Extended Conditions for Answering an Aggregate Query Using Materialized Views

Jae-young Chang and Sang-goo Lee
Department of Computer Science
Seoul National University
{jychang, sglee}@cygnus.snu.ac.kr

Keywords: databases, materialized view, data warehouse, aggregate query, query processing, query reformulation

INTRODUCTION

A view is a virtual relation defined in terms of base relations and is said to be materialized if it is stored in the database. In the past few years, materialized views have attracted significant amount of research in many applications environments as a means of enhancing query performance. Materialized views offer significant performance advantages in evaluating a query by eliminating the need for recomputing the views. Just as a cache, a materialized view is a type of replicated copy of information derived from base relations and hence, the technique is very useful in various applications where access to local or cached views may be cheaper than access to base relations.

Although many approaches for utilizing materialized views in evaluating a query have been proposed [1, 2, 4, 5], there were several restrictions in selecting such views. First, only views whose relations are contained in those of a query were considered. Thus, views referring to relations not mentioned in the query were excluded from consideration. Furthermore, views also excluded if they do not contain the necessary attributes for the query. However, there are many cases where a view cannot be utilized if the above restrictions are not satisfied.

In this paper, we propose a new approach to using materialized views in answering an aggregate query. We extend the practical scope of utilizing materialized views which include those that would have been excluded in previous approaches. We first show the condition for testing whether materialized views can be utilized in answering an aggregate query. These conditions are extended by introducing a method that recovers the missing attributes. All conditions are designed to be applicable to the bag (multiset) semantics. Bag semantics are used in the underlying model of most practical database management systems.

QUERY MODE AND NOTATIONS
We consider queries with group-by and aggregate notation for notational convenience. All queries and materialized views are represented as relational algebra expressions using the following notations:

\[ \pi_{\text{proj}}(R) : \text{Project relation attributes proj.} \]

\[ \sigma_{\text{cond}}(R) : \text{Select relation attributes that satisfy predicates cond.} \]

\[ R \bowtie S \text{ Natural join relation attributes on participating relations.} \]

e. \( R = \{ R_1, \ldots, R_n \} \) represents \( R_1 \bowtie \ldots \bowtie R_n \)

\[ G_{\text{group, agg}}(R) \text{ Group a relation attributes and apply aggregate expressions} \]

We refer to the operation \( G_{\text{group, agg}} \) as \( GA(\text{Group-by and Aggregation}) \) operation. We will use capital letters to represent relations and materialized views, and boldface letters to denote sets of relations. In addition, we define the following notations:

- table \((Q)\) : set of relations referred in query \( Q \)
- attr \((R)\) : set of attributes in \( R \)
- proj \((Q)\) : set of non-aggregated projection attributes in \( Q \)
- proj \(_k(Q)\) : subset of projection attributes \( attr(R) \)
- cond \((Q)\) : set of predicates in \( Q \)
- cond \(_k(Q)\) : subset of predicates defined on \( attr(R) \)
- group \((Q)\) : set of grouping attributes in \( Q \)
- group \(_k(Q)\) : subset of grouping attributes \( attr(R) \)
- agg \((Q)\) : set of aggregate expressions in \( Q \)
- agg \(_k(Q)\) : subset of aggregate expressions defined on \( attr(R) \)

### 3. FOUNDATIONS FOR QUERY REFORMULATION

In this section, we present important properties of the relational model that are helpful in describing our proposal. We briefly review transformation rules of \( GA \) operation for which we propose to reduce the query processing. Let us consider the following queries \( Q \) and \( Q' \):

\[ Q : G_{\text{group}, \text{agg}}(R \bowtie T) \quad (1) \]

\[ Q' : G_{\text{group}, \text{agg}}(G_{\text{group}, \text{agg}}(R) \bowtie T) \quad (2) \]
Intuitively, for $Q$ and $Q'$ to be equivalent grouping results should be contained in each group. These conditions can be represented as:

1. $\text{group}(Q) \leftrightarrow \text{group}'(Q)$
2. $\text{group}(Q) \rightarrow \text{group}(R)$

These conditions are equivalent if

- $\text{group}(Q)$ and $\text{group}(Q')$ are the same, and
- Each group on $\text{group}'(Q)$ should be contained in one and only one group when $R$ is grouped on $\text{group}(R)$.

These conditions can be represented using functional dependencies ($\alpha \rightarrow \beta$ is denoted by $\alpha \rightarrow \beta$, and $\alpha \leftrightarrow \beta$ is denoted by $\alpha \leftrightarrow \beta$).

Let us then consider the relationship between $\text{agg}'$, $\text{agg}''$, and $\text{agg}(Q)$. As an example, if $\text{SUM}(b)$ is in $\text{agg}(Q)$, $\text{SUM}(b)$ should be in $\text{agg}''$ to compute the partially aggregated value in each subgroup, and $\text{SUM}(N)$ should be in $\text{agg}'$ to merge the partially aggregated values, where $N$ is the renamed attribute corresponding to $\text{SUM}(b)$ in $\text{agg}''$.

Table 1. Aggregation decomposition rules

<table>
<thead>
<tr>
<th>$f(b)$</th>
<th>$f'(b')$</th>
<th>$f''(b'')$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>$f$</td>
<td>$b'$</td>
</tr>
<tr>
<td>$f'$</td>
<td>$f''$</td>
<td>$f''$</td>
</tr>
<tr>
<td>$R$</td>
<td>$\text{SUM}$</td>
<td>$N$</td>
</tr>
<tr>
<td></td>
<td>$\text{SUM}$</td>
<td>$b$</td>
</tr>
<tr>
<td></td>
<td>$\text{COUNT}$</td>
<td>$N$</td>
</tr>
<tr>
<td></td>
<td>$\text{COUNT}$</td>
<td>$b$</td>
</tr>
<tr>
<td></td>
<td>$\text{MIN}$</td>
<td>$N$</td>
</tr>
<tr>
<td></td>
<td>$\text{MIN}$</td>
<td>$b$</td>
</tr>
<tr>
<td></td>
<td>$\text{MAX}$</td>
<td>$N$</td>
</tr>
<tr>
<td></td>
<td>$\text{MAX}$</td>
<td>$b$</td>
</tr>
<tr>
<td>$T$</td>
<td>$\text{SUM}$</td>
<td>$N*b$</td>
</tr>
<tr>
<td></td>
<td>$\text{SUM}$</td>
<td>$s$</td>
</tr>
<tr>
<td></td>
<td>$\text{COUNT}$</td>
<td>$N$</td>
</tr>
<tr>
<td></td>
<td>$\text{COUNT}$</td>
<td>$s$</td>
</tr>
<tr>
<td></td>
<td>$\text{MIN}$</td>
<td>$b$</td>
</tr>
<tr>
<td></td>
<td>$\text{MIN}$</td>
<td>$X$</td>
</tr>
<tr>
<td></td>
<td>$\text{MAX}$</td>
<td>$b$</td>
</tr>
<tr>
<td></td>
<td>$\text{MAX}$</td>
<td>$X$</td>
</tr>
</tbody>
</table>

where $N$ is the attribute corresponding to $f''(b'')$, $s$ is arbitrary attribute(s), and $X$ is needless.

**Definition 3.1**

Given a query $Q$ such that $\text{table}(Q) = R \cup T$ (and $R$ and $T$ are disjoint), the following conditions are satisfied, $G_{\text{group}(Q), \text{agg}(Q)}$ is GA-decomposable into $G_{\text{group}', \text{agg}'}$ and $G_{\text{group}'', \text{agg}''}$ with respect to $R$.

1. $\text{group}(Q) \leftrightarrow \text{group}'$ and $\text{group}'' \rightarrow \text{group}(R)$ hold in $Q$.

2. For each $f(b) \in \text{agg}(Q)$, $\text{agg}'$ and $\text{agg}''$ contain the aggregates expressions corresponding to $f(b)$.
The following definition specifies the role of each relation in the query with respect to each operation.

**Definition 3.2**

Let $Q$ be a query such that $\text{table}(Q) = R \cup T$ (and $R$ and $T$ are disjoint).

1. If the following two expressions are equivalent, $R$ is join-useless in $Q$.
   
   $\pi_{\text{proj}(Q)} (R \bowtie T)$
   
   $\pi_{\text{proj}(Q)} (T)$

2. If the following two expressions are equivalent, $R$ is selection-useless in $Q$.
   
   $\sigma_{\text{cond}(Q)} (R \bowtie T)$
   
   $\sigma_{\text{cond}(Q)} (T)$

3. If $\text{proj}(Q) \subseteq \text{attr}(T) \cap \text{attr}(R)$, $R$ is projection-useless in $Q$.

4. If $\text{group}(Q) \subseteq \text{attr}(T) \cap \text{attr}(R)$, $R$ is grouping-useless in $Q$.

5. If all aggregate expressions in $\text{agg}(Q)$ are on $\text{attr}(T)$, $R$ is aggregation-useless in $Q$.

The meaning of the term "join-useless" is that the rows of relations $T$ are not eliminated by the joins with relations in $R$. Selection-useless indicates that the rows of $T$ are not eliminated by the selection predicate on $R$. Projection- and grouping-useless mean that $\text{proj}(Q)$, $\text{group}(Q)$, and $\text{agg}(Q)$ are composed of attributes of $T$.

**QUERY REFORMULATION**

Intuitively, if a materialized view $V$ contains some intermediate result that is needed in the process of evaluating a query $Q$, the cost of processing $Q$ can be reduced by generating intermediate results from $V$. Let $Q$ and $V$ follow:

- $Q : \pi_{\text{proj}(Q), \text{agg}(Q)} G_{\text{group}(Q), \text{agg}(Q)} \sigma_{\text{cond}(Q)} (R \bowtie T)$
- $V : \pi_{\text{proj}(V), \text{agg}(V)} G_{\text{group}(V), \text{agg}(V)} \sigma_{\text{cond}(V)} (R \bowtie S)$

Without loss of generality, we assume that $T$ and $S$ do not have any common attributes except for those participating in the natural joins in $Q$. Since $\text{table}(Q)$ contains $R$ that is also contained in $\text{table}(V)$ if $Q$ is reformulating $V$, the formulated query $Q'$ follows:

- $Q' : \pi_{\text{proj}(Q), \text{agg}'} G_{\text{group}', \text{agg}'} \sigma_{\text{cond}'} (V \bowtie T')$
Let $\text{cond}(Q)$ and $\text{cond}'(Q')$ respectively be equivalent lent. For simplicity, first consider the case where $S \cup \text{cond}(Q')$. Condition 2.

The conditions for $V$ aggregation-useless may contain attributes of $S$. Condition 3.

Since we assumed that $T$ subdivides each group on $G$. There exists a set of predicates cond' such that $\pi$ and $T$ contain all attributes of $R$. We can transform $\text{cond}(Q)$ and $\text{cond}'(Q')$ into the following expression by unfolding $\text{proj}(V)$ agg'.

Therefore, $G$ operations with $S$ defined in (4) contains a GA which has been eliminated or duplicated by joining with $R$. However, the other predicates in $V$ agg should also be selection-useless in $V$.

Selection operation, $\text{cond}(Q)$ should be equivalent to $\text{cond}'(Q')$. In addition, $\text{proj}(V)$ should contain all attributes mentioned in group' $\cup$ $\text{cond}'(Q')$ with respect to $R$ by Definition 3.1. For selection operation, $\text{cond}(Q)$ should be equivalent to $\text{cond}'(Q')$ $\cup$ $\text{cond}_R(V)$ However, the predicate in $\text{proj}(V)$ agg may eliminate the row of $R$ satisfying $\text{cond}_R(V)$. Therefore, $\text{selection-useless} V$. In addition, $\text{proj}(V)$ should contain all attributes mentioned in group' $\cup$ $\text{cond}'(Q')$ with respect to $R$.

Since we assumed that $T$ and $S$ have a common attribute except the common attributes and attributes in group' $\cap$ attr $(R) \cap$ attr $(T)$. In summary, the sufficient condition for $\text{cond}(Q)$ and $\text{cond}'(Q')$ respectively be equivalent lent follows:

**Condition 1.** There exists a predicate cond' such that $\text{cond}(Q)$ is equivalent to $\text{cond}'(Q')$ $\cup$ $\text{cond}_R(V)$, and $S$ join-useless in $V$.

**Condition 2.** There exists group' and agg' such that $G_{\text{group}(Q), \text{agg}(Q)}$ is GA-decomposable into $G_{\text{group}(Q), \text{agg}(Q)}$ and $G_{\text{group}(Q), \text{agg}(Q)}$ with respect to $R$.

**Condition 3.** $\text{proj}(V) \nsubseteq \{a \in \text{attr}(R) \mid \text{a mentioned in group'} \cup \text{cond'}(Q') \}$

Let us consider the case where $S$ aggregating-useless $V$. Although $\text{group}(V)$ may contain attributes of $S$ (i.e. $\text{group}(V)$), these are ignored because $\text{group}(V)$ merely subdivides each group of $\text{group}(V)$. However, group' $\cap$ attributes of $R$ and $T$ (i.e. group') is final. Operation Therefore, even when $S$ no grouping or aggregation-useless $V$. When these evaluating $\text{cond}(Q)$ conditions are satisfied.

**Theorem.**

Let $\text{cond}(Q)$ define by expression (3) and (4) respectively $\text{cond}(Q)$ and $\text{cond}'(Q')$ are satisfied equivalent $Q'$.

**Proof.**

We transform $\text{cond}(Q)$ following expression by unfolding $V$. $\pi_{\text{proj}(Q), \text{agg}} G_{\text{group}(Q), \text{agg}} \sigma_{\text{cond}(Q)} (\pi_{\text{proj}(Q), \text{agg}} G_{\text{group}(V), \text{agg}(V)} \sigma_{\text{cond}'(Q')} R \Join S)) \Join T$ (6)

The $G$ operations $G_{\text{group}(Q), \text{agg}}$ and $G_{\text{group}(V), \text{agg}}$ merge into $G_{\text{group}(Q), \text{agg}}$ by Condition 3.
As discussed in the previous section, if \( \text{proj}(V) \) does not contain the necessary attributes in \( W \), then (3) cannot be used to transform \( W \) into the following expression:

\[
\pi_{\text{proj}(Q)} \text{ and } \pi_{\text{agg}(Q)} \text{ as }
\]

Therefore, (9) is equivalent to (7) if and only if cond' is join-useless in \( Q \).

Example 4.1
Consider the following schema (key underlined).

<table>
<thead>
<tr>
<th>Schema</th>
<th>Sales(eid, item, time, emp)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dept (dept, manager, loc)</td>
</tr>
<tr>
<td></td>
<td>Item(item, brand, size, weight)</td>
</tr>
</tbody>
</table>
| Rows | Requires the following query on a materialized view \( V \).
| \( Q \) | \( \pi_{\text{eid, manager, SUM(\text{vol})}} G(eid, manager, \text{SUM(\text{vol})}) \text{ as } \text{comput} \) \( \text{Emp} \bowtie \text{Sales} \bowtie \text{Dept} \) |
| \( V \) | \( \pi_{\text{eid, dept, item, brand, SUM(\text{vol}), \text{sales_amount}} \text{ as } \text{comput}} \) \( \text{Emp} \bowtie \text{Sales} \bowtie \text{Item} \) |

Based on this schema, suppose the following query on a materialized view \( V \).

\[
\pi_{\text{eid, manager, SUM(\text{vol})}} G(eid, manager, SUM(\text{vol})) \text{ as } \text{comput} \] 

This query can be transformed into the following:

\[
\pi_{\text{eid, manager, SUM(\text{sales_amount})}} G(eid, manager, SUM(\text{sales_amount})) \text{ as } \text{comput} \] 

\( V \bowtie \text{Dept} \)

if and only if \( \text{Villeubonnebetteplanl} \) \( \text{muchmallethan Sales and Emp} \text{ is cut and pasting evaluating Q Villeubonnebetteplanl muchmallethan V} \)
evaluating $Q$ have decided that $V$ and used answering $Q$ however, we can recover those attributes joining with views as relations. $V$ and utilized $L$ assume that $m$ is missing attribute in which we have defined follows:

$$V: \pi_{\text{proj}(V) \cup m, \text{agg}(V)} \ G_{\text{group}(V) \cup m, \text{agg}(V)} \sigma_{\text{cond}(V)} R \bowtie S \quad (10)$$

$$V: \pi_{\text{proj}(V) \cup m, \text{agg}(V)} \ G_{\text{group}(V) \cup m, \text{agg}(V)} \sigma_{\text{cond}(V)} R \bowtie U \quad (11)$$

where $m \subseteq \text{attr}(R) \ \text{proj}(V)$ and $m \subseteq \text{proj}(V)$

We assume that $V$ and $V'$ have common attributes except those participating the natural join $V$ and $V'$ respectively. Then in order to correctly recover $m$ without loss of information, the join of $V$ and $V'$ should eliminate the other $V$.

This condition formally presented the following lemma.

**Lemma.**

Let $V$ be defined as follows: $el$ recover joining with $V'$. Then

1. $\text{join-useless in join-useless} \quad (V \bowtie V)$, and
2. $\text{cond}_k(V)$ logically implies $\text{cond}_k(V \text{ and } \text{selection-useless}'$.

**Proof.**

We prove the lemma by showing the following expressions are equivalent.

$$E_1: \pi_{\text{proj}(V) \cup m, \text{agg}(V)} \ G_{\text{group}(V) \cup m, \text{agg}(V)} \sigma_{\text{cond}(V)} R \bowtie S$$

$$E_2: \pi_{\text{proj}(V) \cup m, \text{agg}(V)} \ V \bowtie V'$$

Both expressions except additional projection attributes $m$ and $E_1$ expression foremost from $V$. Let $E_2$ be where $V$ and $V'$ unfolded from $E_1$ then is added $\text{proj}(V \text{ and } \text{group}(V))$ as follows:

$$E_3: \pi_{\text{proj}(V) \cup m, \text{agg}(V)} \ ((\pi_{\text{proj}(V) \cup m, \text{agg}(V)} \ G_{\text{group}(V) \cup m, \text{agg}(V)} \sigma_{\text{cond}(V)} R_1 \bowtie S)) \bowtie (\pi_{\text{proj}(V) \cup m, \text{agg}(V)} \ G_{\text{group}(V) \cup m, \text{agg}(V)} \sigma_{\text{cond}(V)} R_2 \bowtie U))$$

Each $\text{table}(V \text{ and } \text{table}(V') \text{renamed as } R_1$ and $R_2$ respectively). Then, $V$ and $E_1$ equivalent to $E_2$ and $E_2$ is equivalent to $E_3$. Since $\text{participate}\bowtie \text{join} E$ and $V$, we know that $E_3$ eliminated by $\text{join} E$ and $V$. Hence although $V'$ join-useless $V \bowtie V'$, $V'$ may not join-useless $E_1 \bowtie V'$ however since $\text{cond}_k(V)$ logically implies $\text{cond}_k(V \text{ and } \text{join} \text{and selection-useless } V'$, for each $t$ in $\pi_{\text{attr}(V)} \ \text{cond}(V) R_1 \bowtie S$ is $\pi_{\text{attr}(V)} \ \text{cond}(V)$, $E_1 \bowtie V'$ may not join-useless $E_2$ and $E_3$ only on $R_1$ and $R_2$. Hence $\text{join-useless } (E_1 \bowtie V'$ addition in all attributes in $\text{proj}(V) \cup m$.
arprojection \( E_1 \), it is projection-useless. Therefore, \( V' \) can be removed from \( E \) hence, \( E \) is equivalent to \( E_3 \).

Since \( \text{proj}(E) \) does not affect the result of \( E_1 \circ V' \), the result is not dependent on \( \text{proj}(V) \). In other words, \( \text{proj}(V) \) functionally determines \( m \) in \( E_1 \). Therefore, \( E_3 \) is equivalent to \( E_1 \).

Since \( m \) in \( \text{proj}(E_1) \) does not affect the overall result of \( (E_1 \circ V') \), the values of \( m \) are dependent on \( \text{proj}(V) \). In other words, \( \text{proj}(V) \) functionally determines \( m \) in \( E_1 \). Therefore, the result of the group-by operation on \( \text{group}(V) \cup m \) is the same as that of the group-by operation on \( \text{group}(V) \). As a result, even if \( m \) is removed from \( \text{proj}(E_1) \) and \( \text{group}(E_1) \), the overall result of \( E_3 \) is not changed. Therefore, \( E_3 \) is equivalent to \( E_2 \).

Example 5.1

Based on the schema defined in Example 4.1, consider the following views.

\[ V : \pi_{\{\text{eid}, \text{name}, \text{item}\}}, \text{SUM(\text{vol})} \text{G}_{\{\text{eid}, \text{name}, \text{item}\}}, \text{SUM(\text{vol})} \text{(Emp \Join Sales)} \]

\[ V' : \pi_{\text{eid}, \text{name}, \text{salary}, \text{manager}} \text{(Emp \Join Dept)} \]

Assume that we want to recover \( m = \{\text{salary}\} \). Here, we can let \( R, S \) and \( U \) defined in (10) and (11) be \( \{\text{Emp}\} \), \( \{\text{Sales}\} \) and \( \{\text{Dept}\} \) respectively. Then, \( V \) and \( V' \) satisfy Lemma 1, and thus, the following expressions are equivalent.

\[ \pi_{\{\text{eid}, \text{name}, \text{salary}, \text{item}\}}, \text{SUM(\text{vol})} \text{G}_{\{\text{eid}, \text{name}, \text{salary}, \text{item}\}}, \text{SUM(\text{vol})} \text{(Emp \Join Sales)} \]

\[ \pi_{\{\text{eid}, \text{name}, \text{salary}, \text{item}\}}, \text{SUM(\text{vol})} \text{(V \Join V')} \]

In many cases, missing attributes in \( V \) cannot be recovered by only one join operation. Let us assume that \( m = m_1 \cup \ldots \cup m_n \) is a set of attributes that should be recovered by each \( m_i \). For recovering \( m_i \), we should participate the expression for \( V \) in the join with the view that has \( m_i \) as a non-join attribute. \( V \) should also participate the join with \( V_i \). We may generate an unexpected result because \( V \) has a non-join attribute that is a recovery expression except for its participating join.

We can solve this problem by renaming the attributes of each \( V_i \) so that these have unique names in the recovery expression except for those participating in the join.

We integrate the attribute recovery mechanism with the query reformulation discussed in the previous section. Condition 3 has been modified to reflect the possibility of recovering missing attributes. The modified condition is as follows:

**Condition 3'.** Let \( \{a \leq \text{attr}(R) \text{ and mentioned in } \text{cond', group'} \cup \text{proj}(Q) \text{ or attr}(R) \cap \text{attr}(T)\} \). Then, \( \text{proj}(V) \) is empty or recoverable joining with other views \( V_1, \ldots, V_n \).

The following theorem confirms the soundness of the modified condition.

**Theorem.**
Let $Q$ and $V$ be defined as (3) and (4), respectively. If Conditions 1, 2, and 3' are satisfied, $Q$ is equivalent to the following expression.

$$Q': \pi_{\text{proj}(Q)} \ G_{\text{group}' \ \text{agg}}' \ \sigma_{\text{cond}'} \ (V \Join V_1 \Join \ldots \ Join V_n)$$  \hspace{1cm} (12)

**Proof (sketch)**

Let $W_i (1 \leq i \leq n)$ be defined as follows:

$$W_i: \pi_{\text{proj}(V)} \ \cup m_1 \cup \ldots \cup m_i, \ \text{agg}(V) \ G_{\text{group}(V)} \ \cup m_1 \cup \ldots \cup m_i, \ \text{agg}(V) \ \sigma_{\text{cond}(V)} \ \text{table}(V)$$

where $m_i \subseteq \text{attributes recoverable from } V_i$.

As shown in the proof of Lemma 1, $(V \Join V_1)$ is equivalent to $(W_1 \Join V_1)$. By utilizing this consequence as a base step, we can prove that $(V \Join V_1 \Join \ldots \ Join V_n)$ is equivalent to $(W_n \Join V_1 \Join \ldots \ Join V_n)$ by induction. (We will not show the induction step since it is simple but lengthy.) Since $V_i$ is join-useless in $(W_n \Join V_1 \Join \ldots \ Join V_n)$, $\pi_{\text{attr}}(W_n)$ is also equivalent to $W_n$. Hence, (12) is equivalent to the following expression.

$$Q': \pi_{\text{proj}(Q)} \ G_{\text{group}' \ \text{agg}}' \ \sigma_{\text{cond}'} \ (W_n \Join T)$$  \hspace{1cm} (13)

Since $W_i$ satisfies Conditions 1~3, (13) is equivalent to $Q$ by Theorem 1. \hfill \square

**Example 5.2**

Let us revisit Example 4.1. An attribute $\text{dept}$ in $V$ is essential for satisfying Condition 3 since it is used to join with $\text{Dept}$ in $Q'$. However, assume that $\text{dept}$ is missing in $V$; in which case, $V$ does not satisfy Condition 3. Fortunately, it can be recovered by joining with a base relation $\text{Emp}$ as follows:

$$\pi_{\text{eid}, \text{dept}, \text{item}, \text{brand}} \ \text{SUM( vol)} \ (V \Join \text{Emp})$$

Therefore, Condition 3 is satisfied in the formulated query $Q'$ as follows:

$$Q': \pi_{\text{eid}, \text{manager}, \text{SUM(sales_amount)} \ \text{G}_{\{\text{name}, \text{manager}\} \ \text{SUM(sales_amount)}} \ \text{sigma}_{\text{item} = \text{computer}} \ (V \Join \text{Emp} \ \Join \text{Dept})$$  \hfill \square

**DISCUSSION**

In this paper, we have extended the practical scope of utilizing materialized views. The first extension is to utilize views including relations not referred in the original query. The next extension is to attribute missing from a view under certain conditions. Note that the formulated query using views may not always be better than the original because this will depend on database properties such as indexes and data distribution. Moreover, there are more than one view that satisfies the formulation condition presented in Section 4. Therefore, in order to successfully apply our proposal to practical applications, there should be a mechanism to find the most efficient query among the equivalent ones.
There is a wide choice to compute the costs of queries, from simple to complex ones. For example, the cost of each equivalent query formulated from a given query can be computed by invoking the query optimizer repeatedly.

The techniques proposed in this paper are very useful in a variety of applications such as data warehousing and distributed information systems. Data warehouses are designed for on-line analytic processing (OLAP) where most of the queries analyze large amounts of data. Hence, data warehouses often construct summary tables to avoid accessing base relations which represent materialized aggregate views.

In distributed information systems, query results previously cached in local sites can be used to reformulate a future query in order to minimize the accessing of remote sites. These cached results can be regarded as materialized views. Moreover, even when it is difficult or even impossible to access base relations, we can utilize our proposal by replacing an original query with a new expression that consists only of materialized views.

REFERENCES


