Chapter 5.5
Fine-Grained Data Security in Virtual Organizations

Harith Indraratne
Budapest University of Technology and Economics, Hungary

Gábor Hosszú
Budapest University of Technology and Economics, Hungary

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. In some scenarios, however, 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.

In case of creating the virtual networking infrastructure of virtual organizations, the security of the data stored in database management systems is a very important issue. Several models have been proposed by research community and database vendors for specifying and enforcing row-level access control at the database layer. This article reviews the most important facts of some significant *FGAC* models and current implementations of such in two commercial database management systems. We describe a novel concept of implementing *FGAC* in *SQL Server 2005*, which resembles *Oracle 10g* database management system's *FGAC* solution *virtual private databases* (VPD).

BACKGROUND

Modern database applications, with large numbers of users, require FGAC mechanisms at the level of individual tuples, not just entire relations/views, to control which parts of the data can be accessed by each user. Consider the following scenario:

In a commercial organization's human resources database, the human resources manager should have access to all the personal details of employees. At the same time, individual employees should only be able to see their particulars, not other employees' information.

In the above case, authorization is required at a very fine-grained level, such as at the level of individual tuples. Similar scenarios exist in many environments, including finance, law,

government, and military applications. Consumer privacy requirements are yet another emerging driver for finer control of data.

Currently, general data authorization mechanisms in relational databases permit access control at the level of complete tables or columns, or on views. There is no direct way to specify fine-grained authorization to control which tuples can be accessed by users. In theory, FGAC at the level of individual tuples can be achieved by creating an access control list for each tuple. However, this approach is not scalable (Utkarsh, 2004) and would be totally impractical in systems with millions of tuples and thousands or millions of users, since it would require millions of access control specifications to be provided (manually) by the administrator. It is also possible to create views for specific users, which allow those users access to only selected tuples of a table, but again this approach is not scalable with large numbers of users.

On some occasions, FGAC is often enforced in the application code, which has numerous drawbacks; these can be avoided by specifying/enforcing access control at the database level. Current information systems typically bypass database access control facilities and embed access control in the application program used to access the database. Although widely used, this approach has several disadvantages, such as access control has to be checked at each user interface. This increases the overall code size. Any change in the access control policy requires changing a large amount of code. Further, all security policies have to be implemented into each of the applications built on top of this data. Also, given a large application code, it is possible to overlook loopholes that can be exploited to break through the security policies, for example, improperly designed servlets. Also, it is easy for application programmers to create trap-doors with malicious intent, since it is impossible to check every line of code in a very large application (Rizvi, Mendelzon, Sudarshan, & Roy, 2004).

Fine-grained access control methods based on query modification approaches such as *Oracle VPD* have their drawbacks. Specifically implementing policies on improperly designed tables may result in inconsistent query results and unanticipated execution times. Proper database design and use of indexes for predicate values may overcome these drawbacks.

For the above reasons, FGAC should ideally be specified and enforced at the database level. Today, both *Oracle 10g* and *SQL Server 2005* have captured the attention of database community because of the new exciting database features included in their latest releases (Gornshtein & Tamarkins, 2004). In this article, we present a FGAC security model for SQL Server 2005 similar to the FGAC method implemented in Oracle 10g as Oracle VPD.

FINE-GRAINED ACCESS CONTROL IN ORACLE 10G

The model implements by Oracle (Oracle, 2005) for fine-grained access controls is called *virtual private database* and restricts the rows a user sees based on the user's credentials (Loney, 2004). *Oracle Database 10g* VPD introduces column relevant security policy enforcement and optional column masking. These features provide tremendous flexibility for meeting privacy requirements and other regulations (Needham & Iyer, 2003). As presented in Figure 1, VPD restriction is enforced by a WHERE clause automatically appended to the original query based on the *application context* information gathered at the user log-on time. This clause, called a *predicate*, is generated by a user-defined function called a *policy function*.

The FGAC can be used within database settings to enable multiple users or applications utilizing the same database to have secure access to data. It enables per-user data access within a single database with the assurance of physical data separation. It is enabled by associating one