www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3 Issue 7 July, 2014 Page No. 7186-7190

Protecting Sensitive Information in Relational Databases By Using Null Value Theory

Prashant Gumgaonkar[1], Dr.M.B.Chandak[2], Manisha Kamble[3]

[1]RTM Nagpur University , Maharajbagh Road ,
Nagpur , Maharashtra , India.
Prashantsg90@gmail.com

[2] RTM Nagpur University , Maharajbagh Road ,
Nagpur , Maharashtra , India.
chandakmb@gmail.com

[3] RTM Nagpur University , Maharajbagh Road ,
Nagpur , Maharashtra , India.
manishakamble@gmail.com

Abstract: ***In this paper, our objective is on the study and analysis of Sensitive Information in Relational Database. In recent days we have seen that various information or data is hack by the unauthenticated person. So based upon that concept our main approach is to provide secrecy and privacy. For that purpose we characterize sensitive data or information as the extensions of secrecy views. The database, before returning the answers to a query fired by any restricted user, is updated to make the secrecy views empty or a single tuple showing only null values. Then, a query about any of those views returns no meaningful information because of these the database is not physically change but whatever updates are done is only virtual and minimal. Minimality makes sure that query answers, while being privacy preserving, are also maximally informative. Whatever virtual updates are proportional to the null values as used in the SQL standard. We provide the semantics of secrecy views, virtual updation, and secret answers to queries.***

Keywords: Data security , Null Value, Views, Relational Database.

1. Introduction

DBMS Stands for "Database Management System." In short, in other words a DBMS is a database program or it is a collection of data. Technically speaking, it is a software system that uses a standard method of cataloging, retrieving, and running queries on data. The DBMS manipulate incoming data, organizes it, and gives ways for the data to be updated 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 security are becoming increasingly important and it is 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 hand and at the same time, the database should return as informative answers as possible once the privacy conditions have been taken care of.

2. Problem statement

2.1 Related Work

Based on the above concept, some researchers have investigated the problem of data privacy and access control in relational databases. We described in Section I this concept is based on authorization views. In [19], the privacy and secure of information is specified in terms of 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. [16] Shows the implementation of two models based on query rewriting. Recent work 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, 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]. 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. Some recent papers shows data privacy and access control is on the basis of view base authorization. View-based data privacy or security 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 or outsiders, it checks if that query can be answered using those views alone. More precisely, if the query can be rewritten in terms of the views, for every possible in stance. 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.

3. Proposed System

According to our approach, the information or data 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.*

For our approach to work, we rely on the following assumptions:-
(a) The user or client 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 tha is being protected and how. These assumptions are realistic and into one that returns less information than the original one.

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.

4. Methodology

Now in our approach we require the database through which the client try to access the information from server.so for this we take the company database in which there are various tables like Company , Project , Person Manager, Company Manager , Meeting , Person has a Project etc. all this tables are related to each other by means of primary and foreign keys. It is illustrated in following figure:

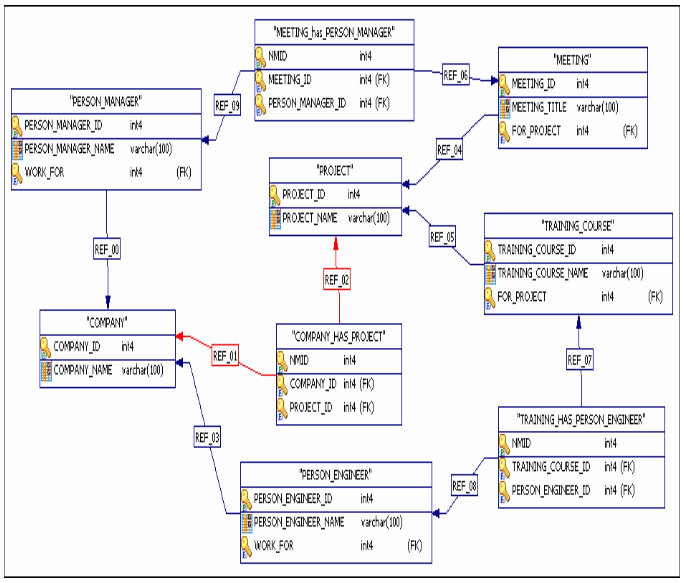


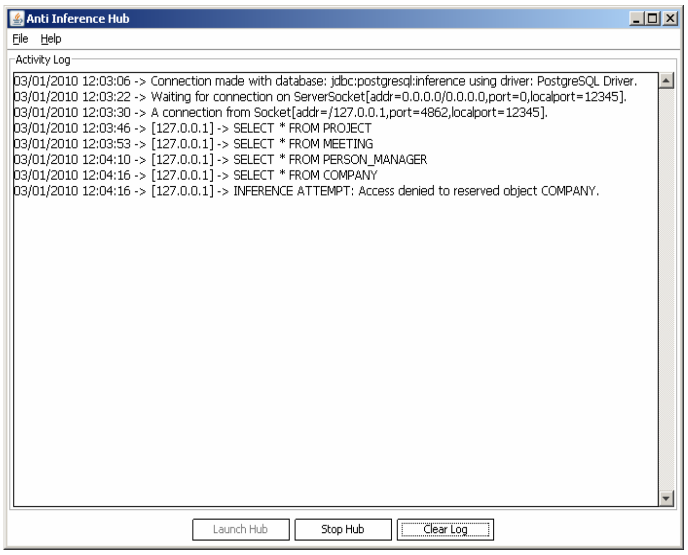
Fig-1: Relational database

In the above figure various tables are related to each other by using the concept of primary keys and foreign key. This database is called as Relational Database Management System. All the tables have various object and attributes of particular table. This database is used for further extracting query from server side. In our methodology first we install the company database by performing the various steps of installation. Then established the server connection by launching the hub. Now connect database through the server by entering URL, user name and password. By doing this step the connection is successful. Now setup the inference channel and add respective object into this inference channel to which the security is provided. After this the keys is initialize here the auto generated keys are automatically created. Now create the client and connect it through server through host name. At the end writing the query in client which will retrieval the data from server. After firing the query we will get the output in XML

format from the given database. So in this way we try to access the information without disturb the original database.
Following example illustrate the above methodology. for this we take the company database and based upon this we make the scenario. In this scenario queries are fired to the database and it try to retrieval the information from the database as given in following example.
Example 1 :
Select Person_Manager_Name, Company_Name
from Person_Manager, Company where Person_Manager.work_for=company.company_id and Person_Manager.Person_Manager_Name = 'Manager 1';
In this query, you are trying to know for which COMPANY a PERSON_MANAGER works. You should receive the following XML output from the Hub:
<? xml version="1.0" encoding="UTF-8"?>
<person_manager_name>MANAGER
</person_manager_name>
<company_name>COMPANY </company_name>
Which tells you that “MANAGER 1” works for “COMPANY 1”? Note the change to "Channel 1" objects keys by opening the Initialize Keys Box in the Hub.
Select Meeting_Title, Person_Manager_Name
From Meeting, Meeting_Has_Person_Manager, Person_Manager where Meeting.Meeting_id = Meeting_Has_Person_Manager.Meeting_id
and Person_Manager.Person_Manager_id =
Meeting_Has_Person_Manager.Person_Manager_id and Meeting.Meeting_Title = 'Meeting 1';
In this query, you are trying to know the PERSON_MANAGER attending a MEETING. You should receive the following XML output from the Hub:
<? xml version="1.0" encoding="UTF-8"?>
<meeting_title>MEETING 1</meeting_title>
<person_manager_name>MANAGER
</person_manager_name>
Which tells you that “MANAGER 1” attends “MEETING 1”? Note the change to "Channel 1" objects keys by opening the Initialize Keys Box in the Hub. Note that
PROJECT object is now a reserved object in "Channel 1" because it has an empty key set. Now, execute the following query in the Hub Client:
Select Meeting_Title, Project_Name
From Meeting, Project
Where Meeting.For_Project = Project.Project_Id
And Project. Project_Name = 'Project 1';
In this query, you are trying to know the MEETING on a PROJECT. If you received a response to this query (actually the response will tell you that "MEETING 1" is on project "PROJECT 1"), then you can immediately infer that "COMPANY 1" is supporting "PROJECT 1"; an inference you base on the results for queries you executed so far. Therefore, result for this query should be blocked. Indeed, it is! The output from the Hub will be:
INFERENCE ATTEMPT: Access denied to reserved object PROJECT.
<?xml version="1.0" encoding="UTF-8"?>
<meeting_title>MEETING 1</meeting_title>
<project_name>PROJECT 1</project_name>
Thus, as a super client, the Hub allows you to infer. Please note that Anti Inference Hub does not take any IP spoofing attacks into consideration. Securing a network against such attacks falls beyond the purpose of Anti Inference Hub.

5. Result and Description

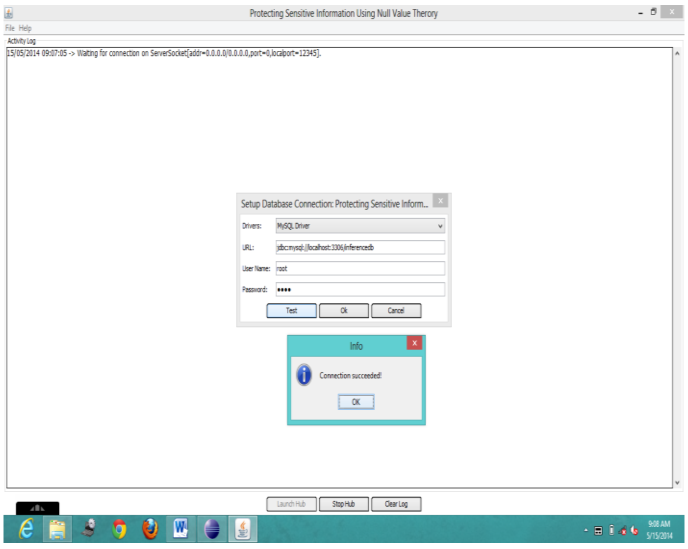
First we create the server and launch the hub so we get following snap shot:



```
03/01/2010 12:03:06 -> Connection made with database: jdbc:postgresql:inference using driver: PostgreSQL Driver.
03/01/2010 12:03:22 -> Waiting for connection on ServerSocket[addr=0.0.0.0/0.0.0.0,port=0,localport=12345].
03/01/2010 12:03:30 -> A connection from Socket[addr=/127.0.0.1,port=4862,localport=12345].
03/01/2010 12:03:46 -> [127.0.0.1] -> SELECT * FROM PROJECT
03/01/2010 12:03:53 -> [127.0.0.1] -> SELECT * FROM MEETING
03/01/2010 12:04:10 -> [127.0.0.1] -> SELECT * FROM PERSON_MANAGER
03/01/2010 12:04:16 -> [127.0.0.1] -> SELECT * FROM COMPANY
03/01/2010 12:04:16 -> [127.0.0.1] -> INFERENCE ATTEMPT: Access denied to reserved object COMPANY.
```

Fig 1: Server connection

Second step is to connect the database in the server by entering the appropriate URL, user name and password. Following figure shows this activity.


Protecting Sensitive Information Using Null Value Therory
File Help
Activity Log
Setup Database Connection: Protecting Sensitive Inform...
Test
Ok
Cancel
Info
OK
Launch Hub
Stop Hub
Clear Log
9:08 AM
5/15/2014

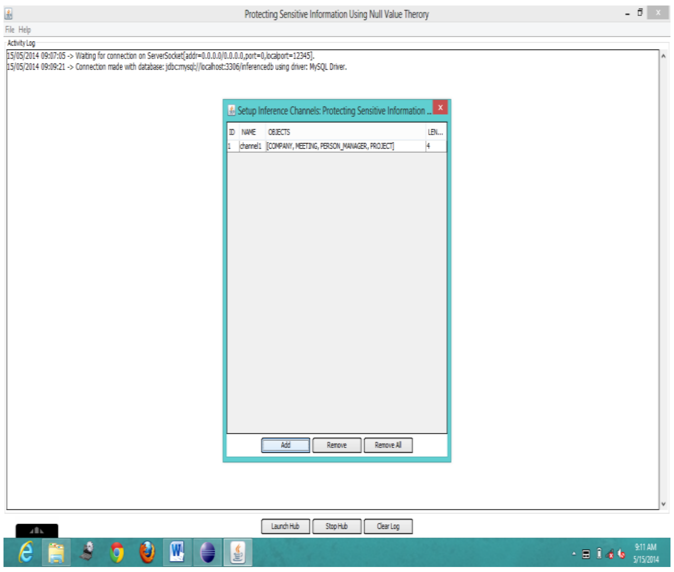
```
15/05/2014 09:07:05 -> Waiting for connection on ServerSocket[addr=0.0.0.0/0.0.0.0,port=0,localport=12345].
```

Drivers: MySQL Driver
URL: jdbc:mysql://localhost:3306/inferencedb
User Name: root
Password: ••••

Connection succeeded!

Fig 2: Database connection

By entering the URL , user name and password we established the database connection.
Now setup the inference channel and add the object into the inference channel to which we provided the security to the respective table in the database. So we get following snap shot :



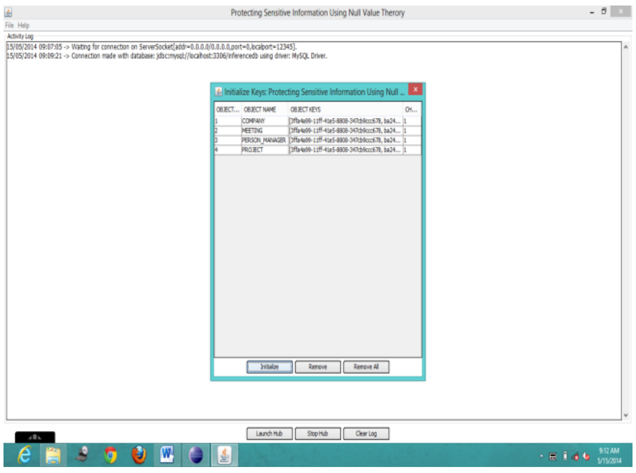
```
15/05/2014 09:07:05 -> Waiting for connection on ServerSocket[addr=0.0.0.0/0.0.0.0,port=0,localport=12345].
15/05/2014 09:09:21 -> Connection made with database: jdbc:mysql://localhost:3306/inferencedb using driver: MySQL Driver.
```

ID	NAME	OBJECTS	LEN...
1	channel1	[COMPANY, MEETING, PERSON_MANAGER, PROJECT]	4

Protecting Sensitive Information Using Null Value Therory
File Help
Activity Log
Setup Inference Channels: Protecting Sensitive Information ...
Add
Remove
Remove All
Launch Hub
Stop Hub
Clear Log

Fig 3: Inference Channel

After this there is auto generated keys which automatically generated accordingly the object add to this inference channel. After executing the query in the client side the keys are decrease according to query. And at last the keys of particular object which are continuously connected to the each query are become zero. This will be seen in inference channel.



```
15/05/2014 09:07:05 -> Waiting for connection on ServerSocket[addr=0.0.0.0/0.0.0.0,port=0,localport=12345].
15/05/2014 09:09:21 -> Connection made with database: jdbc:mysql://localhost:3306/inferencedb using driver: MySQL Driver.
```

OBJECT...	OBJECT NAME	OBJECT KEYS	CH...
1	COMPANY	[3ffa4a99-11ff-41e5-8808-347cb9ccc678, ba24...	1
2	MEETING	[3ffa4a99-11ff-41e5-8808-347cb9ccc678, ba24...	1
3	PERSON_MANAGER	[3ffa4a99-11ff-41e5-8808-347cb9ccc678, ba24...	1
4	PROJECT	[3ffa4a99-11ff-41e5-8808-347cb9ccc678, ba24...	1

Protecting Sensitive Information Using Null Value Therory
File Help
Activity Log
Initialize Keys: Protecting Sensitive Information Using Null ...
Initialize
Remove
Remove All
Launch Hub
Stop Hub
Clear Log

Fig 4: Key initialization

Now client creation and connect it to server through host name

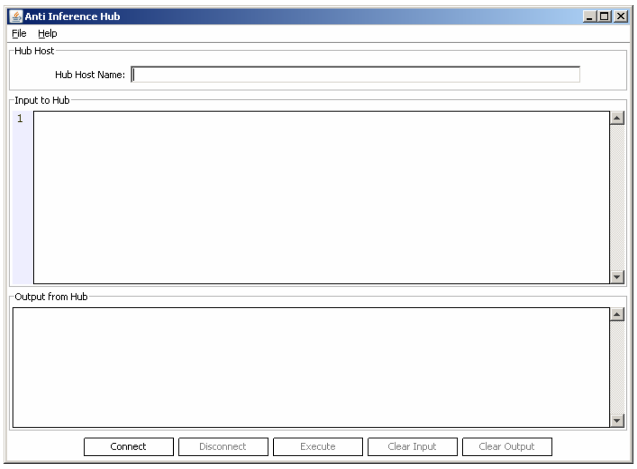
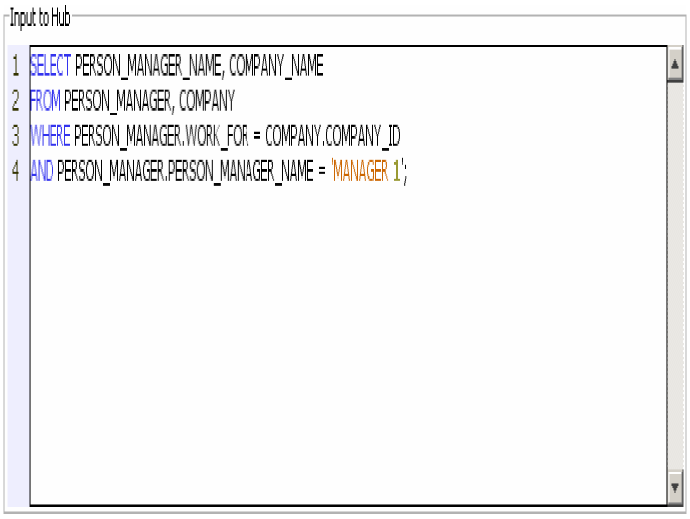


Fig 5: Client creation

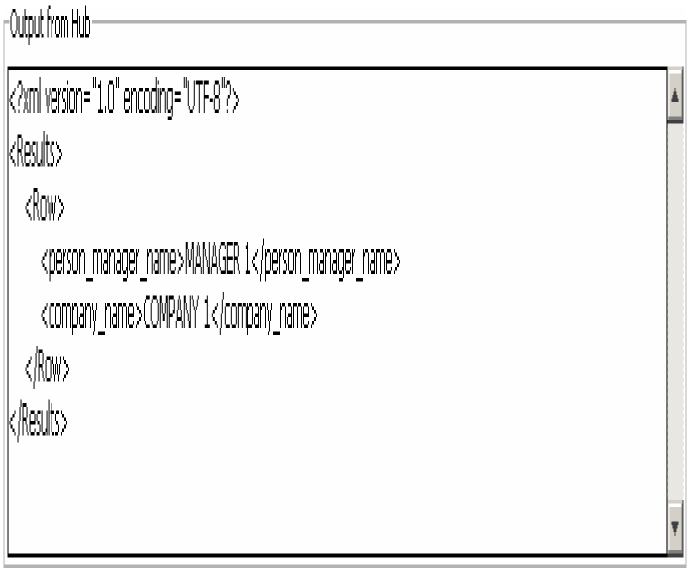
Now writing query in client which will retrieval data from server this is shown in below snap shot:



```
SELECT PERSON_MANAGER_NAME, COMPANY_NAME
FROM PERSON_MANAGER, COMPANY
WHERE PERSON_MANAGER.WORK_FOR = COMPANY.COMPANY_ID
AND PERSON_MANAGER.PERSON_MANAGER_NAME = 'MANAGER 1';
```

Fig 6: Query firing

After firing the query we will get the output in XML format shown in following snap shot:



```
<?xml version="1.0" encoding="UTF-8"?>
<Results>
  <Row>
    <person_manager_name>MANAGER 1</person_manager_name>
    <company_name>COMPANY 1</company_name>
  </Row>
</Results>
```

Fig 7: Output in xml

6. Conclusion

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**.**

7. 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
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