ASSOCIATION RULES BASED QUERY EVALUATION IMPROVEMENT

X. H. TANG, G. Q. CHEN, Q. WEI

School of Economics and Management, Tsinghua University
Beijing, 100084, China

Query evaluation improvement and association rules are two interesting research topics in data query and management. In this paper, the relations are decomposed with respect to the mined association rules, and several basic data query expressions are rewritten which have relatively less time cost. This is desirable for processing queries in an efficient manner.

Keywords: Query expression; Decomposition; Association rule.

1. Introduction

Since we have stridden into a technology and information era, there is an explosion of information that is becoming more and more available to the public due to technological advances. Thus databases, especially the relational database (RDB) have become a fundamental and the most popular tool for managing and exploiting the power of information/data. What is more, data query evaluation improvement/query optimization is a large area within the database field and has gained a lot of attention over the past four decades.

There has been some efforts in enhancing queries or query expansion based on association rules (ARs). This paper will focus on the algebra-based query improvement which mainly uses the equivalent query expressions to get more efficient queries, namely, using the knowledge in ARs to improve the performance of algebraic expressions in the query evaluation. Concretely, if there is an AR in relation R, there may be a lot of tuples having the same values on the corresponding attributes. A database query against R with respect to these attributes will compare the query condition with the same values several or even hundreds of times which is a waste of effort. Therefore, the relation can be decomposed according to the ARs to avoid the repeated comparisons. Meanwhile, the query expressions are also rewritten to cope with the decomposed
relations. Finally, the relative time cost of the rewrote query and the original query are analyzed and compared.

2. Preliminaries

Suppose there is a relation \( R \) of relation schema \( R(U) \) \(^8\), \( X \) and \( Y \) are two subsets of \( U \), \( X \cap Y = \emptyset \), \( x \) and \( y \) are two values of \( X \) and \( Y \) respectively. Then an \( AR \) is an implication of the form \( x \Rightarrow y \) in \( R \). \( AR \) \( x \Rightarrow y \) is usually holds with degree of support \( D_{\text{supp}}(x \Rightarrow y) \) and degree of confidence \( D_{\text{conf}}(x \Rightarrow y) \), which are defined as follows \(^9\):

\[
D_{\text{supp}}(x \Rightarrow y) = \frac{|| x \cup y ||}{|R|}
D_{\text{conf}}(x \Rightarrow y) = \frac{|| x \cup y ||}{|| x ||}
\]

where \( || x || \) is the number of the tuples in \( R \) that have value \( x \) on attribute set \( X \), \( || x \cup y || \) is the number of the tuples in \( R \) that have value \( x \) on attribute set \( X \) and value \( y \) on attribute set \( Y \), and \( |R| \) is the number of the tuples in \( R \).

In the follow discussions, for the sake of simplicity in presentation and understanding, we stick the relation schema to \( R(A, B, C, D) \), and \( S(A, B, E) \). All the results can be easily extended to other complicated relation schemas. For a relation \( R \) of \( R(A, B, C, D) \), suppose there is an \( AR \) \( a \Rightarrow b \) with \( D_{\text{supp}}(a \Rightarrow b) = \alpha \) where \( a \) is a value of \( A \) and \( b \) is a value of \( B \).

\( ARs \) are partial knowledge about the relation \( R \), for example, \( "a \Rightarrow b \) with \( D_{\text{supp}}(a \Rightarrow b) = \alpha \)" means that part of the tuples in \( R \) have value \( a \) on attribute \( A \) and value \( b \) on attribute \( B \). Thus, we need first horizontally and vertically (HV) decompose the original relation \(^{11, 12}\), and then rewrite the query expressions based on the HV-decomposed relations. Here the horizontal decomposition in HV-decomposition is defined as: given a relation \( R \) and an association, e.g. a \( AR \) \( a \Rightarrow b \), then \( R \) can be divided into \( R' \) and \( R'' \), where \( R'' \) is the maximal subset of \( R \), \( R'' = R - R' \), and \( a \Rightarrow b \) totally holds in \( R' \) (i.e., \( D_{\text{supp}}(a \Rightarrow b) = 1 \) in \( R' \)). As \( a \Rightarrow b \) totally holds in \( R' \), \( R' \) can be further vertically decomposed as \( R'_{ab} = \Pi_{A,B}(R') = \{<a, b>\} \) and \( R'_{\text{ACD}} = \Pi_{A,C,D}(R') \). This is a lossless decomposition, i.e., \( R' = R'_{ab} \cup R'_{\text{ACD}} \). For example, suppose there is a relation \( R \) as shown in Table 1. It can be easily mined that \( AR \) \( 2_A \Rightarrow 1_B \) with \( D_{\text{supp}}(2_A \Rightarrow 1_B) = 50\% \) holds, where "2_A" indicates that "2" is a value of attribute \( A \), and "1_B" indicates that "1" is a value of attribute \( B \). Thus, the HV-decomposition of \( R \) based on \( AR \) \( 2_A \Rightarrow 1_B \) is shown in Table 2.

Table 1. Relation \( R \)
Table 2. A HV-decomposition of $R$ based on an AR

<table>
<thead>
<tr>
<th>$R^C$</th>
<th>$R'$</th>
<th>$R'_{AD}$</th>
<th>$R'_{ACD}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td></td>
<td>$A$</td>
<td>$A$</td>
</tr>
<tr>
<td>$B$</td>
<td></td>
<td>$B$</td>
<td>$B$</td>
</tr>
<tr>
<td>$C$</td>
<td></td>
<td>$C$</td>
<td>$C$</td>
</tr>
<tr>
<td>$D$</td>
<td></td>
<td>$D$</td>
<td>$D$</td>
</tr>
</tbody>
</table>

For $R$, $R'$ and $R^C$, the following two equations hold.

$$R = (R^C_A \bowtie R^C_{ACD}) \cup R^C = (\{a, b\} \bowtie R^C_{ACD}) \cup R^C \quad (1)$$

$$\Pi_{A,B}(R) = R^C_A \cup \Pi_{A,B}(R^C) = (\{a, b\} \cup \Pi_{A,B}(R^C) \quad (2)$$

Equation (1) and (2) can help reduce the query time. Some efforts have been made in this direction from the viewpoint of another kind of association, approximate functional dependencies (where only the first half of the two equations holds) \(^1\). So the queries “Select Distinct $R1.A, R1.B$ From $R$ as $R1$ Where $R1.A =$ constant” and “Select Distinct $R1.A, R1.B, S.D$ From $R$ as $R1, S$ Where $R1.B = S.B$” can be rewritten with respect to the first half of Equation (1) and (2). And the author also used some artificial data verified the hypotheses that the rewritten queries performed better than the original queries in some cases. Compared with approximate functional dependencies, $AR$ has its advantages in reducing the query time, as its degree of support has shown how many repeated values are there in the relation.

3. Query Improvement based on ARs

Suppose there is an AR $a \Rightarrow b$ with $Dsupp(a \Rightarrow b) = \alpha$ in relation $R$ of $R(A, B, C, D)$, a relation $T$ of $R(A, B, C, D)$ and a relation $S$ of $S(A, B, E)$. Then, consider the following three queries.

$Q_1$ SELECT DISTINCT $R_{i}.A, R_{i}.B$ FROM $R$ as $R_1$
WHERE $R_{i}.A =$ constant

$Q_2$ SELECT DISTINCT $R_{i}.A, R_{i}.B, S.E$ FROM $R$ as $R_i, S$
WHERE $R_{i}.A =$ $S.A, R_{i}.B =$ $S.B$
\[Q_3\] (SELECT DISTINCT \(R_1.A, R_1.B\) FROM \(R\) as \(R_1\),

EXCEPT

(SELECT DISTINCT \(R_2.A, R_2.B\) FROM \(T\) as \(R_2\))

Translate \(Q_1\) into algebraic expressions, it is equal to \(\Pi_{A,B}(\sigma_{A=constant}(R))\), and according to the transformation rules and Equation (2),

\[
\Pi_{A,B}(\sigma_{A=constant}(R)) = \sigma_{A=constant}(\Pi_{A,B}(R))
\]

\[
= \sigma_{A=constant}(R_A^1 \cup \Pi_{A,B}(R^C))
\]

Thus, \(Q_1\) is equal to \(Q'_1\) as follows.

\[Q'_1\] SELECT DISTINCT * 
FROM \(\{<a, b>\}\) UNION (SELECT DISTINCT \(A, B\) FROM \(R^C\)) as \(R_1\)
WHERE \(R_1.A = constant\)

In the same way, the corresponding algebraic expressions of \(Q_2\) and \(Q_3\) are as Equation (4) and (5).

\[
\Pi_{A,B,E}(R \bowtie S) = (\Pi_{A,B}(R)) \bowtie S = (\{<a,b>\} \cup \Pi_{A,B}(R^C)) \bowtie S \\
\Pi_{A,B}(R - T) = \Pi_{A,B}(R) - \Pi_{A,B}(T) = (\{<a,b>\} \cup \Pi_{A,B}(R^C)) - \Pi_{A,B}(T)
\]

\(Q_2\) and \(Q_3\) can be rewritten as \(Q'_2\) and \(Q'_3\) respectively.

\[Q'_2\] SELECT DISTINCT \(R_1.A, R_1.B, S.E\)
FROM \(\{<a, b>\}\) UNION (SELECT DISTINCT \(A, B\) FROM \(R^C\)) as \(R_1, S\)
WHERE \(R_1.A = S.A, R_1.B=S.B\)

\[Q'_3\] (SELECT DISTINCT * 
FROM \(\{<a,b>\}\) UNION (SELECT DISTINCT \(A, B\) FROM \(R^C\)) as \(R_1\)
EXCEPT
(SELECT DISTINCT \(R_2.A, R_2.B\) FROM \(T\) as \(R_2\))

Now, consider the time cost of each query. It is worth noting that, in general, given a high level language query, for example, a SQL query, it will be firstly been parsed into the basic algebraic expressions, and then presented to the DBMS's query optimizer which can examine and choose the optimal plan for the query. The time cost of executing a query includes the following components: access cost to secondary storage, storage cost, computation cost, memory usage cost and communication cost. Computation cost is the cost of performing in-memory operations on the data buffers during query execution.
Such operations include searching for and sorting records, merging records for a join, and performing computations on attribute values. For smaller databases, the emphasis is on minimizing computation cost\(^{10}\). Thus, in our discussions, the query time of \(Q\) refers to computation cost denoted as \(\text{time}(Q)\). Specifically, we use the comparison frequency of domain values to represent the relative query time. For example, consider the query “\(Q_4\): SELECT DISTINCT * FROM \(R\) WHERE \(R.A = 2\)” against relation \(R\) as shown in Table 1, then, the query time \(\text{time}(Q_4) = 4\). And the query time of “\(Q_5\): SELECT DISTINCT * FROM \(R\) WHERE \(R.A = 1, R.B = 3\)” is \(\text{time}(Q_5) = 8\).

Thus, the relative query time of \(Q_1\), \(Q_2\), \(Q_3\) and the original queries are as follows, where \(|R|\) is the number of the tuples in \(R\), \(|R^{ec}| = (1 - \alpha) |R|\).

\[
\frac{\text{time}(Q'_1)}{\text{time}(Q_1)} = \frac{1 + |\Pi_{AB}(R^{ec})|}{|R|} \leq \frac{1 + |R^{ec}|}{|R|} = \frac{1 + (1 - \alpha)|R|}{|R|} \approx 1 - \alpha \quad (6)
\]

\[
\frac{\text{time}(Q'_2)}{\text{time}(Q_2)} = \frac{2 \times (1 + |\Pi_{AB}(R^{ec})|) \times |S|}{2 \times |R| \times |S|} = \frac{1 + |\Pi_{AB}(R^{ec})|}{|R|} \approx 1 - \alpha \quad (7)
\]

\[
\frac{\text{time}(Q'_3)}{\text{time}(Q_3)} = \frac{2 \times (1 + |\Pi_{AB}(R^{ec})|) \times |\Pi_{A,B}(T)|}{4 \times |R| \times |T|} = \frac{(1 + |\Pi_{AB}(R^{ec})|) \times |\Pi_{A,B}(T)|}{2 \times |R| \times |T|} \approx (1 - \alpha)/2 \quad (8)
\]

Equation (6), (7) and (8) reflect that the rewrote queries has less query times. For example, if the \(AR a \Rightarrow b\) has a support degree of \(D_{supp}(a \Rightarrow b) = 40\%\), then \(\text{time}(Q'_1) / \text{time}(Q_1) \leq 1 - 40\% = 60\%\) and \(\text{time}(Q'_3) / \text{time}(Q_3) \leq (1 - 40\%) / 2 = 30\%\).

4. Conclusions

This paper has concentrated on the problem of utilizing the knowledge in the association rules to enhance query evaluations. Based on a given association rule, the relation was decomposed while the basic query expressions were rewrote. It has been deducted that the rewrote query expressions can result relatively less time cost. The proposed improvement enables the data queries more efficient, it is especially useful in the cases where large amount of queries are executed against the same databases.
Acknowledgments

The work was partly supported by the National Natural Science Foundation of China (70231010/70621061) and the Research Center for Contemporary Management, Tsinghua University.

References