International Journal of Education and Science Research Review
ISSN 2348-6457
Volume-1, Issue-5
October- 2014
www.ijesrr.org
Email- editor@ijesrr.org

Approach for Information Leakage and Redundancy in Fine Grained Access Control

Ranjeet Singh, Chiranjit Dutta
Faculty of Information Technology
SRM University
NCR Campus, Ghaziabad

ABSTRACT:
Fine Grained Access Control (FGAC) provides a row-level security mechanism that's built-in to the Oracle database engine. FGAC allows to granularly specifying the exact information elements that each user may access. Several proposals have been made for fine grained access control which has resulted in information leakage and redundancy of a query. So we have proposed a query evaluation algorithm which supports fine grained access control. There are several models for fine grained access control but major of them follow a view replacement model in which these two problems occur. First is the redundancy of data as data is masked from one level to other second is leakage of information through the error and exception message. In our paper we have proposed our own algorithm for the disclosure policy of data residing in the database that avoids the leakage of information and controls the redundancy of data.
Keywords: FGAC, Sound, Secure, Maximum

1. INTRODUCTION

Controlling the access to data based on user credentials is a fundamental part of database management systems. In most cases, the level at which information is controlled extends only to a certain level of granularity. But in some cases, there is a requirement to control access at a more granular way allowing the users to see only the data they are supposed to see in a database table. Fine grained access control (FGAC) provides row-level security capabilities to secure information stored in modern relational database management systems.

FGAC implemented at the application level, has numerous drawbacks; hard to maintain consistency between Various applications, such control may be bypassed. So placing access control at the database level, one can ensure that access control policies are consistently applied to every user and every application.

Several models have been proposed which include Virtual Private Database in Oracle, label-based access control in DB2. Past work describes how a policy is specified and how to answer a query while enforcing a policy. But a clear specification of the correctness criteria of the enforcement scheme is lacking.

In this paper we identify three criteria for enforcing fine grained access control policies in databases. An algorithm that enforces fine grained access control policies should be sound, secure, and maximum. Intuitively, the algorithm is sound if the answer returned by it is consistent with the answer when there is no fine grained access control. The algorithm is secure if the returned answer does not leak information not allowed by the policy.

The algorithm is maximum if it returns as much information as possible, while satisfying the first two properties.

Our solution consists of a masking approach and a query evaluation algorithm. To strive for the maximum property, we propose a masking approach that makes use of one type of variable rather than NULL to mask unauthorized information. Based on this masking approach, we propose a query evaluation algorithm, and prove that it is both secure and sound, and returns as much information as existing approaches. Finally, we show that our algorithm can be implemented in existing DBMS, such as Oracle, using query modificatio

2. LITERATURE SURVEY

A. Existing Approach

When a query k is issued, the evaluation of k is subjected to a cell-level disclosure policy, which specifies the content of which cells in a table may be used to answer k. To answer k without violating the cell-level disclosure policy r, one first generates masked versions of related tables by replacing all the cells that are not allowed to be seen by r with NULL. After that, k is evaluated as a normal a query on the masked versions of the tables with evaluation rules "NULL ≠ NULL" and "NULL ≠ c" for any constant value c.

The above approach violates the sound property and the maximum property. When a query contains any negation, as expressed using the keywords MINUS, NOT EXISTS or NOT IN, this approach returns incorrect results, violating the sound property. To see this, we consider the following example.

EXAMPLE: We have a relation "Customer", with four attributes: id, name, age, and phone, where id is the primary key (Fig (a)). We mark each cell with "(Y)" or "(N)", indicating whether the cell is allowed by the policy.

Now consider the query k="(SELECT name, phone FROM Customer) MINUS (SELECT name, phone FROM Customer WHERE age ≥25)". Let k_1 = "SELECT name, phone FROM Customer" and k_2 = "SELECT name, phone FROM Customer WHERE age≥ 25". We have k = k_1-k_2. Under the cell-level disclosure policy specified on "Customer", the above algorithm would answer the query k with the relation in Fig (d) which includes Nick, whose age is not in the correct answer. To see that the answer to k is such a relation, observe that the answer to the sub-query k_1 is the relation in Fig (b), and since the age of Nick cannot be accessed, the answer to the sub-query k_2 is the relation in Fig (c).

It violates the maximum property in several ways. First, when the primary key of a record is not allowed to be seen, then no query can be answered, even for queries that project out that field.

ID	name	Age	Phone
C001(Y)	Linda (Y)	32 (Y)	111-1111(Y)
C002(Y)	Mary (Y)	29 (Y)	222-2222(Y)
C003(Y)	Nick (Y)	34 (N)	333-3333(N)
C004(Y)	Jack (Y)	21 (Y)	444-4444(Y)
C005(Y)	Mary (Y)	30 (Y)	555-5555(N)

a. Customer

Name	Phone
Linda	111-1111
Mary	222-2222
Nick	NULL
Jack	444-4444
Mary	NULL

b. Result of k_1

Name	Phone
Linda	111-1111
Mary	222-2222
Mary	NULL

c. Result of k_2

Name	Phone
Nick	NULL
Jack	444-4444

(d) Result of k_1–k_2

Name	Phone
Jack	444-4444

(e)Result without access control policy

Secondly, the approach loses information even if a selection with a trivial condition is performed on a table. Suppose, the query K_3 = "SELECT name FROM Customer WHERE phone = phone". One would expect that names of all people in "Customer" are returned. However, the approach mentioned above only returns three tuples whose phone numbers are viewable.

3. ANALYSIS AND RESEARCH

3.1 Basic Theory of Work

For achieving our goal we will make our system secure, sound and maximum.

To make it secure we will implement following relational algebra expression in query evaluation plan:

- A query processing algorithm F is **sound** if and only if:

$$\forall_r \forall_k \forall_v f(v, r, k) \subseteq S(v, k)$$

The algorithm f takes as input a database v, a disclosure policy r, and a query k, and outputs a result res = f(v, r ,k). Here, the policy r specifies what information in v may be disclosed to answer query k.

We require that f returns only correct information. Let S denote the standard query answering procedure and S(v, k) the result of answering the query k when the database state is v and there is no fine-grained access control policy. Because r restricts access to v f(v, r, k) may return less information than S(v, k) does, but it should not return wrong information. In particular, if a tuple is not in S(v, r), then it should not be in f(v, r, k) either.

- A query processing algorithm f is **secure** if and only if:

$$\forall_r \forall_k \forall_v \forall_{v'} [(v \equiv_P v') \rightarrow (f(v, r, k) = f(v', r, k)]$$

We requires that f(v, r, k) use only information allowed by r i.e. f(v, r, k) should not depend on any information not allowed by r. We observe that the policy r defines an equivalence relation among all database states. Furthermore, if f(v, r, k) does not depend on information not allowed by r, then by issuing queries and observing the results, one should not be able to tell the difference among states that are equivalent under r. The above property says that if the algorithm f is used for query processing, then when v and v' are equivalent under r, one cannot tell whether the database is in state v or in v', no matter what query one issues.

Suppose that f uses a portion in v that is not allowed by r when answering queries, then one should be able to find a query k such that changing that portion to another value affects the query result (if one cannot find such a query, then one can argue that the portion is not actually used). Let v' be the result of the change and we have f(v, r, k) != f(v', r, k). Because the changed part is not allowed by r, we thus have $v \equiv_r v'$. This violates our security definition.

- A query processing algorithm f is **maximum** if and only if:

$$\forall_{v'}[(v \equiv_r v') \rightarrow (res \subseteq (v', k))] \quad (1)$$

we have $res \subseteq f(v, r, k)$

We require that f returns as much information as possible. To appreciate the importance of this property, we observe that a trivial algorithm that always returns no information (i.e., an empty result-set) would satisfy the sound property and the secure property; however, such an algorithm is clearly unacceptable. We formalize this as follows: A query processing algorithm f is maximum if and only if for each (v, r, k), and for each simply generalized relation R that satisfies the above relation.

The above definition says that f(v, r, k) should contain at least as much information as any res that is acceptable as an answer for v ,r ,k. Note that when (1) above is true, one can return res as an answer for query k under any v' such that $v \equiv_r v'$. Because $res \subseteq S(v', k)$, returning res is always sound. As the same res is returned for each v' i.e. equivalent with v with respect to r, returning it under state v is secure.

3.2 A Secure and Sound Query Algorithm

We now proposed a sound and secure query algorithm. In order to preserve useful information for query evaluation, we proposed a type of variable to label unauthorized cells. An unauthorized cell, rather than being replaced by NULL, will be replaced by a variable of type-1. A type-1 variable is a symbol in some alphabet. Given two different type-1 variables v_1, v_2, "$v_1 = v_1$" is true, while "$v_1 = v_2$" and "$v_1 = c$" are unknown, where c is a constant value.

We need to define the semantics of query operators differently depending on whether it is inside the scope of a difference operator.

For a single query k, to ensure soundness, the result of k should return no more information than running k over the unmasked version of a database. Therefore, we need to treat an unauthorized cell in a conservative way so that the query result only contains tuples that are definitely correct. We call this evaluation a low evaluation of k, denoted as k_-. However, if k is on the right hand side of the difference operator, for example, k' - k, the result of k needs to also contain those tuples that are possibly correct. Only by doing so we can ensure the soundness of the whole query k' - k. We call this evaluation a high evaluation of k, denoted as k^-.

Applying the sound and secure query evaluation algorithm to the example. We consider the relation "Customer" and the query k as given in the previous example again. The masked version of "Customer" is shown in Fig (a) below. Let k_1 = "SELECT name, phone FROM Customer" and k_2 = "SELECT name, phone FROM Customer WHERE age $\geq$ 25". We have $k = k_1 - k_2$. In order to ensure soundness, we compute $k_- = k_1 -_a k_2$, which requires the knowledge of k_{1-} and k_2^- . The answers to k_{1-} and k_2^- are presented in Fig (b)

and Fig (c) respectively. In particular, tuple (Nick, v_2) is included in the answer to k^-_2 . The answer to k_- is {(Jack, 444-4444)} (Fig (d)), which is sound.

ID	Name	Age	Phone
C001	Linda	32	111-1111
C002	Mary	29	222-2222
C003	Nick	v_1	v_2
C004	Jack	21	444-4444
C005	Mary	30	v_3

(a)Masked version of customer

(b)Result of k_{1-}

Name	Phone
Linda	111-1111
Mary	222-2222
Nick	v_2
Mary	v_3

b. Result of k2

Name	Phone
Linda	111-1111
Mary	222-2222
Nick	v_2
Jack	444-4444
Mary	v_3

Name	Phone
Jack	444-4444

(d) $k_- = k_1 -_a k_2$

4. FINDINGS

(i). It can be implemented with the business applications of ERP and OLAP oriented database system.
(ii). It will give enhanced security to the system.
(iii). It can be used to implement a better query evaluation algorithm for the database presently used
(iv). It will give better information management.

5. CONCLUSION

In the work our main focus was to develop a sound, secure and maximum query evaluation plan that support fine grained access control technique on all type of widely used database system used now a days. Our work is limited to read access control only. This technique may or may not work on write access control. Fine grained access control for update and deletion will be done in our future work. Major point to be kept in mind is that we can implement our algorithm on only such database software that support basic set operators used in relational algebra.

REFERENCES

1. www.wikipedia.com/ fine grained access control
2. www.w3schools.com/databaseT.Veerarajan, Discrete Mathematics
3. Database Management System, Korth Govind Kabra, Ravishankar Ramamurthy, S. Sudharshan,
4. Fine Grained Access Control Arup Nanda, Proligence, Inc.
5. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In Proceedings of the 24th International Conference on Very Large Databases. ACM Press, Aug. 2002
6. M. Stonebraker and E. Wong. Access control in a relational database management system by query modification. In Proceedings of the 2004 Annual Conference(ACM/CSC-ER), pages 180–186. ACM Press, 2004.
7. LeFevre, R. Agrawal, V. Ercegovac, R. Ramakrishnan, Y. Xu, and D. DeWitt. Limiting disclosure in Hippocratic databases. In Proceedings of the 30th InternationalConference on Very Large Data Bases (VLDB), Aug. 2004
8. S. Rizvi, A. Mendelzon, S. Sudarshan, and P. Roy. Extending query rewriting techniques for fine-grained access control. In Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, pages 551–562, Paris, France, 2004. ACM Press.