Library of Congress Cataloging-in-Publication Data

Faragallah, Osama S.

pages cm

Summary: "Most database security models focus on protecting against external unauthorized users. Because multilevel secure databases provide internal security according to user access type, they are a viable option for the security needs of modern database systems. Covering key concepts in database security, this book illustrates the implementation of multilevel security for relational database models. It considers concurrency control in multilevel database security and presents encryption algorithms. It also includes simulation programs and Visual studio and Microsoft SQL Server code for the simulations covered in the text"-- Provided by publisher.

Includes bibliographical references and index.
1. Database security. I. Title.

QA76.9.D314F37 2014
005.8--dc23 2014020608
## Preface xi

## About the Authors xiii

### Chapter 1  Concepts of Database Security 1

1.1 Database Concepts 1
1.2 Relational Database Security Concepts 5
1.3 Access Control in Relational Databases 7
   1.3.1 Discretionary Access Control 7
   1.3.2 Mandatory Access Control 10
   1.3.3 Role-Based Access Control 12
1.4 Work Objectives 13
1.5 Book Organization 15

### Chapter 2  Basic Concept of Multilevel Database Security 17

2.1 Introduction 17
2.2 Multilevel Database Relations 18
2.3 Polyrinstanitation 19
   2.3.1 Invisible Polyrinstanitation 20
   2.3.2 Visible Polyrinstanitation 21
   2.3.3 Types of Polyrinstanitation 22
   2.3.4 Architectural Considerations in Supporting Polyrinstanitation 23
2.4 Multilevel Database Security Models 24
   2.4.1 SeaView Model 24
   2.4.2 Jajodia–Sandhu Model 26
   2.4.3 Smith–Winslett Model 27
Contents

2.4.4 MLR Model 28
2.4.5 Belief-Consistent Multilevel Secure Data Model 29

2.5 Performance Study 30
2.5.1 Experimental Database Structure 30
2.5.2 Impact of Varying the Number of Tuples 31
2.5.3 Impact of Varying the Number of Attributes 31
2.5.4 Impact of Varying the Number of Security Levels 32
2.5.5 Analysis of Experimental Results 32

2.6 Summary 33

Chapter 3 Implementation of MLS/DBMS Models 35

3.1 Introduction 35
3.2 SeaView Model 35
  3.2.1 Selected Operation Procedure 35
  3.2.2 Insert Operation Procedure 36
  3.2.3 Update Operation Procedure 38
  3.2.4 Delete Operation Procedure 38
3.3 Jajodia–Sandhu Model 40
  3.3.1 Select Operation Procedure 40
  3.3.2 Insert Operation Procedure 41
  3.3.3 Update Operation Procedure 42
  3.3.4 Delete Operation Procedure 43
3.4 Smith–Winslett Model 43
  3.4.1 Select Operation Procedure 43
  3.4.2 Insert Operation Procedure 44
  3.4.3 Update Operation Procedure 46
  3.4.4 Delete Operation Procedure 46
3.5 Multilevel Relational (MLR) Model 47
  3.5.1 Select Operation Procedure 47
  3.5.2 Insert Operation Procedure 48
  3.5.3 Update Operation Procedure 50
  3.5.4 Delete Operation Procedure 50
  3.5.5 Uplevel Operation Procedure 52
3.6 Belief-Consistent Multilevel Secure Relational Data Model 53
  3.6.1 Basic Procedures for Operations 53
    3.6.1.1 Xview (Label) Procedure 53
    3.6.1.2 Pl (Label) Procedure 55
    3.6.1.3 Sl (Label) Procedure 56
    3.6.1.4 Ib (Label) Procedure 57
  3.6.2 Select Operation Procedure 57
  3.6.3 Insert Operation Procedure 57
  3.6.4 Verify Operation Procedure 59
  3.6.5 Update Operation Procedure 60
  3.6.6 Delete Operation Procedure 62
3.7 Comparative Study for Multilevel Database Models  64
3.8 Summary  64

Chapter 4  Fundamentals of Information Encryption  65
4.1 Introduction  65
4.2 Basic Concepts of Cryptography  65
  4.2.1 Goals of Cryptography  65
  4.2.2 Principles of Encryption  66
4.3 Classification of Encryption Algorithms  67
  4.3.1 Classification according to Encryption Structure  67
  4.3.2 Classification according to Keys  68
  4.3.3 Classification according to Percentage of Encrypted Data  70
4.4 Cryptanalysis  70
4.5 Conventional Symmetric Block Ciphers  71
  4.5.1 Data Encryption Standard (DES)  71
  4.5.2 Double DES  72
  4.5.3 Triple DES  74
  4.5.4 International Data Encryption Algorithm (IDEA)  74
  4.5.5 Blowfish  75
  4.5.6 RC5 Algorithm  75
    4.5.6.1 RC5 Encryption Algorithm  75
    4.5.6.2 RC5 Decryption Algorithm  76
    4.5.6.3 RC5 Key Expansion  77
  4.5.7 RC6 Algorithm  78
    4.5.7.1 RC6 Encryption Algorithm  78
    4.5.7.2 RC6 Decryption Algorithm  79
  4.5.8 The Advanced Encryption Standard (AES)  81
4.6 Modes of Operation  83
  4.6.1 The ECB Mode  83
  4.6.2 The CBC Mode  85
  4.6.3 The CFB Mode  85
  4.6.4 The OFB Mode  86

Chapter 5  Encryption-Based Multilevel Model for DBMS  89
5.1 Introduction  89
5.2 The Encryption-Based Multilevel Database Model  90
5.3 Manipulation  92
  5.3.1 The INSERT Statement  92
  5.3.2 The DELETE Statement  93
  5.3.3 The SELECT Statement  94
  5.3.4 The UPDATE Statement  96
  5.3.5 The UPLEVEL Statement  97
5.4 Performance Study 100
5.4.1 Experimental Database Structure 100
5.4.2 SELECT Query 102
  5.4.2.1 Impact of Varying the Number of Tuples 103
  5.4.2.2 Impact of Varying the Number of Attributes 104
  5.4.2.3 Impact of Varying the Number of Security Levels 105
5.4.3 JOIN Query 105
  5.4.3.1 Impact of Varying the Number of Tuples 106
  5.4.3.2 Impact of Varying the Number of Attributes 107
  5.4.3.3 Impact of Varying the Number of Security Levels 107
5.4.4 UPDATE Query 108
5.5 Analysis of Experimental Results 109
5.6 Summary 111

Chapter 6  Formal Analysis for Encryption-Based Multilevel Model for DBMS 113
6.1 Introduction 113
6.2 The Encryption-Based Multilevel Model for DBMS Definition 114
  6.2.1 MLR Model Definition 114
  6.2.2 Encryption-Based Multilevel Model for DBMS Definition 115
6.3 Integrity Properties 117
  6.3.1 Entity Integrity 117
  6.3.2 Polyninstantiation Integrity 118
  6.3.3 Data-Borrow Integrity 118
  6.3.4 Foreign Key Integrity 118
  6.3.5 Referential Integrity 119
6.4 Manipulation 119
  6.4.1 The INSERT Statement 120
  6.4.2 The DELETE Statement 120
  6.4.3 The SELECT Statement 121
  6.4.4 The UPDATE Statement 122
  6.4.5 The UPLEVEL Statement 123
6.5 Soundness 124
  6.5.1 Case 1: In the INSERT Operation 125
  6.5.2 Case 2: In the DELETE Operation 125
  6.5.3 Case 3: In the UPDATE Operation 126
  6.5.4 Case 4: In the UPLEVEL Operation 126
6.6 Completeness 126
6.7 Security 128
6.8 Summary 131

Chapter 7 Concurrency Control in Multilevel Relational Databases 133
7.1 Introduction 133
7.2 Related Work 136
7.3 Enhanced Secure Multiversion Concurrency Control Model 138
7.4 Performance Evaluation 139
7.4.1 Workload Model 140
7.4.2 System Model 140
7.4.3 Experiments and Results 141
7.5 Correctness of the Enhanced Secure Multiversion Concurrency Control Model 143
7.5.1 Proof of Correctness 144
7.6 Summary 146

Chapter 8 The Instance-Based Multilevel Security Model 147
8.1 Introduction 147
8.2 The Instance-Based Multilevel Security Model (IBMSM) 150
8.2.1 Definition 1: The Property View 151
8.2.2 Definition 2: The Class View 151
8.2.3 Definition 3: The Instance View at Classification Level \( L_j \) 151
8.3 The advant address of IBMSM 152
8.4 The Select Operation Procedure of the IBMSM 152
8.5 Insert Operation Procedure of the IBMSM 154
8.6 The Update Operation Procedure of the IBMSM 154
8.7 The Delete Operation Procedure of the IBMSM 157
8.8 Comparative Study for Polyinstantiation Models 157
8.9 Summary 159

Chapter 9 The Source Code 161
9.1 Introduction 161
9.2 Screen Shots of the Prototype 161
9.3 Source Code of the Microsoft SQL Server 163
9.3.1 Source Code of the Data Security Classification Level Tables 164
9.3.2 Source Code of the User Security Classification Levels 166
9.3.3 Source Code of the Modifications to the Base Table 169
9.3.4 Source Code of the View for Each Model of the Multilevel Relational Database Models 174
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.4</td>
<td>Source Code of the Microsoft Visual Studio C#</td>
<td>185</td>
</tr>
<tr>
<td>9.4.1</td>
<td>Source Code of the Classes</td>
<td>185</td>
</tr>
<tr>
<td>9.4.2</td>
<td>Source Code of the Login Form</td>
<td>199</td>
</tr>
<tr>
<td>9.4.3</td>
<td>Source Code of the Queries Form</td>
<td>206</td>
</tr>
<tr>
<td>9.4.4</td>
<td>Source Code of the Query Form</td>
<td>232</td>
</tr>
<tr>
<td>9.4.5</td>
<td>Source Code of the Concurrency Control Form</td>
<td>259</td>
</tr>
</tbody>
</table>

References 269

Index 277
Preface

In this book we try to look at encryption-based multilevel database security through the eyes of database security researchers. Multilevel security for relational databases is an interesting information security topic. Most of the security models available for databases today protect them from outside, unauthorized users. A multilevel secure database provides internal security in relationship with the user’s type of access to the database. A multilevel secure database system has been proposed to address the increased security needs of database systems. Researchers are in need of new algorithms in this area with their software implementation.

We summarize the main contributions of this book as follows:

1. This book is devoted to the issue of multilevel security in the relational database.
2. Multilevel security for relational database models is considered in this book, with a comparison between them using different evaluation metrics.
3. Modifications are presented to an existing multilevel security model in the relational database either to speed or to enhance performance.
4. Formal analysis for data manipulation operations in multilevel security database models and mathematical proofs of soundness, completeness, and security are studied.

5. Simulation experiments are presented for validation of the discussed algorithms and modifications and also for investigating the performance of multilevel database models.

6. The C# and Microsoft SQL server source codes for most of the simulation experiments in this book are included at the end of the book.

Finally, we hope that this book will be helpful for database and information security.
1 Concepts of Database Security

1.1 Database Concepts

A database system is a computerized system whose overall purpose is to store and organize the data in a way that can be accessed, managed, and modified on demand. A database system becomes an important part of information management systems that enhances the ability of organizations to manage their important data in an easy way. A database system has many benefits that are described as follows:

- Reducing the amount of data redundancy by ensuring that the data are stored in one location and can be accessible to all authorized users
- Improving data access to users through use of host and query languages
- Enhancing data security
- Decreasing data entry, storage, and retrieval costs
- Allowing more flexibility for manipulating data
- Presenting greater data integrity and independence from applications programs

The interaction between the user, other applications, and the database itself can be performed through a software system called a database management system (DBMS) [1], which is specially designed to help the user to capture and analyze the stored data. The general purpose of the relational database management system is to be used as a tool to define, create, and manage the relational database.
Databases are classified according to their organizational approach. The most common approach is the relational database [2]. In relational databases, all data are stored in a collection of relations.

A relation contains a group of rows that have the same attributes. A row represents an object and information about that object. A relation is defined as a table, which is organized into rows (tuples) and columns (attributes). All the data in the same attribute have the same domain and stratify the same constraints. A domain presents the possible values for an attribute in the relation and the constraints make some restrictions on the domain of an attribute.

In the relational database, a relation cannot contain duplicate tuples because that would create ambiguities in retrieval.

In the relational database, to ensure uniqueness, each relation should have an attribute (or a set of attributes), called the primary key, that uniquely identifies every tuple of the relation [3]. A primary key is called a simple key if it is a single attribute and it is called a composite key if it is made up of several attributes.

In the relational database, a foreign key is an attribute (or collection of attributes) in one relation that uniquely identifies a tuple of another relation. In other words, a foreign key is an attribute or a group of attributes used to establish and enforce a link between the data in two relations.

A relational database consisting of independent and unrelated relations serves little purpose. The power of a relational database lies in the relationship that can be defined between relations. The most crucial aspect in designing a relational database is to identify the relationships among relations [4]. The types of relationship include:

- One to many: The primary key relation contains only one tuple that relates to no, one, or many tuples in the related relation.
- Many to many: Each tuple in both relations can relate to any number of tuples (or no tuples) in the other relation. Many-to-many relationships require a third relation, known as an associate or linking relation, because relational systems cannot directly accommodate the relationship.
- One to one: Both relations can have only one tuple on either side of the relationship. Each primary key value relates to only one (or no) tuple in the related relation.
In the relational database, a stored procedure is an executable code that is stored in the relational database. Stored procedures group common operations, like inserting a tuple into a relation or encapsulating complex business logic and calculations. Stored procedures are more performance than writing application code, for the following reasons:

- There is no communication between the relational database and other external applications.
- There is no need to compile and execute the stored procedure for each instance, as the stored procedure is compiled once.

In the relational database, structured query language (SQL) is the standard computer language for managing data in the relational database [5]. All relational database management systems, like Oracle, Informix, and SQL Server, use SQL as a basic database language. The SQL operation for manipulating the relation is described as follows:

- Select operation: This operation is used to get groups of tuples from the relation database. The SQL command for the select operation is described as follows:

```
SELECT [A_1, A_2, \ldots, A_n]
FROM R
WHERE P
```

where R is a relation, \( A_1, A_2, \ldots, A_n \) are the attributes from relation R, and P is the condition of the select statement that defines the tuples to be retrieved.

- Insert operation: This operation is used to insert tuples in the relation. The SQL command for the insert operation is described as follows:

```
INSERT INTO R [A_1, A_2, \ldots, A_n]
VALUES [a_1, a_2, \ldots, a_n]
```

where R is a relation, \( A_1, A_2, \ldots, A_n \) are the attributes from relation R, and \( a_1, a_2, \ldots, a_n \) are values from domains of \( A_1, A_2, \ldots, A_n \) that will be inserted.
• Update operation: This operation is used to modify tuples in the relation. The SQL command for the update operation is described as follows:

```
UPDATE R
SET [A_1=a_1,A_2=a_2, ..., A_n=a_n]
WHERE P
```

where \( R \) is a relation, \( A_1, A_2, ..., A_n \) are attributes from relation \( R \), \( a_1, a_2, ..., a_n \) are values from domains of \( A_1, A_2, ..., A_n \) that will be updated, and \( P \) is the condition of the update statement that defines tuples that are to be updated.

• Delete operation: This operation is used to delete tuples from the relation. The SQL command for the delete operation is described as follows:

```
DELETE
FROM R
WHERE P
```

where \( R \) is a relation and \( P \) is the condition of the delete statement that defines tuples that are to be deleted.

The internal mechanisms of SQL statement processing in the relational database management system (RDBMS) are presented in the following five steps:

• The RDBMS parses the SQL statement by dividing it into individual words and validating the statement syntax.
• The statement is then checked by the RDBMS against the information schema. In addition, it ensures the user privileges to execute the statement.
• The next step is the optimization of the SQL statement. The query optimization process is defined as finding the efficient way to execute the SQL statement.
• The next step is the generation of the execution plan for the SQL statement based on the optimization process performed during the previous step.
• The set of binary instructions created in the previous step is executed by the RDBMS.
1.2 Relational Database Security Concepts

In recent years, the need for securing relational databases has been increased because of increased database attacks. Most companies and organizations store their sensitive data in their own relational databases. In recent years, attackers have been able to target large relational databases that belong to large companies or large banks. In the past, relational database attacks were common, but were fewer than attacks on networks. Now, due to the increasing access of relational databases by many people, the chances of relational database attacks have increased. The reason for these attacks is to obtain money by getting sensitive information like credit card numbers or Social Security numbers. Thus, it is important to protect relational databases against these risks, and this is where database security comes into place.

Relational database security can be defined as a system that protects the confidentiality, integrity, and availability of the database [6]. Unauthorized access to a relational database indicates a loss of confidentiality, unauthorized modification to the available data indicates a loss of integrity, and lack of access to relational database services indicates a loss of availability. Loss of one or more of these basic facets will have a bad impact on the security of the relational database.

The protection of the confidentiality, integrity, and availability of the relational database will be illustrated in more detail as follows:

- **Confidentiality** can be defined as a process for preventing unauthorized access to the sensitive data that is stored in the relational database. It can be ensured by applying encryption to the data stored in the relational database. Encryption is a process in which the information is encrypted in a way that only authorized users can manage. The different levels for encryption are described as follows:
  - Data in transit means that an attacker can get access to the sensitive information by observing the network between the sender and the receiver.
  - Data at rest means that an attacker can attack the information stored in the relational database.
There are many algorithms for encryption, such as data encryption standards (DES), triple DES, and advanced encryption standards (AES).

- **Integrity** can be defined as a process for preventing unauthorized alteration to the sensitive data stored in the relational database. The integrity of data is not only whether the data is correct, but also whether it can be trusted and relied upon. Database integrity ensures the accuracy and the consistency of the data entered into the relational database.

- **Availability** can be defined as a process for preventing loss of access to relational database services. Databases must have no unplanned downtime.

In relational databases, many layers of security can be used to ensure database security [7]. The security layers can be classified into the following:

- **Authentication** can be defined as the concept of verifying the identity of a user that needs to access the relational database. Each user should identify himself before having access to data stored in the relational database system. Authentication may happen at different levels; for example, authentication can be performed by the relational database itself or allow other external methods to authenticate users.

- **Access controls (authorization)** can be defined as setting rules that define whether the user has access to the data in the relational database. Authorization rules manage the modification of data in the relational database. Access controls are procedures that are defined to manage authorizations of the data in the relational database.

- **Integrity** can be defined as a group of rules that present the correct state of the relational database during the database modification.

- **Auditing** can be defined as keeping track of all security relevant actions issued by a user.

In this book, the main focus is directed toward aspects related to access controls. An access control mechanism ensures data confidentiality. Whenever a subject tries to access a data object, the access
control mechanism checks the rights of the user against a set of authorizations, usually stated by some security administrator. Access control ensures that all direct accesses to database objects occur only according to the rules governed by protection policies. There are two different ways to enforce access control: discretionary access control and mandatory access control.

1.3 Access Control in Relational Databases

Preventing unauthorized access to the relational database is the main goal in implementing a secure database management system. Most of the database users need only a specific permission on some parts in the relational database to perform their jobs. Allowing them access to the whole database is undesirable. So, a security policy should be developed effectively to enable a group of users to access only required parts of the database. Once the security policy is developed, it should be enforced to achieve the level of security required. Three main approaches in DBMS for access control are discretionary access control, mandatory access control, and role-based access control.

1.3.1 Discretionary Access Control

Discretionary access control (DAC) is based on granting and revoking privileges for the usage of system objects (relations, views, columns, etc.) [8]. The privileges are granted to (or revoked from) every subject (user, account, program) separately. Discretionary access control policies allow access rights to be propagated from one subject to another. This is called discretionary in the sense that the owner of data has complete discretion regarding granting/revoking access privileges to his data.

In the DAC, granting/revoking privileges can be performed by the database administrator (DBA) [9]. The DBA has the following responsibilities:

- Creating accounts for users that want to log on to the relational database system
- Granting/revoking privileges for users that want to access the relational database system
• Monitoring the relational database performance
• Managing the backup and recovery procedures of the relational database

The types of DAC privileges are described as follows:

• The account privilege: Each user holds privileges that are independent of the relations in the database. For example, the DBA grants/revokes privileges to a user to CREATE TABLE, CREATE VIEW, DROP, and ALTER.
• The relation privilege: The DBA can specify the privilege to modify each individual relation in the relational database. For example, the DBA grants/revokes privileges to a user to SELECT/MODIFY/REFERENCE privilege on specific relation R. Discretionary access controls can be granted to many objects in the relational database system, such as the database, group of relations, one relation, set of the attributes of one relation, and group of tuples of one relation.

Making a discretionary access controls decision based on the content of data is called data dependent access control [10,11]. For example, some users cannot see salaries that are over than $100,000. The two approaches for implementing access controls in the relational databases are described as follows:

• View-based access control: A relation is the physical location in the relational database that stores the data in the relational database. A view is the logical set of the stored query on the data. Unlike the physical table in the relational database, a view is a logical table computed from data in the relational database dynamically when access to that view is requested.
• Query modification: A query that is written by a user is altered to include the limitation determined by the user's privileges. For example, the DBA grants user A to select only the employees that are in the material department from the relation of employees by the following grant statement:

  GRANT SELECT ON Employees TO A
  WHERE Department = 'material'
When user A needs to select all records from the employee’s relation, his query will be changed and his privileges will be added as follows:

SELECT * FROM Employees; Will be changed to:
SELECT * FROM Employees
WHERE Department = ‘material’;

In SQL, granting is performed by means of the GRANT statement, which has the following general format:

GRANT privileges
[ON relation]
TO users
[WITH GRANT OPTION]

For example:

GRANT SELECT
ON Employees
TO A

In SQL, revoking is performed by means of the REVOKE statement, which has the following general format:

REVOKE privileges
[ON relation]
FROM users

For example:

REVOKE SELECT
ON Employees
FROM A

DAC suffers from some drawbacks when applied to the relational database:

- Enforcement of the security policy: DAC depends on the concept of ownership of the data. In DAC, the user who
creates the object in the relational database is the owner of this object and can grant access to other users on this object. This has the disadvantage that the enterprise cannot manage and enforce its security requirements without including all the users that create all the objects in the relational database.

- Cascading authorization: For example, consider three users: $U_1$, $U_2$, and $U_3$. User $U_2$ has the privilege on object $O$ from $U_1$ and grants this privilege to $U_3$. Later, $U_1$ grants privilege to $U_3$ on the same object $O$, but $U_2$ revokes privilege from privilege $U_3$ for some reason. The effect of these operations is that $U_3$ still has the access privilege (from $U_1$) to access object $O$ although $U_2$ revoked privilege.
- Trojan horse attacks: A Trojan horse can be used to grant a certain privilege of a user on an object to another user without knowing any information about the user.
- Update problems: In DAC, view-based protection is a logical query that has no physical data in the relational database. The disadvantage of view-based protection is that not all data can be updated through certain views.

1.3.2 Mandatory Access Control

While DAC is concerned with ensuring the privilege to access data in the relational database, mandatory access control (MAC) is in addition ensuring the flow of data in the relational database system. MAC depends on the security level associated with each object in the relational database and each user. A security level on an object is defined as a security classification, while the security level on a user is defined as a security clearance. MAC is defined as multilevel security (MLS); because of each user and each object, one of the multiple security levels can be assigned.

A complete understanding of MLS will not happen without understanding its origins [12]. The U.S. military has a historical isolated database that contains its sensitive information. The sensitive data are classified into different security levels and must be processed on dedicated systems that do not provide access to users.
outside the intended security level. The main limitations can be described as follows:

- Redundant databases: To store data in the relational database into different security levels, a different database should be created for each security level.
- Redundant workstations: There is a need to have different workstations to get each type of datum.
- High cost of IT infrastructure: There is a risk in sharing the network resources.
- Inefficiency: Users need to get privileges on several relational database systems to perform their duties.

Multilevel security was the solution. MLS allows the data in different security classification levels to be accessed by users that have different security clearance levels.

The Bell and LaPadula model was the basic model that introduced the concept of MLS [13]. This model depends on definitions of objects and subjects. An object like relation, a tuple, or an attribute is a passive entity. A subject like user or program is an active process that needs to have a privilege on objects. Every object is assigned to a security level (classification), and every subject is assigned to a security level (clearance). Security levels are defined as labels. A label contains two components: a hierarchical component and a group of unordered categories. The hierarchical component presents the security levels of the data. For example, a company might define the security levels of its sensitive data as top secret, secret, confidential, or unclassified. The unordered categories are used to define the sensitivity of the leveled data.

Multilevel security is based on the Bell and LaPadula model and formalized by two rules. LaPadula rules are described as follows [14]:

- The simple property (no read up): A subject is allowed to read an object if the subject’s security clearance level is greater than or equal to the object’s security classification level.
- The star property (no write down): A subject is allowed to write to an object if the object’s security classification level is greater than or equal to the subject’s security clearance level.
The star property allows a lower security level subject to write data to a higher security level object. This can result in overwriting and therefore modifying of higher security level objects by lower security level subjects. Thus, MLS enforces a stronger star property to restrict each subject to write at his own security level:

- Strong star property: A subject is allowed to write to an object if the subject’s security clearance level is equal to the object’s security classification level.

1.3.3 Role-Based Access Control

The main motivation behind role-based access control (RBAC) is the necessity to simulate the structure of the natural security policies of the organization. RBAC is based on the roles that users have. Roles are similar to those of the user groups in access controls.

In RBAC, a role is defined as a group of actions and duties belonging to a specific activity [15]. The role may present a user’s job (e.g., buyer), or it may define an action that the user should do (e.g., order material). Instead of defining all the permissions to each one of the users that performs the same task, permissions on objects can be defined for roles. The user that is assigned to a role can perform all actions that the role is authorized to do. The components of RBAC can be described as follows:

- Role–permission relationships: This component manages granting/revoking permission to a specific role.
- User–role relationships: This component defines how to assign users to a specific role.
- Role–role relationships: This component defines how to make a role a member of another role.

RBAC has three security principles:

- Least privilege: RBAC allows a user to access objects with the least privilege required for the specific task that is needed to be performed. This minimizes the Trojan horses attack.
- Separation of duties: RBAC ensures that no user has enough privileges to misuse the system on his own.
- Data abstraction: This is supported by means of abstract privileges such as credit and debit for an account.
In RBAC, database administrators can manage access at a level of abstraction that is identical to the way that organizations perform their business. This is achieved by organizing users’ tasks through the implementation of roles, role hierarchies, relationships, and constraints.

In role-based access control, roles can have overlapping duties and privileges, so users assigned many roles may need to do common tasks. Some general tasks may be done by all users. In this situation, there is no need to repeat these common tasks for each role created. Role hierarchies can be performed to present the real structure of an enterprise.

RBAC has two types of role hierarchies:

1. General hierarchical RBAC is based on the concept of multiple inheritances that present the ability to obtain permission inheritance from more than one role and to inherit user membership from more than one role.
2. Limited hierarchical RBAC is limited to a single descendant. Limited role hierarchies do not support multiple inheritances.

1.4 Work Objectives

In the digital world nowadays, database security has become increasingly important since the database is the primary repository of information for organizations and governments. More and more research has been developed in database security to protect the data from possible unauthorized instructions. Most of the security models available for databases today protect them from outside and unauthorized users.

Multilevel security for relational databases provides internal security in relationship with the user’s access to the relational database. Relational database multilevel security systems have been proposed to address the increased security needs of relational database systems. Although multilevel concepts were originally developed to support confidentiality in military systems, there are now many commercial systems that use multilevel security policies.

Although many models have been developed to support multilevel security in the relational database, there are many problems in implementing multilevel security policies. These problems included complexity in designing multilevel security for the relational database and increasing the database size according to the classification level columns added to the original database to support multilevel security in the database.
This book will introduce the concept of multilevel security in the relational database, will present a comparative study for previous models that support multilevel security policies in the relational database, and will show the weakness and the strength of each model.

Also, in this book a prototype will be implemented to be used as a research tool for a performance evaluation of multilevel security for relational database (MLS/DBMS) models.

This book will give a complete view of an encryption-based multilevel security database model, which is a combination of multilevel security for the relational database and encryption system by encrypting each record with an encryption key according to its security class level. This model is characterized by three achievements:

- Utilizing an encryption system as a second security layer over the multilevel security layer for the database
- Reducing the multilevel database size
- Improving the response time of data retrieval from the multilevel database

The goal of these three achievements is to increase robustness against database attacks and enhance the performance of data manipulating operations such as select, insert, update, and delete in the multilevel database. The effectiveness of an encryption-based multilevel security database model is verified through the mathematical proof of the soundness of, completeness of, and security for multilevel security for relational database system.

The encryption-based multilevel security database model achieves good quality because it satisfies integrity properties such as entity integrity, polyinstantiation integrity, data borrowing integrity, foreign key integrity, and referential integrity of the multilevel database. Also, this book will illustrate the C# and Microsoft SQL server source codes for the implementation of the encryption-based multilevel security database model.

Concurrency control is used in relational databases to manage the concurrent execution of operations by different subjects on the same data object. This book will explain the concept of concurrency control and will define its impacts on multilevel security for relational databases. It will create a survey for studying the secure concurrency control protocols that are proposed in multilevel security for relational databases.
databases and will implement a prototype to be used to perform a series of experiments to measure the performance cost for applying the concurrency control in multilevel relational databases.

This book will also define the implementation of the data manipulation operations for the instance-based multilevel security model (IBMSM) since the IBMSM proposes two layers: the instance layer and the class layer.

1.5 Book Organization

- In Chapter 2, the basic concept of multilevel relational database security will be discussed. This chapter will explain the models that support multilevel database security and will introduce a comparative study between the multilevel database security models.
- In Chapter 3, the implementation of multilevel relational database security models will be illustrated and the performance study will be instrumented to compare the multilevel secure database (MLS/DBMS) models.
- In Chapter 4, an overview of the encryption algorithms that are applied will be presented.
- In Chapter 5, the encryption-based multilevel security database model will be described and the implementation of a working prototype to be used as a research tool for studying principles and mechanisms of the model will be explored.
- In Chapter 6, the formal model for the data manipulation operations in the encryption-based multilevel security database model will be presented and the mathematical proofs of soundness, completeness, and security will be proved.
- In Chapter 7, the concept of concurrency control in multilevel security for relational databases will be introduced.
- In Chapter 8, the implementation of the data manipulation operations for the instance-based multilevel security model (IBMSM) will be defined.
- In Chapter 9, the C# and Microsoft SQL server source codes for the implementation of multilevel relational database security models will be presented.