Analytical Study for Measuring the Effects of Database Segregation Multi-tenancy Approaches on Cost and Performance in Cloud Computing

دراسة تحليلية لقياس تأثير مبادئ خاصية تعدد المستأجرين في قواعد البيانات الانفصالية على التكلفة والأداء في الحوسبة السحابية

By

Ala K. I. Dawoud

Supervisor

Prof. Ahmed Kayed

This thesis is submitted to the Department of Computer Information Systems, Faculty of Information Technology, Middle East University in partial fulfillment of the Requirements for Master Degree in Computer Information Systems.

Department of Computer Information Systems

Faculty of Information Technology

Middle East University

December, 2015

Authorization Statement

I, Ala Kamal Ibrahim Dawoud, authorize the Middle East University to supply a copy of my thesis to libraries, establishments or individuals upon their request.

Name : Ala Kamal Ibrahim Dawoud

Date : 2015/12/7

Signature: .....

اقرار تفويض

أنا علاء كمال ابراهيم داوود، أفوض جامعة الشرق الاوسط للدراسات العليا بتزويد نسخ من رسالتي ورقيا و الكترونيا للمكتبات أو المنظمات أو الهيئات و المؤسسات المعنية بالابحاث والدراسات العلمية عند طلبها.

الاسم : علاء كمال ابراهيم داوود

التاريخ : 2015/12/7

التوقيع : .....

علاء داوود

Examination Committee Decision

This is to certify that the thesis entitled "Analytical Study for Measuring the Effects of Database Segregation Multi-tenancy Approaches on Cost, Security, and Performance in Cloud Computing " was successfully defended and approved on 7th December 2015.

Examination Committee Members	Signature
(Head of the Committee)	
Dr. Ahmad Abo Shareeha	
Department of Computer Science	.....
Middle East University	.....
(Supervisor)	
Dr. Ahmad Kayed.	
~~Associate~~ Professor	.....
Dean Faculty of IT	.....
(External Committee Member)	
Dr. Emad Abo Shanab	
Associate Professor	.....
Yarmouk University	
.....	

Acknowledgment

I would like to express my sincere appreciation to Prof. Ahmad Kayed for his guidance, support and motivation throughout my Master's Thesis.

Dedication

- To my parents for their support.
- To my brothers for support and patience me.
- To my wife and children in my family.

Analytical Study for Measuring the Effects of Database Segregation Multi-tenancy Approaches on Cost, Security, and Performance in Cloud Computing

Prepared By:

Ala K. I. Dawoud

Supervised By:

Prof. Ahmad Kayed

ABSTRACT

Software as a Service (SaaS) service model reduces software costs and provides efficient use of applications in the internet environments. Multitenancy is one of the most important features in cloud computing that is responsible to provide software applications for tenants in cloud computing environment. Thus, several database segregations techniques (approaches) are available to be used in multitenant environment. Each approach of database segregation has its own effect on the performance, cost, and security depending on the types of query that is executed.

The main aim of this thesis is to study the performance, cost, and security of the database segregation depending on the utilized database segregation approach. To be able to meet this aim it had been outlined different types of database segregations, database sizes, and queries. The findings of literature review have been deployed to design several experiments for this research. Three experiments have been conducted to evaluate the response time database transaction in term of performance, evaluate the CPU's statistics and disk storage statistics in term of cost, and evaluate the effects of increasing and decreasing the security by manipulating the number of encryption over a database tables in term of cost and performance (transaction time).
The empirical findings in this study showed that selecting shared database shared schema approach give the system the chance to gain 42 % compared with selecting isolated database approach, and 71% compared with selecting shared database with separate schema from performance (transaction time) perspective. However, selecting isolated database approach was given the system a chance to gain 33% compared with selecting B, and 69% compared with selecting shared database shared schema approach from CPU cost perspective. Furthermore, selecting shared database with separate schema approach was given the system chance to gain 37% compared with selecting isolated database approach, and 57% compared with selecting shared database with shared schema approach from security perspective.

Keywords: Cloud Computing, DB segregation techniques, Isolated DB, Shared Schema Shared DB, Separate Schema Shared DB.

دراسة تحليلية لقياس تأثير مبادئ خاصية تعدد المستأجرين في قواعد البيانات الانفصالية على التكلفة والأمان والأداء في الحوسبة السحابية

إعداد : علاء كمال ابراهيم داوود

إشراف: د. أحمد كايد

الملخص

إن نموذج خدمة البرمجيات (SaaS) يقلل من تكاليف البرامج ويوفر كفاءة استخدام التطبيقات في بيئات الإنترنت . خاصية تعدد المستأجرين هي واحدة من أهم سمات في الحوسبة السحابية فهي المسؤولة عن توفير التطبيقات البرمجية للمستأجرين في بيئة الحوسبة السحابية. ولذا فإن تقنيات (نهج) متعددة لقواعد البيانات الانفصالية متوفرة للإستخدام في بيئة تعدد المستأجرين. وكل نهج في بيئة قواعد البيانات الانفصالية له تأثيره الخاص على الأداء ، التكلفة والأمن معتمدة على نوع الاستعلام الذي يتم تنفيذه.

الهدف الرئيسي من هذا البحث هو دراسة الأداء والتكلفة والأمن في قاعدة البيانات الانفصالية معتمدة على نهج فصل قاعدة البيانات المستخدمة . ولكي تكون قادر على تحقيق هذا الهدف تم الاعتماد على استخدام قواعد بيانات انفصالية مختلفة، أحجام قواعد البيانات والإستعلامات . وقد تم نشر نتائج دراسات سابقة لتصميم العديد من التجارب لهذا البحث. وقد أجريت ثلاث تجارب لتقييم ازمنة استجابة قاعدة بيانات لايجاد الأداء ، وتقييم وحدة المعالجة المركزية ومقدار التخزين على القرص لإيجاد التكلفة، وتقييم الآثار المترتبة على زيادة أو خفض خصوصية البيانات عن طريق زيادة أو تقليل عدد مرات تشفير الجداول في قاعدة البيانات لايجاد التكلفة والأداء.

أظهرت التجارب المقترحة في هذه الدراسة أن اختيار مبدأ (Shared DB Shared Schema) في قواعد البيانات قدم للنظام فرصة لربح 42% مقارنة مع اختيار مبدأ (Isolated DB)، و ربح 71% مقارنة مع اختيار (Shared DB Separate Schema) من وجهة نظر الاداء. على الرغم من ذلك أظهرت نتائج التجارب المقترحة أيضا أن اختيار مبدأ (Isolated DB) تتيح للنظام لكسب 33% مقارنة مع استخدام (Shared DB Separate Schema) و بمقدار 69% مقارنة مع (Shared DB Shared Schema) من وجهة نظر التكلفة. علاوة على ذلك كان اختيار (Shared DB Separate Schema) له تأثير ايجابي على النظام لكسب 37% مقارنة مع اختيار مبدأ (Isolated DB) و بمقدار 57% مقارنة مع اختيار (Shared DB Shared Schema) من وجهة نظر تكلفة الامن والحماية لبيانات النظام.

الكلمات المفتاحية: الحوسبة السحابية، تقنيات فصل قواعد البيانات، قواعد البيانات المنفصلة، المخطط وقةواعد البيانات المشتركة، قواعد البيانات المشتركة والمخطط المنفصل.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Meaning
CPU	Central Processing Unit
CPUS	CPU Speed
Data	Amount of Data on Ram
Iaas	Infrastructure as s Service
Paas	Platform as a Service
Saas	Software as a Service
Ram	Random Access Memory
Rams	Ram Size
VM	Virtual Machine
S	Seconds
NIST	National Institute of Standards and Technology
CSP	Cloud Service Provider
APIs	Access Point Interfaces
Haas	Hardware as a Service
OS	Operating System
ITSI	between Isolated Table Shared Instance
STSI	Shared Table Shared Instance
EET	Elastic Extension Table
RDBMS	Relational Data Management System
SQL	Structured Query Language
AES	Advanced Encryption Standard
RSA	Rivest, Shamir and Adelman
XML	Extended Markup Language
I/O	Input Output
ITSI	Isolated Table Shared Instance
STSI	Shared Table Shared Instance
SQLVM	SQL Virtual Machine
GUI	Graphical User Interface
ACL	Access Control List
DES	Data Encryption Standard
CV	Coefficient Variation

CHAPTER ONE

Introduction

1.1. Overview

This chapter explains background about cloud computing, multitenancy, and database segregation techniques. This chapter shows the problem statement of this research, author's contribution, and the outline of thesis chapters.

1.2. Cloud Computing

The use of cloud computing is increasing rapidly. Therefore, many researches showed this technology as a driving force for small, medium, and large sized companies. The most widely used definition of cloud computing is found according to the National Institute of Standards and Technology (NIST) as "*Cloud computing is a model for enabling convenient, on demand network access to a shared pool of configurable computing resources (e.g. network, servers, storage, applications, and services). That can be rapidly provisioned and released with a minimal management efforts or service provider interaction* " (Mell, P., and Grance, T., 2011).

Cloud computing technology follows general layered architecture. Figure 1.1 shows the service models used in cloud computing architecture.

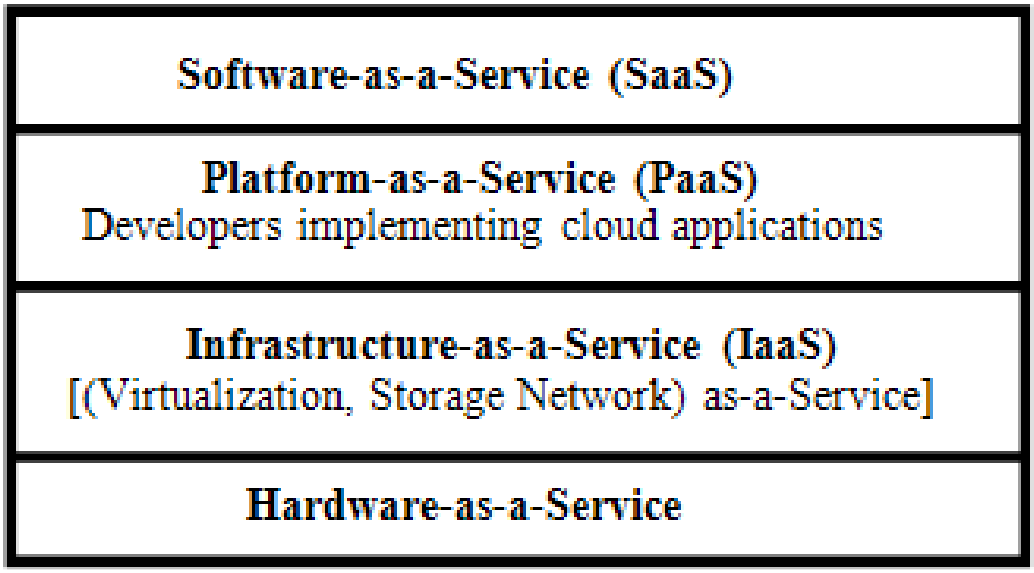

Figure 1. 1. The general layered architecture of cloud computing paradigm (Rimal, B. et al., 2009)

The architecture of cloud computing paradigm consisted into four service models (Abualkibash, M. and Elliethey, K., 2014):

- Infrastructure as a Service (IaaS), this service model is able to provide a low level service such as Virtual Machines (VMs) that uses customers Operating System (OS) images. As well as, the ability to access storage devices from several VMs is another example on this service model.

- Platform as a Service (PaaS), the Cloud Service Provider (CSP) using this model offers Access Point Interfaces (APIs) that are used by customers to develop applications.

- Software as a Service (SaaS), which is used by end users to interact with complete software products as a web based service.

- Hardware as a Service (HaaS), this service model provides the needed hardware to build data centers for any organization. Hence, it offers a reducing in the cost of setting up IT resources.

1.3. Multi-tenancy

In cloud computing, SaaS is represented as a software delivery model. Thus, users can access the available software remotely. In this context, the multi-tenancy feature is one of the most important features that are provided by cloud computing (Sarasathi, M. and Bhuvaneswari, T., 2013). Multi-tenancy feature brings many advantages such as reducing the operational costs by splitting hardware and software resources via sharing them among different tenants, and simplifying maintenance and management efforts. Hence, multi-tenancy brings many benefits for end users such as reducing applications costs, and give the chance to use it from small and medium business enterprises (Sarasathi, M. and Bhuvaneswari, T., 2013).

The architecture of multi-tenancy provide multiple types of models such as shared nothing model, shared hardware model, shared Operating System (OS) model, shared database model, shared everything model, and custom multi-tenancy model (Youssef, E., 2012).

1.4. Database Segregation Approaches

Banvile, R. pointed out the common three different types of database multi-tenancy. Hence, these types are isolated database, separate schema with shared application, and shared schema with shared application (Banville, R., 2014).

Isolated Database (Separate Database)

In isolated database, each tenant has its own application instance, database, and infrastructure. In Infrastructure tenancy, each tenant has its own application, and database instance that have the same underlying infrastructure.

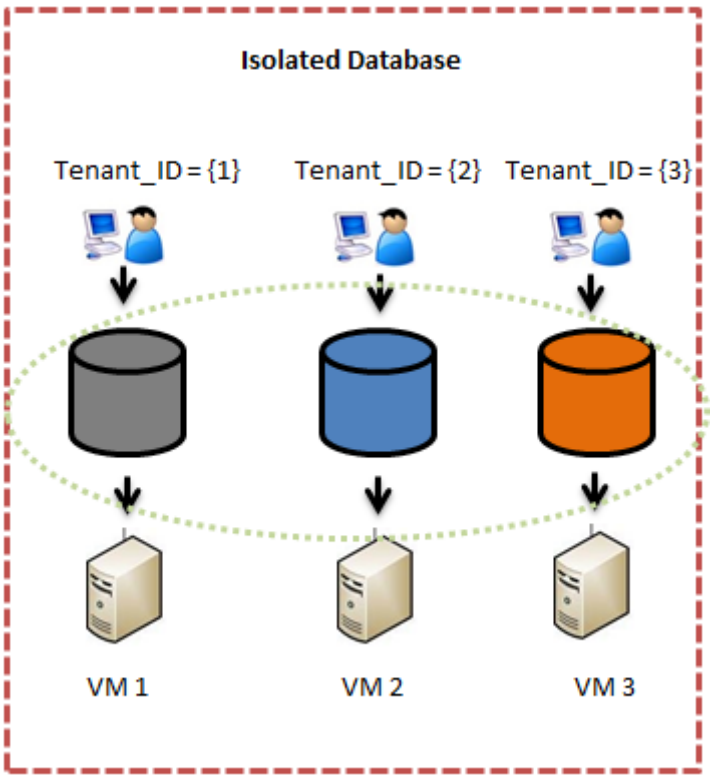

Figure 1. 2. Isolated Database approach (Kun, M. et al., 2012).

Shared database with separate schema

In shared database with separate schema approach, it involves multiple tenants that are working in the same database instance. Hence, each tenant has its own tables that are grouped into schemes. Each database scheme is designed specifically for each tenant

(Kun, M. et al., 2012). Figure 1.3 shows illustration of shared database with separate schema approach.

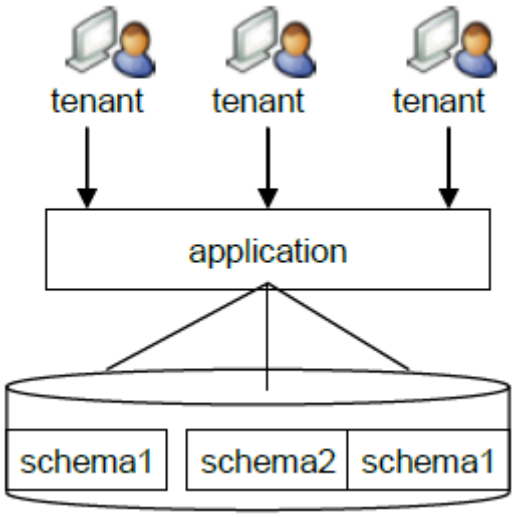

Figure 1. 3. Shared database with separate schema approach (Kun, M. et al., 2012).

Kun, M. et al. explained the weaknesses of shared database with separate schema suffers from different weak points such as the cost of maintaining equipment's, backing up tenant's data, restoring data in the event of failure, and the total number of tenants that can be housed on a given database server is limited by the number of schemes that server can create (Kun, M. et al., 2012).

Shared database with shared schema.

In shared database with shared schema approach, it uses the same database instance and the same set of tables to host multiple tenant's data. Furthermore, a new attribute will be added in each table which represent the tenant's ID in order to connect tenant with its particular records. In this context, this approach has reduced hardware and backup costs (Kun, M. et al., 2012). Figure 1.4 shows an illustration for shared database with shared schema approach.

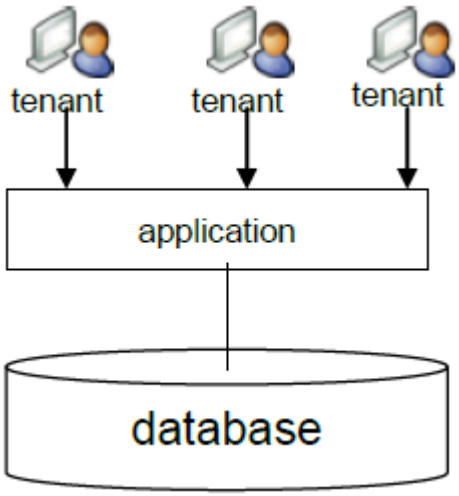

Figure 1. 4. Shared database with shared schema approach (Kun, M. et al., 2012).

1.5. Problem Statement

Implementing multi-tenancy over database is one of the challenges that face CSP for many reasons such as performance, security, isolation and more. Furthermore, finding the suitable approach is another challenge for tenants beside their diversity of requirements that are needed in their enterprises. This research concentrated on finding the impacts of database segregation approaches on the performance, cost, and security of tenants in the cloud. This research revealed a possibility to compare the database segregation approaches based on the performance of tenant's systems, the security of tenant's data, and the cost. Furthermore, finding the critical point of database segregation approach which depends on the number of tenants is another challenge.

1.6. Research Questions

Problem will be accomplished by answering the following questions:

1. What is the suitable database segregation approach according to performance, security, and cost?

2. How do database multi-tenancy approaches affect the performance of tenant's VM?
3. What are the major effects of database segregation techniques on the cost of storage, the size of memory, and number of CPUs?
4. What are the major effects of database segregation techniques on the security?

1.7. Objectives

This research aims at findings to show the impacts of database segregation approaches on tenant's performance, cost, and security. To achieve these goals, this research proposed to run various experiments that cover all database segregation approaches that are available in cloud computing technology. Consequently, the proposed experiments in this research was working in one cloud using the same cloud manager. Performance testing captured in term of response time for CPU, memory, and disk. The security in each approach measured in term of common encryption algorithms (i.e. using RSA algorithm) that preserve data confidentiality.

Many issues will be addressed in this research such as:

- Comparing database segregation approaches based on performance and cost.
- Comparing database segregation approaches based on performance and security.
- Comparing database segregation approaches based on cost and security.

1.8. Motivation

Recently, research studies in cloud computing focused on explaining and recommending the services are provided by cloud computing such as multi-tenancy service. Thus, the literature showed many challenges and obstacles that were classified as a starting point for researchers around the world. One of these obstacles is to find the suitable environment that accomplishes the best performance, the lower cost, and the best security. This research took into consideration the implementation of database segregation approaches that is available for tenants in cloud computing platforms. This motivates the author to find a way to help CSP to define the number of tenants for cloud computing environment based on their needs. Hence, the decision for selecting the appropriate database segregation approach based on the expected scientific recommendations from this research. Furthermore, saving time and money is the common factors that affect tenant's selection.

Schiller O., et al. studied multi-tenancy in RDBMS for SaaS service model. Hence, their research concentrated on using features of RDBM to support tenant area data management cost. Hence, they pointed out to use schema inheritance concept in order to isolate each tenant from others in the cloud, and they shared schema between cloud tenants. Furthermore, their results showed that virtual schemas that were inherited intended to describe application core schemas. As well as, the virtual schemas that were inherited from other virtual schemas enable to specialize the application core schema for specific domain (Schiller O., et al., 2014). In this context, this research will focus on the effects of database segregation approaches on the cost CPUs, and memory, and storage needed.

Ru J., et al. (2014) pointed out multi-tenancy challenges and implications. Thus, data management, security, performance, maintenance, scalability and resource provisioning were presented as the most important challenges faces cloud computing multi-tenancy feature. Actually, they built a cloud system and used software and tools in order to extract the performance of the whole system from different types of architecture (i.e. isolated, semi shared, and shared database). For that purpose, they used stress testing using STRESSCLOUD software. Hence, they compared the results after applying the same cloud in cloud simulator called CLOUDSIM (Ru J., et al., 2014). This research will use the database segregation approaches in order to extract the performance after applying the addition, deletion, and updating SQL queries on database instant.

Sun Y. et al. (2014) showed the most important security issues that represented as a security challenges for multi-tenant environments in cloud computing. Hence, data availability, data confidentiality, data integrity, and privacy in cloud computing were addressed as security challenges affect tenant's security (Sun Y. et al., 2014). This research will take into consideration data confidentiality issue. Hence, this research will extract the data confidentiality by applying a common encryption algorithm (e.g. RSA algorithm) to capture the effects of database segregation approaches on security.

1.9. Contribution

In this research, we made an investigation on the effects of database segregations approaches on the performance, cost, and security. Performance measured from CPU, RAM, and disk point of views. The evaluation of cost found from the number of running CPUs, the consumed memory size, and the needed storage size. The security was considered from data confidentiality using a common encryption algorithm.

Our contribution in this research is to find the suitable database segregation approach that meets multiple parameters concurrently (e.g. decreasing the performance versus increasing the cost of cloud application using a specific database segregation approach). Hence, finding the suitable environment that can handle variable number of cloud tenants that are working concurrently will be represented as critical point. Recently, many studies found in the field of security, cost, and performance.

1.10. Methodology

The methodology that was used to develop our model contains the following phases:

- Study and Analysis Phase.
- Design and Implementation Phase
- Evaluation Phase

Study and Analysis Phase

In this phase the work stared based on the problem statement which was for addressing the common database segregation approaches used in the database multi-tenancy environments. Thus, point out the effects of database segregation approaches on cost,

performance, and security. The acquired information and knowledge from this phase was as follows:

- Studying the specifications for each database segregation approach.
- Studying database segregation performance tools and algorithms.
- Studying the commonly used tools for measuring database segregation cost.

Design and Implementation Phase

This research was carried out a case study which covers building one cloud environment; we was OpenStack as cloud managers. The experiments used to make fair comparison using the following steps:

- Instances specifications and its performance in the real world.
- Measuring CPU and memory response times for each segregation approach.
- Measuring the number of CPUs, memory size, and storage size to extract the cost for each segregation type.

Analytical Phase

We designed three experiments to evaluate the performance, cost, and security database segregation approaches. The results was used to fill the comparison tables. Hence, the result aggregated to find the average of mixing research parameters together.

CHAPTER TWO

Literature Review and Related Work

2.1. Overview

This chapter shows collection of the most relevant work in the literature that relate to the scope of this research. This literature review covers concepts that have been addressed in this research, namely, cloud computing multi-tenancy, database segregation techniques, encryption techniques for cloud databases, and performance testing.

2.2. Literature Review

- **Soofi, A., et al.** discussed the security issues and their existing solutions in SaaS delivery model of cloud computing. Hence, they described some of security issues in SaaS like Data security, availability, authentication and authorization, network security ,backup, data breaches, data integrity and web application security. Then, they discussed some of existing security solutions in term of advantages and disadvantages. (Soofi A., et al., 2014). In this research, we concentrated on the data security which they flag it as one of the important issues for cloud computing environment. Therefore, we assigned data security as a parameter in this research in order to measure its effects on performance as well as finding the suitable security methods based on the DB segregation approaches.

- **Sun Y.,et al.** reviewed some of security techniques and challenge. The authors started in talking about cloud computing and its characteristics, then they talked about privacy and data security. Their way in researching was to give some studies and researches that talked about organization of data security and privacy in cloud computing. The authors studied Data Integrity, data confidentially, data availability and data privacy. Their paper was giving definition of each one then talked about some of studies that talked about their types (if exist) and its advantages and disadvantages (Sun,Y.,et al.,2014). Hence, we took into consideration the commonly used security algorithms that are used in cloud computing environments in order to measure its implications on database segregations.

- **Schiller, O. et al.** studied Multi-tenancy in RDBMS for software as a service. Hence, the authors used feature RDBM to support tenant area data management natively. The authors talked about relational database approaches (shared machine, shared process and shared table).Then they gave idea about mapping schema techniques. The authors introduced tenant as first class database object, on the other hand they used schema inheritance concept which is avoid redundancy. According to their study and measurements they found that the sharing in application core schema will decrease main memory consumption (Schiller, O. et al., 2014). Consequently, in this research we took into consideration the cost parameter in order to measure the effects of DB segregation approaches on the memory cost.

- **Saxena, V. et al.** Studied metadata and data storage with encryption. The authors tried to give more idea about how to increase the privacy preserving approaches in cloud computing. Hence, the authors divided data into three types: Normal, sensitive and critical. In sensitive and critical data, they build application that can defrag data to multi tables, then they used many phases to retrieve it. For that purpose, they used hash solution to avoid data storage to know the result. So the authors used fragment and encryption technique to increase the strength of privacy data (Saxena, V. et al., 2014). Consequently, we used their fragmentation techniques in shared schema shared database approach in order to find its implications on the privacy of tenant's data in this approach.

- **Bardiya, P., et al.** analyzed basic problem of cloud computing data security. Hence, they gave definition of data security and cloud computing, after that they tried to apply data security using Hadoop model with its security goals and architecture (Bardiya, P. et al, 2014). Therefore, in this research we used their metrics to evaluate the cost of adding security feature on a cloud computing environments among several database segregation approaches.

- **Aulbach, S. et al.** described Chunk Folding as a new schema mapping techniques which logical tables are vertically portioned into chunks that folded together into application-specific conventional tables a fixed set of generic Chunk Tables. Hence, underscore the importance of application extensibility, outlines on some common schema mapping techniques, then they introduced chunk folding with explain on some of experiments with managing many tables (Aulbach, S. et al., 2008). Hence, in this research we took their empirical

findings to define the number of records as well as the number of tenants that can interact in cloud environment as a light and hard states for each DB segregation approach.

- **Ru, J., et al.** discussed the multi-tenancy challenges and its implications. The authors started in talking about cloud computing architecture, multi-tenancy implications, and then the challenges of multi-tenancy which are data management, security, performance, maintenance, scalability and resource provisioning. They use many experiment to measure the challenge that facing multi-tenancy. They use software and tools like Cloudsim (simulate multi-tenancy), Hadoop (security) and stress cloud for measuring performance (Ru, J.et al., 2014). In this research, we took their threshold findings that was used to define the highest applicable number of tenants in an private cloud and public cloud in order to obtain the suitable number of tenants in our experiments to be as less faults as possible to realize the proposed methods and experiments in this study.

- **Al-Alwan, M. and Zaghloul, S.** made an analytical study of Multi-Tenant database in a Cloud Environment. The authors started in talking about multitenant database architecture(separate database, shared database with separate schema and shared database with shared schema) after that talked about schema mapping techniques which are : Private table, Universal Table ,Extension table, pivot table, chunk table, chunk folding table and XML data type. Then started to analyze each one with giving advantage and disadvantages (Al-Alwan, M. and Zaghloul, S., 2013). From their empirical findings we

decided to choose the private and pivot table to work in separate shared schema and shared database shared schema approaches.

2.3. Related research works to this research

- Yaish, H. and Goyal, M. proposed an architecture design to build an intermediate database layer to be used between software application and Relational Database Management System (RDBMS) to store and access multiple tenant's data in the Elastic Extension Table (EET). Multitenant database layer combines multitenant relational table and virtual relational table and make them work together to act as one database for each tenant. Thus, based on shared database shared schema data isolation approach, EET, and level 3 of SaaS; they built multitenant database architecture to simplify and speed up the development of multitenant database solutions which permit database service provider to create single database application that support multiple tenants on the same hardware, software, and infrastructure. They found that the future work should focus on evaluating the performance of retrieving and storing tenant's data over multiple server instances (Yaish, H. and Goyal, M., 2013).

- Ni, J. et al. proposed an adaptive database schema design method for multitenant application. They made tradeoff between Isolated Table Shared Instance (ITSI) and Shared Table Shared Instance (STSI) by finding the balance between them to achieve good scalability and high performance with low requirement. Therefore, their core idea based on identifying the important attributes and uses them to generate an appropriate number of base tables. Hence, selecting these

attributes based on a well-known page ranking algorithm. Consequently, their findings showed that by generating adaptive schema design based on their experimental results and synthetic datasets it can yield high performance for the whole environment (Ni, J. et al., 2014).

- Chauhan, R. and Kaur, S. studied the technologies to build a cost effective, protected, and scalable multitenant infrastructure, and how to improve the security and enhance its performance. For this purpose, they explored isolation, security, customization, and scalability. Hence, they evaluated the performance of those patterns using multiple experiments. Their findings showed that it is important to work on a protocol that evaluate performance and find the best performance with cost and make tradeoffs (Chauhan, R. and Kaur, S., 2014).

- Kerb, R. and Loesch, M. made a classification of methods to ensure the performance isolation based on request admission control. As well as, they studied informational requirements. They found out that sharing operations between different tenants is to decrease the operational costs. In contrast, their findings showed that it is complicated to ensure the isolation of the performance observed by different tenants. Thus, they discussed five conceptual approaches with increasing the capabilities to control performance in order to determine the complexity and the need for detailed information about the system at run time. They found out the simplest approach based on a static admission control like a round robin, which had a successful result in selecting tenants requests from tenant specific queries (Kerb, R. and Loesch, M., 2012).

- Narasayya, V. et al. presented SQL Virtual Machine (SQLVM) which represent a light weight abstraction of a VM running within database server that provide resource reservations. Hence, they proposed low overhead techniques to objectively meter resource allocation to establish accountability. Hence, they implemented a prototype of SQLVM in Microsoft SQL Azure to evaluate the performance isolation. In their experiment, four tenants concurrently execute one instance of each work load. Therefore, they ran different workload combinations. Experiment reported in the sections that are focused on scenarios where the server had sufficient resources to meet the promised reservations, the results showed that the sum of all the reservations did not exceed the available resources at the server. In this context, they showed that one of the important future works is the focus on overbooking operation performance and polices that are used to let the system the ability to make tradeoffs (Narasayya, V. et al, 2013).

- Schaffner, J. et al. developed a model for predicting the response time for an in-memory column database running a scan intensive query workload. Hence, they showed how to use this model to predict whether a database instance will be able to meet response time goal for a particular assignment of tenants to the server. They took multiple parameters for the prediction model such as the size and request rates of tenants placed on a server that is responsible to extract how many byte in the memory database instance need to scan in a given interval. Therefore, their results showed that 99% of the values can be obtained for a set of tenants containing less data but high request rates and a setup with more data but lower request rates (Schaffner, J. et al., 2011).

CHAPTER THREE

The Proposed Model and Experiments Design

3.1. Overview

In this chapter, we present a detailed description of the proposed model, as well as, discuss the proposed experiment's design, experiments flow charts, and finally define the evaluation process of research parameters through.

3.2. The Proposed Model Architecture

In this section, we describe the proposed work for this research. Therefore, we made multiple experiments to cover all parameter in this research. Thus, private cloud was built using OpenStack cloud manager. The first experiment took into consideration the isolated database approach. Figure 3.1 shows the general framework of the first experiment.

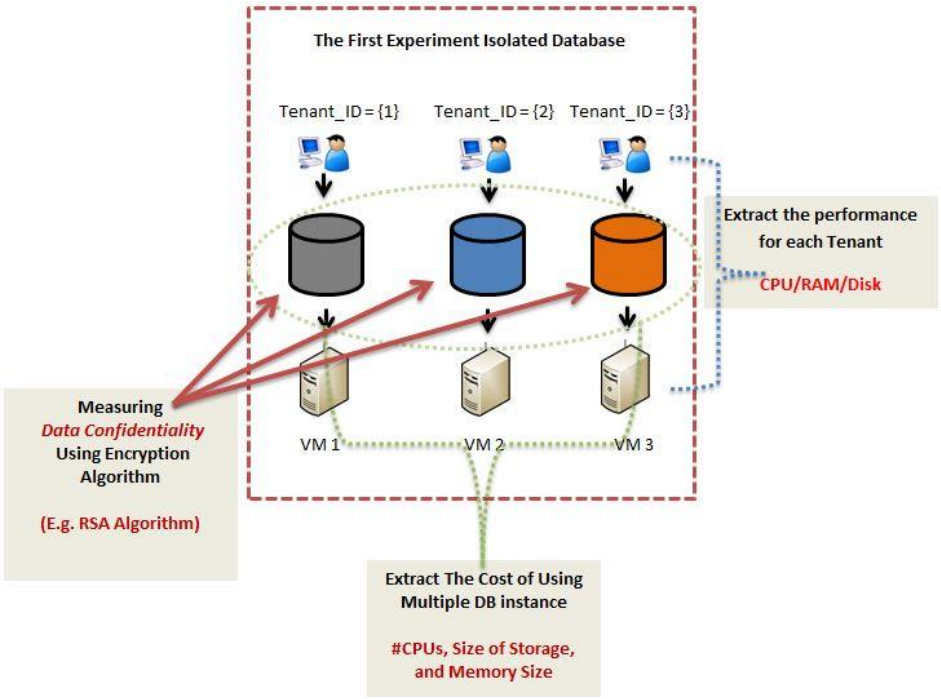

Figure 3. 1. Isolated DB :The general framework of the proposed experiment

In the second experiment, separate schema with shared database approach had been applied in a private cloud that was built using Openstack cloud manager using Xen virtualization hypervisor. Thus, three table's schemas were created in the same database. Hence, each schema was assigned for each tenant in the cloud system. Figure 3.2 shows the general framework of the second experiment.

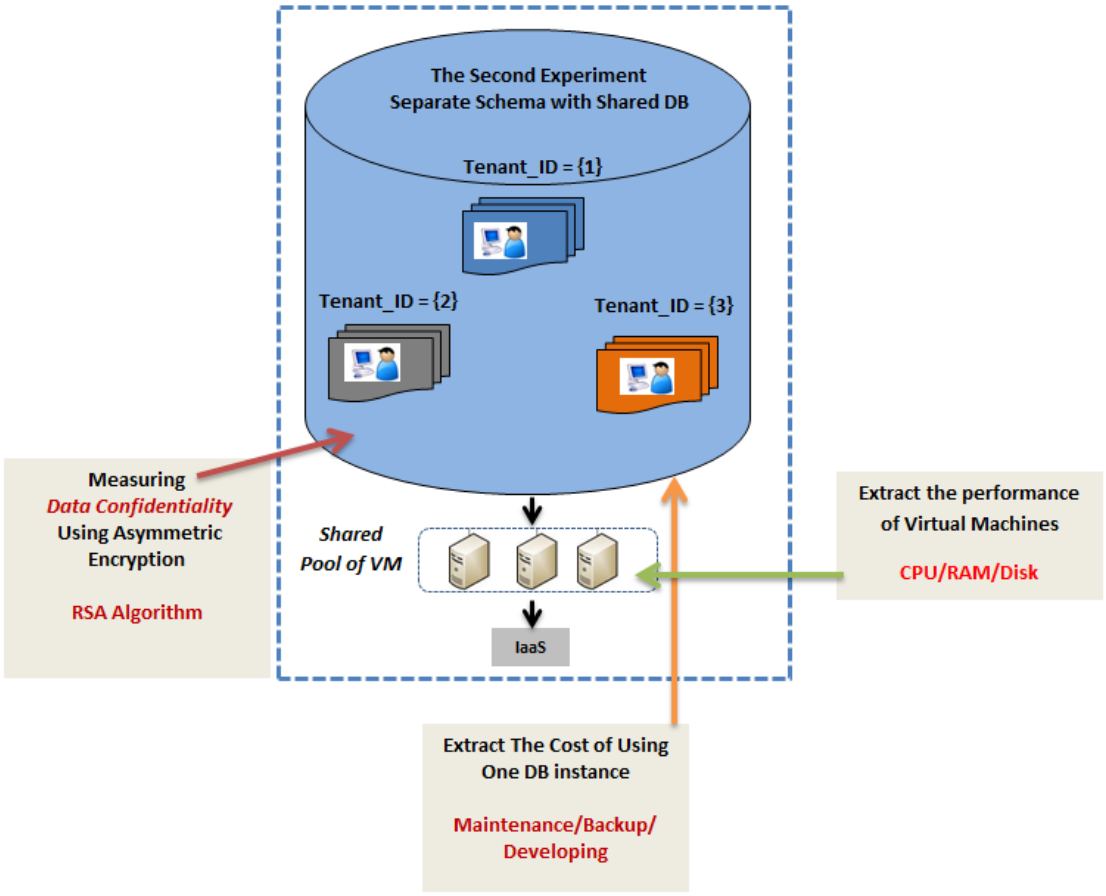

Figure 3. 2. Shared DB with Separate Schema: The general framework of the proposed experiment

In the third experiment, shared schema with shared database approach had been applied in cloud system. Therefore, one database instance with one schema was assigned for multiple tenants in the cloud. Thus, this experiment was capable to satisfy research

objectives. Figure 3.3 shows the illustration of the general framework the third experiment.

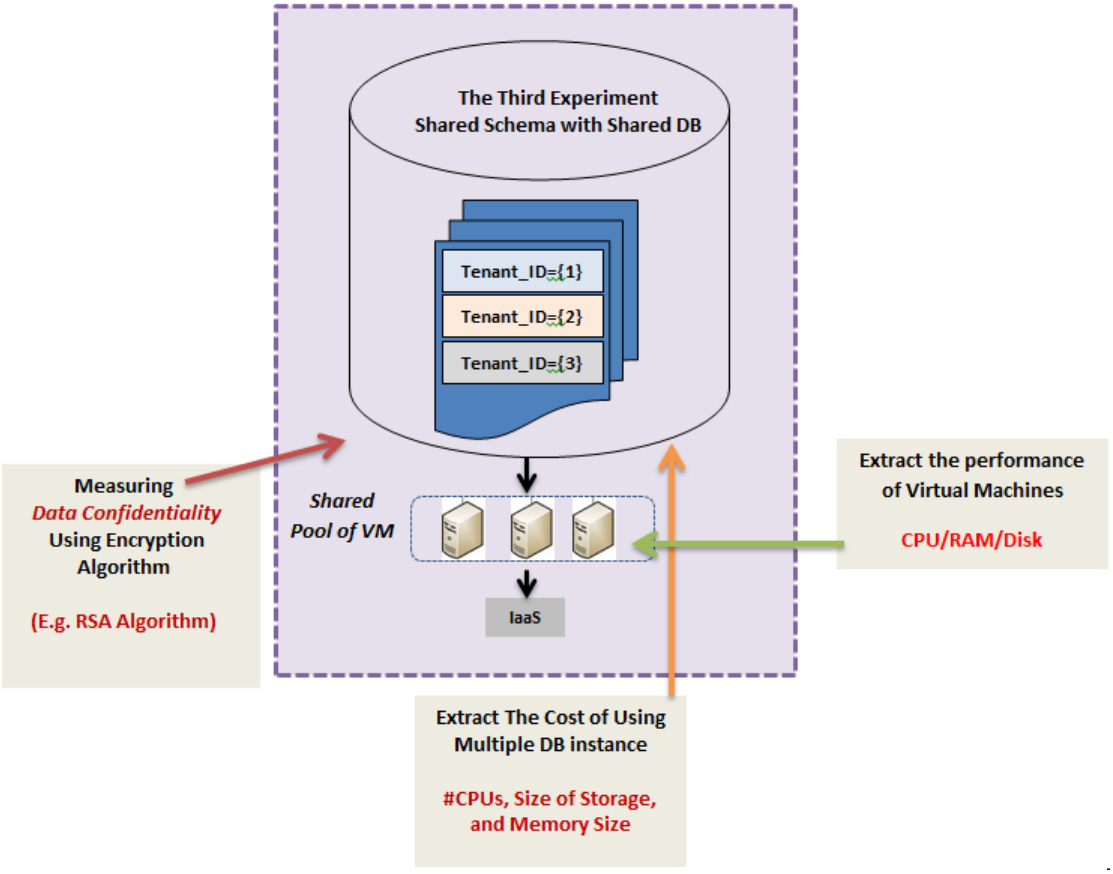

Figure 3. 3. Shared DB with Shared Schema :The general framework of the proposed experiments

3.3. The proposed scenario in the proposed experiments

In this section, we explained the proposed scenario in order to measure the comparative variables in this research (i.e. performance, cost, and security). In this context, performance was extracted based on measuring CPU response time for tenant, as well as, the cost was measured for CPU, RAM, and storage size for tenant's database. As well as, the encryption cost of database segregation approaches using AES algorithm. Thus, performance extraction scenario took into consideration finding the response time for computational components after querying four basic operations (i.e. addition,

retrieving, updating and deletion). The extracted response time was measured in Millisecond (ms) unit. Figure 3.4 shows the flow chart for the whole experiment.

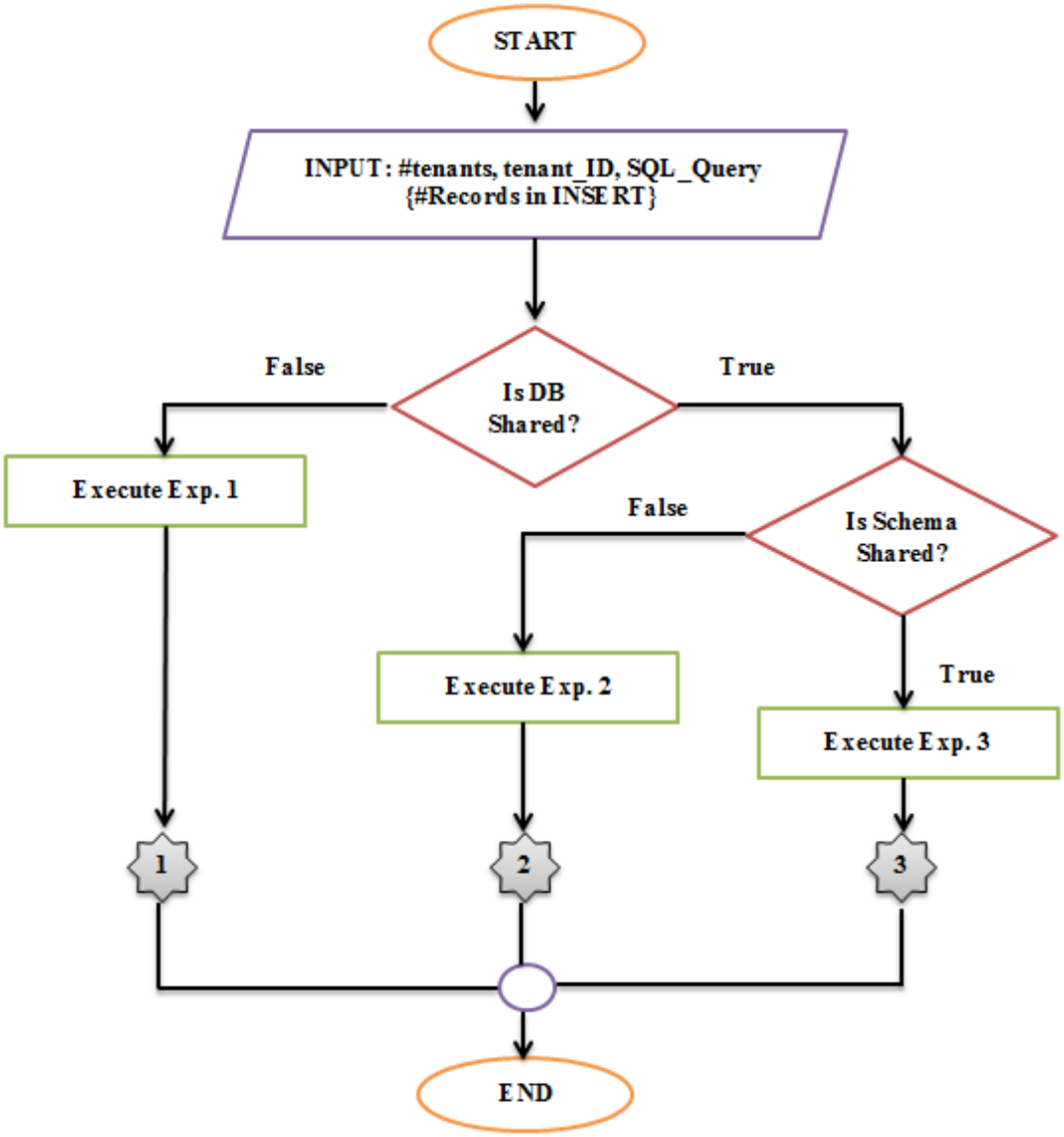

Figure 3. 4. Flow Chart for the whole work.

Basically, the first experiment is denoted as Exp.1 that represents the isolated database, the second experiment is denoted by Exp.2 that represents Shared database with separate schema, and the third experiment is denoted by Exp. 3 that represents the

shared database with shared schema. The first experiment flow chart is illustrated in figure 3.4.2 that shows the procedures were used to implement the experiment.

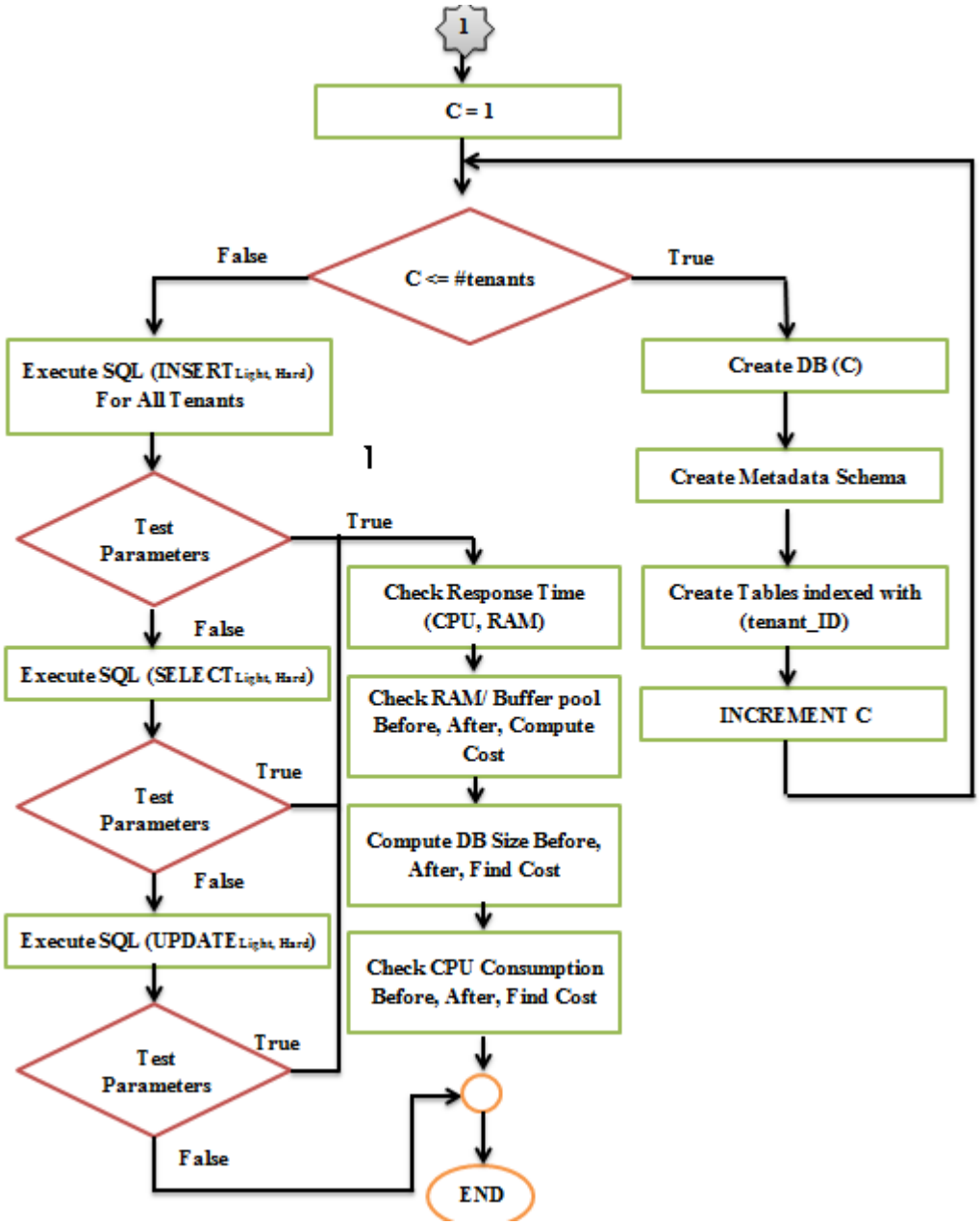

Figure 3.5. Flow Chart for first experiment with testing procedures.

The core idea in the isolated database is that each tenant has its own database. Thus, the components shared in this approach are database server (i.e. Linux, UNIX, or Windows), and database instance. In this experiment tenant_ID will be a postfix for database name. Figure 3.6 shows the database creation command.

```
/* MYSQL Command - Alaa Dawoud*/
/* Creating Database for each tenant in the 1st Experiment */
CREATE DATABASE EMPLOYEES+(tenant_ID);
```

Figure 3. 6. MySQL Create Database Command

Accordingly, sharing the table with separate schema was implemented by adding an attribute to every table in order to segment records, to indicate the owner of the data, this attribute called tenant_ID. Hence, this approach shares database server (i.e. Linux, UNIX, or Windows), database instance, and source table. In this experiment we create a table called transactions to be connected with another table called employee. Figure 3.7 shows MySQL command for creating both tables and adding multitenancy feature on them.

```
/* MYSQL Command - Alaa Dawoud*/
/* Creating Tables for sharing Transaction one in the 2nd Exp. */
CREATE Table Transactions
  (
    Trans_ID Int Auto_Increment Not Null Primary Key
    , Tenant_ID Int Not Null
    , Description Varchar(50) Not Null
    , Trans_Date Timestamp Default Now() Not Null
    ,Complete_Date DateTime Null
  );
```

Figure 3. 7. MySQL create multitenant table in the second experiment.

Consequently, the tenant's table was created in the same way using MySQL in order to connect the two tables for segmenting transactions table. Figure 3.8 shows MySQL command for creating Tenants table.

```
محرر SQL | محرر SQL 2 | محرر SQL 3

/* MYSQL Command - Alaa Dawoud*/
/* Creating Tenant Table to connect Transaction Table in the 2nd Exp. */
CREATE Table Tenants
  (
      Tenant_ID Int Not Null Primary Key
    , Username Varchar(20) Not Null
    , Fname Varchar(20) Not Null
    , Lname Varchar(20)
    , Pass_word Varchar(10) Not Null
  );
```

Figure 3. 8. MySQL command for creating Tenants Table

Figure 3.9 shows the flow chart of the shared table with separate schema approach experiment's design.

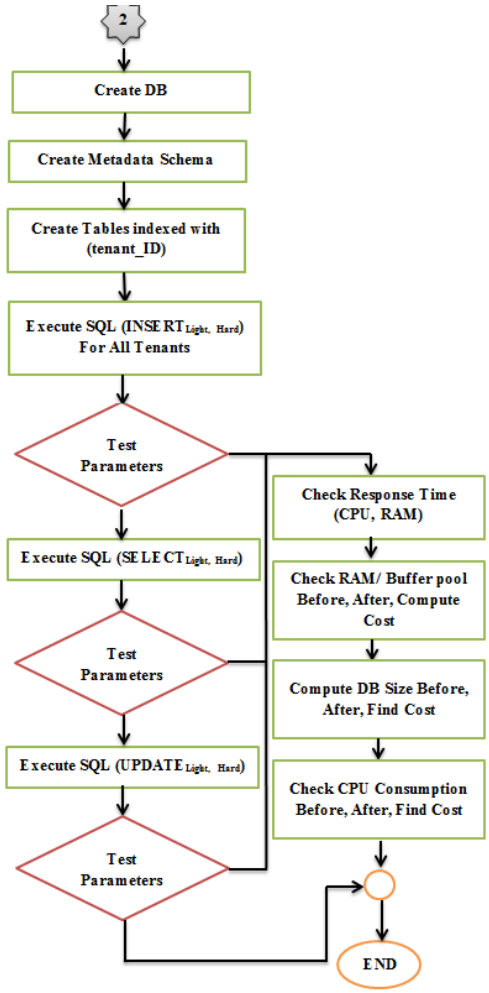

Figure 3.9. Flow chart for shared table with separate schema approach.

In the third experiment of shared database with shared schema approach, we tried to logically group two tables (i.e. objects) in order to provide a unique namespace for Transactions table and Tenants table. Thus, Transactions table may contain different attributes depending on users in the tenants. Therefore, we created a schema called Tenancy that provides both tables as shown in figure 3.10.

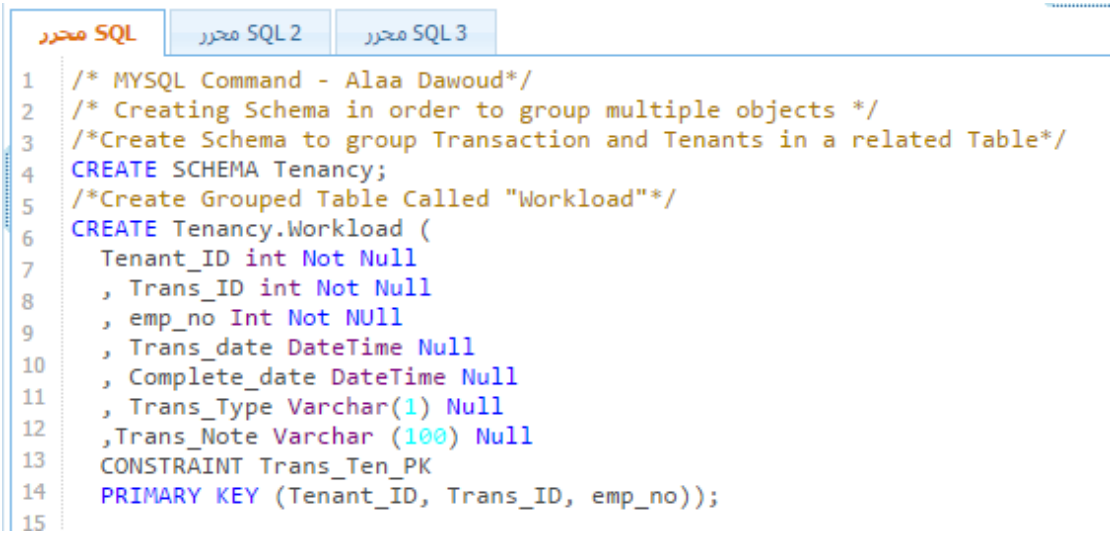

Figure 3. 10. Create schemas in MySQL command and implementing Workload table.

Indeed, multitenant database requires achieving high performance, to serve large number of tenants. Thus, the multitenant database needs to have excellent scalability and low space of requirements. For that purpose, big challenge was put forward is to design high quality schema that merge different core objects to manage data in an effective way. Thus, the result table after sharing and creating that schema named physical table (i.e. in our experiment it is Workload table). Figure 3.11 illustrates the flow chart for implementing the third experiment with testing parameters.

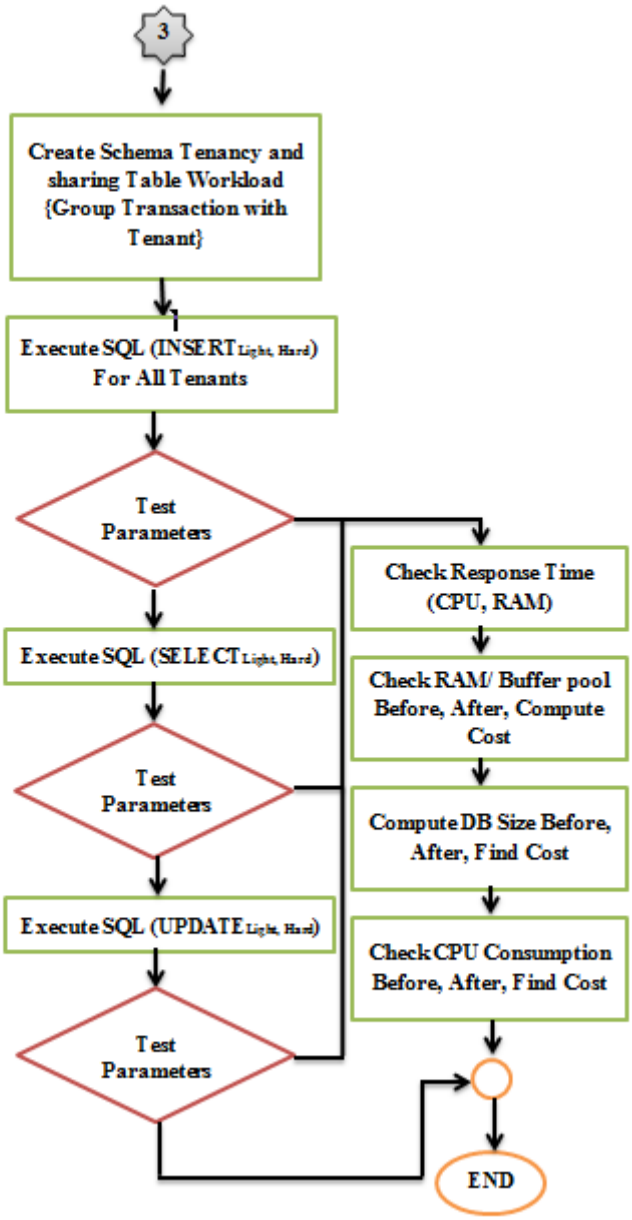

Figure 3.11. Flow chart for sharing schema with shared table

3.4. Workbench MySQL Database Implementation

The Employees sample database was developed by Patrick Crews and Giuseppe Maxia and provides a combination of a large base of data (approximately 160MB) spread over six separate tables and consisting of 4 million records in total. The structure is

compatible with a wide range of storage engine types. Through an included data file, support for partitioned tables is also provided.

In addition to the base data, the Employees database also includes a suite of tests that can be executed across the test data to ensure the integrity of the data that you have loaded. This should help ensure the quality of the data during initial load, and can be used after usage to ensure that no changes have been made to the database during testing. Figure 3.12 shows the Workbench Employees database implementation.

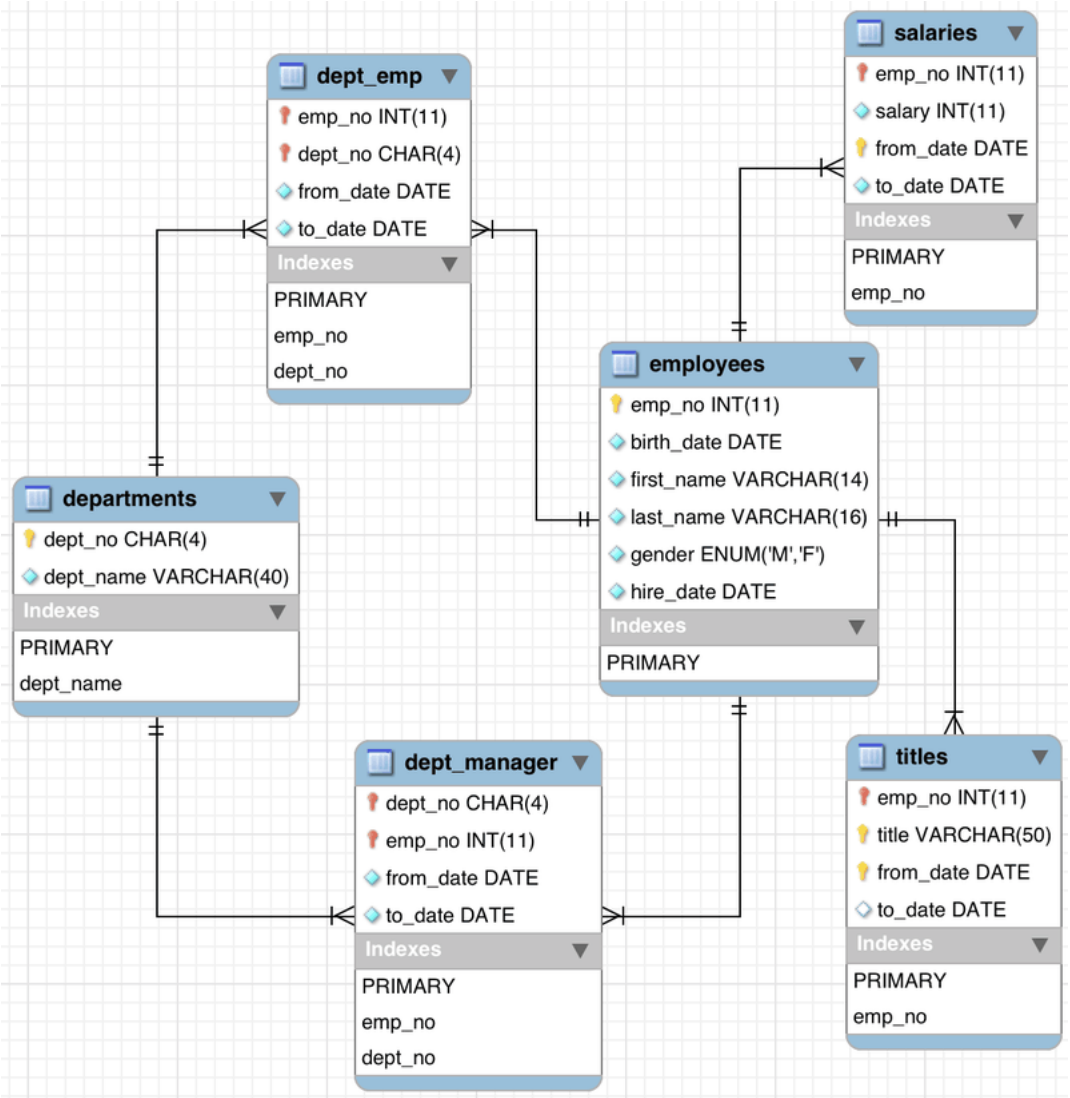

Figure 3.12. Workbench MySQL Database Implementation(Employees)

We modified in employees database according to approach. In First approach (Isolated Database) we created new table called transaction_tenants as the following:

```
CREATE TABLE `transaction_tenants` (
  `Trans_no` int(11) NOT NULL AUTO_INCREMENT,
  `emp_no` int(11) DEFAULT NULL,
  `trans_date` date DEFAULT NULL,
  `trans_type` varchar(1) DEFAULT NULL,
  `trans_description` varchar(200) DEFAULT NULL,
  `trans_notes` varchar(200) DEFAULT NULL,
  PRIMARY KEY (`Trans_no`),
  KEY `emp_no` (`emp_no`),
  CONSTRAINT `transaction_tenants_ibfk_1` FOREIGN KEY (`emp_no`) REFERENCES `e
) ENGINE=InnoDB AUTO_INCREMENT=1153029 DEFAULT CHARSET=utf8;
```

Figure 3.13. MySQL command for creating transaction_tenants Table

Therefore, we exported and imported the database as number of tenants that we will use. Figure 3.14 shows the list of databases in isolated approach.

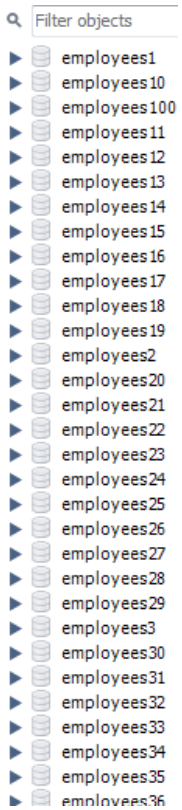

Figure 3. 14. List of databases (Isolated Approach)

In shared database separate schema, we use one database, but each tenant has it is own schema/table in the database, so we created tables depending on numbers of tenants that we used. Figure 3.15 shows the code of creating the transaction table for shared database shared schema approach.

```
CREATE TABLE `transaction_tenants100` (
  `Trans_no` int(11) NOT NULL AUTO_INCREMENT,
  `emp_no` int(11) DEFAULT NULL,
  `trans_date` date DEFAULT NULL,
  `trans_type` varchar(1) DEFAULT NULL,
  `trans_description` varchar(200) DEFAULT NULL,
  `trans_notes` varchar(200) DEFAULT NULL,
  PRIMARY KEY (`Trans_no`),
  KEY `emp_no` (`emp_no`),
  FOREIGN KEY (`emp_no`) REFERENCES `employees` (`emp_no`)
) ENGINE=InnoDB AUTO_INCREMENT=1153029 DEFAULT CHARSET=utf8;
```

Figure 3. 15. Creating transaction_tenants Table (Shared database Separate Schema approach)

For instance, if we will deal with 100 tenants so we must have at least one hundred tables plus the other tables. Figure 3.16 shows the list of schema tables in shared database with separate schema approach.

Figure 3. 16. List of Schema/Tables (Shared Database Separate Schema Approach)

In Shared database shared schema, we created one database with shared schema/tables for all tenants by adding filed tenant_id.

3.5. Software Implementation

Extracting performance and cost for this research was conducted using a VB.NET program that simulates the cloud environment as a distributed system after applying database segregation approaches. Hence, we are using this software in system has the as shown in table 3.1.

Table 3.1. The experimental system's specifications

CPU	:	Intel Core I3 1.9 GHZ
RAM Size	:	2 G.B.
Hard Disk	:	50 G.B.
Operating System	:	Windows 7 64 bit
Database	:	Mysql database

Hence, this software was capable to find the response time of CPU, the cost of CPU, the cost of RAM, and the cost of storage. Figure 3.17 shows the first Graphical User Interface (GUI) that represents the simulation of isolated database approach.

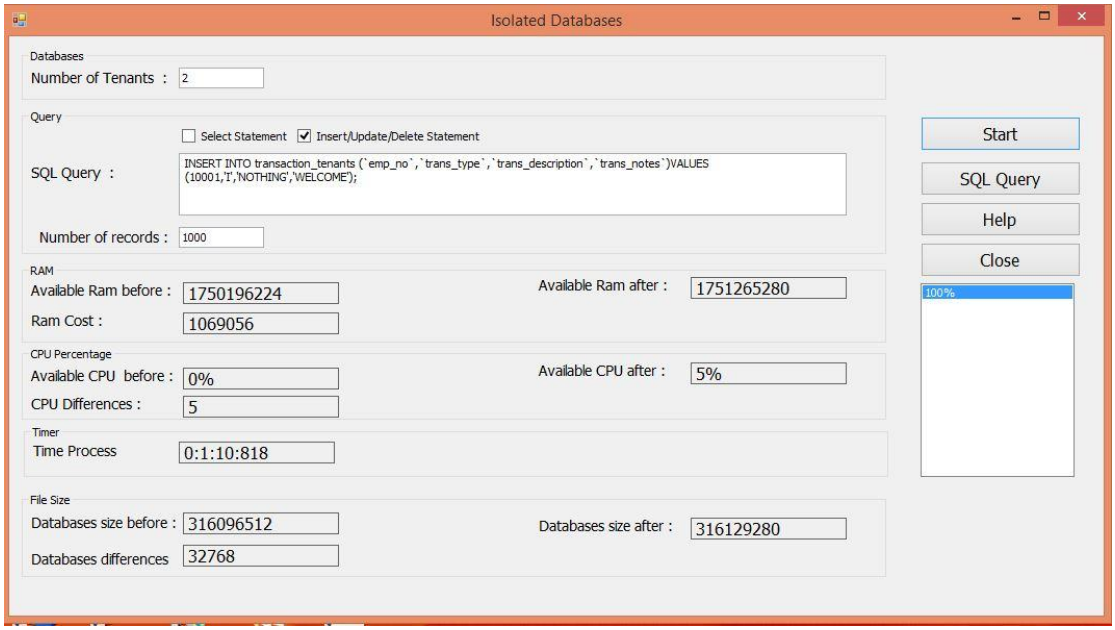

Figure 3. 17. Isolated database approach GUI cost and performance

Consequently, the results after compiling the target software shows that for two tenants the system have to create two separate databases and creating initial tables (e.g. Transactions, and Tenants). Hence, for the purpose of computing the cost of CPU, we measured the available percent of CPU (i.e. 0% without I/O and Graphics) before, and the availability after (i.e. 5% without I/O and Graphics) in order to find the cost of CPU (i.e. the difference between the availability after and before) which was 5%. Therefore, the cost of RAM was computed in the same way through measuring the availability of RAM before (i.e. 1750196224 MB) and the availability of RAM after (i.e. 1751265280 MB). Figure 3.18 shows the mathematical equation that was used to compute the cost of CPU and Disk.

$$CPU\ Cost = CPU\ Availability\ After - CPU\ Availability\ Before$$

$$RAM\ Cost = RAM\ Availability\ Before - RAM\ Availiability\ After$$

Figure 3. 18. The mathematical equations was used to find the cost of CPU and RAM

On the other hand, to find the storage disk was done by getting the differences in databases size before and after querying. Figure 3.19 shows the way how can we get the database size in VB.net for SQL.

```
SELECT  sum( data_length + index_length )  totalsize  FROM
information_schema.TABLES  where table_schema = '" & db1 & "';"
```

Figure 3. 19. The mathematical equations was used to find the databases size

In Isolated database, we apply our experiment into 100 tenants with 100 databases, we use SQL statements (Insert, Update, delete and select) in our experiment. We do three experiment for each sql statement, then we get the average (Response Time, consumed CPU, consumed Ram and Database difference) for each SQL statement to get best result. Table 3.2 shows the list of SQL queries were executed in the experiments.

Table 3.2. Shows the list of SQL queries that were executed in the experiments

Operation	1000 Records	10000 Records	30000 Records
Insert (Without Encryption)	Y	Y	Y
Insert (with Encryption)	Y	Y	Y
Update (without Encryption)	Y	Y	Y
Update (with Encryption)	Y	Y	Y
Select with Index key(without Encryption)	Y	Y	Y
Select with index key (with Encryption)	Y	Y	Y
Select without Index Key (without Encryption)	Y	Y	Y
Select without index key(with Encryption)	Y	Y	Y
Delete (without encryption)	Y	Y	Y
Delete(with Encryption)	Y	Y	Y

Correspondingly, we measure the performance and cost for CPU, RAM, and storage disk. Figure 3.20 shows the GUI of Shared database with separate schema. Hence, there were two tenants that share the same database instance, and both of them were working on the same table. After executing INSERT query for 1000 records the performance of CPU was 1 minute, 2 seconds, and 828 milliseconds. The cost of CPU was 4%, the cost of disk was 98304 MB and the cost of RAM was 11608064 MB.

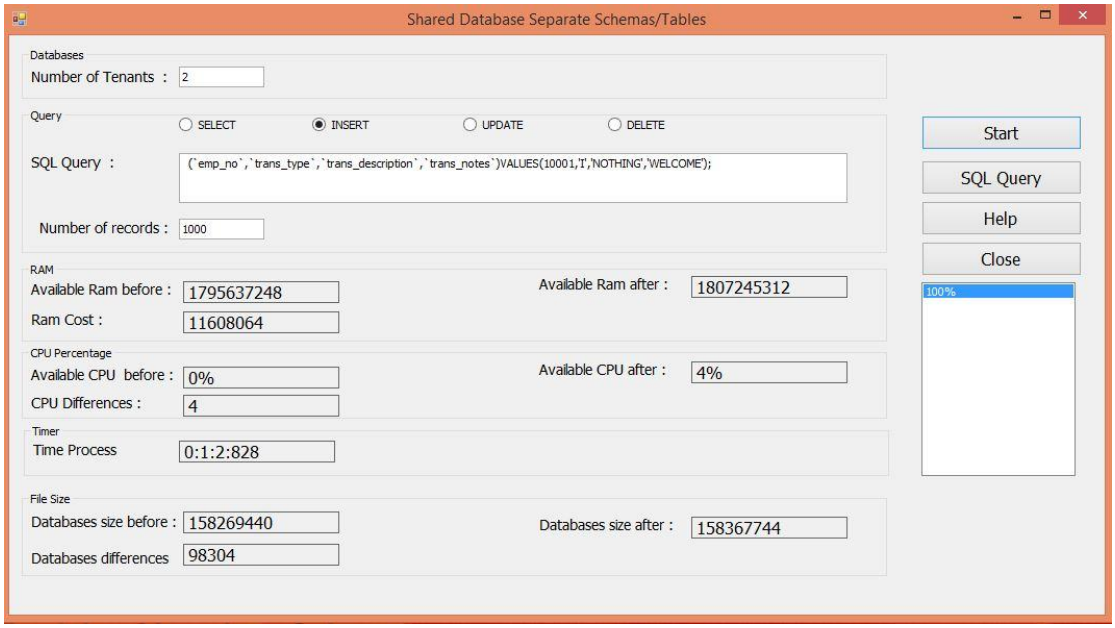

Figure 3. 20. The second experiment of shared database with separate schema GUI

Figure 3.21 shows the GUI of the third experiment. Thus, by reserving three tenants in the system with sharing database instance, table, and the schema the result will be physical table have a unique namespace of objects. Therefore, the INSERT query will take place in the physical table called transaction_ tenants. The performance of CPU was 52 seconds, and 768 milliseconds. The cost of CPU in this approach achieved 5%, the cost of RAM was 7786496 MB, and the cost of disk was 98304 MB.

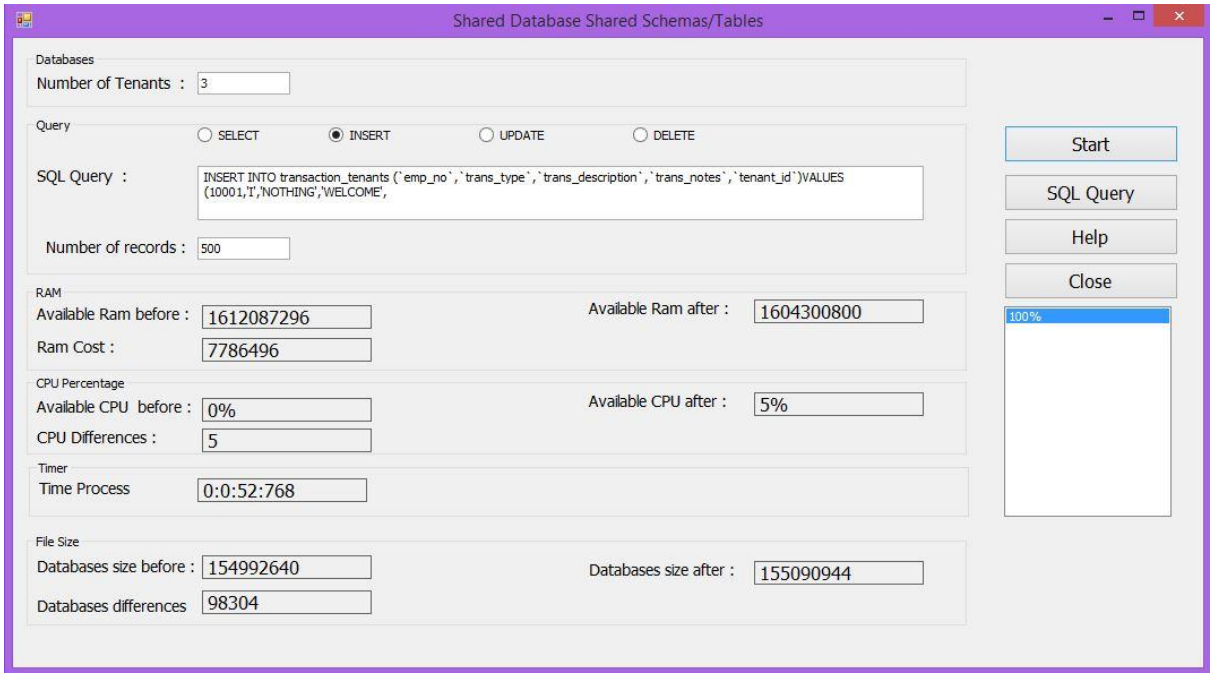

Figure 3. 21. The third experiment of shared database with shared schema GUI

3.6. MySQL queries were used in the experiments

In this context, we are going to discuss the used queries in this experiments. Thus, in this research we used INSERT, UPDATE, and DELETE as a primary queries. Consequently, we applied these queries on all database segregation approaches in order to evaluate the performance of data for each one. Figure 3.22 shows the INSERT Statement to insert 1000 record for each experiment.

```
INSERT INTO transaction_tenants
(`emp_no`,`trans_type`,`trans_description`,`trans_notes`)VALUES(10001,
'I','NOTHING','WELCOME');
```

Figure 3. 22. INSERT Statement in experiment

We measure response time, CPU, Ram and disk storage for each INSERT command by adding 1000 record in each database in each experiment, then we take the average depending on number of tenants each time.

On the other hand, by using UPDATE command, we use the following statement as in figure 3.23

```
UPDATE transaction_tenants  SET  `trans_type` = 'U', `trans_description`
= 'Transaction description is welcome', `trans_notes` = 'Transactions
Notes is nothing'  WHERE `emp_no` = '10001';
```

Figure 3. 23. Update Statement in experiment

The Third SQL command we use is Delete, it is showed in figure 3.24

```
DELETE FROM transaction_tenants where `emp_no`='10001'
```

Figure 3. 24. Delete Statement in experiment

3.7. Encryption and decryption response time for multitenant databases.

Building security for multitenant database must be adequate to cover every aspect of cloud application. Thus, security could be achieved through three core operations which are filtering, permission, and encryption. Filtering operation can be achieved via using an intermediary layer between a tenant and data source that act like a controller where the tenant can see its data as the only data in the database. Permission operation can be achieved through designing Access Control List (ACL); to determine who can access data in the application and what they can do. As well as, using encryption techniques to hide every tenant's critical data; so that it will be inaccessible to unauthorized parties even if they get an access to it. In this research, we made an experiment to evaluate the response time of encrypting data sources after applying data segregation approaches.

Cryptography is the science of using mathematics to encrypt and decrypt data. Cryptography enables you to store sensitive information or transmit it across in secure networks. There is many algorithms for encrypting data, the tow famous algorithms which are RSA and AES. Consequently, RSA is an asymmetric cryptographic algorithm, it has two keys, public and private key. The public key can be known to everyone, we use it to encrypt data, on the other hand, private key we are use it to decrypt data. To encrypt data we used equation 1.

$$c = m^e \mod n \qquad \text{…… Equation (1)}$$

Where:
***n*:** is result of multiplication two random numbers n=p*q
***m*:** is number less than n

On the other hand to decrypt data we used equation (2).

$$m = c^d \mod n \qquad \text{…… Equation (2)}$$

Where:
***n*:** is result of multiplication two random numbers n=p*q
***m*:** is number less than n
***c* :** is the ciphered text

The Advanced Encryption Standard (AES) is formal encryption method adopted by the National Institute of Standards and Technology of the US Government, and is accepted worldwide. In 1997 the National Institute of Standards and Technology (NIST), a branch of the US government, started a process to identify a replacement for the Data Encryption Standard (DES). The AES encryption algorithm is a block cipher that uses an encryption key and a several rounds of encryption. AES is based on the principle Known as Substitution-permutation. AES encryption uses a single key as a part of the encryption process. The key can be 128 bits (16 bytes), 192 bits (24 bytes), or 256 bits (32 bytes) in length. In our experiment we use AES algorithm as a type of encrypting

data. For this purpose, we stored the encrypted data into employee's database. Therefore, security effects could be measured easily on databases approaches. Figure 3.25 shows VB.Net code of encryption procedures using AES algorithm.

```
Imports System.IO
Imports System.Text
Imports System.Security.Cryptography
Public Class Form1
    Private Function Encrypt(ByVal clearText As String) As String
        Dim EncryptionKey As String = "MAKV2SPBNI99212"
        Dim clearBytes As Byte() =
Encoding.Unicode.GetBytes(clearText)
        Using encryptor As Aes = Aes.Create()
            Dim pdb As New Rfc2898DeriveBytes(EncryptionKey, New
Byte() {&H49, &H76, &H61, &H6E, &H20, &H4D, _
             &H65, &H64, &H76, &H65, &H64, &H65, _
             &H76})
            encryptor.Key = pdb.GetBytes(32)
            encryptor.IV = pdb.GetBytes(16)
            Using ms As New MemoryStream()
                Using cs As New CryptoStream(ms,
encryptor.CreateEncryptor(), CryptoStreamMode.Write)
                    cs.Write(clearBytes, 0, clearBytes.Length)
                    cs.Close()
                End Using
                clearText = Convert.ToBase64String(ms.ToArray())
            End Using
        End Using
        Return clearText
    End Function
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load

    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles Button1.Click
        TextBox2.Text = Encrypt(TextBox1.Text)

    End Sub
End Class
```

Figure 3. 25. VB.Net code showing Encryption by AES algorithm

CHAPTER FOUR

Experimental Results

4.1. Overview

In this chapter we discussed the results of the proposed experiments, as well as, calculating the statistical measures (e.g. averages, variance, and Coefficient Variation (CV)) for each experiment in the proposed model. Hence, all comparison criteria in this research covered the research questions and hypothesis. This chapter is consisted into performance evaluation level, cost evaluation level, and security evaluation level.

4.2. Performance evaluation level

The main goal of this level is to discuss the results of performance evaluation based on transaction time in term of the used database segregation approach. Consequently, this level was implemented by employing three experiments to evaluate inserting *n* number of records, retrieving *n* number of records, and deleting *n* number of records. Furthermore, the reliability of each experiment's results in this level was tested by taking the number of records *r* as a control parameter. Hence, each experiment was tested over three phases which were the light phase with a 1000 records, the balanced phase with a 10000 records, and the hard phase with a 30000 records in each database.

Insertion transaction's time evaluation

Basically, we discussed the results of INSERT SQL query in the proposed experiments. Thus, we applied a control parameter (i.e. number of records *n*) in order to test the effect of increasing number of records that were inserted into database. Table 4.1 shows the results of transaction time after executing INSERT query in the experiment.

Table 4. 1. The average of transaction time using INSERT query executed by 91 tenants

Target # Records	Isolated DB (ms)	Shared DB Separate Schema (ms)	Shared DB Shared Schema (ms)
1000	95708.5	98455.5	101203
10000	898861.1	787644.2	1051671.2
30000	2597380.7	3118136.4	5258356.1

In this context, the results showed that by increasing the number of records the transaction time was increased for each database segregation approach. Hence, the control parameter showed that the results of this experiment were reliable due to the increasing in transaction time. Figure 4.1 illustrates the control parameter effects on transaction time.

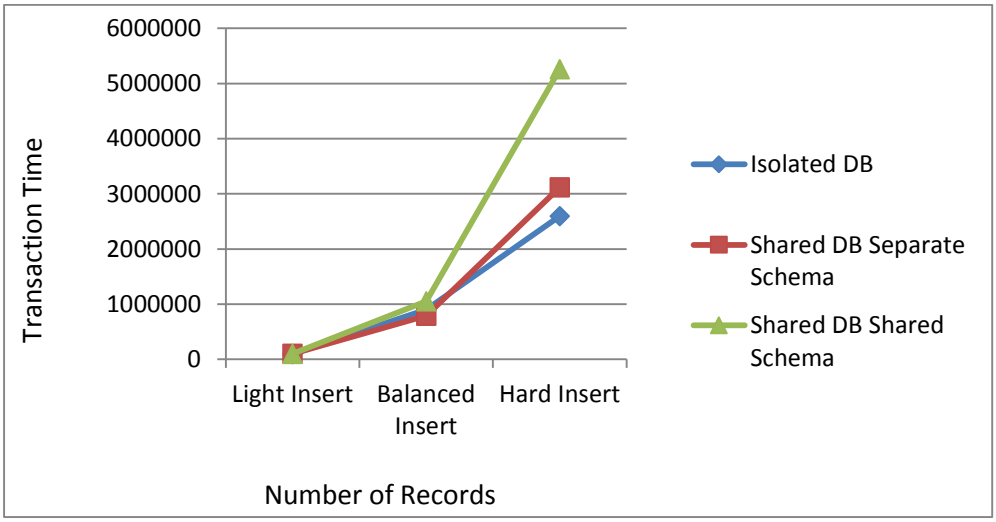

Figure 4. 1. Transaction time for light, balanced, and hard insertion graph

Furthermore, the results of insertion queries showed that the light insertion (i.e. a small number of records) did not affected by the database segregation approach. In contrast, the results showed that the balanced insertion varied in transaction time depending on the database segregation approach. In balanced insertion, we found out that shared database with separate schema had a positive effect on the transaction time compared with shared database with shared schema approach. In hard insertion, the used approach of database segregation had a direct effect on the transaction time as well as a drastic performance gap was existed which showed a negative effect of using shared database with shared schema approach. On the other hand, a positive performance effect had been achieved by using isolated database approach. The expectable results occurred between the isolated DB approach and shared database with shared schema approach because of its results that were equal in all insertion stages. The reason behind this behavior refers to the use of index for all stages in shared DB with shared schema. Figure 4.2 shows the results of database segregation approaches over several weighted insertion.

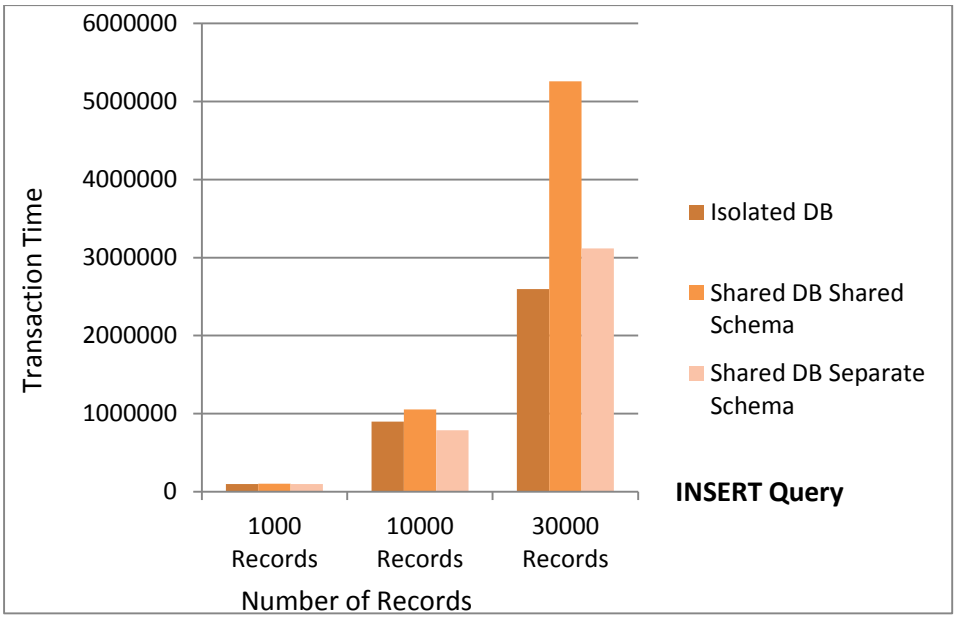

Figure 4. 2. Transaction time results chart of database segregation approaches in insertion queries.

Retrieving transaction's time evaluation

Basically, the retrieving process in the proposed experiments was represented in two cases which were conditional retrieving and non-conditional retrieving. Both conditional and non-conditional retrieving was implemented using SQL SELECT query that was executed in tables from several database segregation approaches. Hence, we assigned the use of condition to identify the suitable database segregation approach in high computational cases. The non-conditional retrieving was used with number of records as a control parameter in this experiment to check result's reliabilities. Table 4.2 shows the average of transaction time using standard SELECT query over database segregation approaches.

Table 4. 2. The average of transaction time using standard SELECT query over database segregation approaches.

Target # Records	Isolated DB (ms)	Shared DB Separate Schema (ms)	Shared DB Shared Schema (ms)
1000	89	98	105.1
10000	253.4	301.9	359.1
30000	524.2	483.1	416.5

Actually, transaction time for retrieving the whole table from database using the standard SQL SELECT query was increasing due to the increase in table's size. Hence, the results showed that the increasing in transaction time was for all segregation approaches which mean that the proposed experiment's results were reliable based on the control parameter. Figure 4.3 illustrates the graph of the transaction time results in database segregation approaches for retrieving *n* number of records.

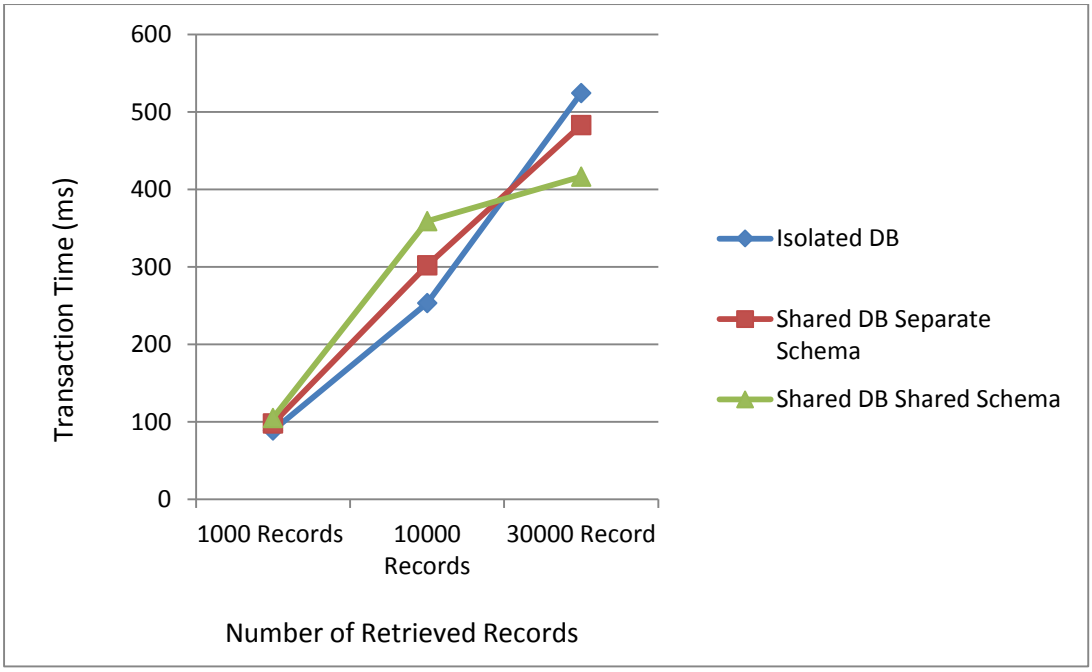

Figure 4. 3. Standard SELECT query results for database segregation approaches

Consequently, we tested the retrieving transaction time using a conditional SQL SELECT query in order to find the suitable database segregation approach in term of intensive computation. Therefore, we added a conditional expression to the retrieving transaction to measure its impacts on transaction time for each database segregation approach, as well as, extract the best and worst cases. Table 4.3 shows the results of intensive computation using conditional SQL SELECT statement in database segregation techniques.

Table 4. 3. The results of intensive computation using conditional SQL SELECT statement in database segregation techniques.

Target # Records	Isolated DB (ms)	Shared DB Separate Schema (ms)	Shared DB Shared Schema (ms)
1000	1560	3133	907
10000	4743	10531	2988
30000	16549	34304	13452

By comparing the results of retrieve transactions using conditional SELECT query, we can find that the shortest transaction time was achieved by employing queries on shared database with shared schema approach. On the other hand, the longest transaction time was achieved by shared database with separate schema approach. Furthermore, from the retrieved number of records point of view a drastic gap in transaction time between segregation approaches was existed due to the increased number of retrieved records. Figure 4.4 shows the graph of the average transaction time for the segregation approaches in the proposed experiment.

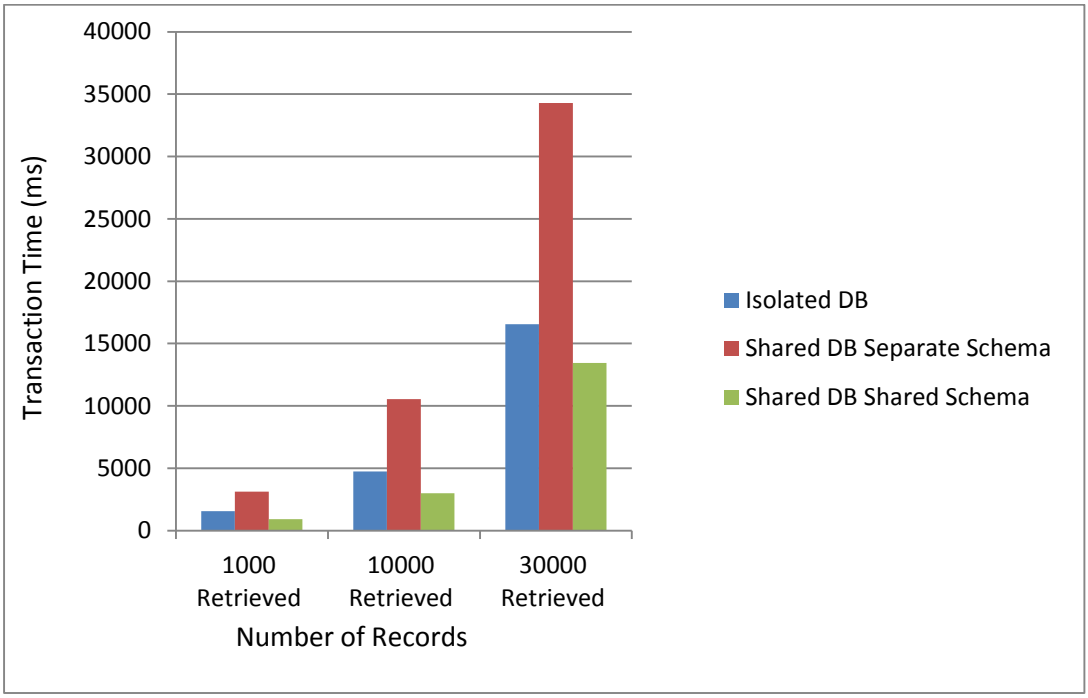

Figure 4. 4. The averages graph in segregation database approaches using select with index query

Delete transaction's time evaluation

In this section, we discussed the results of transaction time needed to delete transaction for *n* number of records from database's tables. In this experiment we chose standard SQL DELETE query. However, it is axiomatic to know that the transaction time

increases while there is an increasing in the number of deleted records. Hence, we chose the number of attribute as a control parameter to check results reliability. On the other hand, we cannot capable to detect the best database segregation approach in term of delete transaction's point of view. Table 4.4 shows the results of SQL DELETE query in the proposed experiment.

Table 4. 4. The average results transaction's time of SQL DELETE query for each DB segregation approach

Target # Records	Isolated DB (ms)	Shared DB Separate Schema (ms)	Shared DB Shared Schema (ms)
1000	5542.6	8638.6	2647
10000	40179	35192	31652
30000	197160	168139.7	151317.6

Basically, the control parameter in this case identified an increasing of transaction's time through the increasing number of attributes. Thus, these results were reliable and could be taken into consideration to find the complexity between database segregation approaches. Figure 4.5 show the results of SQL DELETE transaction time was consumed to handle deleting *n* number of attribute.

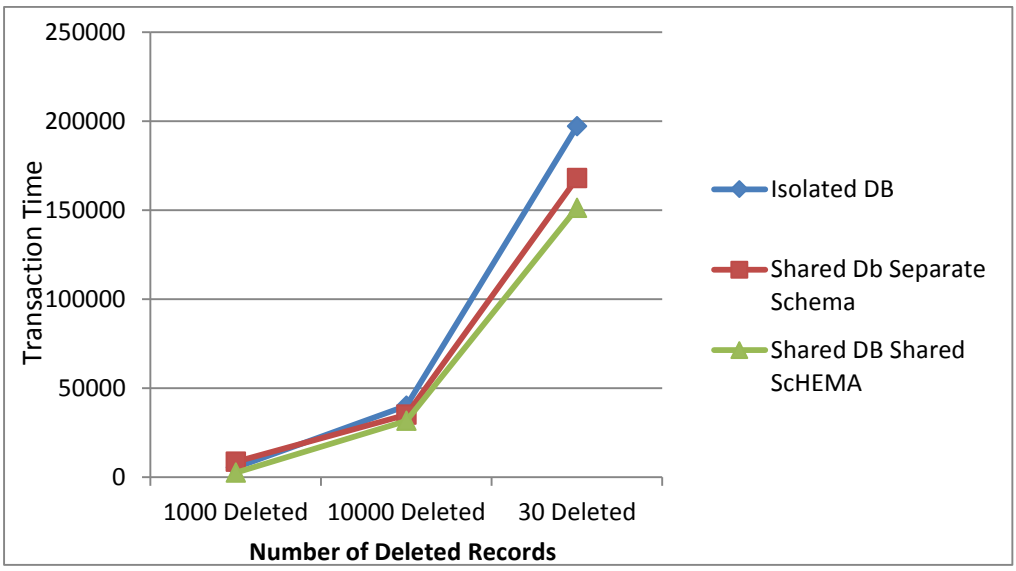

Figure 4. 5. Control parameter graph for delete transaction in the proposed experiment

Furthermore, the results showed that delete transaction's time had been affected by the type of database segregation approach. Thus, by comparing these results we found out that isolated database approach had a negative effect on the delete transaction. However, shared database with shared schema achieved the best case in delete transaction time. Figure 4.6 shows a graph that illustrates these results.

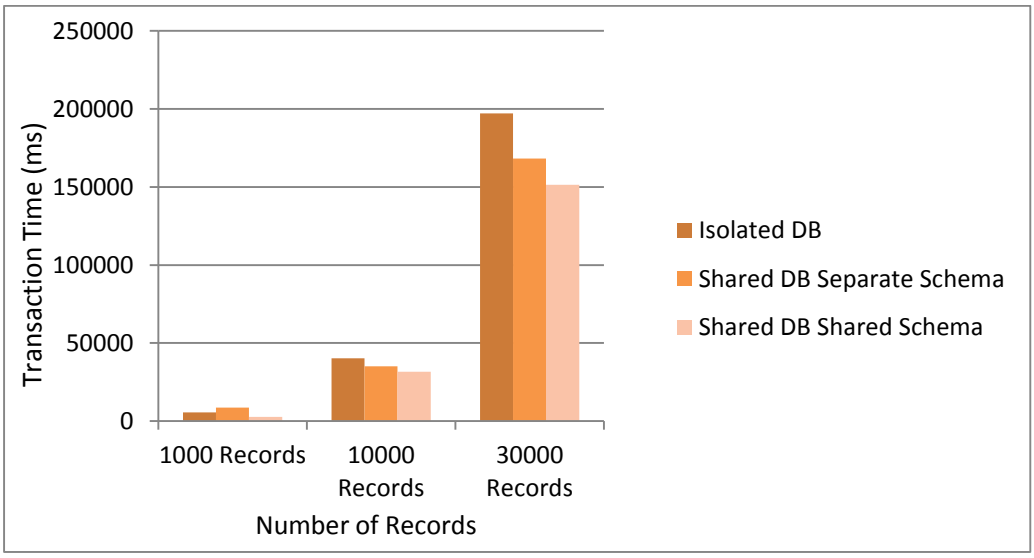

Figure 4. 6. The results of delete transaction's time for each database segregation approach

Conclusions

Actually, performance in the proposed experiments was successfully evaluated in term of transaction time for inserting, retrieving, and deleting transactions. Therefore, the results showed that each type of database segregation approach has its own effects on performance based on type of the tested transaction. Furthermore, we found out that the number of tenants in some cases had its own effects on transaction's time. Table 4.5 shows the analysis of performance evaluation in the proposed experiment.

Table 4. 5. The performance evaluation analysis in the proposed experiment

Transaction Type	High Performance	Balanced Performance	Low Performance
Insertion	Isolated database approach	Shared database with separate schema approach	Shared database with shared schema approach
Number of tenants	Not Sensitive	Sensitive	Dramatically Sensitive
Case Evaluation	Best Case	Active with small number of tenants	Worst Case
Retrieving	Shared database with shared schema	Isolated database approach	Shared database with separate schema
Number of tenants	Sensitive	Sensitive	Dramatically Sensitive
Case Evaluation	Best case	Active with small number of tenants – not recommended for large number of tenants.	Decreases the performance drastically medium number of tenants
Deletion	Shared database with shared schema	Shared database with separate schema	Isolated database approach
Number of tenants	Not Sensitive	Not Sensitive	Sensitive
Case Evaluation	Best Case	Normal Case	Large number of tenants decrease performance in tangible manner

Consequently, the proposed experiments provide cloud users the ability to choose the suitable database segregation approach. Hence, from performance evaluation point of view we can give cloud users the ability to make their own tradeoffs in term of database segregation approach in order to select the appropriate environment based on their system's performance requirements.

4.3 Cost evaluation level

The main goal of this level is to evaluate the database segregation approach from the cost of computing resources point of view. Thus, in this experiment the evaluation criteria took into consideration the cost of CPU, memory, and disk storage. Therefore, the evaluation process focused on measuring the consumed percentage in CPU, the consumed space in memory, and the consumed disk size. Hence, each experiment was

tested over three phases which were the light phase with a 1000 records, the balanced phase with a 10000 records, and the hard phase with a 30000 records in each database after employing insertion and retrieving queries.

CPU cost evaluation level

In this section, we discussed the results of CPU cost based on the insertion and retrieving queries. Hence, CPU consumption was evaluated by measuring the query statistics using *DBCC FREEPROCCACHE* SQL statistics which is responsible to measure the cost of CPU while executing SQL queries. Table 4.6 shows the results of CPU cost after executing INSERT and SELECT queries.

Table 4. 6. The results of CPU cost after executing INSERT and SELECT queries

# of Records	Query Type	Isolated DB (100%)	Shared DB Separate Schema (100%)	Shared DB Shared Schema (100%)
1000	INSERT	25.45	9.17	19.32
	RETRIVE	14.9	22.3	47.9
10000	INSERT	40.3	25.1	28.11
	RETRIVE	23.39	39.8	62.79
30000	INSERT	64.9	54.9	55.47
	RETRIVE	31.97	37.98	79.8

The results showed that the CPU consumption had been affected due to the type of database segregation approach. Therefore, a CPU consumption gaps had been detected after applying the insertion query. The main theme of these gaps had been repeated for all experiment's phases. Thus, we found out that the isolated database segregation approach affected the consumption of CPU dramatically compared with other segregation approaches. On the other hand, with a small number of inserted records the

shared database with separate schema performed well in term of CPU consumption. From shared database with shared schema the consumption of CPU was increasing in regular manner, as well as, its effects on the CPU consumption for the large number of inserted records was approximately synchronized in results with shared database with separate schema. Figure 4.7 shows the results distribution of database segregation approaches in insertion queries for a variable number of records.

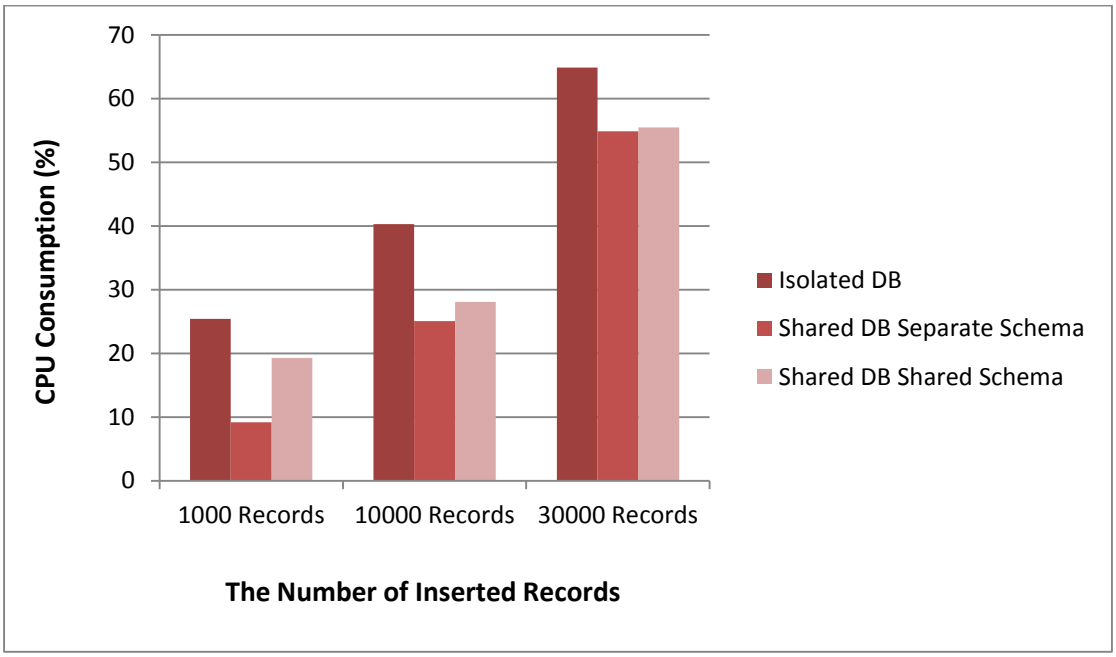

Figure 4. 7. The average of CPU cost for INSERT SQL query.

From retrieving point of view, an inverse case existed based on the results of SELECT query. Hence, we found out that the isolated database segregation approach had a positive effect on the consumption of CPU. A drastic gaps in CPU consumption was existed in all experiment's phases due to the use of shared database with shared schema segregation approach which showed a negative effect. In contrast, the use of shared database with separate schema approach achieved stable results after increasing the

number of retrieved records. Figure 4.8 shows the average results of CPU consumptions for SELECT query in the proposed experiment.

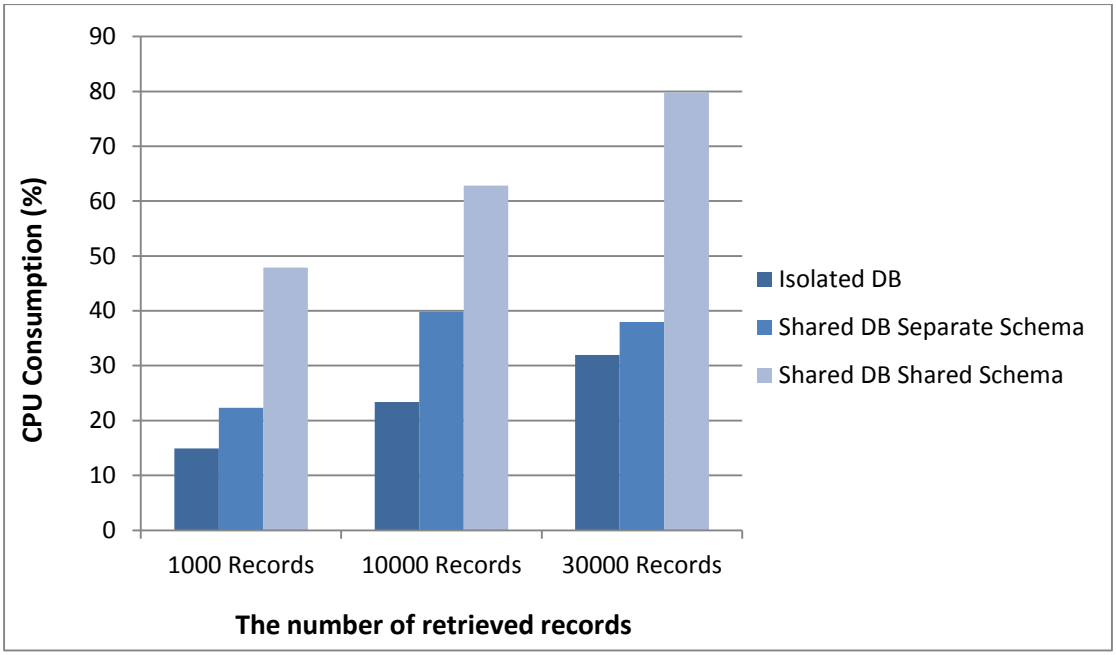

Figure 4. 8. The average of CPU cost for SELECT SQL query.

Disk cost evaluation

In this section, we discussed the effects of database segregation on the cost of disk (i.e. storage size) based on the insertion query over a variable number of records in the experiment. Table 4.7 shows the average results for each segregation approach for INSERT query.

Table 4. 7. The average results of INSERT query on disk cost based on database segregation approaches

Target # Records	Isolated DB (KB)	Shared DB Separate Schema (KB)	Shared DB Shared Schema (KB)
1000	2214.4	1579.6	996
10000	30806.4	2889.4	2468
30000	89980.8	7594.2	7169.8

Basically, the results of disk cost show a gap existence between isolated database approach and the other segregation approaches. Hence, in isolated database approach we found out a huge number of record's replication to cover each tenant's database. Furthermore, a normalized difference between shared databases with shared schema approach and shared database with separate schema approach. Figure 4.9 shows the storage cost after executing INSERT query based on database segregation approaches.

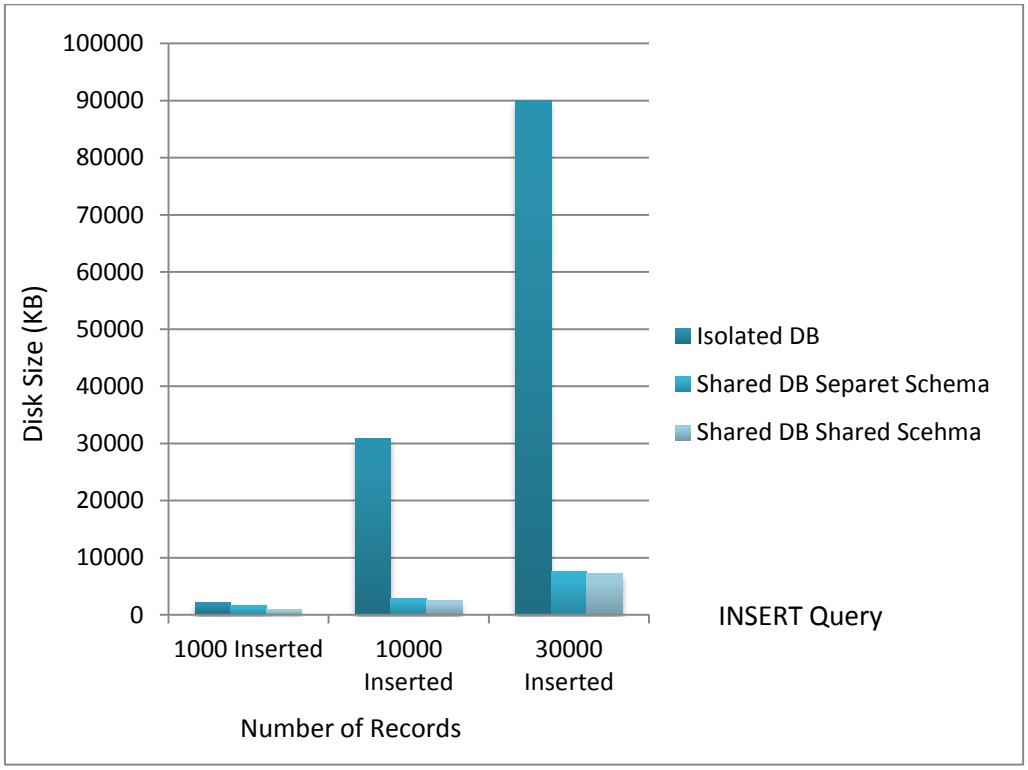

Figure 4. 9. The storage cost after executing INSERT query based in database segregation approaches

Conclusions

Basically, the cost evaluation in this level was concentrated on evaluating the consumption of CPU and disk storage using insertion and retrieving queries. Therefore, the proposed experiments were capable to provide the differences of using database segregation approaches in term of cost. Therefore, the CPU cost was measured using

query cost statistics functions provided MYSQL server DBMS, as well as, the cost of storage was measured using the normalized difference between the size of the system after and before executing the SQL queries. However, in some cases the number of records which were inserted or extracted did not have a direct effect based on the used segregation approach especially in large number of records cases (i.e. more than 30000 records). In contrast, in some cases the number of records were inserted or retrieved had a direct effect especially in small number of records (e.g. 1000 records). Table 4.8 shows the tradeoffs between database segregation approaches in term of CPU consumption and disk storage cost.

Table 4. 8. Tradeoffs table for insertion queries to identify the appropriate segregation approach

Criteria	Light Insertion	Balanced Insertion	Hard Insertion
Light CPU Consumption	SDBSPS	SDBSPS	SDBSPS + SDBSS
Balanced CPU Consumption	SDBSS	SDBSPS + SDBSS	SDBSPS + SDBSS
Hard CPU Consumption	IDB	IDB	IDB
Light Disk Storage Cost	SDBSPS/SDBSS/IDB	SDBSPS + SDBSS	SDBSPS + SDBSS
Balanced Disk Storage Cost	SDBSPS	SDBSS	SDBSPS
Hard Disk Storage Cost	IDB	IDB	IDB
Table Key: *(IDB : Isolated DB) (SDBSS: Shared DB Shared Schema) (SDBSPS: Shared DB Separate Schema)* ** Note: Underlined Approach → Worst Case			

4.4. Security evaluation level

Actually, several security metrics were found in literature to evaluate the security feature in computer systems. Thus, in this research security evaluation was based on

the number of attributes were encrypted using AES algorithm in the proposed system. Furthermore, by evaluating the number of encryption times for each database segregation approach, we can measure the security in term of protecting tenant's data from internal and external attacks. In this context, control parameter was assigned to detect the accuracy of system's results. Hence, we evaluated the performance of a system that was encrypted one, two and three times using AES algorithm. Transactions were used in the proposed experiment were about retrieving the same contents from three systems contains the same data and differ in the used database segregation approach. Table 4.9 shows the average results of retrieving the contents transaction's time of the proposed systems that were encrypted several times.

Table 4. 9. SELECT transaction time in several systems that were designed and employed based on database segregation approaches and was encrypted several times

# of Encryption Times	**Isolated DB (ms)**	**Shared DB Separate Schema (ms)**	**Shared DB Shared Schema (ms)**
One Time	38283.4	24299.8	56947.2
Two Times	121537.6	131458.6	295366.6
Three Times	392786.8	691234.5	835215.1

The control parameter in the proposed experiment showed that the increasing number of encryption time number was reflected as increasing in transaction time for retrieving information in all systems that applied all segregation techniques. Thus, the control parameter proved that the experiment's results were reliable. Figure 4.10 shows the results of transaction time in systems that were encrypted using AES algorithm several times number.

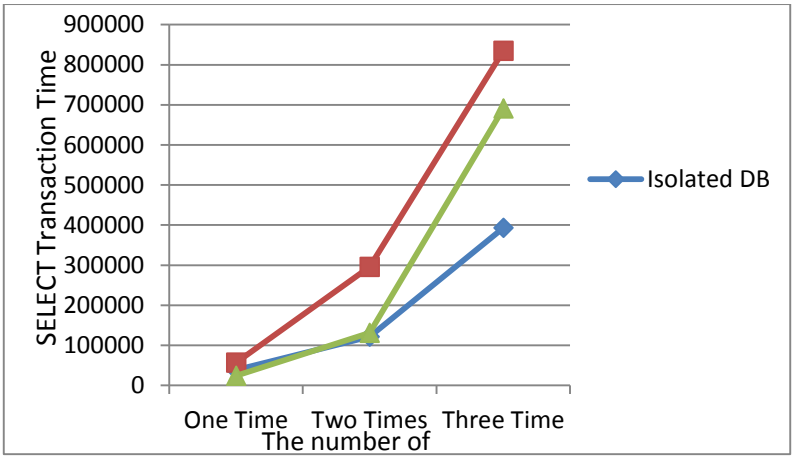

Figure 4. 10. The average transaction time results of system with several encryption time numbers

From security point of view, the more secure environment is the more times encrypted because of several key generation processes. The proposed experiments showed that the database segregation approaches affect the cost of security in term of encryption response time, and it effects was reflected when it extracted by measuring the performance of retrieving transactions. Therefore, the results showed that if we use the shared database with shared schema segregation approach the encryption performance would be the best compared with the other segregation approaches. However, using the shared database with shared schema segregation approach affected the encryption performance in a negative manner. A drastic encryption performance gaps existed in term of database segregation. Figure 3.11 shows the AES algorithm's encryption performance graph based system's database segregation approach.

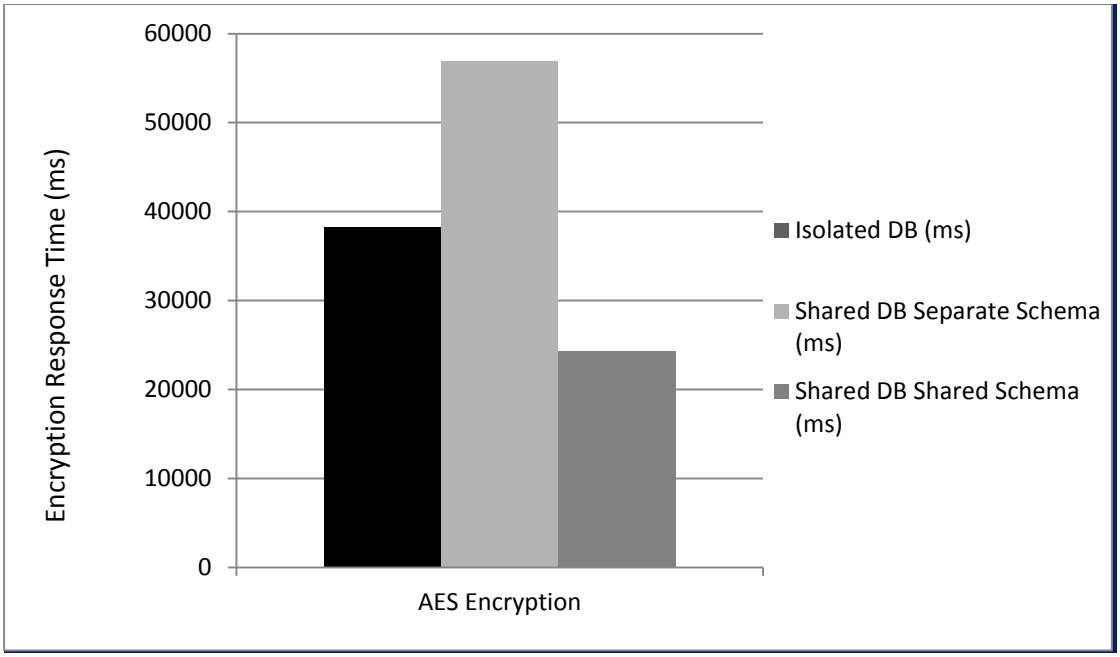

Figure 4. 11. AES algorithm's encryption performance graph for systems that was designed based on database segregation approaches

Consequently, we measured the decryption performance by applying a retrieving queries for systems that were encrypted for one time, two times, and three times in order to extract the suitable database segregation approach in multi-encryption systems. Therefore, the results showed that the isolated database segregation approach is the suitable approach for multi-encryption systems (i.e. in the more secured systems). The shared database with separate schema was performing well with one encryption systems. However, by increasing the number of encryption times shared database with shared schema and shared database with separate schema had a negative effect in term of retrieving point of view. Figure 4.12 shows the performance analysis graph of multi-encryption based on database segregation approaches.

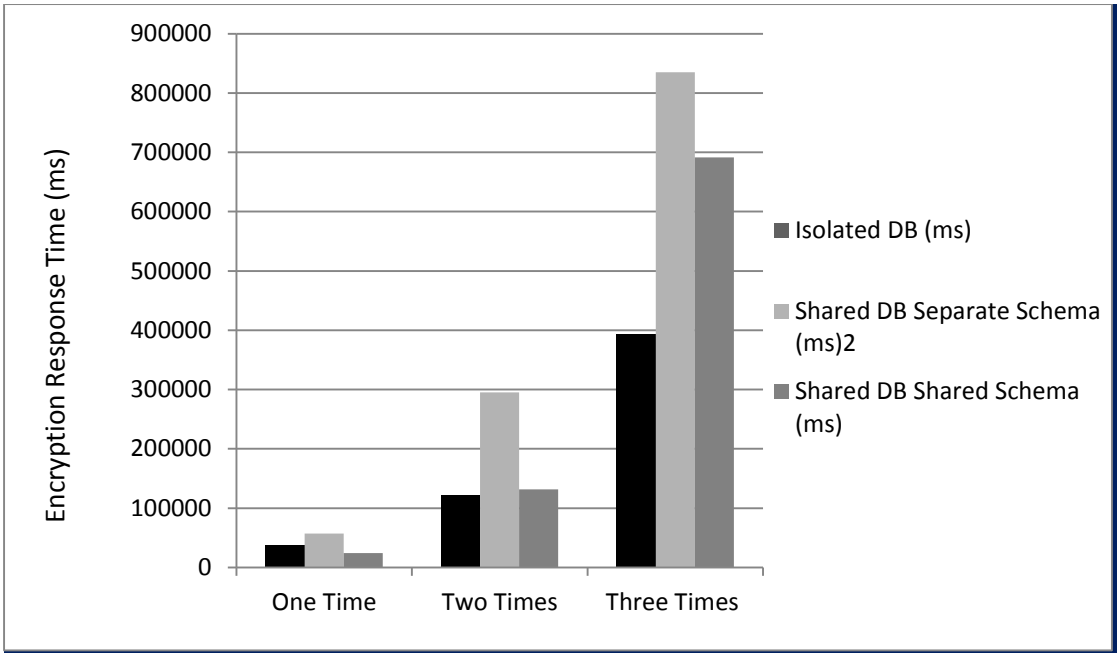

Figure 4. 12. Shows the performance analysis graph of multi-encryption based on database segregation approaches.

4.5. The proposed experimental results discussion

In this section, we discussed the experimental results from all proposed level were mentioned in this chapter, in order to define the benefits of this study and design a tradeoffs table that covers all cases in the study. Thus, CSP server side and cloud customers may benefit from this study in term of choosing the appropriate cloud specification in term of database segregation approaches.

Consequently, the experimental results showed that the tradeoffs take place from different point of views which are performance, cost, or security perspectives. Therefore, in this study the vertical integration was represented by transaction time in insertion, CPU consumption, and the encryption operation insecurity. On the other hand, the horizontal integration was represented by transaction time for retrieving, disk consumption, and security decryption operation. Table 4.10 shows the possible trade offs in this study.

Table 4. 10. The tradeoffs possibilities extracted from this study

		Vertical Integration in this study			
		Transaction Time (Insertion)	CPU Consumption (Insertion)	Disk Consumption (Insertion)	Security (Encryption)
Horizontal Integration in this study	Transaction Time (Retrieving)	Isolated DB Vs. Shared DB Shared Schema	Shared DB Separate Schema Vs. Shared DB Shared Schema	Shared DB shared Schema	Shared DB shared Schema
	CPU Consumption (Retrieving)	Isolated DB	Shared DB Separate Schema Vs. Isolated DB	Shared DB Shared Vs. Isolated DB	Shared DB Shared Schema Vs. Isolated DB
	Disk Consumption (Retrieving)	Isolated DB	Shared DB Separate Schema Vs. Isolated DB	Shared DB Shared Schema Vs. Isolated DB	Shared DB Shared Schema Vs. Shared DB Separate Schema
	Security (Decryption)	Isolated DB Vs. Shared DB Shared Schema	Shared DB Separate Schema	Shared DB Shared Schema Vs. Isolated DB Vs. Shared DB Separate Schema	Shared DB Shared Schema Vs. Shared DB Separate Schema
Notes: Colored boxes means the tradeoffs is possible based on customer requirements.					

4.6. Empirical Findings and Discussion:

In this section, we discussed the empirical findings of this study that summarizes the comparisons between database segregation approaches. Thus, in order to ease reading

the results we assigned A, B, and C to represent the isolated database approach, shared database with separate schema approach, and shared database with shared schema approach respectively. The discussion of the results was considered based on the gain and loss approach among database segregation approaches. Table 4.11 shows the percentage of each database segregation approach in term of query type for performance with regard to transaction time, cost with regard to CPU and disk storage, and security with regard to decryption.

Table 4.11. The database segregation approaches comparison table for research parameters

		No Records	Isolated DB	Shared DB Separate Sc	Shared DB Shared Sc
Performance(Transactio	Inserting	Light	100.00%	97%	95%
		Balanced	85%	100%	75%
		Hard	100%	80%	49%
	Retrieving	Light	58%	29%	100%
		Balanced	63%	28%	100%
		Hard	81%	39%	100%
	Deletion	Light	48%	30%	100%
		Balanced	79%	90%	100%
		Hard	77%	90%	100%
Cost (CPU)	Inserting	Light	36.00%	100%	48%
		Balanced	62%	100%	89%
		Hard	85%	100%	99%
	Retrieving	Light	100%	67%	31%
		Balanced	100%	59%	38%
		Hard	100%	84%	40%
Cost (Storag e)	Inserting	Light	45.00%	63%	100%
		Balanced	8%	31%	100%
		Hard	8%	94%	100%
Security	Retriving	Light	63.00%	100%	43%
		Balanced	100%	92%	41%
		Hard	100%	57%	47%

The analysis of the empirical findings was outlined as trade off cases in order to find the optimal case for user's environment requirements. Hence, in case 1, 2, and 3 the

analysis was concentrated on finding the optimal case in term of query type. However, in case 4 – case 16 the tradeoffs discussion was investigated with regard to performance transaction, cost of CPU, cost of disk storage, and security decryption.

CASE 1: The analysis of performance with regard to transaction time.

Selecting *A* in the light insertion give the system a chance to gain 5% than choosing *C*. In contrast, choosing *A* in the light retrieving let the system to loss 42%, as well as, losing 52% in deletion than choosing *C*. Therefore, we found out that choosing *A* for insertion based systems was represented the optimal solution. On the other hand, choosing *C* was represented the optimal solution for a system that implements the combination of insertion, retrieving, and deletion queries due to 42% gaining in retrieving and 52% gaining in deletion.

CASE 2: The analysis of cost in term of CPU

Selecting *B* in the light insertion give the system a chance to gain 64% than choosing *A*. In contrast, choosing *B* in the light retrieving let the system to loss 33% than choosing *A*. On the other hand, choosing *A* in the light retrieving let the system to gain 33% than choosing *B that lose 33%*. Therefore, we found out that choosing *B* was represented the optimal solution for a system that concerned with CPU cost in insertion and retrieving queries.

CASE 3: The analysis of cost in term of CPU and the cost of disk storage

Selecting *B* in the light insertion of CPU cost give the system a chance to gain 52% than choosing *C*. In contrast, choosing *B* in the light insertion of Storage cost let the system to loss 37% than choosing *C*. On the other hand, choosing *C* in the light insertion let the system to loss 52% in CPU cost and gain 37% in storage cost.

Therefore, we found out that choosing *B* was represented the optimal solution for a system that concerned with CPU and storage costs in insertion queries.

CASE 4: The analysis of Security and performance (transaction time)

Selecting *C* in the light retrieving of performance (transaction time) give the system a chance to gain 71% than choosing *B*. In contrast, choosing *C* in the light retrieving of security (retrieving) let the system to loss 57% than choosing *B*. On the other hand, choosing *B* in the light retrieving let the system to loss 71% in performance (transaction time) and gain 57% in security (retrieving). Therefore, we found out that choosing *C* was represented the optimal solution for a system that concerned with performance (transaction time) with security (retrieving) queries.

CASE 5: The analysis of storage cost with performance transaction time

Selecting *A* in the light insertion of performance (transaction time) give the system a chance to gain 5% than choosing *C*. In contrast, choosing *A* in the light insertion of storage cost let the system to loss 55% than choosing *C*. On the other hand, choosing *C* in the light insertion of performance transaction time let the system to loss 5% in performance (transaction time) and gain 55% in storage cost. Therefore, we found out that choosing *C* was represented the optimal solution for a system that concerned with performance (transaction time) with storage cost for insertion queries.

CASE 6: The analysis of performance (transaction time), CPU cost, and security in term of light retrieving queries.

Selecting C give the system the chance to gain 42 % compared with selecting A, and 71% compared with selecting B in term of performance (transaction time). By selecting A in term of CPU cost gives the system a chance to gain 33% compared with selecting

B, and 69% compared with selecting C. From security point of view, selecting B give the system chance to gain 37% compared with selecting A, and 57% compared with selecting C. Therefore, the suitable DB segregation approach between performance, cost, and security features is found in selecting C approach.

CHAPTER FIVE

Conclusions

5.1. Overview

This chapter summaries the conclusions of our work and suggested recommendations for using the suitable database segregation approach based with regard to performance, cost, and encryption / decryption security algorithm using insertion, deletion, and retrieving SQL queries.

5.2. Conclusions

According to the goals and experimental results, we can find that the used approach of database segregation influenced the performance in term of transaction time, the cost in term of consumption of CPU and consumption of disk storage, and the encryption / decryption operations in security. Therefore, two parameters were employed for the evaluation. Hence, the first parameter was used as a control parameter to check whether the proposed experimental results were in the right manner or not. The control parameter was the number of inserted or retrieved records. The second parameter was used is the primary one based on our study hypothesis which was the type of queries that were used on a running DB in cloud system. Therefore, we used INSERT, DELETE, and SELECT SQL queries.

Furthermore, the evaluation of transaction time, CPU consumption, disk consumption, and security encryption / decryption performance was successfully realized. In transaction time we chose to use the I/O statistics in SQL server DBMS. In CPU consumption evaluation we chose to use query cost statistics. In disk consumption we

chose to compute the normalized difference between the size of DB before and the size of DB after. In security we chose to evaluate the security through the number of encrypting database since the more secure is the less flexible and vice versa. In contrast, our evaluation criteria failed to measure the cost of memory for many reasons such as the total number of programs were cached in the environment as well as the graphics resources such as monitor card and other on start programs which were assigned to the memory and we cannot eliminate their effects in our calculations.

Furthermore, this study provide a tradeoffs table that let cloud users (i.e. server side or client side) to benefit from results in the decision making process of choosing the suitable database segregation approach that suits user's requirements and expectations.

5.3. Future research works

This research opens the door for finding the suitable database segregation approach to achieve the best results. Thus, the results of this study can be used to design an automatic cloud broker that has an intelligence features in order to assign the suitable database segregation approaches for cloud users. This research work open the door to make stress testing technique to evaluate the performance of such multitenant systems as well as evaluating memory performance using load testing approaches. Covering more domains using the proposed test experiment will enable more and more domain to be evaluated. Also, achieving more accurate results is still topic of continuous and constant research.

References

Al-Jahdali, H., Albatli, A., Garraghan, P., Townend, P., Lau, L.and Xu, J. (2014). Multi-Tenancy in Cloud Computing. *2014 IEEE 8th International Symposium on Service Oriented System Engineering*. PP. (344-351), Vol. 1, Issue. 2 doi:10.1109/sose.2014.50. ISBN: 978-1-4799-3616-8.

Almorsy, M., Grunday, J. and Ibrahim, A. (2012). SMURF: Supporting Multi-tenancy Using Re-Aspects Framework. *The 17th IEEE International conference on Engineering of Complex Computer Systems Swinburne*. (ICECCS 2012), Paris, France.

Bardiya, P., Gulhane, R., and Karade, P. (2014). Data Security using Hadoop on Cloud Computing. *International Journal of Computer Science and Mobile Computing*. Vol.3 Issue.4.

Jacob, D. and Allbuch, S. (2007). Ruminations on Multi-Tenant Databases. *Datenbanksysteme in Business, Technologie*. Vol. 1, Issue. 1, PP. (514-521).

Kun, M., Bo, Y and Ajith, A. (2012). A Template-based Model Transformation Approach for Deriving Multi-Tenant SaaS Applications. *Acta Polytechnica Hungarica*. Vol. 9, Issue. 2, 2012.

Mell, P. and Grance, T. (2012). The NIST of Cloud Computing. *NIST Special Publication*. PP. (145-148).

Pallavi B. and Jayarekha, P. (2014). Multitenancy in SaaS: A comprehensive Survey. *International Journal of Scientific & Engineering Research*. Vol. 5, Issue. 7, ISSN: 2229-5518.

Ru, J., Grundy, J., & Keung, J. (2014). Software engineering for multi-tenancy computing challenges and implications. *Proceedings of the International Workshop on Innovative Software Development Methodologies and Practices - InnoSWDev 2014.* PP. (1-10). doi:10.1145/2666581.2666585.

Saxena, V. and Pushkar, S. (2014). State-of-art in Storage Model using Encryption Technique for privacy preserving in Cloud Computing. *International Journal of Computer Science and Information Technologies (IJCSIT)*. Vol. 5, Issue. 1.

Schiller, O., Schiller, B., Brodt, A. and Mitschang, B. (2011). Native Support of Multi-tenancy in RDBMS for Software as a Service. *Applications of Parallel and Distributed Systems,IPVS ACM*. PP. (214-220). ISBN: 978-1-4503-0528-0/11/0003.

Seema, R. (2015). A Compound Algorithm Using Neural and AES for Encryption and Compare it with RSA and existing AES. *Journal of Network Communications and Emerging Technologies (JNCET)*. Vol. 3, Issue. 1. ISSN: 2395-5317

Soofi, A., Khan, M., Talib, R. and Sarwar, U. (2014). Security Issues in Saas Delivery Model of Cloud Computing. *International Journal of computer science and mobile computing (IJCSMC)*. Vol. 3, Issue. 3. ISSN: 2320-088X.

Sun, Y., Zhang, J., Xiong, Y. and Zhu G. (2014). Data Security and Privacy in Cloud Computing. *International Journal of Distributed sensor Networks*. Vol. 1, Article ID: 190903.