2011

TRADE-OFF ANALYSIS OF RELATIONAL DATABASE STORAGE FOR PRIVACY PURPOSES

Md Sadim Mahmud

TRADE-OFF ANALYSIS OF RELATIONAL DATABASE STORAGE FOR PRIVACY PURPOSES

(Spine title: Trade-off Analysis of RDBMS Storage for Privacy Purpose)

(Thesis format: Monograph)

by

Md Sadim Mahmud

Graduate Program in Computer Science

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

THE UNIVERSITY OF WESTERN ONTARIO

School of Graduate and Postdoctoral Studies

CERTIFICATE OF EXAMINATION

Supervisor:

.....

Dr. Sylvia L. Osborn

Examiners:

.....

Dr. Michael A. Bauer

.....

Dr. R. E. Mercer

.....

Dr. Miriam A. M. Capretz

The thesis by

Md Sadim Mahmud

entitled:

Trade-off Analysis of Relational Database Storage for Privacy Purposes

is accepted in partial fulfillment of the

requirements for the degree of

Master of Science

.....

Date

.....

Chair of the Thesis Examination Board

Abstract

In business organizations, person-specific data are collected as part of service requirements from customers or data providers. To maintain the privacy of these personal data from intra-organizational or external unauthorized access, an Role-based access control (RBAC) extension with privacy purposes has been introduced. Research on role-based access control and privacy has been conducted. Despite all this research, not much investigation into efficient ways to store person-specific data with privacy labels has been conducted. To the best of our knowledge, there is no such research to analyze the characteristics and impact of different storage patterns on RBAC with privacy purposes. In this thesis, we propose some storage schemes for extended RBAC with privacy purpose in a relational database environment. Moreover, we analyze the performance characteristics and impact of different SQL operations according to different storage schemes for the extension of RBAC with privacy purposes.

Keywords: RBAC, Trade-off analysis, Privacy purpose, Response time, Throughput, RDBMS.

Acknowlegements

First of all, I would like to thank almighty Allah, for His guidance and strength.

I would like to express my sincere gratitude and appreciation to my supervisor Professor Sylvia L. Osborn for her continuous inspiration, encouragement, patience, and individual feedback throughout M.Sc. study. I feel very grateful and blessed to have worked under her supervision.

All my officemates at the department of Computer Science made it a friendly place to work. All of my friends provided daily inspiration in the research and life through our interactions during the long hours in the lab.

I wish to express my appreciation to my adorable parents and my friends and relatives for their prayers, love and encouragement. At last, but not the least, I extend my sincere and deep appreciation to my beloved wife Rowja for her patience and continuous support. They have always been here with love and compassion to comfort me.

Contents

List of Figures

List of Tables

Chapter 1

Motivation and Research Overview

1.1 Motivation

Role-based access control (RBAC) is a recent access control technique that provides consistent data authorization and protection management. RBAC also provides flexibility and ease of privilege management through the concept of roles. Business organizations protect their intra-organizational and customer data or data providers' personal information using RBAC. In today's globally networked society, protecting the privacy of customers is a primary concern of the organization. To preserve data providers' personal information and prevent other users from accessing data without their consent, the notion of privacy has been added to RBAC [11]. Here the data providers' or customers' personal information is stored in a system with privacy labels. Data which are not person-specific can be handled by the conventional RBAC model. To fulfill the customers' demands and services, the service providers require some confidential information. The customers provide their information and the privacy of customers' information can be maintained by RBAC with a privacy model [11]. To understand the concept, let us use an example of an organization. Suppose it has multiple departments like administration, shipping, marketing, etc. According to the privacy policy of the company, the customer can share his/her information for direct

marketing purposes like a special offer, service update etc. On the other hand, the customer can share his/her information for third party marketing purposes as well. According to the customers' choice, personal data of the customer can be accessed by the marketing department. The data provider or customer has the authority to maintain their own privacy settings. According to the customers' choice, data items can be accessed by the specific personnel of the organization or this access can be prohibited. These data preferences are maintained by different privacy labels. In a commercial environment, confidential data can be accessed according to the privacy agreement with the customer.

Generally, the customer data with privacy labels are stored in the traditional way, in a big data table where customer information is stored with different privacy labels, in a relational database. These privacy labels are stored with each attribute of the data table. There has been a lot of research conducted on role based access control and privacy. However, there has been little attention paid to the way in which the person-specific data with privacy labels can be stored efficiently. In our research, our main focus is to analyze the extension of RBAC with privacy labels for various storage schemes.

1.2 Research Overview

1.2.1 Short Background

This work is related to several topics in the area of privacy preservation, RBAC, storage pattern of privacy data. In this section we will describe the prior research on all mentioned topics. First, we review the RBAC basic models and standards. Second, we review research which extends the original RBAC to maintain the privacy of data providers' information. Finally, we will analyze the traditional storage scheme of extended RBAC and other storage schemes as well. Based on this review, we specify our overall research approach.

In the 1990's the concept of RBAC was first formulated by Ferraiolo and Kuhn[22]. Here the researchers give a formal definition of roles as sets of permissions, role hierarchies, role activation, permission to a role, grant and revoke permissions, as well as constraints on user-role membership and role activation. Later the concepts of static and dynamic separation of duties are formally defined by Ferraiolo et al. [23] in their new RBAC prototype. Sandhu et al. [40] introduced a framework of RBAC models. They described the RBAC in four conceptual models that can be combined to provide a variety of RBAC systems. Sandhu, Ferraiolo, Khun [38] defined a unified RBAC model and proposed an RBAC standard. These models are referred to as the NIST models. These models consist of Flat RBAC, Hierarchical RBAC, Constrained RBAC and Symmetric RBAC. The proposed NIST RBAC standard was adopted by the American National Standards Institute, International Committee for Information Technology Standards (ANSI/INCITS) in 2004 [12].

To meet the various perspectives, researchers have proposed several extensions of RBAC. To prevent the unauthorized access of data providers' personal and sensitive information there are also some extensions of RBAC proposed by researchers. Privacy-aware role based access control [33](P-RBAC) and purpose-based access control [16] are two main privacy related access control models found in the literature. But the research on privacy was started with the technique of the W3C's Platform for privacy preferences (P3P). P3P specifies the formal definition of privacy policy. Moreover, P3P provides a way for a website to encode its data collection to machine readable format called a P3P policy[20]. But P3P does not provide any functionality for intra organizational privacy of the customers' personal data [16]. To provide this functionality the P-RBAC and Purpose-based access control (PBAC) models were introduced. P-RBAC is based on the classical RBAC system. Here mainly condition and obligation are added to the classical RBAC system. Purpose-based access control (PBAC) is another model which introduces the concept of purpose while dealing with privacy violations. According to [16], a purpose describes the reasons for data

Name	$Label_{Name}$	DateOfBirth	$Label_{DateOfBirth}$	CreditInfo	$Label_{CreditInfo}$	PostalCode	$Label_{PostalCode}$
Alice	{A,M,S}	1974	{A, S}	123398238999	{A, S}	N6H4P4	{A, M}
Bob	{A}	1960	{A}	987533453766	{A}	N6G4H6	{A, M}
Ron	{A, M}	1987	{A, M}	768686960766	{A}	N6G4H7	{A, M}
Jack	{A,M,S}	1985	{A, M}	768686960796	{A}	N6G4H7	{A, M}

Table 1.1: Hypothetical database with privacy labels

collection. This model is defined with two types of purposes, the Intended Purpose and Access Purpose. The access purpose is associated with a user accessing customer data and intended purpose is how the customer intends their information to be used. Intended purposes contain the data retention, obligations, conditions and purpose policies. Most recently, Al-Harbi and Osborn [11] define a new extension of RBAC with privacy. In this model the privacy purpose does not attach to the role but rather to the permission. Here the data provider can have the authority to maintain his/her privacy by attaching the applicable permission to their data. They have done their work in the Role Graph Model [35], a unique RBAC system. In this new approach, the authors show that this approach to mix privacy with RBAC is very efficient and simpler than other models like P-RBAC or PBAC. For this reason, we will mainly focus on this model of RBAC with privacy.

Generally, customer sensitive data is stored in a big data table in a relational DBMS with privacy labels. In our study we will analyze the characteristics of the system with traditional storage. Additionally, we will propose some new storage patterns with data table fragmentation [36] and do the trade-off analysis of the system with different storage patterns. To the best of our knowledge there is no single study which specifically focuses on the overall characteristics and the impact of extended RBAC with privacy purposes.

1.2.2 Problem Statement

Role-based access control simplifies the complexity of user management and administrative control in large organizations. For these reasons many researchers consider

RBAC to be suitable access control model to manage a large group of users in an organization. Different formal and extended models have been introduced at various times to overcome the data providers' information privacy and to enhance the original RBAC capabilities. To handle the data providers' personal information and preferences, extended RBAC with privacy purposes has been introduced. Generally, the service providers or different organizations provide the purpose of the information required from the customers and also mention the privacy policy as well. To provide better service according to customer requirements, the organizations now collect sensitive information like credit card number, Date of Birth, Social insurance number etc. Preserving the privacy of the data is a legal and a moral obligation of the organizations to the customers. According to the data providers' or customers' preference, the data are labelled with their privacy preferences.

In a relational DBMS, one can label rows, columns or attributes. Row-based privacy labels are associated with a data row or record in a database table. Column and attribute privacy labels are associated with columns and attributes in a database table respectively. In our study, we will analyze the attribute level privacy labels which are necessary for fine grained privacy labelling. A privacy label is composed of one or more privacy label components. There are three types of data structures that we can use to build the privacy labels e.g. sets, arrays, trees. A set is a collection of elements where the order in which these elements appear is not important. All elements are deemed equal. An array is an ordered set that can be used to represent a simple hierarchy. In an array, the order in which the elements appear is important. For example, the first element ranks higher than the second element and the second higher than the third. A tree represents a more complex hierarchy that can have multiple nodes and branches. For example, trees can be used to represent organizational charts. In our study, we will consider the set form of privacy labels. The data values are stored in the data table with privacy labels which has a set of privacy purposes. Suppose an organization has many privacy purposes like marketing, shipping, admin,

purchasing etc. In Table 1.1, we show how the customer data is stored in a data table in a relational database. Here, A denotes the administration purpose, P denotes the purchasing purpose, M for the marketing purpose, and S for shipping purposes. In Table 1.1, the credit information about Alice is only accessible for the Admin or shipping purposes, not for the marketing purpose. If the customer changes his/ her privacy settings, the privacy data labels will be changed and data can be accessed for an appropriate privacy purpose. In our research, we want to analyze different storage patterns for a relational table with privacy purposes for an RBAC system. This study is a trade-off analysis of the different storage schemes for data in a relational database.

1.2.3 Research Goals

Our overall research goals can be stated as:

1. To determine the ***characteristics*** and ***impact*** of relational operations in extended RBAC with privacy purposes using different storage patterns.

2. To determine an ***efficient way*** to store the RBAC data with privacy purposes. We consider the storage efficiency according to the response time of different relational operations.

The specific questions to determine the **characteristics** are:

1. What is the system behaviour in terms of relational operations when it is dealing with different types of storage patterns?

We want to measure the impact in terms of different system functionalities of RBAC due to different storage schemes when the data is in a relational database. So, the specific question to determine the **impact** is:

1. What is the impact in terms of different database operations on relational tables of RBAC with privacy purposes due to different storage schemes?

In the second part of our research goal we will determine an efficient way to store the RBAC data with privacy purpose labels in a relation database.

1.2.4 Research Methodology

We discuss the approach and execution plan to achieve the objective of our research. We organized this thesis into six chapters. Each one covers a significant part of our research work.

- Chapter 1: In this chapter, we discuss the overall research idea. In this chapter, we provide a short background. The main focus of this chapter is to provide the reader basic information about our work with a brief problem statement and research goals.

- Chapter 2: The main idea of this chapter is to discuss the related research work concerning data privacy and access control systems. In this chapter, we summarize the most significant related research work.

- Chapter 3: This is the main chapter in our work. In this chapter, we introduce our new ideas of different storage organizations. Moreover, we discuss the system design and implementation of our ideas in this chapter.

- Chapter 4: The purpose of this chapter is to introduce the reader to the experimental design to study the performance characteristics of our proposed prototypes.

- Chapter 5: In this chapter, we discuss the results of different experiments.

- Chapter 6: This is the last chapter of this thesis where a summary of our contributions and possible future enhancement to our research are discussed.

Chapter 2

Literature Review

Organizations are facing challenges to prevent unauthorized access of data. To mitigate this problem, at the beginning, some organizations implemented traditional access control systems, namely, Discretionary Access Control (DAC), and Mandatory Access Control (MAC). However, these access control models did not satisfy the current requirements of organizations. As a result, Role-based Access Control (RBAC) systems were introduced; they provide consistent data authorization and protection management, flexibility and ease of privilege management through the concept of roles. The RBAC model solved the limitations of the traditional access control systems. Unfortunately, it has some limitations itself which require some modification and extension of the base RBAC model.

In recent times, privacy preservation of data in an organizational environment is a primary concern. In this age of the internet, organizations' mode of operation and service management is online. To meet service agreements and service quality, these organizations collect personal data from the customers or data providers. To maintain the privacy of these personal data from intra-organizational or external unauthorized access, an RBAC extension with privacy purposes has been introduced. Generally, the customer data with privacy labels are stored in the traditional way, in a big data table where customer information stored with different privacy labels in a relational

database. These privacy labels are stored with each attribute of the data table. Here, the notion of privacy purpose added to RBAC, is the main focus of our study. This literature review report contains a discussion of the basic principles of RBAC, and research in protecting privacy of personal information.

We organize this Chapter as follows. In Section 2.1, we will introduce the concepts of access control. In Section 2.2, we will discuss about the RBAC and the RBAC family of models. Due to the importance in this study, in Section 2.3, we analyze the privacy concepts and RBAC extensions with privacy. In Section 2.4, we will focus on the different label-based data models; namely, labelling for Mandatory Access Control and Temporal database data storage schemes.

2.1 Introduction to Access Control

Protecting data means every access to data should be monitored and checked, and only authorized accesses should be allowed. This process is called access control. The purpose of access control is to enable an authority to control access to system resources and to limit the operations that a user can perform in a computer system. Access control relies on the other security components like authentication, authorization and, audit [37]. When a user attempts to perform an operation directly or indirectly, the access control system determines whether the user has the authority to perform the activity with the help of an authorization database. Access control consists of three main concepts [37]:

Security policy: defines the high level rules according to which access control must be regulated.

Security model: provides a formal representation of the access control security policies and its working.

Security mechanism: defines the low level functions that implement controls

imposed by the policy which are formally stated in the model.

Subject, object, and access mode are the main components of access control. A subject is an entity using a system which wishes to gain access to data or system resources. It can be a user, set of users, a process or a domain. An object is the entity that must be protected. It can be an operating system resource, a file, parts of a database, or even subjects (like a process or a domain). All objects are given unique names to identify them. Access mode refers to some operation on the object, e.g. read, write, execute (a program or a method), use (if the object is a printer). There are two classical models of access control, namely, Discretionary Access Control (DAC) and Mandatory Access Control (MAC).

In Discretionary Access Control (DAC), access is determined by the owner of an object. The owner of the object decides who is allowed to access it and what privileges they have on it. The access matrix proposed by Lampson [29] provides a framework for describing discretionary access control. In an access matrix, each row represents a subject and each column represents an object. Rights or access modes for each subject over an object can be specified in the matrix. The basis for DAC is called the access matrix model [29]. An example of a matrix model is illustrated in Figure 2.1.

In Mandatory Access Control (MAC), access is determined by the system rather than the owner. MAC is used in multilevel systems that process highly sensitive data, such as classified government or military information. In MAC, each subject and object is assigned an access class. These access classes are elements of a partially ordered set of classes which are arranged in lattice.

The Bell-LaPadula Model (BLM) [14], also called the multi-level model, was proposed by Bell and LaPadula for enforcing access control in government and military applications. According this model, Access classes consist of two parts: a security level and a set of categories. The security level is an element of a hierarchically ordered set, such as Top Secret (TS), Secret (S), Confidential (C), and Unclassified (U) where TS > S > C > U. The set of categories is a subset of an unordered set whose

Subjects	Objects		
	O_1	O_j	O_m
S_1	$A[s_1, o_1]$	$A[s_1, o_j]$	$A[s_1, o_m]$
.			
S_i	$A[s_i, o_1]$	$A[s_i, o_j]$	$A[s_i, o_m]$
.			
S_n	$A[s_n, o_1]$	$A[s_n, o_j]$	$A[s_n, o_m]$

Figure 2.1: Access matrix model

elements are functional, competence, areas (e.g., NATO, Nuclear, and Army for military systems). The dominance relation between access classes is defined as follows. An access class c_1 dominates ($\geq$) access class c_2 if security level of c_1 is greater than or equal to security level of c_2, and categories of c_1 include categories of c_2.

2.2 Role-Based Access Control

The origins of Role-Based Access Control (RBAC) [22] [30] are related to the use of roles and groups in UNIX and other operating systems and database management systems [13], and separation of duty concepts [19]. The idea of using roles in access control is not new. In the 1970s to 1980s roles are used in different security administration tools such as Computer Associates' CA-ACF2, IBM's RACF and CATOP SECRET. In the 1990s, the concept of RBAC was first formulated by Ferraiolo and Kuhn [22] as an alternative access control approach to DAC and MAC. DAC and MAC are not suitable for different commercial purposes, because DAC is too weak for effective control of information assets, while at the same time, MAC is too strict and restrictive. In [22], the researchers provide a formal definition of roles as sets of

permissions, role hierarchies, role activation, permission to a role, grant and revoke permissions, constraints on user-role membership and role activation. In RBAC, permissions are granted to roles, and users are granted membership in roles based on their tasks or job responsibilities. This type of access control management is at a level that corresponds to the organization's structure and can help enforce organizational policies.

Over time, RBAC came in different models. Toward developing the original RBAC concept, many RBAC models were introduced. In this section, RBAC models are described in three categories. We discuss Sandhu et al. models [40] which are considered as the first framework RBAC models. Second, the NIST RBAC proposed models [38]. Finally, the RBAC model was adopted by ANSI INCITS in 2004 [12]. Moreover, at the end of this section we will discuss the Role Graph Model [35], one unique variant of RBAC.

2.2.1 Sandhu's RBAC Models

Sandhu et al. [40] introduced a framework of RBAC models. $RBAC_0$ has minimum features, supporting the basic requirements of RBAC. This model consists of the three components: users, roles and permissions. A user is a human being, a role is a set of permissions and a permission is a specific access to one or more system objects or resources. Here, both user assignment and permission assignment relationship are many-to-many. The $RBAC_1$ model introduces role hierarchies as an additional feature. Role hierarchies allow permission inheritance among the roles. In a large and complex system, the use of role hierarchies is inevitable. Usually, role hierarchies are implemented in a tree type data structure where the senior roles are top of the tree and junior roles in the bottom. $RBAC_2$ includes $RBAC_0$ with the additional feature of constraints, which impose restrictions on the configuration of the components of RBAC. This model introduced constraints which are vital aspects and one of the principle motivations of RBAC. To understand the concept we will consider

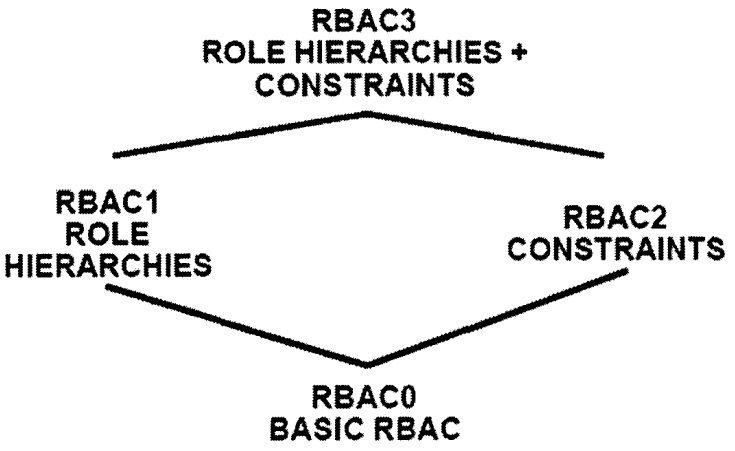


Figure 2.2: RBAC96 framework[40]

one example of mutually exclusive roles, such as purchasing manager and accounts payable manager. In most organizations, the same employee or individual will not be permitted to be a member of both roles because it creates a possibility for committing fraud. This well-known and time-honored principle is called separation of duties [40]. $RBAC_3$ is the combination of $RBAC_1$ and $RBAC_2$ to provide both constraints and role hierarchies. Figure 2.2, shows the complete set of Sandhu RBAC 96 models.

2.2.2 NISTs RBAC Models

Sandhu, Ferraiolo, Khun [38] defined a unified RBAC model and proposed a RBAC standard. These models are referred as the NIST models. These models consist of Flat RBAC, Hierarchical RBAC, Constrained RBAC and Symmetric RBAC. The first level, Flat RBAC, defines features that are a minimal requirements for all RBAC systems like user-role review functionality (mandatory), permission-role review functionality (not mandatory) and a concept of session which enables roles to be activated and deactivated for a user session. The second level, hierarchical RBAC adds requirements for role hierarchies. The third level, Constrained RBAC, adds the requirement for enforcement of separation of duties (SoD) grouped into two types, static and dy-

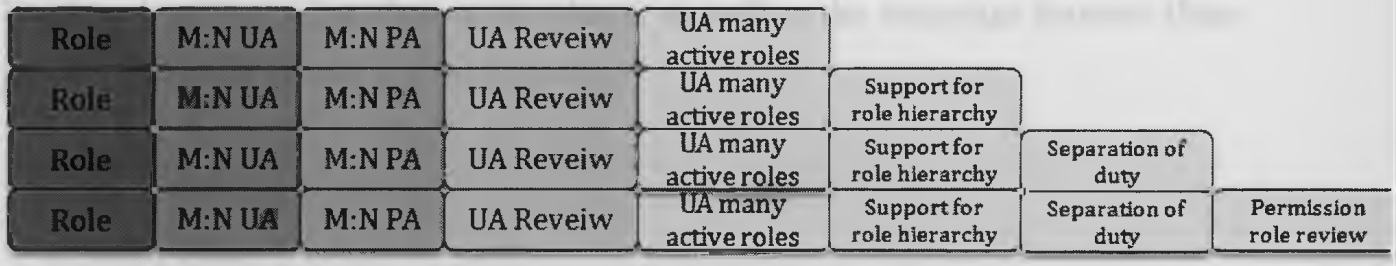


Figure 2.3: Level of NIST models

namic. The fourth level, symmetric RBAC, extends these requirements to include an interface for permission-role review with respect to a defined user or role. Different levels of NIST models are illustrated in Figure 2.3.

2.2.3 ANSI INCITS 359-2004 RBAC Standard

The proposed NIST RBAC standard was adopted by the American National Standards Institute and the International Committee for Information Technology Standards (ANSI/INCITS) in 2004 [12]. This standard organized RBAC into two parts, the RBAC reference model and RBAC functional specification. The RBAC reference model provides the formal definition of RBAC sets and relations, standards for general terms, requirements and features. The Functional specification defines functions of administrative operations, administrative queries and system functions for managing RBAC attributes on user sessions and making access control decisions. The RBAC model and functional specification are organized into four components: Core RBAC, Hierarchical RBAC, Static Separation of Duties (SSD) and Dynamic Separation of Duties (DSD).

2.2.4 Role Graph Model

In 1994, Nyanchama and Osborn introduced the Role Graph Model [34] based on graph theory. The RGM uses graphs to present the hierarchical structures. The

RGM is described using three main planes, as well as the mappings between these planes. The three planes are the following:

1. Groups Plane
2. Roles Plane
3. Privileges Plane

In the RGM, a group is defined as a set of users with a name [35]. The User/Group graph contains users and groups as vertices and the edges define the *isSubgroup* relationship in which the member user set in the lower group is a subset of the member users set of the super group. The relationship between a lower group and a higher group is set containment.

The role graph is an acyclic, directed graph in which the nodes represent the roles in a system, and the edges represent the *is-junior* relationship [31]. Every role graph has a MaxRole. MaxRole represents the union of all the privileges of the roles in the role graph; it does not need to have any users authorized to it. Each node in a role graph presents a role, r, which consists of a name, r_{name} and set of privileges, r_{pest}. The edges show the *is_junior* relationship: when $r_1.r_{pset} \subset r_2.r_{pset}$, r_1 is junior to r_2, denoted by $r_1 < r_2$. The role graph has following properties:

1. It has a MaxRole which contains in its rpset the union of all the privillages of the graph. MaxRole does not need a user assigned to it.
2. The role graph is acyclic.
3. For any two roles, r_i and r_j, in the graph, if $r_i.r_{pset} \subset r_j.r_{pset}$ then there must be path from r_i to r_j.

Each privilege is a pair (o,m), where o is an object and m is an access mode of the object, o. The privilege hierarchy models the implications among privileges. These implications can vary based on the application domain. When a privilege is assigned

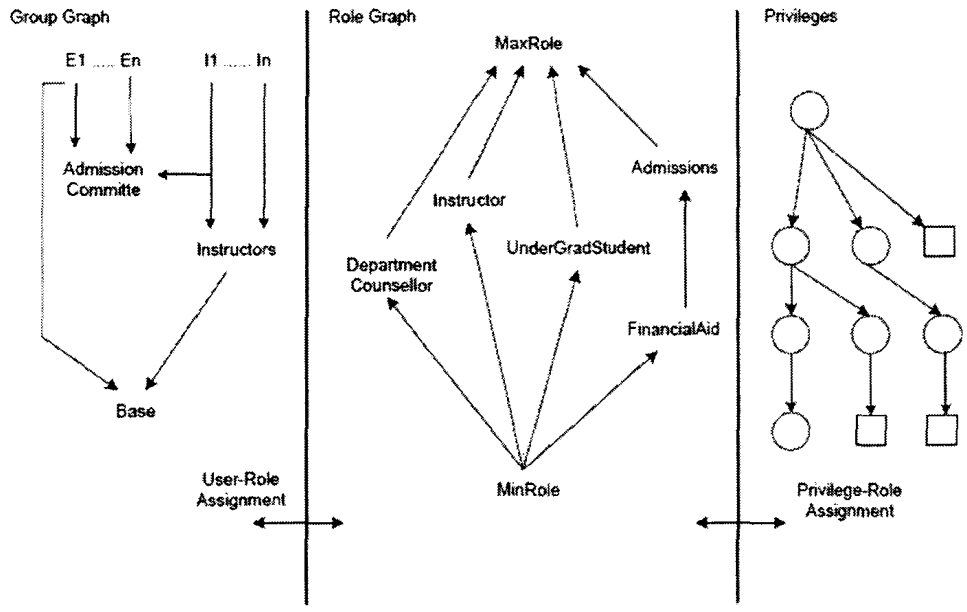


Figure 2.4: The Role Graph Model [35]

to a role, it requires that the implied privilege must also be granted. Figure 2.4, shows the components of the role graph model.

A number of algorithms have been developed to administer and edit the RGM. For all the algorithms, when any error is detected, the operation is rejected and the graph remains unchanged. Here, the direct and effective privileges for all the roles are according to the definitions and there are no redundant edges. There are algorithms for:

- Role addition
- Role deletion
- Privilege addition
- Privilege deletion
- Edge addition
- Edge deletion

All the algorithms are polynomial time algorithms in terms of the number of the edges and nodes in the role graph.

There are some fundamental differences between the ANSI RBAC Model and the Role Graph Model. The idea of user sessions is one of the core components of the ANSI model but the RGM does not have sessions. In the RGM, duplicate roles are forbidden, but the ANSI model allows the duplication of roles. The ANSI model does not support the idea of groups. The role graph model treats a group as a set of users who have similar functionality and authorizations. Actually, in ANSI RBAC, a group can inherit from a role but not add permissions. A lot more operations on the RGM are implemented. Role and edge insertions and deletions are allowed in an easy way by algorithms which are provided by the RGM.

2.3 Extended RBAC with Privacy

Researchers have introduced many extended RBAC models for different purposes. For instance, Klarl et al. [27] introduce RBAC for business usage, Tahir and Muhammad [43] introduced the contextual role-based access control model, Wainer et al. [45] introduced W-RBAC model for workflow systems, Bertino and Piero [15] introduced Temporal Role-Based Access Control Model (T-RBAC) etc.

As data privacy is an important aspect of the modern world, the extension of RBAC with data privacy has also been studied. The main focus of privacy in RBAC is to control unauthorized access to private data concerning individuals. To prevent the unauthorized access of data providers' personal and sensitive information, there are also extensions of RBAC proposed by researchers. In this section, we will discuss these extensions of RBAC.

The research on privacy was started with the technique of the W3C's Platform for privacy preferences (P3P). P3P specifies the formal definition of privacy policies. So we will start the discussion with P3P.

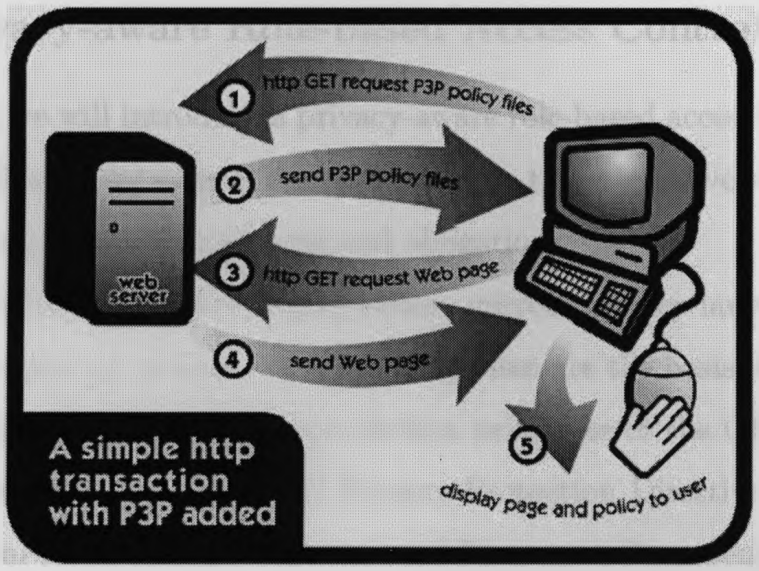


Figure 2.5: Obtaining Privacy Policy Using P3P (Adopted from [4])

2.3.1 Platform for Privacy Preferences (P3P)

The Platform for Privacy Preferences (P3P) is a standard developed by the World Wide Web Consortium (W3C) to maintain customers' or data providers' personal information privacy. This system provides a simple and automated way to control the user personal information while visiting web sites [8]. P3P defines what, where and how the private data should be protected. The first architectural overview was published at the end of 1997. Then the P3P 1.0 working draft was published in 1998. In January 2005, P3P 1.1 working draft was published by W3C [4]. P3P policies provide a detailed report on how a site collects the end user data, handles and uses the data for different purposes.

Even though, P3P provides a standard, machine-readable, unique technique to maintain the privacy preference, it does not provide a mechanism for ensuring the intra-organizational privacy of the customers' personal data [16].

2.3.2 Privacy-aware Role-based Access Control (P-RBAC)

In this section, we will introduce a privacy-aware role-based access control system(P-RBAC) [33]. This model was constructed based on the original version of RBAC with the extension of purposes, conditions and obligations.

OECD guidelines [3] of privacy protection, current privacy laws of different countries and public privacy policies of some enterprises are the basis of purposes, conditions and obligations. The privacy protection principles of the OECD are based on many data-protection laws (e.g. EU Privacy Protection Directive, the Privacy Act and the Personal Information Protection and Electronic Documents Act in Canada) and different public privacy policies. This is the most well known privacy information policy.

Purpose is the intention to perform an activity, in other words it is the justification of the performed task. Purpose is widely used to define and specify privacy. In most of the privacy documentations and studies, purpose is identified as an important component. HIPPA [6] provides a clear definition of purpose for privacy preservation. Condition specifies under which situation one action is allowed for objects in the system. At a specific time, conditions apply to one or more subjects to limit the subjects to perform a specific action to one or more objects. Conditions formulate privacy policies via restricting the actions of subjects. To an enterprise, which might be an online business and financial service, fine grained privacy protection of the end user or customer data is very important. The usual process of protecting privacy is based on conditions and obligations. Generally, enterprises provide privacy notification which includes information collection, usage and retention policies.

2.3.2.1 P-RBAC Models

Dynamic privacy policies and diverse privacy laws increase the complexity to implement and maintain one generic model which satisfies the all the requirements of privacy protection. P-RBAC was designed as a family of conceptual models similar

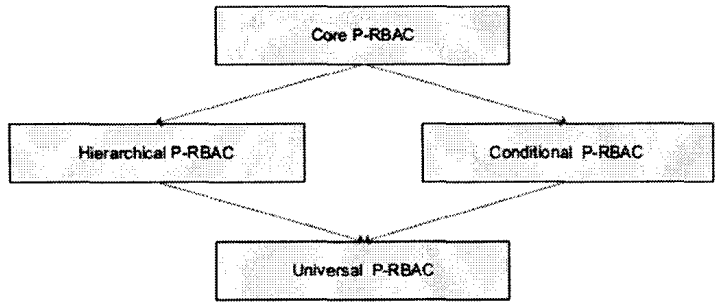


Figure 2.6: The Family of Conceptual P-RBAC Models (Adopted from [33])

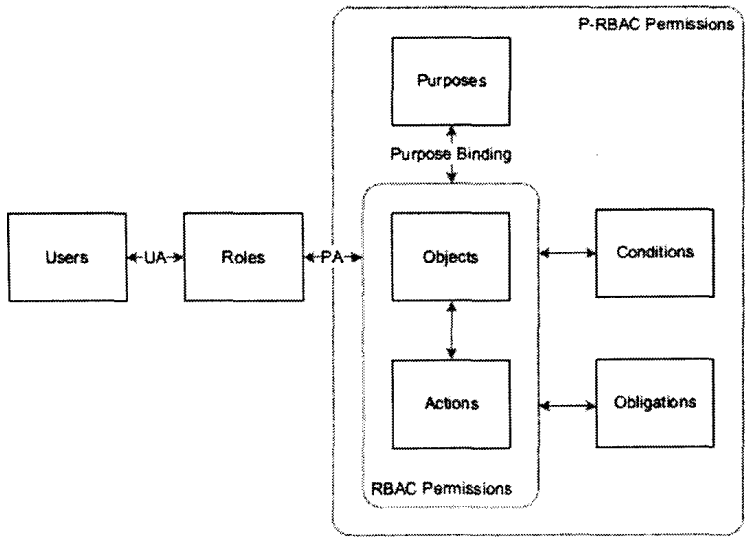


Figure 2.7: Core P-RBAC Model (Adopted from [33])

to classical RBAC. Figure 2.6, shows the relations among the family of privacy aware RBAC models.

Core Privacy-aware RBAC

The core P-RBAC model is the base model, which is illustrated in Figure 2.7. The model includes several components : Users (U), Roles (R), Objects (O), Actions (A), Purposes (Pu), Obligations (Ob) and conditions (C) expressed using a customized language, referred to as LC_0. The components of this system are:

Users (U): In this model users are defined as human beings.

Roles (R): Roles can be defined as the organizational authority or responsibility of a user, more specifically, job functionality of an employee in an organizational environment.

Objects (O): Object of this model is the system resources or information concerning an individual.

Actions (A): Actions are executable programs, which perform some functionality for the user upon invocation.

In Core P-RBAC, permissions are assigned to roles, and the user can obtain these permissions by being assigned to roles. This is a very complex model which guarantees the assignment via purpose declaration, condition and obligation. Core P-RBAC is formulated without the session property of the classical RBAC.

Hierarchical Privacy Aware RBAC

The Core P-RBAC model is extended by the concept of hierarchies. Hierarchical P-RBAC introduces the notion of Role Hierarchy (RH), Object Hierarchy (OH), and Purpose Hierarchy (PH).

RH is the same notation and semantics as the hierarchical RBAC. PH describes a partial order among different purposes. PH uses a tree data structure, where each purpose has only one parent. The top purposes are more general than the bottom one. OH describes a partial order relation ($\leq$) between different objects and each object has at most one parent.

Conditional P-RBAC

Conditional P-RBAC [32] enables the P-RBAC model to express more complex conditions than the core P-RBAC which is supported by the conditional language of core P-RBAC. Moreover, it defines the relations between the permissions assigned to different users. It introduces Permission Assignment Sets and Complex Boolean Expressions [32]. To express the rich conditions, Conditional P-RBAC uses a new type

of context variable, which is a combination of string, integer and common logical operators.

Universal P-RBAC

Universal P-RBAC combines the features of both Conditional P-RBAC and Hierarchical P-RBAC model. It has also introduced three important features such as negative permissions, flow control for obligation execution, and permission combination principles [32]. Negative permissions are not supported by any RBAC model. However, according to the authors, they are useful or even necessary in some situations.

2.3.3 Purpose Based Access Control

To maintain this privacy of data providers' or customers' personal data, a purpose-based access control system is introduced. This model was formulated by Byun and Li [16]. According to them, a purpose describes the reason(s) for data collection and data access: access purpose is the intention for accessing data objects and intended purpose is the specified usages for which the data objects are collected. The PBAC model is formulated with the basic concept of purpose, namely, Intended Purposes and Access Purposes.

According to [16], we can define intended purpose as how the customer intends their information to be used and the access purpose is associated with a user accessing customer data. Generally, the intended purpose summarizes the privacy policies for person-specific data and specifies that for which purposes the data can be accessed. When data items need to be accessed, the intended purpose is checked against the access purpose. In this model, four components are defined which should specify in a privacy policy such as purpose(s), retention, obligation(s), condition(s). In PBAC, intended purposes support both permissive and prohibitive privacy policies. Based on this concern, intended purpose is divided into two parts: allowed intended purposes (AIP) and prohibited intended purposes (PIP). Here, AIP specifies each data element

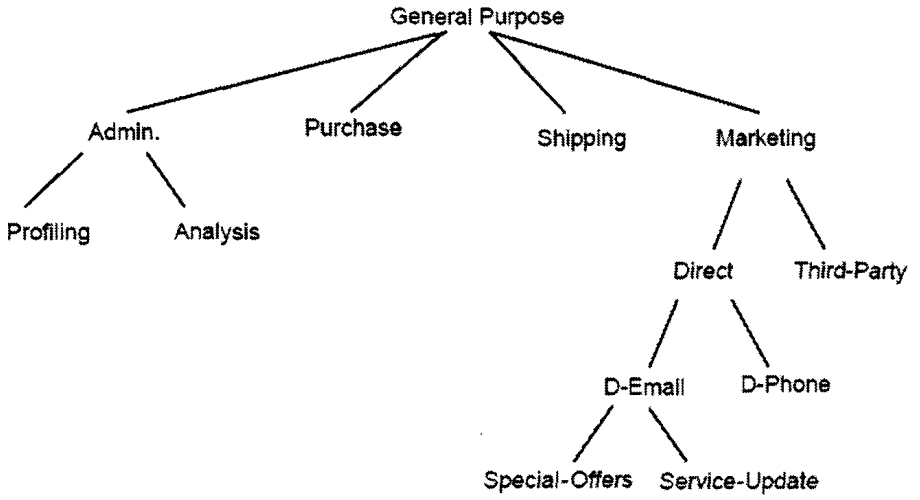


Figure 2.8: Example Purpose Tree (Adopted from [16])

and accesses allowed by the data providers and PIP specifies the data access which is not agreed by the data providers. This structure specifies the model in a flexible and compact manner. Moreover, this structure provides assurance of data providers' privacy preservation. In this design of the PBAC model, when PIP and AIP conflict, PIP always overrides AIP.

In the PBAC model, to specify the purpose, a tree data structure is used. In Figure 2.8, all the purposes are arranged in a tree. An upper purpose of this tree is more general than a lower purpose. Here, each data element has one or more intended purposes. The purposes are depending on the customer pattern and customers' privacy settings which have been specified by the customer during many phases of service agreement with a specific service provider. An access purpose is the reason for accessing a specific data item. If data access requests are issued in the system, access purposes decide whether the data can be accessed by the user or not.

Now, we will describe some alternative methods of determining the access purpose. First, the user(s) provides the access purpose(s) when they are requesting a data access. This is a very simple and effective method from every point view, but it completely relies on the system user. In some cases, this not an effective way.

Second, each application or stored procedure in the system must be registered with one or more access purposes. In this way the user wants to access the data with these applications, the system determines the associated access purpose and acts accordingly. The problem in this approach is that it is not possible for many dynamic and complicated applications. Third, the system determines the access purpose(s) dynamically according to the current context. Here, the current context is determined by analyzing several factors, namely, the job function, role, the nature of data to be accessed, the application identification and the time of the request. Suppose, in an enterprise, an employee of the shipping department wants to get the address of a customer in regular business hour via using a specific application. The system analyzes the job function, time of request, data which the employee wants to access, then the system allows the access according to this request. If the access request is not matched with the conditions, then system will not allow any access to the system. The main challenge in this approach is to determine the access purpose accurately and efficiently.

2.3.4 RBAC with Privacy

To protect the person-specific data, most recently, Al-Harbi and Osborn [11] define a new extension of RBAC with privacy. In this extension, the authors use the Role Graph Model due to the flexibility of this RBAC model. As mentioned in Section 2.2.4, there are three main entities in the Role Graph model, namely, Users/Groups, Roles and Privileges. From Section 2.2.4, we have already discussed the basic functionality of the Role Graph Model. In this privacy model, the main focus was the privileges plane. The Group plane and the Privileges plane are linked with the role plane via the user-role assignment mapping and the role-privilege assignment mapping. This means that each role is associated with a certain number of users and privileges. Therefore, in this system, every user in a role has the specific privilege(s) for the specific object(s). Privacy purposes are added to give privacy protection for

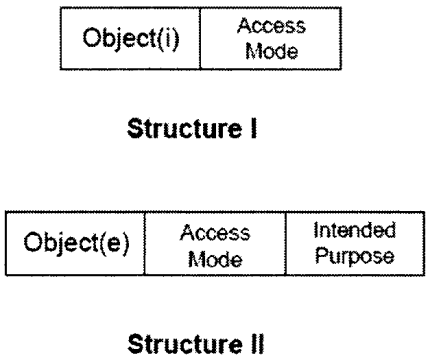


Figure 2.9: The Privilege Component in the RGMP (Adopted from [11])

data providers' personal data. The privacy purpose does not attach to the role but rather to the privileges(s). Here, the data provider can have the authority to maintain his/her privacy by attaching the applicable (intended) purpose to their data. There is some reason to attach the privacy purpose(s) to privilege(s). If a privacy purpose is attached to a role, then all users or groups in that role get access to that specific object of the system. In this approach, the problem is objects may be accessed for purposes which are not intended to be used for accessing them. In this situation, every member of this role can access any object for every mentioned purpose. Suppose, a role R can access some system objects like O1, O2 and O3. Here, let say O1 can be accessed for purpose P1, O2 can be accessed for purpose P2 and O3 can be accessed for purpose P3. If P1, P2 and P3 are purposes which are available to all users who are assigned to the R role, any user can use any one of the purposes in this role to access any of the objects that this role is allowed to access (O1, O2 or O3). If role to privilege is considered as a one to one relation like one role, has only one privilege purpose, then also the aforementioned problem arises. In this case, senior roles can inherit purposes from their junior roles which create the problem. Here, senior roles can have more than one purpose, and a user who is assigned to such a role can use any of the available purposes to access the desired objects.

In the extended RGM with privacy approach, the objects which represent the

data and system resources are divided into two categories: (1) Internal objects and (2) External objects. External objects can be the data which are provided by the data providers or customers, like, customer address, credit card information, home phone number, date of birth etc. Internal objects can be the information or resources which are provided or implemented by the users of the system. In this approach, intended purpose (IP) is only associated with external objects and to get access to the external objects, an access purpose needs to be provided by the system user as well. The privilege component of internal objects will be the same as RMG. Figure 2.9, illustrates the privilege components of both object formats.

In this model, two structures for a privilege are introduced. First, Structure I is for internal objects and Structure II is for external objects. Structure I remains same as RMG, no change applied. Structure II in the privilege component in the new approach consists of three attributes instead of two.

In RGMP, two types of purposes are used: Intended Purpose and Access Purpose. The intended purpose is the purpose associated with external data stored in the system and the access purpose is the reason that a user must state when (s)he requests to access certain data in the system. Here, the access purpose determination is the same process as the PBAC in Section 2.3.3.

Figure 2.10, shows the new Role Graph Model after adding the privacy components. Even though, some parts have been added to the RGM, the mechanism the RGM uses to work will remain the same. Therefore, the new design will keep the old design's properties that have been mentioned in Section 2.2.4. As mentioned earlier, the old structure of a privilege is presented by a pair of (x,m), where x refers to an object and m refers to an access mode. As structure II of the privilege is proposed here in order to let the role graph model be compliant with privacy policies and data providers' preferences. The new structure of the privilege will be the following:

(x,m,ip) where

x denotes an object

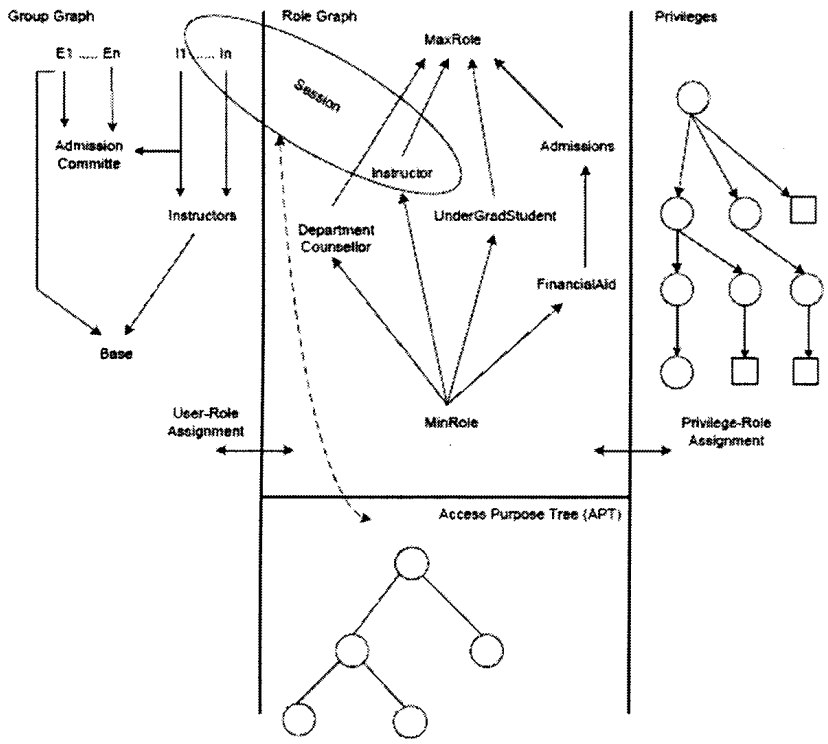


Figure 2.10: The Role Graph Model and Privacy (RGMP) (Adopted from[10])

m denotes an access mode

ip denotes data providers' intended purpose

2.4 Data Modeling of MAC and Temporal DBMS

In this section, we will analyze the data model of MAC and Temporal DBMS. In our study, we will focus on the privacy label for RBAC and the efficient storage scheme for extended RBAC with Privacy labels. Here, we will focus on attribute-based labelling. In MAC and Temporal DBMS, data storage policies are based on the concept of attribute-based labels. In MAC, different data are stored with different levels of security labels. In the case of temporal database systems, time based labelling is introduced, which also a attribute based labelling. In the following subsections we will analyze the data model for MAC and Temporal DBMS. The MAC part of these

discussions is based on [39].

2.4.1 Data Modeling in MAC

Mandatory access control system was used for classified government and military projects. Here, subjects can read/write the object of the system according to their clearance. MAC operations are based on some security labels assigned to both subjects and objects. Using these security labels, the system provides a security clearance level for the subjects and a sensitivity level for the object which stores the information. According to MAC, subjects can read objects at the same or lower classification, and write objects at the same or higher classification, than that of the subject. According to the Bell-LaPadula Model, one classic MAC model, the security levels are divided into two parts: a classification and a set of categories. The classifications typically are Top Secret, Secret, Confidential and Unclassified. These classifications are totally ordered, TS > S > C > U. On the other hand, the categories are not ordered, for example, {p1, p2, p3}. These two pieces of information are combined to make the security level such as (Top Secret, {p2}), (Unclassified, {p1, p3}). As per this model, a security level L1 = (C1, S1) is higher than or equal to (or dominates) level L2 = (C2, S2) if and only if C1 $\geq$ C2 in the total ordering of classifications and S1$\supseteq$S2. In MAC, there are several properties, for reading, a subject s could only read an object o if L(s) >= L(o). For writing, the Liberal *-property defines a subject s can write an object o only if L(s) <= L(o) and with the strict *-property, s can write o only if L(s) = L(o).

The SeaView model combines MAC and a trusted computing base (TCB). The functionality of this model is based on MAC. In this model, data in relational tables is defined as R(A_1, C_1, ..., A_n, C_n, TC), where C_i is the classification attribute for (database) attribute A_i and TC is the access class of the whole tuple. In a MAC system, all subjects and objects must have labels assigned to them. A subject's sensitivity label specifies its level of trust. An object's sensitivity label specifies the

Name	C_{Name}	Dept	C_{Dept}	Salary	C_{Salary}	TC
Bob	U	Dept1	U	10K	S	S
Ann	S	Dept1	S	30K	TS	TS
Sam	S	Dept2	S	20K	S	S
Sam	TS	Dept2	TS	30K	TS	TS

Table 2.1: An example of a multilevel relation [18]

Name	C_{Name}	Dept	C_{Dept}	Salary	C_{Salary}	TC
Bob	U	Dept1	U	10K	S	S
Sam	S	Dept2	S	20K	S	S

Table 2.2: S-instance of a multilevel relation [18]

level of trust required for access. In order to access a given object, the subject must have a sensitivity level equal to or higher than the requested object.

Now we discuss about the data storage of MAC. This discussion is based on [18]. The SeaView model extends the concept of relation to classification labels to deal with the multilevel data. The multilevel data model uses element level data modeling. Multiple instances of one object bearing the same name coexist and are differentiated by their access label. The data storage approach is completely different than other access control policies. The database is decomposed into multiple single-level databases. The idea of fragmentation of the data table is used. For simplicity, the multilevel relations only consider the security levels (i.e., U, C, S or TS). Table 2.1

Name	C_{Name}	Dept	C_{Dept}	Salary	C_{Salary}	TC
Ann	S	Dept1	S	30K	TS	TS
Sam	TS	Dept2	TS	30K	TS	TS

Table 2.3: TS-instance of a multilevel relation [18]

shows an example of a multilevel relation. The multilevel relation is decomposed into different single-level relations according to security levels. Tables 2.2 and 2.3 show the different instance of the multilevel relation according to the access class. This approach provides a convenient way to store and handle different multilevel complex data.

2.4.2 Data Modeling in Temporal database

The first step to building the data model is analyzing the part of the real world which is related to the specific domain. Temporal database management systems deal with time-related data [28]. Time is considered to be a linear concept where each point is called an instant and the time between the instants is called a time period. Here, the length of this time period is called an interval. These three components in a temporal database are known as the temporal data type [42]. The time model of a temporal database system is categorized into three parts, namely, valid-state time, transaction-state time and bitemporal-state time. Bitemporal data is a concept used in a temporal database. It denotes both the valid time and transaction time of the data.The transaction time of an object is the time when the object is stored in the database [44], i.e., the time that it is present in the database. The valid time of a database object is the time when the object is effective or holds (is true) in reality, i.e., the time when the event occurred, took place in reality.

Many different time data models or schemes have been introduced by many researchers [25] [41] [21]. These data modeling concepts are mainly focused on a real world time representation of different data sets[44]. These models are also concerned with the semantics of time representation, how to apply the time data in a database infrastructure like tuples or attributes. How to arrange the time data with different times attributes in single or multiple tuples is also a challenge in time data modeling. In time data modeling, the pattern for a time, continuous or discrete, needs to be determined as well. In general, there are two main approaches for data modeling

of temporal DBMS: attribute timestamp[26] and tuple timestamp [9]. A database object is stored in a database at some point in time. The time model is finite, which means that there is a first and a last time point and all timestamp values associated with tuples in the database fall within this range.

Tuple time stamping is usually applied in temporal relational data models supporting only First normal form (1NF) relations. Data models applying tuple time stamping add timestamps to each tuple in a relation. Time stamping is usually achieved using the extension approach, where special time attributes are added to a non-temporal schema. In relational databases, tuples can be defined as a set of variable or data in general meaning. So in the real world, if the variation in data concerns all elements in a set of data, then validity information of data set is stored in a tuple while considering relational database environment. For example, we use tuple time stamping to express that Bob has worked in the department of CS since 1988, having an employment number e213. In 1988, he earned 20000, and got a salary raise in 1990 to 25000. This information is stored in two tuples, one valid from 1988 to 1990, the other one valid since 1990. The term until changed denotes that it is not yet known when the validity time period of the tuple finishes.

< e213, Bob, 20000, CS > from 1988 until 1990

< e213, Bob, 25000, CS > from 1990 until changed

The advantage of the tuple time stamp modeling is simple that it is easier to apply than the other model. In addition, when the system does not cause data redundancy, a tuple timestamp is more effective. In tuple time stamping, information about a real world entity is spread over several tuples where each tuple represents a state the real world entity was in during a certain time period, called vertical temporal anomaly [24]. If, for example, a tuple containing data on an employee is modified by changing

the salary, all other information in the tuple, such as the name, the employment number and the department the employee is working in, has to be repeated.

Attribute timestamping overcomes the disadvantage of data redundancy introduced when applying tuple timestamping. Attribute time stamping adds timestamps to each attribute value. Values in a tuple which are not accepted by a modification do not have to be repeated. Here, the data value and the time data are stored in a tuple with a triple value. The value is a form of <[l, u), v] in which l represents the lower time bound, u represent the upper time bound and v represent the value of attribute. The form of this temporal data shows the temporal values with closed and opened time attributes.Using the schema extension approach, attribute time stamping demands that the underlying data model supports Non First Normal Form (NFNF) relations or complex objects, since all timestamp attributes are of a complex type, storing the attribute value together with its timestamp. Using attribute time stamping, we can store the information given in previous example in a single tuple:

< {|e213 from 1988 until changed|},
{|Bob from 1988 until changed|},
{|20000 from 1988 until 1990|, |25000 from 1990 until changed|},
{|CS from 1994 until changed|} >

The tuple contains the history of each attribute. An attribute history is a set of value-timestamp pairs.

2.5 Summary

In this chapter, we reviewed the concept and important issues of role- based access control and privacy preservation in a commercial environment. We reviewed how RBAC is used in a commercial environment as an access control system as well,

we reviewed different RBAC models, their components and important aspects and issues. We discussed the standardization of RBAC systems. In the second part of this report, we explained how RBAC with privacy extension works. Different techniques of RBAC privacy extension were analyzed and surveyed. We also discussed the different data modeling processes for label based data storage pattern, more precisely, MAC and Temporal DBMS. After analyzing these thoroughly, we find that there is little research on RBAC with privacy extensions and there are some potential areas which can be analyzed further.

Chapter 3

Methodology

In Chapter 3, our main focus is to discuss different storage patterns for relational tables for Extended RBAC with privacy purposes. Here, we will analyze different kinds of storage schemes, implement the patterns using an example system and determine its characteristics. Before we start our discussion, we will provide an overview. This chapter proceeds as follows: First, we will discuss the basic approach to store the data with privacy purpose, the functionalities of the basic approach and describe the basic approach with a sample system. Second, we will introduce some new approaches to store the privacy purpose data, functionalities and then explain these approaches with examples. Third, we will discuss the implementations of our proposed approaches to store the privacy purpose data with extended RBAC. Finally, we will discuss the basic SQL functionalities of the proposed approaches.

3.1 Basic Storage Design Approach

Enterprises regularly collect customers' personal information along with other attributes during activities such as marketing, sales, or billing. Generally, the customers' data are stored in a table with privacy labels. Each table is physically represented as a collection of customers' information with privacy preferences in an

RDBMS. In a relational table, the information and privacy labels are stored as separate attributes. Each attribute is stored with a set of privacy purposes which defines the accessibility of that information according to the information in the purpose set. Physically the information and privacy purposes are stored in different columns.

Before discussing this mechanism in more detail, we will discuss the different data structures for privacy purposes. Usually, privacy purpose labels are represted by three different data structures: Sets, Arrays sequence and Trees. A set is a collection of elements where the order, in which these elements appear, is not important. For example, if there are multiple purposes like Marketing (M), Sales (S), and Administration (A), the privacy purpose labels would be in a form of {M, S, A} or {M, A} etc., where the order of the elements in the set is not important. On the other hand in an array, the order in which the elements appear is important because it is an ordered set. For example, the first element ranks higher than the second element and the second higher than the third. A tree structure represents a more complex hierarchy that can have multiple nodes and branches. In our thesis, we will consider only the set data structure for privacy labels.

As discussed earlier, the privacy purposes are defined during the data collection process in an enterprise environment. The data provider or customer can restrict the use of their data. So, each customer or data provider can choose for which purpose his/her data can be used by the enterprise. When the personal data of the customer or data provider is accessed by the organizational employees or other parties, it is always accessible for a specific purpose mentioned by employees. Technically, during the data access process, the organizational users provide specific purpose and according to that purpose the system shows the requested data. In this case, at first the system retrives the data from the database and its privacy purposes. Then, it matches the privacy purposes setting for the customer data with the requester mentioned access purpose/purposes. If it matches then the system allows the requester to use the data; otherwise it is prohibited. Similarly, in the case of data insertion into the system,

data is always inserted with some privacy purpose.

In the basic approach, the information and privacy purpose labels are stored in one single table. So, the database engine or system traverses the whole dataset in this database for each relational database operation. For some specific relational database operations this storage model is not quite compatible in terms of performance and other attributes. In the following subsection, for better understanding of the concept and mechanism, we will discuss the storage scheme and its operation with an example system.

3.1.1 Basic Approach with the Sample Database

In this section, we will discuss the concept with the sample system. We will use this system for all of our discussions, implementations, and system testing throughout this whole thesis. We decided to use an e-commerce or online shopping system as a sample system database. E-commerce is increasingly a cooperative business that involves many individuals and organizations. With the rapid development of information technologies, it is increasingly convenient and efficient.

We will focus on the customer data table which is used to store the customer sensitive information like the name, customer address, postal code, credit card information, etc. In an enterprise environment, these customer data are used for different business requirements and purposes, such as different kind of marketing, market research, pre-sales and post-sales product analysis, support, shipment of products, billing etc. So, the customers' information is analyzed and used by all operational employees of the organization for different purposes. In our sample database, we store the data providers' personal information with privacy purposes.

In the basic system the information related to privacy purposes is stored in a different column from the actual data. The user identification number (UserID) is the primary key. Therefore, there is only one record for each UserID. In our example, we employed 5 purposes: Administration (A), Marketing (M), Finance (F), Purchase

UserID	Title	L_Title	FirstName	L_FirstName	LastName	L_LastName
4	Mr	{A,M,F}	Gustavo	{A,S}	Achong	{A,M,F}
5	Ms.	{A,M,F}	Catherine	{A,P}	Abel	{A,M,F}
6	Ms.	{A,M}	Kim	{A,S}	Abercrombie	{A,M}
7	Sr.	{A,S}	Humberto	{A,M,F}	Acevedo	{A,S}
8	Sra	{A,P}	Pilar	{A,M,F}	Ackerman	{A,P}
9	Ms.	{A,S}	Frances	{A,M}	Adams	{A,S}

Table 3.1: Partial customer information table of the sample system

(P), and Shipping (S). According to these five privacy purposes, the customers' or data providers' information can be accessed or used by the employees of the organization. In the sample data table for customers or data providers information, we have tried to collect the data which have the data integrity of a real life system. The data table stores 18 columns. All columns, except for the primary key columns(s), have an additional column giving the privacy label. In Table 3.1, we illustrate a part of the customer data table for the sample system.

As described earlier, in Table 3.1, we store each attribute of customer information with a set of purposes. We store the data randompurpose set. When an organizational employee wants to request some data, according to the purpose sets, data are visible; otherwise it shows the NULL value. Here, the system will have to check each data attribute along with its purpose set as requested by the user and present the data if it matches with the purpose for each request.

3.2 Proposed Approaches

The overarching goal of this dissertation is to come up with ideas for different storage methods for privacy purposes in a relational database and analyze the system characteristics. So, in this section, we will talk about three different proposed storage

schemes to store relational data with privacy purposes.

3.2.1 Approach 1

In the basic approach, each data attribute is labelled according to privacy purposes. Here, each data is matched with a purpose set. In this first alternative, instead of using the attribute-based labelling, we use a relational table based labelling. We partition the data into different tables according to the privacy purpose set. Suppose one data item is accessible for the marketing purpose; we will store that data in the marketing purpose table. There are some steps to perform to store the data according to this approach. First, we will determine all the purposes which are present in this system. Second, we will create the same number of data tables as there are purposes with all data attribute columns; more precisely, we create one data table for each purpose. Last, we will store the data according to the purposes in these tables; otherwise we will store NULL values into each table. If one data attribute contains multiple purposes, then we will store the data into multiple tables; otherwise we will store NULL values. Additionally, one base table is used store the purpose set with the data in a table for different administrative purposes.

3.2.1.1 Approach 1 with the Sample System

We now show Approach 1 with the sample system according to Table 3.1. Our example system has five different purposes in the sample system: Administration (A), Marketing (M), Financial (F), Shipping (S), and Purchase (P). After determining the purposes, we create five separate tables with the same data attributes. We do not need attributes for purpose labels as this information is now represented by the table name. Here, each table is used to store the data for one of the different administration purposes. Next, according to the purpose set for each data item, we store the data in the appropriate table(s). In this approach, each tuple is stored with its primary key, which is same for each user in every table. Tables 3.2, 3.3, 3.4, 3.5, and 3.6 show the

UserID	Title	FirstName	LastName
4	Mr	Gustavo	Achong
5	Ms.	Catherine	Abel
6	Ms.	Kim	Abercrombie
7	Sr.	Humberto	Acevedo
8	Sra	Pilar	Ackerman
9	Ms.	Frances	Adams

Table 3.2: Partial customer information table for Admin purpose (Table name: Admin purpose table)

UserID	Title	FirstName	LastName
4	Mr	NULL	Achong
5	Ms.	NULL	Abel
6	NULL	NULL	NULL
7	NULL	Humberto	NULL
8	NULL	Pilar	NULL
9	NULL	NULL	NULL

Table 3.3: Partial customer information table for finance purpose (Table name: Finance purpose table)

data for the sample system.

In this approach, when a system user requests data, according to the access purpose, the data is retrieved from the appropriate tables. We store the primary key in each purpose table to keep track of the customers' information. When there are no data items for a certain customer are present in the purpose table, we still keep the void information with the primary key, to avoid the extra cost of data insertion during the update process.

UserID	Title	FirstName	LastName
4	Mr	NULL	Achong
5	Ms.	NULL	Abel
6	Ms.	NULL	Abercrombie
7	NULL	Humberto	NULL
8	NULL	Pilar	NULL
9	NULL	Frances	NULL

Table 3.4: Partial customer information table for marketing purpose (Table name: Marketing purpose table)

UserID	Title	FirstName	LastName
4	NULL	NULL	NULL
5	NULL	Catherine	NULL
6	NULL	NULL	NULL
7	NULL	NULL	NULL
8	Sra	NULL	Ackerman
9	NULL	NULL	NULL

Table 3.5: Partial customer information table for purchase purpose (Table name: Purchase purpose table)

UserID	Title	FirstName	LastName
4	NULL	Gustavo	NULL
5	NULL	NULL	NULL
6	NULL	Kim	NULL
7	Sr.	NULL	Acevedo
8	NULL	NULL	NULL
9	Ms.	NULL	Adams

Table 3.6: Partial customer information table for shipping purpose (Table name: Shipping purpose table)

3.2.2 Approach 2

The main problem with Approach 1 is data redundancy. In our next approach, we will try to avoid data redundancy and keep the system operations straightforward and easy. In this approach, we store the data in one table and privacy masks in a table for each privacy label. The main functionality of the purpose table is to mask the customer information with stored values with purpose table. When data is requested with a specific purpose, the system finds the data from the data table and then it checks the stored values in that specific privacy purpose table. If the purpose table contains 1 for that specific user and purpose, it provides the data for that specific request; otherwise it prohibits the data accessibility and shows a NULL value.

The table with the data is called the data table and the tables with purposes are called purpose tables. Instead of having a purpose set for each attribute, we create one table for each purpose present in the system and store the purpose set values as binary values for each customer attribute. In each purpose table, if the data is accessible for that certain purpose, the system stores 1, otherwise 0.

We will now show the concept with a simple example. Suppose, one customer name is Bob and he wants his name accessible only for the marketing purpose but not for

UserID	Title	FirstName	LastName
4	Mr	Gustavo	Achong
5	Ms.	Catherine	Abel
6	Ms.	Kim	Abercrombie
7	Sr.	Humberto	Acevedo
8	Sra	Pilar	Ackerman
9	Ms.	Frances	Adams

Table 3.7: Partial customer information data table (According to Approach 2)

the financial purposes. In this case, customer name Bob is stored in the data table, and the system stores 1 and 0 in marketing and financial purpose tables respectively. When an enterprise user requests the name of Bob for the marketing purpose, the system will show the data after masking it with 1. In case of the financial purpose, it shows a NULL value after masking the name attribute with 0. All customer data are accessible for the administration purpose. Therefore, in this system, there is no purpose table for the administrative purpose to avoid data redundancy. For the administrative purpose, the entire data table is accessible and there is no masking required. In this approach, data redundancy is present only for the privacy purpose sets but not for the customer information.

Now, we will show this approach using our sample system. In Table 3.1, we illustrated the basic storage pattern for the sample system. According this approach, we separate the data and purposes. Then we store the data into the data table. Table 3.7 shows the data table for the sample system for Approach 2.

At first, we will determine the number of purposes in the sample system and create purpose tables accordingly. In our sample system, there are four purpose tables, namely, marketing purpose table, finance purpose table, shipping purpose table and purchase purpose table. According to the purposes mentioned in the sample system, purpose values are stored in four purpose tables. In the sample system, there is

UserID	Title	FirstName	LastName
4	1	0	1
5	1	0	1
6	0	0	0
7	0	1	0
8	0	1	0
9	0	0	0

Table 3.8: Finance purpose table (According to Approach 2)

UserID	Title	FirstName	LastName
4	1	0	1
5	1	0	1
6	1	0	1
7	0	1	0
8	0	1	0
9	0	1	0

Table 3.9: Marketing Purpose table (According to approach 2)

no purpose table for the administrative purpose and for this purpose data can be accessed from the data table. After creating the purpose table with the same number of attributes as the data table, we store the binary values in those purpose tables. Tables 3.8, 3.9, 3.10, and 3.11 show the purpose tables for the example. Here, if one system user requests information for the marketing purpose, the system finds out the data then masks the data against the binary values of the marketing purpose table.

UserID	Title	FirstName	LastName
4	0	0	0
5	0	1	0
6	0	0	0
7	0	0	0
8	1	0	1
9	0	0	0

Table 3.10: Purchase Purpose table (According to approach 2)

UserID	Title	FirstName	LastName
4	0	1	0
5	0	0	0
6	0	1	0
7	1	0	1
8	0	0	0
9	1	0	1

Table 3.11: Shipping Purpose table (According to approach 2)

UserID	Title	FirstName	LastName
4	Mr	Gustavo	Achong
5	Ms.	Catherine	Abel
6	Ms.	Kim	Abercrombie
7	Sr.	Humberto	Acevedo
8	Sra	Pilar	Ackerman
9	Ms.	Frances	Adams

Table 3.12: Partial customer information data table (According to approach 3)

3.2.3 Approach 3

Approach 3 is very similar to the previous approaches: separating data and purposes into multiple separate relational tables. In this approach, there is only one purpose table instead of multiple tables. The architecture of the purpose table is also changed slightly in this approach. Here, one new attribute, purpose, is introduced in the purpose table. As mentioned earlier, in the purpose table, the privacy purpose set is stored in binary data format. Now, this purpose attribute defines the purpose name for each tuple in the purpose table. In the case of the purpose table, the key value is the combination of UserID and Purpose. Here, it requires slightly more disk space than Approach 2 as there is one extra column with the same number of rows. However, for some basic SQL operations this approach works well.

According to this approach, first the system creates the data and new purpose table. After successful creation of data and purpose tables, the system stores information in the data table and the purpose set into the purpose table. In Table 3.12 and 3.13, we show the data table and purpose table according to this approach for our example.

UserID	Title	FirstName	LastName	Purpose
4	1	0	1	F
5	1	0	1	F
6	0	0	0	F
7	0	1	0	F
8	0	1	0	F
9	0	0	0	F
4	1	0	1	M
5	1	0	1	M
6	1	0	1	M
7	0	1	0	M
8	0	1	0	M
9	0	1	0	M
4	0	0	0	P
5	0	1	0	P
6	0	0	0	P
7	0	0	0	P
8	1	0	1	P
9	0	0	0	P
4	0	1	0	S
5	0	0	0	S
6	0	1	0	S
7	1	0	1	S
8	0	0	0	S
9	1	0	1	S

Table 3.13: Purpose table (According to approach 3)

3.3 Design and Implementation

In this section, we describe the implementation of the prototype. We have used Microsoft Visual Studio 2008 as the development environment and the C# language as the programming language to implement the Extended RBAC security model in our system for this thesis. We use SQL Server 2005 compact version as our system database and Microsoft SQL Server Management Studio, a graphical management tool, for the database. In our e-commerce prototype, we have 5 purposes: Administration (A), Marketing (M), Finance (F), Purchase (P) and Shipping (S). In this prototype, we focus on different data storage policies which are described in Section 3.2 and different SQL operations to study the accessibility of customer data. As the main focus is on the database engine and SQL operations, we will not focus on the front-end of the prototype in this thesis.

This section proceeds as follows: Firstly, we will introduce the structure of our

e-commerce system, a 3-Tier Architecture. Secondly, we will illustrate the design of the customer database and data manipulation. Thirdly, we will illustrate data access and SQL operations for the different approaches.

3.3.1 Three-Tier Architecture

We have implemented the e-commence prototype with a 3-tier architecture approach that consists of three layers, and uses data-transfer objects to pass data back and forth. The three-tier architecture, which is a client-server architecture, has three layers: a presentation tier or a GUI tier, a business logic tier and a database tier. The three layers are developed and maintained as independent modules and can run on separate platforms [7]. Figure 3.1, illustrates the 3-Tier Architecture for the e-commerce system.

The presentation layer is the application interface which is visible to the system user; it can be a web page, windows application, etc. In our research, the presentation layer or user GUI is developed using Visual Studio 2008. The business logic tier validates the inputs, check system conditions, and provides the output in a correct format. All the calculations and tests take place in this layer. The database tier, deals with storing the data into the database, data tables, different data related methods, procedures etc. Each tier in this architecture is independent and separated from each other. In our prototype, we will not focus on the presentation or business logic tier; we will only focus on the data access tier.

3.3.2 Database Design

In an e-commerce system, the customer information is critical and personal. This information needs to be stored in a secure database, the design of which is a very crucial part of system development. Usually, in an e-commerce system, there are multiple data tables to make the system fully functional. But, in our implementation, we will focus only on the customer information. In this subsection, we will focus on

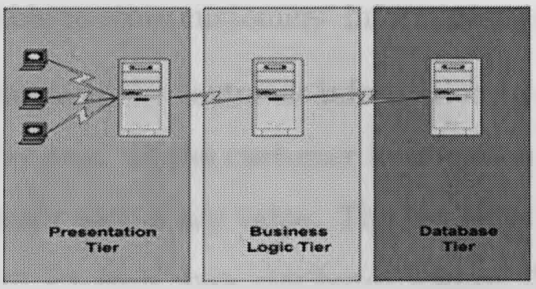


Figure 3.1: 3-Tier Architecture for Extended RBAC application

Table - dbo.CustomerInfo* | Diagram - SADIM-PC.ibac.Diagram_0*

Column Name	Data Type	Allow Nulls
UserId	int	
Title	varchar(50)	☑
L_Title	varchar(50)	☑
FirstName	varchar(50)	☑
L_FirstName	varchar(50)	☑
LastName	varchar(50)	☑
L_LastName	varchar(50)	☑
EmailAddress	varchar(50)	☑
L_EmailAddress	varchar(50)	☑
Phone	varchar(50)	☑
L_Phone	varchar(50)	☑
Address	varchar(50)	☑
L_Address	varchar(50)	☑
City	varchar(50)	☑
L_City	varchar(50)	☑
Province	varchar(50)	☑
L_Province	varchar(50)	☑
PostalCode	varchar(50)	☑
L_PostalCode	varchar(50)	☑
DateOfBirth	varchar(50)	☑
L_DateOfBirth	varchar(50)	☑
MaritalStatus	varchar(50)	☑
L_MaritalStatus	varchar(50)	☑
Gender	varchar(50)	☑
L_Gender	varchar(50)	☑
CardType	varchar(50)	☑
CardNumber	varchar(50)	☑
L_CardNumber	varchar(50)	☑
ExpMonth	varchar(50)	☑
L_ExpMonth	varchar(50)	☑
ExpYear	varchar(50)	☑
L_ExpYear	varchar(50)	☑
BillingAddress	varchar(50)	☑
L_BillingAddress	varchar(50)	☑
ShippingAddress	varchar(50)	☑
L_ShippingAddress	varchar(50)	☑

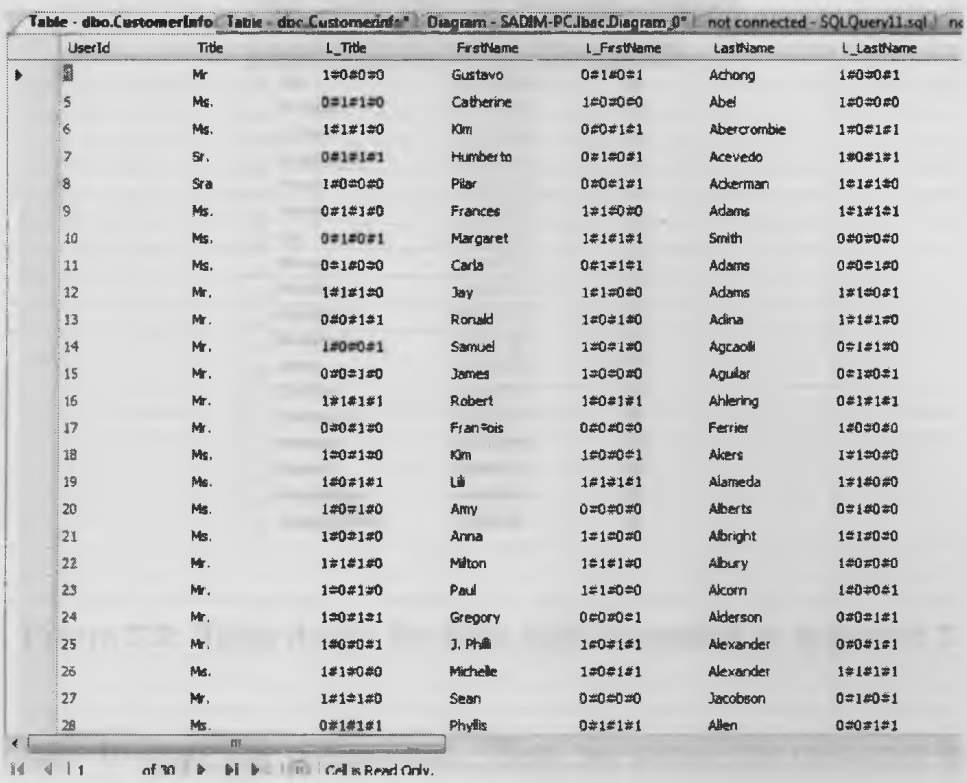
Figure 3.2: Table design for the customer database according to Basic approach

the design of the data table to store customers' information in an e-commerce system.

First, we have implemented the customer information table as the basic approach of the Extended RBAC system. In the customer information table, we have stored 18 attributes for each customer with a key value. The key value is UserID. Moreover, we have stored a purpose set for each data attribute. Figure 3.2, shows the data table design for the basic approach. To store the purpose set in an SQL server 2005 data table, we use a # separated string value, like 1#0#1#1. Here, 1 means the user has chosen this privacy purpose and 0 means otherwise. We defined in the system that the first bit of the purpose string is used for the finance purpose, the second indicates whether or not marketing is chosen, and purchasing and shipping purpose set values are represented by the third and fourth bits. Suppose, in the customer information table, one data item for a customer has the purpose set {M,F}; then, the purpose set is stored as 1#1#0#0. Figure 3.3, illustrates part of the customer information table.

Now, we will discuss the database design for our proposed approach 1 which we have discussed in section 3.2.1. In this approach, we propose to store the data in different relational tables according to chosen purposes. In our implementation, we have created five different data tables as there are five different purposes to access the data in the proposed system. We have created a data table for marketing, finance, admin, purchase and shipping purposes with same number of data attributes. In Figure 3.4, we show the data table design. Here, we store the data for each user, after analyzing the purpose set, into those tables. As mentioned earlier, this approach has data redundancy. Here, the primary key is UserID for each table. For every data table, each user is identified with their assigned UserID.

In our proposed approach 2, we separate the data and purpose into different tables. Here, the purpose table is used for data masking to determine the data accessibility for each data request with a specific purpose. In the process of implementing this approach, first, we created one data table to store the customer information and four



UserId	Title	L_Title	FirstName	L_FirstName	LastName	L_LastName
4	Mr	1#0#0#0	Gustavo	0#1#0#1	Achong	1#0#0#1
5	Ms.	0#1#1#0	Catherine	1#0#0#0	Abel	1#0#0#0
6	Ms.	1#1#1#0	Kim	0#0#1#1	Abercrombie	1#0#1#1
7	Sr.	0#1#1#1	Humberto	0#1#0#1	Acevedo	1#0#1#1
8	Sra	1#0#0#0	Pilar	0#0#1#1	Ackerman	1#1#1#0
9	Ms.	0#1#1#0	Frances	1#1#0#0	Adams	1#1#1#1
10	Ms.	0#1#0#1	Margaret	1#1#1#1	Smith	0#0#0#0
11	Ms.	0#1#0#0	Carla	0#1#1#1	Adams	0#0#1#0
12	Mr.	1#1#1#0	Jay	1#1#0#0	Adams	1#1#0#1
13	Mr.	0#0#1#1	Ronald	1#0#1#0	Adina	1#1#1#0
14	Mr.	1#0#0#1	Samuel	1#0#1#0	Agcaoili	0#1#1#0
15	Mr.	0#0#1#0	James	1#0#0#0	Aguilar	0#1#0#1
16	Mr.	1#1#1#1	Robert	1#0#1#1	Ahlering	0#1#1#1
17	Mr.	0#0#1#0	François	0#0#0#0	Ferrier	1#0#0#0
18	Ms.	1#0#1#0	Kim	1#0#0#1	Akers	1#1#0#0
19	Ms.	1#0#1#1	Lili	1#1#1#1	Alameda	1#1#0#0
20	Ms.	1#0#1#0	Amy	0#0#0#0	Alberts	0#1#0#0
21	Ms.	1#0#1#0	Anna	1#1#0#0	Albright	1#1#0#0
22	Mr.	1#1#1#0	Milton	1#1#1#0	Albury	1#0#0#0
23	Mr.	1#0#1#0	Paul	1#1#0#0	Alcorn	1#0#0#1
24	Mr.	1#0#1#1	Gregory	0#0#0#1	Alderson	0#0#1#1
25	Mr.	1#0#0#1	J. Phill	1#0#1#1	Alexander	0#0#1#1
26	Ms.	1#1#0#0	Michelle	1#0#1#1	Alexander	1#1#1#1
27	Mr.	1#1#1#0	Sean	0#0#0#0	Jacobson	0#1#0#1
28	Ms.	0#1#1#1	Phyllis	0#1#1#1	Allen	0#0#1#1

Figure 3.3: A part of the customer information table

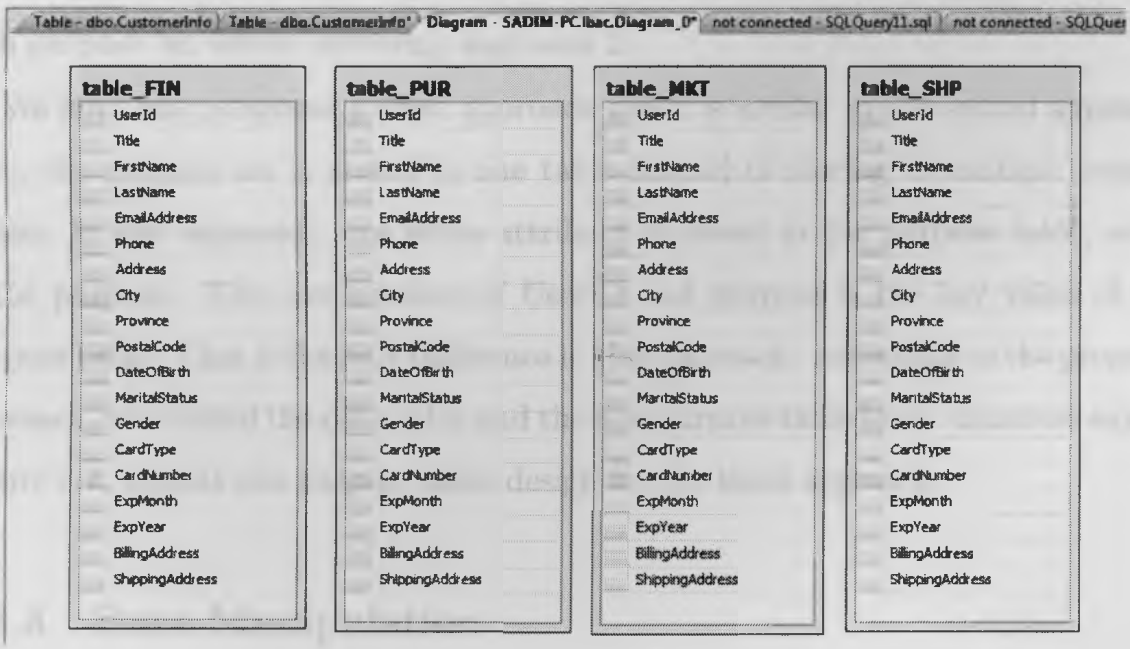


Figure 3.4: Table design for different purposes according to approach 1

Figure 3.5: Table design for data table according to approach 2

different tables to store the purpose set. Then, we stored the customer information with the key (UserID) into the data table and the purposes are set to 1 or 0 in the corresponding purpose table. Figure 3.5, illustrates the table design of the data table according to this approach. Figure 3.6, shows a part of the marketing purpose table with purpose set values according approach 2.

We have also proposed a third approach which is similar to the second approach. Here, the purpose set is stored in one table instead of storing in multiple purpose tables. In this approach, one extra attribute is stored in the purpose table, which is the purpose. The combination of UserID and purpose is the key value of this purpose table. That is the only difference in this approach. According to the proposed approach, we created the data table and the new purpose table in our database engine. Figure 3.7, depicts the purpose table design for the third approach.

3.3.3 Data Manipulation

To analyze the system, we need to use real customer information from an e-commerce system. Due to privacy issues, it is not possible to use the data from an e-commerce

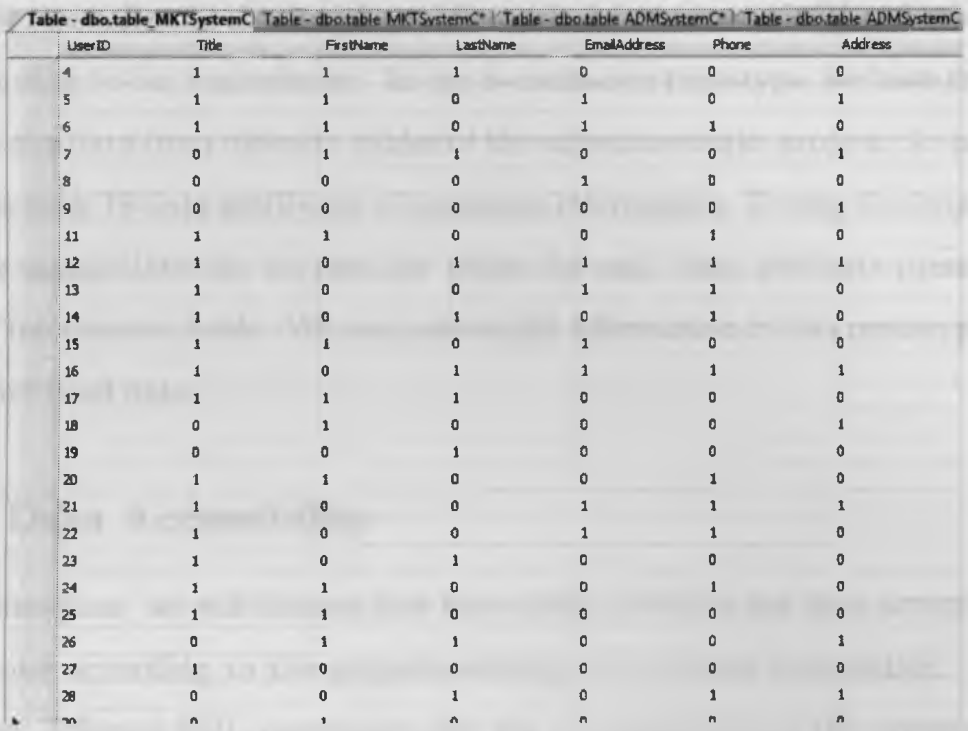


Table - dbo.table_MKTSystemC | Table - dbo.table_MKTSystemC* | Table - dbo.table_ADMSystemC* | Table - dbo.table_ADMSystemC

UserID	Title	FirstName	LastName	EmailAddress	Phone	Address
4	1	1	1	0	0	0
5	1	1	0	1	0	1
6	1	1	0	1	1	0
7	0	1	1	0	0	1
8	0	0	0	1	0	0
9	0	0	0	0	0	1
11	1	1	0	0	1	0
12	1	0	1	1	1	1
13	1	0	0	1	1	0
14	1	0	1	0	1	0
15	1	1	0	1	0	0
16	1	0	1	1	1	1
17	1	1	1	0	0	0
18	0	1	0	0	0	1
19	0	0	1	0	0	0
20	1	1	0	0	1	0
21	1	0	0	1	1	1
22	1	0	0	1	1	0
23	1	0	1	0	0	0
24	1	1	0	0	1	0
25	1	1	0	0	0	0
26	0	1	1	0	0	1
27	1	0	0	0	0	0
28	0	0	1	0	1	1

Figure 3.6: Partial view of marketing purposes table according to approach 2

Gender	CardType	CardNumber	ExpMonth	ExpYear	BillingAddress	ShippingAddress	Purpose
1	0	0	0	1	0	0	MKT
1	1	0	0	1	0	0	MKT
1	1	0	1	1	1	0	MKT
0	0	1	0	0	1	1	MKT
1	0	0	1	1	1	1	MKT
1	1	1	0	1	0	0	MKT
1	0	0	0	1	0	1	MKT
1	1	1	1	0	0	0	MKT
0	1	1	0	0	0	1	MKT
0	1	0	0	0	1	0	MKT
0	0	0	0	1	0	0	FIN
0	0	1	0	0	1	1	FIN
1	1	1	1	1	1	1	FIN
1	1	1	0	0	0	0	FIN
1	1	1	1	0	1	0	FIN
1	1	0	1	1	1	1	FIN
1	0	0	0	1	0	1	FIN
0	0	1	0	0	1	1	FIN
1	0	1	1	1	1	0	FIN
0	1	0	0	1	0	0	FIN
0	1	1	1	0	1	1	FIN
0	1	1	1	1	1	0	FIN
0	1	0	1	1	1	1	FIN
0	0	0	1	0	1	0	FIN

Figure 3.7: Purpose table according to approach 3 (Partial)

system. So, we collected the data from Microsoft Adventureworks[5] and modified the data according to our requirement. In our e-commerce prototype, we have tried to incorporate the data from different tables of the adventureworks project. As mentioned earlier, we took 18 data attributes of customer information. During the implantation phase, we manipulate the purpose set values for each data attribute present in the customer information table. We use meaningful information in this prototype instead of using artificial data.

3.3.4 Data Accessibility

In this subsection, we will discuss how the system provides the data accessibility for each purpose according to the purpose settings of customer information. Here, we will discuss different SQL operations and the functionalities of the system for each approach. In our study, we will discuss basic SQL operations: Select, Insert, Update and Delete for each approach. To implement the SQL operations, we created stored procedures in our database engine for each approach. Similarly, we will analyze the characteristics and performance for each of these SQL operations in the next chapter. We will discuss the basic SQL operations for each approach.

3.3.4.1 SQL Operations of Basic Approach

According to the basic approach, customer information is stored with a purpose set. In Subsection 3.3.2, we have discussed the database design for this approach. Now, we will discuss how the basic SQL operations like select, insert, delete and update work in this system.

In the basic approach, the select operation is most critical and challenging. When a system user or organizational employee requests customer data, the system first checks the customer data and then checks the purpose set of this data to provide the data accessibility and usability. To select the data according to the purpose set, we create a Select stored procedure in our database engine. As discussed earlier,

each data item is stored with a purpose set which is stored in the data table as a # separated string. For each request, we have to find out the purpose setting from this string. To do so, we formulated one split function in the database engine. Two inputs are required for the split function: the purpose string and delimiter value (#). This function splits the string and provides the purpose setting for each customer in the system. The Alter Function command in SQL creats the required stored procedure for later use. The code for this function is given below:

```
ALTER FUNCTION [dbo].[Split](@String varchar(8000), @Delimiter char(1))
returns @temptable TABLE (id int, items varchar(8000))
as
begin
    declare @idx int
    declare @slice varchar(8000)
declare @counter int
    set @counter =0
    select @idx = 1
        if len(@String)<1 or @String is null  return
    while @idx!= 0
    begin
        set @idx = charindex(@Delimiter,@String)
        if @idx!=0
            set @slice = left(@String,@idx - 1)
        else
            set @slice = @String
        if(len(@slice)>0)
set @counter = @counter+1
            insert into @temptable(id,Items) values(@counter, @slice)
set @String = right(@String,len(@String) - @idx)
```

```
        if len(@String) = 0 break
    end
return
end
```

With the help of the split function the select stored procedure finds out the privacy purpose setting for each customer data. After that, the select stored procedure returns the data or NULL values according to the purpose settings. Here, we used a cursor to traverse the whole table and check the settings. The stored procedure is defined as shown below:

```
ALTER PROCEDURE [dbo].[SelectDataSystemA] @userid varchar(10) AS
// Declare variable
SELECT * INTO #temptb FROM CustomerInfo
DECLARE @Select_C CURSOR
SET @Select_C = CURSOR FOR
//Select command to select rows according to request
OPEN @Select_C
FETCH NEXT
WHILE @@FETCH_STATUS = 0
BEGIN
//determine the purpose setting using the split function
// Show the data as requested
```

In this approach, insert, update and delete operations are the same as normal SQL operations. So, for these, we have created three different stored procedures with the normal SQL statement. From the interface of the system the user can input the data and check the output of the data. As the front end is not a part of our research, we do not discuss it.

3.3.4.2 SQL Operations of Approach 1

In approach 1, the data for different purposes are stored separately in different tables. Here, requested data are provided to the requester according to the purpose set from the specific purpose table. In this approach, the select SQL operation is the same as the usual simple database statement. According to the purpose provided with the data request, the system fetches the data from that specific table and shows that to the user. The select operation code is as follows:

```
Select * from table_MKT //Marketing purpose table
```

In the case of the insert operation, the system first inserts the data in the administrative table as this table contains all the data. After that, the system stores the actual data or NULL values according to the purpose set which is provided by the system admin or customer, and inserts the data into the other data tables. The data insertion process is maintained by the presentation layer of the system and provides the general insert command to the database engine.

In this approach, the update operation is complicated to implement. There are two types of update possible: data update and purpose update. In the case of data update, first the update operation updates the admin purpose table, then it retrieves the exiting permission of that attribute and updates this data in other tables as well. To provide this functionality in our system, we implemented an update stored procedure for updating data. User ID, the name of the data attribute (column name), new data to be updated and purpose set of that data are the inputs of this procedure. We show the update stored procedure:

```
ALTER PROCEDURE [dbo].[UpdateDataSystemB] @userid int, @ColumnName
varchar(50), @NewData varchar(50), @permission varchar(50) AS

Select @SQL = 'UPDATE table_ADM SET ' + @ColumnName + ' = '''
+  @NewData + ''' WHERE UserId = ' + convert(varchar(100), @userid)
```

```
exec(@SQL)

// Store the permission value in a parameter
if (@p=1)
begin
Select @SQL2 = 'UPDATE table_FIN SET ' + @ColumnName + ' = '''
+@NewData+ ''' WHERE UserId = ' + convert(varchar(100), @userid)
exec(@SQL2)
end

if (@p=1)
begin
Select @SQL2 = 'UPDATE table_MKT SET ' + @ColumnName + ' = ''
 +@NewData+ ''' WHERE UserId = ' + convert(varchar(100), @userid)
exec(@SQL2)
end

if (@p=1)
begin
Select @SQL2 = 'UPDATE table_PUR SET ' + @ColumnName + ' = '''
+@NewData+ ''' WHERE UserId = ' + convert(varchar(100), @userid)
exec(@SQL2)
end

if (@p=1)
begin
Select @SQL2 = 'UPDATE table_SHP SET ' + @ColumnName + ' = '''
+@NewData+ ''' WHERE UserId = ' + convert(varchar(100), @userid)
```

```
exec(@SQL2)
end
```

Permission update works a different way. The system first updates the permission set in the admin table. After that, the system updates the data value for the other tables, like marketing, finance etc. with that specific data value or NULL. If the provided permission is 1 for a specific purpose, the update procedure updates that attribute with the value from the administration table; otherwise it updates the value to NULL in that specific table. The permission update stored procedure is given below (here the stored procedure is only for updating the finance data table).

```
ALTER PROCEDURE [dbo].[UpdatePermissionSystemB1] @userid int,
@ColumnName varchar(50), @NewPermission varchar(50),
@ColumnName1 varchar(50) AS

Select @SQL = 'UPDATE table_ADM SET ' + @ColumnName + ' = ''
+  @NewPermission + ''' WHERE UserId = ' + convert(varchar(100),
@userid)

SELECT @Data1= Title from table_ADM where UserId =@userid


select @p= Items from dbo.split1(@NewPermission,'#') where id=1
if (@p=0)
begin
Select @SQL1 = 'UPDATE table_FIN SET ' + @ColumnName1 + ' =  null'
 + ' WHERE UserId = ' + convert(varchar(100), @userid)
exec(@SQL1)
end
```

```
else
begin
Select @SQL2 = 'UPDATE table_FIN SET ' + @ColumnName1 + ' = '''
+  @Data1 + ''' WHERE UserId = ' + convert(varchar(100), @userid)
exec(@SQL2)
end
```

The delete operation for this approach works as a normal SQL delete statement but instead of deleting from one table, the deletes stored procedure delete the data from the other four data tables as well.

3.3.4.3 SQL Operations of Approach 2

In Approach 2, we proposed the separation of data and purpose set. In the data table, we stored the data and formulated multiple purpose table to store purpose set values. Here, the data values are masked against the purpose table values. In this case, each SQL operation uses masking to provide the data accessibility. We implemented the masking of data with a purpose through the SQL condition operation.

We have created the select stored procedure to perform the select operation in the system. At the very beginning, the stored procedure joins the data and each purpose table together into a temporary table and starts masking. This procedure then checks the data with the purpose and masks the data against the purpose set values from multiple purpose tables: table_MKT, table_FIN, table_PUR and table_SHP. In this SQL operation, the stored procedure updates the data to NULL values if the procedure found 0 in the purpose set value for that data attribute. Here, the key value is UserID for each customer in the system. The partial select stored procedure (only for masking against the marketing purpose table) is given below:

```
ALTER PROCEDURE [dbo].[SelectMKTSystemC] AS
```

```
SELECT
table_ADMSystemC.userid
,table_ADMSystemC.Title as MTITLE
,table_MKTSystemC.Title as MTITLEPER
...

// Join data and purpose element into a temporary table

INTO #TEMP  FROM table_ADMSystemC, table_MKTSystemC where
table_ADMSystemC.UserID = table_MKTSystemC.UserID

update #TEMP set MTITLE = NULL WHERE MTITLEPER= 0
update #TEMP set MFIRSTNAME = NULL WHERE MFIRSTNAMEPER= 0
update #TEMP set MLASTNAME = NULL WHERE MLASTNAMEPER= 0
update #TEMP set MEMAIL = NULL WHERE MEMAILPER= 0
update #TEMP set MPHONE = NULL WHERE MPHONEPER= 0
update #TEMP set MADDRESS = NULL WHERE MADDRESSPER= 0
update #TEMP set MCITY = NULL WHERE MCITYPER= 0
update #TEMP set MPROVINCE = NULL WHERE MPROVINCEPER= 0
update #TEMP set MPOSTCODE = NULL WHERE MPOSTCODEPER= 0
update #TEMP set MPROVINCE = NULL WHERE MPROVINCEPER= 0
update #TEMP set MDOB = NULL WHERE MDOBPER= 0
update #TEMP set MMARITAL = NULL WHERE MMARITALPER= 0
update #TEMP set MGENDER = NULL WHERE MGENDERPER= 0
update #TEMP set MCARDTYPE = NULL WHERE MCARDTYPEPER= 0
update #TEMP set MCARDNUMBER = NULL WHERE MCARDNUMBERPER= 0
update #TEMP set MEXPMONTH = NULL WHERE MEXPMONTHPER= 0
update #TEMP set MEXPYEAR = NULL WHERE MEXPYEARPER= 0
```

```
update #TEMP set MBILL = NULL WHERE MBILLPER= 0
update #TEMP set MSHIP = NULL WHERE MSHIPPER= 0

SELECT UserId,MTITLE,MFIRSTNAME,MLASTNAME,MEMAIL,MPHONE,
MADDRESS, MCITY,MPROVINCE,MDOB,MMARITAL,MGENDER,MCARDTYPE
,MCARDNUMBER, MEXPMONTH,MEXPYEAR,MBILL,MSHIP FROM #TEMP

DROP TABLE #TEMP
```

In this approach, all other operations like insert, update and delete work as general SQL operations. We create the stored procedure with the basic SQL statements as we did for the insert operation. The system inserts the data into the data table and inserts the purpose value into the purpose tables to perform these operations.

3.3.4.4 SQL Operations of Approach 3

In our proposed Approach 3, we follow the same concept as approach 2. Here, we use only one table to store the purpose set values instead of using multiple tables as in approach 2. One additional attribute added in this approach is the purpose name in the purpose table. In this approach, the combination of UserID and purpose is the key for the purpose table. The select operation follows the same procedure as approach 2 with a slight modification. Instead of using multiple joins with multiple purpose tables, here, this select procedure joins table one to select the data according to purpose settings. The partial stored procedure is given below:

```
ALTER PROCEDURE [dbo].[SelectSystemD] @mpurpose varchar(50)AS
SELECT
   table_ADMSystemD.userid
  ,table_ADMSystemD.Title as MTITLE
      ,table_PermissionSystemD.Title as MTITLEPER
```

```
....

INTO #TEMP  FROM table_ADMSystemD, table_PermissionSystemD where
table_ADMSystemD.UserID = table_PermissionSystemD.UserID and
table_PermissionSystemD.Purpose = @mpurpose

update #TEMP set MTITLE = NULL WHERE MTITLEPER= 0
update #TEMP set MFIRSTNAME = NULL WHERE MFIRSTNAMEPER= 0
update #TEMP set MLASTNAME = NULL WHERE MLASTNAMEPER= 0
update #TEMP set MEMAIL = NULL WHERE MEMAILPER= 0
update #TEMP set MPHONE = NULL WHERE MPHONEPER= 0
update #TEMP set MADDRESS = NULL WHERE MADDRESSPER= 0
update #TEMP set MCITY = NULL WHERE MCITYPER= 0
update #TEMP set MPROVINCE = NULL WHERE MPROVINCEPER= 0
update #TEMP set MPOSTCODE = NULL WHERE MPOSTCODEPER= 0
update #TEMP set MPROVINCE = NULL WHERE MPROVINCEPER= 0
update #TEMP set MDOB = NULL WHERE MDOBPER= 0
update #TEMP set MMARITAL = NULL WHERE MMARITALPER= 0
update #TEMP set MGENDER = NULL WHERE MGENDERPER= 0
update #TEMP set MCARDTYPE = NULL WHERE MCARDTYPEPER= 0
update #TEMP set MCARDNUMBER = NULL WHERE MCARDNUMBERPER= 0
update #TEMP set MEXPMONTH = NULL WHERE MEXPMONTHPER= 0
update #TEMP set MEXPYEAR = NULL WHERE MEXPYEARPER= 0
update #TEMP set MBILL = NULL WHERE MBILLPER= 0
update #TEMP set MSHIP = NULL WHERE MSHIPPER= 0

SELECT UserId,MTITLE,MFIRSTNAME,MLASTNAME,MEMAIL
```

```
,MPHONE,MADDRESS, MCITY,MPROVINCE,MDOB,MMARITAL
,MGENDER,MCARDTYPE,MCARDNUMBER,

MEXPMONTH,MEXPYEAR,MBILL,MSHIP FROM #TEMP
DROP TABLE #TEMP
```

In this approach, all other SQL operations are same as approach 2.

In this Chapter, we have introduced the Extended RBAC system with four different storage orientations for the sample system. From now on, we use the term System A, System B, System C and System D for the basic storage system, proposed approach 1, proposed approach 2, and proposed approach 3 respectively.

Chapter 4

Experiment Design

The purpose of this chapter is to introduce the reader to the experimental design to study performance characteristics of the Extended RBAC system with different storage schemes. This chapter also provides details about the hardware and software used in our experiments. This chapter is organized as follows: first, we will discuss the test environment and different performance measurement factors; second, we will discuss our experimental strategy and different testing scenarios; third, we will introduce different testing tools; and finally, we validate our experiments.

4.1 Test Environment

In our experiments, we test each system with different storage schemes based on basic database operations (SQL operations). We consider the privacy purposes as experimental inputs. We perform the tests to determine the system performance factors for different SQL operations. Before proceeding to the actual experiments, we prepare our test environment with basic software and applications with appropriate configurations. Usually, these configurations mean the complete setup of the database system and its applications.

To build the test environment, we first deploy the database system in two differ-

ent computer systems. We take two high-end computer systems to create the test environments. Each experimental environment is created in a Windows 7 machine with Intel (R) Core (TM) i5 CPU 660, 4 CPUs, 4 GB of physical memory and a 500 GB hard drive. As discussed, the database systems with all the features help us to deploy the sample systems with four different storage orientations. We deploy the system with the basic storage scheme and our three proposed solutions in both computer systems. We use the same configuration and the same volume of data in both systems. In other words, we can say that both systems are identical in terms of applications and resources. After that, we insert the customer information into the database engine to simulate a real situation. Once we have deployed the database engine into both systems with the required data, the system is ready for testing.

We treated and tested both systems independently to maintain maximum reliability and quality of services. Also, this ensures the accuracy of our measured results. There are many external factors which can affect the system performance and quality of experimental measurements, like multiple operations or processes running at the same time, testing and operating the systems over the internet, W-LAN etc. We avoid those external factors to make the system work perfectly and provide accuracy for experimental results.

4.2 Performance Measurements

One important point in developing a test plan is deciding which measurements will be collected. In our experiments, we analyze the system performance based on some basic SQL operations. We measured the following performance measurements or factors during the experiments:

4.2.1 Response Time

Average response time is a very vital performance measurement in a real life database environment. Response time is the time the database takes to execute a single transaction. Response time can be measured as the inverse of throughput of the system, where transactions are executed in a synchronized manner by a single user. In the case of concurrent users, this parameter is measured in a different way. This is so far the most important performance factor that determines the performance of SQL operation based performance experiments.

4.2.2 Throughput

Throughput is the number of transactions per second in a database environment. In this case, transactions mean SQL operations on the database system. It is very important to understand throughput in the extended RBAC system for two basic reasons. First, it provides the sustainability of our test systems with a certain number of transactions which is based on a certain number of users. Second, it provides the understanding of database applications in a real life situation.

Our main focus is to determine the performance of different SQL operations, not the whole database system. Therefore, some other performance parameters, like CPU Usage, Disk I/O, Memory usage etc., we will not consider in our experiments.

4.3 Experimental Strategy and Test Scenarios

In this section, we will discuss different experimental strategies and various test scenarios for different storage approaches for our extended RBAC system. These test scenarios will be used to build test cases, which are collections of one or more scenarios, executed concurrently.

In our experiments, the test cases are a collection of basic SQL operations. We execute different SQL operations for each test case on our database system and such

operations are grouped into transactions. The word transaction refers to a collection of operations. As discussed in Section 4.2, we determine the throughput and response time of each operation executed in our experiments. Here, we determine the transaction per second and time to perform each transaction (second or millisecond) for throughput and response time measurement respectively.

During the experiments, different test scenarios may show completely different performance results. It is not possible to avoid this behaviour, but we can define clearly every test so that we will be aware of the differences between them. There are some key strategies of our experiments:

- Iteration:

During the testing, we follow the process of iteration to collect the performance data. We define a fixed number of task iterations for each test. In order to provide more reliability and data accuracy, we also average the collected measurements of each iteration step. Here, for each test case, we collect the data with 10 iterations.

- Number of Users:

During the design of experiments, we formulate the test plan with different numbers of users. In this way, we can understand and analyze the performance characteristics of the different sample systems more correctly. In our experiments, we perform the testing with a maximum of 100 concurrent users.

- Different Kinds of Transactions:

In order to decide whether the performance test was duly carried out, we performed different kinds of transactions: single transactions and concurrent transactions for each test case. These testing strategies help to achieve different kind of goals of our experiments.

In our performance testing, we will test the basic SQL operations for system A, B, C and D. Four basic SQL operations are used to execute the tests. For adding

content to the database, the Insert command is used. For retrieving content from the database, it is the Select command. For deletion, the Delete command is utilized, and for the update, it is the Update command. Each SQL operation is defined by the following components: business contexts, functional query definition and validation. To maintain a fair comparison among them, the same data set is used in all the systems.

We perform out experiments in two different testing modules. In module one, we measure the response time and throughput of the basic SQL operations according to different numbers of data items for each system. In module 2, we measure the response time and throughput of basic SQL operations with different numbers of concurrent users. There is another factor we need to consider in the testing procedure, which is privacy purposes. Usually, each SQL operation is performed with one or more specific purposes in different sample systems. Suppose we need to select some customer information from our test system; in this case, obviously, we need to mention the purpose for data accessibility. Moreover, for each module, we will measure the performance characteristics of SQL operations as per different kinds of purposes. We will consider the purposes only for select operations; with the other SQL operations (insert, update, delete) when purpose is take into account.

Now, we will discuss the test cases according to the two modules for our four different systems. In system A, we use the basic storage orientation of the extended RBAC sample system. The select operations are different in system A. The system checks each attribute against the privacy purpose set and presents the selected data to the requestor. For system B, the select operation is the same as the normal SQL select. In the case of system C and D, the system checks the data items with the purpose table and retrieves the requested data according to the purpose. In the first phase of this SQL operation testing, we will measure the average response time and transaction per second (TPS) / throughput for both of the testing modules according to each purpose of system A. As there are 5 purposes in our sample system, we

Test Case No	User load	Description	Purpose
1	1	Select 9000 data items from System A Customer Information table	Marketing
2	1	Select 18000 data items from System A Customer Information table	Marketing
3	1	Select 36000 data items from System A Customer Information table	Marketing
4	1	Select 90000 data items from System A Customer Information table	Marketing
5	1	Select 180000 data items from System A Customer Information table	Marketing
6	1	Select 360000 data items from System A Customer Information table	Marketing
7	1	Select 9000 data items from System A Customer Information table	Finance
8	1	Select 18000 data items from System A Customer Information table	Finance
9	1	Select 36000 data items from System A Customer Information table	Finance
10	1	Select 90000 data items from System A Customer Information table	Finance
11	1	Select 180000 data items from System A Customer Information table	Finance
12	1	Select 360000 data items from System A Customer Information table	Finance
13	1	Select 9000 data items from System A Customer Information table	Purchase
14	1	Select 18000 data items from System A Customer Information table	Purchase
15	1	Select 36000 data items from System A Customer Information table	Purchase
16	1	Select 90000 data items from System A Customer Information table	Purchase
17	1	Select 180000 data items from System A Customer Information table	Purchase
18	1	Select 360000 data items from System A Customer Information table	Purchase
19	1	Select 9000 data items from System A Customer Information table	Administration
20	1	Select 18000 data items from System A Customer Information table	Administration
21	1	Select 36000 data items from System A Customer Information table	Administration
22	1	Select 90000 data items from System A Customer Information table	Administration
23	1	Select 180000 data items from System A Customer Information table	Administration
24	1	Select 360000 data items from System A Customer Information table	Administration
25	1	Select 9000 data items from System A Customer Information table	Shipping
26	1	Select 18000 data items from System A Customer Information table	Shipping
27	1	Select 36000 data items from System A Customer Information table	Shipping
28	1	Select 90000 data items from System A Customer Information table	Shipping
29	1	Select 180000 data items from System A Customer Information table	Shipping
30	1	Select 360000 data items from System A Customer Information table	Shipping

Table 4.1: Module one test cases for select operation in system A

take the measurements for five purposes independently. To test the select operation according to module one, we use different numbers of data items selected by a single user. Here, we perform the select operation for 9000, 18000, 36000, 90000, 180000, 360000 data items for each purpose. In Table 4.1, we illustrate the select operation test cases for module one.

Now, we are going to discuss the insert operation of the system A according to module one. Here, we will not consider the different purposes as this operation is completely independent. We insert different numbers of data items into system A. In the insert operation experiment, we will insert the following number of data items: 18, 360, 900, 1260, 1800, 2700, and 3600. In Table 4.2, we illustrate the test cases for

Test Case No	User load	Description
1	1	Insert 18 data items into System A Customer Information table
2	1	Insert 360 data items into System A Customer Information table
3	1	Insert 900 data items into System A Customer Information table
4	1	Insert 1260 data items into System A Customer Information table
5	1	Insert 1800 data items into System A Customer Information table
6	1	Insert 2700 data items into System A Customer Information table
7	1	Insert 3600 data items into System A Customer Information table

Table 4.2: Module one test cases for insert operation in system A

the Insert operation:

In the experiment for the update operation testing for system A, we will update different data items with different numbers of users. We cannot perform the testing according to different modules because it is not possible to update same data items by different concurrent user, which is a deadlock situation. In this experiment, we update different numbers of data items like: 1, 2, 5, 10, and 20 with the same number of users, such as: 1, 2, 5, 10 and 20. After that, we average the collected data to provide the average response times. We perform the same update operation experiments for system C and D except system B.

In the case of system B, the update operation is different. Here, the update operation is divided into two parts: update the data items and update the privacy purposes. Both update operations are not like the sample SQL update operation. In this case, the system first updates the purpose or data item in the main table, then the system updates according to the change in different purposes data tables. Due to these phenomena of update operations we test the update operation according to two different scenarios: update privacy purpose and update data items. For the delete operation, we measure performance factor of one delete operation with a single user in system A due to deadlock problem. We will perform the same module one select and insert operational experiments for system B, C, D. All the test cases are illustrated in Appendix A.

Now, we are going to discuss the module two experiments for different systems. According to module two, we consider a different number of users for the insert and

Test Case No	User load	Description	Purpose
1	1	Select 9000 data items from System A Customer Information table	Marketing
2	10	Select 9000 data items from System A Customer Information table	Marketing
3	20	Select 9000 data items from System A Customer Information table	Marketing
4	30	Select 9000 data items from System A Customer Information table	Marketing
5	40	Select 9000 data items from System A Customer Information table	Marketing
6	50	Select 9000 data items from System A Customer Information table	Marketing
7	60	Select 9000 data items from System A Customer Information table	Marketing
8	70	Select 9000 data items from System A Customer Information table	Marketing
9	80	Select 9000 data items from System A Customer Information table	Marketing
10	90	Select 9000 data items from System A Customer Information table	Marketing
11	100	Select 9000 data items from System A Customer Information table	Marketing

Table 4.3: Module two test cases for select operation in system A (Partial)

Test Case No	User load	Description
1	1	Insert 18 data items into System A Customer Information table
2	10	Insert 18 data items into System A Customer Information table
3	20	Insert 18 data items into System A Customer Information table
4	30	Insert 18 data items into System A Customer Information table
5	40	Insert 18 data items into System A Customer Information table
6	50	Insert 18 data items into System A Customer Information table
7	60	Insert 18 data items into System A Customer Information table
8	70	Insert 18 data items into System A Customer Information table
9	80	Insert 18 data items into System A Customer Information table
10	90	Insert 18 data items into System A Customer Information table
11	100	Insert 18 data items into System A Customer Information table

Table 4.4: Module two test cases for insert operation in system A (Partial)

select SQL operations. As discussed earlier, update and delete operations are not considered in this testing module. Here, we perform the same operations as module one testing but with different levels of concurrent users. For module two experiments, we consider the following number of concurrent users: 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100. Table 4.3 illustrates a part of the test cases for the select operation for system A. Table 4.4 illustrates a part of the insert operation test cases for system A. We perform the same test for systems B, C and D. The complete module two test cases are given in Appendix B.

4.4 Performance Testing Tools

Selection of an appropriate testing tool is a significant issue related to conducting performance tests of SQL operations. Before starting the analysis, we try to analyze different application which meet our requirements. As we are testing basic SQL operations, the performance testing application must able to test different metrics for each SQL operation. Another basic requirement is the testing tool can provide different levels of workload with different numbers of user load. Additionally, the tool should have detailed reporting for each operation, enough flexibility and ease of use. After analyzing all these requirements, we decided to use SQL profiler [2] and Benchmark Factory [1] to test the SQL operations' performance.

The SQL profiler is a very efficient and robust tool to analyze the SQL server queries and performance of stored procedures. This tool provides the performance for each operation in the SQL server database engine. As our sample database is implemented in SQL server 2005, we choose this tool to monitor the performance for each SQL operation test. SQL profiler provides a detailed report for each operation or SQL query running in the SQL server.

To measure the SQL operations performance, we need to create a trace in the SQL profiler with all parameters required for the testing. During the creation of a trace file, we can choose the parameter which is required to analyze a specific SQL operation. After that, we run the trace file to capture the detailed report on each SQL operation. As mentioned earlier, the performance factors which are required for our testing are captured in the trace. Figure 4.1, represents a snapshot of a SQL profiler trace.

Despite the many advantages of SQL profiler, there are some problems also present. This tool does not have any features other than monitoring. There is no feature to define the iteration and put the concurrent user load into the sample system. One of the major goals is to test the system as a real system environment. So, we need to generate multiple user loads like a real system environment. For these situations,

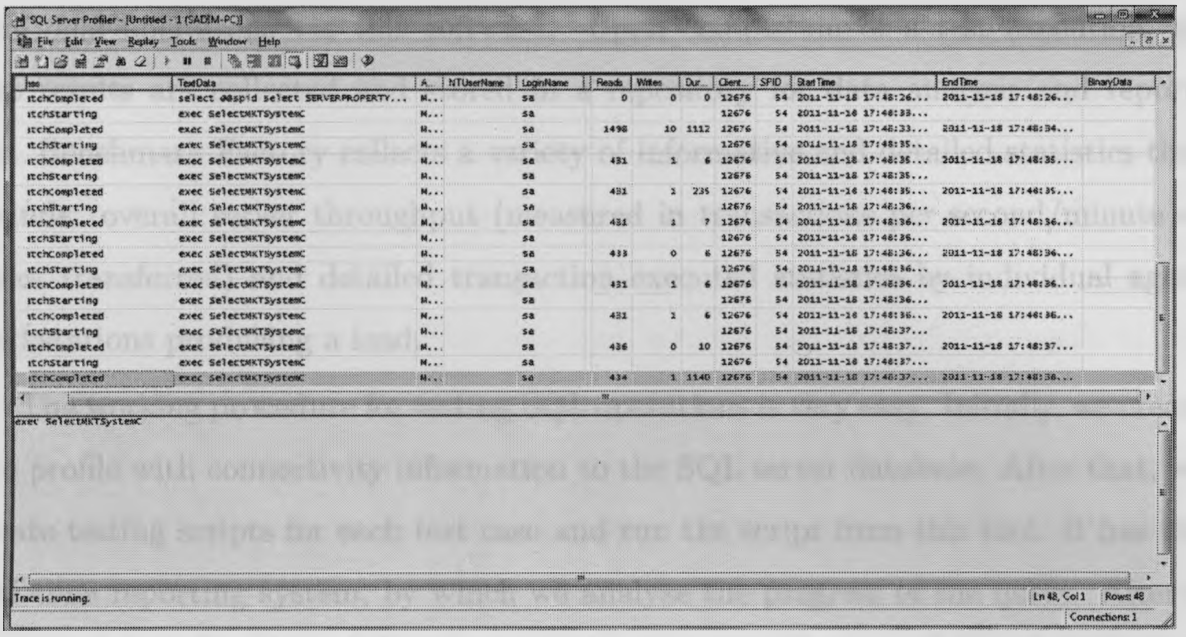


Figure 4.1: A snapshot of a Trace in the SQL Profiler

with SQL profiler, we need to do it manually with different SQL scripts. To overcome these situations, we also use the Benchmark Factory.

The Benchmark Factory is a database performance tool which simulates performance and benchmarks the databases. Benchmark Factory is available for Oracle, SQL Server, DB2, Sybase, MySQL, and other databases via ODBC connectivity. It can be used for base testing, load testing, stress testing and different kinds of performance monitoring purposes. It can simulate different levels of concurrent users with minimal resources. It is very easy to provide iterative SQL operations through this software. We test each SQL operation with different user load and iteration via this testing tool. The reporting system of this tool also is very well organized and useful.

Benchmark Factory for Databases is used to meet the database operations testing needs. Benchmark Factory is a database performance and code scalability testing tool that simulates users executing transactions on the database and replays production workloads in non-production environments. In this study, we simulate thousands of concurrent users with a minimal amount of hardware, simulates different database

relational operations via this software. Upon completion of a test execution, all test results are collected and stored in a repository for data analysis and reporting. Benchmark Factory collects a variety of informative and detailed statistics that include: overall server throughput (measured in transactions per second/minute or bytes transferred) and detailed transaction executed statistics by individual agent workstations producing a load.

The working procedure for testing SQL operations is very easy. Initially, we create one profile with connectivity information to the SQL server database. After that, we create testing scripts for each test case and run the script from this tool. It has the real time reporting system, by which we analyze the progress of the query. After a successful test, Benchmark Factory provides the detailed report for each operation. While we run the SQL operations through Benchmark Factory, the results for these operations are also captured by SQL profiler. We take the performance factors for each operation from SQL profiler and Benchmark Factory to measure each operation performance more accurately.

4.5 Validity of Experiments

Experimental validity [17] refers to the manner in which variables that influence both the results of the research and the generalizability to the population at large. It is very difficult to design an experiment to produce real life application performance. Moreover, every performance test may produce valid results only for a small set of operations; in case of a large operational environment, it works differently. Another problem related to performance analysis testing is researcher bias. During the design and analysis of our experiments, we tried to overcome these issues to maintain the validity of our performance testing.

During the testing, we measure each performance factor for each SQL operation with 10 iterations. For each SQL operation, the testing tool runs the command

10 times and provides the average performance factor as a result. In this way, we maintain the accuracy for each operational measurement in our experiments.

We use two different tools (SQL profiler and Benchmark Factory) to determine the performance factors of our experiments. For each test case, two testing tools provide the performance factor. After getting the results from both tools and analyzing the results, we take the average value reported of these two data as the result of this operational test case.

To provide the result with more accuracy and similarity to a real application in an enterprise environment, we use different numbers of data items and different numbers for user load. This makes the experiments design more accurate, convenient and realistic. Moreover, we run our experiment in two different computer systems with identical hardware and software resources. After analyzing, the design and strategy of our experiments, we believe we have a reliable testing methodology.

Chapter 5

Experimental Results

In this chapter, we will discuss some of the results obtained by execution of different test cases described in Chapter 4. Each test case is executed on two different computer systems with identical hardware and software configurations. The results were obtained from different test cases with five different purposes for four different storage models. The testing mainly observes the response time and throughput of basic SQL operations, namely, select, update, insert and delete. Unless otherwise stated, for all our measurements we repeat the experiment ten times and report the average measurement obtained from these ten runs.

5.1 Performance Analysis of the Select Operation

In this section, we report the measured results for the select operation of our four different systems with different storage scheme: system A, B, C and D. As discussed in Chapter 4, we will divide the testing and performance characteristics analysis in two parts: Module one tests and Module two tests.

In module one, we will test the systems with different numbers of data items. Here, we determine the performance factors (Response time and Throughput/ Transaction per second) of different systems according to different data items. In module one,

for select operation testing, we define the number of data items as standard to test through all the systems: 9000, 18000, 36000, 180000, and 360000. Moreover, we perform the select operation testing for each purpose individually. In this testing module, we also test performance factor for one and 20 concurrent users, to understand the evaluation and convenience of our measured results.

In module one, we will test the systems according to five different purposes and two different numbers of concurrent users. The configurations for these tests are given below:

Test 1 configuration:

- Number of data items : 9000, 18000, 36000, 180000, 360000
- Purpose: Admin
- Number of users : 1
- Environment: Two identical Windows system environments
- Test Systems: System A, B, C and D

Test 2 configuration:

- Number of data items : 9000, 18000, 36000, 180000, 360000
- Purpose: Finance
- Number of users : 1
- Environment: Two identical Windows system environments
- Test Systems: System A, B, C and D

Test 3 configuration:

- Number of data items : 9000, 18000, 36000, 180000, 360000

- Purpose: Marketing
- Number of users : 1
- Environment: Two identical Windows system environments
- Test Systems: System A, B, C and D

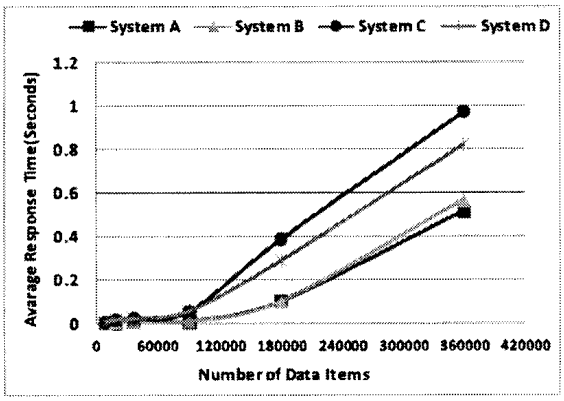
Test 4 configuration:

- Number of data items : 9000, 18000, 36000, 180000, 360000
- Purpose: Purchase
- Number of users : 1
- Environment: Two identical Windows system environments
- Test Systems: System A, B, C and D

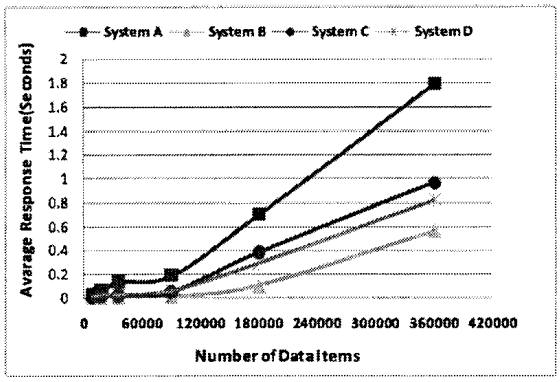
Test 5 configuration:

- Number of data items : 9000, 18000, 36000, 180000, 360000
- Purpose: Shipping
- Number of users : 1
- Environment: Two identical Windows system environments
- Test Systems: System A, B, C and D

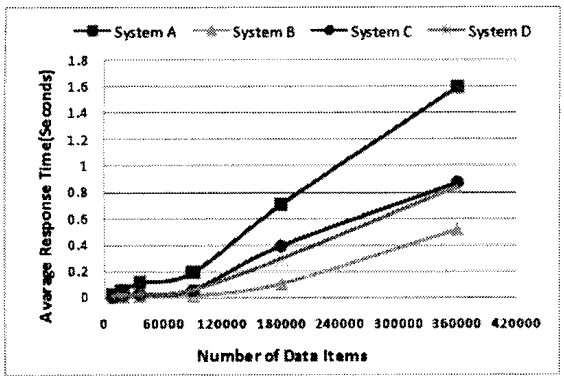
Figure 5.1 shows the results of testing select for module one. In this graph, on the x axis, we show the number of data items and on the y axis, response times. The graph shows the response times of different systems for each purpose. In Figure 5.1a shows the response time in different systems for the administration purpose. As all



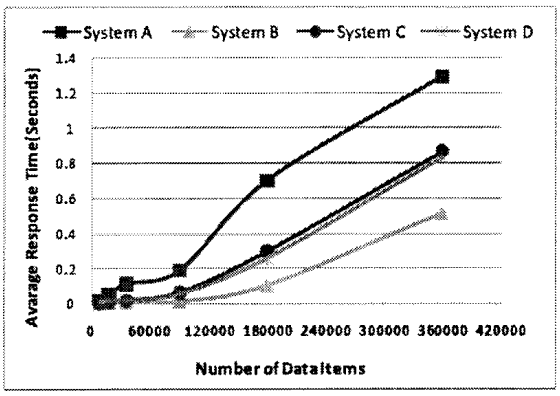
(a) Response time of module one testing for administration purpose



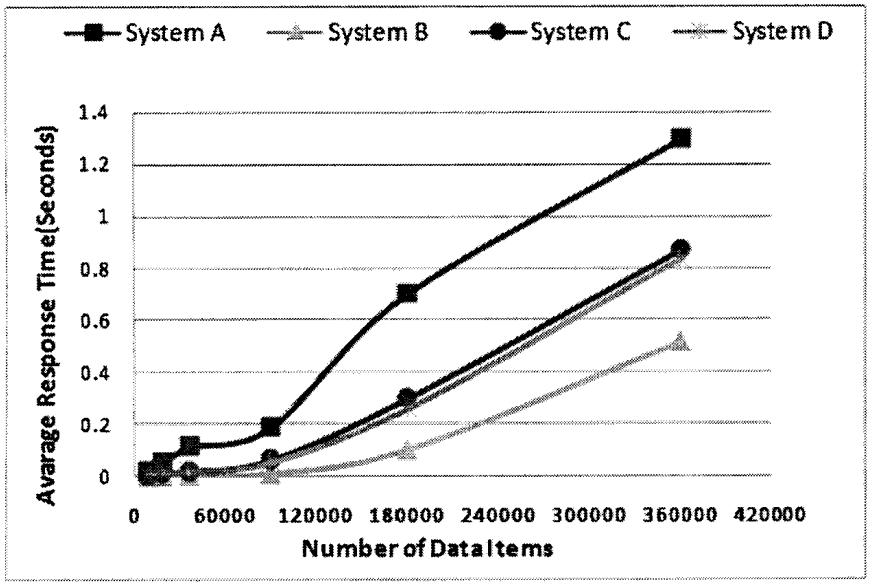
(b) Response time of module one testing for finance purpose



(c) Response time of module one testing for marketing purpose



(d) Response time of module one testing for purchase purpos



(e) Response time of module one testing for shipping purpose

Figure 5.1: Response time of Select module one testing

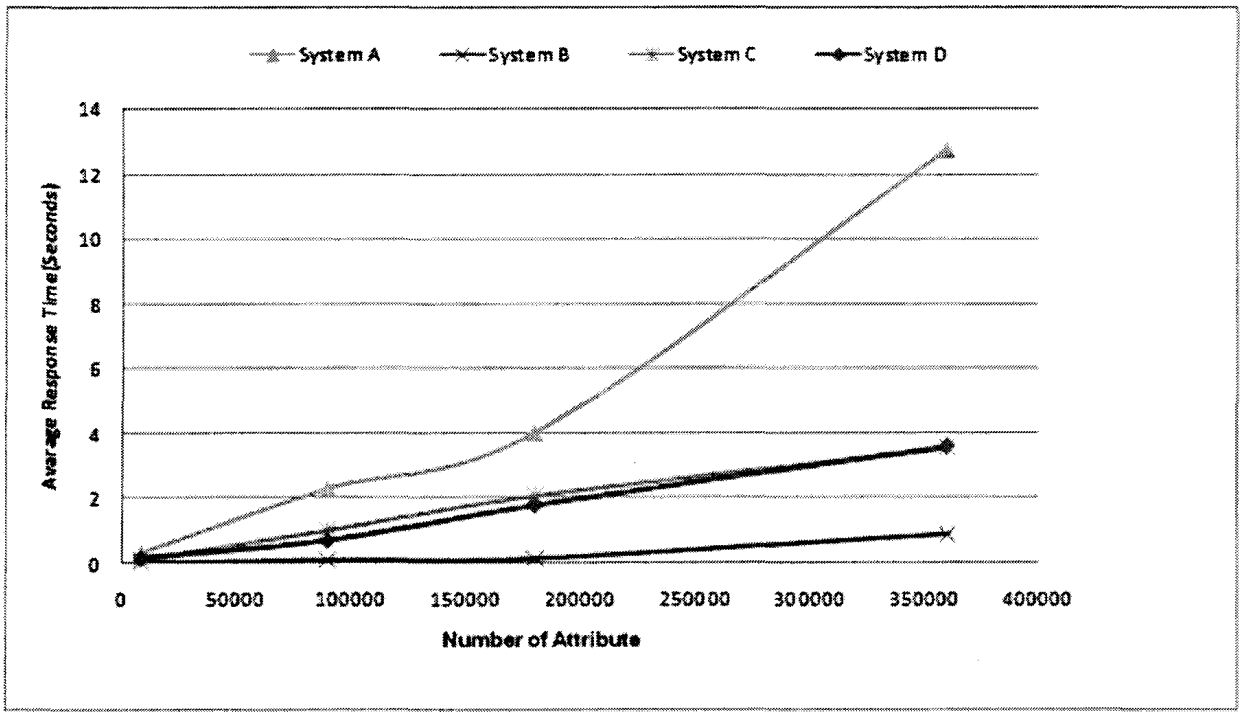


Figure 5.2: Throughput of module one testing Select for marketing purpose

the data is accessible for the administration purpose, during this select operation, there is no checking with the purpose set required. Therefore, the response times are almost the same as system B. System A shows worst performance for all other cases, illustrated in Figures 5.1b, 5.1c, 5.1d, and 5.1e. During the select operation of the system A for other purposes, the system traverses the whole data table to check each attribute with the purpose set defined for that attribute. Therefore, the performance is too low or response time is too high. Response times for other systems (System B, C and D) are pretty much the same for each case. According to the graphs of Figure 5.1, response times for system B are the lowest because during the select operation, there is no checking required for this system. In system B, different information for different purposes is stored in different tables. In Appendix C, we show the response time of Select operation for System A with 10 iterations.

Here, we also measure the throughput of the system. Generally, throughput or transactions per second is inversely proportional to the response time with nominal

changes due to the system environment. In Figure 5.2, we show the throughput for different systems for the financial purpose.

In module two, we test the system with different storage schemes (System A, B, C and D) according to different numbers of users. In this module, we define the different number of users for testing, such as 1, 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100. In this module, we did not consider the purposes as the system performance characteristics for different purposes have a similar pattern which we have shown in Figure 5.1.

In this module, we will determine different system performance factors for different numbers of users and different numbers of data items. For example, in the first test we will measure the response time of the select operation for 9000 data items with different numbers of users. To provide a complete overview of the testing, now we discuss each test configuration:

Test 6 configuration:

- Number of data items : 9000
- Purpose: Shipping
- Number of users : 1,10,20,30,40,50,60,70,80,90,100
- Environment: Two identical Windows system environments
- Test Systems: System A, B, C and D

Test 7 configuration:

- Number of data items : 18000
- Purpose: Shipping
- Number of users : 1,10,20,30,40,50,60,70,80,90,100

- Environment: Two identical Windows system environments
- Test Systems: System A, B, C and D

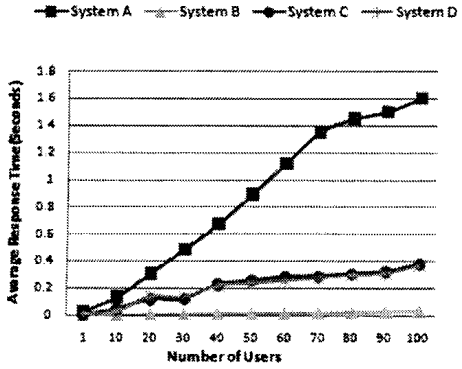
Test 8 configuration:

- Number of data items : 180000
- Purpose: Shipping
- Number of users : 1,10,20,30,40,50,60,70,80,90,100
- Environment: Two identical Windows system environments
- Test Systems: System A, B, C and D

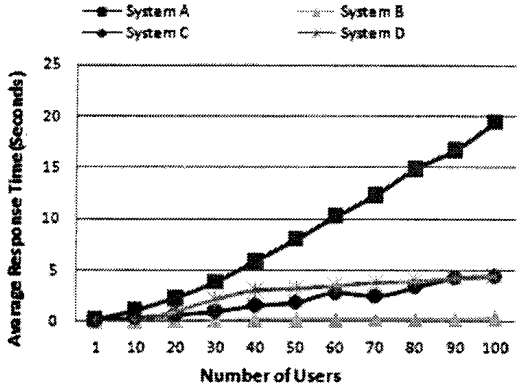
Test 9 configuration:

- Number of data items : 360000
- Purpose: Shipping
- Number of users : 1,10,20,30,40,50,60,70,80,90,100
- Environment: Two identical Windows system environment
- Test Systems: System A, B, C and D

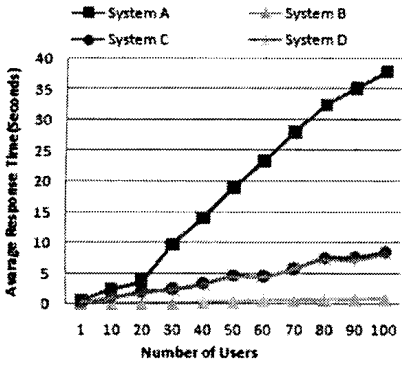
The results for module two testing are shown in Figure 5.3. The graphs show the measured response times for different systems according to different number of users. From these results, it is very clear that the overall response times increases as the number of users increase for each test. The results show similar patterns with each set of data items. As expected, system A shows the worst performance and system B shows the best performance.



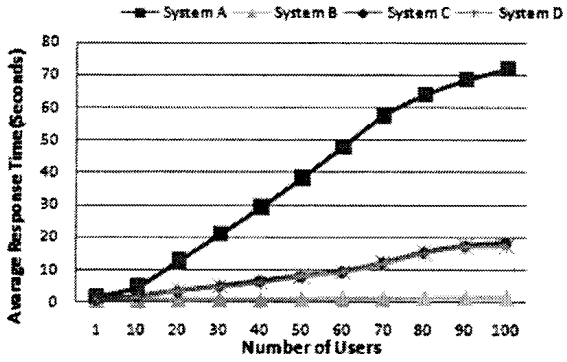
(a) Response time for module two testing 9000 data items



(b) Response time for module two testing 18000 data items



(c) Response time for module two testing 180000 data items



(d) Response time for module two testing 360000 data items

Figure 5.3: Response time for Select for module two testing for different number of user (Purpose: Shipping)

5.2 Performance Analysis of the Insert Operation

In this section, we analyze the performance factors of insert operations for the different systems. As discussed earlier, the insert operations are also tested in two modules. In insert operation testing, privacy purposes are not considered as insert operations of the sample systems have no relations with purposes.

As described, in module one, we test the system performance according to different numbers of data items. The configuration for this test is given below:
Test 10 configuration:

- Number of data items : 18, 360, 900, 1260, 1800 and 3600
- Purpose: N/A
- Number of users : 1
- Environment: Two identical Windows system environment
- Test Systems: System A, B, C and D

Figure 5.4 demonstrates the measured response times for the module one insert test. This graph shows clearly that response times for system A are the lowest as this operation inserts the data into only one data table. In system B, during the insert operation, the system inserts the data into multiple tables according to their purposes. Usually, the insert operations input the data into five different data tables for only one insert operation. Therefore, system B shows the worst performance and highest response time. In the case of systems C and D, the insert operation involves data insertion and permission insertion into different tables. Therefore, the response times for these systems are more than system A. Here, an increase in the number of attributes leads to an increase in the response time. In Figure 5.5, we show the measured systems throughputs for different systems.

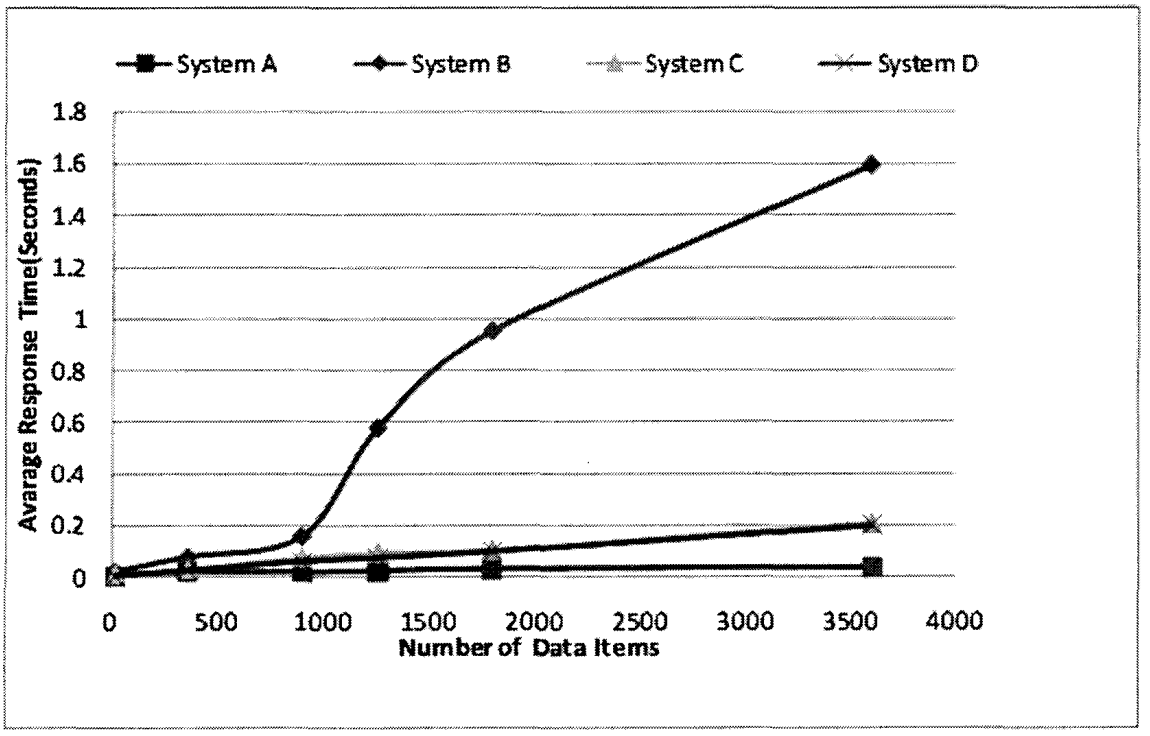


Figure 5.4: Response time of the insert operation

In module 2, we measure the performance factors of insert operations according to different number of users. As discussed for the earlier experiment, we take different numbers of users: 1, 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100. The configurations of module two testing for the insert operation are given below:

Test 11 configuration:

- Number of data items : 18
- Purpose: N/A
- Number of users : 1,10,20,30,40,50,60,70,80,90,100
- Environment: Two identical Windows system environments
- Test Systems: System A, B, C and D

Test 12 configuration:

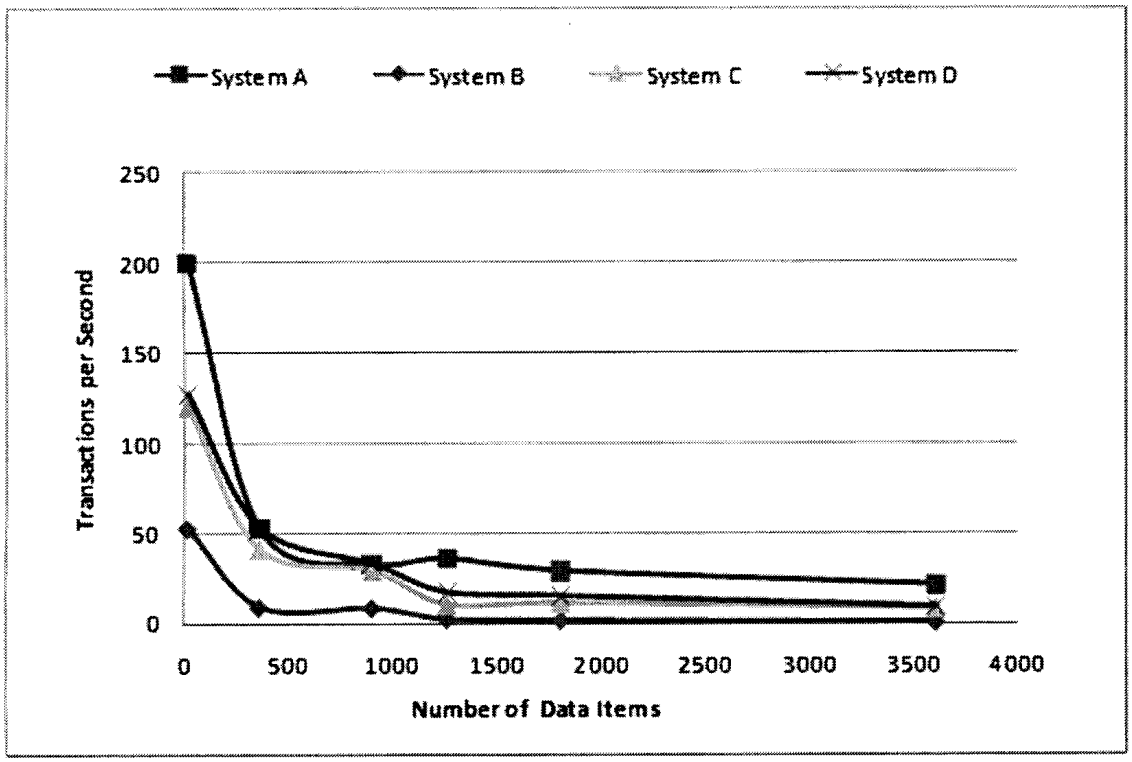


Figure 5.5: Throughput of the insert operation

- Number of data items : 360
- Purpose: N/A
- Number of users : 1,10,20,30,40,50,60,70,80,90,100
- Environment: Two identical Windows system environments
- Test Systems: System A, B, C and D

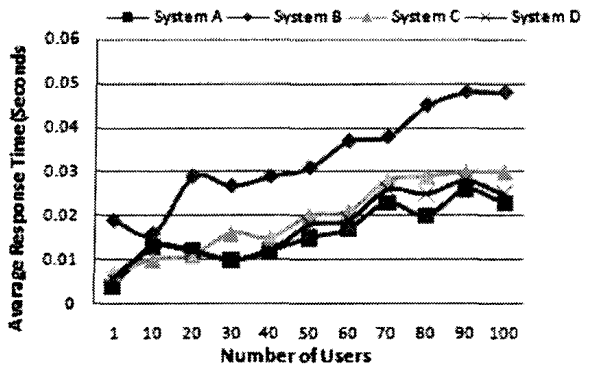
Test 13 configuration:

- Number of data items : 1800
- Purpose: N/A
- Number of users : 1,10,20,30,40,50,60,70,80,90,100
- Environment: Two identical Windows system environments
- Test Systems: System A, B, C and D

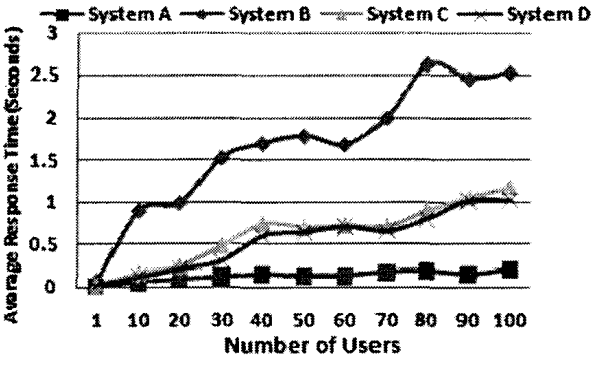
Test 14 configuration:

- Number of data items : 3600
- Purpose: N/A
- Number of users : 1,10,20,30,40,50,60,70,80,90,100
- Environment: Two identical Windows system environments
- Test Systems: System A, B, C and D

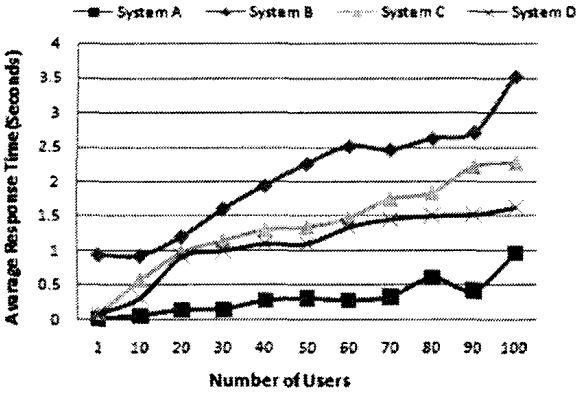
Figure 5.6 represents the results of the module two testing. Here, the overall performance of system A is the best. As per our measured results, system B shows the highest response and so, the performance of this system is lowest. Another definite observation, with the increment of users, the performance decreases which has the effect of high response time.



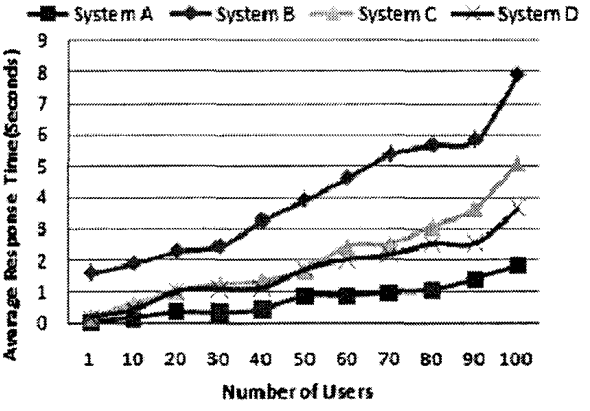
(a) Result of Test 11



(b) Result of Test 12



(c) Result of Test 13



(d) Result of Test 14

Figure 5.6: Response time of module two for insert operation

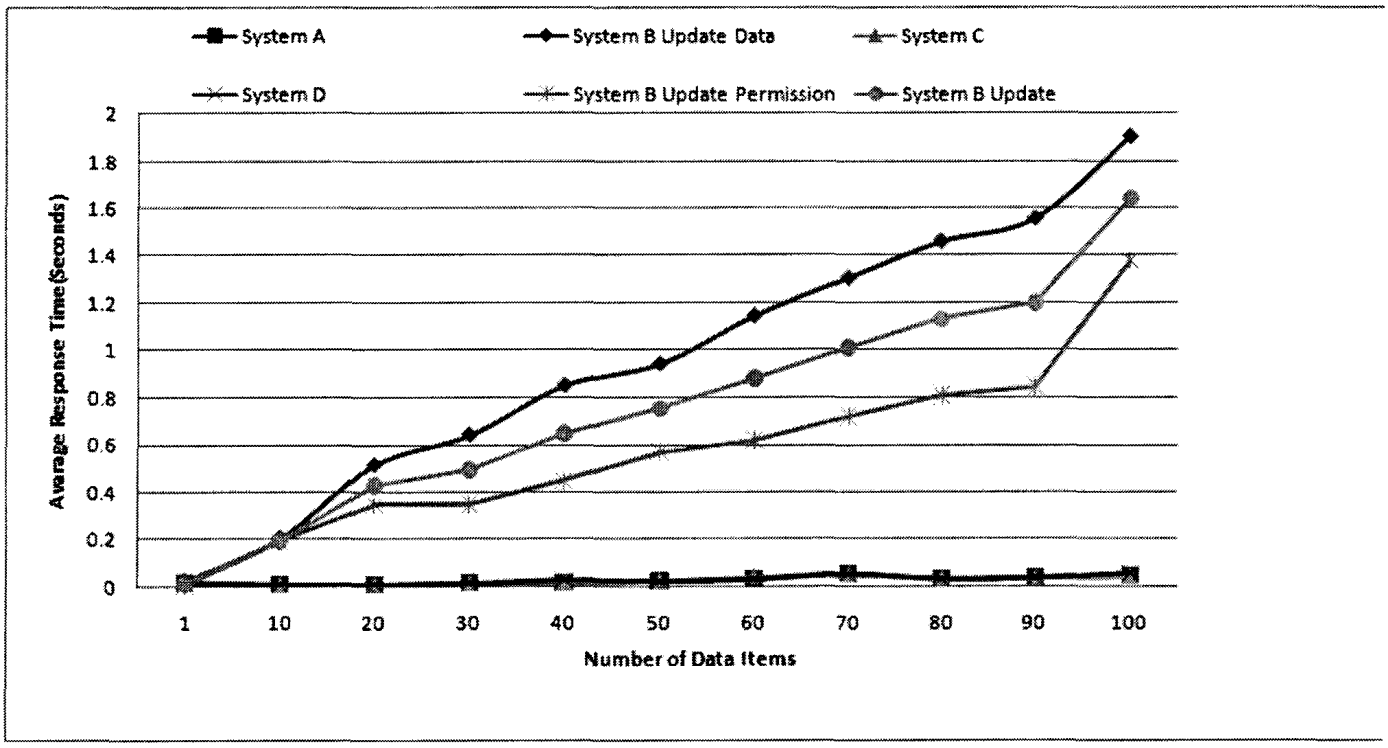


Figure 5.7: Results of the update operations

5.3 Performance Analysis of the Update Operation

In this section, we discuss the performance comparison among each system for update operations. As discussed in Chapter 4, we will test the update operation for different numbers of users for different numbers of data items. The update operations are mostly similar except for system B. In system B, the update operation is divided into two parts: updating the purpose set, and updating data items. In both cases, once the data is updated, the system needs to check the data/ purposes and that leads to updating the data item in multiple tables.

Figure 5.7 visualizes the results of the update operation. Here, we test the update operation for different numbers of users with different numbers of data items. In this performance graph, we show the average response time for different systems.

System Name	SQL operation	Average Response time (Second)
A	Delete	0.001
B	Delete	0.283
C	Delete	0.002
D	Delete	0.004

Table 5.1: Response time of delete operation

The average response time for Systems A, C and D are almost same. Therefore, in Figure 5.7, the response time for Systems A, B and C is superimposed. For system B, the response time for update operations are very high as multiple operations are required for a single update operation here. In this graph, we show response times to update the data items and privacy purposes for system B. Moreover, in this graph, we incorporated the average response time for system B.

5.4 Performance Analysis of the Delete Operation

The deletion process is simpler than inserting, updating or retrieving data. The performance comparison for the delete operation is very simple and straightforward. We perform the delete operation for different numbers of users for different data attributes and measure the response times for each system. From our multiple and independent delete operations testing, we formulate the average response time for delete operations of each system. From Table 5.1, we show the average response time for delete operations:

From Table 5.1, it is completely clear that systems A, C and D require a similar amount of time to perform this operation. But in the case of system B, the response time is high as the system needs to delete the data item from multiple tables.

Chapter 6

Conclusions and Future Directions

This chapter is the concluding chapter of the thesis and it is divided into two sections. In the first section, we will explore some of the major contributions of this thesis. In the second section, we will analyze some future research directions.

6.1 Conclusions

The main objective of this thesis is to analyze the extension of RBAC with privacy labels for various relational database storage schemes. We have studied many different research works to analyze the different storage patterns. In order to understand the extended RBAC system properly, we analyze the system with the basic storage scheme. After analyzing each property and its benefits, we proposed three different storage orientations of the extended RBAC system. To analyze the performance characteristics, we have implemented the extended RBAC with different storage patterns and performed the comparison. The entire test was conducted using the Benchmark factory and the SQL profiler tools. The main types of performance tests used in the experiments are basic SQL transaction tests with diversified load. Moreover, the experiments have been done with different numbers of concurrent users. The results were analyzed against average response time (latency) and throughput (re-

quests per second) of each SQL system in four different RBAC extensions. To verify the correctness of responses, we have conducted these experiments in two different but identical computer systems. The results of experiments have been taken by two database benchmark packages. Moreover, for each test, we have measured the system performance factors after 10 iterations.

Performance tests, for which all results are available in Chapter 5, were divided into four main parts. In each part, we analyze the performance for one basic SQL operation (Insert, Update, Delete and Select). We divided the performance tests in two different modules to measure the system performance accurately. The main focus in those two testing modules is to test each SQL operation's performance according to different numbers of data items and concurrent users.

In our first set of experiments, we test the select SQL operation according to two modules in four different systems. In the case of data retrieval, System A traverses the whole data table and checks each data item with its associated privacy purpose labels. Therefore, System A shows the worst performance in the case of the select operation. In these operational experiments, system B shows the best performance: the system retrieves the data directly from an individual purpose table which is identical to a single select operation. In the case of systems C and D, the performance characteristics of these two systems are almost similar.

In the case of the second group of experiments, our main focus is to analyze the insert operation in four different systems. Here, we followed the same two module test plan to evaluate the performance of these systems according to different users and data items. Here, system B shows the worst performance and System A provides the best performance. In the case of System C, it performs better than system B but worse than System D. Here, for each insertion operation, the system first inserts the data in the data table and then, in the purpose tables. For System D, it shows a little better performance than System C as System D inserts the purpose data items in one purpose table instead of multiple purpose tables.

Update operations in these systems are a little bit different than insert or select operations. Therefore, we design this experiment in a different way, which is discussed in Chapter 4. Here, the update operation for each system is exactly the same except system B. As discussed in Section 4.3., we have done the experiments for the update operation in two parts: update purposes and update data items. Here, for each update operation, the system updates data items/purposes according to data providers' input data items/purposes. As a result of multiple operations, system B provides the worst performance. In the delete operation experiments, the system B shows the worst performance as the system needs to delete the data from multiple data tables. For all other systems, the performance factors are very similar.

All the work of this thesis is concerned with the topic performance characteristics of extended RBAC systems, which have been analyzed from different points of view. There is certainly a major trade-off present concerning the selection of a storage pattern for the extended RBAC system. In the case of the access control system in which the update of customer information is not frequent, System A is the best. System B is the best for the business organizations that do not require data retrieval. In cases C and D, it show average performance factors for each type of transaction. After overall analysis, we think system D provides better performance in term of transaction handling, system operations, and disk space. This thesis is, in fact, very useful for researchers of online business, health care and communication sectors.

6.2 Future Directions

The application developed during this thesis to perform the experiments of extended RBAC with different orientation is quite powerful and flexible. In spite of that, there are some future enhancements which can be done to define the performance characteristics in a more robust way. All conducted experiments in this thesis were based on the privacy purposes as a set. The first extension to consider would be to

have a hierarchy of privacy purposes where there are implications according to the hierarchy. The data storage schemes performance characteristics could be evaluated for this system. Having privacy purposes in a hierarchy gives a more powerful too to companies.

The customer information can be stored in different data tables instead of one table. In our research, we consider the customer information which is stored only in one table. In the future, we incorporate the idea of multiple table of customer information. Due to this factor, our proposed approaches may work differently.

In this thesis, the performance analysis is done only for the SQL server 2005 database engine. As an extension of this work, we can perform the same experiments on other database engines like Oracle, DB2, or Mysql. In this way, the performance characteristics can be determined in a generic way. These experiments would also provide the performance characteristics and significant evolution point for each database engine. This way the SQL operation could be optimized in different database environments.

Tests conducted within this thesis, were not aimed to compare system performance according to different operating systems environment. All of them were carried out under the Windows operating system. It is possible that the system will perform differently under other operating systems. This can be the subject of another research focused on this issue.

This extended RBAC application could be extended also with new hardware infrastructures. At the moment we conducted our experiments using only one computer system. It would be very useful also to have new hardware with multi-core processors and more expandable memory, to analyze the system in an industrial environment.

Finally, in this thesis, we have introduced some ideas of different storage schemes for RBAC extension. However, with the change of technology, this area can be developed and enhanced. Therefore, this idea can still be taken into consideration for future enhancement.

Bibliography

[1] Benchmark Factory for Databases. http://www.quest.com/benchmark-factory/, last accessed August, 2011.

[2] Introducing SQL Server Profiler. http://msdn.microsoft.com/en-us/library/ms181091.aspx, last accessed August, 2011.

[3] Organisation for Economic Co-operation and Development. OECD guidelines on the protection of privacy and transborder flows of personal data of 1980. http://www.oecd.org, last accessed August, 2011.

[4] P3P tool box. http://www.p3ptoolbox.org/guide/section2.shtml, last accessed August, 2011.

[5] Samples and Sample Databases. http://technet.microsoft.com/en-us/library/ms124501(SQL.90).aspx, last accessed September, 2011.

[6] United State Department of Health. Health Insurance Portability and Accountability.Act of 1996. http://www.hhs.gov/ocr/hipaa, last accessed July, 2011.

[7] What is 3-tier architecture? http://wikipedia.org/wiki/Multitier_architecture, last accessed October, 2011.

[8] Milan , Petkovic and Willem Jonker. *Enhanced Privacy for Digital Rights Management, Security, Privacy and Trust in Modern Data Management (Data-*

Centric Systems and Applications). Springer-Verlag New York, Inc., NJ, USA, 2007.

[9] Ilsoo Ahn and Richard Snodgrass. Performance evaluation of a temporal database management system. *SIGMOD Rec.*, 15:96–107, June 1986.

[10] Aiman Lafi Al-Harbi. The role graph model and privacy. Master's thesis, The University of Western Ontario, 2010.

[11] Aiman Lafi Al-Harbi and Sylvia L. Osborn. Mixing privacy with role-based access control. In *C3S2E*, pages 1–7, 2011.

[12] ANSI/INCITS. Role based access control. ANSI/INCITS 359-2004, 2004.

[13] R. W. Baldwin. Naming and Grouping Privileges to Simplify Security Management in Large Databases. In *IEEE Symposium on Security and Privacy*, pages 116–132, 1990.

[14] D E Bell and Leonard J LaPadula. Secure Computer Systems: Mathematical Foundations and Model. *MITRE CORP BEDFORD MA*, 1(M74-244):42, 1973.

[15] Elisa Bertino, Piero Andrea Bonatti, and Elena Ferrari. TRBAC: A temporal role-based access control model. *ACM Trans. Inf. Syst. Secur.*, 4:191–233, August 2001.

[16] Ji-Won Byun and Ninghui Li. Purpose based access control for privacy protection in relational database systems. *The VLDB Journal*, 17:603–619, July 2008.

[17] D.T. Campbell and J.C. Stanley. *Experimental and Quasi-Experimental Designs for Research.* Rand McNally & Company, 1970.

[18] Silvana Castano, Maria Grazia Fugini, Giancarlo Martella, and Pierangela Samarati. *Database Security.* Addison-Wesley & ACM Press, 1995.

[19] David D. Clark and David R. Wilson. A comparison of commercial and military computer security policies. *Security and Privacy, IEEE Symposium*, 0:180–184, 1987.

[20] M. A. C. Dekker, S. Etalle, and J. Den Hartog. *Privacy Policies, Security, Privacy and Trust in Modern Data Management (Data-Centric Systems and Applications).* Springer-Verlag New York, Inc., NJ, USA, 2007.

[21] Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems, Fourth Edition.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[22] David Ferraiolo and Richard Kuhn. Role-Based Access Control. In *15th NIST-NCSC National Computer Security Conference*, pages 554–563, 1992.

[23] D.Richard Kuhn David F. Ferraiolo. Role-Based Access Control (RBAC): Features and Motivations. In *Proc. of the Annual Computer Security Applications Conf., New Orleans: IEEE Press.*

[24] Shashi K. Gadia. A homogeneous relational model and query languages for temporal databases. *ACM Trans. Database Syst.*, 13(4):418–448, 1988.

[25] Shashi K. Gadia and Chuen-Sing Yeung. A generalized model for a relational temporal database. In *SIGMOD Conference*, pages 251–259, 1988.

[26] Iqbal A. Goralwalla, Abdullah Uz Tansel, and M. Tamer Özsu. Experimenting with temporal relational databases. In *CIKM*, pages 296–303, 1995.

[27] Heiko Klarl, Korbinian Molitorisz, Christian Emig, Karsten Klinger, and Sebastian Abeck. Extending role-based access control for business usage. In *SECURWARE*, pages 136–141, 2009.

[28] Boris Kostenko. Temporal preprocessor: Towards temporal applications development. In *SYRCoDIS*, 2007.

[29] Butler W. Lampson. Protection. *SIGOPS Oper. Syst. Rev.*, 8:18–24, January 1974.

[30] Frederick H. Lochovsky and Carson C. Woo. Role-based security in data base management systems. In *DBSec*, pages 209–222, 1987.

[31] Sylvia L.Osborn. *Role-Based Access Control, Security, Privacy and Trust in Modern Data Management (Data-Centric Systems and Applications).* Springer-Verlag New York, Inc., NJ, USA, 2007.

[32] Qun Ni, Dan Lin, Elisa Bertino, and Jorge Lobo. Conditional privacy-aware role based access control. In *ESORICS*, pages 72–89, 2007.

[33] Qun Ni, Alberto Trombetta, Elisa Bertino, and Jorge Lobo. Privacy-aware role based access control. In *Proceedings of the 12th ACM symposium on Access control models and technologies*, SACMAT '07, pages 41–50, New York, NY, USA, 2007. ACM.

[34] Matunda Nyanchama and Sylvia L. Osborn. Access rights administration in role-based security systems. In *DBSec*, pages 37–56, 1994.

[35] Matunda Nyanchama and Sylvia L. Osborn. The role graph model and conflict of interest. *ACM Trans. Inf. Syst. Secur.*, 2(1):3–33, 1999.

[36] M. Tamer Özsu and Patrick Valduriez. *Principles of Distributed Database Systems, Second Edition.* Prentice-Hall, 1999.

[37] Pierangela Samarati and Sabrina De Capitani di Vimercati. Access control: Policies, models, and mechanisms. In *Revised versions of lectures given during the IFIP WG 1.7 International School on Foundations of Security Analysis and Design on Foundations of Security Analysis and Design: Tutorial Lectures*, FOSAD '00, pages 137–196, London, UK, 2001. Springer-Verlag.

[38] Ravi Sandhu, David Ferraiolo, and Richard Kuhn. The nist model for role-based access control: towards a unified standard. In *Proceedings of the fifth ACM workshop on Role-based access control*, RBAC '00, pages 47–63, New York, NY, USA, 2000. ACM.

[39] Ravi S. Sandhu. Lattice-based access control models. *Computer*, 26:9–19, November 1993.

[40] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *Computer*, 29:38–47, February 1996.

[41] Arie Segev and Arie Shoshani. Functionality of temporal data models and physical design implications. *IEEE Data Eng. Bull.*, 11(4):38–45, 1988.

[42] Richard T. Snodgrass. *Developing Time-Oriented Database Applications in SQL.* Morgan Kaufmann, 1999.

[43] M N Tahir. C-RBAC: Contextual role-based access control model. *Ubiquitous Computing And Communication Journal*, 2, 2007.

[44] Abdullah Uz Tansel, James Clifford, Shashi K. Gadia, Sushil Jajodia, Arie Segev, and Richard T. Snodgrass, editors. *Temporal Databases: Theory, Design, and Implementation.* Benjamin/Cummings, 1993.

[45] Jacques Wainer, Paulo Barthelmess, and Akhil Kumar. W-RBAC - a workflow security model incorporating controlled overriding of constraints. *Int. J. Cooperative Inf. Syst.*, 12(4):455–485, 2003.

Appendix A

Experiments for Select Operation

Test Case No	User load	Description	Purpose
1	1	Select 9000 data items from System B Customer Information table	Marketing
2	1	Select 18000 data items from System B Customer Information table	Marketing
3	1	Select 36000 data items from System B Customer Information table	Marketing
4	1	Select 90000 data items from System B Customer Information table	Marketing
5	1	Select 180000 data items from System B Customer Information table	Marketing
6	1	Select 360000 data items from System B Customer Information table	Marketing
7	1	Select 9000 data items from System B Customer Information table	Finance
8	1	Select 18000 data items from System B Customer Information table	Finance
9	1	Select 36000 data items from System B Customer Information table	Finance
10	1	Select 90000 data items from System B Customer Information table	Finance
11	1	Select 180000 data items from System B Customer Information table	Finance
12	1	Select 360000 data items from System B Customer Information table	Finance
13	1	Select 9000 data items from System B Customer Information table	Purchase
14	1	Select 18000 data items from System B Customer Information table	Purchase
15	1	Select 36000 data items from System B Customer Information table	Purchase
16	1	Select 90000 data items from System B Customer Information table	Purchase
17	1	Select 180000 data items from System B Customer Information table	Purchase
18	1	Select 360000 data items from System B Customer Information table	Purchase
19	1	Select 9000 data items from System B Customer Information table	Bdministration
20	1	Select 18000 data items from System B Customer Information table	Bdministration
21	1	Select 36000 data items from System B Customer Information table	Bdministration
22	1	Select 90000 data items from System B Customer Information table	Bdministration
23	1	Select 180000 data items from System B Customer Information table	Bdministration
24	1	Select 360000 data items from System B Customer Information table	Bdministration
25	1	Select 9000 data items from System B Customer Information table	Shipping
26	1	Select 18000 data items from System B Customer Information table	Shipping
27	1	Select 36000 data items from System B Customer Information table	Shipping
28	1	Select 90000 data items from System B Customer Information table	Shipping
29	1	Select 180000 data items from System B Customer Information table	Shipping
30	1	Select 360000 data items from System B Customer Information table	Shipping

Table A.1: Module one test cases for select operation in system B

Test Case No	User load	Description	Purpose
1	1	Select 9000 data items from System C Customer Information table	Marketing
2	1	Select 18000 data items from System C Customer Information table	Marketing
3	1	Select 36000 data items from System C Customer Information table	Marketing
4	1	Select 90000 data items from System C Customer Information table	Marketing
5	1	Select 180000 data items from System C Customer Information table	Marketing
6	1	Select 360000 data items from System C Customer Information table	Marketing
7	1	Select 9000 data items from System C Customer Information table	Finance
8	1	Select 18000 data items from System C Customer Information table	Finance
9	1	Select 36000 data items from System C Customer Information table	Finance
10	1	Select 90000 data items from System C Customer Information table	Finance
11	1	Select 180000 data items from System C Customer Information table	Finance
12	1	Select 360000 data items from System C Customer Information table	Finance
13	1	Select 9000 data items from System C Customer Information table	Purchase
14	1	Select 18000 data items from System C Customer Information table	Purchase
15	1	Select 36000 data items from System C Customer Information table	Purchase
16	1	Select 90000 data items from System C Customer Information table	Purchase
17	1	Select 180000 data items from System C Customer Information table	Purchase
18	1	Select 360000 data items from System C Customer Information table	Purchase
19	1	Select 9000 data items from System C Customer Information table	Cdministration
20	1	Select 18000 data items from System C Customer Information table	Cdministration
21	1	Select 36000 data items from System C Customer Information table	Cdministration
22	1	Select 90000 data items from System C Customer Information table	Cdministration
23	1	Select 180000 data items from System C Customer Information table	Cdministration
24	1	Select 360000 data items from System C Customer Information table	Cdministration
25	1	Select 9000 data items from System C Customer Information table	Shipping
26	1	Select 18000 data items from System C Customer Information table	Shipping
27	1	Select 36000 data items from System C Customer Information table	Shipping
28	1	Select 90000 data items from System C Customer Information table	Shipping
29	1	Select 180000 data items from System C Customer Information table	Shipping
30	1	Select 360000 data items from System C Customer Information table	Shipping

Table A.2: Module one test cases for select operation in system C

Test Case No	User load	Description	Purpose
1	1	Select 9000 data items from System D Customer Information table	Marketing
2	1	Select 18000 data items from System D Customer Information table	Marketing
3	1	Select 36000 data items from System D Customer Information table	Marketing
4	1	Select 90000 data items from System D Customer Information table	Marketing
5	1	Select 180000 data items from System D Customer Information table	Marketing
6	1	Select 360000 data items from System D Customer Information table	Marketing
7	1	Select 9000 data items from System D Customer Information table	Finance
8	1	Select 18000 data items from System D Customer Information table	Finance
9	1	Select 36000 data items from System D Customer Information table	Finance
10	1	Select 90000 data items from System D Customer Information table	Finance
11	1	Select 180000 data items from System D Customer Information table	Finance
12	1	Select 360000 data items from System D Customer Information table	Finance
13	1	Select 9000 data items from System D Customer Information table	Purchase
14	1	Select 18000 data items from System D Customer Information table	Purchase
15	1	Select 36000 data items from System D Customer Information table	Purchase
16	1	Select 90000 data items from System D Customer Information table	Purchase
17	1	Select 180000 data items from System D Customer Information table	Purchase
18	1	Select 360000 data items from System D Customer Information table	Purchase
19	1	Select 9000 data items from System D Customer Information table	Cdministration
20	1	Select 18000 data items from System D Customer Information table	Cdministration
21	1	Select 36000 data items from System D Customer Information table	Cdministration
22	1	Select 90000 data items from System D Customer Information table	Cdministration
23	1	Select 180000 data items from System D Customer Information table	Cdministration
24	1	Select 360000 data items from System D Customer Information table	Cdministration
25	1	Select 9000 data items from System D Customer Information table	Shipping
26	1	Select 18000 data items from System D Customer Information table	Shipping
27	1	Select 36000 data items from System D Customer Information table	Shipping
28	1	Select 90000 data items from System D Customer Information table	Shipping
29	1	Select 180000 data items from System D Customer Information table	Shipping
30	1	Select 360000 data items from System D Customer Information table	Shipping

Table A.3: Module one test cases for select operation in system D

Appendix B

Experiments for Insert Operation

Test Case No	User load	Description
1	1	Insert 18 data items into System B Customer Information table
2	1	Insert 360 data items into System B Customer Information table
3	1	Insert 900 data items into System B Customer Information table
4	1	Insert 1260 data items into System B Customer Information table
5	1	Insert 1800 data items into System B Customer Information table
6	1	Insert 2700 data items into System B Customer Information table
7	1	Insert 3600 data items into System B Customer Information table

Table B.1: Module one test results (Response Time) for insert operation in system B

Test Case No	User load	Description
1	1	Insert 18 data items into System C Customer Information table
2	1	Insert 360 data items into System C Customer Information table
3	1	Insert 900 data items into System C Customer Information table
4	1	Insert 1260 data items into System C Customer Information table
5	1	Insert 1800 data items into System C Customer Information table
6	1	Insert 2700 data items into System C Customer Information table
7	1	Insert 3600 data items into System C Customer Information table

Table B.2: Module one test cases for insert operation in system C

Test Case No	User load	Description
1	1	Insert 18 data items into System D Customer Information table
2	1	Insert 360 data items into System D Customer Information table
3	1	Insert 900 data items into System D Customer Information table
4	1	Insert 1260 data items into System D Customer Information table
5	1	Insert 1800 data items into System D Customer Information table
6	1	Insert 2700 data items into System D Customer Information table
7	1	Insert 3600 data items into System D Customer Information table

Table B.3: Module one test cases for insert operation in system D

Appendix C

Experimental Results for Different Iterations

Number of data items	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5
9000	0.0279	0.0288	0.0250	0.0279	0.0275
18000	0.0580	0.0566	0.0570	0.0568	0.0567
36000	0.1399	0.1381	0.1382	0.1354	0.1396
90000	0.1825	0.1873	0.1881	0.1881	0.1801
180000	0.6933	0.7016	0.7084	0.7049	0.6968
360000	1.8199	1.8092	1.8143	1.8122	1.7890

Table C.1: Response Time of Select Operation for 5 iterations (System A)

Number of data items	Iteration 6	Iteration 7	Iteration 8	Iteration 9	Iteration 10
9000	0.0273	0.0265	0.0288	0.0275	0.0256
18000	0.0579	0.0572	0.0568	0.0571	0.0561
36000	0.1331	0.1388	0.1349	0.1354	0.1320
90000	0.1823	0.1867	0.1891	0.1816	0.1830
180000	0.7031	0.7094	0.7040	0.6973	0.7059
360000	1.8282	1.8255	1.8287	1.7898	1.8118

Table C.2: Response Time of Select Operation for 5 iterations (System A)

Number of Users	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5
1	0.0233	0.0237	0.0242	0.0234	0.0244
10	0.1719	0.1716	0.1730	0.1717	0.1716
20	0.2481	0.2490	0.2432	0.2461	0.2453
30	0.4934	0.4939	0.4933	0.4938	0.4941
40	0.7330	0.7397	0.7481	0.7430	0.7449
50	0.7046	0.7039	0.7036	0.7038	0.7041
60	0.7258	0.7263	0.7269	0.7241	0.7247
70	0.7224	0.7205	0.7168	0.7153	0.7105
80	0.9006	0.9005	0.9031	0.9042	0.9045
90	1.0572	1.0679	1.0615	1.0571	1.0612
100	1.1620	1.1486	1.1888	1.1751	1.1967

Table C.3: Response Time of Insert Operation for 5 iterations (System B)

Number of Users	Iteration 6	Iteration 7	Iteration 8	Iteration 9	Iteration 10
1	0.0244	0.0231	0.0238	0.0238	0.0230
10	0.1717	0.1718	0.1722	0.1725	0.1724
20	0.2462	0.2459	0.2444	0.2447	0.2444
30	0.4941	0.4944	0.4937	0.4939	0.4941
40	0.7314	0.7367	0.7370	0.7464	0.7326
50	0.7043	0.7047	0.7038	0.7049	0.7043
60	0.7258	0.7267	0.7270	0.7257	0.7247
70	0.7154	0.7212	0.7125	0.7120	0.7112
80	0.9002	0.9026	0.9028	0.9010	0.9039
90	1.0651	1.0653	1.0613	1.0658	1.0583
100	1.1439	1.1403	1.1251	1.1998	1.1772

Table C.4: Response Time of Insert Operation for 5 iterations (System B)