A Method for Processing Boolean Queries using a Result Cache

Jae-heon Cheong\textsuperscript{1}, Sang-goo Lee\textsuperscript{1}, and Jonghoon Chun\textsuperscript{2}

\textsuperscript{1} School of Computer Science and Engineering
Seoul National University, Korea
\{cjh,sglee\}@cygnus.snu.ac.kr
\textsuperscript{2} Division of Computer Science and Engineering
Myongji University, Korea
jchun@jju.ac.kr

Abstract. We propose a new method for processing Boolean queries with a collection of previously answered query results which we called a result cache. We present algorithms to effectively recognize the portions of a given query that can be answered from the result cache and from those should be retrieved from the sources distributed over the network. We allow a semantic decomposition of a representation for an efficient manipulation of it.

1 Introduction

The result cache is a collection of documents that are answers to previously processed queries. It is based on a semantic caching mechanism and is managed as a collection of semantic regions. Each semantic region is represented by a Boolean formula and contains the answer documents that satisfy the formula. When a query is issued, it is decomposed into two sub-queries; a hit query that retrieves cached results from the result cache, and a miss query that fetches non-cached results. The final result is obtained by integrating the results of both the hit query and the miss query.

In this paper, we propose a new representation method for the result cache and utilize that representation to newly propose an efficient processing algorithm for general Boolean queries. Query processing in a mediator makes use of the semantic representation of the result cache to determine which results are locally available in the result cache and which results are needed from information sources. As user queries are submitted, the more complex the representation of the result cache gets. Thus we present a mechanism to reduce the complexity of the representation by allowing multiple representations for the result cache.

The rest of the paper is organized as follows. Section 2 introduces related work; sections 3 and 4 present a new method of processing Boolean queries with a result cache; we present an accommodated replacement strategy in section 5; section 6 describes our experimental results; and finally, section 7 concludes the paper.
2 Related Work

There are a number of articles on semantic data caching [3][4][5][6]. However, most of them do not consider general Boolean queries but conjunctive ones.

[3] proposes a query optimization method that uses cached queries in a mediator system named HERMES. This system deals with only simple conditions but not with general Boolean expressions. [4] deals with only conjunctive queries also. [5] focuses on the cache replacement strategies based on recency and semantic distance of cached queries.

[6] employs a signature file to represent a result cache. Since a signature file is based on a conjunctive query model, it cannot be directly used for a general Boolean query model. In addition, a signature file cannot completely recognize which portion of the cached results can be used for the given query due to the false drop problem.

In [2], a query-based virtual index (QVI) is proposed. It is based on the fact that most user queries are redundant. However, it does not consider general Boolean queries with multiple query terms and can utilize previous query results only when the current query is exactly matched with at most one among the previous queries. Our method proposed in this paper can process multiple-query-term queries and outperforms the exact matching method as described in section 6.

3 Result Cache Management

We assume that a query is defined as a Boolean formula with atoms connected via three Boolean operators such as AND, OR, and NOT. For notational convenience, we use the symbol '∧' to denote the Boolean operator 'AND', '∨' to denote the Boolean operator 'OR', and '¬' to denote the Boolean operator 'NOT'.

Definition 1 (Query Result). For a given query \(Q\), \([Q]\) is the resulting documents for \(Q\).

Definition 2 (Sub Query). For two queries \(Q_1\) and \(Q_2\), \(Q_1\) is a sub query of \(Q_2\) if \([Q_1] \subseteq [Q_2]\).

3.1 Representation of a Result Cache

For queries \(Q_1, Q_2, \ldots,\) and \(Q_n\), the result cache denoted by \(C\) is defined as \([Q_1] \cup [Q_2] \cup \ldots \cup [Q_n]\). The result cache is described by a disjunction of Boolean formulas of previously processed queries. In order to formally represent a result cache, we make use of semantic regions. A semantic region groups together semantically related documents. It is represented by a Boolean formula that is qualified for by result documents within the region. The Boolean formula that describes a semantic region is called its region descriptor[6][5].
Definition 3 (Representation of a Result Cache). A result cache consists of one or more semantic regions. A disjunction of their region descriptors is called a representation of the result cache.

Given a user-submitted query \(Q\), the mediator decomposes it into two sub queries, a hit query and a miss query. The hit query, denoted by \(H(Q, C)\), is a sub query of \(Q\) that describes sub results available from the result cache \(C\). This can be represented by \((Q \land C)\). The miss query, denoted by \(M(Q, C)\), is another sub query of \(Q\) that describes the sub results that should be retrieved from information sources other than the result cache \(C\). This can be represented by \((Q \land \neg C)\).

Lemma 1. Suppose \(R\) is a representation of the current result cache. Then, the following equalities are always true.

\[
\begin{align*}
H(a \land b, R) &= H(a, R) \land H(b, R) \\
M(a \land b, R) &= M(a, R) \land M(b, R) \\
H(a \lor b, R) &= H(a, R) \lor H(b, R) \\
M(a \lor b, R) &= M(a, R) \lor M(b, R)
\end{align*}
\]

Proof (of the first equality). By the definition of a hit query, \(H(a \land b, R) = (a \land b) \land R\). Subsequently, \((a \land b) \land R = (a \land R) \land (b \land R) = H(a, R) \land H(b, R)\)

Q.E.D.

3.2 Atomic Representation

An atomic representation represents a result cache \(C\) as a union of atomic regions. An atomic region is expressed as a minterm [7] that is defined as a conjunction of literals in which every atomic term occurs exactly once, either in its positive or negative form. Accordingly, every two atomic regions are pair-wise disjoint with each other.

Definition 4 (Atomic Representation\(^1\)). Let a representation denoted by \(R\) of the current result cache \(C\) is \(S_1 \lor S_2 \lor \ldots \lor S_n\) where \(S_i\) is a region descriptor of a semantic region \([S_i]\). If every two semantic regions \([S_i]\) and \([S_j]\) \((i \neq j\) and \(i \leq i, j \leq n\)) are pair wise disjoint with each other, \(R\) is an atomic representation of \(C\).

Any Boolean expression can be equivalently transformed to a disjunction of minterms. For example, a Boolean formula \(a \lor b\) is equivalent to \((a \land \neg b) \lor (a \land b) \lor (\neg a \land b)\) where \(a\) is replaced with a disjunction of two minterms \((a \land \neg b)\) and \((a \land b)\) and \(b\) with \((\neg a \land b)\) and \((a \land b)\). Accordingly, a result cache can be represented by a disjunction of minterms. In this case, each minterm plays the

\(^1\) Throughout, the term 'representation' refers to an atomic representation
role of a region descriptor. In other words, $[m_i]$ is a semantic region described by a region descriptor $m_i$ where $m_i$ is a minterm.

Semantic regions of the result cache are described by corresponding minterms defined by query terms. When one or more new query terms that do not already occur in the representation of the current result cache are introduced by a given query $Q$, all semantic regions in the representation should be reorganized by splitting. For example, suppose $[S] = [a \land b]$ is a semantic region of the current representation of the result cache $C$, and a query term $c$ is now introduced by a user query. Since $c$ does not occur in $S$, $[S]$ is split into two new regions $[S_1] = [a \land b \land c]$ and $[S_2] = [a \land b \land \neg c]$, where every document that contains a query term $c$ goes to $[S_1]$ and others to $[S_2]$. Consequently, the original semantic region $[S]$ is replaced with $[S_1] \cup [S_2]$. The region descriptor $S$ now becomes as $S_1 \lor S_2$.

### 3.3 Query Processing

In order to process a given query $Q$ with a result cache $C$, the query should be rewritten using minterms defined by the set of query terms in $Q$ and $C$. Once $Q$ and $C$ are represented by minterms, we can get $Q \land C$ and $Q \land \neg C$ simply by comparing minterms in them $[8]$.

Let $R$ be the representation of the current result cache $C$. When a new query $Q$ is submitted, it is processed as follows:

- **Step 1**: If all of the query terms in $Q$ are already in $R$, go to step 2. If $Q$ contains at least one query term that is not in $R$, every semantic region of $R$ is split by the new query terms introduced by $Q$.
- **Step 2**: $Q$ is rewritten into a disjunction of the minterms defined by the set of query terms in $C$ and $Q$, and $C$.
- **Step 3**: Decompose $Q$ into $H(Q, C)$ and $M(Q, C)$. Since $Q$ and $C$ are represented as disjunctions of minterms, $H(Q, C)$ and $M(Q, C)$ can be obtained by simply comparing the minterms of $Q$ and $C$. $H(Q, C)$ is equivalent to a disjunction of common minterms of $Q$ and $C$ while $M(Q, C)$ is equivalent to the set of minterms that occurs in $Q$, but not in $C$.
- **Step 4**: Retrieve the results of $H(Q, C)$ and $M(Q, C)$ from the cache and information sources. New semantic regions described by $M(Q, C)$ are added to the result cache.

### 4 Decomposed Representation

Since every occurrence of a new query term causes the current semantic regions to be doubled, the number of the semantic regions increases exponentially compared to the growth of new query terms. To address this issue, we make it possible for a result cache to have multiple sub-representations by semantically decomposing the original representation.
4.1 Semantic Decomposition

Definition 5 (Semantic Decomposition). Let $R$ be a representation of the result cache. $R^d = \{R_1, R_2, ..., R_m\}$ is a semantic decomposition of $R$ if the following three conditions are satisfied.

$$
[R] \supseteq [R_i],
$$

$$
[R] = [R_1] \cup [R_2] \cup ... \cup [R_m], \text{ and}
$$

$$
|R| > |R_i|
$$

where $i \leq m$ and $|R|$ is the number of unit terms in $R$.

When $R^d = \{R_1, R_2, ..., R_m\}$ is a semantic decomposition of $R$, $R_i$ is called a sub-representation of $R$.

Lemma 2. Suppose $R^d = \{R_1, R_2, ..., R_m\}$ is a semantic decomposition of a representation $R$. The following equalities are always true.

$$
H(Q, R) = H(Q, R_1) \lor H(Q, R_2) \lor ... \lor H(Q, R_m)
$$

$$
M(Q, R) = M(Q, R_1) \land M(Q, R_2) \land ... \land M(Q, R_m)
$$

Proof. By definition of semantic decomposition,

$$
R = R_1 \lor R_2 \lor ... \lor R_m
$$

Then,

$$
H(Q, R) = H(Q, R_1 \lor R_2 \lor ... \lor R_m)
$$

According to the definition of a hit query,

$$
H(Q, R_1 \lor R_2 \lor ... \lor R_m) = Q \land (R_1 \lor R_2 \lor ... \lor R_m)
$$

$$
= (Q \land R_1) \lor (Q \land R_2) \lor ... \lor (Q \land R_m)
$$

$$
= H(Q, R_1) \lor H(Q, R_2) \lor ... \lor H(Q, R_m)
$$

Similarly,

$$
M(Q, R_1 \lor R_2 \lor ... \lor R_m) = Q \land \neg(R_1 \lor R_2 \lor ... \lor R_m)
$$

$$
= (Q \land \neg R_1) \land (Q \land \neg R_2) \land ... \land (Q \land \neg R_m)
$$

$$
= M(Q, R_1) \land M(Q, R_2) \land ... \land M(Q, R_m)
$$

Q.E.D.

If we employ a semantic decomposition, we can bound the number of semantic regions for one sub-representation to a certain value by limiting its number of query terms. Formally, the maximum number of minterms of a semantic decomposition $R^d = \{R_1, R_2, ..., R_m\}$ is $\sum_{i=1}^{m} 2^{|R_i|}$. Since the same query term can occur in one or more sub-representations, $\sum_{i=1}^{m} 2^{|R_i|} \leq 2^{|R|}$. For example, suppose the number of keywords is 20. If the result cache is described with one representation, $2^{20} = 1,048,576$ is the maximal number of minterms to be used. However, suppose there exist two sub-representations $R_1, R_2$ and $|R_1| = 10$ and $|R_2| = 11$, then $2^{10} + 2^{11} = 3072$ is the maximal number of minterms.
4.2 Closed Decomposition

According to lemma 2, in order to get a hit query or a miss query, every decomposed sub-representation must be examined. The number of sub-representations to be examined in computing a hit query and a miss query can be reduced by utilizing the notion of closed decomposition.

Definition 6 (Closed Decomposition). Let \( Q \) be a given query whose query terms are \( q_1, q_2, \ldots, q_n \) and \( R^d = \{ R_1, R_2, \ldots, R_m \} \) be a semantic decomposition for the representation \( R \). \( R^d \) is a closed decomposition for \( Q \) if there exists at least one sub-representation \( R_i (1 \leq i \leq m) \) for every \( q_i (1 \leq j \leq n) \) such that

1. \( [H(q_j, R)] = [H(q_j, R_i)] \) and
2. \( [M(q_j, R)] = [M(q_j, R_i)] \)

where \( [H(q_j, R_i)] \neq \emptyset \) and \( [M(q_j, R_i)] \neq \emptyset \). When \( R_i \) does not contain \( q_j \), \( R_i \) should be split by means of \( q_j \) to get \([H(q_j, R_i)]\) and \([M(q_j, R_i)]\).

If \( R^d \) is a closed decomposition of \( R \), a hit query and a miss query for a given query \( Q \) can be obtained by considering sub-representation as many as query terms in \( Q \).

Definition 7 (Cascading Decomposition). \( R^d = \{ R_1, R_2, R_3 \} \) is a cascading decomposition of \( R \) if it is a semantic decomposition of \( R \) and \([R_1] \subset [R_2] \subset \ldots \subset [R_m] = [R]\)

In a cascading decomposition, every two representations \( R_i \) and \( R_{i+1} \) should satisfy \([R_i] \subset [R_{i+1}]\). A simple way to make this possible is to add every semantic regions of \( R_i \) to \( R_{i+1} \). Unfortunately, the resulting decomposition cannot be a semantic one, because of the third condition of definition 3.

In order to solve this problem, we extend the notion of atomic term in constructing a minterm. Without loss of generality, we can treat an arbitrary Boolean expression \( E \) as a unit term as long as the atoms occurring in \( E \) does not occur any where else in the expression. For example, \((a \land b)\) can be treated as a unit term in constructing minterms such as \((a \land b) \land c, (a \land b) \land \neg c, \neg (a \land b) \land c, \) and \(\neg (a \land b) \land \neg c\). So, in constructing \( R_{i+1} \), we can treat \( R_i \) as a unit term and not allow query terms in \( R_i \) to appear anywhere else in \( R_{i+1} \). Then, the resulting decomposition becomes a semantic decomposition of \( R \). Although we do not include the proof here, we can always obtain such a cascading decomposition for \( R \) and we can get results from \( R_{i+1} \) for query terms in \( R_i \) since \([R_i] \subset [R_{i+1}]\).

Theorem 1. If \( R^d \) is a cascading decomposition of \( R \), then it is a closed decomposition for general Boolean queries with arbitrary number of query terms.

Proof. Let \( R^d \) be a semantic decomposition of \( R \) and \( \{ R_1, R_2, \ldots, R_m \} \)

i: \( Q \) contains only single query term.

Since \( R^d \) is a cascading decomposition of \( R \), \([R_m] = [R]\). Then, the following two equalities are satisfied:

\([H(Q, R)] = [H(Q, R_m)]\)
\[ M(Q, R) = [M(Q, R_m)] \]
As a result, \( R^d \) is a closed decomposition for \( Q \).

ii: \( Q \) contains multiple query terms with AND/OR operators.

We prove this by showing the case with queries with two terms connected by AND/OR operators. Suppose \( Q \) contains two query terms \( a \) and \( b \) connected by AND. According to the lemma 1, the following two equalities are true.

\[
H(a \land b, R) = H(a, R) \land H(b, R) \\
M(a \land b, R) = M(a, R) \land M(b, R)
\]

Since \( R^d \) is a cascading decomposition, \([H(a, R) = [H(a, R_m)]\) and \([H(a, R) = [H(a, R_m)]\). At the same time, \([H(b, R)] = [H(b, R_m)]\) and \([M(b, R)] = [M(b, R_m)]\). Therefore, \( R^d \) is a closed decomposition for \( a \land b \). Similarly for \( Q = a \land b \), \( R^d \) is a closed decomposition.

Thus for an arbitrary Boolean query \( Q \), \( R^d \) id a closed decomposition. Q.E.D.

5 Replacement Strategy

In this section, we propose a new replacement strategy that is based on the standard LRU strategy. For the sake of simplicity, we only consider a single representation but not a semantic decomposition. We employ two factors in calculating the replacement value of a semantic region. One is recency of usage value and the other is semantic significance value. Recency of usage value indicates how recently a semantic region was used while semantic significance value describes the potential usefulness of a semantic region.

5.1 Recency of Usage

Firstly, the most recent value denoted by \( U_{\text{most}} \) is initialized to 1. Each time a query is submitted, \( U_{\text{most}} \) is incremented by 1 and assigned to all semantic regions induced by the given query as their recency of usage values. If \( U_i \) is greater than \( U_j \), it means that a semantic region \( S_i \) was more recently used than \( S_j \) was.

5.2 Semantic Significance

We assume that every query term has the same probability to occur in user’s queries and most users might submit queries only with non-negated query terms. Then, we can reasonably claim that one semantic region denoted by \( S_1 \) might have more possibility to be used in answering user’s queries than the other region denoted by \( S_2 \) does when the number of non-negated query terms in \( S_1 \) is larger than that in \( S_2 \). The number of non-negated terms in a semantic region \( S_i \) is defined as its semantic significance. It is denoted by \( M_i \).
5.3 Replacement Value

The result cache replaces a semantic region whose replacement value is the least among others. The replacement value of a region is calculated by a pre-defined weighted function. If the current replacement value of region $S_i$ is $REP_i$, we calculate a new replacement function as

$$ REP'_i = REP_i + m \cdot M_i + u \cdot U_i $$

(1)

where $0 \leq m, u \leq 1$. If $m$ is set to 0 and $u$ to 1, $REP_i$ equivalently simulates the standard LRU strategy.

Since every semantic region varies in its size, we have to consider it in order to determine the replacement value. We employ a size-adjusted LRU strategy, SLRU, mentioned in [9] and choose larger semantic regions as victims to make room for multiple small regions. If a semantic region has replacement value $v$ and size $s$, the ratio $v/s$ gives the semantic region’s size-adjusted replacement value. Therefore, modified replacement function is as follows:

$$ REP'_i = \frac{1}{N_i} \cdot (REP_i + m \cdot M_i + u \cdot U_i) $$

(2)

where $N_i$ is the number of documents in a semantic region $S_i$.

6 Experiment

6.1 Workload

In our experiments, we used the log of real user queries submitted to the Web search engine at Korea, Naver (http://www.naver.com). Although this search engine is purposed to search and retrieve general Web pages and most queries have a wide variety, query terms used in this workload are characterized by a high duplication and frequency weight: 30.2% of total query terms are repeatedly used in user’s queries and 0.05% of the most frequent query terms occur in 50.4% of the queries. As a whole, 41,000 query terms are used in 1,000 queries.

6.2 Hit Ratio

The size of the result cache varies in range $[20 \cdot S_Q : 200 \cdot S_Q]$ where $S_Q$ is the average size of query answers. The hit ratio is the average size of a query result available from the result cache. For a given query, the hit ratio is calculated as follows:

$$ \frac{N_C}{N_T} $$

(3)

where $N_C$ is the number of results available from the result cache and $N_T$ is total number of results. We examined four replacement strategies such as standard LRU ($m = 0, u = 1$), semantic LRU ($m = 1, u = 0$) that considers semantic
significance only, semantic-recency LRU \((m = 1, u = 1)\) that considers both semantic significance and recency of usage, and size-adjusted LRU \((\text{SLRU} \ m = 1, u = 1)\). Fig. 1 unveils that semantic significance contributes sufficiently more to the hit ratio than recency of usage factor does.

### 6.3 Performance Comparison

We examine the efficiency of the proposed caching method comparing with two different methods. One is a naive query processing method that does not utilize any cached results (we call it 'NC') and the other uses cached results only when the given query is exactly matched with one among the previously submitted (we call it 'EM').
Every query set consists of 50 distinct queries. On the average, the proposed method (we call it 'DR') improved the retrieval performance by almost 50\% as much in the case with no cache as depicted in Fig. 2. Furthermore, DR outperforms EM. The reason is that the EM uses the cached results only when \( H(Q, C) \) is equivalent to \( Q \), while DR can utilize them only if \( H(Q, C) \) is not empty although \( H(Q, C) \) is not equivalent to \( Q \). In other words, DR always guarantees higher hit ratio than EM does.

7 Conclusion

In order to improve the query processing performance of mediator systems, it is essential to efficiently recognize the part of a query that can be directly retrieved from the cache. The caching methodology that has been proposed proved to be efficient for this purpose. In addition, the proposed approach is able to be applied to process general Boolean queries while previous methods were somewhat restricted in this aspect. The proposed approach employs an atomic representation in describing the result cache, which can be effectively managed and utilized in processing general Boolean queries. Furthermore, its representation is managed in semantically decomposed form so that the total number of semantic regions can be reduced only with a negligible slight performance degradation.

References