

# Chasing after Secrets in Relational Databases

**Joachim Biskup, Jan-Hendrik Lochner**

*Faculty of Informatics  
Dortmund University of Technology  
Germany*

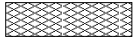
**Sven Hartmann**

*School of Informatics  
Clausthal University of Technology  
Germany*

**Sebastian Link**

*School of Information Management  
Victoria University of Wellington  
New Zealand*



Theorem	Proof
$P \neq NP$	



## Outline

- ▶ Access and Inference Control
- ▶ Controlled Query Evaluation
- ▶ Necessity of Inference Control
- ▶ Examples
- ▶ Conclusion and Future Work

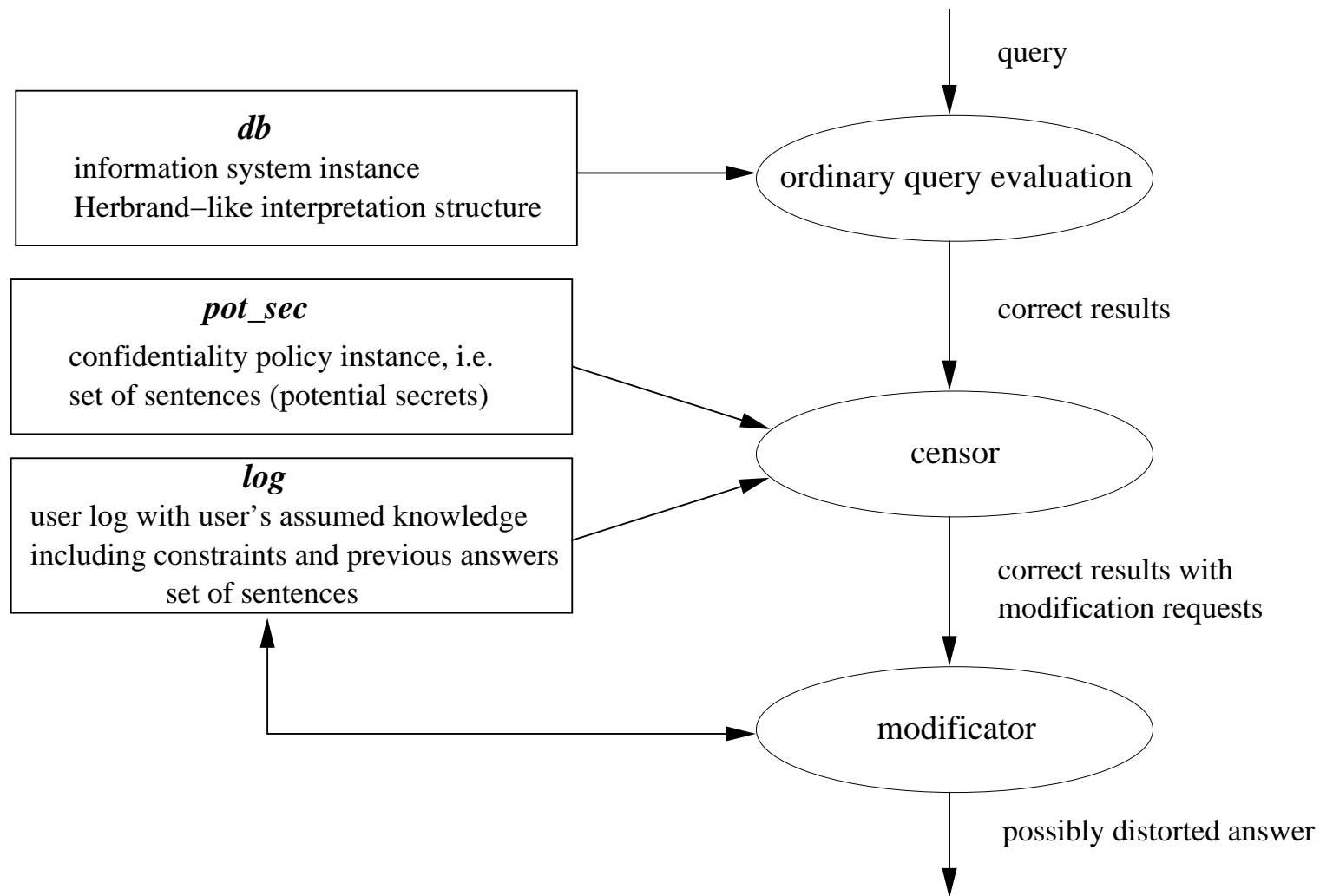


## Access and Inference Control

- ▶ data owners share some of their private data while hiding others
- ▶ balance conflicting security interests
  - ↳ confidentiality: prohibitions for access beyond intended usage
  - ↳ availability: permissions for requested resources as needed
- ▶ *Access control*:
  - ↳ single query answers are granted or rejected
  - ↳ procedurally decided by inspecting access privileges
  - ↳ efficiently implementable in “real-time”
- ▶ *Inference control*:
  - ↳ answer sequence stepwise censored and modified potentially
  - ↳ algorithmically evaluated by deciding implication problems
  - ↳ in general of high computational complexity
- ▶ approach to inference control: *Controlled Query Evaluation*



# Controlled Query Evaluation



## An Example

- ▶ relation schema **EMPLOYEE** with attributes  $Id, Name, Salary$   
↪ and FD:  $ID \rightarrow Name, Salary$ , i.e.,  $ID$  is the unique key
- ▶ potential secret  $\Psi = (\exists X_{ID})\text{EMPLOYEE}(X_{ID}, \text{Steve Jobs}, 500\text{K})$   
↪  $(0001, \text{Steve Jobs}, 500\text{K})$  belongs to **EMPLOYEE**-table
- ▶ user's query sequence:  
↪  $\Phi_1 = (\exists X_N)\text{EMPLOYEE}(0001, X_N, 500\text{K})$   
↪  $\Phi_2 = (\exists X_S)\text{EMPLOYEE}(0001, \text{Steve Jobs}, X_S)$
- ▶ correct answers to both queries and key property would reveal  $\Psi$
- ▶ censor:  
↪ logs correct answer to first query and  
↪ refuses correct answer to second query



## How the Refusal Censor Works

- ▶ tentative system behavior for  $\{\Phi, \Phi \Rightarrow \Psi\}$  *implies*  $\Psi$ 
  - ↳ if correct answer  $\Phi$ , then refuse (return *mum*)
  - ↳ if correct answer  $\neg\Phi$ , then return  $\neg\Phi$
- ▶ “informed users” may use meta-inference:
  - ↳ the system refuses,
  - ↳ this happens only, if the correct answer is  $\Phi$
  - ↳ consequently,  $\Phi$  is true indeed
- ▶ to avoid meta-inferences: refuse in both cases
- ▶ censor inspects if the following together imply a potential secret:
  - ▶ the a-priori knowledge
  - ▶ the answers to previous queries
  - ▶ either the correct query answer or the negated query answer



## CQE is secure in the following sense

- ▶ focus here: existential- $R$ -sentences (closed select-project queries)
  - ↳ as query language and confidentiality policy language
- ▶ For a every finite prefix  $Q'$  of a query sequence  $Q = \langle \Phi_1, \Phi_2, \dots \rangle$  we find that
  - ▶ for every potential secret  $\Psi$ ,
  - ▶ for every  $R$ -table  $r_1$ ,
  - ▶ for every appropriate a-priori knowledge  $log_0$ ,
  - ▶ there is some  $R$ -table  $r_2$  that
    - ▶ satisfies  $log_0$ ,
    - ▶ gives the same controlled answers to  $Q'$  as  $r_1$ , and
    - ▶ does not satisfy the potential secret  $\Psi$
- ▶  $r_1$  (where  $\Psi$  may be true) and  $r_2$  (where  $\Psi$  is false) indistinguishable



## Drawbacks of Inference Control

- ▶ log file with previous query answers costly to maintain at run-time
- ▶ implication problem (with the log file as input) can be computationally hard or even undecidable
- ▶ some authors seriously suggest not to keep a user log
  - ▶ Stonebraker, Wong: Access control in a relational data base management system by query modification, *ACM/CSC-ER Annual Conference*
  - ▶ Huey, et al: Oracle Database Security Guide, 11g Release 1 (Virtual Private Database)
- ▶ example above shows that confidentiality cannot always be guaranteed without a log file
  - ▶ *When can confidentiality be guaranteed without a user log?*
  - ▶ *Can we do something about the user log otherwise?*
  - ▶ *When exactly does inference control become necessary?*



## Natural Access Control

- ▶ Idea: check if user query  $\Phi$  *directly implies* some potential secret  $\Psi$
- ▶ can be decided efficiently for select-project queries  $\Phi, \Psi$ 
  - ▶ by simple pattern matching
  - ▶  $\Phi \models \Psi$  iff each constant  $c$  in  $\Psi$  appears at the same position in  $\Phi$
  - ▶  $(\exists X_S)\text{EMPLOYEE}(0001, \text{Steve Jobs}, X_S) \models (\exists X_{ID})(\exists X_S)\text{EMPLOYEE}(X_{ID}, \text{Steve Jobs}, X_S)$
- ▶ reduces costly inference control to efficient *natural access control*
  - ▶ no user log nor theorem prover needed
  - ▶ highly efficient optimization for controlled query evaluation that preserves confidentiality



## Showcases for Natural Access Control

- ▶ Showcase 1:
  - ▶ DDL: schemata with FDs in Object Normal Form (ONF)
  - ▶ CPL: select-project queries covering schema facts
  - ▶ QL: select-project queries (existential- $R$ -sentences)
- ▶ Showcase 2:
  - ▶ DDL: arbitrary schemata with FDs
  - ▶ CPL: select-project queries (existential- $R$ -sentences)
  - ▶ QL: select queries ( $R$ -sentences)
- ▶ dropping any single restriction results in violation of confidentiality
- ▶ concern: limits availability of data drastically
  - ↳ for instance, neither showcase applies to our example



## Characterizing Violations by Forbidden Structures

- ▶ any exhibited violation constitutes a *forbidden structure*
  - ▶ user must not learn both (0001, Steve Jobs) and (0001, 500K), if (Steve Jobs, 500K) is secret
  - ▶ user must not learn both (*Id*, *Name*)- and (*Id*, *Salary*)-values if the respective (*Name*, *Salary*)-combination is secret
- ▶ idea: explore forbidden structures as *inference signature*:
  - ▶ identify forbidden structures from schema and potential secrets at *declaration time* and compile into inference signature
  - ▶ express them in terms of templates that are implied by  $\Sigma$
  - ▶ monitor user behavior at *run time* to check whether forbidden structure arises and refuse answer before last step
- ▶ neither log nor censor are needed in full generality to detect forbidden structures



## Template Dependencies

- ▶ a *template dependency* (TD) expression  $TD[h_1, \dots, h_l \mid c]$  where
  - ▶  $h_1, \dots, h_l$  are the hypothesis rows
  - ▶  $c$  is the conclusion row
  - ▶ each row consists of  $n$  abstract symbols (one per attribute)
  - ▶ symbols may occur more than once, but not for different attributes
- ▶ for two rows  $t_1, t_2$  the *agree set*  $ag(t_1, t_2) := \{A \mid t_1(A) = t_2(A)\}$
- ▶  $r$  satisfies  $TD[h_1, \dots, h_l \mid c]$  if whenever
  - $r$  contains  $t_1, \dots, t_l$  with  $ag(h_i, h_j) \subseteq ag(t_i, t_j)$  for  $1 \leq i < j \leq l$
  - then  $r$  contains a tuple  $t$  with  $ag(h_i, c) \subseteq ag(t_i, t)$  for  $1 \leq i \leq l$
- ▶  $TD[h_1, \dots, h_l \mid c]$  trivial iff  $c$  obtainable from some  $h_i$  by weakening



## Our Example Re-Considered

- ▶ for our example above, a “forbidden structure” is encoded in the TD

$$\frac{a_{ID} a_N b_S}{a_{ID} b_N a_S} \\ a_{ID} a_N a_S$$

- ▶ encoding the violation of confidentiality:
  - ↪ non-trivial TD implied by  $ID \rightarrow Name, Salary$
  - ↪ answers to queries  $\Phi_1$  and  $\Phi_2$  result in mapping  $a_{ID} \mapsto 0001, a_N \mapsto Steve\ Jobs, a_S \mapsto \$500K$  that instantiate hypotheses
  - ↪ conclusion instantiated with potential secret ( $Steve\ Jobs, \$500K$ )
- ▶ violation of confidentiality occurs precisely when there is a potential secret that results from chasing previous query answers and a non-refused answer by the declared data dependencies



## Forbidden Structure, sufficient for Violation

- ▶ assumptions:
  - ▶  $RS = (R, \mathcal{U}, \Sigma)$  where  $\Sigma$  consists of FDs and full join dependencies
  - ▶ a non-trivial  $TD[h_1, \dots, h_l \mid c]$  implied by  $\Sigma$
- ▶ then there exist
  - ▶ a potential secret  $\Psi \in \mathcal{L}_Q$  with schema  $\mathcal{P} = \cup_{j=1}^l ag(h_j, c)$ ,
  - ▶ an instance  $r$  of schema  $RS$ , and
  - ▶ queries  $\Phi_1, \dots, \Phi_l \in \mathcal{L}_Q$  with schemes  $\mathcal{F}_i$  where  $ag(h_i, h_j) \subseteq \mathcal{F}_i$  for all  $j \neq i$
- ▶ such that
  - ▶ all queries are permitted under natural access control, i.e.,  $\Phi_i \not\equiv \Psi$ ,
  - ▶ all queries are true in the instance  $r$ , i.e.,  $r \models \Phi_i$ , and
  - ▶ the answers are violating, i.e.,  $\Sigma \cup \{\Phi_1, \dots, \Phi_l\} \models \Psi$



## Forbidden Structure, necessary for Violation

- ▶ assumptions:
  - ▶ relation schema  $R$  with a set  $\Sigma$  of FDs and full join dependencies
  - ▶ potential secret  $\Psi \in \mathcal{L}_Q$  with scheme  $\mathcal{P}$ ,
  - ▶ instance  $r$  of schema  $RS$ ,
  - ▶ queries  $\Phi_1, \dots, \Phi_l \in \mathcal{L}_Q$  with schemes  $\mathcal{F}_i$
- ▶ such that
  - ▶ all queries are permitted under natural access control, i.e.,  $\Phi_i \not\models \Psi$ ,
  - ▶ all queries are true in the instance  $r$
  - ▶ the answers are violating, i.e.,  $\Sigma \cup \{\Phi_1, \dots, \Phi_l\} \models \Psi$
- ▶ then there is a non-trivial  $TD[h_1, \dots, h_l \mid c]$  implied by  $\Sigma$  such that
  - ▶  $\mathcal{P} = \bigcup_{j=1}^l ag(h_j, c)$
  - ▶  $ag(h_j, h_i) \subseteq \mathcal{F}_i$  for all  $j \neq i$



## Another Example

- ▶ Consider the relation schemas  $\langle R_1, \mathcal{U}_1, \Sigma_1 \rangle$  and  $\langle R_2, \mathcal{U}_2, \Sigma_2 \rangle$  over

$$\mathcal{U}_1 = \{S(\text{ymptom}), M(\text{ethod\_of\_Examination})\},$$

$$\mathcal{U}_2 = \{S, D(\text{iagnosis}), P(\text{atient})\}.$$

- ▶ GPs see view  $V$ : `SELECT * FROM R1, R2 WHERE R1.S = R2.S`

↳ MVDs  $S \twoheadrightarrow M$  and  $M \twoheadrightarrow D$  hold

- ▶ GPs only allowed to see diagnosis for own patients

↳  $\Psi = (\exists X_s)(\exists X_m)V(X_s, X_m, \text{Cancer}, \text{Smith})$  potential secret

- ▶ queries

▶  $\Phi_1 = (\exists X_m)(\exists X_p)V(\text{Fever}, X_m, \text{Cancer}, X_p),$

▶  $\Phi_2 = (\exists X_d)V(\text{Fever}, \text{Xray}, X_d, \text{Smith})$

with schemes  $F_1 = \{S, D\}$  and  $F_2 = \{S, M, P\}$ , respectively



## Example Continued

- ▶  $\Psi$  can still be inferred:
  - ↳ chase  $V(\text{Fever}, X_m, \text{Cancer}, X_p) \ \& \ V(\text{Fever}, X_{\text{ray}}, X_d, \text{Smith})$  by  $S \twoheadrightarrow M$
  - ↳ leads to  $V(\text{Fever}, X_m, X_d, \text{Smith})$  and  $V(\text{Fever}, X_{\text{ray}}, \text{Cancer}, X_p)$
  - ↳ chase  $V(\text{Fever}, X_{\text{ray}}, X_d, \text{Smith}) \ \& \ V(\text{Fever}, X_{\text{ray}}, \text{Cancer}, X_p)$  by  $M \twoheadrightarrow D$
  - ↳ leads to  $V(\text{Fever}, X_{\text{ray}}, \text{Cancer}, \text{Smith})$
- ▶ data-dependent derivation of prohibited information can already be anticipated from the view declaration
- ▶ two MVDs imply non-trivial template dependency:

$$\frac{a_S \ b_1 \ a_D \ b_2}{a_S \ a_E \ b_3 \ a_P}$$

$$a_S \ a_E \ a_D \ a_P$$

- ▶  $a_S \mapsto \text{Fever}, a_D \mapsto \text{Cancer}, a_P \mapsto \text{Smith}$



## Conclusion and Future Work

- ▶ cqe guarantees confidentiality by inference control
- ▶ costly to implement: log file and theorem proving
- ▶ characterized necessity for inference control in terms of FDs and JDs
- ▶ natural access control
  - ▶ prevents any forbidden structures to ever arise
  - ▶ very efficient but maximal availability not always achievable
- ▶ signature-based access control
  - ▶ refuses rise of forbidden structure in last step
  - ▶ maximal availability and still efficient

