Authorization Policies for Materialized Views

Sarah Nait Bahloul, Emmanuel Coquery, Mohand-Said Hacid

Authorization Policies for Materialized Views

Sarah Nait-Bahloul, Emmanuel Coquery, and Mohand-Saïd Hacid

Université de Lyon
Université Claude Bernard Lyon 1 LIRIS CNRS UMR 5205
43, bd du 11 novembre 1918
69622 Villeurbanne cedex, France
sarah.nait-bahloul,emmanuel.coquery,mshacid@liris.cnrs.fr

Abstract. In this paper, we propose a novel approach to facilitate the administration of access control policies to ensure the confidentiality of data at the level of materialized views. A materialized view stores both the definition of the view and the rows resulting from the execution of the view. Several techniques and models have been proposed to control access to databases, but to our knowledge the problem of automatically generating from access control policies defined over base relations the access control policies that are needed to control materialized views is not investigated so far. We are dealing with this problem by resorting to an adaptation of query rewriting techniques. We choose to express fine-grained access control through authorization views.[1]

Keywords: Database Security, Access Control, Authorization Views, Materialized Views, Datalog.

1 Introduction

In the area of data management, the problem of access control was one of the most sensitive issues. Several techniques and models have been proposed to improve data security and to ensure data confidentiality (see, among others, [10][7]). With the use of large systems like Data Warehouses [15] or Distributed Database Systems [3], new security issues arise [9][13]. In our work, we focus on the problem of securing materialized views. A lot of organizations use relational DBMS to store and manage their information and very often they resort to materialized views. A materialized view records the results returned by the query into a physical table. In many settings, the views are materialized in order to optimize access. For instance, in Data Warehouses, they can be used to precompute and store complex aggregations. Thus, the user can use the materialized view as any other base relation. In this context, ensuring security at the materialized view level is as important as ensuring security at the level of tables. The question is then *how to ensure data security at the level of a materialized view*? So far, access control rules on materialized views are defined manually by an administrator by trying to comply with the basic ones.In a system containing tens or

[1] This work is partially supported by the Rhône-Alpes Region, Cluster ISLE (Informatique, Signal, Logiciel Embarqué).

hundreds of tables controlled by tens or hundreds of rules, it becomes impossible for administrators to deal with such large sets of rules and consider all the relevant ones.
Based on our previous work [6], we propose a novel approach that compute new access rules based on existing access rules over base relations. In our approach, we consider fine-grained authorization policies that are defined and enforced in the database through authorization views [10]. Figure 1 summarizes our approach. The idea is the following: Given a set of base relations (with the corresponding authorizations) and a set of materialized view definitions, synthesize a set of authorization views that will be attached to the materialized views, such that querying the materialized views through those authorization views does not deliver more information than querying the database through the original ones. In this paper, authorization and materialized views are restricted to conjunctive queries.
The rest of the paper is organized as follows: Section 2 discusses authorization views. Section 3 presents our approach. In section 4, we present the related work. We conclude in section 5.

Fig. 1. Security policies for materialized views: System architecture

2 Datalog for Authorization

Among the several proposed techniques and models to ensure data confidentiality, we are particularly interested by the "Authorization views" [10]. They are a well known database technique that provides content-based and fine-grained access control. Authorization views are logical tables that specify exactly the accessible data, either drawn from a single table or from multiple tables.
The goal of our work is to automatically determine the set of "authorization views" that will be attached to materialized views. Doing so, the user can query her/his appropriate authorization views or (s)he can query the materialized views and the system will rewrite the query using the authorization views. Thus, in our proposal, we are independent of the way the materialized views are accessed. Nevertheless, to facilitate the understanding of our approach and without loss of generality, we assume that the user can query only the authorization views.
As we restrict our self to conjunctive queries, we use non recursive Datalog without negation [1] as a formal framework for expressing access control rules. For

example, the following rule defines an authorization view on the Doctor table:

av$_1$*(IdD, Dname, Dfname, Dspecialty)*←
doctor (IdD, Dname, Dfname, Dadr, Dphone, Dspecialty, Dsalary).

The user has right to view the last name, the first name and the speciality of doctors in the hospital, but not personal nor salary information.

3 Contribution

In this section we present our algorithm *ACMV* (Access Control to Materialized Views) that generates a set of authorization views $\mathcal{AVMV}$ that should be attached to and defined on materialized views. The algorithm takes as input two sets: (1) A set of views ($\mathcal{AV}$), representing the authorization views defined on the base tables and (2) a set of views ($\mathcal{MV}$), representing the definitions of materialized views.
The generated views should be *secure*. Secure means that the generated views should not give access to information that are no allowed by the basic authorization views. We have to guarantee that for each query on $\mathcal{AVMV}$, there exists an equivalent query on $\mathcal{AV}$.
In order to automatically generate the relevant set of authorization views $\mathcal{AVMV}$, we propose a novel approach based on query rewriting techniques. To ensure the security property, we propose an algorithm that performs a double rewriting, using the two sets of views ($\mathcal{AV}$ and $\mathcal{MV}$).
We first describe the core of our algorithm which is based on query rewriting technique, namely MiniCon [8]. We propose an adaptation of this algorithm to the security context.

S-MiniCon: An Adaptation of the MiniCon algorithm to the security context

The MiniCon algorithm [8] was initially proposed as an efficient method for answering queries using views. It takes as input a query Q and a set of views V and calculate all possible rewritings of Q using views in V, such that, for each rewriting rw, we have $rw \subseteq$ Q.
Let us take a simple example to show why we have to adapt the MiniCon algorithm to the security context. Assume the query (1) shown below, which makes a copy of the table $patient(IdP, Name, Disease)$. The authorization view (2) specifies an authorization access to the tuples $(IdP, Name)$ of the *patient* relation.

$$q(IdP, Name, Disease) \leftarrow patient(IdP, Name, Disease). \quad (1)$$

$$av(IdP, Name) \leftarrow patient(IdP, Name, Disease). \quad (2)$$

We propose to rewrite the query q using the authorization view av in order to determine which set of tuples in q is accessible given av. If we apply the original MiniCon algorithm, the authorization view av will be considered as irrelevant.

The condition regarding the head variables is not satisfied [8]. But, if we do not take the authorization view as relevant, no rewriting will be generated, i.e. no tuple in q is accessible. It is too restrictive, since one can have access to the tuples $(IdP, Name)$ by projecting them out. Therefore, in our framework, we propose to adapt the MiniCon algorithm by relaxing the condition on the head variables.

ACMV algorithm: Effectively and efficiently apply the S-MiniCon algorithm

In this section, we present our proposal that exploits a double rewriting. The algorithm takes as input a set of views $\mathcal{Q}$ to rewrite and the two sets of views $\mathcal{AV}$ and $\mathcal{MV}$. For the first iteration, we define the set of views $\mathcal{Q}$ that specify a full access to $\mathcal{MV}$. The algorithm starts by rewriting each q_i of $\mathcal{Q}$ using $\mathcal{AV}$, the result is a set of rewritings $\mathcal{RW}_{q_i}$. Let $\mathcal{RW}_{q_i} = \{rw_1, ..., rw_n\}$. This first step determines which set of tuples of q_i is accessible from $\mathcal{AV}$. The second step consists in checking if this set is also accessible from $\mathcal{MV}$. For this, the algorithm rewrites each rw_j of $\mathcal{RW}_{q_i}$ using $\mathcal{MV}$. We note $\mathcal{RW}_{rw_j}$, the rewritings generated with this second rewriting.

Algorithm 1: *Double rewriting*

Input: *q the view to rewrite*
$\mathcal{AV}$:Set of authorization views
$\mathcal{MV}$: Set of materialized views
Output: *$\mathcal{RW}$: Set of Rewritings*

q^{exp} = *expansion(q)*
$\mathcal{RW}_q$ = *S-MiniCon(q^{exp},$\mathcal{AV}$)*
foreach *rewriting* rw_j *of* $\mathcal{RW}_q$ **do**
  rw_j^{exp} = *expansion(rw_j)*
  $\mathcal{RW}_{rw_j}$ = *S-MiniCon(rw_j^{exp},$\mathcal{MV}$)*
  add $\mathcal{RW}_{rw_j}$ *to* $\mathcal{RW}$
end
return $\mathcal{RW}$

In order to ensure the equivalence of the security property, the algorithm must check whether the application of the double rewriting using $\mathcal{AV}$ and $\mathcal{MV}$ has filtered tuples of $\mathcal{Q}$. For this, we propose to check, for each q_i, if the union of the generated rewritings contains q_i. We rely on subsumption [2] algorithm for this check. We recall here that the application of the S-Minicon algorithm will generate rewritings that do not necessarily have the same schema (we have relaxed the condition on the head variables). So, to verify the containment, the algorithm selects only the comparable rewritings ($\mathcal{CRW}$) and verify if $q_i \not\subseteq \bigcup_{j=1}^{n} \mathcal{CRW}_{rw_j}$. A comparable rewriting [14] is a rewriting that has the

same schema as the query. The subsumption test verifies that no tuple has been filtered by the double rewriting. In other words, any tuple in q_i can be accessed by both $\mathcal{AV}$ and $\mathcal{MV}$. The algorithm then terminates and returns q_i as a new authorization view on $\mathcal{MV}$. Otherwise, $q_i \not\subseteq \bigcup_{j=1}^{n} \mathcal{CRW}_{rw_j}$, which means that one of the two rewriting steps has filtered some tuples. In this case the double rewriting algorithm is applied again by considering $\mathcal{RW}_{rw_j}$ (for j from 1 to n) as the set of queries to be rewritten.

Algorithm 2: *$\mathcal{ACMV}$Algorithm*

Input: *$\mathcal{Q}$: Set of views which give a full access on $\mathcal{MV}$*
$\mathcal{AV}$: Set of authorization views on basic relations
$\mathcal{MV}$: Set of materialized views
Output: *$\mathcal{AVMV}$: Set of authorization views on $\mathcal{MV}$*

```
while Q is not empty do
    pick q_i in Q
    RW = DoubleRewriting(q_i, AV, MV)
    if ⋃CRW subsumes q_i then
        Add q_i to AVMV
    else
        Add RW to Q
return AVMV;
```

4 Related Work

Rosenthal and Sciore [11] have considered the problem of how to automatically coordinate the access rights of the warehouse with those of sources. The framework proposed by the authors determine only if a user has right to access a derived table (based on explicit permission) but our proposal goes further by determining which part the user has right to access in the derived table. Also, the authors stated the inference rules at a high level. The properties of the underlying inference system and the efficiency of the proposed algorithm were not investigated and remain an open research issue.
In [4], the authors have built on [5] to provide a way to select access control rules to be attached to materialized view definitions based on access control rules over base relations. They resort to the basic form of the bucket algorithm which does not allow to derive all relevant access control rules. Another limitation of this work is that since they only deal with selection of rules, the framework remains strongly dependent of the base relations. That is, the body of the derived rules involves base relations only. In our work, we synthesize new rules from existing rules where the body of the new rules makes reference to materialized views.

5 Conclusion

In the case of large organizations, the management of thousands of datasets is very common. Ensuring data confidentiality in the presence of materialized

views is also important. In this work, we presented a novel approach for an automated method to derive authorization views to be attached to materialized views. We also presented S-MiniCon algorithm, an adaptation of a query rewriting algorithm to the security context. As mentioned above, we have discussed only conjunctive queries. In large systems, (e.g., data warehouses), materialized views can be used to precompute and store aggregated data (e.g., sum of sales). This framework should be extended to accommodate materialized views with aggregations. For this purpose, we will consider algorithms for rewriting aggregate queries using views [12].

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
2. U. S. Chakravarthy, J. Grant, and J. Minker. Logic-based approach to semantic query optimization. *ACM Trans. Database Syst.*, 15(2):162–207, 1990.
3. L. W. F. Chaves, E. Buchmann, F. Hueske, and K. Böhm. Towards materialized view selection for distributed databases. In *EDBT*, pages 1088–1099, 2009.
4. A. Cuzzocrea, M.-S. Hacid, and N. Grillo. Effectively and efficiently selecting access control rules on materialized views over relational databases. In *IDEAS*, pages 225–235, 2010.
5. S. Nait-Bahloul. Inference of security policies on materialized views. rapport de master 2 recherche. http://liris.cnrs.fr/∼snaitbah/wiki, 2009.
6. S. Nait-Bahloul, E. Coquery, and M.-S. Hacid. Access control to materialized views: an inference-based approach. In *EDBT/ICDT Ph.D. Workshop*, pages 19–24, 2011.
7. L. E. Olson, C. A. Gunter, and P. Madhusudan. A formal framework for reflective database access control policies. In *ACM Conference on Computer and Communications Security*, pages 289–298, 2008.
8. R. Pottinger and A. Y. Levy. A scalable algorithm for answering queries using views. In *VLDB*, pages 484–495, 2000.
9. T. Priebe and G. Pernul. A pragmatic approach to conceptual modeling of olap security. In *In Proc. ER*, pages 311–324. Springer-Verlag, 2001.
10. S. Rizvi, A. O. Mendelzon, S. Sudarshan, and P. Roy. Extending query rewriting techniques for fine-grained access control. In *SIGMOD Conference*, pages 551–562, 2004.
11. A. Rosenthal and E. Sciore. Administering permissions for distributed data: Factoring and automated inference. In *In Proc. of IFIP WG11.3 Conf*, 2001.
12. D. Srivastava, S. Dar, H. V. Jagadish, and A. Y. Levy. Answering queries with aggregation using views. In *VLDB*, pages 318–329, 1996.
13. J. Steger, H. Gnzel, and A. B. 0004. Identifying security holes in olap applications. In B. M. Thuraisingham, R. P. van de Riet, K. R. Dittrich, and Z. Tari, editors, *DBSec*, volume 201 of *IFIP Conference Proceedings*, pages 283–294. Kluwer, 2000.
14. J. Wang, M. Maher, and R. Topor. Rewriting unions of general conjunctive queries using views. In *Proc. Conf. on Extending Database Technology, LNCS 2287*, 2002.
15. J. Yang, K. Karlapalem, and Q. Li. Algorithms for materialized view design in data warehousing environment. In *VLDB*, pages 136–145, 1997.