Answering Queries by Semantic Caches

Parke Godfrey Jarek Gryz

Department of Computer Science
York University
Toronto, Ontario, Canada
{godfrey, jarek}@cs.yorku.ca

August 1999

What is Semantic Query Caching?

Semantic query caching (SQC): Use the results of old queries to answer new queries.

A *semantic query cache* (SQC) is a

- a local materialization of a query, annotated with
- a query expression.

Other types of caching used in databases:

- tuple-based
- page-based

It is unclear how tuple-based or page-based could be extended for heterogeneous database environments.

Semantic query caches also offer advantages. They

- exploit *semantic locality*.
  (Dar, Franklin, Jonsson, & Srivastava [VLDB'96])
- offer greater flexibility.
  - Caches can be *combined* to answer queries.
  - Can determine when caches completely answer query.
- are easy to capture and store.

Applications of Semantic Query Caching

What can semantic query caching buy us, especially in a heterogeneous, mediated environment?

- **Query optimization**
  - Improvement in overall query response time
    (Traditional optimization)
  - Saving money
  - Optimization of queries with few answers
- **Data Security**
- **Fault tolerance**
- **Approximate answering (aggregates)**
  (Hellerstein, Haas, & Wang [SIGMOD'97])
- **Better user interaction**
  - Answer set pipelining
  - Indirect answering
  - Limiting the size of the answer set

Our Goals

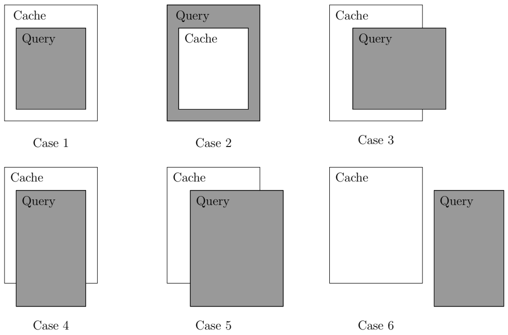
Seek to define a **general framework** in *logic* for semantic query caching, and the use of semantic caches. Framework should be

- **Relationally Complete**
  - *All* the relational algebra—including *join* and *union*—can be used across the caches to answer queries.
- **Flexible**
  - Query may be only *partially* answerable via cache. In this case, the query should be answered in part via cache and the rest via evaluation.
- **Parameterizable**
  - SQC usage can be parameterized to be optimized for different purposes. For example, query optimization, and answer pipelining.

Problems at hand: (Outline)

1. Deciding when answers are in cache.
2. Extracting answers from cache.
3. Accessing semantic overlap / semantic independence.
4. Evaluating semantic remainders.

Relationships between Caches and Queries



Horizontal	=	rows / tuples
Vertical	=	attributes

Not interested in the *actual* tuples in common, but the tuples that *must* be in common.

Notation (Datalog)

Conjunctive Queries

$$\mathcal{Q}: \leftarrow \textit{employee}\,(\underline{N},\ S,\ A),\ \textit{benefits}\,(S,\ P).$$

Views / Rules (Intensional Predicates)

$$\textit{employee}\,(N,\ S,\ A) \leftarrow \textit{payroll}\,(S,\ \_),\ \textit{personnel}\,(S,\ N,\ A).$$
$$\textit{employee}\,(N,\ S,\ A) \leftarrow \textit{contractor}\,(S,\ N,\ A,\ C),\ \textit{contract\#}\,(C,\ \textit{active}).$$

IDB & EDB

IDB: view definitions / rules
EDB: tables / facts

Cache Rules and Predicates

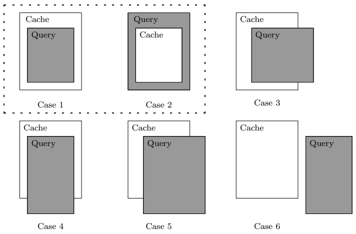
$$c_i\,(N) \leftarrow \textit{employee}\,(N,\ S,\ A),\ \textit{benefits}\,(S,\ P).$$

Cache Expression ($\mathcal{E}$)

Any conjunctive view (SPJ) written only with cache predicates.

Containment

When the query is contained by the caches



Questions

1. When is the query contained by the caches?
2. When can one answer, or partially answer, the query by the caches?

$$\mathsf{IDB} \models \forall.\ \mathcal{Q} \rightarrow (\mathcal{E}_1 \vee \ldots \vee \mathcal{E}_n)$$

Containment

Example

$$\mathcal{Q}: \leftarrow \textit{employee}\,(\underline{N},S,A),\ \textit{benefits}\,(S,P).$$
$$\mathcal{C}_1\text{: } c_1\,(N) \leftarrow \textit{employee}\,(N,S,A),\ A < 50.$$
$$\mathcal{C}_2\text{: } c_2\,(N) \leftarrow \textit{employee}\,(N,S,A),\ A > 20.$$

$\mathcal{E}_1$: $c_1\,(N)$
$\mathcal{E}_2$: $c_2\,(N)$

$$\mathsf{IDB} \models \forall.\ \mathcal{Q} \rightarrow (\mathcal{E}_1 \vee \mathcal{E}_2)$$

However, one cannot extract the answers to $\mathcal{Q}$ from $\mathcal{C}_1$ and $\mathcal{C}_2$.

Containment

When the caches (partially) answer the query

$$\mathsf{IDB} \models \forall.\, \mathcal{E} \rightarrow \mathcal{Q}$$

Equivalence

$$\mathsf{IDB} \models \forall.\, \mathcal{Q} \rightarrow (\mathcal{E}_1 \vee \ldots \vee \mathcal{E}_n)$$

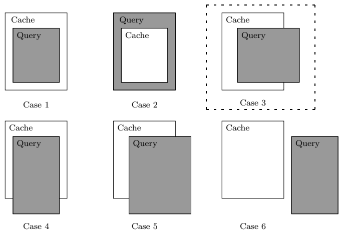
and, for each $\mathcal{E}_i$,

$$\mathsf{IDB} \models \forall.\, \mathcal{E}_i \rightarrow \mathcal{Q}$$

The only known way to show equivalence is to show containment in both directions.

- There are cases when *all* answers are contained, but *cannot* be retrieved.
- If one can only answer part of the query by the caches, how does one (efficiently) answer the *rest*?

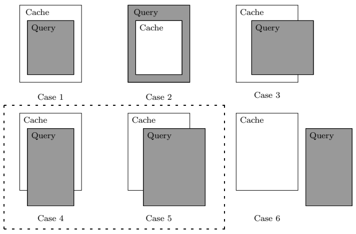
Abbreviated Containment



Abbreviated containment: Not all the attributes of the query can be retrieved, but a projection of the query is contained by the caches.

Semantic Overlap

Or how containment is not the whole story



$$\begin{array}{lrl} \mathcal{Q}: & \mathit{employee}\,(X) \leftarrow & \boxed{\mathit{payroll}\,(X),}\ \mathit{position}\,(X). \\ \mathcal{C}: & \mathit{taxed}\,(X) \leftarrow & \boxed{\mathit{payroll}\,(X),}\ \mathit{national}\,(X). \end{array}$$

Trickier to capture than one might expect.

$$\begin{array}{rl} a\,(X) \leftarrow & \boxed{c\,(X),}\ X > 5. \\ b\,(X) \leftarrow & \boxed{c\,(X),}\ X \leq 5. \end{array}$$

Semantic Overlap

Overlap Witness

First, there must exist a conjunctive query formula $\mathcal{W}$, called an *overlap witness*, such that

$$\models \forall.\, (\mathcal{Q} \rightarrow \mathcal{W}) \wedge (\mathcal{E} \rightarrow \mathcal{W})$$

For example,

$$\models \forall X.\ \mathit{payroll}\,(X) \wedge \mathit{position}\,(X) \rightarrow \mathit{payroll}\,(X)$$
$$\models \forall X.\ \mathit{payroll}\,(X) \wedge \mathit{national}\,(X) \rightarrow \mathit{payroll}\,(X)$$

This means that there is a shared resource.

Problems:

- *True* for $\mathcal{W}$ works.
- Does not guarantee that $\mathcal{Q}$ and $\mathcal{E}$ are semantically connected.

Semantic Overlap

Overlap Formula

Second, there must exist a conjunctive query formula $\mathcal{F}$, called the *overlap formula*, such that

$$\models \forall.\ (\mathcal{F} \rightarrow \mathcal{Q}) \wedge (\mathcal{F} \rightarrow \mathcal{E})$$

For example,

$$\models \forall X.\ \mathit{payroll}\ (X) \wedge \mathit{position}\ (X) \wedge \mathit{national}\ (X) \rightarrow \mathit{payroll}\ (X) \wedge \mathit{position}\ (X)$$
$$\models \forall X.\ \mathit{payroll}\ (X) \wedge \mathit{position}\ (X) \wedge \mathit{national}\ (X) \rightarrow \mathit{payroll}\ (X) \wedge \mathit{national}\ (X)$$

Problems:

- *False* for $\mathcal{F}$ works.

Note that $\mathcal{Q} \wedge \mathcal{E}$ always works.

Semantic Overlap

Both overlap witness and formula

If there is a non-tautological overlap witness and $\mathcal{Q} \wedge \mathcal{E}$ is not a contradiction (so there exists a non-contradictory overlap formula), then $\mathcal{Q}$ and $\mathcal{E}$ *extensionally overlap*.

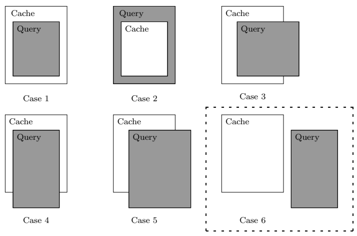
Interested in *most general* overlap formulas. $\mathcal{F}$ is most general if there exists no $\mathcal{G}$ such that

$$\models \forall.\ (\mathcal{F} \rightarrow \mathcal{G}) \text{ but } \not\models \forall.\ (\mathcal{G} \rightarrow \mathcal{F})$$

Intensional Overlap

Overlap with respect to IDB: There exist unfoldings $\mathcal{U}_{\mathcal{Q}}$ and $\mathcal{U}_{\mathcal{E}}$ of $\mathcal{Q}$ and $\mathcal{E}$, respectively, such that $\mathcal{U}_{\mathcal{Q}}$ and $\mathcal{U}_{\mathcal{E}}$ extensionally overlap.

Semantic Independence



Only once we have defined semantic overlap can we then define *semantic independence*.

$\mathcal{Q}$ and $\mathcal{E}$ are *semantically independent iff* they do not intensionally overlap in any way.

Semantic Remainder

$\mathcal{Q}$: $\leftarrow$ *employee* ($\underline{N}$,*S*,*A*).
$\mathcal{C}$: *c* (*N*) $\leftarrow$ *employee* (*N*,*S*,*A*), *benefits* (*S*,_).
$\mathcal{R}$: *benefits* (*S*,*B*) $\leftarrow$ *position* (*S*,*P*), *package* (*P*,*B*).

One can partially answer $\mathcal{Q}$ by the cache $\mathcal{C}$. Next, how to find the remaining answers?

Let $[\![\mathcal{Q}]\!]$ denote the answer set of $\mathcal{Q}$.

Let $\mathcal{Q}\backslash\mathcal{E}$ be called a *discounted query*. It at least evaluates to those answers of $\mathcal{Q}$ that cannot be retrieved via $\mathcal{E}$.

Two degenerate ways to define $\mathcal{Q}\backslash\mathcal{E}$ are

1. $\mathcal{Q}\backslash\mathcal{E} \equiv \mathcal{Q}$ $\quad([\![\mathcal{Q}\backslash\mathcal{E}]\!] = [\![\mathcal{Q}]\!])$
2. $\mathcal{Q}\backslash\mathcal{E} \equiv \mathcal{Q} \wedge \textsf{not}\, \mathcal{E}$ $\quad([\![\mathcal{Q}\backslash\mathcal{E}]\!] = [\![\mathcal{Q}]\!] - [\![\mathcal{E}]\!])$

Properties for Discounted Queries

- **soundness**

$$[\![\mathcal{Q} \backslash \mathcal{E}]\!] \subseteq [\![\mathcal{Q}]\!]$$

- **completeness**

$$[\![\mathcal{Q} - \mathcal{E}]\!] \subseteq [\![\mathcal{Q} \backslash \mathcal{E}]\!]$$

- **independence**

  $\mathcal{Q} \backslash \mathcal{E}$ and $\mathcal{E}$ should be semantically independent.

- **uniformity**

$$[\![\mathcal{Q} \backslash \mathcal{E}]\!] - [\![\mathcal{E} \backslash \mathcal{Q}]\!] = [\![\mathcal{Q}]\!] - [\![\mathcal{E}]\!]$$

- **cost effectiveness**

  Evaluating $\mathcal{Q} \backslash \mathcal{E}$ and $\mathcal{E}$ should cost less than evaluating $\mathcal{Q}$.

Related Work

- **Semantic Query Caching**
  - Adalı, Candan, Papakonst., & Subrahmanian [SIGMOD'96]
  - Dar, Franklin, Jonsson, Srivastava, & Tan [VLDB'96]
  - Godfrey & Gryz [KRDB'97]
  - Godfrey & Gryz [ICDT'99]
  - Keller & Basu [VLDB Journal 1996]
- **Answering Queries using Views (Logical Views, Mat. Views, & Query Folding)**
  - Chen & Roussopoulos [EDBT'94]
  - Gupta, Mumick, & Ross [SIGMOD'95]
  - Levy, Mendelzon, Sagiv, Srivastava [PODS'95]
  - Qian [ICDE'96]
  - Shmueli [PODS'87]
  - Ullman [ICDT'97]
- **Description Logics**
  - Goñi, Bermúdez, Blanco, & Illarramendi [KRDB'96]
  - Levy & Rousset [KRDB'96]
- **Semantic Query Optimization**
  - Godfrey, Gryz, & Minker [ISMIS'96]
  - Godfrey & Gryz [DDLP'96]
  - Godfrey & Gryz [DOOD'97]

Future Work

- **formalization**
  - Formalize notion, or notions, of $\mathcal{Q} \backslash \mathcal{E}$.
- **algorithms**
  - Reasoning over conjunctive query containment and Datalog containment is computationally hard.
  - What are good (possibly incomplete) tractable algorithms for important sub-classes of containment and overlap?
- **optimization**
  - ◦ What would *cost models* for SQC be?
  - What are good evaluation strategies for discounted queries?
- **cache currency**
  - ◦ Can caches be kept “reasonably” current inexpensively?
- **cache maintenance**
  - ◦ What would be a reasonable cache maintenance strategy?
  - ◦ When should caches be combined / split?