Fine-Grained Access Control for Database Management Systems*

Hong Zhu[1] and Kevin Lü[2]

[1] Huazhong University of Science and Technology, Wuhan, Hubei, 430074, P.R. China
[2] Brunel University, Uxbridge, UK UB8 3PH

Abstract. A practical approach for developing fine-grained access control (FGAC) for database management systems is reported in this paper. We extend SQL language to support security policies. The concept of the *policy type* for databases is proposed. We implement the policy reuse through the use of policy types and policy instances to alleviate the administration workload of maintaining security policies. The policies for rows and columns can be expressed with policy types. Moreover, complicated database integrity constraints can also be expressed by policy types, and no further purpose-built programs are needed to create specific security control policies. We implement the fine-grained access control in a relational database management system DM5 [4]. The performance test results based on TPC-W are also presented.

1 Introduction

With the wide integration of Internet and database technology, the resources in information systems have been shared by more and more users. The first feature of these systems is that the number of users is enormous, but the number of roles for users is relatively small. The second feature is that different users with the same role may access different data sets. When more and more data is stored in database systems, privacy and security become important issues in these systems. The privacy and security concerns mean that Internet-based information systems must provide fine-grained access control for users, and, in many cases, even single user-based access control is needed[1][2][8].

However, the current standard SQL language for access control is coarse grained, in that it grants access to all rows of a table or none at all. It provides access controls on rows on a table by views. The access control by views is suitable for applications with a fixed small number of users. It is not applicable to the cases with a large number of users, such as in the Internet environment. Also, with this method, the administrators have to manage and maintain many views; this aggravates the workload. For a long time, the fine-grained access control has been embedded in application programs according to the requirements of applications. This approach has several disadvantages. The first is that access control has to be checked at every

* This paper is supported by 863 hi-tech research and development program of China, granted number: 2006AA01Z430.

R. Cooper and J. Kennedy (Eds.): BNCOD 2007, LNCS 4587, pp. 215–223, 2007.

user-interface. This increases the overall code size. The second is that the access control in the application tier can be bypassed easily. The third is that it is easy for application programmers to create trap-doors with malicious intents, because it is almost impossible to check every line of code in a large application. For the above reasons, fine-grained access control should be enforced for database management systems.

This study focused on the issues of constructing FGAC in database systems. The main contributions of this paper are:

(1) The concept of *policy type* for databases is proposed. We extend SQL statements to support the security policy type. Security policy instances based on the security policy type can be created to express different security policy requirements. Moreover, complicated integrity constraints in database can be expressed by policy types to specify the condition of the policy to take effect.

(2) We implement the FGAC in the relational database management system DM5. The performance evaluations based on TPC-W benchmark have been conducted.

This paper is organized as follows: Section 2 describes the related work. Section 3 presents the detailed extension of SQL statements. Section 4 reports performance tests based on TPC-W and the results. Section 5 presents a summary.

2 Related Work

The first FGAC access control model was proposed by M. Stonebraker in Ingres system [9]. It is implemented by a "query modification" algorithm. But the algorithm does not handle rows and columns symmetrically. The Virtual Private Database (VPD) in ORACLE [11] has provided a PL/SQL procedure function to describe the security policy. Nevertheless, writing policy functions corresponding to business policies requires a large amount of work [6]. Also, it is difficult to write predicates involving cases of cross-ref, joins of tables etc., and no element level security in VPD is presented.

Based on the access control in System R and Ingres, Motro proposed an FGAC model for database based on algebraic manipulation of view definitions [7]. That model has some limitations [8]. The costs of the query optimiser in DBMS are high. The Non-Truman model [8] is based on authorization views and a validity notion of queries. Using an inference mechanism, when a query is submitted from a user, the query is evaluated and rewritten according to the accessible authorization views. However, in the worst case, the algorithm may not be able to infer validity of some unconditionally valid queries.

Recently, work on the policy for preserving privacy has boosted the research of FGAC [1, 3]. Elisa Bertino et al [3] proposed a privacy preserving access control model for relational databases. Actually, the model proposed needs a basis of FGAC in a database system. Nevertheless, they did not describe how to implement the model in a database management system. Agrawal et al [1, 2] proposed a framework for FGAC implementation in a Hippocratic database system. In their work, the DBMS