4

Semantic Cache Reasoners

Muhammad Azeem Abbas, Muhammad Abdul Qadir
and Muhammad Tanvir Afzal
Centre for Distributed and Semantic Computing,
Faculty of Engineering and Applied Sciences,
Mohammad Ali Jinnah University Islamabad,
Pakistan

1. Introduction

Semantic caching (Ren, Q et al., 2003),(Dar et al., 1996) is said to be a technique for storing data and their corresponding semantic descriptions. Concept of semantic cache itself is quite simple but the reasoning required to evaluate any query over a semantic cache can be very complex (Godfrey P. and Gryz J., 1997). The reasoning over stored semantics is a determination process to know how query and cache formulas are related semantically. This reasoning is termed as semantic cache query processing (Ren, Q et al., 2003),(Dar et al., 1996). In this chapter we demonstrate several semantic cache query processing techniques for relational queries, web queries, xml queries, answering queries form materialized views and logic based subsumption analysis queries.

Mainly there are two types of semantic query processing approaches, structured-semantics and unstructured-semantics. In structured-semantics original problem or query is represented in a structure that has the ability to contain semantics along with its structure. Examples of structured-semantics are ontology, resource description framework (RDF) and extensible markup language (XML) etc. Unstructured-semantics approaches perform reasoning for semantic extraction from structures that do not posses semantics in their representations. Semantic cache query processing is an example of unstructured-semantics reasoning. Since standard query language (SQL) is structured but it do not contains semantics of data to be answered against a query and query itself.

In this chapter we demonstrate several semantic cache reasoners for unstructured-semantics. All of these semantic cache reasoning techniques represent query language to a mediate structured-semantic representation for semantic extraction.

2. Semantic cache query processing

In general research a semantic cache system can be grouped into two parts i) cache management and ii) query processing. Strategies for data management, replacing, coalescing, and indexing results of previously evaluated queries are mainly the part of semantic cache management. Query processing involves techniques that compute available and unavailable data from a semantic cache by performing some sort of reasoning over

semantic descriptions. Also query processing technique handles local query execution, retrieval of unavailable data from a remote server and formulation of the end results. In this chapter we focus on semantic cache query processing.

At finer granularity semantic cache is a collection of semantic regions or semantic segments. Associated semantics for a cached query, which is a query specification (Lee et al., 1999) are stored in semantic cache along with resultant data is called a semantic region (Dar et al., 1996) or semantic segment (Ren et al., 2003).

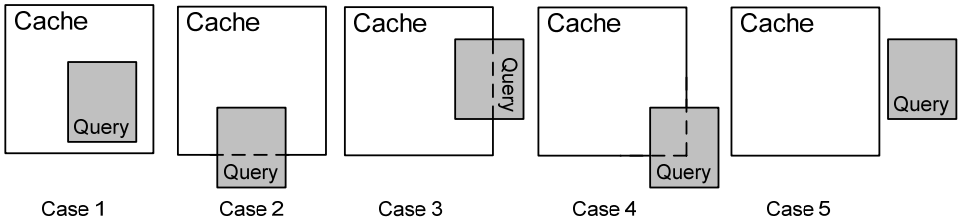


Fig. 1. Relationship between Cached Query (Q_C) and User Query (Q_U).

Formal definition of semantic segment can be seen in (Ren et al., 2003). A query processing technique can perform reasoning over semantic segments to determine whether cached data fully or partially or do not contributes to an incoming query. If the incoming query is fully answerable from a semantic cache, then no communication with the server is required. Similarly a partial answer to a query will reduce the amount of data retrieved from the server.

In case of a partial answer, the user query is trimmed into two disjoint sub queries (Keller A.M. and Basu J., 1996): the query executed locally called Probe Query (ProbQ) and the query sent to the server named Remainder Query (RemQ) (Dar et al., 1996). The previous literature (Ren, Q et al., 2003),(Dar et al., 1996),(Lee et al., 1999),(Godfrey P. and Gryz J., 1997),(Keller A.M. and Basu J., 1996) shows that this trimming is performed on the basis of relationship between the content of a semantic segment and the result required by an incoming query. Possible cases of the relationship between the incoming query and the semantics stored in the cache (as reported in the literature) is shown in Figure 1. White boxes represent previously stored query results and gray boxes shows incoming user queries. In Figure 1 rows (tuples) are represented horizontally and columns (attributes) are vertically and only select-project queries are considered. In each case a user query overlaps semantic cache region in a certain way. Case 2 depicts a horizontal partition in which some part of the incoming query tuples satisfied by cache semantics. Where in case 3, a projection of the query is available in cache and some attributes are missing, this situation is called a vertical partition. This figure represents that a partial answer is possible in case 2,3 and 4, where a user query can be fully answered from the cache in case 1. This figure is used to evaluate a semantic cache query processing scheme, too, i.e. whether a scheme incorporates all the cases or not. We argue that due to this misleading diagram, the missing implicit semantics are not being considered in the previous query processing techniques. Therefore, in this thesis we have adopted a new way of comparing the semantics of a user query and the cache semantics in the coming sections.

2.1 Semantic cache query processing criteria

Previous surveys (Bashir M. F. and Qadir M. A., 2006a), (Ahmad, M and Qadir, M.A., 2008), (Jónsson B. Þór et al., 2006), (Hao X et al., 2005), (Halevy, A.Y., 2001), (Makki K. S and Andrei S, 2009) conducted over semantic cache query processing identified two main parameters for evaluation i.e. Maximum Data Retrieval (MDR) and fast query processing. Quantification of the MDR was not given in those surveys. Here we quantify it with the test, data from server ($\mathbf{D_s}$) intersection data from cache ($\mathbf{D_c}$) should be empty set i.e. $\mathbf{D_s \cap D_c = \Phi}$. In general any technique which retrieves maximum possible or complete results from local cache in tractable time with this given quantification is said to be an efficient semantic cache query processing technique.

2.2 Query

A select-project query is a tuple < Q_{UA}, Q_{UR}, Q_{UP},Q_{UD}>, where Q_{UA} is *Select Clause* of query which contains projected attributes. Q_{UR} is the *From Clause* which contains relation of a database D, from which data is to be retrieved. Q_{UP} is *Where Clause* which contains conjunctive or disjunctive compare predicates, a compare predicate is of the form $P = a\ op\ c$, where $a \in A$ *{Attributes Set}*, $op \in \{<, \leq, >, \geq, =\}$, c is a constant in a specific domain (Ren et al., 2003), Q_{UD} is the *resultant data* of this query. A query can be represented as $\boldsymbol{\pi}_{QUA}\ \boldsymbol{\sigma}_{QUP}\ (\boldsymbol{Q_{UR}})$ in relational algebra.

2.3 Amending query

A query that only request a key attribute of a relation from a remote server to extract known available data from cache is called an amending query. When we know that some data is available in semantic cache but could not extract it precisely. Than we request the server for a key attribute for a user query and extract cached attributes (data) against those keys from cache. Requesting only keys require less computation on database server and low bandwidth over network, in general.

Consider the following employee database information provided in example 1 below, which shall be used throughout evaluation in this chapters. The semantic cache model we follow is similar to the relational database model. The basic building blocks of the relational model are attributes (columns), rows (tuple), tables (relations) and relation schema. The schema defines the relations and the attributes with their data type in each relation. A row or a tuple is a set of attribute's instances.

2.4 Example 1

Consider an employee database with a relation name *Emp (Empid, Ename,Department, Age, Salary,Exp)*. The domain of the *Age, Salary*, *Department* and *Exp* attributes of *Emp* are {20,…,100},{0.1K,…,1K,…15K},{CS, EE, BI, BA},{1,..,50} respectively as shown in Figure 2. Also suppose that a cache already has following cached queries shown in Figure 3.

3. Query processing techniques

Work on query processing over semantic cache is mainly classified in query intersection (Lee et al., 1999), query trimming (Keller A.M. and Basu J., 1996),(Ren, Q et al., 2003), answering queries using views (Levy A.Y et al., 1996), (Duschka O.M. and Genesereth M.R.

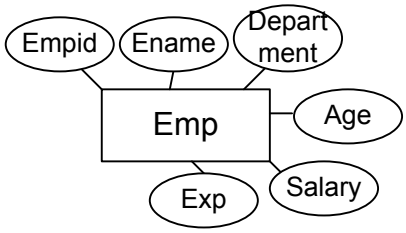


Fig. 2. Employee relation.

$$Q_{C1} \leftarrow \pi_{Ename,\ Department}\ \sigma_{Age \geq 50}\ (Emp);$$
$$Q_{C2} \leftarrow \pi_{Age,\ Department}\ (Emp);$$
$$Q_{C3} \leftarrow \pi_{Ename,\ Department}\ \sigma_{Salary>10k}\ (Emp);$$
$$Q_{C4} \leftarrow \pi_{Age,\ Salary}\ \sigma_{Salary \leq 30k}\ (Emp);$$
$$Q_{C5} \leftarrow \pi_{Ename,\ Salary,\ Exp}\ (Emp);$$
$$Q_{C6} \leftarrow \pi_{Age}\ \sigma_{Age < 70}\ (Emp);$$
$$Q_{C7} \leftarrow \pi_{Age,Salary,Exp}\ \sigma_{Salary \geq 1K\ \wedge\ Salary <= 40K}$$

Fig. 3. Cached Queries.

1997), (Pottinger R. and Levy A. 2000), semantic cache for web queries (Chidlovskii B and Borghoff U. M., 2000), (Qiong L and Jaffrey F. N., 2001), xml based semantic cache (Mandhani B. and Suciu D., 2005), (Sanaullah, M., 2008) and description logic based subsumption analysis in semantic cache (Baader et al., 1991a) (Hollunder et al., 1990) techniques.

3.1 Query intersection

Query processing for the five scenarios similar to Figure 1 and one additional scenario which shows cache as a subset of incoming queries (reverse of case 1 of Figure 1) was presented by Lee (1999, pp.28-36). Against each scenario probe and remainder query were computed based on cache and query intersection or difference. Intersection and difference of cache semantics and a posed query were mentioned at a very abstract level.

Definition of intersection (Lee et al., 1999) between semantics of cache region Q_C and a user query Q_U on relation R is shown in statement (i) of Figure 4. This intersection consists of two parts. One is the common projected attributes while the other is combined condition of a user and cached query predicates (Shown in statement (ii) of Figure 4). A query or cache semantics are represented as a triple $< \pi_Q, \sigma_Q, operand_Q >$. π_Q is the projected attributes, $operand_Q$ is the base relation. Where any predicate condition (σ_Q) is represented as a value domain list $\{d_{Q,1}, d_{Q,2}, ..., d_{Q,n}\}$ and a condition is satisfiable if none of the value domain is null. We elaborate this concept with an example.

Consider the database schema information provided in example 1 above. A user query Q_U over cached query Q_{C1} of Figure 3 are represented as triple $< \pi_Q, \sigma_Q, operand_Q >$ in statement (iv) and (iii) of Figure 4 respectively. The query Q_U is statisfiable (or completely answerable) from Q_{C1} because intersection of projected attributes is not empty and there is no null value

domain in predicate condition. According to Lee (1999, pp.28-36) two queries are disjoint if either intersection of their projected attributes is empty or there is no combined condition between user and cached query predicates.

$$Q_U \cap Q_C = <Q_{UA} \cap Q_{CA}, Q_{UP} \cap Q_{CP}, R> \quad \text{(i)}$$

$$\pi_{QUA \cap QCA}\ , \sigma_{QUP \cap QCP} \quad \text{(ii)}$$

$$Q_{C1} = <\{Ename,Department\},\ \{-,-,-,\{50,\ldots,100\},-\},\ Emp> \quad \text{(iii)}$$

$$Q_U = <\{Ename,Department\},\ \{-,-,-,\{55,\ldots.,100\},-\},\ Emp> \quad \text{(iv)}$$

Fig. 4. Query Intersection (Lee et al., 1999).

3.2 Query trimming

The concept of query trimming was introduced by (Keller A.M. and Basu J., 1996) and formally given by Ren (2003, pp.192-210). Ren (2003, pp.192-210) gave a comprehensive algorithm for query processing. In the start of the algorithms it is checked if the user query attributes are subset of cached semantics attributes, then perform query trimming based upon the implication or satisfiability of predicates. If the user query attributes is not a subset of cached semantics attributes, then there may be some common attributes. In this case, if query predicate implies cache predicates or there are common predicates between the query and cache semantics, then form the probe and remainder query as per the logic given by the algorithm. In other words the logic is based on checking implication and satisfiability of a user and cached query predicates (based upon the already published material, as explained in the next section) and finding common part between the user and cached query attributes.

Much work has been contributed towards finding implication and satisfiablity between a user and cached query predicates (Guo S et al., 1996), (Härder T. and Bühmann A., 2008). Simplified concept of implication and satisfiability is, let us have a user query predicate Q_{UP} and semantic segment predicate Q_{CP}, then there are three scenarios:

- $Q_{UP} \rightarrow Q_{CP}$, i.e. User predicate implies segment predicates, implying that the whole answer of Q_{UP} is contained in Q_{CP}.
- ($Q_{UP} \wedge Q_{CP}$ is satisfiable), implying that part of Q_{UP} answer is contained in Q_{CP}.
- ($Q_{UP} \wedge Q_{CP}$ is unsatisfiable), implying that there is no common part between Q_{UP} and Q_{CP}.

Remainder queries were trimmed again after comparing with other semantic cache segments with the same algorithm. It continues until it is decided that the cache does not further contribute to the query answering. This approach forms an iterative behavior, which was handled by a proposed query plan tree structure. This plan tree expresses the relationship of cache items and query subparts.

Query trimming techniques have some short comings, such as time and space efficiency, and complexity of the trimming process (Makki K. S and Andrei S, 2009), (Makki K. S and Rockey M., 2010). When query is trimmed into probe ($Q_{UP} \wedge Q_{CP}$) and remainder ($Q_{UP} \neg \wedge Q_{CP}$) part, the negation of the cached query predicate in remainder part make it much more expanded term if it contains disjunctions. This expansion created by negation of a term was shown with example (Makki K. S and Andrei S, 2009).

A relational query can be visualized as a rectangle with boundaries set by query predicate values. So according to (Makki K. S and Andrei S, 2009), (Makki K. S and Rockey M., 2010) semantic cache query processing based on query trimming is problem of finding intersection between two finite rectangles. Six cases that are extended form of Figure 1.1 are given in (Makki K. S and Andrei S, 2009), (Makki K. S and Rockey M., 2010) to show relationship between rectangles of user and cached queries. These rectangular representations do not depict implicit knowledge present in the semantics of user and cached queries. An technique named Flattening Bi-dimensional Interval Constraints (FBIC) was proposed (Makki K. S and Andrei S, 2009). Based on FBIC an algorithm for handling disjunctive and conjunctive queries was given by Makki (Makki K. S and Rockey M., 2010). The algorithm works for only single disjunctive case, where conjunctive cases are same as provided by (Makki K. S and Andrei S, 2009).

Finding intersection between rectangles of user and cached queries was done by comparing Bounds (Lower or Upper) of both rectangles. But computing comparable bounds were not given (Makki K. S and Andrei S, 2009), (Makki K. S and Rockey M., 2010).

3.3 Satisfiability and implication

Finding whether there exists a satisfiable part between two formulas or whether one implies the other is central to many database problems such as query containment, query equivalence, answering queries using views and database cache. So according to Guo (Guo S et al., 1996) implication is defined as "S implies T, denoted as S $\rightarrow$ T, if and only if every assignment that satisfies S also satisfies T". Similarly satisfiability is defined as "S is satisfiable if and only if there exists at least one assignment for S that satisfies T." (Guo S et al., 1996) had given algorithm to compute implication, satisfiability and equivalence for given conjunctive formulas in integer and real domain. Let us have a formula (*Salary* < *20K AND Salary* > *8K AND Department* = *'CS'*) is satisfiable, because the assignment {12K/*Salary* , CS/*Department*} satisfies the formula. Similarly a formula (*Salary* >*10K OR Salary* < *12K)* is a tautology, because every assignment under this formula is satisfiable.

Satisfiability and implication results in databases (Guo et al., 1996),(J.D. Ullman, 1989),(A.Klug, 1988),(Rosenkrantz and Hunt, 1980), (Sun et al., 1989) are relevant to the computation of probe and remainder query in semantic cache query processing for a class of queries that involve inequalities of integer and real domain. Previous work models the problem into graph structure.

Rosenkrantz and Hunt (Rosenkrantz and Hunt, 1980) provided an algorithm of complexity $O(|Q|^3)$ for solving satisfiability problem; the expression S to be tested for satisfiability is the conjunction of terms of the form $X\ op\ C$, $X\ op\ Y$, and $X\ op\ Y + C$.

Guo et al. (Guo et al., 1996) provided an algorithm (GSW) for computing satisfiability with complexity $O(|Q|^3)$ involving complete operator set and predicate type X op C, X op Y and X op Y + C. Here we demonstrate GSW algorithm (Guo et al., 1996) for finding implication and satisfiability between two queries.

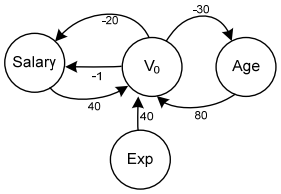
The GSW algorithm starts with transforming all inequalities into normalized form through given rules. It was proved by Ullman (J.D. Ullman, 1989) that these transformations still holds equality. After these transformation remaining operator set become $\{\leq, \neq\}$.

(1) $(X \geq Y+C) \equiv (Y \leq X - C)$
(2) $(X < Y+C) \equiv (X \leq Y + C) \wedge (X \neq Y + C)$
(3) $(X > Y + C) \equiv (Y \leq X - C) \wedge (X \neq Y + C)$
(4) $(X = Y + C) \equiv (Y \leq X - C) \wedge (X \leq Y + C)$
(5) $(X < C) \equiv (X \leq C) \wedge (X \neq C)$
(6) $(X > C) \equiv (X \geq C) \wedge (X \neq C)$
(7) $(X = C) \equiv (X \leq C) \wedge (X \geq C)$

Satisfiability of a conjunctive query Q is computed by constructing a connected weighted-directed graph $G_Q=(V_Q,E_Q)$ of Q after above transformation. Where V_Q are the nodes representing predicate attributes of an inequality and E_Q represent an edge between two nodes. An inequality of the form X op Y + C has X and Y nodes and an edge between them with C weight. The inequality X op C is transformed to X op V_0 + C by introducing a dummy node V_0.

According to GSW (Guo et al., 1996) algorithm, for any query Q if a negative-weighted cycle (a cycle whose sum of edges weight is negative) found in G_Q then Q is unsatisfiable. Otherwise Q is satisfiable. Testing satisfiability among user query Q_U and cached segment Q_S require us to construct a graph ($G_{Qu \wedge Qs}$) of ($Q_U \wedge Q_S$) and check $G_{Qu \wedge Qs}$ for any negative weighted cycle. Negative weighted cycle is found through Floyd-Warshall algorithm (R.W. Floyd, 1962). Complexity of Floyd-Warshall algorithm is $O(|V|^3)$, so finding satisfiability become $O(|Q_U \wedge Q_S|^3)$.

An algorithm with $O(|S|^3 + K)$ complexity for solving the implication problem between two conjunctive inequalities S and T was presented by Ullman (J.D. Ullman, 1989) and Sun (Sun et al., 1989). Conjunctive queries of the form X op Y were studied by (A.Klug, 1988) and (Sun et al., 1989). Implication between conjunctive queries of the form X op Y +C was addressed by GSW algorithm (Guo et al., 1996) with complexity $O(|Q_U|^2 + |Q_C|)$. GSW Implication (Guo et al., 1996) requires that Q_U is satisfiable. At first the implication algorithm constructs the closure of Q_U i.e., a universal set that contains all those inequalities that are implied by Q_U. Then, $Q_U \rightarrow Q_S$ if Q_S is a subset of the Q_U closure.



	Salary	V_0	Age	Exp
Salary	20	40	10	∞
V_0	-20	20	-30	∞
Age	60	80	50	∞
Exp	20	40	10	0

[(V_0 ≤ Salary -1), (Salary ≤ V_0 +40), (V_0 ≤ Salary -20), (V_0 ≤ Age -30), (Age ≤ V_0 +80), (Exp ≤ V_0 +40)]

Fig. 5. (a) $[Q_{U1} \wedge Q_S]$ and $G_{QU1 \wedge QS}$ (b) Shortest Path Table

Example 2: Let us have a user query $Q_{U1} = \pi_{\text{Age,Salary,Exp}} \sigma_{\text{Salary}\geq 20K \wedge \text{Age}\geq 30 \wedge \text{Age}\leq 80 \wedge \text{Exp}\leq 40}$ over cached segment Q_{C7} of Example 1. The directed weighted graph $G_{QU1 \wedge QC7}$ of $Q_{U1} \wedge Q_{C7}$ is shown in Figure 5(a). Q_{U1} is satisfiable with respect to Q_{C7}, as there is no negative weighted cycle in $G_{QU \wedge QC7}$.

3.4 Bucket algorithm

As discussed earlier, a user of data integration system poses query in term of mediated schema, because root sources are transparent in such systems. A module of data integration system translate/reformulate a user query that refers directly to the root sources. Several reputed algorithms exist for such query reformulation/rewriting (Levy A.Y et al., 1996), (Duschka O.M. and Genesereth M.R. 1997), (Pottinger R. and Levy A. 2000). In context of semantic cache the root sources are the cache segments and the mediated schema is the cache description. The goal of the bucket algorithm (Levy A.Y et al., 1996) is to reformulate a user query that is posed on the mediated (virtual) schema into a query that refers directly to the available (local/cached) data sources. This reformulation is known as query-rewriting. Both the query and the sources are described by select-project-join queries that may include atoms of arithmetic comparison predicates. The bucket algorithm returns the maximally-contained rewriting of the query using the views. This rewriting is a maximally-contained but not an equivalent one.

We demonstrate working (in context of semantic cache query processing) of bucket algorithm with example.

$$Q_{C1} \leftarrow \pi_{Ename,\ Department}\ \sigma_{Age \geq 50}\ (Emp);$$
$$Q_{C2} \leftarrow \pi_{Age,\ Department}\ (Emp);$$
$$Q_{C3} \leftarrow \pi_{Age,\ Department}\ \sigma_{Exp < 15}\ (Emp);$$
$$Q_{U} \leftarrow \pi_{Age,\ Department}\ \sigma_{Exp > 20\ \wedge\ Age < 70}\ (Emp);$$

Fig. 6. User Query (Q_U) Over Cached Queries

Let us have Q_{C1}, Q_{C2} and Q_{C3} (shown in Figure 6) in cache, and a user query Q_U (shown in Figure 6) is posed over them. As shown in Table 1 below, according to bucket algorithm both cached queries Q_{C1} and Q_{C2} are candidate selection for its bucket. Since there is no inconsistency between user query predicate and cached queries (i.e. *Age* ≥*55* consistent with *Age* < *70*) when compared in isolation (atomically). Where Q_{C3} is excluded due to predicate inconsistency (i.e. *Exp* < *15* inconsistent with *Exp* > *20*). In second step of bucket algorithm, elements of buckets are combined together to form a rewriting of the user query. The rewritten query (*Q'*) in this case is shown in Figure 7 below.

Emp(Empid,Ename,Departmen t,Age,Salary,Exp)
Q_{C1}*(Ename', Department)* Q_{C2}*(Age, Department)*

Table 1. Contents of Bucket. The attribute not required by user query is shown as primed attribute.

$$Q' \leftarrow \pi_{Age,\ Department}\ \sigma_{Age < 70}\ (Q_{C1}, Q_{C2});$$

Fig. 7. Rewritten Query Q'.

3.4.1 Example

We follow the results produced by maximally-contained query rewriting algorithm named bucket algorithm (Levy A.Y et al., 1996) provided above. The predicate (*Exp* > *20*) is pruned because query cannot be executed over cached data as there is no information present against *Exp* attribute. Further more if the rewritten query (*Q'* shown in Figure 7) executed locally, it will give unnecessary/incorrect results. These results are maximally-contained or maximum data retrieval (MDR) but the results contain tuples that are not part of the actual user query (Q_U). Figure 8 (a) shows the collective dataset of cached queries Q_{C1} and Q_{C2} (Figure 6). The rewritten query (*Q'*) executed over cached data is shown in Figure 8 (b). The data items shown as strike circle (-●-) in Figure 8 (b) are the required results of user query (Q_U). Where results retrieved by the rewritten query (*Q'*) are not the precise answer for the user query (Q_U).

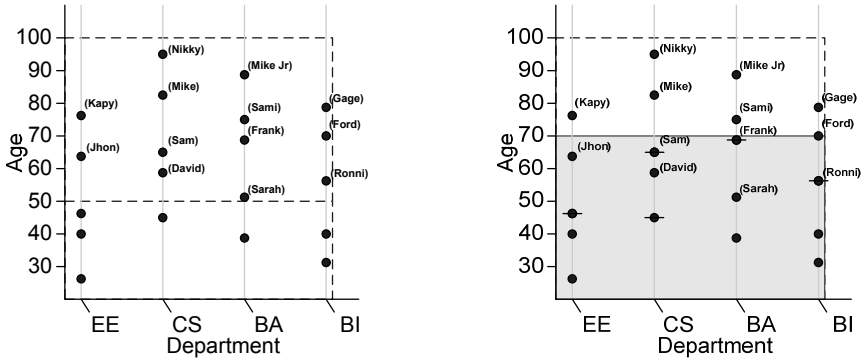


Fig. 8. (a) Collective Data of Cached Query Q_{C1} and Q_{C2}. (b) Rewritten Query Q' Over Cached Query Q_{C1} and Q_{C2}.

Bucket algorithm does not compute probe and remainder queries separately. So there is no way to determine the available and unavailable answer from the cache.

3.5 Semantic reasoning for web queries

A web-based cache system is different from data caching. In web cache, special proxy servers store recently visited pages for later reuse. A uniform resource locater (URL) is a user query that is posed over web cache system. Any page in cache is being used whenever a user given URL matches the cache page header. This type of caching strategy is similar to page-caching, where binary results (complete answer or no-answer) are possible but partial answer cannot be determined.

However, searching performed over web resources through Boolean queries (keywords conjunction with AND & NOT operators) do not work in a plain page caching system. Because the user query in this case is not a URL, and extracting qualified tuples against an individual keyword or whole query from page headers is not possible (Chidlovskii B and Borghoff U. M., 2000), (Qiong L and Jaffrey F. N., 2001). Semantic cache was introduced as an alternative to plain page caching where cache is managed as semantic regions.

Web queries over web resources are different than queries posed over databases. As there is no attribute and predicate part in web queries, also it neither contain join operator. And the problem of answering web-queries can be reduced to *set containment problem*.

There is a lot of research work on semantic caching for web queries. Such as (Chidlovskii B and Borghoff U. M., 2000) addressed both semantic cache management and query processing of web queries for meta-searcher systems. Their technique is based on a signature file method. In which a signature is given to every semantic region for processing all cases (similar to Figure 1) of containment and intersection.

A cache model was proposed for database applications using web techniques (Anton J. et al., 2002). Cache elements were stored as web pages/sub pages called fragments and sub fragments with their header information called template. Fragments can be indexed or shared among different templates. Fragments, sub-fragments and templates were updated or expired based on their unique policy which included expiration, validation and invalidation information. In this case data retrieval is performed by matching template information with requested query and subsequent fragments or sub-fragments are returned. Partial answer retrieval is possible in this technique as sub-fragments alone can be resulted to a user query. But still this technique is closer to page cache technique, where each fragment is itself a page.

3.6 Pattern Prime Product (PPT) reasoning for XML queries

The information that is available on the web is unstructured, extensible mark-up language XML is used to provide the structure to the web information/data. As described above the querying mechanism for current web is keyword based search. Keyword based search is considered to be the non-semantic (Mandhani B. and Suciu D., 2005), (Sanaullah, M., 2008).

A novel method of checking containment is proposed by Gang Wu and Juanzi Li (Gang Wu and Juanzi Li, 2010). Each node in the query is assigned a unique prime number and then the product of these prime numbers is calculated by a specific method. This product is called Pattern's Prime producT (PPT). The query is stored in the cache along with this PPT.

On each next issued query the same procedure is followed to assign unique prime numbers to each node and if any node of the query matches with any existing stored view's node then the same prime number is assigned to new node as it was allotted to previously stored node. The PPT of the new query is calculated and then divided by the PPT of all stored views. If any of stored views completely divides the PPT of the query then that view is selected and rest are rejected. The selected view further processed to make sure whether the occurrences of the nodes in the query and view is similar, i.e Qk = Vk where k is the position of kth axis node. The PPT of each infix is also checked.

3.6.1 Example

An XML document is shown in Figure 9. A user issues a query /lib/book and as a result the technique loads all the results of "lib", "book" nodes in the cache and assigns prime numbers to each node i.e. "lib"=2, "book"=3. After assigning the prime numbers a prime product is calculated as follows.

(2*3), here 6 is the Tree Pattern Prime Product of the view.

Now if the user again issues the query /lib/book/author then each node in the query is assigned the same prime number as it was previously assigned to the nodes in the view. Here 2 is assigned to "lib" and 3 is assigned to "book". "author" appeared first time so a new prime number i.e. 5 is assigned to author node. Dividing the prime product of query (90) by the prime product of view (6) will yields the result 15, means query is completely divided by the view. If the prime product of a view completely divides the prime product of a query then it further checks the following conditions. Whether the order of appearance of each axis node in the view and query is similar and if the answer is true then it means that the query is contained in the view.

3.6.2 Example

If a query contains predicates, for example A[b[b[a]]]/c/d the tree of this query is shown in figure 9. The prime product is calculated as shown below

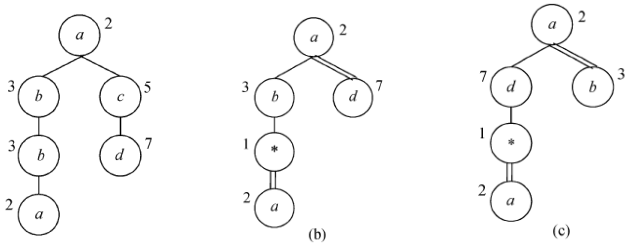


Fig. 9. Prime Product Calculation.

PPT of b=(2*3)*(3*1)*(1*2)*(2*7) = 504
PPT of c= (2*7)*(7*1)*(1*2)*(2*3) = 1176

Now only the PPT of b completely divides the PPT of the query so b is selected in the first condition of the algorithm.

This algorithm retrieves the results of all axis nodes given in the query for example if we issue following query to the document shown in figure 1 "\lib\book[price>30]". Then apart from the presence of a predicate it retrieves all the result of book node and stores it in the cache. This action requires more cache space.

3.7 Subsumption analysis reasoning

Description logics (a language of logic family) DL claims that it can express the conceptual domain model/ontology of the data source and provide evaluation techniques. Since structured query language (SQL) is a structured format, it can be classified under

subsumption relationship. A well known technique named Tabulex provides structural subsumption of concepts. Description logic (DL) is assumed to be useful for semantic cache query processing and management (Ali et al. 2010). The relational queries can be modelled / translated in DL and DL inference algorithms can be used to find query containments. The translation of relational query to DL may have not the same spirit as that of querying languages for DL systems, but is sufficient for finding the query containment of relational queries (Ali et al. 2010). The subsumption reasoning (containment) of the semantics of the data to be cached is very useful in eliminating the redundant semantics and minimizing the size of semantic cache for the same amount of data.

The tableaux algorithm (Baader et al., 1991a) (Hollunder et al., 1990) is instrumental to devise a reasoning service for knowledge base represented in description logic. All the facts of knowledge base are represented in a tree of branches with intra-branch logical AND between the facts and inter-branch logical OR, organized as per the rules of tableaux algorithm (Baader et al., 2003). A clash in a branch represents an inconsistency in that branch and the model in that branch can be discarded. The proof of subsumption or unsatisfiability can be obtained if all the models (all the branches) are discarded this way (Baader et al., 2003).

The proposed solution (Ali et al. 2010) consists of two basic steps: First user query (relational) is translated into DL. The translated query is then evaluated for subsumption relationship with previously stored query in the cache by using the sound and complete subsumption algorithm given in (Baader et al., 91b) (Lutz et al., 2005).

3.7.1 Example

Considering, another scenario having predicates conditions with disjunctive operator in Figure 10. All the three branches yields to clash in checking $Q3 \sqsubseteq Q4$; therefore, *Q4* contain *Q3*. In first branch (Line 8 in Figure 10) after applying the *Or rule, Emp* and $\neg Emp$ yields to clash. In second branch (Line 9 in Figure 10) *ename* and $\neg ename$ yields to clash, and in third *branch (Line 10* Figure 10), $\geq 30k(sal)$ and $\leq 19k(sal)$ yields to clash. All tree branches (Line 8, 9, 10 of Figure 10) yield to clash in opening the tableaux algorithm; therefore, $Q3 \sqsubseteq Q4$.

1.	Q3: Select ename, age from Emp where sal ≥ 30k	
2.	Q4: Select ename from Emp where sal ≥ 20k ⊔ age ≥20	
3.	$Q3 \sqsubseteq Q4$	
4.	$Q3 \sqcap \neg Q4$	
5.	(Emp ⊓ ename ⊓ age ⊓ ≥ 30k(sal)) ⊓ ¬(Emp ⊓ ename ⊓ ≥ 20k(sal) ⊔ ≥20(age))	
6.	(Emp ⊓ ename ⊓ age ⊓ ≥ 30k(sal)) ⊓ (¬Emp ⊔ ¬ename ⊔ 19k(sal) ⊓ 19(age))	**Move negation inward**
7.	(Emp , ename , age , ≥ 30k(sal) , ¬Emp ⊔ ¬ename ⊔ 19k(sal) , 19(age))	**AND rule**
8.	Emp , ename , age , ≥ 30k(sal) , ¬Emp, , 19(age)	**Clash**
9.	Emp , ename , age , ≥ 30k(sal), ¬ename, 19(age)	**Or Rule Clash**
10.	Emp , ename , age , ≥ 30k(sal), 19k(sal) , 19(age))	**Clash**
11.	$Q3 \sqsubseteq Q4$	***All branches leads to Clash***

Fig. 10. Query Containment using Tableux.

4. Conclusion

In this chapter we demonstrated several reasoning techniques of query processing in semantic cache. This chapter provides overview of semantic cache application in different domains such as relational databases, web queries, answering from views, xml based queries and description logic based queries.

Semantic cache query processing techniques are unstructured-semantics approaches, in which semantics are extracted from structured representations that have no semantics within their representations.

5. Acknowledgment

We would like to acknowledge Mr. Ishtique Ahmad for providing help on XML based semantic cache, Mr. Tariq Ali for DL based subsumption analysis, and Mr. Munir Ahmed for providing useful feedbacks.

6. References

A. Klug, "On Conjunctive Queries Containing Inequalities," *ACM,* vol. 35, no. 1, pp. 146-160, Jan. 1988.

Ahmad, M and Qadir, M.A., (2008). "Query Processing Over Relational Databases with Semantic Cache: A Survey", 12th IEEE International Multitopic Conference 2008, INMIC08.

Ahmad, M and Qadir, M.A., "Query Processing and Enhanced Semantic Indexing for Relational Data Semantic Cache", Center for Distributed and Semantic Computing, Mohammad Ali Jinnah University, Islamabad, Pakistan 2007.

Ahmed M. U., Zaheer R. A., and Qadir M. A., "Intelligent Cache Management for Data Grid", Australasian Workshop on Grid Computing and e-Research (AusGrid 2005).

Ali, T., Qadir, M. A., Ahmed, M., Translation of Relational Queries into Description Logic for semantic Cache Query Processing, International Conference on Information and Emerging Technologies (ICIET) 2010.

Andrade H., Kurc T., Sussman A., Saltz J., "Active semantic caching to optimize multidimensional data analysis in parallel and distributed environments" Elsevier Journal on Parallel Computing, volume 33, pp 497-520, 2007.

Anton J., Jacobs L., Liu X., Parker J., Zeng Z. and Zhong T., "Web caching for database applications with Oracle Web Cache", Proceedings of the 2002 ACM SIGMOD international conference on Management of data, June 03-06, 2002, Madison, Wisconsin.

Baader, F., Hollunder, B., A terminological knowledge representation system with complete inference algorithm. In: Proc. Workshop on Processing Declarative Knowledge, PDK-91Lecture Notes in Artificial Intelligence, Springer, Berlin, pp. 67–86, 1991.

Baader, F., Hanschke, P. A schema for integrating concrete domains into concept languages. In Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91), pages452–457, 1991.

Baader, F., McGuinness, D., Nardi, D., Patel-Schneider, P., The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2003.

Bashir M F. and Qadir M. A., "Survey on Efficient Content Matching over Semantic Cache", 10th IEEE International Multitopic Conference 2006, INMIC06.

Bashir M F. and Qadir M. A., "HiSIS: 4-Level Hierarchical Semantic Indexing for Efficient Content", 10th IEEE International Multitopic Conference 2006, INMIC06.

Bashir M.F., Zaheer R.A., Shams Z.M. and Qadir M.A.. "SCAM: Semantic Caching Architecture for Efficient Content Matching over Data Grid". AWIC, Springer Heidelberg, Berlin, 2007. pp. 41-46.

Bashir, M.F and Qadir, M.A., "ProQ – Query Processing Over Semantic Cache For Data Grid", Master's Thesis, Center for Distributed and Semantic Computing, Mohammad Ali Jinnah University, Islamabad, Pakistan 2007.

Brunkhorst I. and Dhraief H., "Semantic Caching in Schema-based P2P-Networks", DBISP2P 2005/2006, LNCS 4125, pp. 179-186, 2007. Springer-Verlag Berlin Heidelberg 2007.

Chidlovskii B and Borghoff U. M., "Signature file methods for semantic query caching". In: Proc. 2nd European Conf. on Digital Libraries, September, 1998, Heraklion, Greece, Berlin Heidelberg New York: Springer, LNCS 1513, pp. 479–498

Chidlovskii B., Borghoff U. M., "Semantic caching of Web queries", The International Journal on Very Large Data Bases, v.9 n.1, p.2-17, March 2000.

Dar S, M. J. Franklin, B. T. Jonson, D. Srivastava, M. Tan, "Semantic Data Caching and Replacement", in proceedings of 22nd VLDB Conference, Mumbai, 1996.

Duschka O.M. and Genesereth M.R. "Query planning in infomaster". In: Proc.ACM Symposium on Applied Computing. pp 109–111, San Jose, Calif., USA, 1997.

Gang Wu and Juanzi Li, "Semantic Web", Intech Publisher 2010, pp 97-106.

Godfrey P. and Gryz J., "Semantic Query Caching for Heterogeneous Databases," Proc. KRDB Conf. Very Large Databases, vol. 6, pp. 1-6, 1997.

Guo S., Sun W., and Weiss M.A., "Solving Satisfiability and Implication Problems in Database Systems," ACM Trans. Database Systems, vol. 21, no. 2, pp. 270-293, 1996.

Halevy, A.Y., "Answering queries using views", VLDB J., vol. 10, pp. 270-294, 2001.

Hao X., Zhang T., and Li L., "The Inter-clause Optimization Technique in Semantic Caching Query Evaluation", Journal of Information & Computational Science volume 2, no. 1, pp 27-33, 2005.

Härder T. and Bühmann A., "Value complete, column complete, predicate complete", VLDB Journal, 2008, pp 805-826.

Hector García-Molina, Jeffrey D. Ullman and Jennifer Widom., "Database Systems: the Complete Book", GOAL Series, 2001.

Hollunder, B., Nutt, W., Subsumption algorithms for concept languages. Research Report RR-90-04, Deutsches Forschungszentrum fur Kunstliche Intelligenz GmbH (DFKI), April 1990.

J.D. Ullman, Principles of Database and Knowledge-Base Systems, vol. 11. Computer Science Press, 1989.

Jónsson B. Þór., Arinbjarnar M., Þórsson B., Franklin M. J., Srivastava D., "Performance and overhead of semantic cache management", ACM Transactions on Internet Technology (TOIT), v.6 n.3, p.302-331, August 2006.

Keller A.M. and Basu J., "A Predicate-Based Caching Scheme for Client-Server Database Architectures," VLDB J., vol. 5, no. 2, pp. 35-47, Apr. 1996.

Lee D. and Chu W. W., "Semantic Caching via Query Matching for Web Sources", CIKM, ACM, 1999.

Lee K.C.K, H.V. Leong, and A. Si, "Semantic Query Caching in a Mobile Environment," Mobile Computing and Comm. Rev., vol. 3, no. 2, pp. 28-36, Apr. 1999.

Levy A.Y., Rajaraman A., Ordille J. J., "Querying Heterogeneous Information Sources Using Source Descriptions", Proceedings of the 22th International Conference on Very Large Data Bases, p.251-262, September 03-06, 1996

Levy A.Y. "Logic-based techniques in data integration", Minker J (ed.) Logic-based artificial intelligence. Kluwer Academic, Dordrecht, 2000, pp 575–595

Lutz, C., Milicic., M. A tableau Algorithm for Description Logics with Concrete Domains and GCIs, Tableaux 2005, LNAI 3702, pp. 201-216, 2005

Makki K. S and Andrei S. "Utilizing Semantic Caching in Ubiquitous Environment", IWCMC'09, June 21-24,2009, Leipzig, Germany.

Makki K. S and Rocky M., "Query Visualization for Query Trimming in Semantic Caching", IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2010.

Mandhani B. , Suciu D., "Query caching and view selection for XML databases", Proceedings of the 31st international conference on Very large data bases (VLDB), 2005, Trondheim, Norway.

Nama B., Shin M., Andrade H., and Sussman A., "Multiple query scheduling for distributed semantic caches", In Proc. of Journal of Parallel and Distributed Computing, 70- (2010) pp. 598-611

Pottinger R. and Levy A. "A scalable algorithm for answering queries using views". In: Proc. of VLDB. pp 484-495, Cairo, Egypt, 2000.

Qiong L and Jeffrey F. N., "Form-Based Proxy Caching for Database-Backed Web Sites", Proceedings of the 27th International Conference on Very Large Data Bases, p.191-200, 2001.

Ren, Q., Dunham, M.H., and Kumar, V., (2003). "Semantic Caching and Query Processing". IEEE Transactions on Knowledge and Data Engineering, IEEE Computer Society, 2003, pp. 192-210.

Rosenkrantz D.J. and Hunt H.B., "Processing Conjunctive Predicates and Queries," Proc. Conf. Very Large Databases, pp. 64- 71, 1980.

R.W. Floyd, "Algorithm 97 Shortest Path" *Comm. ACM,* vol. 5, no. 6, p. 345, June 1962.

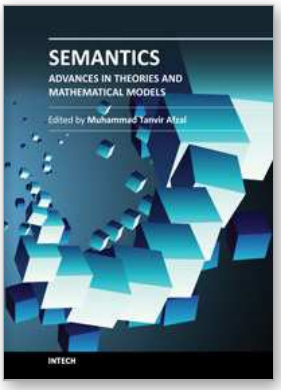
Safaeei A.A, Haghjoo M. Abdi S., "Semantic cache schema for query processing in mobile databases", 3rd IEEE International Conference on Digital Information Management, 2008. ICDIM 2008.

Sanaullah, M., Qadir, M.A., and Ahmad, M., (2008) "SCAD-XML: Semantic Cache Architecture for XML Data Files using XPath with Cases and Rules ". 12th IEEE

International Multitopic Conference, INMIC 2008, IEEE, Karachi, Pakistan, December 2008.

Sun, X., Kamell, N. N., AND NI, L.M. 1989. Processing implication on queries. *IEEE Trans. Softw. Eng. 5*, 10 (Oct.), 168–175.

The TimesTen Team Mid-tier caching: the TimesTen approach. In: SIGMOD Conference, pp. 588–593 (2002).



Semantics - Advances in Theories and Mathematical Models

Edited by Dr. Muhammad Tanvir Afzal

ISBN 978-953-51-0535-0
Hard cover, 284 pages
Publisher InTech
Published online 25, April, 2012
Published in print edition April, 2012

The current book is a nice blend of number of great ideas, theories, mathematical models, and practical systems in the domain of Semantics. The book has been divided into two volumes. The current one is the first volume which highlights the advances in theories and mathematical models in the domain of Semantics. This volume has been divided into four sections and ten chapters. The sections include: 1) Background, 2) Queries, Predicates, and Semantic Cache, 3) Algorithms and Logic Programming, and 4) Semantic Web and Interfaces. Authors across the World have contributed to debate on state-of-the-art systems, theories, mathematical models in the domain of Semantics. Subsequently, new theories, mathematical models, and systems have been proposed, developed, and evaluated.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821