International Journal of Emerging Engineering Research and Technology
Volume 2, Issue 5, August 2014, PP 146-152
ISSN 2349-4395 (Print) & ISSN 2349-4409 (Online)

Implementing Virtual Private Database Using Security Policies

H. Lakshmi[1], T. Sukanya [2], A. Pathanjali Sastri[3]

[1]Persuing M.Tech (CSE), M.V.R.College of Engg. & Technology, Paritala, India
[2]Asst. Professor, Dept. of CSE, M.V.R.College of Engg. & Technology, Paritala, India
[3]Asst. Professor, Dept. of Computer Applications, V.R.Sidhhartha Engg. College, Vijayawada

Abstract: *Database management systems are increasingly being used to store information about all aspects of an enterprise. Like all tangible assets that have to be protected by a company, valuable information stored in a DBMS is often vital to the business interests of the organization and is regarded as a corporate asset of the company that must be protected. In addition to protecting the intrinsic value of the data, corporations must consider ways to ensure privacy and to control access to data that must not be revealed to certain groups of users for various reasons.*

Database security is the degree to which all data is fully protected from tampering or unauthorized acts. In this paper, we survey different techniques and approaches to assure data Secrecy, Integrity, and Availability. This paper focused on Virtual private database, allows fine-grained access control down to the tuple level based on the use of predicates. Database returns customized results to each user using Virtual Private Database.

Keywords: *Virtual Private Database, Data Secrecy, Fine-grained access control and Policies*

1. INTRODUCTION

IT is today well understood that databases represent an important asset for many applications and thus their security is crucial. Data confidentiality is particularly relevant because of the value, often not only monetary, that data have. For example, medical data collected by following the history of patients over several years may represent an invaluable asset that needs to be adequately protected. Such a requirement has motivated a large variety of approaches aiming at better protecting data confidentiality and data ownership. Relevant approaches include risk management, database encryption, and authenticity techniques.

Database security concerns the use of a broad range of information security controls to protect databases includes the data, the database applications or stored functions, the database systems, the database servers against compromises of their Secrecy, integrity and availability.

1.1 Organization of the Paper

The remainder of the paper is organized as follows: Section 2 discusses Security levels of Database. Section 3 focuses VIEW- Based security mechanisms that protect data from unauthorized users. Section 4 presents an overview of Virtual Private database concepts and views as a solution and its limitations to VPD. Section 5 describes a solution based on polices. Finally, Section 6 presents Limitations and conclusion.

2. DATABASE SECURITY LEVELS

As database technology moves into new applications areas and are made accessible via the Internet and web-based applications, their exposure to security threats will rise, the Security requirements changes and become more demanding. The objective is to reduce susceptibility to these threats. A

relational database is a database that conforms to the relational model, and refers to a database's data and schema (the database's structure of how those data are arranged). Relational databases have become the first choice for the storage of the tabular information that supports the world economy, personnel data and much more. Most commercial RDBMS's use the Structured Query Language (SQL) to access the database, although SQL was invented after the development of the relational model and is not necessary for its use. As there are many Relational DBMS's, in this paper we mainly focuses on Oracle.

A relational database is a collection of data files; Data file is a collection of related relations (table); a relation is a collection of related rows; and a row is a collection of related columns, the columns of a relation are referred to as attributes. The degree of a relation is the number of attributes defined for that relation. The rows of a particular relation are referred to as tuples. The cardinality of a relation is just the number of (unordered) tuples it contains. In a particular tuple, the field which corresponds to a particular attribute may contain any one of the values in the domain of that attribute as shown in the fig.1.

The structure of database is organized in levels, and each level can be protected by different security mechanisms. For instance, a column can be protected by using a VIEW database object. A VIEW database object is a stored query that returns columns and rows from the selected tables. The data provided by the VIEW object is protected by the database system functionality that allows schema owners to grant or revoke privileges.

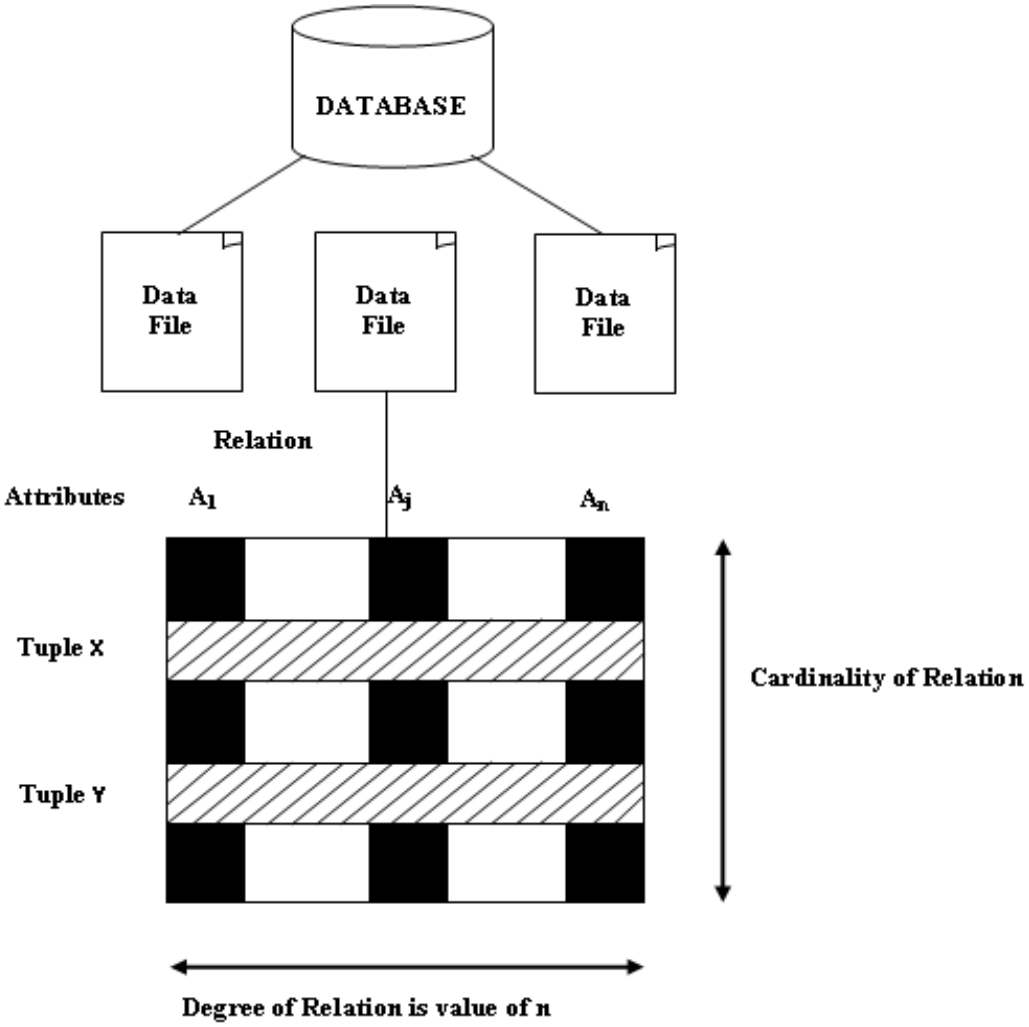

Fig1. *Levels of Database*

Data files in which data resides are protected by database and that protection is enforced by operating system file permissions. Finally, the database is secured by the database management system through the use of user accounts and password mechanisms as well as by the privileges and permissions of main database functions – database shutdown, creating user accounts, and database backup and recovery.

3. VIEW – BASED ACCESS CONTROL

In a relational DBMS, a view is defined as a "virtual table" derived by a specific query on one or more base tables. The relational operations join, restrict and project as well as statistical summaries of tables may be used to define a view. Access rights may be granted and revoked on views just as though they were ordinary tables. This allows users to selectively share data, preventing unauthorized users from reading sensitive information. A view is a way of building even greater abstraction.

3.1 Protecting Privacy using Views

A common use of views is protecting confidential data. For example, suppose that all the people who work in a hospital collaborate by using a relational database. Here is the data model:

```
CREATE TABLE PATIENTS_HIV_INS (
    PATIENT_ID     NUMBER (10) PRIMARY KEY,
    PATIENT_NAME   VARCHAR2 (100),
    HIV_POSITIVE CHAR (1),
    INSURANCE_P    CHAR (1),
    DOCTOR_ID NUMBER (10));
```

If a bunch of hippie idealists are running the hospital, they'll think that the medical doctors shouldn't be aware of a patient's insurance status. So when a doctor is looking up a patient's medical record, the looking is done through,

```
CREATE VIEW PATIENTS_CLINICAL
AS
SELECT PATIENT_ID, HIV_POSITIVE FROM PATIENTS;
```

The staff over in accounting shouldn't get access to the patients' medical records just because they're trying to squeeze money out of them:

```
CREATE VIEW PATIENTS_ACCOUNTING
AS
SELECT PATIENT_ID, PATIENT_NAME, INSURANCE_P FROM PATIENTS;
```

Relational databases have elaborate permission systems similar to those on time-shared computer systems. Each person in a hospital has a unique database user ID. Permission will be granted to view or modify certain tables on a per-user or per-group-of-users basis. Generally the RDBMS permissions facilities aren't very useful for Web applications. It is the Web server that is talking to the database, not a user's desktop computer. So the Web server is responsible for figuring out who is requesting a page and how much to show in response. View – Based Access control [5] enables Content – Based and Context – Based Security.

4. VIRTUAL PRIVATE DATABASE

Virtual Private Database (VPD) enables you to create security policies to control database access at the row and column level. Essentially, Virtual Private Database adds a dynamic WHERE clause to a SQL statement that is issued against the table, view, or synonym to which a Virtual Private Database security policy was applied. You can apply Virtual Private Database policies to SELECT, INSERT, UPDATE, INDEX, and DELETE statements. VPD enables administrators to define and enforce row-level access control policies based on session attributes using two features called **Fine – grained access control**: associate security policies to database objects **Application Context**: define and access application or session attributes.

4.1 Implementing a VPD Using Views And Its Limitations

Loss prevention of data and in particular protection of data from unauthorized accesses remains important goal of any database management system. While VIEW can provide fairly granular access control, they have limitations which make them less than optimal for very fine-grained access control. VIEWS are not always practical when user need a lot of them to enforce user policy. In our example Health Clinic management has 7 departments so we created 7 VIEW objects; this becomes overhead when size of organization is very large that is an organization has hundreds of departments. In case of Column – Level Security, maintaining triggers on VIEW objects for Insert Operation is extra burden on DBA. While applications may incorporate and enforce security through views, users often need access to base tables to run reports or conduct ad-hoc queries. Users who have privileges on base tables are able to bypass the security enforcement provided by views. Note that this is a general problem of embedding security in applications instead of enforcing security through database mechanisms, but it is exacerbated when security is enforced on views and not on the data itself. VPD using POLICIES provides a flexible mechanism for building applications that enforce the security policies only where such control is necessary by dynamically appending SQL statements with a predicate, VPD limits access to data at the Row Level and ties the security policy to the table itself.

4.2 Application context

Application context is functionality specific to Oracle that allows user to set database application variables that can be retrieved by database sessions. These variables can be used for security context – based or user – defined environmental attributes. User can identify client host name, an IP address of the connected session, or the operating system user name of a connected session using application context function SYS_CONTEXT in conjunction with predefined user-environment attributes, known as USERENV attributes, which are grouped as a namespace.

Example:

SQL> SELECT SYS_CONTEXT ('USERENV', 'CURRENT_USER') FROM DUAL;

SYS_CONTEXT ('USERENV','CURRENT_USER')

SYSTEM

The database session – based application context is managed entirely within Oracle Database. Oracle Database sets the values, and then when the user exits the session, automatically clears the application context values stored in cache. Any application that accesses this database will need to use this application context to permit or prevent user access to that application.

Application contexts are useful for the following purposes:

- Enforcing fine-grained access control
- Preserving user identity across multitier environments
- Serving as a holding area for name-value pairs that an application can define, modify, and access

4.3 Fine – Grained Access Control/Policy – Based Access control

Fine – Grained mechanisms supports access control down to the tuple level. The conventional view mechanisms have a number of shortcomings. A naïve solution to enforce fine – grained authorization would require the specification of a view for each tuple or part of a tuple that is to be protected. Moreover, because access control policies are often different for different users, the number of views would further increase. Furthermore, application programs would have to code different interfaces for each user, or group of users, as queries and other data management commands would need to use for each user, or group of users, the correct view. Modifications to access control policies would also require the creation of new views with consequent modifications to application programs. Alternative approaches that address some of these issues have been proposed, and these approaches are based on the idea that queries are written against the base tables and, then, automatically rewritten by the system against the view available to the user. These approaches do not require that we code different interfaces for different users and, thus, address one of the main problems in the use of conventional view mechanisms.

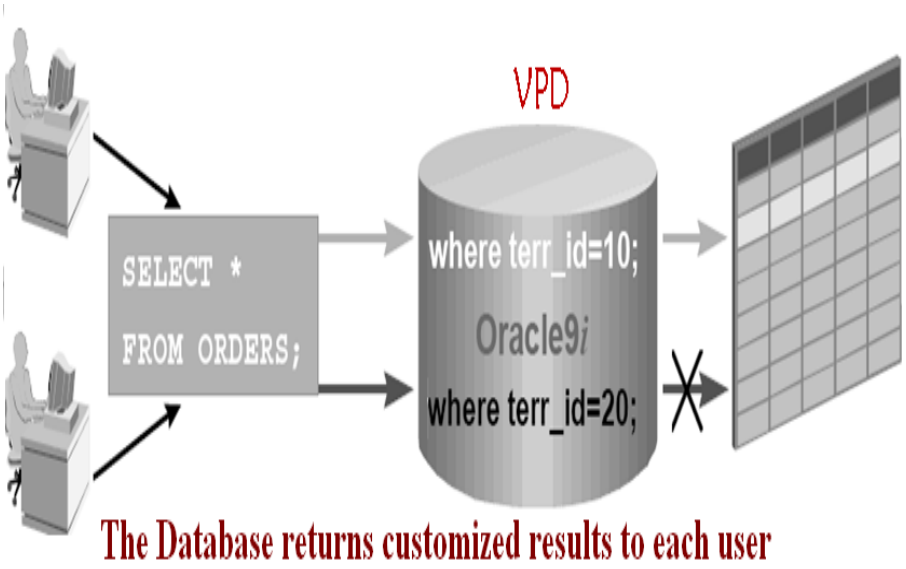

Fig2. *Fine – grained access control*

5. POLICIES AS A SOLUTION

Virtual Private Database enforces security, to a fine level of granularity, directly on database tables, views, or synonyms. Because you attach security policies directly to these database objects, and the policies are automatically applied whenever a user accesses data, there is no way to bypass security. When a user directly or indirectly accesses a table, view, or synonym that is protected with a Virtual Private Database policy, Database dynamically modifies the SQL statement of the user. This modification creates a WHERE condition (called a predicate) returned by a function implementing the security policy. Database modifies the statement dynamically, transparently to the user, using any condition that can be expressed in or returned by a function.

This example allows specific users to see only the rows of patients who has HIV negative i.e., it restricts the user to access tuples of the patients who are suffering with HIV.

Policy Function which returns predicate to the policy:

CREATE OR REPLACE FUNCTION PATIENT_HIV (P_SCHEMA IN VARCHAR2, P_OBJECT IN VARCHAR2)

RETURN VARCHAR2

AS

BEGIN

RETURN ' HIV_POSITIVE! =1 ';

END;

Syntax for adding a Policy:

BEGIN

DBMS_RLS.ADD_POLICY

("Object schema name",

"Object name",

"Policy name",

"Function schema name",

"Function name",

"Statement type");

END;

6. CONCLUSION

We asserted that current access control models do not achieve the crucial goals that we set out with. We described two models for fine-grained access. Both models support authorization-transparent querying. Unlike the VIEW- Based, the Policy- Base model avoids the pitfalls of the query modification approach and allows a great deal of flexibility in authorization, such as authorization of aggregate results. We outlined an approach to validity testing, based on extending an existing query optimizer to carry out validity checking, minimizing the extra effort required during coding as well as during validity testing. VPD using POLICIES provides a flexible mechanism for building applications that enforce the security policies only where such control is necessary by dynamically appending SQL statements with a predicate, VPD limits access to data at the Row Level and ties the security policy to the table itself. Like a two sides of coin, policies has pitfalls. It is difficult, if not impossible, to verify whether or not a particular user has access to a particular data item in a particular table in a particular state.

– Such verification requires checking all policy functions.

– As policy functions are too "flexible", it is computationally impossible to analyze them.

REFERENCES

[1] Simon Liu and Rick Kuhn, "Data Loss Prevention", Published by the IEEE Computer Society ©2010 IEEE

[2] Meg Coffin Murray, "Database Security: What Students Need to Know", Journal of Information Technology Education Volume 9, 2010

[3] Sohail IMRAN, "Security Issues in Databases", 2009 Second International Conference on Future Information Technology and Management Engineering

[4] Ravi Sandhu, David Ferraiolo and Richard Kuhn,"The NIST Model for Role - Based Access Control: Towards a Unified Standard."

[5] Xiaolei Qian, Computer Science Laboratory, SRI International "View - Bases Access Control with High Assurance."

[6] Ravi S. Sandhu and Sushil Jajodia, " Data and Database Security and Controls", Handbook of Information Security Management, Auerbach Publishers, 1993

[7] The virtual private database in oracle9ir2: An oracle technical white paper.

[8] http://www.cgisecurity.com/database/oracle/pdf/VPD9ir2twp.pdf

[9] E.Bertino, L.M. Haas, B.G.Lindsay, View Management In Distributed Data Base Systems

[10] Surajit Chaudhuri, Raghav Kaushik, Ravi Ramamurthy, Microsoft Research, "Database Access Control & Privacy: Is There a Common Ground?"

[11] B. Lakshmi, K. Parish Venkata Kumar, A. Shahnaz Banu and K. Anji Reddy, "Data Confidentiality and Loss Prevention using Virtual Private Database."

AUTHORS' BIOGRAPHY

Mrs. H. Lakshmi is persuing M.Tech (CSE) at M.V.R.College of Engineering and Technology, Paritala. Her areas of interest include Database Management Systems, Data Mining, Computer Networks, and Artificial Intelligence. She had received M.C.A from Acharya Nagarjuna University, Ratified under Nagarjuna and JNTUK, Kakinada. She had completed the OCA certification. She is a Member of CSI and IEEE.

Miss. T. Sukanya has 4 years experience in teaching and working as an Asst. Professor in MVR College of Engineering and Technology. Her areas of interest are operating sytems, Data structures, and Database management systems. She has published a paper in International Journal of Mobile and Adhoc Netwok.

Mr. A. Pathanjali Sastri is currently pursuing Ph.D Computer Science from Raylaseema University, Kurnool. He is woking as a Lecturer in Velagapudi Ramakrishna Siddhartha Engineering college since 2008 and has 10 years of Industrial experience. He has published papers in reputed journals/International conferences recently. His area of interest includes Software Engineering, Quality Assurance, Artificial Intelligence and RDBMS.