Efficient Inference Control for Open Relational Database Queries

Jan-Hendrik Lochner

Faculty of Computer Science · TU Dortmund University

Mini-Workshop TCS, July 14, 2010
J. Biskup, S. Hartmann, S. Link, J.-H. Lochner:

Efficient Inference Control for Open Relational Queries.

Agenda

1. Introduction and Motivation
2. Stateless CQE for Closed Queries
3. Stateless CQE for Open Queries
4. Conclusion
1. Introduction and Motivation

2. Stateless CQE for Closed Queries

3. Stateless CQE for Open Queries

4. Conclusion
Security

Interests
- Authenticity
- Availability
- Confidentiality
- Integrity
- Non-Repudiation
- ...

More specifically
- User should be able to get the information that is necessary to complete his/her task
- Information that has been declared confidential should not be disclosed to the user
Inferences in Relational Databases

Database

- schema: \( \langle ACC, U, \Sigma \rangle \) with
  - \( U = \{ \text{bank}, \text{acc\_no}, \text{acc\_holder}, \text{balance} \} \) and
  - \( \Sigma = \{ \{ \text{bank}, \text{acc\_no} \} \rightarrow \{ \text{acc\_holder}, \text{balance} \} \} \)

- instance:

<table>
<thead>
<tr>
<th>acc</th>
<th>bank</th>
<th>acc_no</th>
<th>acc_holder</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bank A</td>
<td>101</td>
<td>Smith</td>
<td>1,000</td>
</tr>
<tr>
<td></td>
<td>Bank A</td>
<td>102</td>
<td>Jones</td>
<td>2,500</td>
</tr>
<tr>
<td></td>
<td>Bank A</td>
<td>103</td>
<td>Smith</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Bank B</td>
<td>101</td>
<td>Anderson</td>
<td>1,500</td>
</tr>
<tr>
<td></td>
<td>Bank B</td>
<td>105</td>
<td>Brown</td>
<td>1,000</td>
</tr>
<tr>
<td></td>
<td>Bank C</td>
<td>201</td>
<td>Smith</td>
<td>50</td>
</tr>
</tbody>
</table>

← confidential!
Inferences in Relational Databases (cont’d)

Queries

- $\Phi_1 \equiv (\exists X) ACC(\text{bankA, 102, jones}, X)$ [true in acc]
- $\Phi_2 \equiv (\exists Y) ACC(\text{bankA, 102, Y, 2500})$ [true in acc]
Inferences in Relational Databases (cont’d)

Queries

- \( \Phi_1 \equiv (\exists X) \text{ACC}(\text{bankA}, 102, jones, X) \) [true in acc]
- \( \Phi_2 \equiv (\exists Y) \text{ACC}(\text{bankA}, 102, Y, 2500) \) [true in acc]

\[ \downarrow \]

[ \text{bank, acc_no } \rightarrow \text{acc_holder, balance} \]

Inference

- \( \text{ACC}(\text{bankA}, 102, jones, 2500) \) is true in acc
Agenda

1. Introduction and Motivation
2. Stateless CQE for Closed Queries
3. Stateless CQE for Open Queries
4. Conclusion
Confidentiality is preserved for declared potential secrets even if the user is capable of logical reasoning and exploits his a priori knowledge.

Potential Secrets
- Declared by security admin. (independent of db instance)
- Must be kept confidential \( \text{iff} \) they are true in the db instance

Definition: Preservation of Confidentiality (informal)

A CQE is *confidentiality preserving* if

for every query sequence and for every declared secret there exists a database instance such that

1. the database instance “matches” the answers to the queries;
2. the declared secret is *false* in the database instance.
 Controlled Query Evaluation
 Stateful Version

Stateful CQE preserves confidentiality.
Theorem prover is needed to implement censor.

- **query**
  - ordinary query evaluation
    - correct result
      - censor
        - correct result + modification request
          - modifier
            - (possibly distorted) answer
Controlled Query Evaluation
Stateless Version

Query

ordinary query evaluation

query

r

pot_sec

ordinary query evaluation

correct result

censor

correct result + modification request

modifier

(possibly refused) answer

Stateless CQE efficiently preserves confidentiality under certain assumptions.
Assumptions for Stateless CQE

Schema

- Object Normal Form (ONF):
  1. Unique key $\mathcal{K}$
  2. Left-hand sides of FDs are superset of $\mathcal{K}$ (Boyce-Codd NF)
- Usually satisfied in “real-world databases”
**Assumptions for Stateless CQE (cont’d)**

**Query Language**
- Closed *select-project* queries:
  
  \[(\exists X)\text{ACC}(bankA, 102, jones, X)\]

- Restricted usage of logical connectives

**Policy Language**
- Secret: “select part” must be subset of \(\mathcal{K} \cup \{N\}\) with \(N \in \mathcal{U} \backslash \mathcal{K}\)
- For example:
  - \(\mathcal{K} = \{\text{bank, acc_no}\}\)
  - \(N = \text{acc_holder}\) or \(N = \text{balance}\)
  - “Select part” of secret must be subset of
    - \(\{\text{bank, acc_no, acc_holder}\}\) or \(\{\text{bank, acc_no, balance}\}\)
  - \(\Psi \equiv (\exists X)(\exists Y)\text{ACC}(bankA, X, smith, Y)\)
Agenda

1. Introduction and Motivation
2. Stateless CQE for Closed Queries
3. Stateless CQE for Open Queries
4. Conclusion
**Ordinary Evaluation of Open Queries**

**Open Queries**
- Query contains free variable(s)
- \[ \text{eval}^*(\Phi(\vec{V}))(r) = \{ \Phi(\vec{c}) | \vec{c} \in \text{Const}^k, \ r \models M \Phi(\vec{c}) \} \]

*variable assignments... that make \( \Phi(\vec{V}) \) true*

**Example**
- \( \Phi(Z) \equiv (\exists X)(\exists Y)\text{ACC}(\text{bankB}, X, Z, Y) \)
  
  ["Account holders of Bank B"]

- \[ \text{eval}^*(\Phi(Z))(\text{acc}) = \{ \]
  \( (\exists X)(\exists Y)\text{ACC}(\text{bankB}, X, \text{anderson}, Y), \]
  \( (\exists X)(\exists Y)\text{ACC}(\text{bankB}, X, \text{brown}, Y) \} \)
Stateless CQE for Open Queries: First Approach

Idea

Treat each element of the answer set to a query like a closed query. Filter “harmful” elements from answer set.

Example

Query: $\Phi(Z) \equiv (\exists X)(\exists Y)ACC(bankB, X, Z, Y)$

Secret: $\Psi \equiv (\exists X)(\exists Y)(\exists Z)ACC(X, Y, brown, Z)$

“Harmful”: $((\exists X)(\exists Y)ACC(bankB, X, brown, Y) \models \Psi)$

$refuse(\Phi(Z), \{\Psi\}) : \{(\exists X)(\exists Y)ACC(bankB, X, brown, Y)\}$

Answer: $ans = eval^*(\Phi(Z))(r) \setminus refuse(\Phi(Z), \{\Psi\})$

$= \{(\exists X)(\exists Y)ACC(bankB, X, anderson, Y)\}$
Harmful Inferences: User Knowledge

Completeness
Considering a query $\Phi$, each variable assignment $\alpha$ either
1. makes $\Phi$ false or
2. makes $\Phi$ true and is part of the contr. answer or
3. makes $\Phi$ true and is not part of the contr. answer.

Policy Awareness
Considering a query $\Phi$, for each variable assignment $\alpha$ it holds that
- if $\alpha$ makes $\Phi$ true and is not part of the contr. answer
- then $\alpha(\Phi) \models \Psi$ for a $\Psi \in pot\_sec$. 
Harmful Inferences: User Knowledge

Completeness
Considering a query $\Phi$, each variable assignment $\alpha$ either

1. makes $\Phi$ false or
2. makes $\Phi$ true and is part of the contr. answer or
3. makes $\Phi$ true and is not part of the contr. answer.

Policy Awareness
Considering a query $\Phi$, for each variable assignment $\alpha$ it holds that

- if $\alpha$ makes $\Phi$ true and is not part of the contr. answer
- then $\alpha(\Phi) \models \Psi$ for a $\Psi \in \text{pot}_\text{sec}$. 
## Harmful Inferences: Example

### Setting

- **Schema:** $RS = \langle R, A, \emptyset \rangle$
- **Instance:** $r = \{ R(a) \}$
- **Policy:** $pot_{sec} = \{ R(a) \}$
- **Queries:**
  - $\Phi_1(X) \equiv R(X)$ (all tuples)
  - $\Phi_2 \equiv (\exists X) R(X)$ (existence of tuples)

- $ans_1 = \emptyset$
- $ans_2 = \{ (\exists X) R(X) \}$

### Completeness Information ($\Phi_1$)

$$(\forall X)[\neg R(X) \lor (R(X) \land \bigvee_{R(c) \in ans_1} X = c) \lor (R(X) \land \bigwedge_{R(c) \in ans_1} X \neq c)]$$

### Policy Awareness Information ($\Phi_1$)

$$(\forall X)[(R(X) \land \bigwedge_{R(c) \in ans_1} X \neq c) \implies X = a]$$
Harmful Inferences: Example

Setting

- schema: $RS = \langle R, A, \emptyset \rangle$; instance: $r = \{R(a)\}$
- policy: $pot\_sec = \{R(a)\}$
- queries: $\Phi_1(X) \equiv R(X)$
  - all tuples
  - $ans_1 = \emptyset$

$\Phi_2 \equiv (\exists X)R(X)$
  - existence of tuples
  - $ans_2 = \{(\exists X)R(X)\}$

Completeness Information ($\Phi_1$)

$$(\forall X)[\neg R(X) \lor (R(X) \land \lor_{R(c)\in ans_1} X = c) \lor (R(X) \land \land_{R(c)\in ans_1} X \neq c)]$$

$$\equiv (\forall X)[R(X) \implies (\lor_{R(c)\in ans_1} X = c \lor \land_{R(c)\in ans_1} X \neq c)]$$

Policy Awareness Information ($\Phi_1$)

$$(\forall X)[(R(X) \land \land_{R(c)\in ans_1} X \neq c) \implies X = a]$$
Harmful Inferences: Example

Setting

- schema: $RS = \langle R, A, \emptyset \rangle$; instance: $r = \{R(a)\}$
- policy: $pot\_sec = \{R(a)\}$
- queries: $\Phi_1(X) \equiv R(X)$ (all tuples) $\Phi_2 \equiv (\exists X)R(X)$ (existence of tuples)

$ans_1 = \emptyset$ $ans_2 = \{(\exists X)R(X)\}$

Completeness Information ($\Phi_1$)

$(\forall X)[\neg R(X) \lor (R(X) \land \bigvee_{R(c) \in ans_1} X = c) \lor (R(X) \land \bigwedge_{R(c) \in ans_1} X \neq c)]$

$\equiv (\forall X)[R(X) \implies (\bigvee_{R(c) \in ans_1} X = c) \lor \bigwedge_{R(c) \in ans_1} X \neq c)]$

Policy Awareness Information ($\Phi_1$)

$(\forall X)[(R(X) \land \bigwedge_{R(c) \in ans_1} X \neq c) \implies X = a]$
Harmful Inferences: Example

Setting

- **schema**: $RS = \langle R, A, \emptyset \rangle$; instance: $r = \{R(a)\}$
- **policy**: $pot_{sec} = \{R(a)\}$
- **queries**: 
  - $\Phi_1(X) \equiv R(X)$
  - $\Phi_2 \equiv (\exists X)R(X)$

Completeness Information ($\Phi_1$)

$$
(\forall X)[\neg R(X) \lor (R(X) \land \bigvee_{R(c) \in ans_1} X = c) \lor (R(X) \land \bigwedge_{R(c) \in ans_1} X \neq c)]
$$

$$
\equiv (\forall X)[R(X) \Rightarrow (\bigvee_{R(c) \in ans_1} X = c \lor \bigwedge_{R(c) \in ans_1} X \neq c)]
$$

Policy Awareness Information ($\Phi_1$)

$$
(\forall X)[(R(X) \land \bigwedge_{R(c) \in ans_1} X \neq c) \Rightarrow X = a]
$$
Harmful Inferences: Example

**Setting**

- **schema:** \( RS = \langle R, A, \emptyset \rangle \);
- **instance:** \( r = \{ R(a) \} \)
- **policy:** \( pot_{sec} = \{ R(a) \} \)
- **queries:**
  
  - \( \Phi_1(X) \equiv R(X) \) (all tuples)
  - \( \Phi_2 \equiv (\exists X)R(X) \) (existence of tuples)

\[
\begin{align*}
\text{ans}_1 &= \emptyset \\
\text{ans}_2 &= \{ (\exists X)R(X) \}
\end{align*}
\]

**Completeness Information (\( \Phi_1 \))**

\[
(\forall X)\left[ \neg R(X) \lor (R(X) \land \bigvee_{R(c)\in\text{ans}_1} X = c) \lor (R(X) \land \bigwedge_{R(c)\in\text{ans}_1} X \neq c) \right] \\
\equiv (\forall X)[R(X) \Rightarrow (\bigvee_{R(c)\in\text{ans}_1} X = c \lor \bigwedge_{R(c)\in\text{ans}_1} X \neq c)]
\]

**Policy Awareness Information (\( \Phi_1 \))**

\[
(\forall X)[(R(X) \land \bigwedge_{R(c)\in\text{ans}_1} X \neq c) \Rightarrow X = a]
\]
Theorem

Stateless CQE for open queries, i.e., filtering of “harmful” elements from the answer set, preserves confidentiality if the user is not aware of the policy.

Complexity of Implementation

Basically $O(k \cdot \log(m))$ for controlling a single query with

- $k$: size of answer set
- $m$: size of policy
1. Introduction and Motivation
2. Stateless CQE for Closed Queries
3. Stateless CQE for Open Queries
4. Conclusion
Summary

- We proposed stateless CQE for open relational queries
- We showed that there is no straightforward transition from closed to open queries
- Hiding the policy from the user is sufficient to regain preservation of confidentiality
- We developed an efficient algorithm for enforcing stateless CQE for open queries

Future/Ongoing Research

- Transfer enhancements for closed queries to open queries
- Availability-oriented query answering
- Comprehensive inference control algorithm
Questions?