Information Systems ■ (■■■■) ■■■–■■■

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Inference control of open relational queries under closed-world semantics based on theorem proving ☆

Joachim Biskup *, Martin Bring, Michael Bulinski

Fakultät für Informatik, Technische Universität Dortmund, 44221 Dortmund, Germany

ARTICLE INFO

Article history:
Received 9 February 2016
Received in revised form
15 June 2016
Accepted 22 July 2016

Keywords:
Active domain
A priori knowledge
Binary search
Closed-world assumption
Combined lying and refusal
Confidentiality policy
Constant symbol
Completeness sentence
Data publishing
Diagonalization function
Dictionary
Enumeration of a domain
Free variable
Inference control
Lying
Open query
Pairing function
Refusal
Theorem proving

ABSTRACT

Relational database systems may serve to evaluate an open query under closed-world semantics. The evaluation returns an explicit output relation complemented with an often implicit statement about the completeness of that relation. The output relation is formed from all those tuples that both fit the format and satisfy the properties expressed in the query. Using first-order logic for specifying formal semantics, the output relation can be seen as a set of (ground) sentences obtained from the query formula by suitable substitutions of free variables by constants. A statement about the completeness of a relation can also explicitly be formalized as a sentence of first-order logic. Inference control for enforcing a confidentiality policy has to inspect and to possibly distort not only the sentences representing the tuples of the output relation but also the completeness sentences. Previously designed and formally verified control procedures employ theorem-proving for such inspections while iteratively considering candidates for those sentences and determining termination conditions, respectively. In this article, we outline an implementation of these control procedures and treat improvements of their runtime efficiency, in particular to overcome shortcomings of the underlying theorem prover, which is repeatedly called with an input comprising a completeness sentence of increasing size. The improvements are obtained by an equivalent rewriting of completeness sentences, exploiting the active domain or introducing new constants for combinations of the original constants, respectively, as well as by optimizing the number of such calls. Besides theoretical complexity considerations, we also present practical evaluations for some examples. These examples include queries that—without control—would return the whole underlying database relations and—with control—can be used for confidentiality-preserving data publishing.

© 2016 Elsevier Ltd. All rights reserved.

UNCORRECTED PROOF

1. Introduction

Inference control for information systems in general and relational databases in particular is a mechanism to confine the information content and thus the usability of data made accessible to a client to whom some piece(s) of information should be kept confidential, see, e.g., [1–3]. Thus inference control aims at protecting *information* rather than just the underlying *data*, as achieved by traditional access control or simple encryption. Though

☆ This work has been partially supported by the Deutsche Forschungsgemeinschaft (German Research Council) under grant no. BI 311/12-2.

* Corresponding author.
E-mail addresses: joachim.biskup@cs.tu-dortmund.de (J. Biskup), martin.bring@cs.tu-dortmund.de (M. Bring), michael.bulinski@cs.tu-dortmund.de (M. Bulinski).

http://dx.doi.org/10.1016/j.is.2016.07.008
0306-4379/© 2016 Elsevier Ltd. All rights reserved.

Please cite this article as: J. Biskup, et al., Inference control of open relational queries under closed-world semantics based on theorem proving, Information Systems (2016), http://dx.doi.org/10.1016/j.is.2016.07.008

protection of information is a crucial requirement for many applications, the actual enforcement is facing great challenges arising from conceptual and computational problems.

In this work, we focus on the problems arising from controlling *open queries* to a *relational database*, as managed by well-known products of a DBMS complying with the SQL-standard. Basically, given a database schema and corresponding relation instances (sets of tuples) an open query requests to return an answer relation (set of tuples) that contains exactly those tuples that both fit the format and satisfy the properties expressed in the query. Notably, all tuples fitting the format but not satisfying the properties are not explicitly returned. Rather, the issuer of the query and receiver of the answer is assumed to apply a *closed-world assumption*, which says that each format-fitting tuple not contained in the answer does not satisfy the requested properties. Under the assumption that infinite type extensions (domains) are declared by the schema, there are infinitely many such "negative tuples". Accordingly, controlling an open query necessarily has to identify and, as far as requested by a confidentiality policy, to confine the information supplied by all these tuples.

We will treat the problems raised in a formal approach to relational databases based on first-order logic with DB-semantics [4–6], where

- a stored finite database instance is treated as an Herbrand-like semantic model,
- tuples are seen as ground atoms,
- queries are expressed by formulas which may contain free variables ranging over a fixed infinite domain of constant symbols,
- semantic constraints and other a priori knowledge are specified by sentences, and
- potential secrets, i.e., elements of a confidentiality policy, are declared by sentences as well.

Example 1. Consider a database *db* with a relation *ill* relating patients to illnesses as present for example in hospitals:

$db := \{ill(Smith, cancer),\ ill(Miller, flu),\ ill(Miller, rheumatism)\}.$

Let *psec* be the confidentiality policy stating that the information of person *Smith* suffering from *cancer* should be kept confidential:

$psec := \{ill(Smith, cancer)\}.$

Additionally the requestor is assumed to have the knowledge *prior* that person *Smith* or person *Miller* is actually suffering from *cancer*:

$prior := \{ill(Smith, cancer) \vee ill(Miller, cancer)\}.$

Assuming the requestor is interested in the illnesses of person *Miller*, he submits the following open query with a free variable *x* to the information system:

$ill(Miller, x).$

The answer relation of this query would then consist of the two tuples $ill(Miller, flu)$ and $ill(Miller, rheumatism)$. Applying the closed world assumption yields, among others, the "negative tuple" $\neg ill(Miller, cancer)$ which enables the requestor to infer the confidential information of person *Smith* suffering from *cancer*, as formally captured by the following entailment:

$$(ill(Smith, cancer) \vee ill(Miller, cancer)) \wedge \neg ill(Miller, cancer) \models ill(Smith, cancer).$$

Elaborated as part of a specific approach to inference control called Controlled Interaction Execution, see [7–11] for an introduction, Biskup/Bonatti [6] proposed and verified control procedures for open relational queries within a dedicated logic-oriented relational model dealing with different settings of a reaction on detecting harmful information, including *refusal*, *lying* and the *combination* thereof. As already indicated above, suitably *representing* and *handling* the pertinent closed-world assumptions are a most crucial aspect.

Representation is enabled by expressing the information content of the infinitely many "*negative tuples*" by a single *completeness sentence* in first-order logic. Conceptually, handling of completeness sentences is managed in two ways, either in advance by determining a suitable bound for the set of tuples to be explicitly inspected for inclusion into the controlled answer or repeatedly while inspecting tuples for inclusion one after another until the pertinent completeness sentence for the overall answer becomes true, i.e., all remaining tuples are guaranteed to not satisfying the pertinent properties. For both ways, not only the set of "*positive tuples*" but also the corresponding completeness sentence are explicitly returned to the requestor, and memorized by the control system.

Example 2. As seen in Example 1, the information $\neg ill(Miller, cancer)$ has to be restricted in order to avoid an information flow violating the confidentiality policy. This may lead to the following result of an explicit finite answer relation together with a suitable completeness sentence:

$answer := \{ill(Miller, flu),\ ill(Miller, rheumatism)\}$

$Complete: \equiv \forall x\ [(x \neq flu \wedge x \neq rheumatism \wedge x \neq cancer) \Rightarrow \neg ill(Miller, x)].$

Algorithmic handling of completeness sentences, however, turned out to be a major obstacle to achieve *efficient* and *scalable* controlled query evaluation. To confine information, possible inferences revealing confidential information have to be detected by the control system by employing a *theorem prover*. The difficulties in handling completeness sentences arise in the internal treatment of completeness sentences by theorem provers. In this report we present a detailed description of these difficulties together with some approaches to overcome them. More specifically, we will

- summarize and explain the basic control procedures for open queries, emphasizing the lying approach and the approach of combining refusal and lying, as designed and verified regarding preservation of confidentiality in previous theoretical work (Section 2);
- outline an implementation of those control procedures including some details concerning the enumeration of

UNCORRECTED PROOF

Please cite this article as: J. Biskup, et al., Inference control of open relational queries under closed-world semantics based on theorem proving, Information Systems (2016), http://dx.doi.org/10.1016/j.is.2016.07.008

Download English Version:

https://daneshyari.com/en/article/4945057

Download Persian Version:

https://daneshyari.com/article/4945057

Daneshyari.com