Combining Consistency and Confidentiality Requirements in First-Order Databases

Lena Wiese and Joachim Biskup

Fakultät für Informatik
LS 6 (ISSI)

September 7–9, 2009
Outline

1. Introduction
   - Inference Control
   - Controlled Query Evaluation
   - Preprocessing for CQE

2. Automizing Inference-Proofness

3. Prototype

4. Conclusion
Inference control

- Protect confidential and private information in database instance $db$
- Personalized security (confidentiality) policy $pot_{sec}$
- User profile (a priori knowledge) $prior$
- IC system automatically distorts some answers
  - Avoids harmful user inferences
- Here: modify input database
  - Remove tuples (like Data Privacy; Stouppa/Studer, 2009)
  - Add tuples (like Cover Stories; eg. Galinovic et al, 2007)
- Automatically generate “inference-proof” output instance
- Also related to Data Exchange (eg. Fagin et al, 2005) and Consistent Query Answering (eg. Chomicki, 2007)
Prior work: Controlled Query Evaluation (CQE)

- (Biskup/Bonatti, 2007; Biskup/Wiese, 2008)...
- Logical view of relational data model
- Database schema $\mathcal{DS} = \langle \mathcal{P}, \mathcal{D} \rangle$ with relation names $\mathcal{P}$ and database dependencies $\mathcal{D}$
- Infinite domain of values (constants) $\text{dom}$
- Complete database instance as finite set of tuples (ground atoms) + closed world world assumption
- Relational calculus as query language
preCQE: Preprocessing for CQE

- `dbadm` maintains `db`
- `secadm` declares `pot_sec`
- `useradm` declares `prior`

\[ Q = \langle \Phi_1, \Phi_2, \ldots \rangle \]

\[ A = \langle \text{eval}^*(\Phi_1)(db'), \text{eval}^*(\Phi_2)(db'), \ldots \rangle \]
**preCQE:** Preprocessing for CQE

- **dbadm** maintains `db`.
- **secadm** declares `pot_sec`.
- **useradm** declares `prior`.

**Q** = \(\langle \Phi_1, \Phi_2, \ldots \rangle\)

**A** = \(\langle \text{eval}^*(\Phi_1)(db'), \text{eval}^*(\Phi_2)(db'), \ldots \rangle\)
Preprocessing for CQE

Definition: Inference-proofness of $db'$

1. [Consistency] $I^{db'} \models prior$
2. [Confidentiality] $I^{db'} \not\models \Psi$ for every $\Psi \in pot_{sec}$

Definition: Distortion distance (amount of modified tuples)

[Availability] $db_{dist}(db') := card((db \setminus db') \cup (db' \setminus db))$

- Find $db'$ that satisfies constraint set $C := prior \cup Neg(pot_{sec})$
- Minimize amount of modified tuples $db_{dist}$ (maximize availability)
- No impact on runtime performance
- No user history ($log$ file) has to be stored
Example

\[ \mathcal{P} = \{ \text{Ill}, \text{Treat} \}, \]
\[ \text{dom} = \{ \text{Pete}, \text{Mary}, \text{Lisa}, \text{Paul}, \ldots, \]
\[ \text{Aids}, \text{Flu}, \text{Cancer}, \text{Myopia}, \ldots \]
\[ \text{MedA}, \text{MedB}, \text{MedC}, \ldots \} \]

<table>
<thead>
<tr>
<th>db:</th>
<th>Ill</th>
<th>Name</th>
<th>Diagnosis</th>
<th>Treat</th>
<th>Name</th>
<th>Treatment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pete</td>
<td>Aids</td>
<td></td>
<td>Pete</td>
<td>MedA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mary</td>
<td>Cancer</td>
<td></td>
<td>Mary</td>
<td>MedB</td>
<td></td>
</tr>
</tbody>
</table>

\[ \text{prior} = \{ \forall x (\text{Treat}(x, \text{MedA}) \rightarrow \text{Ill}(x, \text{Aids}) \lor \text{Ill}(x, \text{Cancer})) , \]
\[ \forall x (\text{Treat}(x, \text{MedB}) \rightarrow \text{Ill}(x, \text{Cancer}) \lor \text{Ill}(x, \text{Flu})) \} \]

\[ \text{pot}_{-}sec = \{ \exists x \text{Ill}(x, \text{Aids}), \exists x \text{Ill}(x, \text{Cancer}) \} \]
Example

\[ prior = \{ \forall x (\text{Treat}(x, \text{MedA}) \rightarrow \text{Ill}(x, \text{Aids}) \lor \text{Ill}(x, \text{Cancer})), \]
\[ \forall x (\text{Treat}(x, \text{MedB}) \rightarrow \text{Ill}(x, \text{Cancer}) \lor \text{Ill}(x, \text{Flu}))) \} \]
\[ \text{pot}_\text{sec} = \{ \exists x \text{Ill}(x, \text{Aids}), \exists x \text{Ill}(x, \text{Cancer}) \} \]
\[ \text{Neg}(\text{pot}_\text{sec}) = \{ \forall x \neg\text{Ill}(x, \text{Aids}), \forall x \neg\text{Ill}(x, \text{Cancer}) \} \]

Constraint set \( C := prior \cup \text{Neg}(\text{pot}_\text{sec}) \)

<table>
<thead>
<tr>
<th>Name</th>
<th>Diagnosis</th>
<th>Ill</th>
<th>Name</th>
<th>Treatment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pete</td>
<td>Aids</td>
<td></td>
<td>Pete</td>
<td>MedA</td>
</tr>
<tr>
<td>Mary</td>
<td>Cancer</td>
<td></td>
<td>Mary</td>
<td>MedB</td>
</tr>
</tbody>
</table>
Example

\[
\text{prior} = \{ \forall x \left( \text{Treat}(x, \text{MedA}) \rightarrow \text{Ill}(x, \text{Aids}) \lor \text{Ill}(x, \text{Cancer}) \right), \\
\forall x \left( \text{Treat}(x, \text{MedB}) \rightarrow \text{Ill}(x, \text{Cancer}) \lor \text{Ill}(x, \text{Flu}) \right) \}\]

\[
\text{pot\_sec} = \{ \exists x \text{Ill}(x, \text{Aids}), \exists x \text{Ill}(x, \text{Cancer}) \}\]

\[
\text{Neg}(\text{pot\_sec}) = \{ \forall x \neg \text{Ill}(x, \text{Aids}), \forall x \neg \text{Ill}(x, \text{Cancer}) \}\]

Constraint set \( C := \text{prior} \cup \text{Neg}(\text{pot\_sec}) \)

\[
\begin{array}{c|c|c|}
\text{Name} & \text{Diagnosis} \\
\hline
\text{Pete} & \text{Aids} \\
\text{Mary} & \text{Cancer} \\
\end{array}
\begin{array}{c|c|c|}
\text{Name} & \text{Treatment} \\
\hline
\text{Pete} & \text{MedA} \\
\text{Mary} & \text{MedB} \\
\end{array}
\]

\[
db\_\text{dist}(db_1) = 4
\]
Example

\[ prior = \{ \forall x (Treat(x, MedA) \rightarrow Ill(x, Aids) \vee Ill(x, Cancer)), \]
\[ \forall x (Treat(x, MedB) \rightarrow Ill(x, Cancer) \vee Ill(x, Flu)) \} \]

\[ pot\_sec = \{ \exists x Ill(x, Aids), \exists x Ill(x, Cancer) \} \]

\[ Neg(pot\_sec) = \{ \forall x \neg Ill(x, Aids), \forall x \neg Ill(x, Cancer) \} \]

Constraint set \( C := prior \cup Neg(pot\_sec) \)

<table>
<thead>
<tr>
<th>Ill</th>
<th>Name</th>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pete</td>
<td>Aids</td>
</tr>
<tr>
<td></td>
<td>Mary</td>
<td>Cancer</td>
</tr>
<tr>
<td></td>
<td>Mary</td>
<td>Flu</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Treat</th>
<th>Name</th>
<th>Treatment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pete</td>
<td>MedA</td>
</tr>
<tr>
<td></td>
<td>Mary</td>
<td>MedB</td>
</tr>
</tbody>
</table>

\[ db\_dist(db'_2) = db\_dist(db'_1) = 4 \]
Outline

1. Introduction

2. Automizing Inference-Proofness
   - Restricted Constraints
   - $preCQE$ Algorithm

3. Prototype

4. Conclusion
Undecidability of satisfiability problem

- For complete database, known potential secrets & data modification, find $db'$ such that $I^{db'} \models C$ and $db\_dist(db') \rightarrow \min$

- Undecidability of satisfiability problem for predicate logic (reproduced in (Börger et al, 2001))

- Identify syntactical restrictions for constraint set $C$ to make problem decidable

- “Allowed universal formulas” in prenex literal normal form
  - Subset of “allowed formulas” (Van Gelder/Topor, 1991)
  - “Active domain” semantics, $adom$: constants in $db$ and $C$
preCQE subprocedures

Branch-and-Bound depth-first search tree

- INIT
- GROUND
  - find relevant ground instantiations for universal quantifiers
- SIMP
- SPLIT
  - try two truth values for ground atom
- MARK
  - mark ground atom as k(empt), a(dded), r(emoved) or l(eft out)
Example: *preCQE* search tree

3 branches instead of $2^{10} = 1024$

At most 5 markers on a branch
Example: *preCQE* search tree

3 branches instead of $2^{10} = 1024$

At most 5 markers on a branch.
Example: $precQE$ search tree

3 branches instead of $2^{10} = 1024$.
At most 5 markers on a branch.
Example: *preCQE* search tree

3 branches instead of $2^{10} = 1024$

At most 5 markers on a branch

- **r:** *Ill* (Pete, Aids)
- **r:** *Ill* (Mary, Cancer)

**Splitting**

- **r:** *Treat* (Pete, MedA)

**Splitting**

- **r:** *Treat* (Mary, MedB)

- **k:** *Treat* (Pete, MedA)
- **a:** *Ill* (Pete, Cancer)

**PRUNE**

- **db**
- **db**

- **db**
- **db**
Example: \textit{preCQE} search tree

3 branches instead of $2^{10} = 1024$
At most 5 markers on a branch

```
Example:
p: 1
m: 10

r: \textit{Ill}(Pete, Aids)

r: \textit{Ill}(Mary, Cancer)

r: \textit{Treat}(Pete, MedA)

k: \textit{Treat}(Pete, MedA)
a: \textit{Ill}(Pete, Cancer)

r: \textit{Treat}(Mary, MedB)

k: \textit{Treat}(Mary, MedB)
a: \textit{Ill}(Mary, Flu)
```

\textbf{db}_1

\textbf{db}_2

PRUNE

Splitting

Splitting
Example: $preCQE$ search tree

3 branches instead of $2^{10} = 1024$
At most 5 markers on a branch

Splitting

$r: \text{Treat}(Pete, MedA)$
$k: \text{Treat}(Pete, MedA)$
$a: \text{Ill}(Pete, Cancer)$

Splitting

$r: \text{Treat}(Mary, MedB)$
$k: \text{Treat}(Mary, MedB)$
$a: \text{Ill}(Mary, Flu)$

PRUNE

$db_1'$

$db_2'$
Key results

**Theorem: Termination of preCQE**

For a set $C$ of allowed universal constraints, preCQE terminates in a finite amount of time.

**Proof:**

- At most $k$ different ground atoms with $\text{adom}$ constants where
  
  $$k := \sum_{P \in \mathcal{P}} \text{card}(\text{adom})^{\text{arity}(P)}$$

- At most $2^k$ branches in the search tree.
Key results

Theorem: Satisfiability soundness of $preCQE$
For a set $C$ of allowed universal constraints, if $db'$ is a database instance, it is inference-proof (hence, a model of $C$)

Proof:
- No violated constraints left

Corollary: Refutation completeness of $preCQE$
For a set $C$ of allowed universal constraints, if $C$ is unsatisfiable, $db'$ is undefined
Key results

**Theorem: Satisfiability soundness of \( \text{preCQE} \)**

For a set \( C \) of allowed universal constraints, if \( db' \) is a database instance, it is inference-proof (hence, a model of \( C \))

**Proof:**
- No violated constraints left

**Corollary: Refutation completeness of \( \text{preCQE} \)**

For a set \( C \) of allowed universal constraints, if \( C \) is unsatisfiable, \( db' \) is undefined
Key results

Theorem: Refutation soundness of \textit{preCQE}

For a set $C$ of allowed universal constraints, if $db'$ is undefined, then $C$ is unsatisfiable

- Not trivial because of efficiency of \textit{preCQE}
- Not all \textit{adom}-ground atoms explicitly handled
  1. Only violated constraints and affected ground atoms are considered
  2. If a truth assignment is unequivocal, ground atoms are marked directly without splitting
  3. Branches are pruned if a better solution has already been found
  4. Branches are pruned as soon as a conflict occurs
- \textit{preCQE} search tree in best case does not contain all possible $2^k$ branches
Proof of refutation soundness

Herbrand’s Theorem (Herbrand, 1930; here as in Cook/Nguyen, 2009)
Let $S$ be a set of closed universal formulas. Then $S$ is unsatisfiable iff some finite set $S_0$ of ground instances of formulas in $S$ is propositionally unsatisfiable

Herbrand’s Theorem with semantic tree (Chang/Lee, 1973; for clauses)
Let $S$ be a set of closed universal formulas. Then $S$ is unsatisfiable iff for some finite set $S_0$ of ground instances of formulas in $S$ there is a closed semantic tree

- Construct semantic tree out of $\text{preCQE}$ search tree
- Identify set $C_0$ of ground instances
- Show that semantic tree is closed for $C_0$
Key results

Corollary: Satisfiability completeness of \( \text{preCQE} \)
For a set \( C \) of allowed universal constraints, if \( C \) is satisfiable, \( db' \) is a database instance

Theorem: Distortion minimality of solution
For a set \( C \) of allowed universal constraints, if \( \text{preCQE} \) finds a solution \( db' \), then it is distortion-minimal

- For other constraints (existential or weakly acyclic fragments) similar:
  - Depth and width of \( \text{preCQE} \) search tree is bounded
  - Fix mapping of existentially quantified variables to invented constants for refutation soundness
Key results

Corollary: Satisfiability completeness of \( \text{preCQE} \)
For a set \( C \) of allowed universal constraints, if \( C \) is satisfiable, \( db' \) is a database instance

Theorem: Distortion minimality of solution
For a set \( C \) of allowed universal constraints, if \( \text{preCQE} \) finds a solution \( db' \), then it is distortion-minimal

- For other constraints (existential or weakly acyclic fragments) similar:
  - Depth and width of \( \text{preCQE} \) search tree is bounded
  - Fix mapping of existentially quantified variables to invented constants for refutation soundness
Key results

Corollary: Satisfiability completeness of \( \text{preCQE} \)

For a set \( C \) of allowed universal constraints, if \( C \) is satisfiable, \( db' \) is a database instance.

Theorem: Distortion minimality of solution

For a set \( C \) of allowed universal constraints, if \( \text{preCQE} \) finds a solution \( db' \), then it is distortion-minimal.

- For other constraints (existential or weakly acyclic fragments) similar:
  - Depth and width of \( \text{preCQE} \) search tree is bounded
  - Fix mapping of existentially quantified variables to invented constants for refutation soundness
Outline

1. Introduction
2. Automizing Inference-Proofness
3. Prototype
   - Implementation
   - Test Cases
4. Conclusion
User interface

![Image of user interface with COE Framework window]
## Average test results

<table>
<thead>
<tr>
<th>rep.</th>
<th>avg. msec total</th>
<th>avg. msec solver</th>
<th>dec. vars.</th>
<th>clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1930</td>
<td>184</td>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td>25</td>
<td>11974</td>
<td>3092</td>
<td>3000</td>
<td>3000</td>
</tr>
<tr>
<td>50</td>
<td>31304</td>
<td>6135</td>
<td>6000</td>
<td>6000</td>
</tr>
<tr>
<td>75</td>
<td>60459</td>
<td>8991</td>
<td>9000</td>
<td>9000</td>
</tr>
<tr>
<td>100</td>
<td>95792</td>
<td>8902</td>
<td>12000</td>
<td>12000</td>
</tr>
<tr>
<td>125</td>
<td>142843</td>
<td>11171</td>
<td>15000</td>
<td>15000</td>
</tr>
<tr>
<td>150</td>
<td>202067</td>
<td>16429</td>
<td>18000</td>
<td>18000</td>
</tr>
</tbody>
</table>
The graph shows the relationship between the number of repetitions per patient type and various metrics such as deviation, total runtime, and solver runtime. As the number of repetitions increases, so do the deviation, total runtime, and solver runtime, indicating a linear increase in these metrics with respect to the number of repetitions.
Outline

1 Introduction
2 Automizing Inference-Proofness
3 Prototype
4 Conclusion
Achievements

- Consistency with *prior*, confidentiality of *potsec*, maximal availability of unmodified tuples with *dbdist*
- Unique combination of model generation and distance minimization in infinite domain
- Quantifier handling without a need to expand them into ground conjunctions or disjunctions
- Only minor restrictions on the syntax of constraint formulas
- Both addition and deletion of tuples as modification primitives
- Optimized for complete databases with efficient query evaluation function
- Output in form of complete database instance
- Termination, soundness and completeness for appropriate fragments