Providing Security to Sensitive Information in Relational Database Using The Concept of Null Value Theory

Prashant S Gumgaonkar[1] ; Prof. Chetan Bawankar[2] & Dr.M.B.Chandak

[1] Department of Computer Science & Engineering, Wainganga COE&M, RTM Nagpur University, Nagpur

[2] Assistant Professor Department of Computer Science & Engineering, Wainganga COE&M, RTM Nagpur University.

[3] Professor and Head Department of Computer Science & Engineering, Ramdeobaba College of Engineering, Nagpur.

[1]prashantsg90@gmail.com, [2] chetan251@rediffmail.com, [3]chandakmb@gmail.com

Abstract:

In this paper, our focus is on the study of Sensitive Information in Relational Database. In recent days lots of data or information is hack by the outsiders or we can say by the hackers. So our main approach is to provide secrecy and privacy to the database. So because of these purposes we keep sensitive information in a relational database hidden from a user or group thereof. We characterize sensitive data as the extensions of secrecy views. The database, before returning the answers to a query posed by a restricted user, is updated to make the secrecy views empty or a single tuple with null values. Then, a query about any of those views returns no meaningful information For these purpose the database is not change physically but whatever updates are done is only virtual and minimal.. Minimality makes sure that query answers, while being privacy preserving, are also maximally informative. The virtual updates are based on null values as used in the SQL standard. We provide the semantics of secrecy views, virtual updates, and secret answers to queries. The different instances resulting from the virtually updates are specified as the models of a logic program with stable model semantics, which becomes the basis for computation of the secret answers.

Keywords—

Data security; views; query answering; null values; view updates; answer set programs; database

I. INTRODUCTION

DBMS Stands for "Database Management System." In short, a DBMS is a database program. Technically speaking, it is software

System that uses a standard method of cataloging, retrieving, and running queries on data. The DBMS manages incoming data, organizes it, and provides ways for the data to be modified or extracted by users or other programs. Database management systems allow for massive storage of data, which can be efficiently accessed and manipulated. However, at the same time, the problems of data privacy are becoming increasingly important and difficult to handle. For example, for commercial or legal reasons, administrators of sensitive information may not want or be allowed to release certain portions of the data. It becomes crucial to address database privacy issues.

In this scenario, certain users should have access to only certain portions of a database likewise certain user don’t have access some portion of database this is the declaration. This declaration should be used by the database engine when queries are processed and answered. We would expect the database to return answers that do not reveal anything that should be kept protected from a particular user. On the other side and at the same time, the database should return as informative answers as possible once the privacy conditions have been taken care of.

II. BACKGROUND

For our approach to work, we rely on the following assumptions

(a) The user interacts via conjunctive query answering with a possibly incomplete database, meaning that the latter may contain null values, and this is something the former is aware of, and can count on (as with databases used in common practice). In this way, if a query returns answers with null values, the user will not know if they were originally in the database or were introduced for protection at query answering time.

(b) The queries request data, as opposed to schema elements, like integrity constraints and view definitions. Knowing the ICs (integrity constraint)(and about their satisfaction) in combination with query answers could easily expose the data protection policy. The most clear example is the one of a NOT NULL SQL constraint, when we see nulls where there should not be any.

(c) In particular, the user does not know the secrecy view definitions. Knowing them would basically reveal the data that is being protected and how.

These assumptions are realistic and make sense in many scenarios, for example, when the database is being accessed through the web, without direct interaction with the DBMS via complex SQL queries, or through ontology that offers a limited interaction layer. After all, protecting data may require additional measures, like withholding from certain users certain information that is, most likely, not crucial for many applications. From these assumptions and Proposition, we can conclude that the user cannot obtain information about the secrecy views through a combination of SAs(secret answer) to conjunctive queries. Therefore, there is not leakage of sensitive information.

III.RELATED WORK

Other researchers have investigated the problem of data privacy and access control in relational databases. We described in Section I the approach based on authorization views [27], [33]. In [19], the privacy is specified through values in cells within tables that can be accessed by a user. To answer a query Q without violating privacy, they propose the table and query semantics models, which generate masked versions of the tables by replacing all the cells that are not allowed to be accessed with NULL. When the user issues Q, the latter is posed to the masked versions of the tables, and answered as usual. The table semantics is independent of any queries, and views. However, the query semantics takes queries into account. [19] shows the implementation of two models based on query rewriting. Recent work [30] has presented a labeling approach for masking unauthorized information by using two types of special variables. They propose a secure and sound query evaluation algorithm in the case of cell-level disclosure policies, which determine for each cell whether the cell is allowed to be accessed or not. The algorithm is based on query modification, into one that returns less information than the original one. Those approaches propose query rewiring to enforce fine-grained access control in databases. Their approach is mainly algorithmic. Data privacy and access control in incomplete propositional databases has been studied in [6], [7], [22]. They take a different approach, *control query evaluation* (CQE), to fine-grained access control. It is policy-driven, and aims to ensure confidentiality on the basis of a logical framework. A security policy specifies the facts that a certain user is not allowed to access. Each query posed to the database by that user is checked, as to whether the answers to it would allow the user to infer any sensitive information. If that is the case, the answer is distorted by either *lying* or *refusal* or *combined lying and refusal*. In [8], they extend.

IV.PROBLEM STATEMENT

Some recent papers approach data privacy and access control on the basis of authorization views. View-based data privacy usually approaches the problem by specifying which views a user is allowed to access.

For example, when the database receives a query from the user, it checks if the query can be answered using those views alone. More precisely, if the query can be rewritten in terms of the views, for every possible instance. If no complete rewriting is possible, the query is rejected. In the problem about the existence of a *conditional* rewriting is investigated, i.e. relative to an instance at hand.

According to our approach, the information to be protected is declared as a *secrecy view*, or a collection of them. Their extensions have to be kept secret. Each user or class of them may have associated a set of secrecy views. When a user poses a query to the database, the system virtually updates some of the attribute values on the basis of the secrecy views associated to that user. In this work, we consider updates that modify attribute values through null values, which are commonly used to represent missing or unknown values in incomplete databases. As a consequence, in each of the resulting updated instances, the extension of each of the secrecy views either becomes empty or contains a single tuple showing only null values. Either way, we say that the secrecy view becomes null. Then, the original query is posed to the resulting class of updated instances. This amounts to: (a) Posing the query to each instance in the class. (b) Answering it as usual from each of them. (c) Collecting the answers that are shared by all the instances in the class. In this way, the system will return answers to the query that do not reveal the secret The next example illustrates the gist of our approach

V. CONCLUSION & FUTURE WORK

In this work, we propose a logical framework and a methodology to answer conjunctive queries that do not reveal secret information as specified by secrecy views. We have concentrate on the case of conjunctive secrecy views and conjunctive queries, but it is possible to relax these restrictions. We assume that the databases may contain nulls, and also nulls are used to protect secret information, by virtually updating with nulls some of the attribute values.

The update semantics enforces (or captures) two natural requirements. That the updates are based on null values, and that the updated instances stay close to the given instance. In this way, the query answers become implicitly maximally informative, while not revealing the original contents of the secrecy views.

The null values are treated as in the SQL standard, which in our case, and for conjunctive query answering, is reconstructed in classical logic. This reconstruction captures well the “semantics” of SQL nulls (which in not clear or complete in the standard), at least for the case of conjunctive query answering, and some extensions thereof.

The null values are treated as in the SQL standard, which in our case, and for conjunctive query answering, is reconstructed in classical logic. This reconstruction captures well the “semantics” of SQL nulls (which in not clear or complete in the standard), at least for the case of

conjunctive query answering, and some extensions thereof. This is the main reason for concentrating on conjunctive queries and views. In this case, queries and views can be syntactically transformed into conjunctive queries and views for which the evaluation or verification can be done by treating nulls as any other constant.

The secret answers are based on a skeptical semantics. In principle, we could consider instead the more relaxed possible or brave semantics: an answer would be returned if it holds in some of the secrecy instances. The possibly secret answers would provide more information about the original database than the (certainly) secret answers. However, they are not suitable for our privacy problem.

In future work first we create the database and try to access the information in to the database after that providing null value to the sensitive data so that the original data is not seen.

VI.REFERENCES

[1] Abiteboul, S., Hull, R. and Vianu, V. Foundations of Databases, Addison-Wesley, 1995.

[2] Bertossi, L. Consistent Query Answering in Databases. ACM Sigmod Record, June 2006, 35(2):68 76.

[3] Bertossi, L. Database Repairing and Consistent Query Answering, Morgan & Claypool, Synthesis Lectures on Data Management, 2011.

[4] Biskup, J. and Weibert, T. Confidentiality Policies for Controlled Query Evaluation. In Data and Applications Security, Springer LNCS 4602, 2007, pp. 1-13.

[5] Biskup,J. and Weibert. Keeping Secrets in Incomplete Datbabases. International Journal of Information Sercurity, 2008, 7(3):199-217.

[6] Biskup, J., Tadros, C. and Wiese, L. Towards Controlled Query Evaluation for Incomplete First-Order Databases. In Proc. FoIKS'10, Springer LNCS 5956, 2010, pp. 230-247.

[7] Bravo, L. and Bertossi, L. Semantically Correct Query Answers in the Presence of Null Values. Proc. EDBT WS on Inconsistency and Incompleteness in Databases (IIDB'06), J. Chomicki and J. Wijsen
(eds.), Springer LNCS 4254, 2006, pp. 336-357

[8] Bravo, L. Handling Inconsistency in Databases and Data Integration Systems. PhD. Thesis, Carleton University, Department of Computer Science, 2007.
http://people.scs.carleton.ca/∼bertossi/papers/Thesis36.pdf

[9] Caniupan, M. and Bertossi, L. The Consistency Extractor System: Answer Set Programs for Consistent Query Answering in Databases. Data & Knowledge Engineering, 2010, 69(6):545-572.

[10] Codd, E.F. Extending the database relational model to capture more meaning. ACM Trans. Database Syst., 1979, 4(4):397-434.

[11] Cosmadakis, S. and Papadimitriou, Ch. Updates of Relational Views.

[12] Chomicki, J. and Marcinkowski, J. Minimal-Change Integrity Maintenance Using Tuple Deletions. Information and Computation, 2005, 197(1-2):90-121.

[13] Gelfond, M. and Lifschitz, V. Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing, 1991, 9:365-385.

[14] Gelfond, M. and Leone, N. Logic Programming and Knowledge Representation: The A-Prolog Perspective. Artificial Intelligence, 2002, 138(1-2):3-38.

[15] Gupta, A. and Singh Mumick, I. Maintenance of Materialized Views: Problems, Techniques, and Applications. IEEE Data Engineering Bulletin, 1995, 18(2):3-18.

[16] Imielinski, T. and Lipski, W. Jr. Incomplete Information in Relational Databases. Journal of the ACM, 1984, 31(4):761-791.

[17] LeFevre, K., Agrawal, R., Ercegovac, V., Ramakrishnan, R., Xu, Y. and DeWitt, D. Limiting Disclosure in Hippocratic Databases. In

Proc. International Conference on Very large Data Bases
(VLDB’04), 2004, pp. 108-119.
[18] Lechtenb¨orger, J. and Vossen, G. On the Computation of Relational View Complements. Proc. ACM Symposium on Principles of Database Systems (PODS’02), 2002, pp. 142-149.
[19] Lechtenb¨orger, J. The Impact of the Constant Complement Approach towards View Updating. Proc. ACM Symposium on Principles of Database Systems (PODS’03), 2003, pp. 49-55.
[20] Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S. and Scarcello, F. The DLV System for Knowledge Representation and Reasoning. ACM Transactions on Computational Logic, 2006, 7(3):499-562.
[21]Levene, M. and Loizou, G. A Guided Tour of Relational Databases and Beyond. Springer, 1999.
[22] Li, L. Achieving Data Privacy Through Virtual Updates. MSc. Thesis, Carleton University, Department of Computer Science, 2011.
http://people.scs.carleton.ca/∼bertossi/papers/thesisLechen.pdf
[23] Nash, A., Segoufin, L. and Vianu, V. Views and Queries: Determinacy and Rewriting. ACM Transactions on Database Systems, 2010, 35(3).
[26] Reiter, R. Towards a Logical Reconstruction of Relational Database Theory. In On Conceptual Modelling, M.L. Brodie, J. Mylopoulos and J.W. Schmidt (eds.), Springer, 1984, pp. 191–233.
[24] Rizvi, S., Mendelzon, A., Sudarshan, S. and Roy, P. Extending Query Rewriting Techniques for Fine-Grained Access Control. In Proc. Proc. ACM International Conference on Management of Data (SIGMOD’04), 2004, pp. 551-562.
[25] Traylor, B. and Gelfond, M. Representing Null Values in Logic Programming. In Logical Foundations of Computer Science, Proc. LFCS’94. Springer LNCS 813, 1994, pp. 341-352.