of materialized views in a data warehouse environment. Moreover, a SPJ view can be easily extended to accommodate one with aggregation using generalized projections [7]. A view $V$ over n base relations $(R_1, R_2, \ldots, R_n)$ is defined as follows:

$$V = \Pi_L \sigma_C (R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n)$$

where $L$ is a list of project attributes and C is a selection condition.

For a base relation $R$, the notation AR means the changes of $R$. In AR, there are additional count information indicating the occurrence of each tuple. The positive count value for each tuple of AR represents the insertion of so many copies of the tuple into the base relation $R$. The negative count value, on the contrary, means the deletion of so many copies of the tuple from the base relation $R$. The union operator, $\cup$, unifies two relations with the count information. The details are described in [3]. $R'$ represents the relation $R$ to which AR are reflected. Hence, $R'$ can be denoted as: $R' = R \cup \mathrm{AR}$. Especially, the propagation of AR to $R$ will be denoted by the expression $R \leftarrow R \cup \mathrm{AR}$. For example, if the changes of a view $V$ is computed as $(\Delta R_1 \bowtie R_2) \cup (R_1' \bowtie \Delta R_2)$, then the update of $V$ can be expressed as follows:

$$V \leftarrow V \cup (\Delta R_1 \bowtie R_2) \cup (R_1' \bowtie \Delta R_2)$$

### 2.2 Cost Model

In this paper we adopt the linear work metric developed in [16] as a cost model to compare many applicable view maintenance expressions. Although the linear work metric is relatively simple, it can estimate the cost of processing complex maintenance expressions very well [16]. In the linear work metric, the cost of processing a maintenance expression is the sum of the costs of processing each term of the expression. Because each term is a join of relations, the processing cost of a term is proportional to the total size of the relations included in the term. Let Cost(E) be the cost of processing the expression $E$. If a view $V$ is defined as $R_1 \bowtie R_2$ and its changes AV is given as $(\Delta R_1 \bowtie R_2) \cup (R_1' \bowtie \Delta R_2)$, then the cost of processing AV can be computed as follows:

$$\begin{aligned} Cost(\Delta V) &= Cost((\Delta R_1 \bowtie R_2) \cup (R_1' \bowtie \Delta R_2)) \\ &= Cost(\Delta R_1 \bowtie R_2) + Cost(R_1' \bowtie \Delta R_2) \\ &= c \cdot (|\Delta R_1| + |R_2|) + c \cdot (|R_1'| + |\Delta R_2|) \end{aligned} \quad (5)$$

Here, c is a proportionality constant and $|R|$ is the size of relation $R$. Thus an efficient maintenance expression must be one that can reduce the cost of processing AV, $Cost(\Delta V)$, by minimizing the accesses of base relations.

## 3. DELTA PROPAGATION STRATEGY

We present an efficient view maintenance method called *delta* propagation which can minimize the cost of computing the changes of a view. Because a view is built by joining several base relations and its changes can be computed by propagating the changes of each base relation, the change propagation order has close relation to the total size of relations accessed during computing the changes of a view. In general, the change propagation order can be decided by the delta propagation tree defined in the following. As a result, our method finds out an optimal delta propagation tree.

### 3.1 Delta Expression

For a set of base relations $P = \{R_s, R_t, \ldots, R_u\}$, we define two notations $\bowtie P, \bowtie P'$ as follows:

$$\bowtie P = R_s \bowtie R_t \bowtie \ldots \bowtie R_u$$
$$\bowtie P' = R_s' \bowtie R_t' \bowtie \ldots \bowtie R_u'$$

Using these notations, the definition of the *delta* expression can be stated as:

**Definition** 1. For a join expression of n base relations, i.e., $R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n$, $Delta(\{R_1, R_2, \cdots, R_n\})$ meaningthe changes of this join expression, i.e., $\Delta(R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n)$ occurred by the changes of some base relations is defined as:

$$\begin{aligned} &Delta(\{R_i\}) = \Delta R_i \\ &Delta(\{R_1, R_2, \cdots, R_n\}) = \\ &\quad Delta(P_1) \bowtie (\bowtie P_2) \bowtie (\bowtie P_3) \bowtie \cdots \bowtie (\bowtie P_m) \\ &\quad \cup (\bowtie P_1') \bowtie Delta(P_2) \bowtie (\bowtie P_3) \bowtie \cdots \bowtie (\bowtie P_m) \\ &\quad \cdots \\ &\quad \cup (\bowtie P_1') \bowtie (\bowtie P_2') \bowtie \cdots \bowtie (\bowtie P_{m-1}') \bowtie Delta(P_m), \end{aligned}$$

where $\{P_1, P_2, \cdots, P_m\}$ is a partition of $\{R_1, R_2, \cdots, R_n\}$ such that $1 < m \leq n$, $\bigcup_{i=1}^{m} P_i = \{R_1, R_2, \ldots, R_n\}$, $P_i \neq \phi$ $(1 \leq i \leq m)$ and $P_i \cap P_j = \phi$ $(i \neq j)$. □

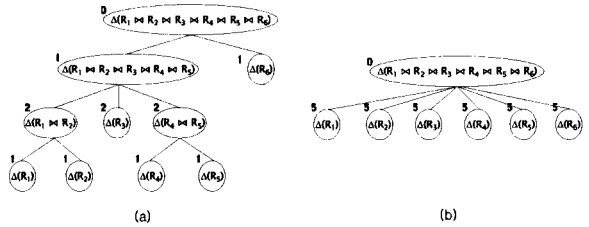

**Figure 1: Delta Propagation Trees**

**THEOREM** 1. The *delta expression* $Delta(\{R_1, R_2, \ldots, R_n\})$ *computes the* changes of view $V = R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n$ *correctly.*

**PROOF.** The proof is straightforward and trivial. □

Especially, when $P_1 = \{R_1\}$, $P_2 = \{R_2\}$, $\cdots$, and $P_n = \{R_n\}$, $Delta(\{R_1, R_2, \cdots, R_n\})$ is resolved into the same expression as expression (1) in Section 1.1. Suppose that a view $V$ is defined as $R_1 \bowtie R_2 \bowtie R_3$. With $P_1 = \{R_1, R_2\}$ and $P_2 = \{R_3\}$, an expression of AV using the delta expression can be specified as:

$$\begin{aligned} AV &= Delta(\{R_1, R_2, R_3\}) \\ &= Delta(\{R_1, R_2\}) \bowtie R_3 \cup R_1' \bowtie R_2' \bowtie Delta(\{R_3\}) \\ &= (Delta(\{R_1\}) \bowtie R_2 \cup R_1' \bowtie Delta(\{R_2\})) \bowtie R_3 \\ &\quad \cup R_1' \bowtie R_2' \bowtie \Delta R_3 \\ &= (\Delta R_1 \bowtie R_2 \cup R_1' \bowtie \Delta R_2) \bowtie R_3 \\ &\quad \cup R_1' \bowtie R_2' \bowtie \Delta R_3 \qquad (6) \end{aligned}$$

As you can notice in expression (6), the base relation $R_3$ is accessed only once in computing AV. If we use an expression like expression (3), $R_3$ must be accessed two times for the same work. Hence, we can reduce the cost of computing AV considerably, especially, in case that relation $R_3$ is very large compared to relation $R_1$ and $R_2$.

Definition 2. An incremental view maintenance strategy based on the delta expression is called the **delta propagation strategy.** For a view $V = R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n$, the delta propagation strategy maintains the view with the following expression.

$$V \leftarrow V \cup Delta(\{R_1, R_2, \cdots, R_n\})$$

□

## 3.2 Cost Analyses of the Delta Propagation Strategy

In this section, we analyze our delta propagation strategy in the basis of the cost model specified in Section 2.2. As we mentioned, [16] showed that the propagation strategy of [3] forms the most efficient maintenance expression of the expressions proposed so far. Thus, we will compare our delta propagation strategy with it.

Any delta expression can be represented as a tree like Figure 1 which we call a **delta propagation tree** in the paper. For each delta expression, there exists a unique delta propagation tree corresponding to it. For a view $V = R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n$, the root node of its delta propagation tree is $\Delta(R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n)$. Assume that for each node $\Delta(R_s \bowtie R_t \bowtie \cdots R_u)$ a partition $\{P_1, P_2, \cdots, P_k\}$ of $\{R_s, R_t, \cdots, R_u\}$ *is* given. Then the children of the node $\Delta(R_s$ cd $R_t \bowtie \cdots R_u)$ are $\Delta(P_1)$, $\Delta(P_2)$, $\cdots$, $\Delta(P_k)$ with $\Delta(P_i)$ being the ith child from the left. Here, each node $\Delta(R_s \bowtie R_t \bowtie \ldots R_u)$ means the delta expression $Delta(R_s \bowtie R_t \bowtie \ldots R_u)$. For example, Figure 1 shows two delta expressions, each of which is defined as follows:

(a)

$$\begin{aligned} &Delta(\{R_1, R_2, R_3, R_4, R_5, R_6\}) = \\ &\quad Delta(\{R_1, R_2, R_3, R_4, R_5\}) \bowtie R_6 \\ &\quad \cup R_1' \bowtie R_2' \bowtie R_3' \bowtie R_4' \bowtie R_5' \bowtie Delta(\{R_6\}) \\ &Delta(\{R_1, R_2, R_3, R_4, R_5\}) = \\ &\quad Delta(\{R_1, R_2\}) \bowtie R_3 \bowtie R_4 \bowtie R_5 \\ &\quad \cup R_1' \bowtie R_2' \bowtie Delta(\{R_3\}) \bowtie R_4 \bowtie R_5 \\ &\quad \cup R_1' \bowtie R_2' \bowtie R_3' \bowtie Delta(\{R_4, R_5\}) \\ &Delta(\{R_1, R_2\}) = \\ &\quad Delta(\{R_1\}) \bowtie R_2 \cup R_1' \bowtie Delta(\{R_2\}) \\ &Delta(\{R_4, R_5\}) = \\ &\quad Delta(\{R_4\}) \bowtie R_5 \cup R_4' \bowtie Delta(\{R_5\}) \end{aligned}$$

(b)

$$\begin{aligned} &Delta(\{R_1, R_2, R_3, R_4, R_5, R_6\}) = \\ &\quad Delta(\{R_1\}) \bowtie R_2 \bowtie R_3 \bowtie R_4 \bowtie R_5 \bowtie R_6 \\ &\quad \cup R_1' \bowtie Delta(\{R_2\}) \bowtie R_3 \bowtie R_4 \bowtie R_5 \bowtie R_6 \\ &\quad \cup R_1' \bowtie R_2' \bowtie Delta(\{R_3\}) \bowtie R_4 \bowtie R_5 \bowtie R_6 \\ &\quad \cup R_1' \bowtie R_2' \bowtie R_3' \bowtie Delta(\{R_4\}) \bowtie R_5 \bowtie R_6 \\ &\quad \cup R_1' \bowtie R_2' \bowtie R_3' \bowtie R_4' \bowtie Delta(\{R_5\}) \bowtie R_6 \\ &\quad \cup R_1' \bowtie R_2' \bowtie R_3' \bowtie R_4' \bowtie R_5' \bowtie Delta(\{R_6\}) \end{aligned}$$

As you can notice, Figure 1-(b) is a delta propagation tree derived from the view maintenance expression (2) proposed in [3]. Our delta propagation tree, therefore, can cover all possible propagation strategies in the incremental maintenance problem. We can estimate how many base relation $R_i$ or $R_i'$ is accessed in a delta propagation tree. By inference from the definition of the delta expression, if a delta expression is composed of n terms, the number of access to each base relation would be (n − 1). Here, n terms is to be siblings in the same depth of a delta propagation tree. Because the delta expression is resolved to relations $R_i$, $R_i'$, or $\Delta R_i$ by recursively applying Definition 3.1, its delta propagation tree can be a tree with height $h$ ($h \geq 1$) and subtrees of the tree are also to be other delta expression. Hence, each node in the tree is corresponding to delta expression defined in the paper and its $k$ children, who are siblings one another, are to be terms of the expression specified by the delta expression. Thus, the figure stated in the upper left corner of a node in Figure 1 is the number of siblings except itself and means that base relations included in the delta expression of the node are accessed so many times during the incremental view maintenance procedure. With a delta

propagation tree, the access number of base relation $R_i$ during the view maintenance is equal to the sum of the figures from the leaf node A(&) to the root node.

In case of Figure l-(a), base relation $R_1$, $R_2$, $R_3$, $R_4$, $R_5$, and $R_6$ are accessed 4, 4, 3, 4, 4, and 1, respectively, during the view maintenance. In Figure l-(b), all base relations are accessed 5 times. Note that we can reduce the number of access to base relation $R_6$ to one time. In consequence, the cost of accessing base relations can be reduced considerably if we use the delta propagation strategy proposed in the paper and can find out an optimal delta propagation tree.

THEOREM 2. *For* any **delta** propagation *tree generated by the delta* expression, *the* number **of access** *to base* **relation** $R_i$ ($1 \leq i \leq$ n), denoted *by* $A_i$, is less than *equal to* $(n - 1)$ *where* n *is the* number ***of*** **base relations participating in** *the* **view definition, i.e.,** $1 \leq A_i \leq (n - 1)$.

PROOF. We assume that the number of access to a base relation is equal to frequencies that the base relations appears in the maintenance expression. From the Definition 3.1 of the delta expression, the access to the base relation $R_i (1 \leq i \leq$ n) can only occur in the term that includes $Delta(P_k)$ where $R_i \notin P_k$. Thus, the frequencies that $R_i$ appears in any delta expression can not exceed the number of all possible forms of $Delta(P_k)$. Because $P_k$ is a subset of $\{R_1, R_2, \ldots, R_n\}$ and any two $P_k$ are disjoint each other, the number of all possible forms of $Delta(P_k)$ is at most $|\{R_1, R_2, \ldots R_n\} - \{R_i\}| = (n - 1)$. Hence, this corollary may follow. □

Let us consider the delta propagation strategy under the linear work metric which is the cost model specified in Section 2.2. When a view $V$ is defined as $R_1 \bowtie R_2 \bowtie R_3$, the cost, of computing the changes of $V$, denoted by $Cost(\Delta V)$ or $Cost(\Delta(R_1 \bowtie R_2 \bowtie R_3))$, *can* be as follows:

In case of applying the view maintenance strategy proposed in [3],

$$\begin{aligned} Cost(\Delta(R_1 \bowtie R_2 \bowtie R3)) &= Cost(\Delta R_1 \bowtie R_2 \bowtie R3) \\ &\quad + Cost(R_1' \bowtie \Delta R_2 \bowtie R3) \\ &\quad + Cost(R_1' \bowtie R_2' \bowtie \Delta R_3) \\ &= 2|R_1' + |R_2| + |R_2'| \\ &\quad + 2|R_3| + |\Delta R_1| \\ &\quad + |\Delta R_2| + |\Delta R_3| \end{aligned} \quad (7)$$

In case of applying our delta propagation strategy with the following delta expressions:

$$Delta(\{R_1, R_2, R_3\}) = Delta(\{R_1, R_2\}) \bowtie R_3 \cup R_1' \bowtie R_2' \bowtie Delta(\{R_3\})$$

$$Delta(\{R_1, R_2\}) = Delta(\{R_1\}) \bowtie R_2 \cup R_1' \bowtie Delta(\{R_2\})$$

$$\begin{aligned} Cost(Delta(\{R_1, R_2, Rs\})) &= 2|R_1'| + |R_2| + |R_2'| + |R_3| \\ &\quad + |\Delta(R_1 \bowtie R_2)| + |\Delta R_1| \\ &\quad + |\Delta R_2| + |\Delta R_3| \end{aligned} \quad (8)$$

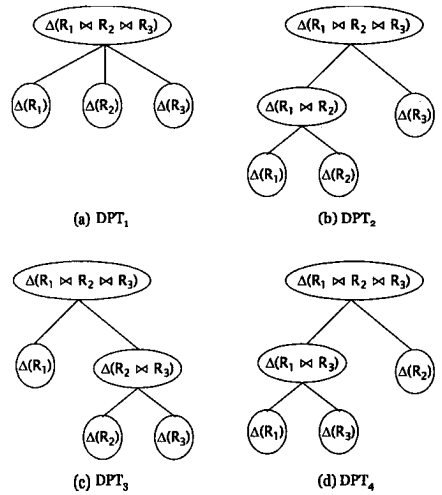


**Figure 2: Possible Delta Propagation Trees of** $Delta(\{R_1, R_2, R_3\})$

While expression (7) includes $2|R_3|$, expression (8) has $R_3| + |\Delta(R_1 \bowtie R_2)|$ instead of $2|R_3|$. This means that in expression (7) relation *R3* is accessed once more and the temporal relation $\Delta(R_1 \bowtie R_2)$ is additionally accessed in expression (8). If we assume that relation $R_3$ *is* very large compared to $R_1$ and $R_2$, we can consider that $|R_3|$ is bigger than $|\Delta R_1$ and $|\Delta R_2|$ so that $|R_3|$ is to be bigger than $|\Delta(R_1 \bowtie R_2)|$, because the change relations can be assumed to be generally much smaller than their base relations. Thus, the cost of expression (2) is less than that of expression (1) under the assumption that relation $R_3$ is much bigger than $R_1$ and $R_2$. As we mentioned in the paper, the major limitation of [3]'s strategy is that every base relation must equally be accessed (n − 1) times regardless of its size. In contrast to this, the delta propagation strategy can reduce the number of access to such large relations like $R_3$ as in expression (8), resulting in reducing the linear work metric cost.

Now, the problem of how to find out, an optimal delta expression (or delta propagation tree) which can minimize the view maintenance cost remains to us. In the next section, we will discuss methods to find the optimal delta expression.

## 4. OPTIMAL DELTA EXPRESSION

In this paper, the optimal delta expression means one that requires the minimal maintenance cost in the aspect of the cost, model defined in Section 2.2. We now present a dynamic programming algorithm that can find the optimal delta expression.

For a set, $\{R_1, R_2, \cdots, R_n\}$, the number of possible partitions is B(n), which is the nth Bell number [19]. Thus, for $Delta(\{R_1, R_2, \ldots, R_n\})$, there can be B(n) delta expressions based on the partition of $\{R_1, R_2, \ldots, R_n\}$. For a large value n, B(n) is approximately $n^n$ [20]. Therefore, we need to find the optimal delta expression among these expressions to minimize the maintenance cost. For example, Figure 2 shows some possible delta propagation trees of $Delta(\{R_1, R_2, R_3\})$. Each delta propagation tree repre-

sents the corresponding delta expression. Based on the delta expression of Definition 3.1, the maintenance cost for each delta propagation tree can generally be computed by the following form when we assume the proportionality constant c with 1 in the cost model defined in Section 2.2.

$$
\begin{aligned}
&Cost(Delta(\{R_1, R_2, \cdots, R_n\})) = \\
&\quad \sum_{i=1}^{m} Cost(Delta(P_i)) \\
&\quad + (|\Delta(\bowtie A)] + |P_2| + \cdots + |P_m|) \\
&\quad + (|P_1'| + |\Delta(\bowtie P_2)| + \cdots + |P_m|) \\
&\quad + \cdots \\
&\quad + (|P_1'| + |P_2'| + \cdots + |\Delta(\bowtie P_m)|) \qquad (9)
\end{aligned}
$$

For $P_i = \{R_s, R_t, \cdots, R_u\}$ $(1 \leq i \leq m)$, the notation $|P_i|$, $|P_i'|$, *and* $|\Delta(\bowtie P_i)|$ denote $|R_s| + |R_t| + \cdots + |R_u|$, $|R_s'| + |R_t'| + \ldots + |R_u'|$ and $|\Delta(R_s \bowtie R_t \bowtie \cdots \bowtie R_u)|$, respectively. The optimal cost of the delta expression is the cost of an optimal propagation tree, i.e., the minimum cost of all delta propagation trees. Hence, an optimal delta expression is one with constructed by an optimal propagation tree.

Because dynamic programming is typically applied to optimization problems, we define the value of an optimal solution in finding an optimal delta expression as the minimum cost value in expression (9). In the computation of expression (9) in dynamic programming, a way to compute $|\Delta(\bowtie P_i)|$ needs, while $|P_i$ and $|P_i'|$ can be easily computed. When $JS(R_1, R_2)$ denotes the join selectivity of relation $R_1$ and $R_2$, i.e., the expected size of the join result divided by the maximum size $|R_1| \times |R_2|$, then $|R_1 \bowtie R_2|$ is equal to $JS(R_1, R_2) \times R_1 \times R_2|$. We can easily extend this to any number of relations as follows:

$$
|R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n| = \prod_{(1 \leq i,j \leq n) \wedge (i<j)} JS(R_i, R_j) \times |R_1| \times |R_2| \times \cdots \times |R_n|
$$

If we assume that $|(\Delta R \bowtie S) \cup (R' \bowtie AS)]$ is equal to $|\Delta R \bowtie S| + |R' \bowtie AS]$, then we have

$$
\begin{aligned}
&|\Delta(R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n)| = \\
&\prod_{(1 \leq i,j \leq n) \wedge (i<j)} JS(R_i, R_j) \times \\
&\quad \{(|\Delta(\bowtie P_1)| \times \prod_{R_k \in P_2} |R_k| \times \cdots \times \prod_{R_k \in P_m} |R_k|) \\
&\quad + (\prod_{R_k' \in P_1'} |R_k'| \times |\Delta(\bowtie P_2)| \times \cdots \times \prod_{R_k \in P_m} |R_k|) \\
&\quad + \cdots \\
&\quad + (\prod_{R_k' \in P_1'} |R_k'| \times \prod_{R_k' \in P_2'} |R_k'| \times \cdots \times |\Delta(\bowtie P_m)|)\}
\end{aligned}
$$

From the above expressions, we can easily obtain the cost value of expression (9).

Our dynamic programming algorithm named as *FindOptimalDeltaExpression* based on the above observations is presented in Figure 3. *FindOptimalDeltaExpression* finds out an optimal delta expression for a given view, based on

```
Procedure FindOptimalDeltaExpression
Input: view V = R1 ⋈ R2 ⋈ . . . ⋈ Rn

  /* Initialize */
  - Find n optimal delta expression for each base
    relation.

  For (i = 2 to n) do
    - Find out an optimal delta expression for a subset
      with size i of {R1, R2, ···, Rn},
      using already obtained optimal delta expression
      for the subsets of size 1 to (i − 1).
      (There exist Cn,i subsets.)
  End For

  /* As a result, we can obtain an optimal delta
     expression for the input view V */

End Procedure
```

**Figure 3: Dynamic Programming Algorithm**

the bottom-up approach. The algorithm first finds the optimal delta expression for each base relation, which is unique as you expect. Then, it iteratively finds an optimal delta expression for each subset of $\{R_1, R_2, \ldots, R_n\}$ from size 2 to n. When we try to search an optimal delta expression for a subset of size i, the algorithm uses optimal delta expressions already computed for the subsets of size 1 to (i − 1). Eventually, the algorithm can find out an optimal delta expression for the view defined over $n$ base relations.

To find out an optimal delta expression for a subset $P$ of $\{R_1, R_2, \cdots, R_n\}$, the algorithm traces all possible partitions of $P$. Because the number of possible partitions for the subset of size $i$ is equal to $B(i)$[19], the total number of partitions the algorithm needs to consider is

$$
\sum_{i=1}^{n} C_{n,i} \times B(i),
$$

where $C_{n,i}$ is the number of combinations of size $i$ from $n$ distinct relations and so means the number of subsets of size i. Therefore, the time complexity of the algorithm is approximately

$$
O(\sum_{i=1}^{n} C_{n,i} \times B(i)) \approx O(2^n \times B(n)).
$$

For a large value n, $O(2^n \times B(n))$ is approximately $O(2^n \times n^n)$. We know that this algorithm's time complexity is very high due to its exhaustive search space. However, since many materialized views in a data warehouse may be defined over less than 10 base relations in practice, the algorithm can be utilized properly in finding out the optimal delta expression for these kinds of views.

## 5. EXPERIMENTS

We show in this section various performance experimental results among several view maintenance mechanisms. In the experiments, we used the TPC-D benchmark schema and data [21]. And, Oracle7 database system running on a Sun Ultra-Enterprise with 256MB RAM is used as the data ware-

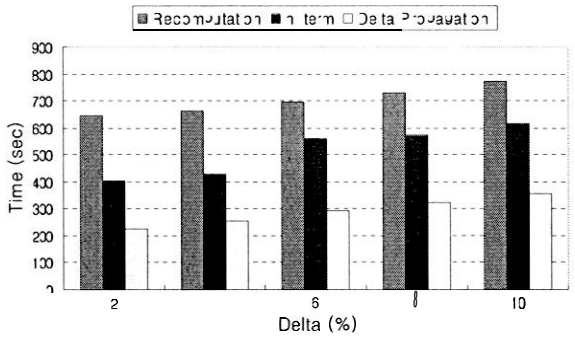


**Figure 4: The performance evaluation by varying the size of changes**

house environment. We assume that the view used in the experiments is defined as the join of six base relations, i.e., *CUSTOMER, ORDERS. LINEITEM. SUPPLIER. NATION,* and *REGION* from [21]. Thus, the view V is represented as follows:

$$V = CUSTOMER \bowtie ORDERS \bowtie LINEITEM \bowtie SUPPLIER \bowtie NATION \bowtie REGION$$

In the experiments, we compare our delta propagation strategy with the recomputation method and the maintenance strategy proposed in [3]. Let us call [3]'s view maintenance strategy n-term in this paper. As we mentioned in Section 1.1, [16] proved that the n-term strategy is the best strategy among existing incremental strategies. When we change the base relations from 2% to 10% of their original size, the time required to execute each view maintenance strategy is shown in Figure 4. The changes of base relations can be done by inserting or deleting tuples. From Figure 4, we can see that incremental view maintenance strategies, i.e., the n-tam strategy and our delta propagation strategy are more efficient than the recomputation strategy within 10%s' change. And, our delta propagation strategy outperforms other strategies.

According to the cost model specified in the paper, the size of base relations may considerably affect the performance of view maintenance strategies. Figure 5 shows the performance evaluation among three different methods by scaling the size of base relations from 50% to 200%. As a result, our proposed method gains a performance benefit of about 100% compared to the recomputation method and 80% compared to the $n$-term method. Hence, we can confirm that our delta propagation strategy is more efficient than existing ones in maintaining materialized views.

## 6. CONCLUSIONS AND FUTURE WORK

Data warehouses store a large amount of summarized data to support decision making process. These summarized data can be seen as materialized views defined over some data sources. When data sources change, these materialized views need to be updated to reflect the changes of data sources. Since the updates of views may impose a significant overhead on the warehouse, it is very important to update the warehouse views efficiently. We presented the delta propagation strategy that can incrementally maintain materialized

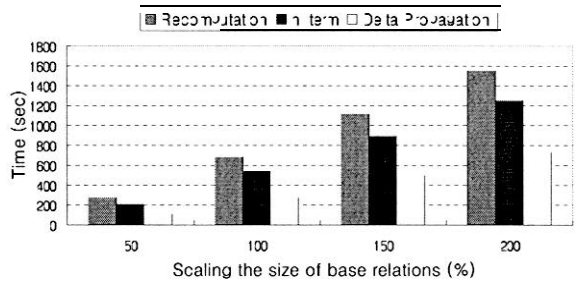


**Figure 5: The performance evaluation by scaling the size of base relations**

views efficiently in the data warehouse environment. The concept of the delta expression and a delta propagation tree with considering the size of each base relation makes it possible to minimize the cost of maintaining views. As a result, the experimental results show the efficiency of the proposed strategy compared to previous strategies.

In general, data warehouses have many materialized views which may share some base relations. In this case, if we can reuse the intermediate results generated during maintaining some materialized views in updating other materialized views, we can considerably improve the overall performance of the data warehouse system. For this purpose, we are currently developing a view maintenance algorithm reusing the results of delta expressions among views.

## 7. REFERENCES

[1] J. A. Blakeley, P. Larson, and F. W. Tompa, "Efficiently Updating Materialized Views," In Proceedings of ACM SIGMOD Conference, pp. 61-71, 1986.

[2] S. Ceri and J. Widom, "Deriving production rules for incremental view maintenance," In Proceedings of the 7th International Conference on Very Large Databases, pp. 108-119, 1991.

[3] A. Gupta, I. S. Mumick, and V. S. Subrahmanian, "Maintaining views incrementally," In Proceedings of ACM SIGMOD Conference, pp. 157-166, 1993.

[4] T. Griffin and L. Libkin, "Incremental maintenance of views with duplicates," In Proceedings of ACM SIGMOD Conference, pp. 328-339, 1995.

[5] A. Gupta and I. S. Mumick, "Maintenance of Materialized Views: Problems, Techniques, and Applications," IEEE Data Engineering Bulletin, Special Issue on Materialized Views and Data Warehousing, Vol. 18, No. 2, 1995.

[6] L. S. Colby, T. Griffin, L. Libkin, I. S. Mumick, and H. Trickey, "Algorithms for deferred view maintenance," In Proceedings of ACM SIGMOD Conference, pp. 469-492, 1996.

[7] D. Quass, 'Maintenance expressions for views with aggregation," In Workshop on Materialized Views: Techniques and Applications, June, 1996.

[8] I. S. Mumick, D. Quass, and B. S. Mumick, "Maintenance of Data Cubes and Summary Tables in a Warehouse," In Proceedings of ACM SIGMOD Conference, 1997.

[9] H. Gupta and IS. Mumick, "Incremental maintenance of aggregate and outerjoin," Technical Report, Stanford University, 1999.

[10] X. Qian and G. Wiederhold, "Incremental recomputation of active relational expressions," IEEE Transactions on Knowledge and Data Engineering, pp. 337-341, 1991.

[11] Y. Zhuge, H. Garcia-Molina, J. Hammer, and J. Widom, "View Maintenance in a Warehousing Environment," In Proceedings of ACM SIGMOD Conference, 1995.

[12] Y. Zhuge, H. Garcia-Molina, and J. L. Wiener, "The Strobe Algorithms for Multi-Source Warehouse Consistency," In Proceedings of the International Conference on Parallel and Distributed Information Systems, December, 1996.

[13] D. Agrawal, A. E. Abbadi, A. Singh, and T. Yurek, "Efficient View Maintenance at Data Warehouses," In Proceedings of ACM SIGMOD Conference, 1997.

[14] K. Salem, K. Beyer and B. Lindsay, "How To Roll a Join: Asynchronous Incremental View Maintenance," In Proceedings of ACM SIGMOD Conference, pp. 129-140, 2000.

[15] H. Mistry, R. Roy, S. Sudarshan, K. Ramamritham, "Materialized View Selection and Maintenance Using Multi-Query Optimization," In Proceedings of ACM SIGMOD Conference, 2001.

[16] W. J. Labio, R. Yerneni, and H. Garcia-Molina, "Shrinking the Warehouse Update Window," In Proceedings of ACM SIGMOD Conference, 1999.

[17] J. Widom, "Research Problems in Data Warehousing," In Proceedings of 4th International Conference on Information and Knowledge Management, November, 1995.

[18] W. H. Immon, *Building* the Data Warehouse, Wiley Computer Publishing, 1996.

[19] B. Niamir, "Attribute Partitioning in a Self-Adaptive Relational Database System," Technical Report, Cambridge, Mass.: Laboratory for Computer Science, Massachusetts Institute of Technology, 1978.

[20] M. Hammer and B. Niamir, "A Heuristic Approach to Attribute Partitioning," In Proceedings of ACM SIGMOD Conference, pp. 93-101, 1979.

[21] TPC Committee, Transaction Processing Council, http://www.tpc.org/