

University of Montana

## ScholarWorks at University of Montana

---

Graduate Student Theses, Dissertations, &  
Professional Papers

Graduate School

---

2001

### Computer database security and Oracle security implementation

Mei Ke

*The University of Montana*

Follow this and additional works at: <https://scholarworks.umt.edu/etd>

**Let us know how access to this document benefits you.**

---

#### Recommended Citation

Ke, Mei, "Computer database security and Oracle security implementation" (2001). *Graduate Student Theses, Dissertations, & Professional Papers*. 5092.  
<https://scholarworks.umt.edu/etd/5092>

This Thesis is brought to you for free and open access by the Graduate School at ScholarWorks at University of Montana. It has been accepted for inclusion in Graduate Student Theses, Dissertations, & Professional Papers by an authorized administrator of ScholarWorks at University of Montana. For more information, please contact [scholarworks@mso.umt.edu](mailto:scholarworks@mso.umt.edu).



**Maureen and Mike  
MANSFIELD LIBRARY**

The University of  
**Montana**

---

Permission is granted by the author to reproduce this material in its entirety,  
provided that this material is used for scholarly purposes and is properly cited in  
published works and reports.

**\*\*Please check "Yes" or "No" and provide signature\*\***

Yes, I grant permission     X    

No, I do not grant permission           

Author's Signature: Mei Ke

Date: 06/29/2001

Any copying for commercial purposes or financial gain may be undertaken only with  
the author's explicit consent.

---

# **Computer Database Security and Oracle Security Implementation**

by

**Mei Ke**

M.S., Hefei University of Technology, 1994

B.S., Hefei University of Technology, 1991

Presented in partial fulfillment of the requirements

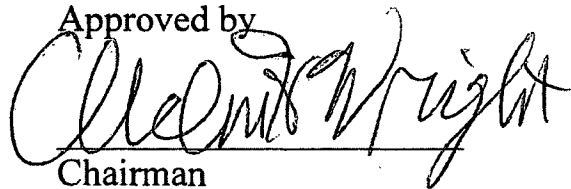
for the degree of

Master of Science

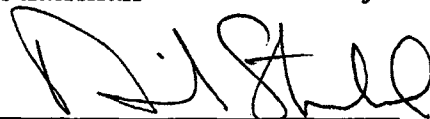
The University of Montana

June 2001

Approved by

A handwritten signature in black ink, appearing to read "Clinton Wright", written over a horizontal line.

Chairman

A handwritten signature in black ink, appearing to read "D. J. Stuehl", written over a horizontal line.

Dean, Graduate School

7.12.01

Date

UMI Number: EP40556

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.

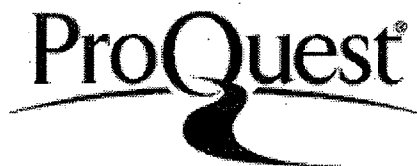


UMI EP40556

Published by ProQuest LLC (2014). Copyright in the Dissertation held by the Author.

Microform Edition © ProQuest LLC.

All rights reserved. This work is protected against unauthorized copying under Title 17, United States Code

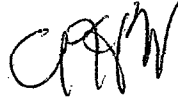


ProQuest LLC.  
789 East Eisenhower Parkway  
P.O. Box 1346  
Ann Arbor, MI 48106 - 1346

## **Computer Database Security and Oracle Security Implementation**

**(84 pp.)**

Advisor: Dr. Alden H. Wright



Today organizations are using computers to support material areas of their activities: controlling warehouses, assisting in production design, controlling vital processes, improving clerical productivity as well as traditional accounting. Most of these computers store the data necessary for their activity in the form of databases. Therefore, the issue of database security is increasingly a matter of concern to all organizations. An organization needs to ensure that its valuable data is not exposed to unauthorized individuals or corrupted by hostile parties. The goals in database security relate to how well the system achieves these security goals, and the relationship between mechanisms provided by the operating system and those provided by the database system, and architecture of a secure Database Management System. This paper is a result of research on Database System Security. The security properties that need to be considered during the definition and design phase of a Database System are detailed. A modeling, technologies and methodological framework for security system development in Database Systems is summarized. As a Database Administrator working on an Oracle Database Management System in a software company, I feel that database security is essential to ensure system continuity and reliability and to protect data and programs from intrusions, modifications, theft and unauthorized disclosure. This paper also includes some research on a real Database Security System: the Oracle Database Security System. Issues involving the Oracle database security system in practice include establishing an organization's security policy and plan, protecting system files and users' passwords, controlling access to the database objects including tables, views, rows, columns, packages, building appropriate user roles, privileges and profiles, developing appropriate backup and recovery strategies, etc.

## Acknowledgments

I would like to express my deepest gratitude to my advisor, Professor Alden Wright, whose advice, constructive criticism, and encouragement made it possible to develop and complete this work.

I am very grateful to the committee members, Professor Jerry Esmay and Professor Tomas Tonev. They both advised me and encouraged me during my graduate studies.

I thank Kathy Lockridge for her excellent administrative support in preparing various paper work for the Graduate School throughout this project.

I am also indebted to my father, my mother, my sister, my grandmother and to my husband. They all offered me unlimited support and constant encouragement when I needed it.

Finally, I would like to thank all professors that I took classes from in the Department of Computer Science at the University of Montana.

# Table of Contents

<b>Abstract.....</b>	<b>ii</b>
<b>Acknowledgments .....</b>	<b>iii</b>
<b>List of Figures.....</b>	<b>vi</b>
<b>List of Tables .....</b>	<b>vi</b>
<b>1 Introduction .....</b>	<b>1</b>
1.1 Overview.....	1
1.2 Proposal.....	3
<b>2 Database Security and Models.....</b>	<b>6</b>
2.1 Database Management System .....	6
2.1.1 Concepts of Database Management System .....	6
2.1.2 Data Description Levels.....	9
2.2 Database Security .....	10
2.2.1 Security Problems in Databases .....	10
2.2.2 Database Security Protection Requirements.....	12
2.3 Secure Database .....	15
2.4 Database Security Controls.....	17
<b>3 Designing Database Security.....</b>	<b>23</b>
3.1 Secure DBMS Design.....	23
3.2 Security Models in DBMS.....	25
3.2.1 Elements of DBMS Security Models.....	26
3.2.2 The Wood Model .....	28

3.2.3	The Dion Model.....	36
3.2.4	The RBAC96 Model.....	44
3.3	The Architecture of Secure DBMSs .....	48
<b>4</b>	<b>A Real Case: Oracle Security Database Implementation .....</b>	<b>51</b>
4.1	Oracle and Security.....	51
4.2	Security in an Oracle System.....	53
4.2.1	Oracle System Files and Database Objects.....	55
4.2.1.1	Oracle System Files .....	56
4.2.1.2	Oracle Database Objects .....	60
4.2.2	Oracle Data Dictionary .....	63
4.2.3	Other Elements of Oracle Security System .....	65
4.3	Methodologies of Designing A Secure Oracle Database.....	69
4.3.1	Developing a Database Security Plan.....	70
4.3.2	Developing an Audit Plan.....	72
4.3.3	Backing Up and Recovering the Database.....	76
4.3.3.1	Backing Up the Database System.....	76
4.3.3.2	Recovering the Database System.....	78
<b>5</b>	<b>Conclusions .....</b>	<b>80</b>
5.1	Main Results .....	80
5.2	Future Research Directions.....	81
	<b>Bibliograph .....</b>	<b>83</b>



## List of Figures

<b>Figure 2.1</b> Database Management System Architecture .....	8
<b>Figure 2.2</b> Architecture of a DBMS Including Security Features.....	16
<b>Figure 2.3</b> Access Control System.....	19
<b>Figure 2.4</b> Mandatory Access Control .....	19
<b>Figure 2.5</b> Discretionary Access Control .....	20
<b>Figure 3.1</b> Elements of DBMS Security Models .....	27
<b>Figure 3.2</b> Three-level Database Architecture of The Wood Model .....	30
<b>Figure 3.3</b> Access Control Steps in The Wood Model .....	34
<b>Figure 3.4</b> The RBAC96 Model.....	46
<b>Figure 3.5</b> The Kernelized Architecture .....	49
<b>Figure 4.1</b> Components of the Database System After Startup .....	56
<b>Figure 4.2</b> Oracle View Implementation.....	62
<b>Figure 4.3.1</b> An Example of Role Hierarchy in Oracle DB System .....	67
<b>Figure 4.3.2</b> An Example of Administrative Role Hierarchy in Oracle DB System .....	68
<b>Figure 4.4</b> The Thin Client Architecture in Oracle DB System.....	70

## List of Tables

<b>Table 3.1</b> An Example of an Access Constraint (AC) Table .....	32
<b>Table 3.2</b> An Example of a Conceptual Rules (CR) Table .....	33
<b>Table 3.3</b> An Example of an External Rules (ER) Table .....	35
<b>Table 4.1</b> Oracle Database Data Dictionary Views.....	64
<b>Table 4.2</b> Oracle Default Roles .....	67

# Chapter 1

## Introduction

### 1.1 Overview

Today organizations are using computers to support material areas of their activities: controlling warehouses, assisting in production design, controlling vital processes, improving clerical productivity as well as traditional accounting. Most of these computers store the data necessary for their activity in the form of databases. Database Management Systems (DBMSs) and the database technology have been largely used for these purposes.

*A database is a collection of physical datafiles and DBMS software that manipulates the data [21].* Although the increasingly widespread use of both centralized and distributed databases has proved that the databases are very important to support business functions, it has also posed the serious problem of data security. On an almost daily basis, we hear stories of computer systems that have been damaged by someone – a curious teenager, a disgruntled employee, or a corporate or political spy. We are baffled and bemused by the number of “hackers” who have been able to get into system that have been viewed as invulnerable. For example, government computers, bank accounting systems, and university systems should not be compromised by hackers. But, all of these kinds of systems have been compromised by hackers. It would appear that no computer system is completely safe. In fact, damage in a database environment does not only affect a single user or application but rather the whole organization or the whole information

system. Therefore, in computer-based information systems, technologies, tools and procedures concerning security are essential both to ensure system continuity and reliability and to protect data and information from intrusions, modifications, theft and unauthorized disclosure. The issues of database security are increasingly a matter of concern to all such systems.

*There are many facets to computer security. These aspects of security include security and confidentiality; accuracy, integrity and authenticity; availability and recoverability [20].*

- Ensuring security and confidentiality means that data should not be disclosed to anyone who is not authorized to access it. For example, the patient's medical records are confidential and should not be improperly disclosed.
- Ensuring accuracy and integrity means that data cannot be maliciously or accidentally corrupted or modified. Ensuring authenticity means providing a way to verify the origin of the data. For example, in a military environment, the target coordinates of a missile should not be improperly modified.
- Ensuring availability and recoverability means data can be recovered efficiently and completely without loss of accuracy or integrity in case of loss, and the systems can keep working. For example, in a commercial environment, payment orders regarding taxes should be made on time as fixed by law. Analogously, in a military environment, when the proper command is issued, the missile should fire.

The goals in database security relate to how well the system achieves security goals, the relationship between the security mechanisms provided by the operating

system and those provided by the database system, and the architecture of a secure Database Management System.

## **1.2 Proposal**

The goal of this project is to do research on Database System Security. The security properties that need to be considered during the definition and design phase of a Database System are detailed. The security control (flow control, inference control and access control), the security models (discretionary security models and mandatory security models), and technologies for security system development in Database Systems are summarized. As a Database Administrator working on an Oracle Database Management System in a software company, I feel that database security is essential to ensure system continuity and reliability, and to protect data and programs from intrusions, modifications, theft and unauthorized disclosure. Another goal of this project is to do research on a real Database Security system: the Oracle Database Security System. Issues involving the Oracle database security system include:

- Establishing an organization's security policy and plan;
- Protecting system files and users' passwords;
- Controlling access to the database objects;
- Building appropriate user roles, privileges, views and profiles;
- Developing appropriate backup and recovery strategies.

This paper will include five chapters. Chapter 1 illustrates the importance of computer database security in many organizations, the facets of database security, and the necessity of doing research on this area.

In Chapter 2, the database management system, data description levels and relational data model are briefly described. The security problems (improper release of information; improper modification of data; denial of service; natural or accidental disasters; errors or bugs in hardware or software, and human errors) are examined in this chapter. After examining the existing security problems, the database security protection requirements (integrity; authorization; inference; access control; and confinement) are discussed here. Then, the structure of a Database Management System including the security features is presented. The basic security models for security controls including access control, flow control and inference control are also described and classified in this chapter.

After analyzing the main desirable criteria and requirements, the development process for secure database systems is illustrated in Chapter 3. The policy plan for designing a secure Database Management System is discussed and presented. The basic security models are described and classified in this chapter. First, the elements of DBMS security models are illustrated. Then, several specific security models (The Wood Model, The Dion Model, and The RBAC96 Model) are presented in this chapter. The security models are classified in two traditional categories: discretionary security models and mandatory security models.

As a Database Administrator working in a company using Oracle software, I am doing research on the Database Security and Oracle Security system implementation as part of my current daily work. Chapter 4 describes the security in an Oracle database system including the main files and database objects of special significance to Oracle security, for example, privileges, views, default roles, user accounts, profiles, password

and synonyms. This chapter also illustrates the specific steps to build a secure Oracle database system. The development of a security plan and an audit plan is explained in this chapter. This chapter also discusses the specific strategies for starting up Oracle and backing up and restoring Oracle databases in the most secure manner.

Chapter 5 is the conclusion of the research paper. It summarizes the research effort. The future research directions of Computer Database Security are briefly discussed here. These include the application of computer immune systems to the distributed database systems and the design of multilevel secure database systems.

## Chapter 2

### Database Security and Models

#### 2.1 Database Management System

*A database is a collection of mutually correlated data stored on persistent storage supports [21].* The database refers not only to the physical data but also to the combination of physical, memory and process objects. It is used within an organization by different applications, each one having its own business functions and purposes. In a database, data (entity) and data associations (relationship) that characterize an application are described.

##### 2.1.1 Concepts of Database Management System

A Database Management System (DBMS) is a software system which includes a set of data management functions. The DBMS manages a three-level structure and its necessary interface. The three-level structure includes the conceptual level, the internal level and the external level. Great quantities of data can be accessed rapidly and efficiently through the schema management, concurrent transaction management, data access controls, logging, and recovery of the database. These functions are very useful for both the database and the application management. The Entity-Relationship model is one of the most popular DBMS conceptual models. An entity is a class of real-world objects to be described in the database, and a relationship is a set of modeling associations between two or more entities. The logical model describes the data and the relationships

between data in a DBMS. During logical design, the conceptual entities and relationships are translated into a logical schema, i.e. a data schema. The data schema describes the data and the relationships according to the logical model. The logical model includes relational model, network model and hierarchical model, managed by the DBMS technology. The Data Definition Language (DDL), Data Manipulation Language (DML) and Query Language (QL) are the languages available in a DBMS. DDL is the Structured Query Language (SQL) construct used to define data in the database. DDL supports the definition of the logical database schema. DML is the SQL construct used to manipulate the data in the database. DML and QL support the operations on data that includes data retrieval, insertion, deletion and update. The typical architecture of a DBMS is shown in Figure 2.1.

Figure 2.1 shows the essential parts of a DBMS. The modules of Figure 2.1 correspond to DDL compilation, DML language processing, database querying, database management and file management. The database description tables, authorization tables and concurrent access tables are a set of data to support the functionality of these modules. The execution of a DML instruction accesses the database corresponding to a DBMS procedure. The procedure retrieves data from the database into the application work area using the retrieval instructions, moves data from a work area into the database using the insert instructions, modifies the data in the database using the update instructions, or deletes data from the database using the delete instructions. By communicating with the authorization tables, the database manager checks the users' or programs' authorizations to data access. Authorized operations are forwarded to the file manager. The database manager is also responsible for concurrent access management. In



another word, database manager manages simultaneous access requests on the data by applications.

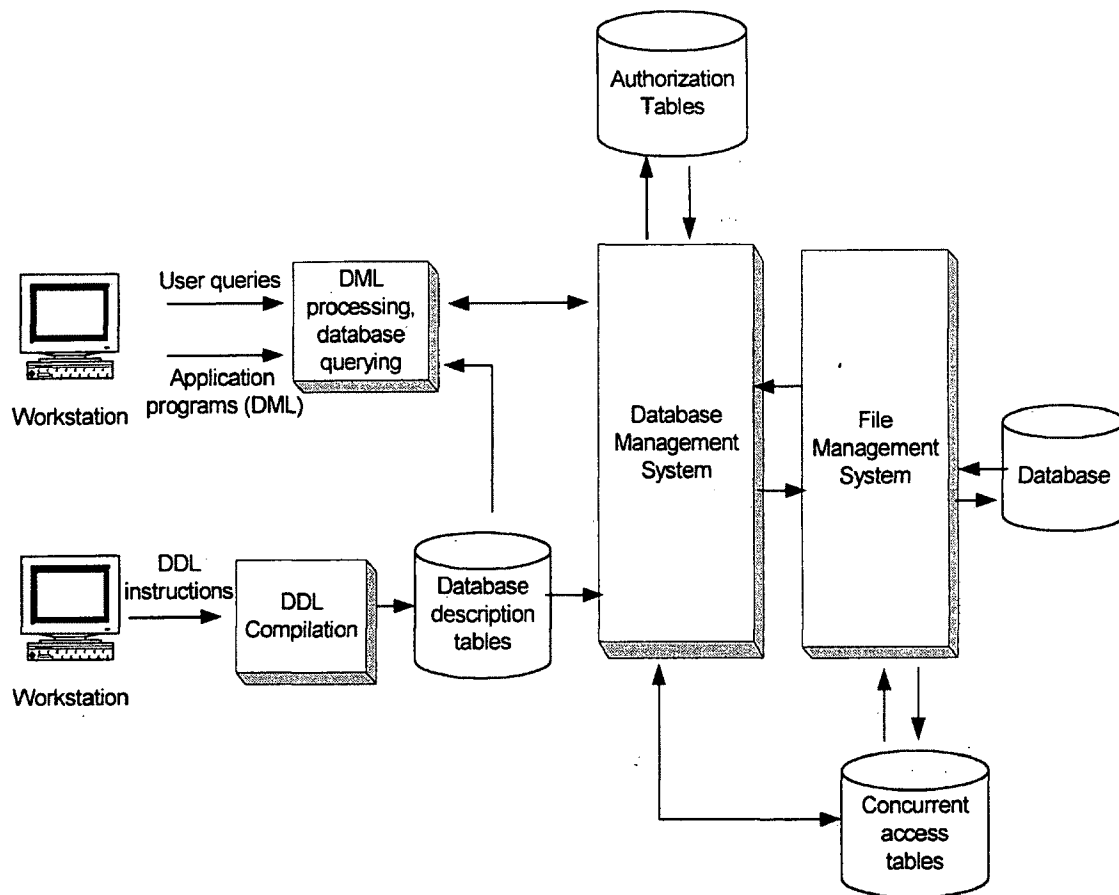


Figure 2.1 Database Management System Architecture

Many modern systems use a client-server architecture. Users work with a PC (client) and communicate with a large central computer (server). An assortment of network software is the third component, allowing communication between the client and the server. Most current database systems also utilize a client-server architecture. In the simplest client/server architecture, the entire DBMS is contained on the server. The client sends the requests to the server (the DBMS) for execution. The requests from the client to

the server are represented by SQL language in a relational system. After the processing, The database server then sends the answer, in the form of a table or relation, back to the client.

### **2.1.2 Data Description Levels**

The data description levels of a DBMS include the conceptual level, the internal level and the external level.

- The conceptual level is referred to as the conceptual or logical database. It is located between the other two levels. It consists of the abstract representation of the database which is independent from the physical implementation.
- The internal level is referred to as the physical database or internal database that is the implementation of the logical database. The physical database consists of physical files. It is concerned with data types, data lengths, record formats, storage structures, and access methods. It represents the database as actually stored and retrieved.
- The external level is referred to as the external database. It is concerned with the views created from the logical database by users. Each view consists of some entities and attributes of the logical database.

*The logical independence of data and the physical independence of data are supported by the above three levels allowed by a DBMS in the description of data [3].*

Logical data independence means that the application programs which operate on the logical schema are unaffected when the changes are made to the logical schema. In this case modifications of the logical schema correspond to redefining the relationship between the logical schema and the logical views of the applications. Physical data

independence means that the applications working on the data are unaffected when the changes are made to the physical data. The physical structures for the data storage can be modified without affecting the structure of the logical data schema.

A model of the database is the most important part of the design of a DBMS. It is also known as a data model and is used for modeling a logical database. There are many data models proposed and available in the literature, such as the entity-relationship model [10], the network model [21], the hierarchical model [21], etc. The most popular data storage model in DBMSs is the relational database. The reason for using the relational model is that they provide high independence of the physical structures of data. Additionally, unlike the hierarchical and network models, the relational model is flexible in that it allows a variety of operations and queries that are not bound to the underlying physical features.

## **2.2 Database Security**

In database environments, the different users of an organization work on a unique integrated set of data through the DBMS for the different applications. The same tables may be referred by different applications. On one hand, this solves such problems as duplication, data inconsistency, or dependence between the programs and the data structures; on the other hand, the issues of security threats are increasingly a matter of concern to such an organization.

### **2.2.1 Security Problems in Databases**

Violations to database security consist of improper accesses, modifications or deletions of data. The violations can be classified into three categories.

- *Improper release of information:* This is caused by intentional or accidental access of information by improper users. For example, unauthorized information is inferred through the authorized observation of data.
- *Improper modification of data:* This involves all violations to the data integrity through improper data handling or modifications. The tampered data may or may not be read. So, the improper modifications do not necessarily relate to unauthorized reading.
- *Denial of service:* These involve the actions that could prevent the users from accessing the database or using resources.

According to the way they can occur, the security violations can also be grouped into *non-fraudulent* (accidental) threats and *fraudulent* (intentional) threats [20].

Non-fraudulent threats are the damage caused by accidents. These threats involve:

- *Natural or accidental disasters:* These accidents can damage the system hardware and the stored data of the database system, for example, earthquakes, storm, water damage or fire. They always cause a data integrity violation or failure of service access.
- *Errors or bugs in hardware or software:* This may cause the unauthorized access, reading or modification of data, or the failure of access to the authorized users. This could cause some serious problem which waste a lot of time and resources. This could also cause weakness that hostile agents could take advantage of.

- *Human errors*: This causes unintentional violations such as the incorrect understanding or incorrect use of applications. This may lead to incorrect application of security policies.

Fraudulent or intentional threats are the damage caused by an explicit and determined fraudulent. Violations involve two classes of user:

- Hostile agents, improper users (outside and inside) breaking into system and executing damage to the software and/or system hardware, improperly accessing or modifying data. For example, all kinds of viruses, Trojan Horses and trapdoors are attacks of hostile agents.
- Authorized users who can abuse their rights, privileges and authority to cause a security breach.

### **2.2.2 Database Security Protection Requirements**

Protecting a database from possible threats means protecting the resources and the stored data from accidental or intentional unauthorized access and/or modification. The goal for every database is to make it available and useful to all the users who need that information. On the other hand, an organization needs to ensure that this same valuable data is not exposed to unauthorized individuals or corrupted by hostile parties. If the correct balance between security concerns is not met, not all users that could benefit from the information will have access to it. Although the security concerns for a database are the same as those for any other information system, integrity, access control, authorization, privacy, and confidentiality, database systems present some unique and challenging issues [14].

- Integrity: The integrity includes the integrity of the database, the operational integrity of data and the semantic integrity of data. Working with the database environment to achieve high quality is complicated by the processing needs, distance from sources, and different requirements of database users.

*The integrity of the database* concerns database protection from unauthorized access that could modify the contents of data, and database protection from viruses, sabotage, errors, or failures in the system. In case of failure, the state of the database may no longer be consistent. To preserve database consistency, we require that each transaction should terminate correctly and be able to modify the accessed data; or to terminate unsuccessfully and not be able to modify the accessed data. Backup and recovery procedures are widely used in DBMS. Basically, the recovery system reads a log file to determine the set of transactions need to be undone and the set of transactions which need to be redone. To undo (rollback) a transaction that has not been committed means to copy the old value of each operation back to the involved record. To redo a transaction that has been committed in the log means to copy the new value of each operation in the record.

*Operational integrity of data* is to ensure the logical consistency of data in a database during concurrent transactions. *Lock and unlock techniques are commonly used to solve the problem of ensuring that the concurrent access to the same data item by different transactions does not cause data inconsistency [3].* A transaction can lock a data item and make the data inaccessible to other transactions. The item is accessible again when the transaction has been completed and the locking is released.

*Semantic integrity of data* is to ensure the logical consistency of modified data by controlling data values in the allowed range. This includes the conditions for the whole database which are used to set up the correct state of a database and the conditions for transactions which need to be verified when executing a modification action to the database.

- **Authorization:** This includes the authentication of the user at log-on time and the use of authorization to protect database objects. Passwords can be used to distinguish various types of access (full, read-only, etc.), and need to be kept in an encrypted file. The authorization of objects includes the schema level (internal, conceptual, external) to be protected, the granularity of protection and the method of protection, such as protection through views, triggers, and stored procedures.
- **Inference:** Inference denotes the possibility of obtaining confidential information through access of non-confidential data. In statistical databases, users must be prevented from tracking back to information on individual entities through the statistical aggregated information.
- **Access control:** Access control for a database containing mixed data consists primarily of protecting the confidentiality of sensitive data. Identifying and implementing an access control policy for a database involves a number of unique challenges. Access control allows access only to authorized users who are granted a set of privileges on the sensitive data and prevented from propagating their privileges. Access requests have to be checked by the DBMS against the user's or application's authorization. Also, access controls need to apply to objects, such as records, attributes and values.

- **Confinement:** Confinement is needed to avoid undesired information transfer between system programs, in another word, transfer of the critical data into unauthorized program.

## 2.3 Secure Database

An overall view of the architecture of a DBMS including security features appears in Figure 2.2 [22].

After user *login* and *authentication*, each subsequent access query to the database which is made through an application program is mediated by the *Authorization System* procedures. The *Authorization System* procedures consult the *authorization rules* to check query compatibility with these rules. The access is allowed if the query and the rules match. If the query and the rules do not match, an error message will be sent to the user and the violation will be registered in a log file together with date, time and user references by the Authorization System. The *Security Administrator (authorizer)*, who defines access authorizations and/or security axioms through DDL or DML language, is responsible for defining the authorization rules derived from the security requirements of the organization. The *auditor's* responsibility is to detect possible suspect behaviors or to verify recurring types of violations and periodically examine the log file. The *authorizer* may be the same as the *Auditor*. The *Database Administrator (DBA)* is responsible for managing the conceptual schemas, logical schemas and internal schemas of the database.

The secure DBMS module manages all the queries. It includes *Authorization rules* and *Security axioms*. The *authorization rules* are used for discretionary controls and the *security axioms* are used for mandatory controls. One, or both, is used by the



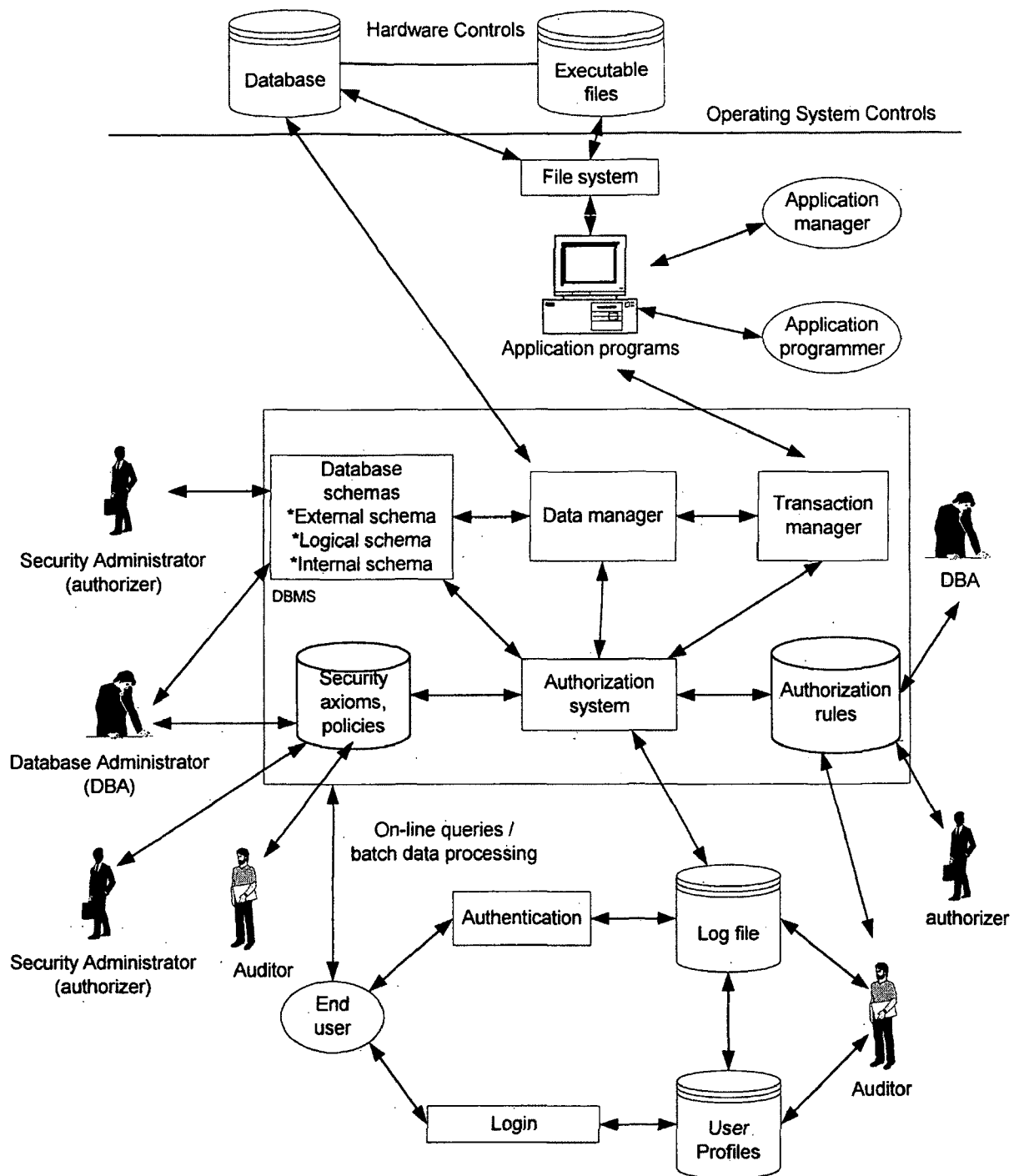


Figure 2.2 Architecture of a DBMS Including Security Features

Authorization System depending on the system protection requirements. The same module includes *Database schemas*, which are the protected-objects such as views, logical schemas and internal schemas.

Through the *Transaction manager*, authorized access queries are translated into program calls from the library of *Application programs*. Then, the authorized access queries are transformed into data access requests through the *Data manager*. The *Application manager* is responsible for the development and maintenance of the library programs such as PL/SQL library programs in the Oracle database system.

The control of file access can be achieved through the Operating System and hardware to ensure the data is exclusively transferred into the work area of a requesting user. Cryptography-techniques and backup copies are the usual means for protecting the physical stored data and the stored programs.

## 2.4 Database Security Controls

Access control, flow control, and inference control are three database security controls. Database protection can be obtained through the security measures for these three controls.

- *Access control:*

In the database systems, access controls are responsible for ensuring that all direct accesses to the system objects must follow the rules and procedures based on the protection policies. These rules are information stored in the system, stating the access mode to be followed by subjects upon access to the objects. The set of control procedures

checks the access requests (queries) against the stated rules and determines the validation of the queries. Queries may be allowed, denied, or modified.

The access control system is shown in Figure 2.3. In a *closed system*, only explicitly authorized accesses are allowed. In an *open system*, accesses are allowed if they are not explicitly forbidden. In a closed system, the minimum privilege policy is enforced, whereas, in an open system, maximized sharing is enforced. In a security system, the definition of 'who' can grant or revoke access rights is a relevant procedure. Grant and revocation are not always the prerogative of an authorizer or of a single security officer. Typically, in distributed systems, the authorization management is decentralized to different authorizers. Each is managed by a local Database Administrator (DBA).

Access control for multilevel system can be either mandatory or discretionary. Mandatory access controls restrict the access of subjects to objects based on the security labels. Mandatory controls prevent information flow towards objects of a lower classification. Through the definition of subject and object security classes, a mandatory control determines the access to data. The mandatory access control is shown in Figure 2.4. Discretionary access controls allow access rights to be propagated from one object to another. The discretionary access control requires that the authorization rule should be defined to specify the privileges owned on the system objects. Access checked by the discretionary control is granted only to subjects for which an authorization rule exists and is verified. The discretionary access control is shown in Figure 2.5.

- *Flow control:*

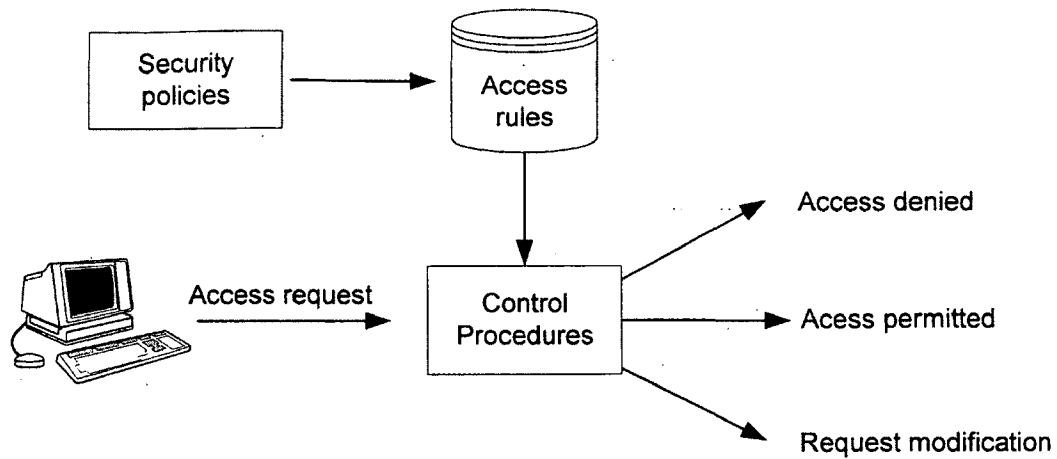


Figure 2.3 Access Control System

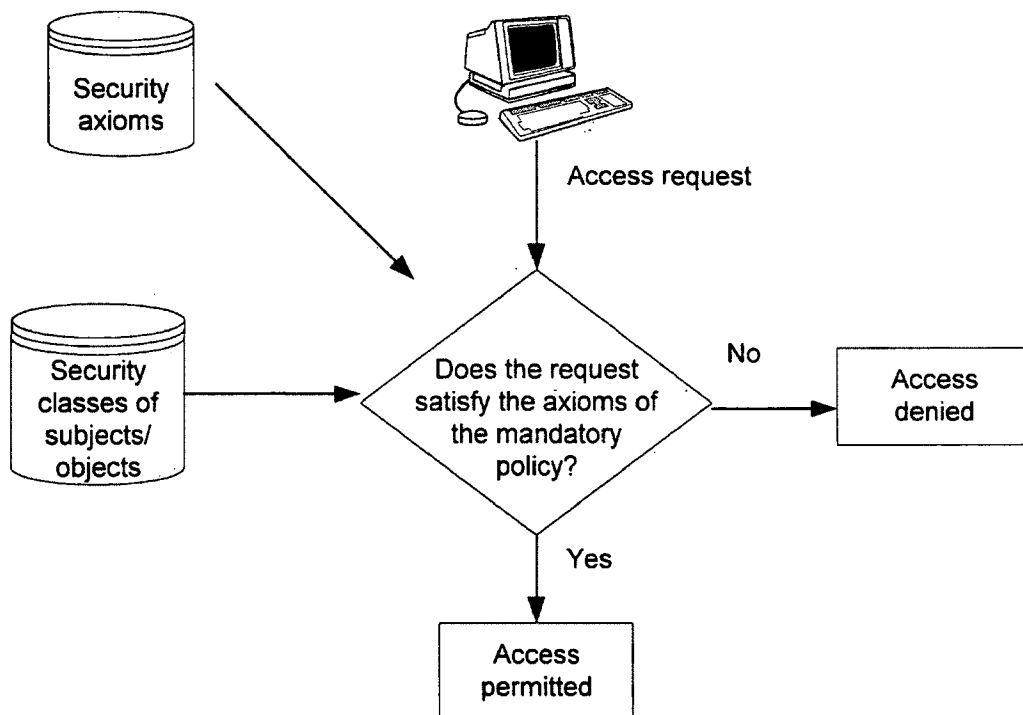


Figure 2.4 Mandatory Access Control

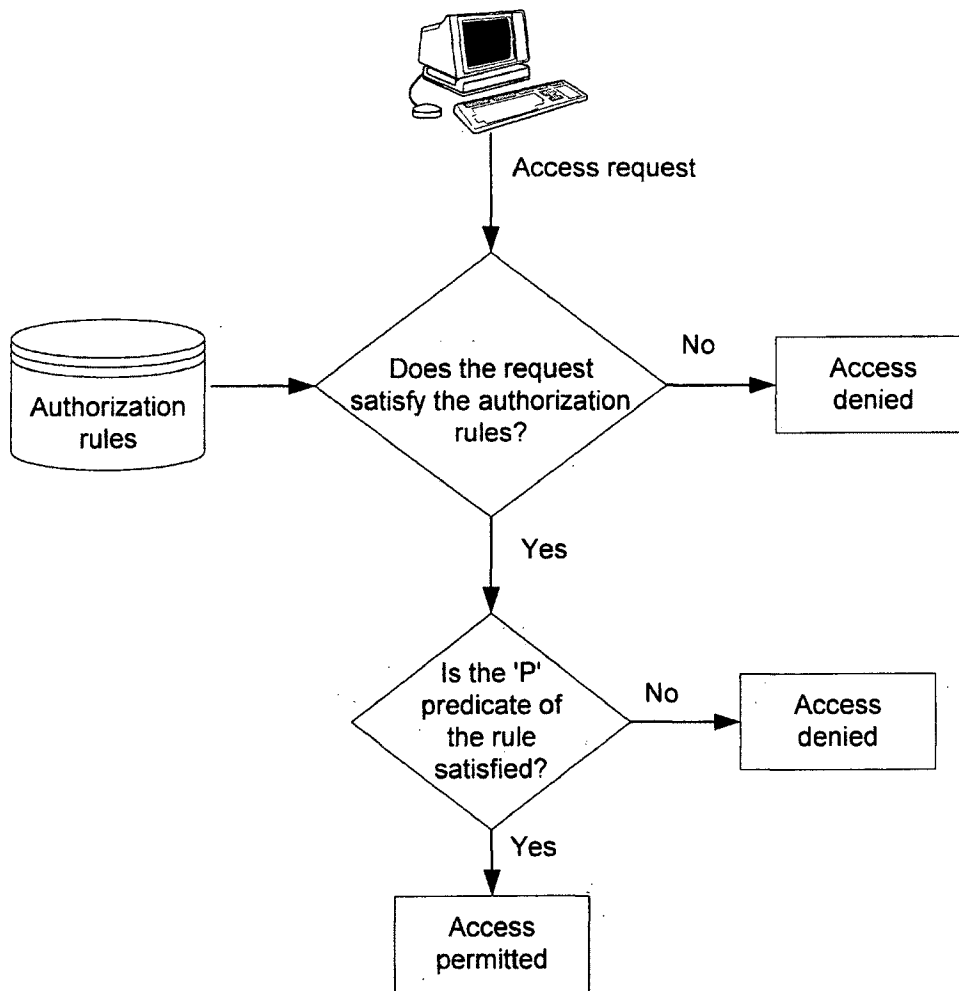


Figure 2.5 Discretionary Access Control

When a statement reads values from X and writes values into Y, a flow between object X and object Y occurs. A typical example of information flow is copying data from object X to object Y. The user could get data indirectly in object Y what he cannot get directly from object X. In another case, when part of the data in X is moved to Y, as a result, the information about the data in X is provided even if the data is not exactly data contained in X. This can lead to the secrecy violation. Moreover, a flow violation

occurs via a transfer request for data. The transfer occurs between two objects on the admitted flows, but the transfer is unauthorized. Flow control should efficiently deny the execution of this kind of requests.

The operations that are defined by the flow control policies allow transfer of objects to low levels, while keeping the sensitivity of data in these objects to be unaltered. Higher-class objects are more protected during 'read' access than lower-class objects. Flow controls prevent violations like information being transferred to the lower levels since the low levels are more accessible levels. Transfer to higher levels is allowed since the higher levels are more protected levels.

- *Inference control:*

Inference controls are to protect data from indirect detection. An inference channel is a channel where a set of data items in  $X$  to be read by a user can be used to obtain the set of data items in  $Y$  by applying a function  $f$  to  $X$ :  $Y = f(X)$ .

One main inference channel in a database is indirect access. Suppose an unauthorized user is not authorized to access data in  $X$ . But he can succeed to know the set of data items in  $X$  through a query on the set of data on  $Y$ , which he has rights to access, and undergoing conditions on  $Y$ . For example,

```
SELECT X FROM R WHERE Y = <value>
```

```
INSERT INTO R(X) WHERE Y = <value>
```

The data items in  $X$ , which is not visible to the user, can be read or modified through the data items in  $Y$ , which is visible to the user. Indirect access can be viewed as a flow violation, and the flow controls can sometimes be included in inference control.

Another main inference channel is correlated data. For example,  $t$  and  $Y$  are visible and  $X$  is invisible. The data  $X$  can be inferred from the existing arithmetical relation:  $X = t \times Y$ . Here, the invisible data  $X$  is semantically connected to visible data  $Y$ . Inference control can be considered as decreasing the uncertainty degree about a data value and increasing the authorized requests.

The third main inference channel is missing data. Users can come to know the set of data items  $X$  which is not visible to him through the missing data. For example,

```
SELECT X, Y FROM r WHERE Y = NULL;
```

Through the authorized information and the null values which mask the sensitive inaccessible data, the user's query about the relation can display the data set  $X$  which he has no authority to access. These null values make it possible to infer the existence of the masked item value.

## Chapter 3

### Designing Database Security

#### 3.1 Secure DBMS Design

*A database is a collection of data that are organized and managed by a DBMS (Database Management System) [21]. Database security is achieved through a set of mechanisms at both the DBMS level and operating system (OS) level. It has been shown that the OS level's management functions and shared resources management are very useful to the security in DBMS environment. But the DBMS level has its own security concerns which are different from the OS level [24].*

- *Data life cycle.* Data in a database system usually has a long life cycle since the application may last for many years. The DBMS must protect the data throughout the whole life cycle of the data.
- *Logical and physical objects.* A database has logical objects and physical objects. The logical objects in a database include views, relations, keys, etc. The physical objects in a database include data files, data blocks, memory, processes etc. The DBMS logical objects are independent of OS physical objects. So, the security mechanisms in DBMS must protect both the database logical objects and the database physical objects.
- *Dynamic and static objects.* Logical objects in databases can be dynamically created and may not have corresponding physical objects. For example, the query results are different for different queries. Views are dynamically created from the base tables.



The virtual relations are derived from the base relations, which are actually stored in the databases. Physical objects, which are managed by the OS, are static. The security mechanisms in DBMS must specifically protect the database dynamic and static objects.

- *Metadata.* In a database, there is a “metadata” object that provides information about the structure of the data in the database. For example, in relational database, the metadata object describes the relationships between entities, the domains of attributes, the storage of the tables, the constraints of the table, etc. In fact, the metadata has the sensitive and important information such as relationships and data types of the database contents and can be used as a method for controlling access to the underlying data. In a database, the metadata is stored in data dictionaries, which are tables that are different from the application tables. The metadata object should be protected in the same way as the usual data is.
- *Semantic correlation among data.* In a database, data has semantics and data is related with other data through the semantic relationships. The security mechanism in a DBMS should avoid the security violations which are related to the semantic correlation among objects. For example, the inference threats are related to semantic relationships. In a statistical database, users must be prevented from inferring individual data items from aggregated statistical data obtained through statistical queries.
- *Multiple data types.* Databases are characterized by a variety of data types. So, in a DBMS, multiple access modes are required. For example, these can include administrative mode, statistical mode, etc. In relational databases, access modes are

expressed in terms of the basic SQL statements such as SELECT, DELETE, INSERT and UPDATE. There are only physical access modes for read, write and execute operations at the OS level.

- *Multilevel transaction.* In a database, transactions may involve data at different security levels. The DBMS security mechanism must avoid security violations in these multilevel transactions. For example, the data may flow from the high security level to the low security level. This action might cause a security violation.
- *Object granularity.* The granularity of objects in a DBMS is finer than the granularity of objects in an OS. For example, the granularity of objects in a DBMS include the database, collection of relations, one relation, some rows of one relation, some columns of one relation, etc. A finer granularity for data sharing is also required at DBMS level. The DBMS must ensure the access controls at various degree of granularity.
- *Multilevel protection.* The DBMS should enforce multilevel protection, in that different security labels can be assigned to data items in the same object. For example, in relational databases, a relation has a security label and contains attributes which have their own security labels. The mechanisms need to define the security labels and to assign the security labels to the objects and subjects.

### **3.2 Security Models in DBMS**

Security models in DBMS can be broadly classified in two categories: the discretionary model and the mandatory model.

Discretionary security models allow users to grant other users authorizations to access the objects. The creator of an object is allowed to grant and revoke other users' access and privileges to the object created. Discretionary security models govern the access of users to the information on the basis of the users' identity and on the basis of rules that specify the types of access that the user is allowed for the objects.

Mandatory security models govern the access to the information by the individuals on the basis of the classifications of subjects and objects in the database system. Objects are the passive entities that store information of the database, such as data files, tables, records, fields, columns, etc. Subjects are the active entities that access the objects. If the relationship is satisfied between the classifications of the subject and object depending on the access mode, the access of a subject to an object is granted.

### **3.2.1 Elements of DBMS Security Models**

The elements of a DBMS security model are shown in Figure 3.1 [22].

- *Subjects.* Subjects can be considered to be active processes operating on behalf of a user. These are the active entities of the system that request access to the objects. Subjects represent possible threats to the data security. The access requests must be verified in order to ensure they match the security requirements of the database.
- *Objects.* Objects contain information to be protected from unauthorized accesses or unauthorized modifications. Objects include the physical tables, rows, columns, data dictionaries, etc.

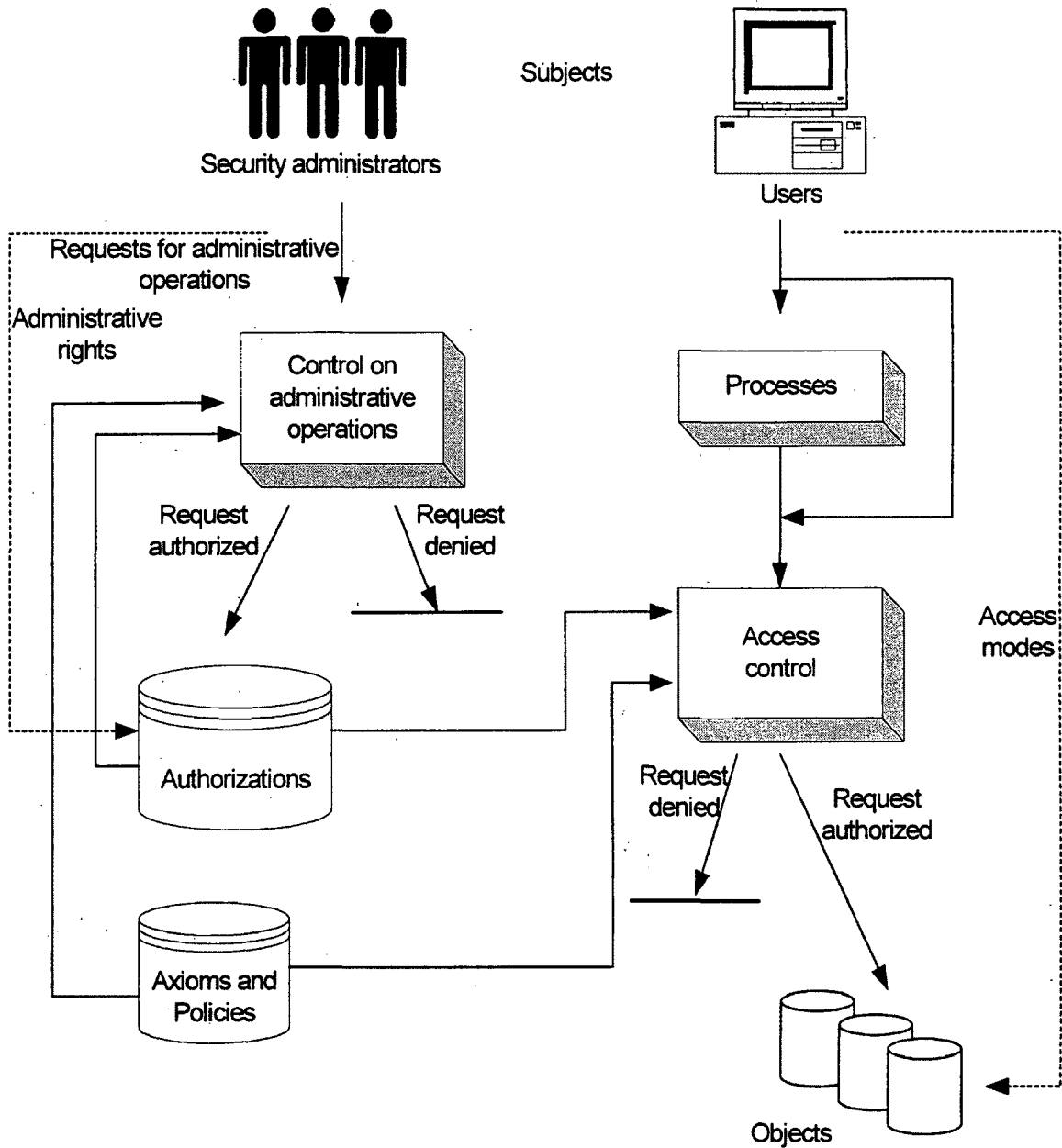


Figure 3.1 Elements of DBMS Security Models

- *Access modes.* The execution of an access mode causes a data information flow from the object to the subject and vice versa. Access modes are the types of access through which the subjects access the objects.

- *Policies.* These are the rules based on which accesses are managed. They are the security requirements of the database.
- *Authorization.* The authorization state includes the set of rights and privileges that the users have for accessing and working on the objects of the database system.
- *Administrative rights.* The privileges such as *grant*, *revoke* and *own* allow the modification of the authorizations. For example, in discretionary systems, subjects' administrative rights enable them to grant the rights/privileges to other subjects and revoke the rights/privileges from other subjects.
- *Axioms.* These are properties that must be satisfied in the system. Axioms state conditions which must be satisfied by the authorization state. A modification to the authorization state such as subjects, objects or authorization through the administrative privileges is only allowed if the resulting authorization state still satisfies the axioms.

### 3.2.2 The Wood Model

*This model, proposed by Wood et al. [22], is oriented to authorization management and access control in multilevel-schema databases.* This is a discretionary model. The model uses a three-level architecture for databases. The three levels are the external level, the conceptual level and the internal level. According to this architecture, a database is characterized by an internal schema, a conceptual schema, and a number of external schemas. Each of the schemas may be based on a different data model and specify a number of object types.

- *Architecture of model*

Based on a three-level view of the database, the architecture of the Wood model has the following three levels:

- (1) The external level: This level is the closest to the users and represents the users' view on the database. Users' requests to access the objects of external schema are translated into the operations on objects of the conceptual level and internal level.
- (2) The conceptual level: This level represents the structure of the data stored in the database. Multiple external objects may be supported by one conceptual object.
- (3) The internal level: This is the nearest level to the physical storage. This is the lowest level of the representation of the data stored in the physical database.

Figure 3.2 shows a three-level database architecture of the Wood model. According to this architecture, there are several external schemas, a conceptual schema and an internal schema in a database. The Wood model considers a relational database for the external level and an entity-relational model for the conceptual level. It pays more attention to the consistency among the authorization rules at the different levels and on the validation of access requests based on these rules.

- ***Subjects:***

The subjects in the model are the users who access the database system. There are two distinguished types of users in this model. One is the *authorizers* who are responsible for administering the authorizations such as granting the access rights on the database objects to the users and revoking the access rights on the database objects from the users. Another is the *users* who access the system based on their access rights which are granted by the authorizers.

- ***Objects:***

The objects in this model include the conceptual-level objects and the external-level objects. Each object  $o$  in the system has a type category. The possible type categories at the conceptual level include attribute, entity and relationship between the different entity

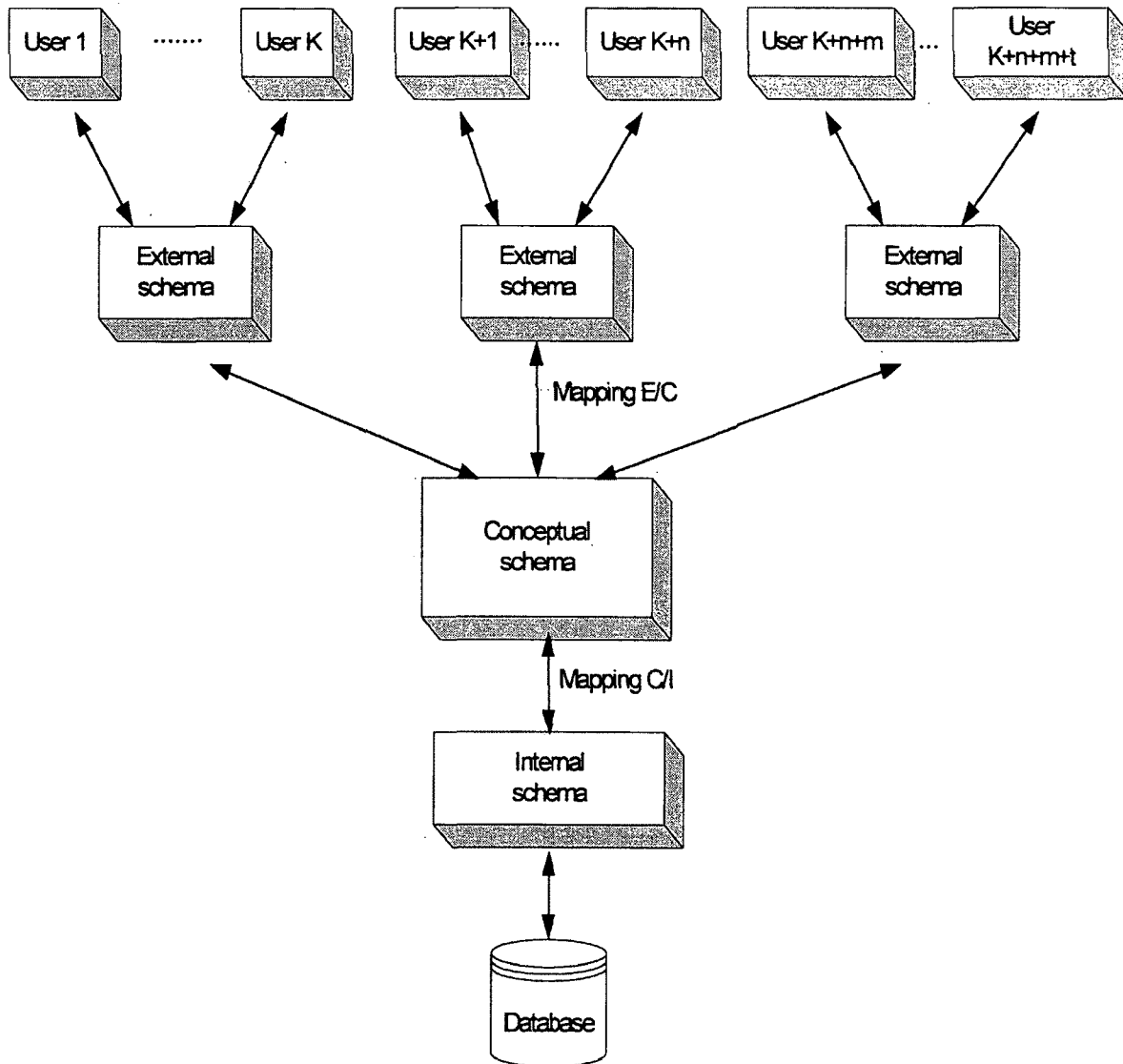


Figure 3.2 Three-level Database Architecture of The Wood Model

sets. The relationship type is a bi-directional relationship type between two entities. For each direction, the relationship has its own meaningful name. The type categories at the external level are table/view and field/column. The tables can be described in terms of conceptual-level objects.

- **Access modes:**

There are two access modes in this model, one is the conceptual-level access modes and another is the external-level access modes. A set  $A$  of access modes is defined for each type category.

(1) Conceptual-level access modes:

$$A_{entity} = \{\text{insert, delete}\};$$

$$A_{attribute} = \{\text{update, read, use}\};$$

$$A_{relation} = \{\text{insert, delete, use}\}.$$

For all conceptual-level type category  $C$ , the access modes  $A_c$  are:

$$C = \{\text{entity, attribute, relation}\}$$

$$A_c = A_{entity} \cup A_{attribute} \cup A_{relation} = \{\text{insert, delete, update, read, use}\}$$

(2) External-level access modes:

$$A_{table / view} = \{\text{insert, delete}\};$$

$$A_{field / column} = \{\text{update, read, use}\};$$

For all external-level objects  $E$ , the access modes  $A_e$  are:

$$E = \{\text{table, column}\}$$

$$A_e = A_{table / view} \cup A_{field / column} = \{\text{insert, delete, update, read, use}\}$$

- **Access rules:**



The authorizations are described by access rules which are 4-tuples of the form  $\langle s, o, t, p \rangle$ . The access rule  $\langle s, o, t, p \rangle$  means that the subject  $s$  can execute the access mode  $t$  on the object  $o$  under conditions expressed by predicate  $p$ . Since the conceptual level provides a global view of the database structure, the basic access rules should be defined at this level. These access rules can be transformed into the access rules on the external-level objects. The authorizer of the external object has the authorization to specify the access types for one specific object by accepting the conceptual-level rules and by defining some new or more restrictive rules. For example, a table definition includes the access constraints which define the access rights for each external-level table and field. These constraints are stored in an *Access Constraint table (AC)*. An example of an AC table is illustrated in Table 3.1. The AC table has two columns, one is the external objects O and another is access types T.

O	T
EMPLOYEE	delete
EMPLOYEE	insert
EMPLOYEE.name	read
EMPLOYEE.ssn	read
EMPLOYEE.manager	read
EMPLOYEE.salary	read
EMPLOYEE.manager	update
EMPLOYEE.salary	update

Table 3.1 An example of an *Access Constraint (AC)* Table

The access rules for the conceptual level are stored in the *Conceptual Rules (CR)* table.  $\langle s_c, o_c, t_c, p_c \rangle$  is defined for the entity sets, attributes and relationships. An example of CR table is illustrated in Table 3.2. The access rules for the external level are

stored in the *External Rules (ER)* table.  $\langle s_e, o_e, t_e, p_e \rangle$  is defined for tables and fields. The access type specified in the external rule for an external object must be one of the allowable accesses to that object specified in table AC. The access rules in external level must be consistent with underlying conceptual-level rules.

$S_c$	$O_c$	$T_c$	$P_c$
Jones	EMPLOYEE.name	read	-
Jones	EMPLOYEE.ssn	read	-
Jones	EMPLOYEE.manager	read	-
Jones	EMPLOYEE.salary	read	-
Jones	EMPLOYEE.salary	update	where EMPLOYEE.manager= Jones
Smith	EMPLOYEE.name	read	-
Smith	EMPLOYEE.ssn	read	-

Table 3.2 An example of a *Conceptual Rules (CR)* Table

- **Access control:**

An access request submitted by a user is checked against the external-level rules. If the request matches a rule in the external-level rules, the access control process continues. Otherwise, the process terminates. To restrict the user access only to the occurrences of the objects which satisfied both the predicate given by the user and the predicate stated in the access rules, the external-level rule predicate is appended via the AND operator to the query. Then, the modified request on the external object is transformed into the operation on the underlying conceptual objects. The operation and objects are modified by the restricting operations with the predicates in the conceptual level. Figure 3.3 illustrates the access control steps in the Wood model.

For example:

Manager Jones wants to update Bob's salary to \$40,000. The request can be expressed as:

$\langle S, O, T, P \rangle = \langle \text{Jones}, \text{Employee.salary}, \text{Update}, \text{where Employee.name} = \text{'Bob'} \rangle$

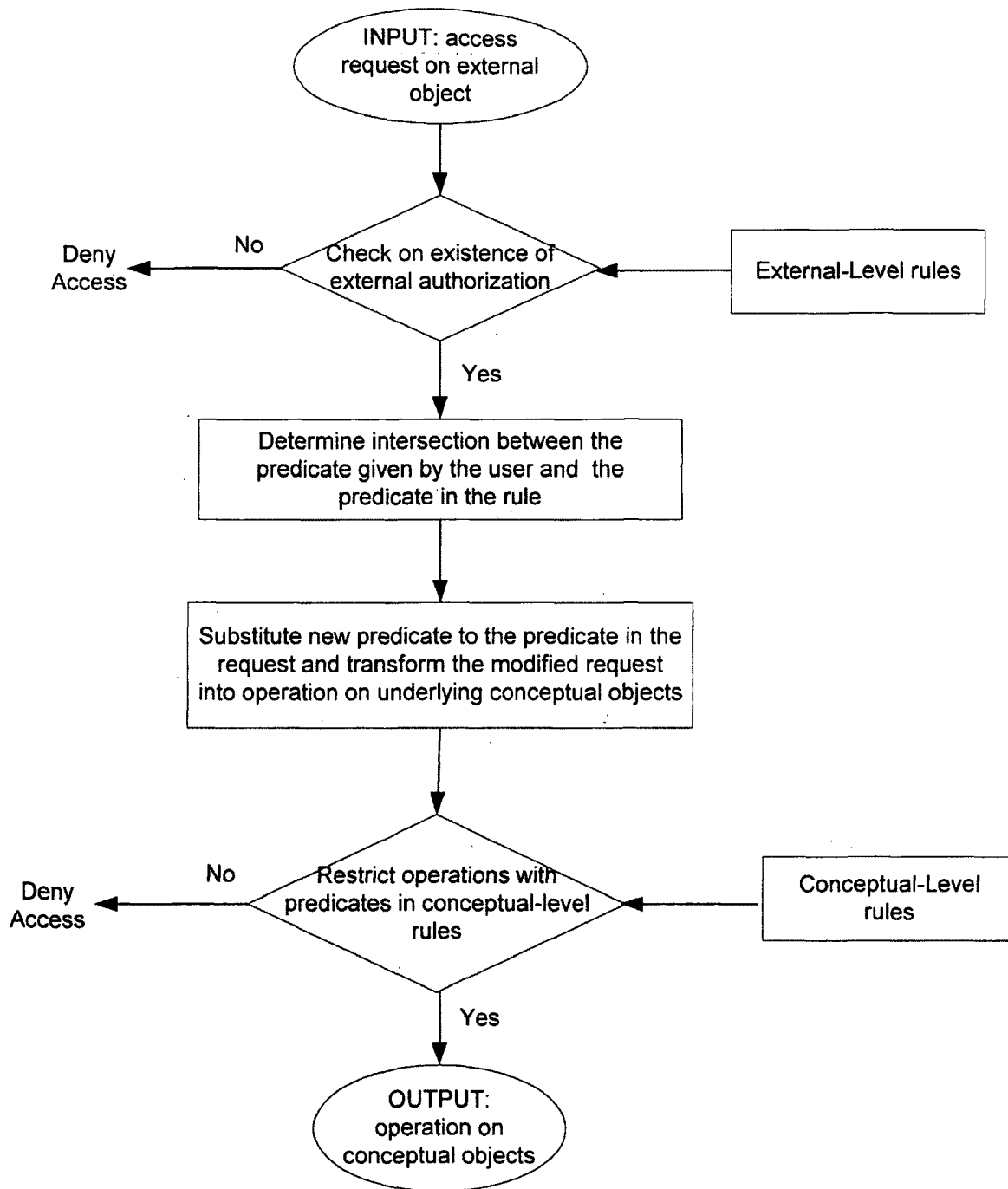


Figure 3.3 Access Control Steps in The Wood Model

$S_e$	$O_e$	$T_e$	$P_e$
Jones	EMPLOYEE.name	read	-
Jones	EMPLOYEE.name	update	-
Jones	EMPLOYEE.manager	read	-
Jones	EMPLOYEE.salary	read	-
Jones	EMPLOYEE.salary	update	Where EMPLOYEE.name ◇ EMPLOYEE.manager AND EMPLOYEE.name ◇ Jones
Jones	EMPLOYEE.ssn	read	-
Jones	EMPLOYEE.ssn	update	-
Smith	EMPLOYEE.name	read	-
Smith	EMPLOYEE.ssn	read	-

Table 3.3 An example of an *External Rules (ER)* Table

First, the system checks the  $\langle S, O, T, P \rangle$  in the ER table (Table 3.3). If  $T = T_e$  exists in the ER table where  $S_e$  is Jones and  $O_e$  is EMPLOYEE.salary, the access control processing will continue. Otherwise, the access will be denied. The predicate for  $S_e$  (Jones),  $O_e$  (EMPLOYEE.salary) and  $T_e$  (Update) is ‘where EMPLOYEE.name ◇ EMPLOYEE.manager AND EMPLOYEE.name ◇ Jones’ in this ER table because Jones should not be able to update the salary for herself and for her managers. Then, the system checks the modified  $\langle S, O, T, P \rangle$  in the CR table (Table 3.2). Here, the modified predicate is ‘where EMPLOYEE.name ◇ EMPLOYEE.manager AND EMPLOYEE.name ◇ Jones AND EMPLOYEE.name = Bob’. In the CR table, there is a predicate, “where EMPLOYEE.manager = Jones”, where  $S_c$  is Jones,  $O_c$  is EMPLOYEE.salary and  $T_c$  is Update. So, the new predicate is ‘where EMPLOYEE.name ◇ EMPLOYEE.manager AND EMPLOYEE.name ◇ Jones AND EMPLOYEE.name = Bob AND EMPLOYEE.manager = Jones’. Since Jones submits the

request to update employee Bob's salary, the modified predicate is satisfied. After this access control is processed, Jones can perform the update action on the database objects – Bob's salary.

### 3.2.3 The Dion Model

*The Dion (1981) model [23] proposes a mandatory model that protects the secrecy of data as well as the integrity of data. This model does not allow the information transfer from the objects to the subjects through the operations such as write. The information flow is only allowed between the objects, not allowed between the objects and subjects. In this model, all accesses satisfying the mandatory policy are considered authorized. This model is based on a classification of subjects and objects which include both security and integrity levels. Security and integrity are expressed as a set of rules.*

*A classification (C) and a set of categories (S) define each security level. The classification includes Top Secret (TS), Secret (S), Confidential (C) and unclassified (U). They are ordered:  $TS > S > C > U$ . The elements of a set of categories depend on the database system and the application. A security level  $L_1 = (C_1, S_1)$  is higher or equal to security level  $L_2 = (C_2, S_2)$ , i.e.  $SL(L_1) \geq SL(L_2)$  if and only if the following relationships are valid:  $C_1 \geq C_2$  and  $S_1 \supseteq S_2$ .*

*Each integrity level also consists of two elements: a classification (C) and a set of categories (S). The classification consists of Crucial (C), Very Important (VI) and Important (I). They are ordered:  $C > VI > I$ . The elements of a set of categories depend on the database system and the application. A integrity level  $L_1 = (C_1, S_1)$  is higher or*

equal to integrity level  $L_2 = (C_2, S_2)$ , i.e.  $IL(L_1) \geq IL(L_2)$  if and only if the following relationships are valid:  $C_1 \geq C_2$  and  $S_1 \supseteq S_2$ .

- **Subjects:**

The subjects in the Dion model are considered as the programs which execute in the system and the acting on behalf of users. Each system subject is assigned three security levels and three integrity levels. The security level assigned to a subject reflects the subject's trustworthiness not to disclose the sensitive information to individuals who do not have the appropriate security level. A subject can access the system at any level dominated by its security level. The following three security levels are assigned to each subject and the following security relationship must be satisfied for each subject:

$$RSL(s) \geq ASL(s) \geq WSL(s)$$

- (1) **Absolute Security Level (ASL):** This security level is assigned to the subject when it is created or executed. This level stays throughout the whole life cycle of the subject. More specifically, this level is the security level of the user whom the subject is acting for.
- (2) **Read Security Level (RSL):** This is the highest level. The subject is allowed to read if the Read Security Level is assigned to the subject.
- (3) **Write Security Level (WSL):** This level is the lowest level. The subject is allowed to write if the Write Security Level is assigned to the subject.

Each subject in the system is assigned an integrity level. The integrity level assigned to a subject reflects the subject's trustworthiness for inserting, modifying or deleting information. The following three integrity levels are assigned to each subject and the following integrity relationship must be satisfied for each subject:

$$WIL(s) \geq AIL(s) \geq RIL(s)$$

- (1) Absolute Integrity Level (AIL): This integrity level stays throughout the whole life cycle of the subject. This level is the integrity level of the user for whom the subject is acting. This integrity level is assigned to the subject when it is created or executed.
- (2) Read Integrity Level (RIL): The subject is allowed to read if the Read Integrity Level is assigned to the subject. This level is the lowest integrity level.
- (3) Write Integrity Level (WIL): The subject is allowed to write if the Write Integrity Level is assigned to the subject. This level is the highest integrity level.

A subject can be trusted from the viewpoint of security and integrity, depending on the number of strictly satisfied inequalities. For a subject, if at least one of above inequalities is satisfied, the subject is said to be trusted. The subject is said to be untrusted if the above four relationships are satisfied by equality for the subject.

- **Objects:**

In this model, the data storage entities are considered to be the objects. Each object is assigned three security levels and three integrity levels. The security level of an object reflects the sensitivity of the information stored therein and the potential damage which could result from unauthorized disclosure of information. The following three security levels are assigned to each object and the following security relationship must be satisfied for each object:

$$MSL(o) \geq ASL(o) \geq CSL(o)$$

- (1) Absolute Security Level (ASL): This level stays throughout the whole life cycle of the object. More specifically, this level is the security level of the data which is contained in the object.

(2) Migration Security Level (MSL): This is the highest security level. The data in the object is allowed to flow to this level.

(3) Corruption Security Level (CSL): This level is the lowest level. The data is allowed to flow into the object from this level.

Each object in the system is assigned an integrity level. The integrity level assigned to an object reflects the degree of trust that can be placed in the information stored in the object. It also reflects the potential damage that could result from unauthorized modification of the information. The following three integrity levels are assigned to each object and the following integrity relationship must be satisfied for each object:

$$CIL(o) \geq AIL(o) \geq MIL(o)$$

(1) Absolute Integrity Level (AIL): This integrity level stays throughout the whole life cycle of the object. More specifically, this level is the integrity level of the data which is contained in the object.

(2) Migration Integrity Level (MIL): The data in the object is allowed to flow to this level. This level is the lowest integrity level of an object.

(3) Corruption Integrity Level (CIL): The data is allowed to flow into the object from this level. This level is the highest integrity level of an object.

- **Axioms:**

The request by subject  $s$  to establish a connection between object  $O_1$  and object  $O_2$  is granted only if the levels of the subject and objects satisfy the restrictions stated in the axioms of the model. The axioms in the Dion model include:

(1) Migration property: The information in object  $O_1$  is written into object  $O_2$ . The migration levels of  $O_2$  must be at least as restrictive as the migration level of  $O_1$ .



$$MSL(O_1) \geq MSL(O_2), MIL(O_2) \geq MIL(O_1).$$

If the migration property were not satisfied, data in object  $O_1$  would be allowed to migrate improperly to the security levels, which are not allowed, through the mediation of object  $O_2$ .

- (2) Corruption property: The information in object  $O_1$  is written into object  $O_2$ . The corruption levels of  $O_1$  must be at least as restrictive as the corruption level of  $O_2$ .

$$CSL(O_1) \geq CSL(O_2), CIL(O_2) \geq CIL(O_1).$$

If this corruption property were not satisfied, data, which should not be allowed to flow into object  $O_2$ , would be able to flow into object  $O_2$  through the mediation of object  $O_1$ .

- (3) Security property: For the object  $O_1$  from which the information is read, the subject requesting a connection must have read access to the object  $O_1$ , i.e. the Read Security Level (RSL) must have been assigned to the subject. For the object  $O_2$  to which the information is written, the subject requesting a connection must have write access to the object  $O_2$ , i.e. the Write Security Level (WSL) must have been assigned to the subject. The 'Read Security Level' of the subject making a connection dominates the 'Absolute Security Level' of the object from which the information is read. The 'Absolute Security Level' of the object to which the information is written dominate the 'Write Security Level' of the subject making a connection.

$$RSL(s) \geq ASL(O_1), ASL(O_2) \geq WSL(s).$$

This Security property ensures that a subject can only read the objects whose security level is dominated by the level of the subject. It also ensures that a subject can only write objects whose security level dominates the security level of the subject.

- (4) Integrity property: For the object  $O_1$  from which the information is read, the subject establishing a connection must have read access to the object  $O_1$ . For the object  $O_2$  to which the information is written, the subject establishing a connection must have write access to the object  $O_2$ . The 'Absolute Integrity Level' of the object from which the information flow dominates the 'Read Integrity Level' of the subject requesting a connection. The 'Write Integrity Level' of the subject requesting a connection dominates the 'Absolute Integrity Level' of the object to which the information is transferred.

$$AIL(O_1) \geq RIL(s), WIL(s) \geq AIL(O_2).$$

This Integrity property ensures that a subject can only read the objects whose integrity level dominates the level of the subject. It also ensures that a subject can only write objects whose integrity level is dominated by the integrity level of the subject.

- (5) Write/corruption property: The information is written to object  $O_2$ . The 'Absolute Security Level' of the subject establishing a connection dominates the 'Corruption Security Level' of object  $O_2$ . The 'Corruption Integrity Level' of object  $O_2$  dominates the 'Absolute Integrity Level' of the subject establishing a connection.

$$ASL(s) \geq CSL(O_2), CIL(O_2) \geq AIL(s).$$

This property ensures the satisfaction of the corruption level of the object  $O_2$  to which the data is written with respect to the subject making the connection. It derives directly from the above definitions.

- (6) Read/migration property: The information is read from object  $O_1$ . The 'Migration Security Level' of object  $O_1$  dominates the 'Absolute Security Level' of the subject making a connection. The 'Absolute Integrity Level' of the subject making a connection dominates the 'Migration Integrity Level' of object  $O_1$ .  
 $MSL(O_1) \geq ASL(s)$ ,  $AIL(s) \geq MIL(O_1)$ .

This property ensures the satisfaction of the migration level of the object  $O_1$  from which the data is read with respect to the subject making the connection. It derives directly from the above definitions.

For example:

Subject  $Sbj$  wants to insert the data, which is in table  $Obj_1$ , into table  $Obj_2$ .

The Read Security Level for  $Sbj$ : classification – Secret (S),

The Absolute Security Level for  $Sbj$ : classification - Secret (S),

The Write Security Level for  $Sbj$ : classification – Confidential (C),

The Write Integrity Level for  $Sbj$ : classification – Crucial (C),

The Absolute Integrity Level for  $Sbj$ : classification – Very Important (VI),

The Read Integrity Level for  $Sbj$ : classification – Important (I),

A set of categories for  $Sbj$ :  $C_1, C_2, C_3, C_8$ .

The Migration Security Level for  $Obj_1$ : classification – Top Secret (TS),

The Absolute Security Level for  $Obj_1$ : classification - Confidential (C),

The Corruption Security Level for  $Obj_1$  : classification – Confidential (C),

The Corruption Integrity Level for  $Obj_1$  : classification – Very Important (VI),

The Absolute Integrity Level for  $Obj_1$  : classification – Important (I),

The Migration Integrity Level for  $Obj_1$  : classification – Important (I),

A set of categories for  $Obj_1$  :  $C_1, C_2, C_3, C_8$ .

The Migration Security Level for  $Obj_2$  : classification – Secret (S),

The Absolute Security Level for  $Obj_2$  : classification - Secret (S),

The Corruption Security Level for  $Obj_2$  : classification – Unclassified (U),

The Corruption Integrity Level for  $Obj_2$  : classification – Crucial (C),

The Absolute Integrity Level for  $Obj_2$  : classification – Very Important (VI),

The Migration Integrity Level for  $Obj_2$  : classification – Very Important (VI),

A set of categories for  $Obj_2$  :  $C_1, C_2, C_3, C_8$ .

From the above, we can see the following relationships are satisfied:

$$RSL(Sbj) \geq ASL(Sbj) \geq WSL(Sbj), WIL(Sbj) \geq AIL(Sbj) \geq RIL(Sbj).$$

$$MSL(Obj_1) \geq ASL(Obj_1) \geq CSL(Obj_1), CIL(Obj_1) \geq AIL(Obj_1) \geq MIL(Obj_1).$$

$$MSL(Obj_2) \geq ASL(Obj_2) \geq CSL(Obj_2), CIL(Obj_2) \geq AIL(Obj_2) \geq MIL(Obj_2).$$

From the above, we can see the following relationships satisfy the Migration property

$$\text{axioms: } MSL(Obj_1) \geq MSL(Obj_2), MIL(Obj_2) \geq MIL(Obj_1).$$

From the above, we can see the following relationships satisfy the Corruption property

$$\text{axioms: } CSL(Obj_1) \geq CSL(Obj_2), CIL(Obj_2) \geq CIL(Obj_1).$$

From the above, we can see the following relationships satisfy the Security property axioms:  $RSL(Sbj) \geq ASL(Obj_1)$ ,  $ASL(Obj_2) \geq WSL(Sbj)$ .

From the above, we can see the following relationships satisfy the Integrity property axioms:  $AIL(Obj_1) \geq RIL(Sbj)$ ,  $WIL(Sbj) \geq AIL(Obj_2)$ .

From the above, we can see the following relationships satisfy the Write/corruption property axioms:  $ASL(Sbj) \geq CSL(Obj_2)$ ,  $CIL(Obj_2) \geq AIL(Sbj)$ .

From the above, we can see the following relationships satisfy the Read/migration property axioms:  $MSL(Obj_1) \geq ASL(Sbj)$ ,  $AIL(Sbj) \geq MIL(Obj_1)$ .

Therefore, subject  $Sbj$  can insert the data, which is in table  $Obj_1$ , into table  $Obj_2$ .

### 3.2.4 The RBAC96 Model

*In role-based access control (RBAC), permissions are associated with roles, and users are made members of appropriate roles thereby acquiring the roles' permissions[12].* Roles are created for the various job functions in an organization. Users are assigned the roles based on their responsibilities and qualifications. Users can be reassigned from one role to another. Roles can be granted new permissions as new applications and systems are incorporated, and permissions can be revoked from roles as needed. Figure 3.4 illustrates the most general RBAC96 model[12].

In figure 3.4, a single headed arrow indicates a one to one relationship. A double-headed arrow indicates a many to many relationship. The top half of the figure includes the roles and permissions in the system that controls the access to the data and the access to the resources. The bottom half includes the administrative roles and the administrative permissions.

RBAC96 is based on five sets of entities which are users (U), roles (R), permissions (P), their administrative counterparts called administrative roles (AR), and administrative permissions (AP). The administrative roles are disjoint from the regular (i.e., non-administrative) roles. The administrative permissions are disjoint from the regular permissions. Regular permissions can only be assigned to regular roles and administrative permissions can only be assigned to administrative roles. Intuitively, a user is a human being or an organization. A role is a job function or job title within the organization regarding the authority and responsibility. A permission is an approval of the access mode to one or more objects in the system. The user assignment (UA) and permission assignment (PA) and administrative permission assignment (APA) relations of Figure 3.4 are many-to-many relationships, as indicated by double-headed arrow. A user can be a member of many roles, and a role can have many different users. Also, a role can have permissions, and the same permission can be assigned to many different roles. There is a role hierarchy RH written as  $\geq$ . For example,  $x \geq y$  means that role  $x$  inherits the permissions which are assigned to role  $y$ . In another words, a user who is a member of  $x$  is also a member of  $y$ . For  $x > y$ ,  $x$  is senior to  $y$ , i.e.,  $y$  is junior to  $x$ . There is similarly a partially ordered administrative role hierarchy ARH.

The RBAC96 model has the following components [12]:

- $U$  is users set;
- $R$  are roles set and  $AR$  are administrative roles set;
- $P$  are permissions set and  $AP$  are administrative permissions set;
- $UA \subseteq U \times (R \cup AR)$ , is a many-to-many user to role relation, and many-to-many user to administrative role relation;

- $PA \subseteq P \times R$  many-to-many permission to role assignment relations, and  $APA \subseteq AP \times AR$  are many-to-many administrative permission to administrative role assignment relations;
- $RH \subseteq R \times R$  are role hierarchies, and  $ARH \subseteq AR \times AR$  are administrative role hierarchies;

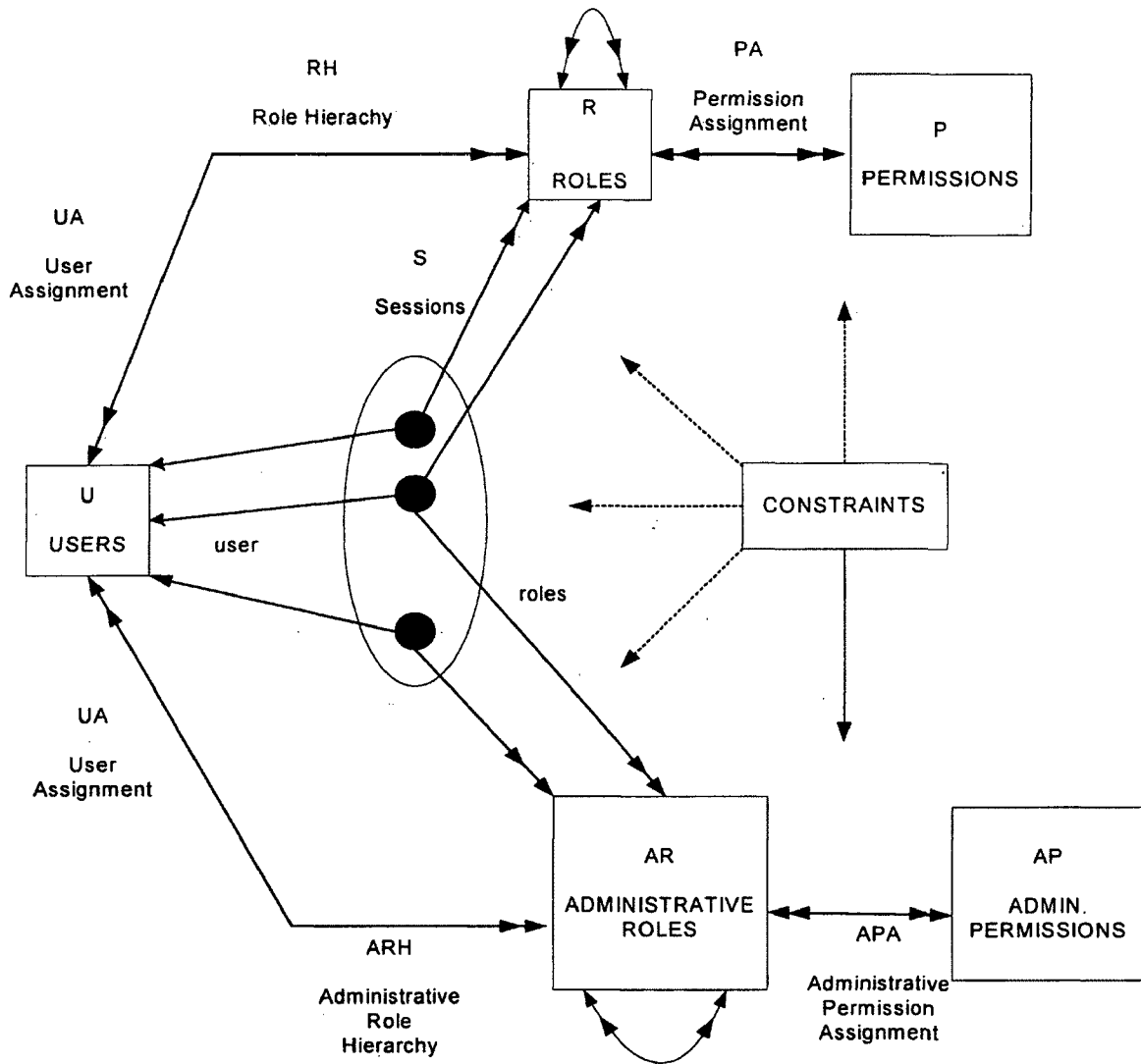


Figure 3.4 The RBAC96 Model

- $S$  is sessions set;
- $user: S \rightarrow U$ , is a function mapping from each session  $S_i$  to the single user  $user(S_i)$ . The relation is constant during the session's lifetime;
- $roles$ : is a function mapping each session  $S_i$  to a set of roles and function mapping each session  $S_i$  to a set of administrative roles;
- There is a collection of constraints forcing which action of the various above components are allowed or forbidden.

Each session in figure 3.4 relates one user to possibly many roles. Intuitively, a user establishes a session and activates some roles that he/she is a member of. The double-headed arrows from a session to  $R$  and  $AR$  indicate that multiple roles and administrative roles can be activated at the same time. The permissions available to the user are the union of permission from all roles which are activated in that session. Each session is associated with a single user, as indicated by the single-headed arrow from the session to  $U$ . A session is a unit of access control, and a user may have multiple sessions with different permissions active at the same time.

Figure 3.4 shows a collection of constraints. Constraints can be applied to any of the preceding components. An example of constraints is mutually disjoint roles, such as purchasing manager and accounts payable manager, where the both roles can not be assigned to the same user.

*Role-based access control has recently received considerable attention as a promising alternative to the traditional discretionary and mandatory access controls [12]. It greatly simplifies the management of permissions. It manages the access control*



in a DBMS through the role hierarchy, user hierarchy, and user-role relationship. The RBAC96 model can be used in conjunction with other traditional discretionary and mandatory security models in a DBMS. The RBAC96 model is readily implemented in existing database systems such as Oracle.

### 3.3 The Architecture of Secure DBMSs

Secure DBMSs operate according to two possible modes, one is 'system high' mode and another is 'multilevel' mode. In the 'system high' DBMSs, all the users are responsible for reviewing the data before properly releasing them. The users in this system are cleared to the highest security level, i.e. the highest clearance level. This mode allows the users to use the existing DBMS technologies without applying any changes, but it takes additional costs on the user-clearance procedure and the manual review of the data. In an Oracle database system, a more desirable mode, 'multilevel mode', is used. With 'multilevel' mode, the different types of architectures are based on the use of trusted DBMSs and untrusted DBMSs. The *multilevel* architectures include the *Trusted Subject Architecture* and *Woods Hole Architecture*. A trusted DBMS and a trusted OS are used in the Trusted Subject Architecture. The Trusted Subject Architecture requires the extension of the security features of an existing DBMS or the development of a new DBMS from the scratch. In the Woods Hole Architecture, an untrusted DBMS is applied with an additional trusted features added into the system. The Kernelized architecture that Oracle systems currently utilize is one of the Woods Hole Architecture.

The Kernelized architecture is shown in Figure 3.5. A set of trusted front-ends is used to interface users with different clearances, shown as High and Low in the Figure 3.5.

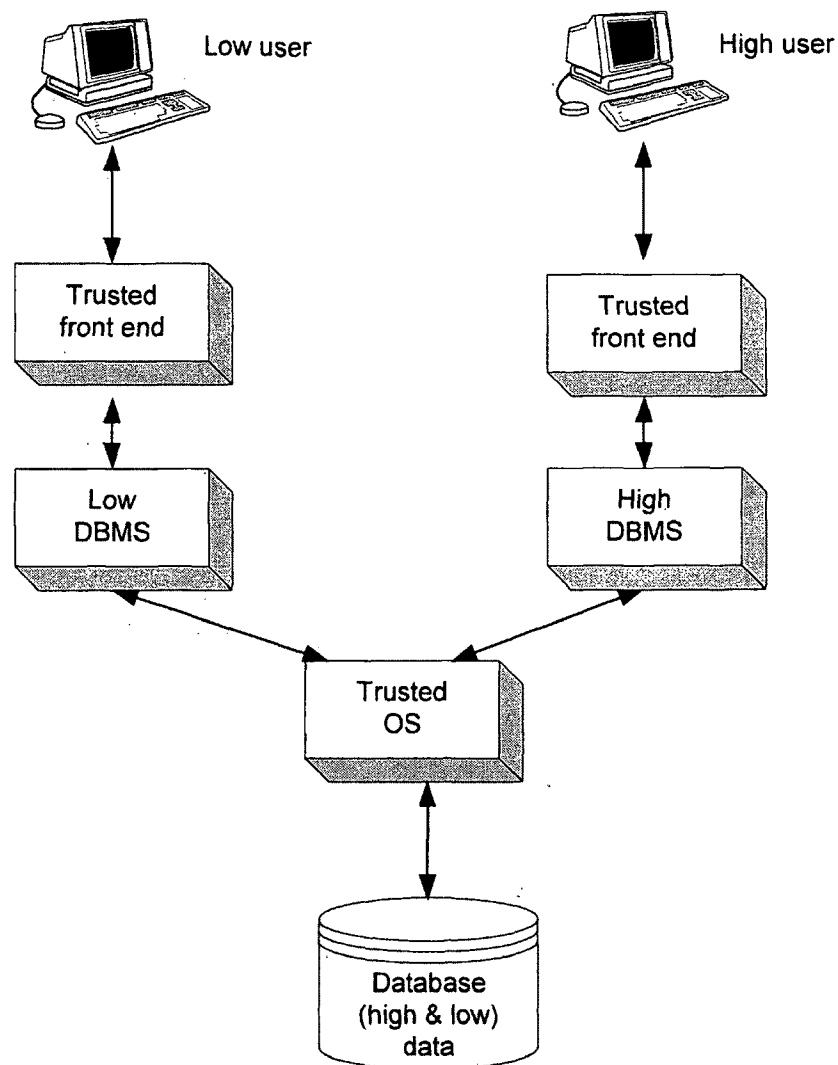


Figure 3.5 The Kernelized Architecture

A DBMS level, High and Low, is defined. The High level dominates the Low level. The users who operate at the High level interact with a High DBMS through a Trusted Front-End (TFE). The users who operate at the Low level interact with a Low DBMS through a Trusted Front-End. The TFE acts as a reference monitor, that is, it cannot be bypassed. The TFE is responsible for enforcing the security functions and multilevel protection.

Then, users' requests are passed to the underlying operating system. Here, a trusted operating system is used, which is responsible for performing the physical accesses to the data in the database, and for enforcing the mandatory protection of the physical datafiles. The trusted OS retrieves the proper data from the underlying database. It performs the security controls on the database objects. The multilevel decomposition and recovery processes, which must be properly defined to ensure the correctness and system efficiency, are required in this architecture. The decomposition process transforms a multilevel relation into several single level relations. Each single relation contains only the data which are at a given security level and are stored in the operating system objects. The recovery process generates a multilevel view that contains the data the user requests through query. This process is performed on the single-level relations when they are retrieved.

## **Chapter 4**

### **A Real Case: Oracle Security Database Implementation**

#### **4.1 Oracle and Security**

As a Database Administrator working on an Oracle Database Management System in a software company, I feel that database security is essential to ensure system continuity and reliability and to protect the data and programs from intrusions, modifications, theft and unauthorized disclosure. 70 percent of database failure reports are due to the serious failure of security, including the denial-of-service attacks and the theft of proprietary information from outside the system. At the same time, the security threats from inside the company are also growing. According to Information Security's 2000 Industry Survey, the number of companies experiencing the insider access control-related breaches increased by 12 percent from 1999 to 2000. The inappropriately configured database and hosting operating systems become the targets for attackers. Misconfigured database access controls also make the proprietary information accessible to unauthorized company employees, resulting in the loss or disclosure of valuable information [16].

We need to build a secure Oracle database to ensure:

- **Secrecy and confidentiality:** Data should not be disclosed to anyone who is not authorized to access it.
- **Accuracy, integrity and authenticity:** Data can not be maliciously or accidentally modified or corrupted. The origin of the data needs to be verified when needed.

- Availability and recoverability: The system can not be interrupted and data can be recovered completely with no loss of accuracy and integrity in case of damage.

Creating and enforcing security procedures helps to protect data what is rapidly becoming the most important corporate asset. There are all kinds of potential intruders who might want to access a database because they perceive that the data in the database are valuable to them. While storing that data in a database makes it more useful and available company wide, it also makes it susceptible to unauthorized access. Such unauthorized access attempts must be detected and prevented before the database is compromised.

The Oracle DBMS, a product of the Oracle corporation, is available for several operating systems, such as UNIX and Window NT. The UNIX version and Window NT version of Oracle have been certified at different security levels. Oracle has a multi-layered security model. It can protect the files and objects both inside and outside the database. It also can incorporate a variety of administrative policies and technical strategies. The layers of security that we can implement in an Oracle database system include the following:

- Controlling the access to the database;
- Controlling the access to the database tables through the views, triggers, roles, privileges, packages/procedures/functions;
- Controlling the access to the operating system files which make of the Oracle database;
- Protecting the application codes which interact with the database;
- Ensuring sufficient backup and recovery of the database.

Oracle makes several levels of security available to the Database Administrator which are the account security for validation of users, the access security for the database objects, and the system-level security for managing the global privileges. Oracle also has the ability to perform auditing at each level.

Oracle corporation releases new security patches quite often. In response to the growing need for database security, Oracle corporation continues to apply the robustness and functionality of the security features in the database packages. The security features in Oracle have matured as new product versions are released. From simple password protection in Oracle 7, Oracle has expended the security offerings and functionality with each release [16]:

- Oracle 7.1 added password encryption for security over the network.
- Oracle 7.2.3 added network encryption via a production called Secure Network Services (SNS).
- Oracle 8 added protection of password-cracking detection, account locking and strong password enforcement.
- Oracle 8i has gone even further in securing the database for the global community.
- Oracle9i minimizes the risk with unique, multi-layer security such as the access control to data at a granular or row level, ensuring the data confidentiality and integrity across the network, etc.

## **4.2 Security in an Oracle System**

- *Subjects:*

In Oracle, subjects are users and user groups who want to query the database. Oracle provides the *CREATE USER* command for creating the users, *ALTER USER* command for altering the users, and *DROP USER* command for deleting the users from the system. Using the *IDENTIFIED BY* clause of the corresponding *SQL* command, the administrator can choose between password-based and operating-system-based authorization of users. Through the special *ADMIN OPTION* command, a database administrator can grant system privileges to users who, in turn, can grant such privileges to other users. In Oracle, a role can be granted to other roles, making a hierarchical organization of privileges/roles. This role organization is similar to the organization of user groups. The *SET ROLE* statement is used to assign a role to an application, which is enabled via password. Users have default roles which are active at login time. Three basic privileges are defined as *Connect*, *Resource*, and *DBA*. The *Connect* privilege allows users to connect to the database, and to access the database objects, and to update the tables to which they have access rights. The *Resource* privilege allows users to create tables, and to grant access on the tables to other users. The *DBA* privilege is the highest privilege of the database that has all privileges including creating user accounts, creating roles, granting roles, granting privileges, etc. *Sys*, *System* and *Public* are three special accounts that are installed within the Oracle system. *Sys* and *System* have the *DBA* privilege. *Public* corresponds to the basic group of Oracle, and all the privileges that are granted to *Public* are automatically granted to all other user accounts.

- Objects:

The Objects that need to be protected are database objects. These objects in Oracle include databases, catalogues, tables, views, procedures, function, package, etc.

- Operations:

For Operations, the *SELECT*, *INSERT*, *UPDATE*, *DELETE*, *ALTER*, *INDEX* and *REFERENCE* privileges are available on tables. But only *SELECT*, *INSERT*, *UPDATE* and *DELETE* are available on views. The *EXECUTE* privilege is available for procedures. The *SELECT* privilege is available for Sequences. The *GRANT OPTION* is applied to enable the granted users to transfer the acquired privileges to other users. Only some DBMS servers provide this functionality, and the Oracle server provides it. The column-level control is supported through the *UPDATE*, *INSERT* and *REFERENCE* privileges. The *AUDIT* command of Oracle is used by the Database Administrator to audit operations that are related to the privilege, both successful and unsuccessful access statements, and the objects. Audit records can be stored in either a database or in the operating system audit trail.

#### **4.2.1 Oracle System Files and Database Objects**

The components of an Oracle database system can be divided into two main areas - Oracle operating system files (the physical entities) and Oracle database objects (the logical entities). In our company, we currently use a Windows NT version of Oracle. The physical files should be protected in the operating system level. Window NT's weaknesses also affect the Oracle security performance. For example, the Oracle's media failure, which causes the database to go down, is related to Window NT. But, Database Scanner provides extensive OS-level checks on file permissions. Furthermore, the DBA can set up the roles/privileges and create the view/triggers within the framework of Oracle to ensure the security of the database from the logical entity level.



#### 4.2.1.1 Oracle System Files

The following files are significant from a security view. It is very important that we can protect these files to the full extent allowed by the system.

Figure 4.1 shows the components of an Oracle database system after startup.

- *The physical files that make up the database tablespaces:* These physical files include the control file, redo log files, archive log files, datafiles containing data, datafiles containing indexes, datafiles for making the rollback segments, and datafiles for making the temporary segments.

The files that are used for creating the *tablespaces* can be placed on the computer's storage disks. But, file protection should prevent any user except the Database

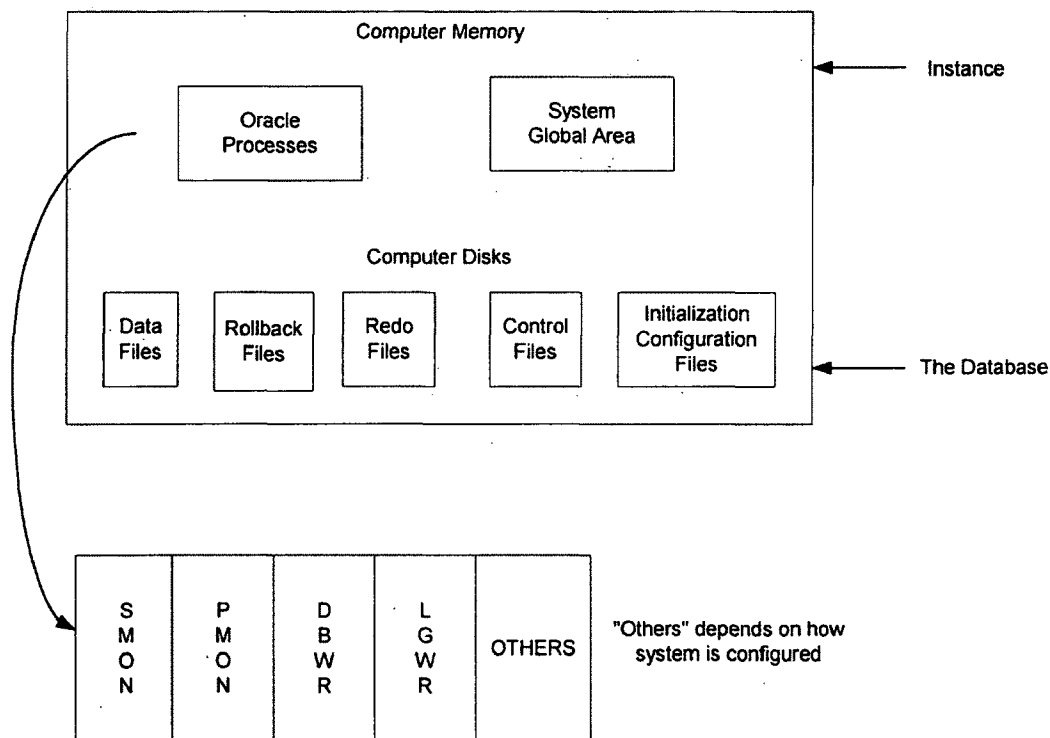


Figure 4.1 Components of the Database System After Startup

Administrator (DBA) from ever being able to access and manipulate these files directly from the operating system level. Modification of these files is the sole responsibility of the Oracle processes. From a security view, maintaining tablespace security is the access control of the files on the operating system level. The underlying files for making the tablespaces must be owned by the account that is used to install Oracle software usually named '*oracle*'.

*Redo log files* are a set of binary storage file that is used to keep track of all changes made to the database. At least two online redo logs are associated with each database. Only one redo log or redo log group is in use at a time. The online redo logs are written and reused in a circular fashion. After each redo log file is filled, an *archive log file* is created with a copy of the filled redo log file. These *archive log files* are the companion files to the redo log files. The archive log files are sequentially numbered so that the old archive log file will not be overwritten by the new archive log file.

Redo logs are primarily used for database recovery from a disk crash. In the event that the database was corrupted or damaged because of a malicious action or a break-in action, the database can be restored using the redo logs and archive redo logs to bring the database back to a good state prior to the corruption. Redo logs and archive logs coupled with scheduled file-level backups could recover the database and bail us out of a disaster situation.

The redo log files are created and owned by the Oracle *system* account and should never be made accessible to any other user for any purpose. From a security view, it is better to use redo log groups rather than a single set of redo log files in the database system. For example,

```

CREATE DATABASE sample_db1

DATAFILE m:/oracle/database/sample_db1/data/user_data.dbf SIZE 50M reuse

CONTROLFILE REUSE

LOGFILE GROUP 1 (M:/oracle/database/logfile/sample_db1_log01a.dbf,
                 H:/oracle/database/logfile/sample_db1_log01b.dbf) SIZE 10M,

LOGFILE GROUP 2 (M:/oracle/database/logfile/sample_db1_log02a.dbf,
                 H:/oracle/database/logfile/sample_db1_log02b.dbf) SIZE 10M,

LOGFILE GROUP 3 (M:/oracle/database/logfile/sample_db1_log03a.dbf,
                 H:/oracle/dataabse/logfile/sample_db1_log03b.dbf) SIZE 10M,

MAXDATAFILES 300;

```

In the above example, there are three sets or groups of redo logs with two members each on two separate disks *M* and *H*. The files *sample\_db1\_log01a.dbf* and *sample\_db1\_log01b.dbf* will be written on disks *M* and *H* respectively. At switch time, both *sample\_db1\_log02a.dbf* and *sample\_db1\_log02b.dbf* files will be opened for writing. By creating the log files in the set of files, preferably on two or more different disks, if one member of a set is damaged, the other member of the set will be available to use for quick database recovery. In our company, we recently restructured the redo logs and adopted using redo log groups instead of the old only one file defined for each redo log.

- *The initialization parameter file:* This file with the name *INIT<database\_sid>.ORA* contains Oracle's initialization parameters. There is no information in the initial file that can be accessed by the users except the DBA, so it should be protected the same way as the other datafiles so that the users can not read it. The initialization file can

only be modified and maintained by the DBA and should be available only through the Oracle *system* account. A wise DBA should always insert an entry in the bottom of the file reflecting each change that has been made to this file in order for the database maintenance.

- *Control files*: The information in the control file is very important for the operating and recovery of the database. The database can not start up without the control file. The control files, like other Oracle files, should never be available to any user and must be owned by the Oracle install user. Each database has at least two control files located on different disks, and preferably more copies. The information in all of the copies is identical and the copies are placed in the different directories on the different disks. If one of these control files on a system becomes damaged and the database can not start up, when the specific damaged control file is determined, we can replace the damaged control file with one of the other copies of the control file from a different disk and then restart the database.
- *The configuration file*: The file named as *CONFIG.ORA* contains the configuration information about the database. This file includes the information of the database name, the location of the control files, the location of the dump files and the database block size, etc. This file can only be modified and maintained by the DBA through the *oracle* installation account and should be available only to the *oracle* system account.
- *Network configuration files*: These files include *LISTENER.ORA*, *TNSNAMES.ORA* and *SQLNET.ORA*. These files are used for the connection between client's workstations and the database server.

- *The Oracle distribution files:* These files include the code for building, connecting and maintaining the Oracle databases. Some usernames and passwords are embedded in the Oracle distribution files. The users are prevented from accessing and modifying these system files.

#### **4.2.1.2 Oracle Database Objects**

Within the database, we can grant the users access to the Oracle database objects or revoke the users access from the Oracle database objects such as tables and views. There are different levels and forms of access to the data within the database tables. We can control the access by creating views and only granting the selected users the access rights to the views instead of granting the selected users access to all underlying tables. Also, we can create triggers and procedures to perform the actions behind the scenes to control the security and protect the data in the database.

- *Tables:* Tables are basic building blocks used to store data in the database. A table is a file that is created and maintained with the data files. The user who creates a table usually owns that table. Users can access another user's table only if the owner of the table or a DBA grants the privileges on the table to them. The privileges on these tables can be granted directly to the users, to a special user named '*public*', or to a role.

The privileges on table level include *SELECT*, *DELETE*, *UPDATE* and *INSERT*. We can also grant the privileges on a column of one table to users. The privileges on column level also include *SELECT*, *DELETE*, *UPDATE* and *INSERT*. For security and monitoring purpose, tables are preferably created with extra fields to capture the

username, the time of creating the data, and modification action performed. In our database for each database table, we require to have four common fields to capture the creation action and modification action information of a table. These four fields are *CREATED\_BY*, *DATETIME\_CREATED*, *MODIFY\_BY* and *DATETIME\_MODIFY*.

- *Views*: Within an Oracle database, views are used to enable a subset of information to be extracted from a table or from several tables. Views can be used for several security purposes. For example, views can control user access by pre-joining tables to limit the data they can retrieve. When the view is created, its definition specifies the column to be retrieved from the tables. This approach provides column-level security. We can grant access rights on the views to selected users instead of granting full access on the tables to selected users.

Figure 4.2 shows how a user would access the view and how execution of the view causes required data to be retrieved from several tables.

- *Triggers*: Triggers are stored programs associated with a table. A trigger is fired either before or after an event which is the set of commands that performs *INSERT*, *DELETE*, or *UPDATE* actions. From a security standpoint, triggers are very useful to track activities that change the data and prevent the unauthorized access to the data.
- *Stored programs*: The stored programs written in PL/SQL can be stored in a compiled form within the database. There are two types of stored programs which are the procedures and functions. For ease in maintenance, the schema owner owns all

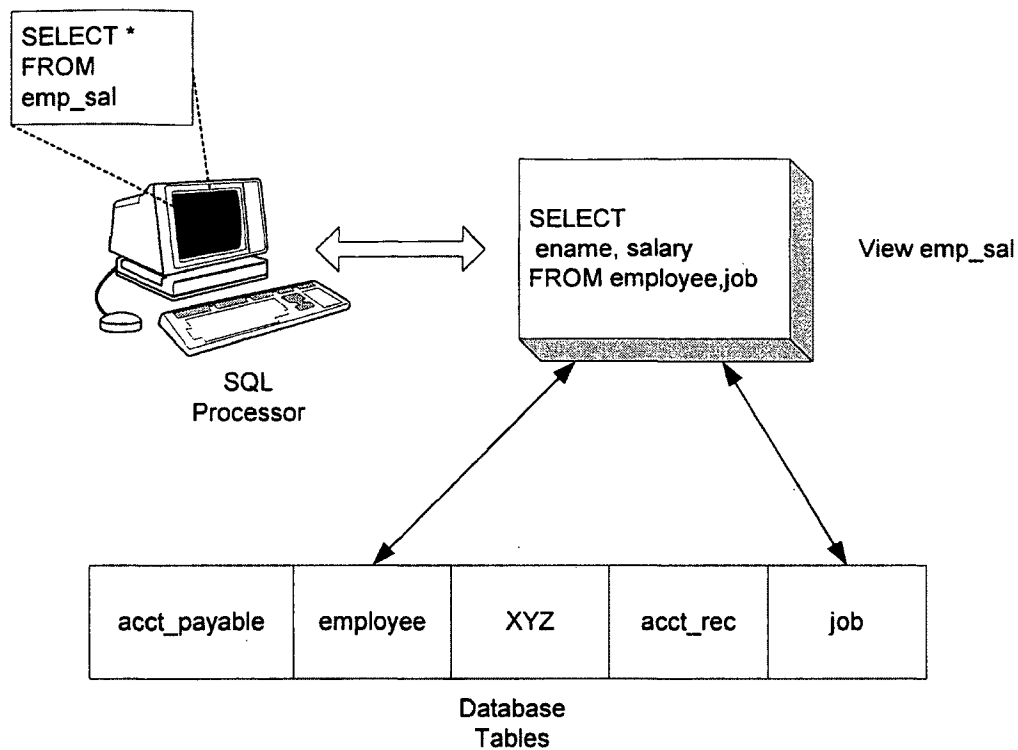


Figure 4.2 Oracle View Implementation

stored programs. Users or applications can execute stored programs if the *EXECUTE* permission on the program has been granted to the users.

Triggers or stored procedures can populate another table with information based on the action being taken on the primary table. They also can remove information from another table based on the action being taken on the primary table. Triggers or stored procedures capture information about the data before the data is modified. They can issue an alert to let someone know a table has been modified. Through views, triggers, and stored procedures, we can control the user access to a table.

When we deliver our application code outside of our own company, it is recommended to run the *PL/SQL Wrapper* utility, so that the readable ASCII text

source code can be converted into unreadable code. Then the unreadable code can be delivered to anywhere in the Oracle database and its contents are secured.

- *Synonyms*: The synonym is an alias name of an object. A synonym is used in an Oracle database to provide the location transparency of objects and object owners. By using the synonyms, the user not only does not have to know who actually owns the table, but also does not have to know in which database the table is located. The use of synonyms helps to hide the underlying database structure. This also helps to protect the database from curious or malicious break-in actions.

## 4.2.2 Oracle Data Dictionary

Oracle's data dictionary consists of two layers which are the tables that make up the real data dictionary, and a series of views that allow us to access the information in the data dictionary. Oracle's data dictionary includes all database objects regardless of who owns the objects. Users are only allowed to look at the data in the data dictionary based on their job requirements. Users can only see the information which is appropriate for their level of privilege. The privileges granted to the users on the data dictionary views are different from the privileges granted to the underlying tables. The data dictionary views include the views showing the data about a user's own objects labeled "*USER\_*", the views showing the data available to any user in the database are labeled "*ALL\_*", and the views showing the data available to only the Database Administrator labeled "*DBA\_*". Different users with different privileges can access the different objects. Table 4.1 lists some important views for the Oracle database security [4].



A user granted the DBA role can access and possibly modify any objects within the database. Some of the dictionary views can only be accessed by the users with DBA

View Name	Type of Information Available	Tables on Which View Is Built
DBA_PROFILES	Profiles and their associated resource and time limits	profile\$, profname\$, resouce_map, obj\$
DBA_ROLES	All roles that exist in the database	user\$
DBA_ROLE_PRIVS	Roles granted to users and other roles	user\$, sysauth\$, defrole\$
DBA_SYS_PRIVS	System privileges granted to users and roles	user\$, sysauth\$, system_privilege_map
DBA_TAB_PRIVS	Privileges like SELECT, INSERT, UPDATE, etc. that each user or role has per object	user\$, objauth\$, obj\$, table_privilege_map
DBA_USERS	Who has an account for the database; also, which profile is assigned to the user	user\$, ts\$, profname\$, profile\$, user_astatus_map
ROLE_ROLE_PRIVS	Roles granted to roles	user\$, sysauth\$
ROLE_SYS_PRIVS	System privileges granted to roles	user\$, system_privilege_map, sysauth\$
ROL_TAB_PRIVS	Table privileges granted to roles	user\$, table_privilege_map, objauth\$, obj\$, col\$
USER_ROLE_PRIVS	Roles granted to the current user	user\$, sysauth\$, defrole\$, and x\$skzdos

Table 4.1 Oracle Database Data Dictionary Views

privileges. When we described the view *DBA\_USERS*, *ALL\_USERS*, and *USER\_USERS*, we can see a different number of columns and different column information for each view:

*SQL> desc all\_users;*

<u>Name</u>	<u>Null?</u>	<u>Type</u>
USERNAME	NOT NULL	VARCHAR2(30)
USER_ID	NOT NULL	NUMBER
CREATED	NOT NULL	DATE

SQL> desc dba\_users;

<u>Name</u>	<u>Null?</u>	<u>Type</u>
USERNAME	NOT NULL	VARCHAR2(30)
USER_ID	NOT NULL	NUMBER
PASSWORD		VARCHAR2(30)
ACCOUNT_STATUS	NOT NULL	VARCHAR2(32)
LOCK_DATE		DATE
EXPIRY_DATE		DATE
DEFAULT_TABLESPACE	NOT NULL	VARCHAR2(30)
TEMPORARY_TABLESPACE	NOT NULL	VARCHAR2(30)
CREATED	NOT NULL	DATE
PROFILE	NOT NULL	VARCHAR2(30)
EXTERNAL_NAME		VARCHAR2(4000)

SQL> desc user\_users;

<u>Name</u>	<u>Null?</u>	<u>Type</u>
USERNAME	NOT NULL	VARCHAR2(30)
USER_ID	NOT NULL	NUMBER
ACCOUNT_STATUS	NOT NULL	VARCHAR2(32)
LOCK_DATE		DATE
EXPIRY_DATE		DATE
DEFAULT_TABLESPACE	NOT NULL	VARCHAR2(30)
TEMPORARY_TABLESPACE	NOT NULL	VARCHAR2(30)
CREATED	NOT NULL	DATE
EXTERNAL_NAME		VARCHAR2(4000)

### 4.2.3 Other Elements of Oracle Security System

The following objects are the logical entities in an Oracle database. Oracle protects and maintains these entities and uses them to enforce security in the database system.

- **Users:** The Database Administrator is responsible for maintaining the database user accounts. The DBA creates a user account, assigns a password, defines a default tablespace/temporary tablespace/quota for the user. The users are granted the appropriate privileges to connect to a database to perform their work on the database system. The privileges are granted to users by the DBA based on the user's required job responsibilities.

- *Schema*: Schema is the complete collection of objects owned by a user account. The schema includes the tables, views, sequences, packages, functions, procedures, triggers, etc.

Only the owner of an object or a user with DBA role can grant the access privileges to other users for interacting with the objects. DBA should be very careful when granting the access rights on any objects to any users.

- *Privileges*: Privileges include two general categories which are the system privileges and the object privileges. The access rights for users to log on to the database system and to create/manipulate the database objects are the system privileges. The access rights for the user to access the data within a database object, or to allow the user to execute a stored program are the object privileges. There are more than 80 system privileges on the Oracle database system. For example, *CREATE TABLE*, *CREATE SYNONYM*, *CREATE ANY VIEWS*, *CREATE SEQUENCES*, etc. There are total of 16 object privileges in an Oracle database. For example, *ALTER*, *AUDIT*, *SELECT*, *UPDATE*, *EXECUTE*, *REFERENCES*, etc.
- *Roles*: A role is a collection of privileges. A role can be assigned to the users. The function of a role is to allow the group of privileges to be passed to the users by referencing the role. If the role's privileges are changed later, the new privileges will be in effect when the users log into the database at the next time. Figure 4.3.1 shows an example of role hierarchical organizations in an Oracle Database System. Figure 4.3.2 shows an example of an Administrative Role Hierarchy in Oracle Database System.

Oracle supplies several default roles with the installation of the database. Table 4.2

shows the major default roles in an Oracle database system.

<i>Role Name</i>	<i>Type of Privileges</i>
CONNECT	Allows login and ability to create tables, views, synonyms, and database links.
RESOURCE	Add cluster, procedure, and trigger privileges.
DBA	Complete authority to manage database and users. Can create users.
SYSOPER	Ability to start up and shut down the database.
SYSDBA	All privileges available to the DBA role with the ability to create, start up, shut down, and recover a database.

Table 4.2 Oracle Default Roles

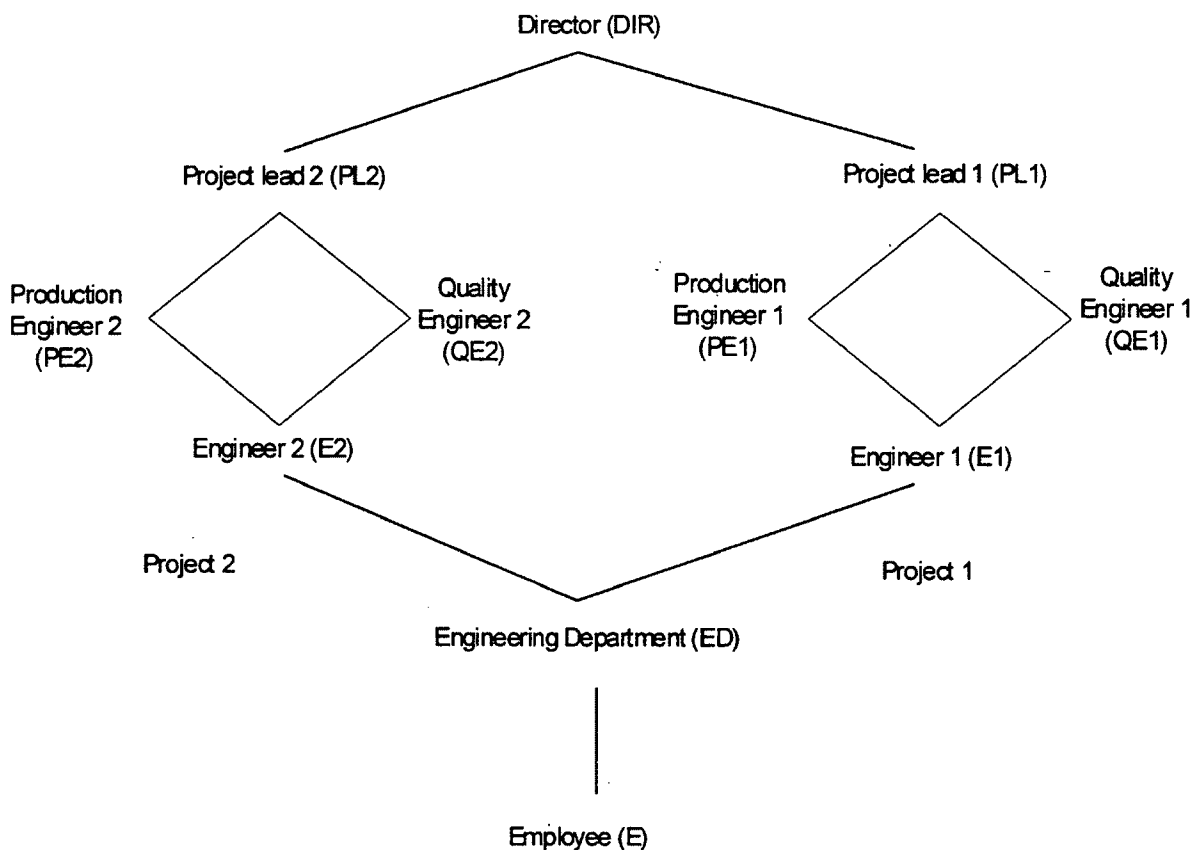


Figure 4.3.1 An Example of Role Hierarchy in Oracle DB System

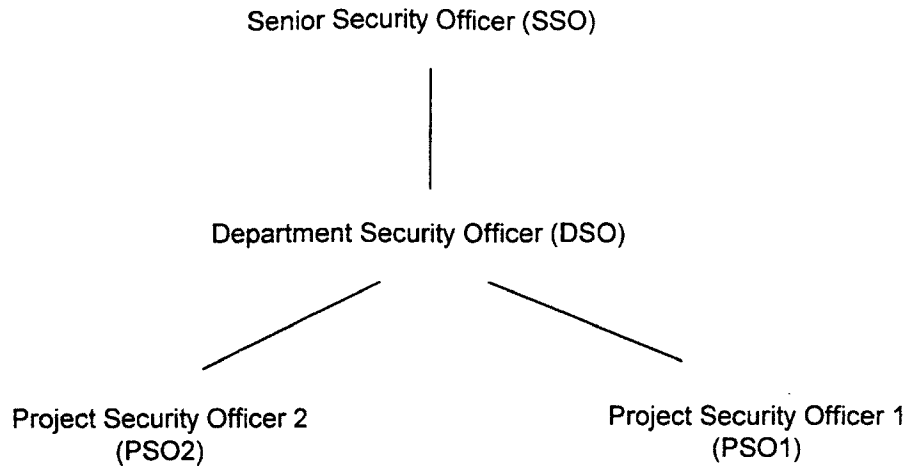


Figure 4.3.2 An Example of Administrative Role Hierarchy in Oracle DB System

- *Profiles:* Oracle provides two different types of profiles which are the product profiles and the system resource profiles. Product profiles limit user access to certain Oracle commands or Oracle products. System resource profiles limit a particular user's use of certain system resources. Using product profiles, we can block users' access to Oracle's SQL, SQL\*Plus and PL/SQL product.

The *PRODUCT\_PROFILE* table provides the mechanism used to block commands. We can use *PRODUCT\_PROFILE* to enforce security. When users log into the database, the user has minimum privilege directly through SQL\*Plus. However, a user could use the *SET ROLE* command from SQL\*Plus to gain access privileges normally enabled only when the application is used. Then, the user could issue SQL statements from SQL\*Plus to change database tables which is detrimental

to the security of the database system. We need to disable the *SET ROLE* privilege for the specific users to prevent this damage.

The system resource profile is a set of restrictions or limits that can be placed on a database resource. These resources include the amount of time which a process can be idle before the process will be disconnected, the number of failed login attempts before an account is locked, etc. For example;

```
CREATE PROFILE db_profile LIMIT  
SESSIONS_PER_USER 6  
CPU_PER_CALL 3000  
CONNECT_TIME 60  
FAILED_LOGIN_ATTEMPTS 5  
PASSWORD_LOCK_TIME 1/12
```

### **4.3 Methodologies of Designing A Secure Oracle Database**

Before discussing the methodologies of designing a secure Oracle database, I would like to mention that the Oracle database management system currently uses the thin-client architecture. Figure 4.4 shows the three-tier thin client architecture. Using this architecture, it enables [17]:

- The Low resources on the client side to have minimal security requirements.
- The medium amounts of resources on the application server to have higher security requirements.
- The High resources requirements for one or more back end servers to have varying amounts of security requirements based on the different application which a particular database supports.

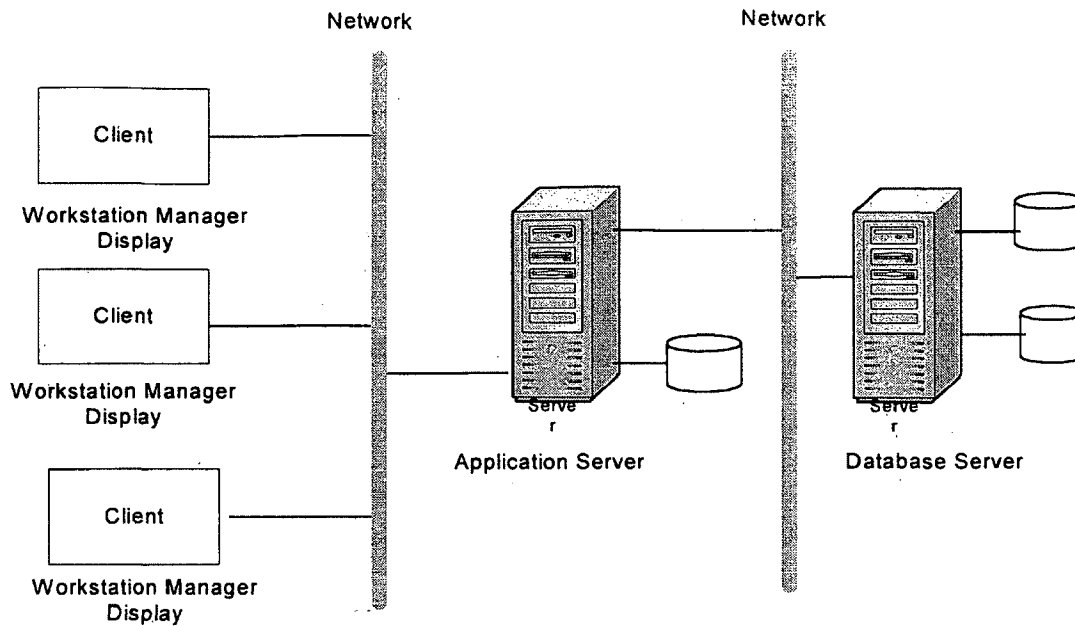


Figure 4.4 The Thin Client Architecture in Oracle DB System

### 4.3.1 Developing a Database Security Plan

It is very important to create policies and a security plan in securing the database system. And this is the first step in the secure database system development. Before we create a database, we need to identify and address all of the necessary issues:

- Collect and be familiar with all the database requirements and information; set up a uniform approach to security across the computer systems and databases.
- Identify all affected applications on each system; determine the access requirements based on each different application.
- Identify all of the key players and the managers in the database and identify the specific types of accounts for the operating system and database system.

There are several different types of Oracle users: *Database Administrator* and *security manager* who is responsible for creating user accounts and monitoring user access and the database security, *application manager* who is responsible for handling the administrative tasks for the specific database in which their application is housed, *network manager* who is responsible for administering the network products and Names Servers and network configurations for Oracle databases, *application schema* who is the owner of a specific database and may require more privileges than a general user, and the *general user* whose privileges are determined within the application.

- Identify the username and password structure in the database system.

The structure includes password aging and expiration, password reuse, failed login attempts, account locking and unlocking.

Password aging and expiration: To help to ensure that a password will not be compromised, the password needs to be changed on a system at least every three months (recommended). The longer a password remains in effect for an account, the greater the possibility that the password can be compromised.

Password reuse: It is recommended to exclude a password that has been used from being used by that person again.

Failed login attempts: The numbers of failed login attempts that will be tolerated in a system should be determined.

Account locking and unlocking: The decision might be made to never enable account locking or never enable automatic account unlocking. If account locking is going to



be enabled, we need to define the personnel who will be in charge of performing the account unlocking.

- Determine who has the authority to approve the accounts and who is responsible for creating/dropping/managing accounts.
- Determine a user tracking system and implementation.

Identify how the users are tracked to ensure that when an employee's job description or location changes, the access to the database remains correct.

- Establish the standards and structures for the roles, privileges, views, triggers, etc.
- Determine what constitutes a security breach and the appropriate penalty for each breach.
- Identify all sensitive information/data on the database system and define the methods/steps to protect these data.
- Determine the types of monitoring to be used.
- Determine the types of backup to be used and define the recovery procedures for the system.

### **4.3.2 Developing an Audit Plan**

Database auditing is the monitoring and recording of activities occurring within a database. The audit helps to:

- Ensure that no unauthorized users can access the tables to which they do not have the rights to access;
- Ensure that no unauthorized users can remove data from the data dictionary that they do not have the privileges to see;

- Audit the specific tables that help to determine the volume of access occurring at peak times.

The audit includes the *Security Auditing* and the *Performance Auditing*. The Security Auditing is to determine if someone is attempting to break into the database system. When some problem has occurred in the system, this audit helps to determine the reasons. For example, we found that some data might have been deleted from a table in which the data should not be dropped. The audit can be used to track the deletions in that specific table. The Performance Auditing is used to determine the reason why the system is slow. This audit helps to determine the volume of traffic interacting with the specific areas of the database. The audit trail can be written either directly to an operating system file or to the database. Two parameters in the *INIT.ORA* file control the auditing actions:

- **AUDIT\_FILE\_DEST:** This parameter tells Oracle the directory name in which the audit trail is to be written to. The default value for this parameter is *\$ORACLE\_HOME/RDBMS/AUDIT*.
- **AUDIT\_TRAIL:** This parameter tells Oracle to enable or disable auditing. If the parameter is set to be *NONE*, this means that no non-default auditing will occur. If the parameter that is set to be *OS*, this means that system-wide auditing will occur and the results will be written to a file in the directory which is defined through the *AUDIT\_FILE\_DEST* parameter. If the parameter that is set to be *DB*, this means that system-wide auditing will be turned on and the results will be written to a table *SYS.AUD\$* in the *SYS* schema in an unencoded, readable format. Oracle provides some standard auditing views for performing the auditing in the system. For example,

*DBA\_AUDIT\_ONJECT, DBA\_AUDIT\_SESSION, DBA\_AUDIT\_STATEMENT, DBA\_AUDIT\_TRAIL, DBA\_AUDIT\_ONJECT, AUDIT\_ACTIONS*, etc.

There are three different types of auditing in the Oracle database system which are statement-level auditing, system-level or privilege auditing, and object-level auditing.

- *Statement-level auditing*: This auditing includes the auditing for Data Definition Language statements and the auditing for Data Manipulation Language statements. For example, a statement-level audit can audit any action performed on the tables, such as *CREATE TABLE, ALTER TABLE, DROP TABLE, TRUNCATE TABLE*, etc. If we wanted to capture the number of times the database was accessed but the attempt to log on failed, we could use the following command:

```
SQL> AUDIT CONNECT WHENEVER UNSUCCESSFUL;
```

The summary table for this audit might contain a count, by 24-hour period, of all of the failed attempts to connect to the database. The hacker's attempts to guess the correct password for an account are recorded as unsuccessful logons. A higher than normal number of failures in the summary table might indicate that someone was trying to break into the database by guessing account names and passwords. Through this auditing, we will be able to identify an unauthorized person's attempts to gain the access to the database system.

- *Privilege auditing*: We can enable the auditing on specific privileges via the *AUDIT* command. The monitoring of who holds which privileges is a manual or automated task which is performed through *SQL* code, for example:

```
SELECT grantee, privilege
```

```
FROM dba_sys_privs

WHERE GENTEE NOT IN ('SYS', 'SYSTEM', 'RECOVERY_CATALOG_OWNER',
'IMP_FULL_DATABASE', 'EXP_FULL_DATABASE', 'DBA', 'CONNECT',
'RESOURCE');
```

Through this auditing, we can audit who has been granted the specific privileges on the database. The hacker will have a much harder time to build 'hidden' accesses to the database system.

- *Object-level auditing:* Through this audit, we can audit for inserts to a table, updates to a table, deletions of rows in a table, or the viewing of information contained in the table. By using the audit table and creating the 'before update' trigger, the appearance of the data can be effectively captured prior to allowing the changes to occur.

It should be mentioned that for each action being audited, there may be a large volume of information generated. If too much is audited, then no one will have time to look at the generated log files. For this reason, we may need to create a summary table to separate and/or summarize the information of interest from the total volume of information collected. By creating a smaller table and populating it with a summary of information, we accomplish two very important things [25]:

- We enable removal of information from the underlying SYS.AUD\$ audit table to conserve storage space.
- We gain easier access to our trending data.

Also, we should protect the audit trail so that audit information cannot be added, modified, or deleted. We can issue the following command:

```
AUDIT delete ON sys.aud$ BY ACCESS;
```

To protect the audit trail from unauthorized deletions, only the DBA should have the *DELETE\_CATALOG\_ROLE* role.

### 4.3.3 Backing Up and Recovering the Database

Availability is a key tenet of computer security. In every database system, backup and recovery facilities are integral to keeping the system and data available for use. Backup and recovery facilities keep a loss from becoming a disaster [11].

#### 4.3.3.1 Backing Up the Database System

There are several different forms of backups that can be performed to ensure the data is available to the database. These backups include:

- *Cold database backups*: This backup is performed when the database is shut down.

The DBA must ensure that no processes are active within the database when the attempt is made to shut down the database. All files that make up the database are copied either to another disk or to a tape after the database is shut down. These files include all datafiles, redo logs, control files, and the *INIT.ORA* file. Many different system-dependent utilities can be used to perform the actual backing up of the files.

In our company, we use the Backup Manager to back up the database. We shut down our databases at 10:00pm every night for performing the backup. After the databases are backed up successfully, the databases are restarted at 5:00am every morning.

If a disaster occurs, the DBA would first save off the current redo log files and control files to a separate disk or tape, then restore the files from the backup. After

this, the DBA would replace the older log files with the current saved redo log files and roll forward through the archive log files until the point just prior to the disaster.

- *Hot database backups:* The database remains up and running while the hot database backup is performed. During the hot backups, the datafiles corresponding to the tablespace are copied to the separate disk or tape as with the *cold database backups*. To begin a hot backup of a tablespace's datafiles, the following command is issued:

*ALTER TABLESPACE <tablespace\_name> BEGIN BACKUP;*

- *Logical database backups:* A logical database backup involves reading a set of database records and writing them to a file. This backup includes a full database export, an export by user or a full schema export, and an individual table or set of tables export. To perform an export of a database, the database must be up and running. When we plan to make a big database change to a schema, we usually export this schema before making any changes. Using an export file, the database can only be recovered to the time at which the export was performed. The export files are very portable and can be used to move or copy a database from one system to another system.

Between these backups, the off-site tape storage backup (i.e. cold database backup) is a whole database backup, which performs a backup on all physical files and all database objects. When the off-site file-level copy is completed, the database is restarted. When the copy is completed, if a disaster occurs, we can recover all the files which make up the database from the tape. The database can be recovered to the condition it was in when the copy was made. When an off-site tape storage is used in conjunction with archive log files, the database can be recovered to a 'point in time'.

Also, we can use the file-level copy on the tape to recover one or more individual tables.

#### **4.3.3.2 Recovering the Database System**

There are several different ways in which a database can be recovered. Generally, there are two approaches:

- Recovery from export or file-level backups without the archive logging enabled.
- Recovery from the file-level backups plus archive logs.

More specifically, there are three basic types of recovery – online block recovery, thread recovery, and media recovery.

- **Block-level recovery:** This recovery is automatically performed by Oracle during normal operation of the database. When Oracle detects a corrupted block in the cache, it attempts to pull the block off disk and recover it using the online log file.
- **Thread recovery:** If a database crashes while it has the database open, it is necessary to do thread recovery. This involves applying to the database all the redo changes in the thread that have occurred since the last time the thread was checkpointed.
- **Media recovery:** It is used to make backup data files current or to restore changes that were lost when a data file went offline without a checkpoint by Oracle. The database can not be opened if any of the online data files needs media recovery. There are primarily three options we can choose while doing media recovery.
  - **Database recovery:** This means that we can restore all (or some) data files from the backup and recover the entire database.

- Tablespace recovery: We can perform media recovery on a specific tablespace. This means all datafiles that belong to the tablespace will be recovered.
- Data file recovery: we can recover a specific data file while the rest of the database is in use.

The database recovery can be performed online, i.e. with the database open. The recovery can also be performed offline, i.e. with the database closed. For an online recovery, within the database, we can use the commands *RECOVER TABLESPACE* and *RECOVER DATAFILE*. From the outside of the database, we can use the Import utility to recover the entire database, schema, or tables from an export file. The import include the full database mode, schema-level mode, and the table or set of tables mode.

For an offline recovery with the database closed, we can recover the *system* tablespace as well as an entire database. Using this method, if a tablespace has been corrupted, the DBA can alter the datafiles with the *OFFLINE DROP* option and rebuild the tablespace, and restart the entire database.



## **Chapter 5**

### **Conclusions**

#### **5.1 Main Results**

This project consists of two parts. The first part is to do some research on the Database System Security. The second part is to do some research on a real Database Security System: the Oracle Database Security System.

The DBMS, which is discussed in this project, is based on the relational model, rather than the network model, the hierarchical model or other models. The relational model, which provides high independence of the physical structure of data and allows a variety of operations and queries that are not bound to the underlying physical features, is the most up-to-date, popular and useful data model for modeling a database. In Chapter 2, the concepts of Database Management System (DBMS), the architecture of a DBMS, the data description levels of a DBMS which includes the conceptual level, the internal level and the external level, and the relational model were briefly discussed. Then, the security problems in a database and the database security protection requirements were examined. The access control, the flow control and the inference control were illustrated in this Chapter since the database protection can be obtained through the security measures for these three controls. The structure of a Database Management System including the security features was also presented in this Chapter.

A DBMS should be designed with security in mind to ensure that the security features of a DBMS are considered from the earliest development stages. In Chapter 3, the concerns and the security measures of a secure DBMS design were illustrated. The

basic elements of DBMA security models were presented. Generally, the database security models are classified into two traditional categories: the discretionary security models and the mandatory security models. In Chapter 3, we studied The Wood Model, which is the representative of the discretionary security models, and The Dion Model, which is the representative of the mandatory security models. A role-based access control model (RBAC96 Model) which is currently applied in the Oracle database system for the security control was also presented in detail in this Chapter. The architecture of a secure DBMS, specifically the Kernelized architecture which the Oracle database system currently utilizes, was illustrated here.

In Chapter 4, we studied the Oracle subjects, system files, database objects, data dictionary and other elements of the Oracle security system, which are significant from a security view. The basic methodologies of designing a secure Oracle database, which includes developing a database security plan, developing an audit plan, and developing a backing up and recovering the database system plan, were discussed and illustrated in this Chapter. These research results, considered to be valuable and very useful, were adopted by our company and applied into our current Oracle database system. For example, in our company, at the beginning stage of the database system development, a database security plan is required to be set up based on the security plan checklist that we developed during work on this research project.

## **5.2 Future Research Directions**

There are some other interesting research topics in the Database Security area, for example, the application of computer immune systems to the distributed database

systems. An immunological model of distributed detection has been designed and developed for computer security. One of the domains that is suitable to the model of distributed detection is the distributed database. Consider a database that is distributed across many locations in a network. These locations can consist of computers connected in LANs or individual computers. The communication between systems at different locations may have to pass through multiple intermediate local system. Natural immune systems protect animals from dangerous foreign pathogens, including bacteria, viruses, parasites, and toxins. Although there are many differences between the living organisms and computer systems, the similarities are compelling and could point the way to improve the computer security. The immune system defends the body against harmful diseases and infections. To do this, it performs pattern recognition tasks to distinguish molecules and cells of the body (called “self”) from dangerous foreign ones (called “nonself”), and then eliminate nonself from the body. The problem of protecting computer database systems from malicious intrusions can similarly be viewed as the problems of distinguishing self from nonself. Nonself might be an unauthorized user, foreign code in the form of a computer virus or worm, unanticipated code in the form of a Trojan horses, or corrupted data. This is a very interesting and potential research topic in the Database Security area [19].

There are some other interesting and potential database security research topics including the design of multilevel secure database systems, the application of LOCK data views, etc [1].

## Bibliography

- [1] T. F. Lunt, Research Directions in Database Security, *Springer-Verlag*, 1992.
- [2] T.Y.Lin and S. Qian, Database Security XI Status and Prospects, *Chapman & Hall*, 1998.
- [3] R. Clark, S. Holloway and W. List, The Security, Audit and Control of Databases, *Avebury Technical*, 1991.
- [4] K. Loney, Oracle 8 DBA Handbook, *The McGraw-Hill Companies*, 1998.
- [5] L. Guiri, A New Model for Role-Based Access Control, In *Proc. of 11<sup>th</sup> Annual Computer Security Application Conference*, 1995.
- [6] S. Feuerstein and B. Pribyl, Oracle PL/SQL Programming, *O'Reilly & Associates, Inc.* 1997.
- [7] R. Papaj, and B. Donald, Oracle Database on the Web, *The Coriolis Group*, 1997.
- [8] H. Toledo, Oracle Networking, *Osborne McGraw – Hill (Oracle Press)*, 1996.
- [9] M. Nyanchama and Sylvia Osborn, Access Rights Administration in Role-Based Security Systems, *Database Security VIII Status and Prospects*, 1995.
- [10] D. Antonellis, Relational Database Theory, *Benjamin Cummings*, 1993.
- [11] R. Velpuri, Oracle 8 Backup & Recovery Handbook, *Osborne McGraw-Hill (Oracle Press)*, 1997.
- [12] R. Sandhu and V. Bhanidipati, “The URA 97 Model for Role-Based User-Role Assignment, *Database Security XI Status and Prospects*, 1998.
- [13] K. Owens, Building Intelligent Database with Oracle PL/SQL, Triggers and Stored Procedures (2<sup>nd</sup> Edition), *Prentice Hall*, 1998.
- [14] C. Pfleeger, Security in Computing, *Prentice Hall*, 1996.
- [15] S. Garfinkel, Web Security & Commerce, *O'Reilly & Associates*, 1997.
- [16] Oracle, Database Security in Oracle 8i, *an Oracle Technical White Paper*, 1999.
- [17] J. H. Heimann, Security THREE-TIER Systems with Oracle 8i, *an Oracle Technical White Paper*, 1999.

- [18] S. Hofmeyr and S. Forest, Architecture for an Artificial Immune System, *Evaluational Computation*, Morgan-Kaufmann, 1999.
- [19] T. J. Teorey, Database Modeling and Design, *Morgan-Kaufmann*, 1998.
- [20] P. J. Denning, Computers Under Attack: Intruders, Worms and Viruses, *Addison-Wesley*, 1990.
- [21] H.F. Korth and A.Silberschatz, Database System Concepts, *McGraw-Hill*, 1988.
- [22] E. B. Fernandez, R. C. Summers, C. Wood, Database Security and Integrity, *Addison-Wesley*, 1981.
- [23] L.C. Dion, A Complete Protection Model, *In Proceedings of the IEEE Symposium on Security and Privacy*, 1981.
- [24] R. R. Henning, The functional Security Responsibilities of a Database Management System and an Operating System, *Database Security Status and Prospects*, 1988.
- [25] Oracle, Oracle 8: Database Administration , *an Oracle Technical Write Paper*, 1998.