

STATUS OF THESIS

Title of thesis

DESIGN AND DEVELOPMENT OF KEY REPRESENTATION
AUDITING SCHEME FOR SECURE ONLINE AND DYNAMIC
STATISTICAL DATABASES

I ASIM ABDALLAH ELSHIEKH MOHAMMED AHMED

hereby allow my thesis to be placed at the Information Resource Center (IRC) of Universiti Teknologi PETRONAS (UTP) with the following conditions:

1. The thesis becomes the property of UTP
2. The IRC of UTP may make copies of the thesis for academic purposes only.
3. This thesis is classified as

Confidential

Non-confidential

If this thesis is confidential, please state the reason:

The contents of the thesis will remain confidential for _____ years.

Remarks on disclosure:



Signature of Author

Endorsed by



Signature of Supervisor

Permanent address: Khartoum North
P. O. Box 175, Khartoum – SUDAN

Name of Supervisor
Assoc. Prof. Dr. P. P. D. DOMINIC

Date : 6/9/2010

Date : 6/9/2010

UNIVERSITI TEKNOLOGI PETRONAS

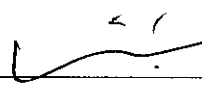
DESIGN AND DEVELOPMENT OF KEY REPRESENTATION AUDITING
SCHEME FOR SECURE ONLINE AND DYNAMIC STATISTICAL DATABASES

by

ASIM ABDALLAH ELSHIEKH MOHAMMED AHMED

The undersigned certify that they have read, and recommend to the Postgraduate
Studies Programme for acceptance this thesis for the fulfilment of the requirements
for the degree stated.

Signature:



DR. P. P. D. DOMINIC
Associate Professor
Computer & Information Sciences Department
Universiti Teknologi PETRONAS
Bandar Seri Iskandar, 31750 Tronoh
Perak Darul Ridzuan, MALAYSIA


Main Supervisor:

Assoc. Prof. Dr. P. P. D. DOMINIC

Signature:

Co-Supervisor:

Signature:



Dr Mohd Fadzil B Hassan
Head
Computer & Information Sciences Department
Universiti Teknologi PETRONAS

Head of Department:

Dr. MOHD FADZIL BIN HASSAN

Date:

6/9/10

DESIGN AND DEVELOPMENT OF KEY REPRESENTATION AUDITING
SCHEME FOR SECURE ONLINE AND DYNAMIC STATISTICAL DATABASES

by

ASIM ABDALLAH ELSHIEKH MOHAMMED AHMED

A Thesis

Submitted to the Postgraduate Studies Programme

as a Requirement for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTER AND INFORMATION SCIENCES

UNIVERSITI TEKNOLOGI PETRONAS

BANDAR SERI ISKANDAR,

PERAK

SEPTEMBER 2010

DECLARATION OF THESIS

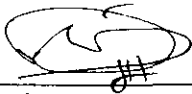
Title of thesis

DESIGN AND DEVELOPMENT OF KEY REPRESENTATION
AUDITING SCHEME FOR SECURE ONLINE AND DYNAMIC
STATISTICAL DATABASES

I ASIM ABDALLAH ELSHIEKH MOHAMMED AHMED

hereby declare that the thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTP or other institutions.

Witnessed by



Signature of Author



Signature of Supervisor

Permanent address: _____

Date : 6/9/2010

Name of Supervisor
Assoc. Prof. Dr. P.P.D.DOMINIC

Date : 6/9/2010

DR. P. D. D. DOMINIC
Associate Professor
Computer & Information Sciences Departmen-
Universiti Teknologi PETRONAS
Bandar Seri Iskandar, 31750 Tronoh
Perak Darul Ridzuan, MALAYSIA

ACKNOWLEDGEMENTS

First and foremost, I would like to thank Allah the Almighty for the innumerable gifts that He has granted me, for guiding me along in completing this work and for giving me an opportunity to undergo higher education.

I would like to express my total appreciation to the people for their support and for guiding me along in completing this thesis. Very special thanks to my supervisor Dr. Dhanapal Durai Dominic for his invaluable time and guidance on this research throughout the three years.

I would like to express my gratitude to Associate Prof. Dr. Hamidah Ibrahim, Department of Computer Science, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM), who was abundantly helpful and offered invaluable assistance, support and guidance. Deepest gratitude is also due to Prof. Alan Oxley, Dr. Azween Bin Abdullah, Dr. Rohiza Binti Ahmad and Mr. Low Tang Jung for their invaluable comments and guidance.

I would like to express my utmost appreciation to my dearest wife, father, daughter, brothers and sisters for their encouragement throughout my whole educational life. I could have not completed my degree without their continuous and immeasurable support.

Finally, huge thanks also go to my friends for their consistent support, help and encouragement.

ABSTRACT

A statistical database (SDB) publishes statistical queries (such as sum, average, count, etc.) on subsets of records. Sometimes by stitching the answers of some statistics, a malicious user (snooper) may be able to deduce confidential information about some individuals. When a user submits a query to statistical database, the difficult problem is how to decide whether the query is answerable or not; to make a decision, past queries must be taken into account, which is called SDB auditing. One of the major drawbacks of the auditing, however, is its excessive CPU time and storage requirements to find and retrieve the relevant records from the SDB.

The key representation auditing scheme (KRAS) is proposed to guarantee the security of online and dynamic SDBs. The core idea is to convert the original database into a key representation database (KRDB), also this scheme involves converting each new user query from a string representation into a key representation query (KRQ) and storing it in the Audit Query table (AQ table). Three audit stages are proposed to repel the attacks of the snooper to the confidentiality of the individuals. Also, efficient algorithms for these stages are presented, namely the First Stage Algorithm (FSA), the Second Stage Algorithm (SSA) and the Third Stage Algorithm (TSA). These algorithms enable the key representation auditor (KRA) to conveniently specify the illegal queries which could lead to disclosing the SDB.

A comparative study is made between the new scheme and the existing methods, namely a cost estimation and a statistical analysis are performed, and it illustrates the savings in block accesses (CPU time) and storage space that are attainable when a KRDB is used. Finally, an implementation of the new scheme is performed and all the components of the proposed system are discussed.

ABSTRAK

Pangkalan data statistik (SDB) menghasilkan permintaan statistik (seperti jumlah, purata, kiraan dan sebagainya) terhadap sebahagian rekod. Adakalanya dengan menggabungkan keputusan yang didapati daripada beberapa statistik, pengguna yang tidak sah (pengintip) mungkin boleh mencungkil maklumat sulit tentang seseorang individu. Apabila pengguna menghantar permintaan kepada pangkalan data statistik, masalah utama ialah bagaimanakah cara untuk mengenal pasti bahawa permintaan tersebut boleh dijawab atau tidak; untuk membuat keputusan, sejarah permintaan perlu diambil kira. Salah satu daripada kelemahan audit, ialah berlebihan masa diperlukan untuk unit pemprosesan pusat dan storan yang diperlukan untuk menyimpan dan menjalankan proses akumulasi log. Wakil kunci skim audit (KRAS) telah dicadangkan untuk menjamin keselamatan pangkalan data statistik secara online dan dinamik. Cara utama ialah dengan menukarkan pangkalan data yang asal kepada wakil kunci pangkalan data (KRDB), skim ini juga menukarkan setiap permintaan pengguna baru di dalam bentuk ayat ke dalam bentuk permintaan wakil kunci (KRQ) dan menyimpannya ke dalam jadual Audit Permintaan (AQ Table). Tiga peringkat audit disarankan untuk mengatasi serangan daripada pengintip terhadap maklumat sulit individu. Tambahan pula, kecekapan algoritma untuk setiap peringkat telah dipersembahkan, yang dinamakan Algoritma Peringkat Pertama (FSA), Algoritma Peringkat Kedua (SSA) dan Algoritma Peringkat Ketiga (TSA). Algoritma-algoritma ini membenarkan auditor wakil kunci (KRA) dengan mudah mengenalpasti permintaan yang tidak sepatutnya yang dikhuatiri boleh membawa kepada pendedahan pangkalan data statistik. Perbandingan kajian telah dijalankan di antara skim baru dengan cara yang sedia ada sekarang, yang dipanggil anggaran kos dan analisis statistik telah dijalankan, dan kami telah berjaya menunjukkan penjimatan di dalam blok penggunaan (CPU time) dan ruang penyimpanan yang diperlukan sewaktu KRDB digunakan. Akhir sekali, pelaksanaan skim yang baru ini telah dijalankan dan semua komponen system baru ini dibincangkan.

In compliance with the terms of the Copyright Act 1987 and the IP Policy of the university, the copyright of this thesis has been reassigned by the author to the legal entity of the university,
Institute of Technology PETRONAS Sdn Bhd.

Due acknowledgement shall always be made of the use of any material contained in, or derived from, this thesis.

© Asim Abdallah Elshiekh, 2010
Institute of Technology PETRONAS Sdn Bhd
All rights reserved.

TABLE OF CONTENTS

STATUS OF THESIS.....	i
APPROVAL PAGE.....	ii
TITLE PAGE.....	iii
DECLARATION OF THESIS	iv
ACKNOWLEDGEMENTS.....	v
ABSTRACT.....	vi
ABSTRAK.....	vii
COPYRIGHT PAGE.....	viii
TABLE OF CONTENTS.....	ix
LIST OF TABLES.....	xii
LIST OF FIGURES	xiii
LIST OF ABBREVIATIONS.....	xv

Chapter

1. INTRODUCTION.....	1
1.1 Introduction.....	1
1.2 Database Security.....	1
1.2.1 Threats to Databases.....	2
1.2.2 Control Measures.....	3
1.2.2.1 Access Control.....	3
1.2.2.2 Inference Control	4
1.2.2.3 Flow Control	4
1.2.2.4 Data Encryption	5
1.3 Statistical Database Security.....	5
1.3.1 Methods of Attacks.....	6
1.3.1.1 Small and Large Query Set Attacks.....	6
1.3.1.2 Tracker Attacks	7
1.3.1.3 Insertion and Deletion Attacks.....	9
1.3.2 Overview of Solution Approaches	9
1.3.3 Attribute Classification.....	11
1.4 Problem Statement.....	11
1.5 Objectives.....	12
1.6 Methodology	12
1.7 Scope of Research.....	13
1.8 Research Contributions.....	14
1.9 Limitations of Research	15
1.10 Structure of Thesis	15
2. LITERATURE REVIEW.....	16

2.1	Introduction.....	16
2.2	Online Auditing.....	16
2.3	Offline Auditing.....	28
2.4	Summary	33
3.	RESEARCH METHODOLOGY AND CONVERSION METHOD OF THE KEY REPRESENTATION AUDITING SCHEME	35
3.1	Introduction.....	35
3.2	Research Methodology.....	35
3.2.1	Developing a New Scheme.....	37
3.2.1.1	Audit Stages	37
3.2.2	Comparative Study between the New Scheme and the Existing Methods.....	38
3.2.2.1	Cost Estimation.....	38
3.2.2.2	Statistical Analysis.....	39
3.2.3	Implementation of the New Scheme	39
3.3	Conversion Method of the Key Representation Auditing Scheme.....	39
3.3.1	Statistical Database Model	40
3.3.2	Key Representation Database (KRDB)	40
3.3.3	Key Representation Query (KRQ)	42
3.3.3.1	The Relational Operators in KRQ.....	46
3.3.3.2	The Logical Operators in KRQ.....	46
3.3.3.3	Examples of KRQs.....	47
3.3.4	Audit Query Table (AQ table).....	47
3.3.5	Examples of SDB and its Corresponding KRDB.....	48
3.3.5.1	The First Example.....	48
3.3.5.2	The Second Example.....	51
3.3.5.3	The Third Example.....	54
3.4	Summary	57
4.	AUDIT STAGES OF THE KEY REPRESENTATION AUDITING SCHEME.....	59
4.1	Introduction.....	59
4.2	Audit Stages	59
4.2.1	The First Audit Stage.....	60
4.2.2	The Second Audit Stage	67
4.2.2.1	The Second Audit Stage: Case 1.....	67
4.2.2.2	The Second Audit Stage: Case 2.....	70
4.2.3	The Third Audit Stage	75
4.2.3.1	The Third Audit Stage: Case 1.....	75
4.2.3.2	The Third Audit Stage: Case 2.....	77
4.3	Summary	83
5.	COST ESTIMATION FOR THE KEY REPRESENTATION AUDITING SCHEME.....	84
5.1	Introduction.....	84
5.2	Cost Estimation.....	84
5.3	Parameters of the Cost Estimation.....	85
5.4	Comparisons between the KRDB and the Original Database.....	86

5.4.1 Case Study I – One Data Attribute.....	86
5.4.2 Case Study II – More than One Data Attribute.....	93
5.5 Summary	99
6. STATISTICAL ANALYSIS FOR THE KEY REPRESENTATION DATABASE AND THE ORIGINAL DATABASE.....	100
6.1 Introduction.....	100
6.2 Comparisons between the KRDB and the Original Database.....	100
6.2.1 Record Size: the Original Database Vs the KRDB.....	101
6.2.1.1 Comparing the Means in terms of Record Size	103
6.2.1.2 Comparing the Variances in terms of Record Size	105
6.2.2 Number of Blocks: the Original Database Vs the KRDB.....	106
6.2.2.1 Comparing the Means in terms of Number of Blocks	108
6.2.2.2 Comparing the Variances in terms of Number of Blocks.....	110
6.2.3 Linear Search: the Original Database Vs the KRDB	111
6.2.3.1 Comparing the Means in terms of Linear Search.....	113
6.2.3.2 Comparing the Variances in terms of Linear Search ..	115
6.2.4 Binary Search: the Original Database Vs the KRDB.....	116
6.2.4.1 Comparing the Means in terms of Binary Search	118
6.2.4.2 Comparing the Variances in terms of Binary Search..	119
6.2.5 Sorting: the Original Database Vs the KRDB	120
6.2.5.1 Comparing the Means in terms of Sorting	122
6.2.5.2 Comparing the Variances in terms of Sorting.....	124
6.3 Summary	125
7. CONCLUSION AND FUTURE DIRECTIONS	127
7.1 Research Summary.....	127
7.2 Research Contributions	129
7.3 Future Directions.....	130
7.4 Research Conclusions	130
PUBLICATIONS.....	131
REFERENCES	133
APPENDIX A.....	140
APPENDIX B	145
APPENDIX C	155

LIST OF TABLES

Table 3.1: The Relational Operators in KRQ.....	46
Table 3.2: The Logical Operators in KRQ.....	46
Table 3.3: Example I - The Original Database D	49
Table 3.4: Example I - The Key Representation Database (KRDB) D^1	50
Table 3.5: Example I - Examples of User Queries Converted into KRQs.....	51
Table 3.6: Example II - The Original Database D	52
Table 3.7: Example II - The Key Representation Database (KRDB) D^1	53
Table 3.8: Example II - Examples of User Queries Converted into KRQs	54
Table 3.9: Example III - The Original Database D	56
Table 3.10: Example III - The Key Representation Database (KRDB) D^1	57
Table 5.1: The Parameters of the Cost Estimation.....	86
Table 5.2: Case Study I - the KRDB Vs the Original Database	89
Table 5.3: Case Study II - the KRDB Vs the Original Database	95
Table 6.1: Record Size: R Vs R^1	101
Table 6.2: Number of Blocks: b Vs b^1	106
Table 6.3: Linear Search: Original DB Vs KRDB.....	111
Table 6.4: Binary Search: Original DB Vs KRDB	116
Table 6.5: Sorting: Original DB Vs KRDB	121
Table 6.6: Statistical Analysis Results	126

LIST OF FIGURES

Figure 1.1: Query-set-size Control	7
Figure 1.2: Data Perturbation Approach.....	10
Figure 1.3: Output Perturbation Approach	10
Figure 1.4: Query Restriction Approach.....	11
Figure 2.1: Deletion that Causes a False Alarm	18
Figure 2.2: Deletion that Causes Disclosure of Secret Information	19
Figure 2.3: A Related Set.....	20
Figure 3.1: Research Stages	36
Figure 3.2: The Original Database Conversion Algorithm.....	41
Figure 3.3: The User Query Conversion Algorithm	43
Figure 3.4: Statistical Database Model for the Key Representation Auditing Scheme	45
Figure 4.1: The First Stage Algorithm (FSA).....	62
Figure 4.2: Flow Chart for the First Stage Conditions (FSCs).....	63
Figure 4.3(a): The Second Stage Algorithm (SSA).....	73
Figure 4.3(b): The Second Stage Algorithm (SSA).....	74
Figure 4.4(a): The Third Stage Algorithm (TSA).....	80
Figure 4.4(b): The Third Stage Algorithm (TSA)	81
Figure 4.5: Flow Chart for the Key Representation Auditing Scheme.....	82
Figure 5.1(a): Case Study I - Linear Search: the Original Database Vs the KRDB ..	90
Figure 5.1(b): Case Study I - Binary Search: the Original Database Vs the KRDB .	91
Figure 5.1(c): Case Study I - Sorting: the Original Database Vs the KRDB.....	92
Figure 5.2(a): Case Study II - Linear Search: the Original Database Vs the KRDB.	96

Figure 5.2(b): Case Study II - Binary Search: the Original Database Vs the KRDB	97
Figure 5.2(c): Case Study II - Sorting: the Original Database Vs the KRDB	98
Figure B.1: The Main Screen	146
Figure B.2: Employee Data Screen	147
Figure B.3: Browse Employee Data Screen	148
Figure B.4: Browse KRDB Data Screen	149
Figure B.5: User Query Screen.....	150
Figure B.6: Permitted Query Result	151
Figure B.7: Prevented Query Message	152
Figure B.8: Browse Query Result Screen.....	153
Figure B.9: Browse AQ Table Screen.....	154

LIST OF ABBREVIATIONS

AQ table	Audit Query table
Bio	Biological
BSc	Bachelor of Science
CE	Civil Engineering
CPU	Central Processing Unit
CS	Computer Science
DB	Database
DBMS	Database Management System
Dept	Department
df	Degrees of Freedom
EE	Electrical Engineering
EMP	Employee
F	Female
FSA	First Stage Algorithm
FSCs	First Stage Conditions
GP	Grade-Point
HIPAA	Health Insurance Portability and Accountability Act.
KRA	Key Representation Auditor
KRAS	Key Representation Auditing Scheme
KRDB	Key Representation Database
KRQ	Key Representation Query
KS	Knowledge Space
M	Mail
ME	Mechanical Engineering
MSc	Master of Science
PE	Petroleum Engineering
PhD	Doctor of Philosophy

Psy	Psychology
SAT	Scholastic Aptitude Test
SDB	Statistical Database
SQL	Structured Query Language
SSA	Second Stage Algorithm
SSN	Social Security Number
TSA	Third Stage Algorithm
TSC	Third Stage Condition

CHAPTER 1

INTRODUCTION

1.1 Introduction

In this chapter, an introduction to database security is given, its threats and its control measures as well. Also, an introduction to statistical database security, methods of attacks, overview of solution approaches and attribute classification. In addition, the problem statement, objectives, methodology, scope, contributions and limitations of this research are discussed.

1.2 Database Security

The increasing development of information technology in the past few years has led to the widespread use of computer systems in various public and private organizations, such as banks, universities, companies, hospitals, libraries and so on. The increased reliability now offered in hardware and software technologies, coupled with the continuous reduction of costs, the increasing professional expertise of information specialists and the availability of support tools, have all contributed to encourage the widespread use of computing services. This has meant that more data than ever before is now stored and managed by computer systems, or rather by the tools and techniques capable of supporting and meeting these application requirements. Such requirements have been largely satisfied by database technology employing *Database Management Systems (DBMSs)* [1].

Although the increasingly widespread use of both centralized and distributed databases has proved necessary to support business functions, it has also posed serious problems of data security. The term **security** refers to the protection of the

database against unauthorized access, either intentional or accidental. In fact, damage in a database environment does not only affect a single user or application but rather the whole information system. Advances in information processing techniques (tools and languages) aimed at a simplification of human/machine interfaces have served to make databases available to different types of user; consequently more serious security problems arise. Therefore, in computer-based information systems, technologies, tools and procedures concerning security are essential both to assure system continuity and reliability and to protect data and programs from intrusions, modifications, theft and unauthorized disclosure [1][2].

1.2.1 Threats of Databases

Database security is the mechanisms that protect the database against intentional or accidental threats. A threat can be defined as any situation or event, whether intentional or accidental, that may adversely affect a system and consequently the organization. A threat may be caused by a situation or event involving a person, action or circumstance that is likely to bring harm to an organization. The harm may be tangible, such as loss of hardware, software or data, or intangible, such as loss of credibility or client confidence. The problem facing any organization is to identify all possible threats which result in the loss or degradation of some or all of the following commonly accepted security goals: integrity, availability and confidentiality [1]-[3].

- **Loss of integrity:** Database integrity refers to the requirement that information be protected from improper modification. Modification in data includes creation, insertion, modification, changing the status of data and deletion. Integrity is lost if unauthorized changes are made to the data by either intentional or accidental acts. If the loss of system or data integrity is not corrected, continued use of the contaminated system or corrupted data could result in inaccuracy, fraud or erroneous decisions.
- **Loss of availability:** Database availability refers to making objects available to a human user or a program to which they have a legitimate right.

- **Loss of confidentiality:** Database confidentiality refers to the protection of data from unauthorized disclosure. The impact of unauthorized disclosure of confidential information can range from violation of the Data Privacy Act to the jeopardization of national security. Unauthorized, unanticipated or unintentional disclosure could result in loss of public confidence, embarrassment or legal action against the organization.

To protect databases against these types of threats, it is common to implement four kinds of control measures: access control, inference control, flow control and encryption. The following section discusses each of these.

1.2.2 Control Measures

There are four main control measures that are used to provide security of data in databases [1][3]. They are as follows:

- (1) Access Control.
- (2) Inference Control.
- (3) Flow Control.
- (4) Data Encryption.

1.2.2.1 Access Control

A security problem common to computer systems is that of preventing unauthorized persons from accessing the system itself, either to obtain information or to make malicious changes in a portion of the database. The security mechanism of a DBMS must include provisions for restricting access to the database system as a whole. This function is called access control and is handled by creating user accounts and passwords to control the login process by the DBMS.

1.2.2.2 Inference Control

Statistical databases (SDBs) are used to provide statistical information or summaries of values based on various criteria. For example, a database for population statistics may provide statistics based on age groups, income levels, household size, education levels and other criteria. Statistical database users such as government statisticians or market research firms are allowed to access the database to retrieve statistical information about a population but not to access the detailed confidential information about specific individuals. Security for statistical databases must ensure that information about individuals cannot be accessed. It is sometimes possible to deduce or infer certain facts concerning individuals from queries that involve only summary statistics on groups; consequently, this must not be permitted either. This problem is called statistical database security. The corresponding control measures are called inference control measures which aim at protecting data from indirect detection.

1.2.2.3 Flow Control

Flow control regulates the distribution or flow of information among accessible objects. A flow between object X and object Y occurs when a program reads values from X and writes values into Y. Flow controls check that information contained in some objects does not flow explicitly or implicitly into less protected objects. Thus, a user cannot get indirectly in Y what he or she cannot get directly in X. Most flow controls employ some concept of security class; the transfer of information from a sender to a receiver is allowed only if the receiver security class is at least as privileged as the sender's.

1.2.2.4 Data Encryption

A final control measure is data encryption, which is used to protect sensitive data (such as credit card numbers) that are transmitted via some type of communications network. Encryption can be used to provide additional protection for sensitive portions of a database as well. The data are encoded using some coding algorithm. An unauthorized user who accesses encoded data will have difficulty deciphering it, but

authorized users are given decoding or decrypting algorithms (or keys) to decipher the data.

1.3 Statistical Database Security

A statistical database (SDB) is a database that is used for statistical queries (such as sum, average, count, etc.) on subsets of the database entities [1]. Many government agencies, businesses and nonprofit organizations need to collect, analyze and report data about individuals in order to support their planning activities. SDBs therefore contain confidential information such as income, credit ratings, type of disease or test scores of individuals. Such data are typically stored online and analyzed using sophisticated database management systems (DBMSs). On one hand, such database systems are expected to satisfy user requests of aggregate statistics related to non confidential and confidential attributes. On the other hand, the system should be secure enough to guard against the ability of a malicious user (snooper) to infer any confidential information about any individual represented in the database [4]. Although users are only allowed to access the statistical information from an SDB, malicious users (snoopers) can deduce confidential information about some individuals by stitching the answers of some legal queries [5][6].

Protecting an SDB means preventing and avoiding statistical inference. Inference in SDB means the possibility of obtaining confidential information on single entity, by taking advantage of (sequences of) statistical queries issued against a set of entities stored in the SDB. When confidential information about individuals is obtained, the database is said to be disclosed [1].

SDBs may be *online* or *offline*. In an *online* SDB, users get real-time responses to their statistical queries. Whereas, in an *offline* SDB, users do not know when their statistics will be processed, making disclosure more difficult. Also, SDBs may be *static* or *dynamic*. *Static* SDBs do not change during their lifetime (namely, no insertion or deletion operations occur), and possible changes would give rise to a new static database. In contrast, *dynamic* SDBs can change continuously. Protecting a dynamic SDB is more complex, since variations in the database state provide additional information to snoopers (malicious users) [1][4].

Hence, our new auditing scheme, namely the key representation auditing scheme (KRAS), is proposed to protect online and dynamic SDBs from being disclosed.

1.3.1 Methods of Attacks

There are several kinds of threats and disclosure techniques [7], the most important of these threats are as follows:

- (1) Small and large query set attacks.
- (2) Tracker attacks.
- (3) Insertion and deletion attacks.

1.3.1.1 Small and Large Query Set Attacks

It is easy to compromise a database that releases statistics about small and large query sets. To protect against this kind of attack, statistics based on small or large query sets must be restricted (see Figure 1.1).

Query-Set-Size Control:

A user query q is permitted only if:

$$n \leq |q| \leq N-n$$

where, N and $|q|$ are the database size and query-set-size, respectively.

And $n \geq 0$ is a parameter of the database.

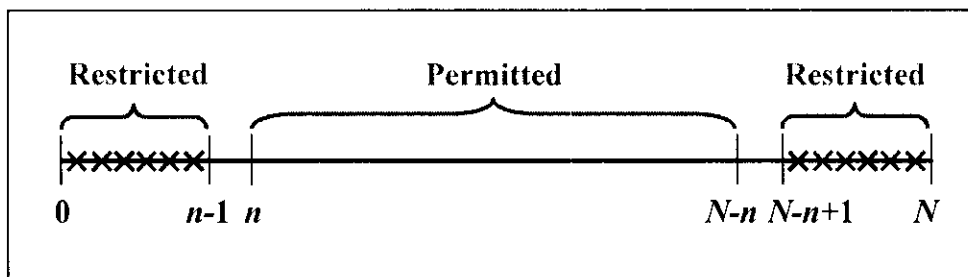


Figure 1.1: Query-set-size Control

1.3.1.2 Tracker Attacks

The basic idea is to pad small query sets with enough extra records to put them in the allowable range, and then subtract out the effect of the padding. The different types of trackers include: *individual trackers*, *general trackers* and *double trackers* [7][8].

(i) Individual Trackers:

Suppose that a user knows an individual I who is uniquely characterized by a formula C . If C can be decomposed into the product $C = C_1.C_2$, such that $\text{count}(C_1, \overline{C_2})$ and $\text{count}(C_1)$ are both permitted:

$$n \leq \text{count}(C_1, \overline{C_2}) \leq \text{count}(C_1) \leq N - n$$

The pair of formulas $(C_1, C_1, \overline{C_2})$ is called the individual tracker of I .

Individual Tracker Compromise:

Let $C = C_1.C_2$ be a formula uniquely identifying individual I , and let $T = C_1, \overline{C_2}$. Using the permitted queries $q(C_1)$ and $q(T)$ the restricted query $q(C)$ can be computed from

$$q(C) = q(C_1) - q(T) \tag{1.1}$$

(ii) General Trackers:

A general tracker is any characteristic formula T such that

$$2n \leq |T| \leq N - 2n$$

The queries $q(T)$ are always answerable because $|T|$ is well within the allowable range $[n, N-n]$.

General Tracker Compromise:

Let T be a general tracker and let $q(C)$ be a restricted query. First calculate

$$q(All) = q(T) + q(\overline{T})$$

If $|C| < n$, $q(C)$ can be computed from

$$q(C) = q(C + T) + q(C + \overline{T}) - q(All) \quad (1.2)$$

And if $|C| > N - n$, $q(C)$ can be computed from

$$q(C) = 2q(All) - q(\overline{C} + T) - q(\overline{C} + \overline{T}) \quad (1.3)$$

If the user does not know whether the query set is too small or too large, Formula (1.2) can be tried first; if the queries on the right-hand side are permitted, the user can proceed; otherwise, Formula (1.3) can be used. Thus, $q(C)$ can be computed with at most six queries.

(iii) Double Trackers:

A double tracker is a pair of characteristic formulas (T, U) for which

$$T \subseteq U,$$

$$n \leq |T| \leq N - 2n, \text{ and}$$

$$2n \leq |U| \leq N - n$$

Double Tracker Compromise:

Let $q(C)$ be a restricted query, and let (T, U) be a double tracker. If $|C| < n$, $q(C)$ can be computed from

$$q(C) = q(U) + q(C + T) - q(T) - q(\overline{C.T.U}) \quad (1.4)$$

And if $|C| > N - n$, $q(C)$ can be computed from

$$q(C) = q(\overline{U}) - q(\overline{C} + T) + q(T) + q(\overline{\overline{C.T.U}}) \quad (1.5)$$

Thus, $q(C)$ can be computed with at most seven queries.

1.3.1.3 Insertion and Deletion Attacks

Dynamic databases that allow insertions and deletions of records are vulnerable to additional attacks. A query-set-size restriction of n can be subverted if records can be added to the database. If $|q| < n$, then dummy records satisfying the characteristic formula of q are added to the database; if $|q| > N-n$, then dummy records not satisfying the characteristic formula of q are added.

1.3.2 Overview of Solution Approaches

There are many inference control methods proposed to protect various database systems. Those methods for SDBs can be classified under three general approaches: *data perturbation*, *output perturbation* and *query restriction*. *Data perturbation approach* (see Figure 1.2) introduces noise in the data. The original SDB is typically transformed into a modified (perturbed) SDB, which is then made available to researchers. The *output perturbation approach* (see Figure 1.3) perturbs the answer to user queries while leaving the data in the SDB unchanged. While *query restriction approach* (see Figure 1.4) imposes extra restriction on queries, which includes query-set-size control, query-set-overlap control, auditing, cell suppression and partitioning [4]-[6].

Auditing of an SDB involves keeping up-to-date logs of all queries made by each user and constantly checking for possible compromise whenever a new query is issued. Auditing has the advantages such as allowing the SDB to provide users with unperturbed responses that will not be disclosed. It has long been believed that auditing is an effective tool for protection [7].

This work focuses on the query restriction approach, which prevents malicious inferences by denying illegal queries. In particular, this research deals with auditing online and dynamic statistical databases problem.

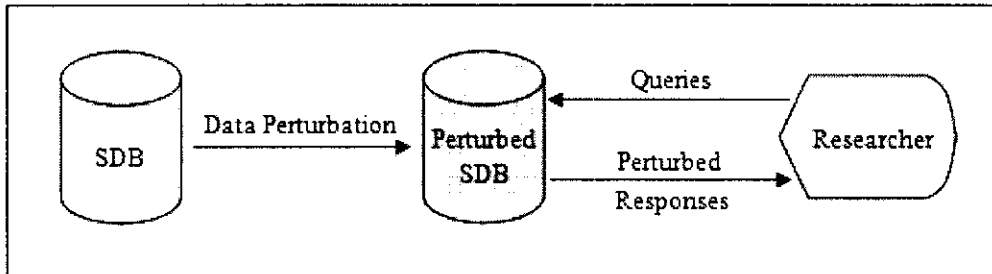


Figure 1.2: Data Perturbation Approach

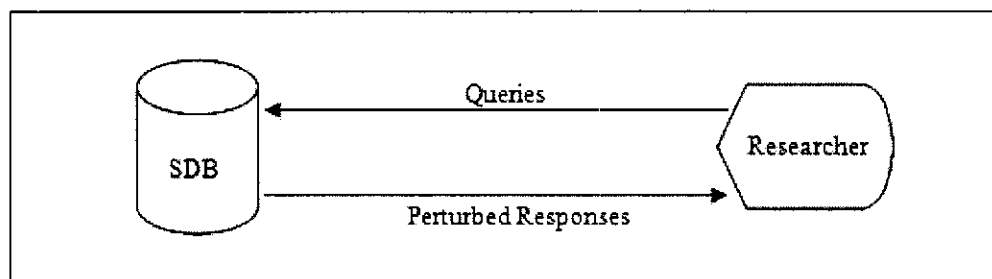


Figure 1.3: Output Perturbation Approach

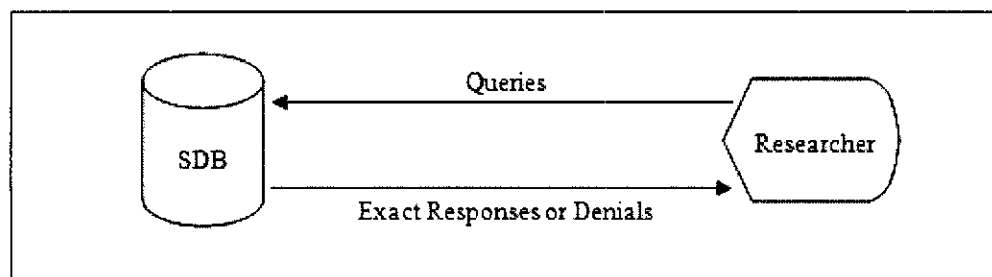


Figure 1.4: Query Restriction Approach

1.3.3 Attribute Classification

Each attribute can be classified into two types: *category* and *data attributes*. The *category attributes* are used to identify and select records, and each of them contains a specific domain. For example, assume that the domains of the category attributes Gender and Department are {M, F} and {CS, EE, ME, CE, PE}, respectively. While the *data attributes* hold other information, usually numerical, for which some statistical queries may be desired such as Salary, Score, Income, etc [9][10].

1.4 Problem Statement

In this research the problem of protecting an SDB is examined to prevent an SDB from being disclosed, through the attacks of the snoopers (malicious users) to the confidentiality of the individuals, and no sequence of legal queries is sufficient to infer protected information about any individual. When a user submits a query to statistical database, the difficult problem is how to decide whether the query is answerable or not; to make a decision, past queries must be taken into account, which is called SDB auditing [11]. One of the major drawbacks of the auditing is its excessive CPU time and storage requirements to find and retrieve the relevant records from the SDB.

The problem is that statistics contain vestiges of the original information. By correlating different statistics, a malicious user (snooper) may be able to deduce confidential information about some individuals. Also, the problem in SDB protection is the achievement of a compromise between the privacy needs of an individual and the right of organizations to know and process precise and accurate information with:

- less storage space, namely less SDB record size and number of blocks. And
- less CPU time to retrieve the relevant records from the SDB, namely less block accesses to perform linear search, binary search and sorting.

This entails the right to release statistical information, while at the same time assuring that confidential information about the individuals represented in the SDB is maintained.

1.5 Objectives

Query auditing is an effective strategy for guarding the confidentiality of the individual in the statistical database [12], that is because it provide users with precise and accurate answers (unperturbed responses). Since a detailed examination of the inference problem reveals that we have not yet arrived at a general and acceptable solution [13], the objectives of this research can be summarized as:

- To develop a new scheme for auditing online and dynamic SDBs.
- To guarantee the security of online and dynamic SDBs by preventing illegal queries which could lead to disclosing the SDB.
- To provide precise and accurate responses.
- To reduce CPU time and storage space during query processing.

1.6 Methodology

This section provides all the sequence steps that have been followed in order to satisfy the research objectives. In general this research can be divided into three stages.

In the first stage, a new scheme for auditing online and dynamic SDBs is developed, namely the key representation auditing scheme (KRAS). The core idea is to convert the original database, which is in both string and numerical representations, into a key representation database (KRDB). Also, this scheme involves converting each new user query from a string representation into a key representation query (KRQ) and storing it in the Audit Query table (AQ table). Three audit stages are proposed to repel the attacks of snoopers to the confidentiality of the individuals. Efficient algorithms for these stages are presented, namely the First Stage Algorithm (FSA), the Second Stage Algorithm (SSA) and the Third Stage Algorithm (TSA).

These algorithms enable the key representation auditor (KRA) to conveniently specify the illegal queries which could lead to disclosing the SDB.

In the second stage, a comparative study is made between the new scheme and the existing methods, namely a cost estimation and a statistical analysis are performed, and this study illustrates the savings in block accesses (CPU time) and storage space that are attainable when a KRDB is used. The cost estimation comparisons between the KRDB and the original database are performed in terms of number of blocks, linear search, binary search and sorting. The statistical analysis is performed to compare between means and variances of the original database and the KRDB populations. The size of the sample drawn from each population is 13. The statistical analysis tests between the two populations are provided in terms of record size, number of blocks, linear search, binary search and sorting to examine whether the KRDB is better than the original database or not.

In the final stage, the implementation of the new scheme is performed and all the components of the proposed system are discussed.

1.7 Scope of Research

This research concentrates on protecting online and dynamic statistical databases (SDBs) with the least CPU time and storage space, as possible, during query processing. It intends to overcome one of the major drawbacks of the auditing, namely its excessive CPU time and storage requirements to store and process the accumulated logs. This research proposes the key representation auditing scheme (KRAS) to guarantee the security of online and dynamic SDBs. The proposed scheme is considered as an effective scheme to repel the attacks of snoopers (malicious users) to the confidentiality of the individuals. Moreover, the proposed scheme shows vast improvement in terms of block accesses (CPU time) and storage space that are attainable when a KRDB is used.

1.8 Research Contributions

The contributions of this research can be summarized as:

- The new scheme guarantees the security of online and dynamic SDBs. The three audit stages could prevent the SDB threats such as individual trackers, general trackers, double trackers and insertion and deletion attacks. Moreover, it could prevent the following three new types of threats which have not been discussed previously:
 - (i) Stitching two answerable queries using two different category attributes.
 - (ii) Hiding an unanswerable key representation query (KRQ), which satisfies the first stage conditions (FSCs) with one of the previous KRQs, inside the parts of the user query.
 - (iii) Hiding a repeated unanswerable KRQ, which does not satisfy the third stage condition (TSC), inside the parts of the new user query.
- The new scheme provides precise and accurate responses, while most of the previous works resort to estimate the value of the new response according to the distribution of the previous answered queries.
- The new scheme, which depends directly on the key representation database (KRDB), saves CPU time and storage space compared to the original database. All schemes proposed by previous works depend directly on the original database.

1.9 Limitations of Research

In this research, our proposed scheme, namely the key representation auditing scheme (KRAS), includes only auditing count and sum statistical queries. The other statistical queries (or aggregate functions) such as average, min, max and median are not included in our proposed scheme.

1.10 Structure of Thesis

This thesis is structured in seven chapters. The first chapter gives an introduction to database security, its threats and its control measures. Also, this chapter gives an

introduction to statistical database security. In addition, the problem statement, objectives, methodology, scope, contributions and limitations of this research are discussed. Chapter two provides related works and mentions review of literature. Methodology of this research and conversion method of our proposed scheme, namely the key representation auditing scheme (KRAS), are illustrated in chapter three. Chapter four discusses the three audit stages of our proposed scheme, which is proposed to protect online and dynamic SDBs from being disclosed. In chapter five, a cost estimation for the proposed scheme is performed, and this research illustrate the savings in block accesses (CPU time) and storage space that are attainable when a key representation database (KRDB) is used. In chapter six, statistical analysis is performed to compare between means and variances of the original database and the KRDB populations. Comparisons are made between the KRDB and the original database in terms of record size, number of blocks, linear search, binary search and sorting to examine whether the KRDB is better than the original database or not. The last chapter is the conclusion which concludes this research and it does include some recommendations for further research directions.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Problems with security of SDBs have been increasing in concern recently. This chapter provides related work. Related work can be divided into online and offline auditing. In the *offline auditing* problem, the auditor is given a series of queries and exact answers and the goal is to decide whether a privacy breach has occurred ex post facto or not. In the *online query auditing* problem, users get real-time responses to their statistical queries: given a series of queries that have already been posed, their corresponding answers and a new query; deny the answer if privacy may be breached or give the true answer otherwise.

2.2 Online Auditing

Problems with security of SDBs have been increasing in concern recently [14]. The problem with *online query auditing* is: given a sequence of queries q_1, \dots, q_{t-1} having already been asked and the corresponding answers a_1, \dots, a_{t-1} , each a_j being either the true answer to the query q_j or “denied” for $j = 1, \dots, t-1$. Being given a new query q_t , prevent the answer if confidentiality might be breached or provide the true answer otherwise.

Audit Expert is a practical approach based on auditing, which was proposed in [15]. The approach maintains a matrix used for auditing the history of users’ queries and detects all of the possible breaches. The columns of the binary matrix represent database entities while, the rows represent the users’ queries that have already been answered. When a new query is issued, the matrix is updated. A row with all zeros

except for an i th column is an indicator that exact disclosure of the confidential attribute of the corresponding entity is possible. Whereby, the answer to the new query should be prevented.

According to Chin's scheme [15], the SDB consists of N individuals x_i , $1 \leq i \leq N$. Each individual x_i is assumed to have a single protected numerical attribute value, and each answered query reveals a set of individual records $\{x_i, x_m, x_n, \dots\}$. Therefore, each answered query can be represented by a vector (a_1, a_2, \dots, a_N) , where $a_i = 1$ if x_i is accessed in this query, and $a_i = 0$ otherwise. The users' knowledge space KS is the vector space which is spanned by the set of vectors of answered queries AQ . Formally, KS has the following properties:

- 1) If $q^1 \in AQ$, then $q^1 \in KS$.
- 2) If $q^1 \in KS$, then $b \cdot q^1 \in KS$; b is a real number.
- 3) If $q^1, q^2 \in KS$, then $q^1 + q^2 \in KS$.
- 4) Nothing else is in KS .

A maximal set of non-redundant vectors of AQ can represent KS .

It is noticed in [15] that the vectors in KS are linear independent. Therefore, the number of rows cannot exceed the number of columns in KS . The SDB is disclosed if there exists a vector of the form $(0, \dots, 0, 1, 0, \dots, 0)$ in KS .

Unfortunately, Chin's scheme faces problems with space explosion if the SDB is dynamically updated. In Chin's scheme, when an individual inserted to an SDB, a new corresponding column is inserted to the KS for this individual. Since the new individual has not yet been queried, all entries of the new column are zeros. On the other hand, for the protection of the individual information, when an individual is deleted, the corresponding column, called the *dangling column*, cannot be immediately removed from the KS matrix.

If the dangling columns are removed immediately to reduce the size of KS , the deletion may cause both *false alarms* and *security disclosure*. A false alarm is raised when a vector with a single "1" is found in the audit matrix but the corresponding

individual is not disclosed. For example in Figure 2.1, the individual x_3 is deleted from SDB. If the corresponding column c_3 in the KS is removed, the audit matrix reports that x_5 is breached and the SDB is compromised. In fact, x_5 is still unbreached at this time. Thus, a false alarm has been raised.

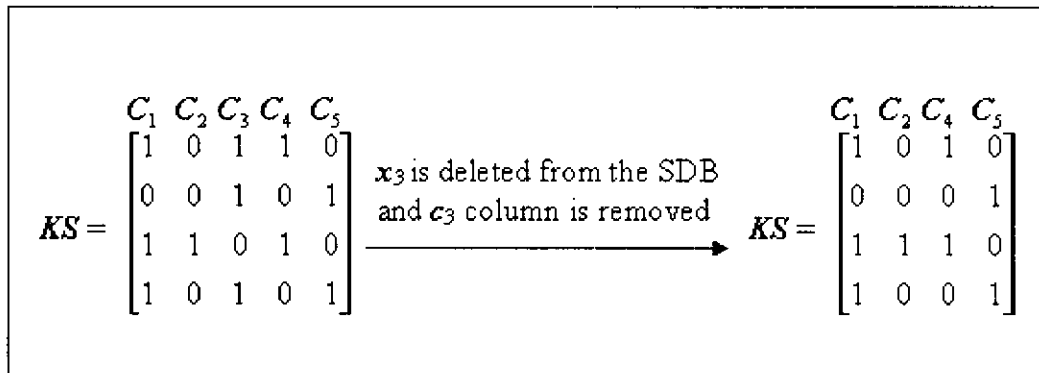


Figure 2.1: Deletion that Causes a False Alarm [5][6]

On the other hand, security disclosure occurs when the audit matrix does not have any vector with a single “1”, but the secret of an individual is disclosed. For example in Figure 2.2, the individual x_4 is deleted from the SDB. It seems reasonable to also remove the corresponding column c_4 . However, by removing this column, disclosure of confidential information will occur.

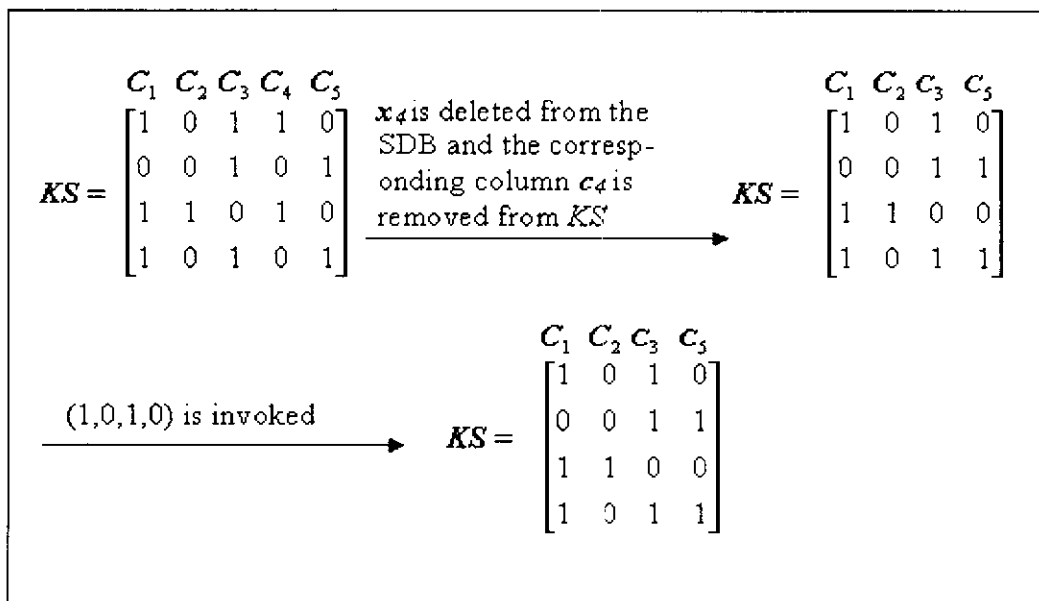


Figure 2.2: Deletion that Causes Disclosure of Secret Information [5][6]

Assume that a new answered query, $(1,0,1,0)$, is invoked in the KS after the deletion. The KS will be checked by the audit scheme and considered as redundant answerable query, which is the same as r_1 . As a result, KS remains unchanged and the query is answered. This in turn, breaches the confidential information of the deleted individual x_j is disclosed.

The two examples above demonstrate that when an individual is deleted from the SDB, the corresponding column in the KS cannot be arbitrarily removed. Therefore, the size of KS will be continuously expanded without any limit when the individuals of a finite-size SDB are dynamically inserted, deleted or updated. It is possible therefore, to have a large KS for a small SDB. Substantial memory and CPU time are subsequently wasted in handling the columns. It is not efficient to check the entire KS matrix for every query, when the number of the rows and the columns in the KS is large. To handle this problem, Chin imposes the restriction on the scheme that it can only be used in static SDBs. As a result, usage of this scheme is limited.

The authors in [5][6] proposed an algorithm to reduce the size of the KS . Chin's scheme can be enhanced so that it can be used in a dynamic SDB using this algorithm. In order to guarantee the security of an SDB, for the most part, all dangling columns cannot be arbitrarily removed from the KS . However, if the deletion will not cause a false alarm or security disclosure, it is possible to delete some of the dangling columns. The removable part of the audit matrix is called a *related garbage set* [5][6].

The authors in [5][6] defined the directly and indirectly related relations in a related garbage set as follows: in an audit matrix, an entry can only be either '1' or '0'. A column and a row are *directly related* if their shared entry is '1'. Indirectly related relation can be defined recursively. A column/row is *indirectly related* to a column/row if a directly related column/row of the former is directly/indirectly related to the latter. If a column/row is directly or indirectly related to another column/row, then they are *related*. Otherwise, they are *unrelated*. All related columns and rows form a *related set*. All elements of a related set are related to each other, and no element outside of the related set can be related to any element of the set. For example, in Figure 2.3, r_1 and r_4 are directly related to c_1 ; r_1 is indirectly related to r_4 ; therefore, $\{c_1, c_3, c_4, r_1, r_2, r_4\}$ is a related set.

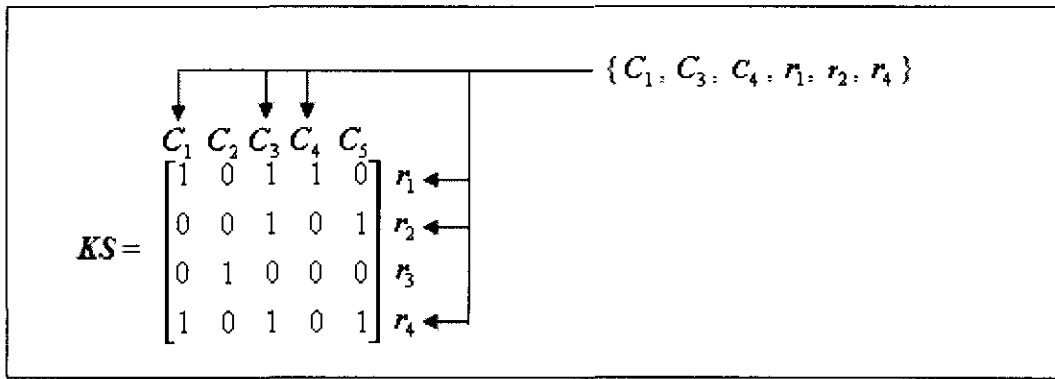


Figure 2.3: A Related Set [5][6]

Being unrelated to other columns and rows, the garbage columns and rows of a related set can be removed without affecting the subsequent security of the audit matrix. An algorithm FINDING_GARBAGE based on the concept that garbage columns and rows are related was proposed in [6]. Whenever an individual is deleted, the algorithm is able to locate all the columns and rows related to the new dangling column. If these columns are also dangling, then these columns and rows are all garbage and can be safely removed.

FINDING_GARBAGE is effective in reducing the memory requirement and improving the performance of Chin's audit scheme. However, algorithm itself also introduces overhead for the deletion of individuals from an SDB.

An implementation of the auditing strategy to avoid both exact and approximate disclosure was presented in [12]. The key data structure of their study is a graphical summary of answered queries in the form of a query map. Since the size of a query map could be exponential in the number of answered queries, a query restriction criterion was introduced to make every query map a graph. An auditing procedure on such a graph was also presented and the computational issues connected with its implementation were discussed.

In [12], q is assumed to be the current query and it is not sensitive. To decide if q can be answered safely, the query-system must hypothesize a user (the "snooper") is knowledgeable and able to disaggregate the values of answered queries. Explicitly, such a user must be assumed to have such a semantic competence that he correctly guesses which queries are overlapping and which ones are not. Furthermore, he would

have to have such a computational competence that he knows how to find out all the aggregate data that are “inferable” from the values of answered queries. The authors in [12] argued that in order to repel the attacks of the snooper, the query-system will decide that q can be safely answered if and only if either

- The value of q is inferable from the values of previously answered queries, or
- There is no sensitive query among those whose values are inferable from the value of q and the values of previously answered queries.

The security attacks delivered by the snooper are successfully repelled by the query-system if its inference model is at least as powerful as the snooper’s one. Based on all previously answered queries and the overlapping relationships among their query-sets, the authors in [12] set up a system Γ of linear constraints which allows it to identify every query whose value is inferable. In the case of a large underlying database, the number of variables in Γ is always less (and is often far less) than the size of the database; however, it may be exponential in the number of previously answered queries, so that after answering a large number of queries, the query system will spend a lot of time deciding if the current query can be safely answered or not. To overcome this difficulty, the following query-restriction criterion was introduced by the authors in [12]. Given a set Q of answered queries and a new non sensitive query q with query-set R , a necessary condition for q to be safely answerable is that either the value of q be inferable from Q or, for every two queries q_1 and q_2 from Q , $R_1 \cap R_2 \cap R$ be empty, where R_i is the query-set of q_i ($i = 1, 2$). So, the number of variables in Γ is $O(|Q|^2)$. The resulting auditing procedure is as follows:

Input: A set Q of answered queries and a new query q , all of the same type; the query-set R of q .

Output: A Boolean variable *safe* which is TRUE if and only if q can be safely answered given the values of all the queries in Q .

(Initialization) Set *safe* := FALSE.

(Phase I) If the value of q is inferable from the answers to the queries in Q , then set *safe* := TRUE and Exit.

(Phase II) If q violates the query-overlap constraint, then Exit.

(Phase III) If no query whose value is inferable from $Q \cup \{q\}$ is sensitive, then set $safe := \text{TRUE}$. Exit.

The computational issues connected with the implementation of the above auditing procedure were discussed by the authors in [12]. It was shown that the problems related to Phases I and II could be solved efficiently, and that the problem related to Phase III is provably intractable.

The authors in [10] focused on sum-queries which have a response variable of non negative real type. They proposed a compact representation of answered sum-queries, called an information model in “normal form”. This model allows the query-system to decide whether the value of a new sum-query can be safely answered or not. If not, then the query system will issue the range of feasible values of the new sum-query consistent with previously answered sum-queries.

The authors in [10] argued that the confidentiality of a response variable σ can be attacked either (in a direct way) by an intrusive sum-query or (in an indirect way) by a non intrusive sum-query whose value on D , combined with the responses to previously answered sum-queries on D , leads to an accurate estimate of the total of σ for some category that is sensitive in D . In the latter case, the sum-query is called *tricky*.

An SDB can be made secure when a new instance D is created, for each confidential attribute σ the sensitive categories in D are identified and each of them is assigned a fixed non negative number which is its *protection level*. Such a category S is considered *protected* at a particular time if its protection level is less than the width of the interval of the feasible values for the total of σ for S that are permitted by the responses to previously answered sum-queries. This interval is known as the *feasibility range*. The authors in [10] argued that if the current sum-query Q is intrusive or tricky, then a non informative response to Q will be given by the query-system, by issuing the feasibility range for Q . This makes deciding whether Q is intrusive or not easy, since it is sufficient to check the presence of the category specified by Q in the list of the categories that are sensitive in D for σ . However,

deciding whether Q is tricky or not requires “auditing” the previously answered sum-queries on D with response variable σ and comparing the protection level assigned to S with the width of the feasibility range for the total of σ for S . Q can be *safely* answered and the value of Q will be issued if each sensitive category is protected. A special case occurs when the value of Q is uniquely determined by previously answered sum-queries, that means Q is *evaluable*; then, Q is neither intrusive nor tricky and it can be safely answered.

In much of the previous work [4][15]-[19], the auditing technique was applied assuming that the snooper also knows the query-set of each answered sum-query. Thus, enabling the snooper to write down an equation for each answered sum-query, whose unknowns represent the unknown values of the response variable for the tuples in the query-set. Consequently, the size of the snooper’s model is proportional to the size of the instance D of the SDB, which may contain a very large number of tuples [20]-[25]. On the other hand, it is not realistic to hypothesize that the snooper knows the query-sets of the answered sum-queries. It has been suggested by some authors that by working with categories instead of query-sets, the snooper’s model could be made independent of the size of the instance of the SDB.

The authors in [10] stated that in order to repel the attacks of the snooper, the query- system will make use of its own information model, which essentially is the same as the snooper’s model and will be constructed incrementally as the value of a new-sum query is issued. However, such an information model might suffer from certain drawbacks like redundancy, so the authors in [10] proposed a procedure for getting a “compact” representation of the information model that the drawbacks are missing from. This model is called a *normal form*. Finally, the authors addressed the question of whether or not a new sum-query can be safely answered by using a normal form of the current information model.

The authors in [10] discussed that answering this question has raised some computational problems such as recognizing evaluable sum-queries, updating the information model and computing a feasibility range. The solutions of these problems depend on the response variable data type. Standard algebraic methods can be used to solve all of these problems efficiently if the response variable data type is of a real

type. However, if it is of a non negative type, then linear-programming or integer linear-programming methods can be resorted depending on the specific data type. In general, if the response variable is of a non negative-integer type, it is extraordinarily difficult from a computational viewpoint. A general theory has not yet been developed to solve this problem.

The authors' work [10], only considered the case where the response variable is of a non negative real type. Therefore, a natural approach consists in resorting to standard linear-programming algorithms for example the simplex method. Unfortunately, the majority of these algorithms are not polynomial. Furthermore, the existing polynomial linear-programming algorithms such as the ellipsoid method have bad performances in practice. Therefore, in order to solve the computational problems raised by the security of the SDB, it is convenient to make a minimal use of standard linear-programming algorithms, so "there is considerable interest in finding alternative techniques".

In [26], the online query auditing was considered. It was illustrated how denials that depend on the answer to the current query may leak information and the notion of simulatability to tackle this problem was introduced. Simulatable algorithms for auditing sum queries and max queries were provided. In addition, a probabilistic notion of disclosure was introduced and an algorithm for auditing sum queries over real-valued data drawn uniformly from a bounded range under this notion was provided.

The authors' work goal was to design algorithms that never allow a sequence of queries that breaches the data, regardless of the actual data. Essentially, a denial is never required for this formulation. The authors in [26] called this type of auditing *query monitoring* or, simply *monitoring*. In terms of utility however, monitoring may be too restrictive as it may prevent queries that do not breach confidentiality. This concern may try to be answered by constructing auditing algorithms where every query is checked with respect to the data set, and a denial occurs only when an 'unsafe' query occurs.

The authors in [26] discussed the following question: *Can an offline auditing algorithm directly solve the online auditing problem?* In the traditional algorithms literature [27]-[30], it is stated that an offline algorithm can always be used to solve the online problem — the only penalty is in the efficiency of the resulting algorithm. In clarification of the question for the auditing context, to determine whether to answer q_t , can we consult the data set for the true answer a_t and then run an offline algorithm to determine if providing a_t would lead to a breach?

Surprisingly, this question was answered negatively by the authors [26]. Their main reason being that is denials leak information. The authors in [26] stated the following simple example: suppose that the underlying data set is real-valued and that a query is denied only if some value is fully compromised. Suppose that the snooper poses the first query $\text{sum}(x_1, x_2, x_3)$ and the auditor answers 15. Suppose also that the snooper then poses the second query $\text{max}(x_1, x_2, x_3)$ and the auditor prevents the answer. The denial tells the snooper that if the true answer to the second query were given then some value could be uniquely deduced. Note that $\text{max}(x_1, x_2, x_3) \not\leq 5$ since then the sum could not be 15. Further, if $\text{max}(x_1, x_2, x_3) > 5$ then the query would not have been prevented since no value could be uniquely deduced. Consequently, $\text{max}(x_1, x_2, x_3) = 5$ and the attacker learns that $x_1 = x_2 = x_3 = 5$ — a confidentiality breach of all three entries. The problem here is the reduction of the space of possible consistent solutions resulting from query denials. Moreover, this reduction is not explicitly accounted for in existing offline auditing algorithms.

Posterior probabilities are computed by the new auditing algorithm by utilizing existing randomized algorithms. The authors in [26], to guarantee simulatability, made sure that the auditing algorithm does not access the data set while deciding whether to allow the newly posed query q_t (particularly, by not computing the true answer to q_t). Instead, the auditor draws many data sets according to the underlying distribution, assuming the previous queries and answers, and then computes an expected answer a'_t and checks whether revealing it would breach confidentiality for each of the randomly generated data sets. If the data set is not breached then the query is answered, otherwise the query is prevented.

The authors [26] believed that this definition overcomes some of the limitations. However, the current definition does not ensure that the confidentiality of a group of individuals or any function of a group of individuals is kept secure. Also, their model assumes that the data set is static, but in practice data is inserted and deleted over time.

The online query auditing problem was considered in [31]. Auditors were constructed for max queries and bags of max and min queries in settings for both the partial and full disclosure. The authors' partial disclosure setting algorithm involves a novel application of probabilistic inference techniques.

In much of the work done previously [32]-[36], compromise corresponds to the notion of full disclosure and occurs when the confidential data of any individual can be exactly determined. This is called classical compromise [37]-[41]. The authors in [26] introduced probabilistic compromise for bounded range data where a significant change in the snooper's confidence about the range of a data point constitutes a confidentiality disclosure. This is related to the notion of partial disclosure. A new algorithm was introduced by the authors in [31] for auditing max queries and bags of max and min queries under this definition. While in the case of classical compromise algorithms are known for auditing sum, average, min and max queries separately, auditing of combinations of these queries is hard to do. In the case of classical compromise, they presented an auditor for bags of max and min queries.

For example, suppose a snooper asks for $q_1 = \max\{x_a, x_b, x_c\}$ and receives the answer 9. Later the snooper asks for $q_2 = \max\{x_a, x_b\}$. If the answer to q_2 is less than 9, then the snooper can infer that x_c must be 9 and q_2 should be prevented. If however, the answer is exactly 9, answering q_2 would not leak information under the classical definition of compromise. In this case, if the auditor does look at the answer to q_2 when deciding to prevent, a denial would immediately imply that x_c must be 9 and confidentiality is disclosed. Therefore, the auditor should ignore the true answer to the current query when making a decision. In reality, the snooper should be able to "simulate" the auditor and predict on his own when queries will be prevented. This would ensure that confidentiality is never disclosed. Thus, the algorithms that the authors in [31] looked for had to be online and simulatable. With classical

compromise, it is enough that the auditor determine if there is any possible answer to the current query that is consistent with past queries that could lead to breach. With probabilistic compromise, it is enough that the auditor determine if breach would occur in a large fraction of data sets drawn from the original distribution D based on past query answers.

2.3 Offline Auditing

In the problem of *offline auditing*, the auditor is given an offline set of queries q_1, \dots, q_t and true answers a_1, \dots, a_t and must decide whether a compromise of confidentiality has occurred ex post facto or not.

The authors in [42] proposed an auditing method based on the offline auditing subcube queries model used in [43]. It views an SDB as a function f from strings of k bits to the positive and negative integers with the keys being the domain of f . A query is always of length k bits; for example, for $k = 7$ a possible query could be $0^{*}11^{*}0$, with s 0's and 1's (in this case $s = 4$) and the $*$ standing for "do not care". The result of a query Q that is of length k and has s 0's and 1's is given by:

$$\sum_{\text{Key } i \text{ matches } Q} f(i)$$

In the clinic database, for example, the key could consist of 16 bits xxxxywwwwwzzzzzz as follows:

- xxx is a code for the physician,
- y is a code for the patient's gender (0 = Male, 1 = Female),
- wwwww is a code for the patient's age,
- zzzzzz is a code for the type of disease.

Thus, the query $***1101110111001$ would represent the sum of all female patients, independent of which physician they are with, of age 46 who have a disease type 111001.

The author in [42] stated that the amount of information gained by posing a query Q is given by:

$$\left\{ \begin{array}{ll} \log\left(\frac{L}{\min(|C|, L-|C|)}\right) & , \text{ if } |C| \leq L \text{ or } |C| \leq 0 \\ \log L & , \text{ if } |C| = L \\ 0 & , \text{ otherwise} \end{array} \right. \quad (2.1)$$

where L and $|C|$ are the database size and the query-set size, respectively.

It is argued by the author that minimizing this information function corresponds to increasing the possibility of breaching the database. Given a query of length k bits issued to an SDB of 2^k entities, it is shown that the expected value of the information gained by issuing such query is:

$$k - (k-1) \left(\frac{1}{k}\right)^{1/k-1}$$

and is minimized when:

$$p = \left(\frac{1}{k}\right)^{1/k-1} \quad , \text{ for } k > 1 \quad (2.2)$$

where p is the probability of an * occurring in any given bit position.

A summarization of the security control method proposed in [42] is as follows:

Audit trails of the sequence of queries can be kept in the following ways:

- Observe the actual value of p for that sequence of queries and determine if it is statistically significantly close to the minimum value given by (2.2). If so, there is a high likelihood that the user is attempting to compromise the database.
- Evaluate the information function for each query in the sequence and study statistically the deviation of this value from the minimum expected value given by (2.2). Based on these deviations, determine the likelihood that the user is attempting to compromise the database.

Since the study is preliminary in nature, no implementation details such as the computational time and storage requirements have been addressed [4].

The authors in [44] studied the Boolean auditing problem for offline SDBs. The data elements of this study are Boolean and the queries are sum queries over the integers. Certain complexity results were proven that suggest that there is no general efficient solution for the auditing problem in this case. Two algorithms were proposed: the first is applicable when the sum queries are one-dimensional range queries (they proved that the problem is NP-hard even in the two-dimensional case). The second is an approximate algorithm that maintains security, although it may be too restrictive.

As an example, the authors in [44] considered an SDB with attributes (name, age, score) supporting statistical queries of the form “give me the sum of scores of all individuals whose age x satisfies condition $C(x)$ ”, where C is an arbitrary predicate on the domain of age, such as $35 \leq x \leq 45$. They also assumed that the projection (name, age) is publicly available, but the attribute score is private. The authors posed the following question: “What measures suffice to protect the confidentiality of the private information?”

It is assumed in most work in this area that the private data are real-valued and essentially unbounded. However, there are certain important applications where data may attain discrete values, or have maximum or minimum values that are fixed a priori and frequently attainable. In case such as these, traditional methods for maintaining security are inadequate. As an example of this, if a predicate only samples minimum values (e.g., if all individuals whose age satisfies $C(x)$ are achieved the minimum legal score), then all those individual values are definitely disclosed. Discreteness of values has even more subtle effects. Of course, the problems of discrete and bounded variables are combined by Boolean attributes. For example, consider an SDB with the attributes (name, age, hivpos), where the last attribute has values restricted to 0 or 1 so, sum queries are again allowed.

The authors [44] studied the mathematical and algorithmic problems that always arise when anyone tries to audit statistical queries on Boolean attributes. A “dual” situation was also studied, in which the data is continuous but the query discrete. A

setting in which we have a collection of (private) Boolean variables was considered and the results of some statistical queries to this set were also considered. Queries like this simply specify a subset S of the variables; the response of the returned values to these queries are the sum of the values of all variables in S .

An offline auditing framework was provided by the authors in [45][46] to determine if a database system adheres to its data disclosure policies or not. The auditor checks queries accessing confidential data by formulating an audit expression that declaratively specifies sensitive table cells.

A vision for a Hippocratic database suggests ten privacy principles for managing confidential data responsibly. Compliance is a vital principle among these. Compliance requires the database to verify that it adheres to its declared data disclosure policy.

The authors in [45][46] stated the following example of Alice who gets a blood test done at Healthco, a company whose privacy policy stipulates that it does not release patient data to external parties without the patient's consent. After some time, Alice starts receiving advertisements for an over-the-counter diabetes test. She suspects that Healthco might have released the information that she is at risk of developing diabetes. The United States Health Insurance Portability and Accountability Act (HIPAA) empowers Alice to demand from Healthco the name of every entity to whom Healthco has disclosed her information. The authors also considered the case of Bob who consented that Healthco can provide his medical data to its affiliates for the purposes of research, provided his personally identifiable information was excluded. Later on, Bob could ask Healthco to show that they indeed did exclude his name, social security number and address when they provided his medical record to the Cardio Institute. A company may institute periodic internal audits to proactively guard against potential exposures.

One approach proposed by the authors to verifying that a database adheres to its disclosure policies could be by supporting data disclosure auditing by physically logging the results of each query. There are, however, some problems with this approach include the following:

- it imposes a substantial overhead on normal query processing, particularly for queries that produce many results, and
- the actual disclosure auditing it supports is limited, since data disclosed by a query is not necessarily reflected by its output.

Consider P3P [47] as an example of the limitations on disclosure auditing. It allows individuals to specify whether a particular enterprise can have access to their data in an aggregation. Verifying that database accesses have been compliant with such user preferences is not possible when only given a log of results of statistical queries. The authors in [45][46] addressed by stating that one might consider logging the tuples “read” by a query during its execution instead of its output. However, to determine which tuples accessed during query processing were actually breached is important. Moreover, a change such as this dramatically increases logging overhead [47]-[52].

A system which audits whether the database system a query in the past that accessed the specified data was proposed by the authors in [45][46]. During normal operation, using this system, the text of every query processed by the database system is logged along with annotations such as the time when the query was run, the user posing the query and the purpose of the query. Database triggers in the system are used to capture and record all updates to base tables in backlog tables in order to recover the state of the database at any past point in time.

The authors in [45][46] stated that in order to perform an audit, audit expression is formulated by the auditor which declaratively specifies the data of interest. These audit expressions are designed to be identical to the SQL queries. This in turn allows to be performed at the level of an individual cell of a table. The audit query generator processes the audit expression. It first performs a static analysis of the expression to select a subset of logged queries that have the potential to breach the specified information. After that, the selected queries are combined and transformed into a single audit query by augmenting them with additional predicates derived from the audit expression. In this system, the audit query, expressed in standard SQL, when run against the backlog database yields the precise set of logged queries that accessed the designated data.

The authors [45][46] stated the following assumptions: the combination of the results of a series of queries may, in subtle ways, reveal certain information. For example, it is discussed in statistical database literature how individual information can be inferred by running several aggregate queries. Moreover, database security literature shows how information can be leaked by using covert channels. The authors limited themselves to the problem of determining whether the specified data was breached or not by a single query when that query is considered isolated. It is also assumed that the queries do not use outside knowledge to infer information without detection.

2.4 Summary

In this chapter, the problem of auditing SDBs was discussed. Also, it provided related work and mentioned review of literature. Related work was divided into online and offline auditing. This chapter discussed auditing methods that have been proposed and used in the literature and their different aspects and researcher's view points. Query auditing is an effective strategy to protect the privacy of individuals in SDBs that is because it provides users with precise and accurate answers (unperturbed responses). Since a detailed investigation of the inference problem revealed that we have not yet arrived at a general and acceptable solution, a new auditing scheme is proposed in this work, namely the key representation auditing scheme (KRAS), which can guarantee the security of online and dynamic SDBs, provide precise and accurate responses, and moreover it needs less CPU time and storage space during query processing. In the next chapters, the new scheme will be discussed in detail.

CHAPTER 3

RESEARCH METHODOLOGY AND CONVERSION METHOD OF THE KEY REPRESENTATION AUDITING SCHEME

3.1 Introduction

This chapter provides all the sequence steps that have been followed in order to satisfy the research objectives. Also, in this chapter a new auditing scheme is proposed, namely the key representation auditing scheme (KRAS), and the conversion method is provided to convert the original database and the user query into key representation database (KRDB) and key representation query (KRQ), respectively.

3.2 Research Methodology

This research was developed in three stages. In the first stage, a new scheme for auditing online and dynamic SDBs was developed. In the second stage, a comparative study between the new scheme and the existing methods was provided. In the final stage, the implementation of the new scheme was performed. Figure 3.1 depicts the research stages.

3.2.1 Developing a New Scheme

In the first stage of this research, a new scheme for auditing online and dynamic SDBs is developed, namely the key representation auditing scheme (KRAS). The core idea is to convert the original database, which is in both string and numerical representations and consists of t category attributes and d data attributes, into a key

representation database (KRDB), which consists of $(1+d)$ cells. The t category attributes in the original database are converted into one cell, and the d data attributes are separated by the sign ‘.’

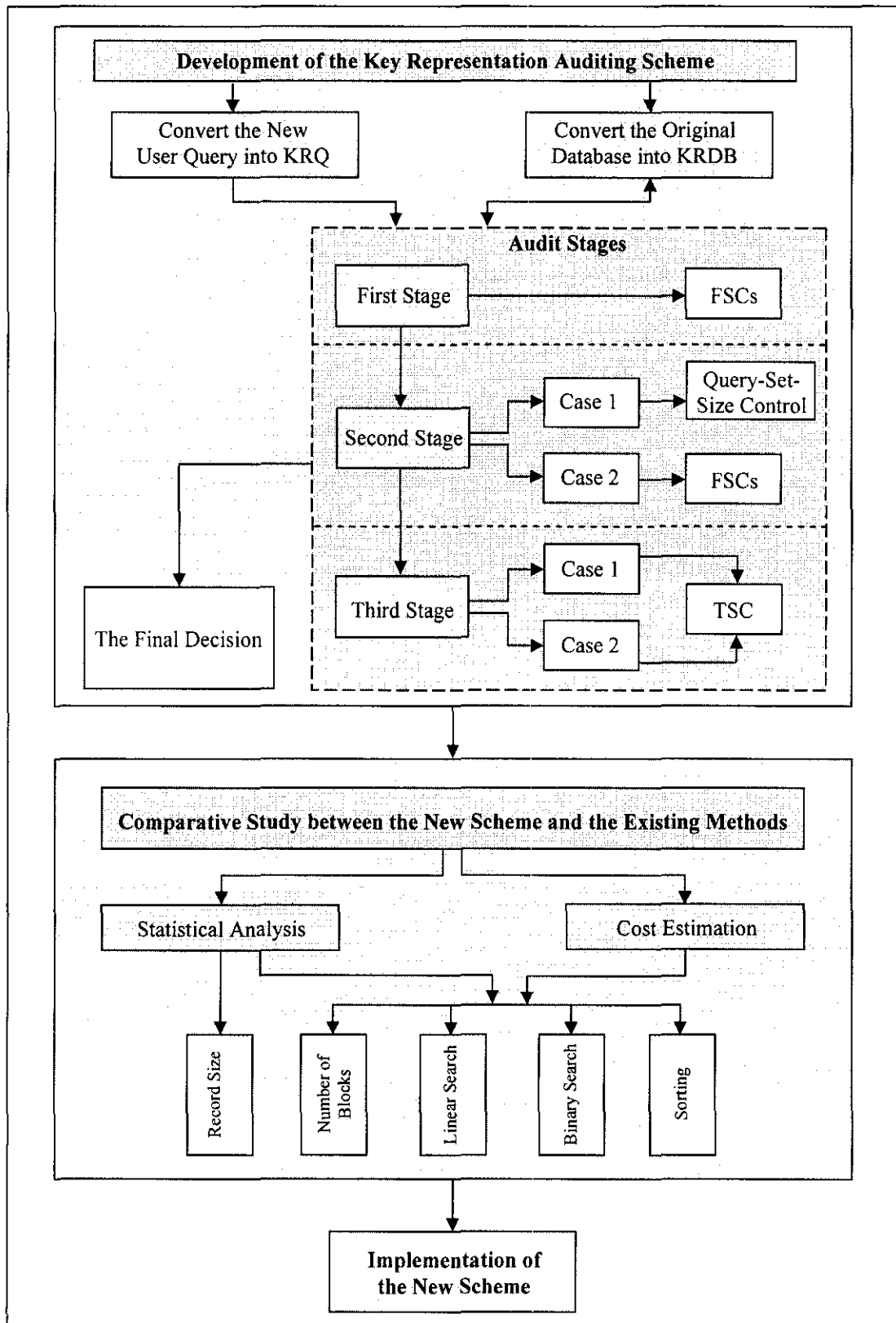


Figure 3.1: Research Stages

Also, this scheme involves converting each new user query from a string representation into a key representation query (KRQ) and storing it in the Audit Query table (AQ table). The key representation query contains, for each category attribute, either the specific category attribute class or *. The * has the intuitive meaning ‘any’, namely all category attribute classes for the corresponding column.

3.2.1.1 Audit Stages

Three audit stages are proposed to repel the attacks of snoopers to the confidentiality of the individuals. Also, efficient algorithms for these stages are presented, namely the First Stage Algorithm (FSA), the Second Stage Algorithm (SSA) and the Third Stage Algorithm (TSA). These algorithms enable the key representation auditor (KRA) to conveniently specify the illegal queries which could lead to disclosing the SDB. The three audit stages are:

(1) The First Audit Stage:

In the first audit stage, the new key representation query (KRQ) is prevented if it satisfies the first stage conditions (FSCs) with one of the previous KRQs that has already been posed. Otherwise, the second audit stage will be checked by the KRA.

(2) The Second Audit Stage:

In the second audit stage, we have two cases. In the first case, the new KRQ is prevented if one of its parts does not satisfy the query-set-size control. In the second case, the new KRQ is prevented if one of its parts satisfies the first stage conditions (FSCs) with one of the previous KRQs that has already been posed. If the new KRQ is not prevented in this stage, the third audit stage will be checked by the KRA.

(3) The Third Audit Stage:

In the third audit stage, we also have two cases. In the first case, the new KRQ is permitted if it is equal to one of the previous KRQs that has already been posed, and satisfies the third stage condition (TSC). In the second case, the new KRQ is

permitted if one of its parts is equal to one of the previous KRQs that has already been posed, and satisfies the third stage condition (TSC).

3.2.2 Comparative Study between the New Scheme and the Existing Methods

In the second stage of this research, comparisons are made between the new scheme and the existing methods, namely a cost estimation and a statistical analysis are performed, and this study illustrates the savings in block accesses (CPU time) and storage space that are attainable when a KRDB is used.

3.2.2.1 Cost Estimation

The cost estimation comparisons between the KRDB and the original database are performed in terms of number of blocks, linear search, binary search and sorting. This work applies the cost estimation comparisons between the KRDB and the original database based on three case studies of statistical databases (SDBs) and their corresponding key representation databases (KRDBs).

3.2.2.2 Statistical Analysis

The statistical analysis is performed to compare between means and variances of the original database and the KRDB populations. The size of the sample drawn from each population is 25. The comparisons between the two populations are provided in terms of record size, number of blocks, linear search, binary search and sorting to examine whether the KRDB is better than the original database or not. This statistical analysis uses t-test and F-test to evaluate the differences in means and variances, respectively, between the two populations. This study tests the null hypothesis, that there will be no significant difference between the two populations' means/variances, against the alternative hypothesis, that there will be a significant difference between the two populations' means/variances.

3.2.3 Implementation of the New Scheme

In the final stage of this research, the implementation of the new scheme is performed and all the components of the proposed system are discussed. And by applying the three audit stages, the proposed system is capable of conveniently specifying whether the user query is answerable or not.

3.3 Conversion Method of the Key Representation Auditing Scheme

In this section, a new auditing scheme is proposed, namely the key representation auditing scheme (KRAS), and the conversion method is provided to convert the original database and the user query into key representation database (KRDB) and key representation query (KRQ), respectively. In addition, three case studies of original databases and their corresponding KRDBs are provided. Moreover, some examples of user queries and their corresponding KRQs are presented.

3.3.1 Statistical Database Model

Assume that the original database D , which in both string and numerical representations, contains N records of individuals. Each record has t category attributes and d data attributes ($A_1, A_2, \dots, A_t, A_{t+1}, A_{t+2}, \dots, A_{t+d}$) [53]-[57].

Each category attribute A_j ($1 \leq j \leq t$) has $|A_j|$ possible values, namely the domain of each category attribute has $|A_j|$ classes. For example, the attribute Gender whose two possible values (or classes) are Male and Female. Let Θ_{ikj} be the domain of the category attribute A_j ($1 \leq i \leq N$; $1 \leq k \leq |A_j|$; $1 \leq j \leq t$), where the subsets i , k and j represent the record number, the category attribute class and the attribute number, respectively. Thus, the domains of the t category attributes are as follows:

$$\text{Domain}(A_1) = \{\Theta_{i11}, \Theta_{i21}, \dots, \Theta_{i|A_1|1}\}$$

$$\text{Domain}(A_2) = \{\Theta_{i12}, \Theta_{i22}, \dots, \Theta_{i|A_2|2}\}$$

⋮

$$\text{Domain}(A_j) = \{\Theta_{i1j}, \Theta_{i2j}, \dots, \Theta_{i|A_j|j}\}$$

$$\text{Domain}(A_t) = \{\Theta_{i1t}, \Theta_{i2t}, \dots, \Theta_{i|A_t|t}\}$$

While each data attribute A_j ($t+1 \leq j \leq t+d$) holds other information, usually numerical, for which some statistical queries may be desired such as Salary, Score, Income, etc. Let V_{ij} ($1 \leq i \leq N$; $1 \leq j \leq d$) be the values of the d data attributes ($A_{t+1}, A_{t+2}, \dots, A_{t+d}$). Thus, the values of the d data attributes are as follows:

$$V_{i1}, V_{i2}, \dots, V_{ij}, \dots, V_{id}$$

3.3.2 Key Representation Database (KRDB)

The core idea of this work is to convert the original database D into key representation database D^k , which consists of $(I+d)$ cells. The t category attributes in the original database D are converted into one cell, and the d data attributes are separated by the sign ‘.’. Each category attribute value Θ_{ikj} ($1 \leq i \leq N$; $1 \leq k \leq |A_j|$; $1 \leq j \leq t$) in the original database D is replaced by its category attribute class ($k = 1, 2, \dots$, or $|A_j|$). The converted $(I+d)$ cells are as follows:

$$U_{i1}U_{i2} \dots U_{it}.V_{i1}.V_{i2}. \dots .V_{id}$$

where, U_{ij} ($1 \leq i \leq N$; $1 \leq j \leq t$) represents the category attribute class corresponding to the category attribute A_j for the record number i . And V_{ij} ($1 \leq i \leq N$; $1 \leq j \leq d$) represents the d data attributes for the record number i separated by the sign ‘.’.

Figure 3.2 shows the original database conversion algorithm, which convert the original database into KRDB.

For example, let $t = 4$ and $d = 2$, also assume the following information describe the record number i in the original database D :

$$(\Theta_{i31}, \Theta_{i12}, \Theta_{i53}, \Theta_{i34}, 2000, 1500)$$

This record is converted into $(I+d)$ cells as follows:

$$3153.2000.1500$$

where:

- 3, 1, 5 and 3 are the classes of the first, second, third and fourth category attributes, respectively.

- 2000 and 1500 are the values of the first and the second data attributes, respectively.

```

Procedure Find_Category_Attribute_Class (i, j, Xij)
Begin
  For e = 1 to |Aj|
  Begin
    if (Xij ==  $\Theta_{iej}$ )
      return(e);
    endif;
  end;
end;
Procedure Convert_OriginalDB_to_KRDB (N, t, d)
Begin
  For i = 1 to N
  Begin
    For j = 1 to t
    Begin
      K = Find_Category_Attribute_Class (i, j, Xij);
      Uij = K;
    end;
    For j = t+1 to t+d
    Begin
      Vij = Xij;
    end;
  end;
End;

```

Figure 3.2: The Original Database Conversion Algorithm

3.3.3 Key Representation Query (KRQ)

Also, this scheme involves converting each new user query q from string representation into key representation query q^k and storing it in the Audit Query table (AQ table). The key representation query q^k contains, for each category attribute A_j ($1 \leq j \leq t$), either the specific category attribute class or the sign *. The * has the intuitive meaning ‘any’, namely all category attribute classes for the corresponding column. Also, the key representation query q^k contains, for each data attribute A_j ($t+1 \leq j \leq t+d$), the value of the data attribute itself, a logical formula over its value or over any value with the same data type using the relational operators ($>$, \geq , $<$, \leq , $=$), or the sign *.

The key representation query q^k will be as follows:

$$u_1 u_2 \dots u_t \cdot v_1 \cdot v_2 \cdot \dots \cdot v_d$$

where:

- $u_j = 1 | 2 | \dots | |A_j| | * \quad , (1 \leq j \leq t)$
- $v_j = V_{ij} | f(V) | * \quad , (1 \leq i \leq N; 1 \leq j \leq d), f(V)$ logical formula over any value V with the same data type.

Figure 3.3(a) and Figure 3.3(b) show the user query conversion algorithm, which convert each new user query into KRQ.

Figure 3.4 depicts the statistical database model for the key representation auditing scheme.

```

Procedure Find_Category_Attribute_Class (j, C)
Begin
  For e = 1 to |Aj|
    Begin
      if (C == Θej)
        return(e);
      endif;
    end;
  End;
Procedure Convert_UserQuery_to_KRQ (t, d, p)
Begin
  For m = 1 to p
    Begin
      GreenNS_ctr = 0, RedNS_ctr = 0;
      For j = 1 to t
        Begin
          if (Aj == NULL)
             $u_j = '*';$ 
            NS(j) = 0;
          else
             $K = \text{Find\_Category\_Attribute\_Class}(j, \text{Val}(A_j));$ 
             $u_j = K;$ 
            read GreenNS;
            if (GreenNS == 0)
              read RedNS;
              if RedNS == 0
                NS(j) = 0;
              else

```

Figure 3.3(a): The User Query Conversion Algorithm


```

        NS(j) = 2;
        RedNS_ctr++;
    endif;
else
    NS(j) = 1;
    GreenNS_ctr++;
endif;
endif;
end;
if (RedNS_ctr == 1)
    //Red Not Sign should be at least for two categories
    exit;
endif;
for j = 1 to t+d
Begin
    if (Aj == NULL)
        vj = '*';
    else
        read op;
        if (op in {'=', '>', '>=', '<', '<=', '<>'})
            read value1;
            vj = op.value1;
        elseif (op in {'[]', '()', '[]', 'O'})
            read value1, value2;
            vj = op.value1, value2;
        endif;
    endif;
end;
end;
End;

```

Figure 3.3(b): The User Query Conversion Algorithm

Statistical Database Model

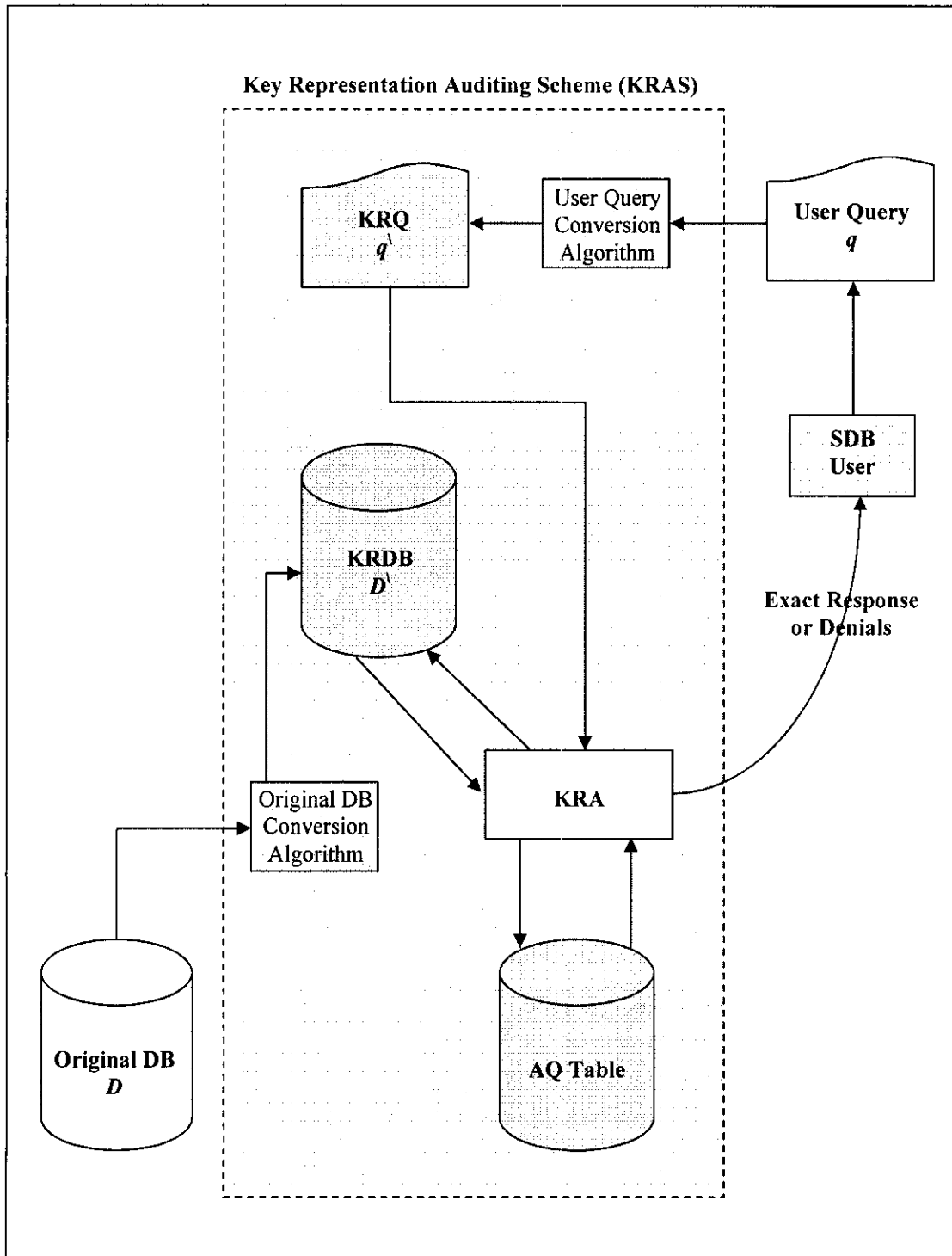


Figure 3.4: Statistical Database Model for the Key Representation Auditing Scheme

3.3.3.1 The Relational Operators in KRQ

The relational operators in the key representation query (KRQ) are signed as follows:

Table 3.1: The Relational Operators in KRQ

Expression		The sign in q^k
Greater than	$>$	\rightarrow
Greater than or equal	\geq	\rightarrow
Less than	$<$	\leftarrow
Less than or equal	\leq	\leftarrow
$v_1 \leq x \leq v_2$		$[v_1, v_2]$
$v_1 < x \leq v_2$		$(v_1, v_2]$
$v_1 \leq x < v_2$		$[v_1, v_2)$
$v_1 < x < v_2$		(v_1, v_2)

3.3.3.2 The Logical Operators in KRQ

The logical operators in the key representation query (KRQ) are signed as follows:

Table 3.2: The Logical Operators in KRQ

Logical Operator	The sign in q^k
And	\cdot
Or	$+$
Not	—

3.3.3.3 Examples of KRQs

Let $t = 4$ and $d = 2$, and consider the following user queries:

- (1) $q_1 : (A_2=\Theta_{24}) \bullet (A_4=\Theta_{41}) \bullet (A_5=2500)$
- (2) $q_2 : (A_1=\Theta_{12}) \bullet (A_2=\Theta_{21}) \bullet (A_6 > 3000)$
- (3) $q_3 : (A_2=\Theta_{21}) \bullet (A_3=\Theta_{34}) \bullet (2000 < A_5 < 4000)$
- (4) $q_4 : \text{not}((A_3=\Theta_{32}) \bullet (A_4=\Theta_{43})) \bullet (A_6 \leq 2000)$
- (5) $q_5 : (A_1=\Theta_{12}) \bullet (A_2=\Theta_{21}) + (A_1=\Theta_{13}) \bullet \text{not}(A_2=\Theta_{23})$

The key representation queries (KRQs) for the above string representation queries will be as follows:

- (1) $q_1^k : *4*1,2500.*$
- (2) $q_2^k : 21**.*.\overrightarrow{3000}$
- (3) $q_3^k : *14*.(2000,4000).*$
- (4) $q_4^k : **\overleftarrow{23}.*.2000$
- (5) $q_5^k : 21**.*.* + 33\overline{**}.*.*$

3.3.4 Audit Query Table (AQ table)

The Audit Query table (AQ table) is used in our proposed scheme for storing each new legal key representation query q^k . This table consists of the following columns:

- The key representation query: q^k
- The query result (aggregation result): $\mathbf{R}(q^k)$
- The query-set size: $|q^k|$
- The latest query-set size: $\mathbf{L}|q^k|$

Since the SDB in this scheme is online and dynamic, the individuals' records of an SDB need to be inserted, deleted and updated dynamically. Consequently, this scheme uses the latest query-set size column $\mathbf{L}|q^k|$ to deal with the previous key representation

queries q^i 's, which were legal. After updating the SDB, the snooper may repeat invoking one of the previous queries again. By using the value of this column, the KRA can decide whether or not this query could lead to the disclosure of the SDB.

3.3.5 Examples of SDB and its Corresponding KRDB

In this section, three examples, namely three case studies, of statistical databases (SDBs) and their corresponding key representation databases (KRDBs) are provided. Also, examples of user queries which have been converted into key representation queries (KRQs) are presented in this section.

3.3.5.1 The First Example

A typical example of an SDB can be illustrated based on the data held in Table 3.3. In the SDB, the salary of specific individual should not be disclosed. Table 3.3 shows the original database D summarizing confidential information about employees. Each employee is classified in three categories and has one data attribute. The possible category attributes' values are as follows:

- Gender: $\{M, F\} = \{1, 2\}$
- Dept: $\{CS, EE, PE\} = \{1, 2, 3\}$
- Level: $\{BSc, MSc, PhD\} = \{1, 2, 3\}$

The possible data attribute's values are:

- Salary (in \$): any integer ≥ 0

Table 3.4 shows the key representation database (KRDB) D^k , which is the conversion result of the original database D by converting the three category attributes (Gender, Dept and Level) into one cell ($U_{i1}U_{i2}U_{i3}$), and the data attribute value V_{i1} is separated by the sign '.'. The converted two cells are as follows:

$$U_{i1}U_{i2}U_{i3} \cdot V_{i1}$$

where, the cell $U_{i1}U_{i2}U_{i3}$ represents the category attributes' classes corresponding to the category attributes (Gender, Dept and Level) and V_{i1} represents the value of the data attribute (Salary).

Table 3.3: Example I - The Original Database D

RecNo	Name	Gender	Dept	Level	Salary
1	Adil	M	CS	MSc	200
2	Omer	M	EE	MSc	150
3	Sara	F	EE	MSc	250
4	Saria	F	CS	MSc	150
5	Samy	M	PE	MSc	180
6	Maisoon	F	PE	BSc	220
7	Gasim	M	CS	MSc	100
8	Ahmed	M	EE	MSc	180
9	Fatima	F	CS	PhD	30
10	Nasir	M	PE	BSc	200
11	Mahasin	F	EE	MSc	250
12	Khalid	M	CS	PhD	30

Table 3.4: Example I - The Key Representation Database (KRDB) D^1

RecNo	Record's key $U_{11}U_{12}U_{13} \cdot V_{11}$
1	112.200
2	122.150
3	222.250
4	212.150
5	132.180
6	231.220
7	112.100
8	122.180
9	213.30
10	131.200
11	222.250
12	113.30

Examples of user queries q_s (using the statistical query *sum*) for this database, which are converted into key representation queries (KRQs) q^1 's, are as follows:

Table 3.5: Example I - Examples of User Queries Converted into KRQs

User query q	KRQ q^1	Query set	Answer	$ q^1 $
$q_1 = M.CS$	$11^{*.*}$	{1, 7, 12}	330	3
$q_2 = F.(CS+EE).MSc$	$212.^{*} + 222.^{*}$	{3, 4, 11}	650	3
$q_3 = M+\overline{CS}$	$1^{**.*} + *1^{*.*}$	{1, 2, 3, 5, 6, 7, 8, 10, 11, 12}	1760	10
$q_4 = \text{Salary} > 200$	$***.\overrightarrow{200}$	{3, 6, 11}	720	3
$q_5 = \text{Salary} \leq 150$	$***.\overleftarrow{150}$	{2, 4, 7, 9, 12}	460	5
$q_6 = F.CS.MSc$	$212.^{*}$	{4}	150	1
$q_7 = \overline{F.CS.MSc}$	$\overline{212.^{*}}$	{1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12}	1790	11

3.3.5.2 The Second Example

A second example of an SDB can be illustrated based on the data held in Table 3.6. In the SDB, the grade-point (GP) of a specific student should not be disclosed. Table 3.6 shows the original database D summarizing confidential information about students. Each student is classified in two categories and has two data attributes. The possible category attributes' values are as follows:

- Gender: {M, F} = {1, 2}
- Dept: {CS, Math} = {1, 2}

The possible data attributes' values are:

- Age: $18 \leq \text{Age} \leq 25$
- GP: $0 \leq \text{GP} \leq 4$

Table 3.7 shows the key representation database (KRDB) D^1 , which is the conversion result of the original database D by converting the two category attributes (Gender and Dept) into one cell ($U_{i1}U_{i2}$), and the data attributes' values (V_{i1} , V_{i2}) are separated by the sign '.'. The converted three cells are as follows:

$$U_{i1}U_{i2}.V_{i1}.V_{i2}$$

where, the cell $U_{i1}U_{i2}$ represents the category attributes' classes corresponding to the category attributes (Gender and Dept). And the cells $V_{i1}.V_{i2}$ represent the values of the data attributes (Age and GP).

Table 3.6: Example II - The Original Database D

RecNo	Name	Gender	Dept	Age	GP
1	Ahmed	Male	CS	20	2
2	Sara	Female	CS	18	4
3	Omer	Male	Math	21	3
4	Gasim	Male	Math	21	2
5	Fatima	Female	Math	20	1
6	Adil	Male	Math	21	2
7	Maisoon	Female	Math	20	1
8	Nasir	Male	CS	21	2
9	Khalid	Male	CS	19	2
10	Ebrahim	Male	CS	18	2
11	Jaafar	Male	CS	19	4
12	Mahasin	Female	Math	19	4
13	Samy	Male	CS	23	4
14	Fady	Male	Math	22	4

Table 3.7: Example II - The Key Representation Database (KRDB) D^1

RecNo	Record's Key $U_{i1}U_{i2}.V_{i1}.V_{i2}$
1	11.20.2
2	21.18.4
3	12.21.3
4	12.21.2
5	22.20.1
6	12.21.2
7	22.20.1
8	11.21.2
9	11.19.2
10	11.18.2
11	11.19.4
12	22.19.4
13	11.23.4
14	12.22.4

Examples of user queries q_s (using the statistical query **sum**) for this database, which are converted into key representation queries (KRQs) q^1 's, are as follows:

Table 3.8: Example II - Examples of User Queries Converted into KRQs

User query q	KRQ q^1	Query set	Answer	$ q^1 $
$q_1 = M.CS$	$11.*.*$	{1, 8, 9, 10, 11, 13}	16	6
$q_2 = F.(CS+Math)$	$21.*.* + 22.*.*$	{2, 5, 7, 12}	10	4
$q_3 = M+\overline{CS}$	$1*.*.* + *1.*.*$	{1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}	33	13
$q_4 = \text{Age} > 20$	$**.\overrightarrow{20}.*$	{3, 4, 6, 8, 13, 14}	17	6
$q_5 = GP \leq 3$	$**.*.\overleftarrow{3}$	{1, 3, 4, 5, 6, 7, 8, 9, 10}	17	9
$q_6 = F.CS$	$21.*.*$	{2}	4	1
$q_7 = \overline{F.CS}$	$\overline{21.*.*}$	{1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}	33	13

3.3.5.3 The Third Example

The third example of an SDB can be illustrated based on the data held in Table 3.9. In the SDB, the Scholastic Aptitude Test (SAT) and the Grade-Point (GP) of a specific student should not be disclosed. Table 3.9 shows the original database D summarizing confidential information about students. Each student is classified in three categories and has two data attributes. The possible category attributes' values are as follows:

- Gender: {Male, Female} = {1, 2}
- Major: {Bio, CS, EE, Psy} = {1, 2, 3, 4}
- Class: {1978, 1979, 1980, 1981} = {1, 2, 3, 4}

The possible data attributes' values are:

- SAT: $300 \leq SAT \leq 800$

- GP: $0 \leq GP \leq 4$

Table 3.10 shows the key representation database (KRDB) D^1 , which is the conversion result of the original database D by converting the three category attributes (Gender, Major and Class) into one cell ($U_{i1}U_{i2}U_{i3}$), and the data attributes' values (V_{i1}, V_{i2}) are separated by the sign '.'. The converted three cells are as follows:

$$U_{i1}U_{i2}U_{i3}.V_{i1}.V_{i2}$$

where, the cell $U_{i1}U_{i2}U_{i3}$ represents the category attributes' classes corresponding to the category attributes (Gender, Major and Class) and the cells $V_{i1}.V_{i2}$ represent the values of the data attributes (SAT and GP).

Table 3.9: Example III - The Original Database D

RecNo	Name	Gender	Major	Class	SAT	GP
1	Allen	Female	CS	1980	600	3.4
2	Baker	Female	EE	1980	520	2.5
3	Cook	Male	EE	1978	630	3.5
4	Davis	Female	CS	1978	800	4.0
5	Evans	Male	Bio	1979	500	2.2
6	Frank	Male	EE	1981	580	3.0
7	Good	Male	CS	1978	700	3.8
8	Hall	Female	Psy	1979	580	2.8
9	Iles	Male	CS	1981	600	3.2
10	Jones	Female	Bio	1979	750	3.8
11	Kline	Female	Psy	1981	500	2.5
12	Lane	Male	EE	1978	600	3.0
13	Moore	Male	CS	1979	650	3.5

Table 3.10: Example III - The Key Representation Database (KRDB) D^1

RecNo	Record's key $U_{11}U_{12}U_{13}.V_{11}.V_{12}$
1	223.600.3.4
2	233.520.2.5
3	131.630.3.5
4	221.800.4.0
5	112.500.2.2
6	134.580.3.0
7	121.700.3.8
8	242.580.2.8
9	124.600.3.2
10	212.750.3.8
11	244.500.2.5
12	131.600.3.0
13	122.650.3.5

3.4 Summary

In this chapter all the sequences and several stages that have been followed in this research to successfully accomplish the objectives were clearly discussed. In general, this research was divided into three stages. In the first stage, a new scheme for auditing online and dynamic SDBs was developed. In the second stage, comparisons were made between the new scheme and the existing methods. In the final stage, the implementation of the new scheme was performed. Also, the new auditing scheme was discussed, namely the key representation auditing scheme (KRAS), which was proposed to protect online and dynamic SDBs from being disclosed. Also, this chapter provided the statistical database model for the proposed scheme, the key representation database (KRDB) and the key representation query (KRQ). Moreover, three case studies of original SDBs and their corresponding KRDBs were provided. In

addition, some examples of user queries and their corresponding key representation queries (KRQs) were presented.

CHAPTER 4

AUDIT STAGES OF THE KEY REPRESENTATION AUDITING SCHEME

4.1 Introduction

This work provides presentation of an auditing method that can be used to repel the attacks of the snooper to the confidentiality of the individual data in the SDB. In this chapter, three audit stages are proposed to protect online and dynamic SDBs from being disclosed. Also, efficient algorithms for these stages are presented, namely the First Stage Algorithm (FSA), the Second Stage Algorithm (SSA) and the Third Stage Algorithm (TSA). These algorithms enable the key representation auditor (KRA) to conveniently specify the illegal queries which could lead to disclosing the SDB.

4.2 Audit Stages

Before implementing the audit stages, the new user query-set size must fall in the allowable range $[n, N-n]$, for some positive integer n . That is, the new user query q must satisfy the query-set-size control [1].

Query-Set-Size Control:

A user query q is permitted only if:

$$n \leq |q| \leq N-n,$$

Where, $n \geq 0$ is a parameter of a database.

Given a sequence of key representation queries $q^1_1, q^1_2, \dots, q^1_h$ that have already been posed and stored in the Audit Query table (AQ table), and a new key representation query q^1_{h+1} :

$$q^1_i = U_{i1}U_{i2} \dots U_{it} \cdot V_{i1} \cdot V_{i2} \cdot \dots \cdot V_{id} \quad , (1 \leq i \leq h)$$

$$q^1_{h+1} = U_{h+1,1}U_{h+1,2} \dots U_{h+1,t} \cdot V_{h+1,1} \cdot V_{h+1,2} \cdot \dots \cdot V_{h+1,d}$$

To decide whether the new key representation query q^1_{h+1} should be answered or not, the following audit stages should be applied [54]-[57].

4.2.1 The First Audit Stage

Consider a sequence of key representation queries $q^1_i (1 \leq i \leq h)$ and a new key representation query q^1_{h+1} . For the sake of simplicity, we shall write $X_{ij} (1 \leq i \leq h; 1 \leq j \leq t+d)$ to denote both $U_{ij} (1 \leq i \leq h; 1 \leq j \leq t)$ and $V_{ij} (1 \leq i \leq h; 1 \leq j \leq d)$.

$$q^1_i = X_{i1}X_{i2} \dots X_{it} \cdot X_{i,t+1} \cdot X_{i,t+2} \cdot \dots \cdot X_{i,t+d} \quad , (1 \leq i \leq h)$$

$$q^1_{h+1} = X_{h+1,1}X_{h+1,2} \dots X_{h+1,t} \cdot X_{h+1,t+1} \cdot X_{h+1,t+2} \cdot \dots \cdot X_{h+1,t+d}$$

It is assumed that $a = i$ and $b = h+1$, if $|q^1_i| > |q^1_{h+1}|$. Otherwise, $a = h+1$ and $b = i$. Accordingly, the new key representation query q^1_{h+1} should be prevented if the KRA found q^1_i (for some $i \in \{1, 2, \dots, h\}$) satisfies the following conditions:

The First Stage Conditions (FSCs):

A new KRQ q^1_{h+1} is prevented if:

- (i) $|q^1_a| - |q^1_b| = 1$, and
 - (ii) Each $X_{bj} (1 \leq j \leq t+d)$, in q^1_b , corresponds, in q^1_a , to either * or X_{aj} where $X_{aj} = X_{bj}$, namely each cell X_{aj} should not correspond to a different value in q^1_b cells.
- And

- (iii) The result query q_r^1 returns one record, where q_r^1 can be computed by subtracting the cells of the query q_b^1 from its corresponding cells in q_a^1 , excluding the common cells between them.

The proposed First Stage Algorithm (FSA) is shown in Figure 4.1, and Figure 4.2 depicts the flow chart for the first stage conditions (FSCs).

Individual trackers can be prevented by using the first audit stage; examples 4.1, 4.4 and 4.5 below show how the proposed scheme could prevent this attack. Moreover, example 4.2 shows that this stage could prevent another new threat which can occur by stitching two answerable queries using two different category attributes. On the other hand, Example 4.3 shows that this stage permits the query which does not satisfy FSCs.

Example 4.1:

Based on Table 3.3 and Table 3.4, assume that the following query has already been posed and stored in the AQ table:

$$q_1 = F$$

$$\text{then, } q_1^1 = 2^{**}. * = \{3, 4, 6, 9, 11\} \quad , \quad |q_1^1| = 5$$

And the new user query is posed as follows:

$$q_2 = \overline{F.CS.MSc}$$

$$\text{then, } q_2^1 = \overline{212}. * = \{3, 6, 9, 11\} \quad , \quad |q_2^1| = 4$$

```

AQ = { $q^1_1, q^1_2, \dots, q^1_h$ }
// $q^1_i = X_{i1}X_{i2} \dots X_{it}X_{i,t+1}X_{i,t+2} \dots X_{i,t+d}$ 
Procedure First_Stage ( $q^1_{h+1}$ )
Begin
    Prevent = False;
    For each  $q^1_i$  in AQ
        Begin
            if (ABS(| $q^1_i$ | - | $q^1_{h+1}$ |) == 1)
                if (| $q^1_i$ | > | $q^1_{h+1}$ |)
                     $a = i$ ;
                     $b = h+1$ ;
                else
                     $a = h+1$ ;
                     $b = i$ ;
                endif;
                For each  $X_{ij}$  in  $q^1_i$ 
                    Begin
                        if ( $X_{bj} == X_{aj}$ )
                            Prevent = True;
                        elseif (( $X_{bj} != '*'$ ) and ( $X_{aj} == '*'$ ))
                            Prevent = True;
                        elseif (( $X_{bj} == '*'$ ) and ( $X_{aj} != '*'$ ))
                            Prevent = True;
                        else
                            Prevent = False;
                            break;
                        endif;
                    end;
                    | $q^1_r$ | = | $q^1_a$ | - | $q^1_b$ |;
                    if ((Prevent == True) and (| $q^1_r$ | == 1))
                        Inform the SDB to prevent  $q^1_{h+1}$ ;
                        return Prevent;
                    endif;
                endif;
            end;
        end;
        if (Prevent == False)
            Inform the SDB to permit  $q^1_{h+1}$ ;
            return Prevent;
        endif;
    End;

```

Figure 4.1: The First Stage Algorithm (FSA)

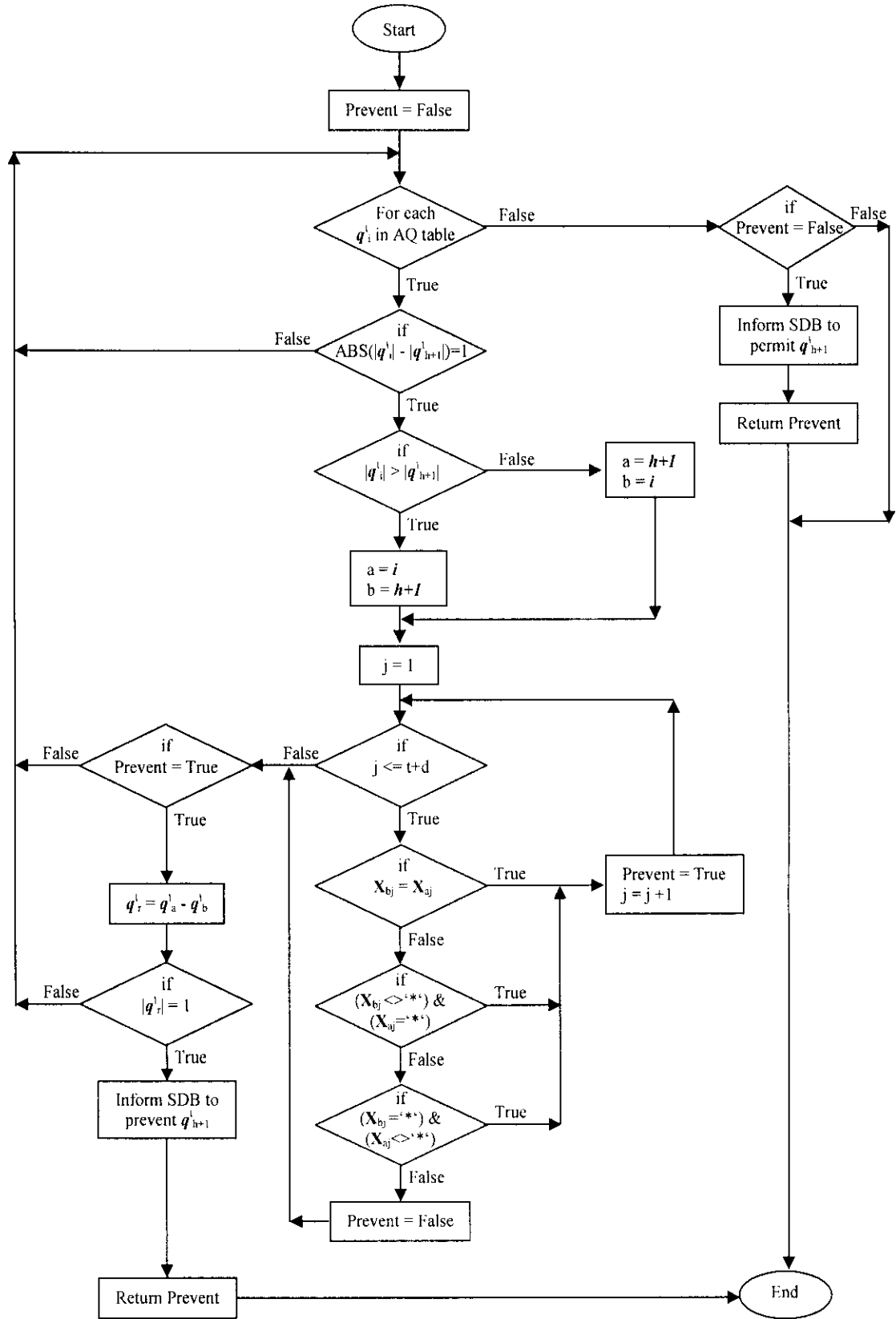


Figure 4.2: Flow Chart for the First Stage Conditions (FSCs)

By using the KRA, the new user query q_2^1 should be prevented, since (i) $|q_1^1| - |q_2^1| = 1$, and (ii) each X_{2j} ($1 \leq j \leq 4$), in q_2^1 , corresponds, in q_1^1 , to either * or X_{1j} where $X_{2j} = X_{1j}$ ($1 \leq j \leq 4$). And, (iii) by subtracting the cells of the query q_2^1 from its corresponding cells in q_1^1 , excluding the common cells between them, the result query q_r^1 would return one record (see Appendix A).

$$q_r^1 = q_1^1 - q_2^1 = 2^{**}.* - \overline{212}.* = 212.* = \{4\} \quad \blacksquare$$

Example 4.2:

Based on Table 3.3 and Table 3.4, assume that the following query has already been posed and stored in the AQ table:

$$q_1 = PE$$

$$\text{then, } q_1^1 = *3*.* = \{5, 6, 10\} \quad , \quad |q_1^1| = 3$$

And the new user query is posed as follows:

$$q_2 = BSc$$

$$\text{then, } q_2^1 = **1.* = \{6, 10\} \quad , \quad |q_2^1| = 2$$

By using the KRA, the new user query q_2^1 should be prevented, since (i) $|q_1^1| - |q_2^1| = 1$, and (ii) each X_{2j} ($1 \leq j \leq 4$), in q_2^1 , corresponds, in q_1^1 , to either * or X_{1j} where $X_{2j} = X_{1j}$ ($1 \leq j \leq 4$). And, (iii) by subtracting the cells of the query q_2^1 from its corresponding cells in q_1^1 , the result query q_r^1 would return one record (see Appendix A).

$$q_r^1 = q_1^1 - q_2^1 = *3*.* - **1.* = *3\overline{1}.* = \{5\} \quad \blacksquare$$

Example 4.3:

Based on Table 3.3 and Table 3.4, assume that the following query has already been posed and stored in the AQ table:

$$q_1 = \text{BSc}$$

$$\text{then, } q_1^1 = **1.* = \{6, 10\} \quad , \quad |q_1^1| = 2$$

And the new user query is posed as follows:

$$q_2 = \text{M.CS}$$

$$\text{then, } q_2^1 = 11*.* = \{1, 7, 12\} \quad , \quad |q_2^1| = 3$$

By using the KRA, the new user query q_2^1 should be permitted, since by subtracting the cells of the query q_1^1 from its corresponding cells in q_2^1 , the result query q_r^1 would return more than one record (see Appendix A).

$$q_r^1 = q_2^1 - q_1^1 = 11*.* - **1.* = 11\bar{1}.* = \{1, 7, 12\} \quad \blacksquare$$

Example 4.4:

Based on Table 3.6 and Table 3.7, assume that the following query has already been posed and stored in the AQ table:

$$q_1 = \text{CS}$$

$$\text{then, } q_1^1 = *1.*.* = \{1, 2, 8, 9, 10, 11, 13\} \quad , \quad |q_1^1| = 7$$

And the new user query is posed as follows:

$$q_2 = \bar{\text{F.CS}}$$

$$\text{then, } q_2^1 = \bar{2}1.*.* = \{1, 8, 9, 10, 11, 13\} \quad , \quad |q_2^1| = 6$$

By using the KRA, the new user query q_2^1 should be prevented, since (i) $|q_1^1| - |q_2^1| = 1$, and (ii) each X_{2j} ($1 \leq j \leq 4$), in q_2^1 , corresponds, in q_1^1 , to either * or X_{1j} where $X_{1j} = X_{2j}$ ($1 \leq j \leq 4$). And, (iii) by subtracting the cells of the query q_2^1 from its corresponding cells in q_1^1 , excluding the common cells between them, the result query q_r^1 would return one record (see Appendix A).

$$q_r^1 = q_1^1 - q_2^1 = *1.*.* - \bar{2}1.*.* = 21.*.* = \{2\} \quad \blacksquare$$

Example 4.5:

Based on Table 3.6 and Table 3.7, assume that the following query has already been posed and stored in the AQ table:

$$q_1 = \text{Math.Age}=21$$

$$\text{then, } q_1^1 = *2.21.* = \{3, 4, 6\} \quad , \quad |q_1^1| = 3$$

And the new user query is posed as follows:

$$q_2 = \text{Age}=21$$

$$\text{then, } q_2^1 = **.21.* = \{3, 4, 6, 8\} \quad , \quad |q_2^1| = 4$$

By using the KRA, the new user query q_2^1 should be prevented, since (i) $|q_2^1| - |q_1^1| = 1$, and (ii) each X_{1j} ($1 \leq j \leq 4$), in q_1^1 , corresponds, in q_2^1 , to either $*$ or X_{2j} where $X_{2j} = X_{1j}$ ($1 \leq j \leq 4$). And, (iii) by subtracting the cells of the query q_1^1 from its corresponding cells in q_2^1 , the result query q_r^1 would return one record (see Appendix A).

$$q_r^1 = q_2^1 - q_1^1 = **.21.* - *2.21.* = \bar{*}2.21.* = \{8\} \quad \blacksquare$$

4.2.2 The Second Audit Stage

If the new key representation query q_{h+1}^1 consists of p parts, for some positive integer p .

$$q_{h+1}^1 = q_{h+1,1}^1 + q_{h+1,2}^1 + \dots + q_{h+1,p}^1,$$

the query will need to be checked by the second audit stage.

The proposed Second Stage Algorithm (SSA) is shown in Figure 4.3(a) and Figure 4.3(b). For this stage we have the following two cases:

4.2.2.1 The Second Audit Stage: Case 1

If one of the q_{h+1}^1 parts returns one record, namely $|q_{h+1,k}^1| = 1$ (for some $k \in \{1, 2, \dots, p\}$), the new key representation query q_{h+1}^1 should be prevented. This is because if the KRA permitted this query and the snooper poses another query q_{h+2}^1 with the same parts of q_{h+1}^1 excluding the k th part, then he can deduce the individual's information by subtracting the answers of the two queries.

But if at least two of its parts return one record for each part, namely $|q_{h+1,k}^1| = 1$ and $|q_{h+1,j}^1| = 1$ (for some $k, j \in \{1, 2, \dots, p\}$ and $k \neq j$), the new key representation query q_{h+1}^1 should be permitted.

General trackers and double trackers can be prevented by using the second audit stage, examples 4.6 and 4.7 below show how the proposed scheme can prevent the general tracker.

Example 4.6:

Based on Table 3.3 and Table 3.4, the query $q = \text{F.CS.MSc}$ uniquely identifies the employee "Saria". A general tracker's query set size must fall in the range $[2n, N-2n]$ [7][8], that is $[4, 8]$ with $n=2$ and $N=12$. The formula $T = M$ qualifies as a general tracker since $|T| = 7$. The snooper applies the following equation to discover S , the total sum of all salaries.

$$\begin{aligned} S &= \text{sum}(M; \text{Salary}) + \overline{\text{sum}}(M; \text{Salary}) \\ &= 1040 + 900 = 1940 \end{aligned}$$

A tracker is obtained by defining:

- A query:

$$q_1 = \text{F.CS.MSc} + M$$

$$\text{then, } q_1^1 = 212.* + 1**.* = \{4\} + \{1, 2, 5, 7, 8, 10, 12\}$$

$$= \{1, 2, 4, 5, 7, 8, 10, 12\}$$

$$\mathbf{sum}(q_1; \text{Salary}) = 1190$$

- And, a second query:

$$q_2 = \text{F.CS.MSc} + \overline{\text{M}}$$

$$\text{then, } q_2^1 = 212.* + \overline{1**.*} = \{4\} + \{3, 4, 6, 9, 11\}$$

$$= \{3, 4, 6, 9, 11\}$$

$$\mathbf{sum}(q_2; \text{Salary}) = 900$$

The forbidden query $q = \text{F.CS.MSc}$ can be computed using the following formula:

$$\mathbf{sum}(q; \text{Salary}) = \mathbf{sum}(q_1; \text{Salary}) + \mathbf{sum}(q_2; \text{Salary}) - \mathbf{S}$$

$$= 1190 + 900 - 1940 = 150 \quad \blacksquare$$

☞ This is Saria's Salary.

By using the KRA, the query $q_1^1 = 212.* + 1**.*$ should be prevented, since one of its parts $q_{1,1}^1 = 212.*$ returns one record. Also, the query $q_2^1 = 212.* + \overline{1**.*}$ should be prevented, since one of its parts $q_{2,1}^1 = 212.*$ returns one record.

Example 4.7:

Based on Table 3.6 and Table 3.7, the query $q = \text{F.CS}$ uniquely identifies the student "Sara". A general tracker's query set size must fall in the range $[2n, N-2n]$ [7][8], that is $[4, 10]$ with $n=2$ and $N=14$. The formula $T = M$ qualifies as a general tracker since $|T| = 10$. The snooper applies the following equation to discover S , the total sum of all GPs.

$$\mathbf{S} = \mathbf{sum}(M; \text{GP}) + \mathbf{sum}(\overline{M}; \text{GP})$$

$$= 27 + 10 = 37$$

A tracker is obtained by defining:

- A query:

$$q_1 = F.CS + M$$

$$\text{then, } q_1^1 = 21.*.* + 1*.*.*$$

$$= \{2\} + \{1, 3, 4, 6, 8, 9, 10, 11, 13, 14\}$$

$$= \{1, 2, 3, 4, 6, 8, 9, 10, 11, 13, 14\}$$

$$\text{sum}(q_1; GP) = 31$$

- And, a second query:

$$q_2 = F.CS + \overline{M}$$

$$\text{then, } q_2^1 = 21.*.* + \overline{1*.*.*} = \{2\} + \{2, 5, 7, 12\}$$


$$= \{2, 5, 7, 12\}$$

$$\text{sum}(q_2; GP) = 10$$

The forbidden query $q = F.CS$ can be computed using the following formula:

$$\text{sum}(q; GP) = \text{sum}(q_1; GP) + \text{sum}(q_2; GP) - S$$

$$= 31 + 10 - 37 = 4 \quad \blacksquare$$

 This is Sara's GP.

By using the KRA, the query $q_1^1 = 21.*.* + 1*.*.*$ should be prevented, since one of its parts $q_{1,1}^1 = 21.*.*$ returns one record. Also, the query $q_2^1 = 21.*.* + \overline{1*.*.*}$ should be prevented, since one of its parts $q_{2,1}^1 = 21.*.*$ returns one record.

4.2.2.2 The Second Audit Stage: Case 2

If one of the q_{h+1}^1 parts, say $q_{h+1,k}^1$ (for some $k \in \{1, 2, \dots, p\}$), satisfies the *first stage conditions (FSCs)* with one of the previous KRQs, say q_i^1 (for some $i \in \{1, 2, \dots, h\}$), that has already been posed and stored in the AQ table. And if:

$$q_{h+1,k}^1 \cap q_{h+1,j}^1 = \Phi \quad (1 \leq j \leq p \text{ and } j \neq k)$$

then, the new key representation query q_{h+1}^1 should be prevented. That is because if the KRA permitted this query and the snooper poses another query q_{h+2}^1 with the same parts of q_{h+1}^1 excluding the k th part, then he can deduce the individual's information by subtracting the answers of these queries.

Examples 4.8 and 4.9 show that this stage could prevent another new threat which can occur by hiding an unanswerable KRQ, which satisfies the FSCs, inside the p parts of the new user query.

Example 4.8:

Based on Table 3.3 and Table 3.4, assume that the following query has already been posed and stored in the AQ table:

$$q_1 = \text{PE}$$

$$\text{then, } q_1^1 = *3*.* = \{5, 6, 10\}, \quad |q_1^1| = 3$$

$$\text{sum}(q_1; \text{Salary}) = 600$$

And, the new user query is posed as follows:

$$q_2 = \text{BSc} + \text{MSc}$$

$$\text{then, } q_2^1 = **1.* + **2.*$$

$$= \{6, 10\} + \{1, 2, 3, 4, 5, 7, 8, 11\} = \{1, 2, 3, 4, 5, 6, 7, 8, 10, 11\}, \quad |q_2^1| = 10$$

$$\text{sum}(q_2; \text{Salary}) = 1880$$

Then the snooper can pose the following query:


$$q_3 = \text{MSc}$$

$$\text{then, } q_3^1 = **2.* = \{1, 2, 3, 4, 5, 7, 8, 11\} \quad , |q_3^1| = 8$$

$$\text{sum}(q_3; \text{Salary}) = 1460$$

The snooper applies the following formula to deduce Samy's salary:

$$\begin{aligned} \text{Salary} &= \text{sum}(q_1; \text{Salary}) - (\text{sum}(q_2; \text{Salary}) - \text{sum}(q_3; \text{Salary})) \\ &= 600 - (1880 - 1460) = 180 \quad \blacksquare \end{aligned}$$

 This is Samy's Salary.

By using the KRA, the query $q_2^1 = **1.* + **2.*$ should be prevented, since one of its parts $q_{2,1} = **1.*$ satisfied the *first stage conditions (FSCs)* with the query $q_1^1 = *3*.*$ and $q_{2,1}^1 \cap q_{2,2}^1 = \Phi$.

Example 4.9:

Based on Table 3.6 and Table 3.7, assume that the following query has already been posed and stored in the AQ table:

$$q_1 = \text{CS}$$

$$\text{then, } q_1^1 = *1.*.* = \{1, 2, 8, 9, 10, 11, 13\} \quad , |q_1^1| = 7$$

$$\text{sum}(q_1; \text{GP}) = 20$$

And, the new user query is posed as follows:

$$q_2 = \text{Math} + \bar{\text{F}}.\text{CS}$$

$$\text{then, } q_2^1 = *2.*.* + \bar{2}1.*.*$$

$$= \{3, 4, 5, 6, 7, 12, 14\} + \{1, 8, 9, 10, 11, 13\}$$

$$= \{1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14\} \quad , |q_2^1|=13$$

$$\text{sum}(q_2; \text{GP}) = 33$$

Then the snooper can pose the following query:

$$q_3 = \text{Math}$$

$$\text{then, } q_3^1 = *2.*.* = \{3, 4, 5, 6, 7, 12, 14\} \quad , |q_3^1| = 7$$

$$\text{sum}(q_3; \text{GP}) = 17$$

The snooper applies the following formula to deduce Sara's GP:

$$\text{GP} = \text{sum}(q_1; \text{GP}) - (\text{sum}(q_2; \text{GP}) - \text{sum}(q_3; \text{GP}))$$

$$= 20 - (33 - 17) = 4 \quad \blacksquare$$

☞ This is Sara's GP.

By using the KRA, the query $q_2^1 = *2.*.* + \bar{2}1.*.*$ should be prevented, since one of its parts $q_{2,1}^1 = \bar{2}1.*.*$ satisfied the *first stage conditions (FSCs)* with the query $q_1^1 = *1.*.*$ and $q_{2,1}^1 \cap q_{2,2}^1 = \Phi$.

```

 $q_{h+1}^1 = \{q_{h+1,1}^1, q_{h+1,2}^1, \dots, q_{h+1,p}^1\}$ 
Procedure Second_Stage ( $q_{h+1}^1, p$ )
Begin
    // The Second Stage: Case 1
    counter = 0;
    Prevent = False;
    For each  $q_{h+1,k}^1$  in  $q_{h+1}^1$ 
    Begin
        if ( $|q_{h+1,k}^1| == 1$ )
            counter++;
        endif;
    end;

```

Figure 4.3(a): The Second Stage Algorithm (SSA)

```

if (counter == 1)
    Prevent = True;
    Inform the SDB to prevent  $q_{h+1}^1$ ;
    return Prevent;
endif;
// The Second Stage: Case 2
counter = 0;
Prevent = False;
For each  $q_{h+1,k}^1$  in  $q_{h+1}^1$ 
    Begin
        Intersection = False;
        if (First_Stage ( $q_{h+1,k}^1$ ) == True)
            For each  $q_{h+1,j}^1$  in  $q_{h+1}^1$ 
                Begin
                    if ( $j \neq k$ )
                        if ( $q_{h+1,k}^1 \cap q_{h+1,j}^1 \neq \Phi$ )
                            Intersection = True;
                            break;
                        endif;
                    endif;
                end;
            if (Intersection == False)
                counter++;
            endif;
        endif;
    end;
if (counter == 1)
    Prevent = True;
    Inform the SDB to prevent  $q_{h+1}^1$ ;
    return Prevent;
else
    Prevent = False;
    Inform the SDB to permit  $q_{h+1}^1$ ;
    return Prevent;
endif;
End;

```

Figure 4.3(b): The Second Stage Algorithm (SSA)

4.2.3 The Third Audit Stage

Since the SDB in this scheme is online and dynamic, the individuals' records of an SDB need to be inserted, deleted and updated dynamically. After updating the SDB, the snooper may repeat invoking one of the previous queries again. In this case the third audit stage can be used to decide whether the new user query is legal or not. The proposed Third Stage Algorithm (TSA) is shown in Figure 4.4(a) and Figure 4.4(b), while Figure 4.5 depicts the flow chart for the key representation auditing scheme (KRAS).

Insertion and deletion attacks can be prevented by using the third audit stage; examples 4.10 and 4.11 below show how the proposed scheme could prevent this attack. Moreover, examples 4.12 and 4.13 show that this stage could prevent another new threat which can occur by hiding a repeated unanswerable KRQ, which does not satisfy TSC, inside the p parts of the new user query.

For this stage we have the following two cases:

4.2.3.1 The Third Audit Stage: Case 1

If the new key representation query q_{h+1}^1 is equal to one of the previous key representation queries which has already been posed and stored in the AQ table, namely $q_{h+1}^1 = q_i^1$ (for some $i \in \{1, 2, \dots, h\}$), then the KRA compares between the query-set size column and the latest query-set size column, namely $|q_i^1|$ and $L|q_i^1|$, respectively.

The Third Stage Condition (TSC):

A new KRQ q_{h+1}^1 is permitted if:

$$(|q_i^1| - L|q_i^1| = 0) \text{ or } (\text{ABS}(|q_i^1| - L|q_i^1|) \geq n) \quad , \text{ (for some } i \in \{1, 2, \dots, h\})$$

where, $n \geq 0$ is a parameter of a database.

If the above condition is satisfied, then invoking the query q_{h+1}^1 again is permitted. Otherwise, the query q_{h+1}^1 is prevented.

Example 4.10:

Based on Table 3.3 and Table 3.4, let us suppose that the user posed the query:

$$q = \text{PE}$$

$$\text{then, } q^1 = *3*.* = \{5, 6, 10\} \quad , |q^1| = 3, L|q^1| = 3$$

After inserting a new record (13, Farid, M, PE, MSc, 250), the $L|q^1|$ value will be 4. If the user posed the query $q = \text{PE}$ again,

$$\text{Then, } q^1 = *3*.* = \{5, 6, 10, 13\} \quad , |q^1| = 3, L|q^1| = 4$$

By using the KRA with $n=2$, the query $q^1 = *3*.*$ should be prevented, since $\text{ABS}(|q^1| - L|q^1|) = 1$ doesn't satisfy the *third stage condition (TSC)*.

Example 4.11:

Based on Table 3.6 and Table 3.7, let us suppose that the user posed the query:

$$q = \text{Math}$$

$$\text{then, } q^1 = *2*.* = \{3, 4, 5, 6, 7, 12, 14\} \quad , |q^1|=7, L|q^1|=7$$

After inserting a new record (15, Zainab, F, Math, 22, 3), the $L|q^1|$ value will be 8. If the user posed the query $q = \text{Math}$ again,

$$\text{then, } q^1 = *2*.* = \{3, 4, 5, 6, 7, 12, 14, 15\} \quad , |q^1|=7, L|q^1|=8$$

By using the KRA with $n=2$, the query $q^1 = *2*.*$ should be prevented, since $\text{ABS}(|q^1| - L|q^1|) = 1$ doesn't satisfy the *third stage condition (TSC)*.

4.2.3.2 The Third Audit Stage: Case 2

If the new key representation query q^1_{h+1} consists of p parts, for some positive integer p :

$$q_{h+1}^1 = q_{h+1,1}^1 + q_{h+1,2}^1 + \dots + q_{h+1,p}^1,$$

and one of its parts, say $q_{h+1,k}^1$ (for some $k \in \{1, 2, \dots, p\}$), is equal to one of the previous KRQs which has already been posed and stored in the AQ table, namely $q_{h+1,k}^1 = q_i^1$ (for some $i \in \{1, 2, \dots, h\}$), and if:

$$q_{h+1,k}^1 \cap q_{h+1,j}^1 = \Phi \quad (1 \leq j \leq p \text{ and } j \neq k)$$

then the KRA checks the *third stage condition (TSC)*. If the condition is satisfied, then the new user query q_{h+1}^1 is permitted, otherwise the query q_{h+1}^1 is prevented. That is because if the KRA permitted this query and the snooper poses another query q_{h+2}^1 with the same parts of q_{h+1}^1 excluding the k th part, then he can deduce the individual's information by subtracting the answers of these queries.

Example 4.12:

Based on Table 3.3 and Table 3.4, let us suppose that the user posed the query:

$$q_1 = \text{PE}$$

$$\text{then, } q_1^1 = *3*.* = \{5, 6, 10\} \quad , |q_1^1|=3, L|q_1^1|=3$$

$$\text{sum}(q_1; \text{Salary}) = 600$$

After inserting a new record (13, Farid, M, PE, MSc, 250), the $L|q_1^1|$ value will be **4**.

If the user posed the query:

$$q_2 = \text{PhD} + \text{PE}$$

$$\text{then, } q_2^1 = **3*.* + *3*.* = \{9, 12\} + \{5, 6, 10, 13\} = \{5, 6, 9, 10, 12, 13\}$$

$$\text{sum}(q_2; \text{Salary}) = 910$$

Then the snooper can pose the following query:

$$q_3 = \text{PhD}$$

then, $q_3^1 = **3.* = \{9, 12\}$

$$\mathbf{sum}(q_3; \text{Salary}) = 60$$

The snooper applies the following formula to deduce Farid's salary:

$$\begin{aligned} \text{Salary} &= \mathbf{sum}(q_2; \text{Salary}) - \mathbf{sum}(q_3; \text{Salary}) - \mathbf{sum}(q_1; \text{Salary}) \\ &= 910 - 60 - 600 = 250 \quad \blacksquare \end{aligned}$$

 This is Farid's Salary.

By using the KRA with $n = 2$, the query $q_2^1 = **3.* + *3*.*$ should be prevented, since one of its parts $q_{2,2}^1 = *3*.*$ is equal to one of the previous KRQs and $\text{ABS}(|q_i^1| - \mathbf{L}|q_i^1|) = 1$ doesn't satisfy the *third stage condition (TSC)*.

Example 4.13:

Based on Table 3.6 and Table 3.7, let us suppose that the user posed the query:

$$q_1 = \text{Math}$$

$$\text{then, } q_1^1 = *2*.* = \{3, 4, 5, 6, 7, 12, 14\} \quad , |q_1^1| = 7, \mathbf{L}|q_1^1| = 7$$

$$\mathbf{sum}(q_1; \text{GP}) = 17$$

After inserting a new record (15, Salih, M, Math, 22, 3), the $\mathbf{L}|q_1^1|$ value will be 8.

If the user posed the query:

$$q_2 = \overline{\text{F}}.\text{CS} + \text{Math}$$

$$\begin{aligned} \text{then, } q_2^1 &= \overline{2}1*.* + *2*.* = \{1, 8, 9, 10, 11, 13\} + \{3, 4, 5, 6, 7, 12, 14, 15\} \\ &= \{1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\} \end{aligned}$$

$$\mathbf{sum}(q_2; \text{GP}) = 36$$

Then the snooper can pose the following query:

$$q_3 = \overline{F}.CS$$

$$\text{then, } q_3^1 = \overline{2}1.*.* = \{1, 8, 9, 10, 11, 13\}$$

$$\text{sum}(q_3; GP) = 16$$

The snooper applies the following formula to deduce Salih's GP:

$$GP = \text{sum}(q_2; GP) - \text{sum}(q_3; GP) - \text{sum}(q_1; GP)$$

$$= 36 - 16 - 17 = 3 \quad \blacksquare$$

☞ This is Salih's GP.

By using the KRA with $n = 2$, the query $q_2^1 = \overline{2}1.*.* + *2.*.*$ should be prevented, since one of its parts $q_{2,2}^1 = *2.*.*$ is equal to one of the previous KRQs and $\text{ABS}(|q_i^1| - \text{L}|q_i^1|) = 1$ doesn't satisfy the *third stage condition (TSC)*.

```

AQ = { $q_1^1, q_2^1, \dots, q_h^1$ }
 $q_{h+1}^1 = \{q_{h+1,1}^1, q_{h+1,2}^1, \dots, q_{h+1,p}^1\}$ 
Procedure Third_Stage_Condition ( $q_m^1$ )
Begin
    If ((( $|q_m^1| - \text{L}|q_m^1|$ ) == 0) or ( $\text{ABS}(|q_m^1| - \text{L}|q_m^1|) \geq n$ ))
        return True;
    else
        return False;
    endif;
End;
Procedure Third_Stage ( $q_{h+1}^1$ )
Begin
    // The Third Stage: Case 1
    Permit = False;
    For each  $q_i^1$  in AQ
        Begin
            If ( $q_{h+1}^1 == q_i^1$ )

```

Figure 4.4(a): The Third Stage Algorithm (TSA)

```

        If (Third_Stage_Condition ( $q^i$ ) == True)
            Permit = True;
            Inform the SDB to permit  $q^i_{h+1}$ ;
            return Permit;
        else
            Permit = False;
            Inform the SDB to prevent  $q^i_{h+1}$ ;
            return Permit;
        endif;
    endif;
end;
// The Third Stage : Case 2
counter = 0;
Permit = False;
For each  $q^i_{h+1,k}$  in  $q^i_{h+1}$ 
Begin
    Intersection = False;
    If (Third_Stage( $q^i_{h+1,k}$ ) == True)
        For each  $q^i_{h+1,j}$  in  $q^i_{h+1}$ 
            Begin
                If ( $j \neq k$ )
                    If ( $q^i_{h+1,k} \cap q^i_{h+1,j} \neq \Phi$ )
                        Intersection = True;
                        break;
                    endif;
                endif;
            end;
        endif;
    end;

    if (Intersection == False)
        counter++;
    endif;
endif;
end;
If (counter >= 1)
    Permit = True;
    Inform the SDB to permit  $q^i_{h+1}$ ;
    return Permit;
else
    Permit = False;
    Inform the SDB to prevent  $q^i_{h+1}$ ;
    return Permit;
endif;
End;

```

Figure 4.4(b): The Third Stage Algorithm (TSA)

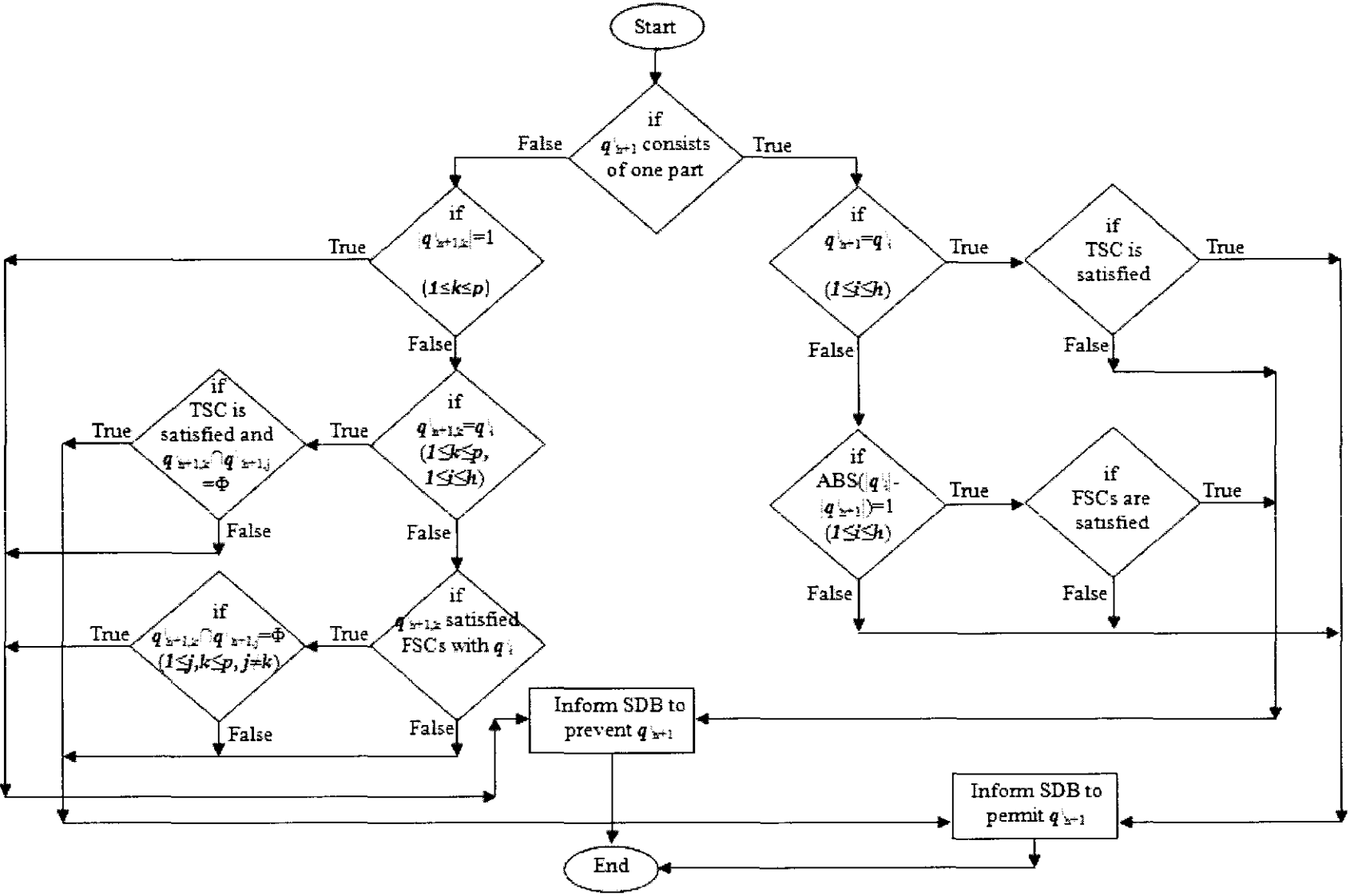


Figure 4.5: Flow Chart for the Key Representation Auditing Scheme

4.3 Summary

Three audit stages of our proposed scheme were provided in this chapter, these stages were proposed to protect online and dynamic SDBs from being disclosed. Efficient algorithms for these stages were presented. These algorithms enable the key representation auditor (KRA) to conveniently specify the illegal queries which could lead to disclosing the SDB. In addition, many examples for each stage were provided. These examples showed how the KRA could prevent the SDB threats such as individual trackers, general trackers, double trackers and insertion and deletion attacks. Also, the new scheme could prevent another three new types of threats which can occur by stitching two answerable queries using two different category attributes, hiding an unanswerable KRQ, which satisfies the FSCs, inside the p parts of the new user query and hiding a repeated unanswerable KRQ, which does not satisfy the TSC, inside the p parts of the new user query.

CHAPTER 5

COST ESTIMATION FOR THE KEY REPRESENTATION AUDITING SCHEME

5.1 Introduction

This chapter provides cost estimation for our proposed scheme (KRAS), and this study illustrates the savings in block accesses (CPU time) and storage space that are attainable when a KRDB is used. Comparisons between the KRDB and the original database are provided in terms of number of blocks, linear search, binary search and sorting. The results of the comparisons showed that there is vast improvement in terms of number of blocks, linear search and sorting. And slight improvement in terms of binary search. The proposed scheme (KRAS) depends directly on the KRDB, which saves block accesses (CPU time) and storage space compared to the original database, while all the schemes proposed by previous works depend directly on the original database.

5.2 Cost Estimation

The records of a table must be allocated to disk blocks, because a block is the unit of data transfer between disk and memory [3]. The key representation database (KRDB) D^1 needs substantially fewer blocks than does the original database D . That is because each KRDB record is typically smaller in size than an original database record since it has only two attributes; consequently, more KRDB records than original database records can fit in one block.

5.3 Parameters of the Cost Estimation

Suppose that the block size is B bytes. For the original database D of fixed-length records of size R bytes, with $B \geq R$, we can fit $bfr = \lfloor B/R \rfloor$ records per block, where “ $\lfloor \]$ ” denotes flooring (round down to nearest integer). The value bfr is called the blocking factor for the table [3][2]. R can be computed as follows:

$$R = \sum_{j=1}^{t+d} \text{Size}(A_j)$$

The number of blocks b needed for the original database of N records is $b = \lceil N/bfr \rceil$ blocks, where “ $\lceil \]$ ” denotes ceiling (round up to nearest integer).

For the KRDB D^1 of fixed-length records of size R^1 bytes, we can fit $bfr^1 = \lfloor B/R^1 \rfloor$ records per block. R^1 can be computed as follows:

$$R^1 = t + d + \sum_{j=t+1}^{t+d} \text{Size}(A_j)$$

because for each KRDB record there are only t bytes for the t category attributes (A_1, A_2, \dots, A_t), d bytes for the d data attributes' separators and d data attributes ($A_{t+1}, A_{t+2}, \dots, A_{t+d}$). The number of blocks b^1 needed for the KRDB of N records is $b^1 = \lceil N/bfr^1 \rceil$ blocks.

Table 5.1 shows the parameters of the cost estimation for the original database D and the key representation database D^1 .

Table 5.1: The Parameters of the Cost Estimation

	Original Database D	KRDB D^1
Record Size (bytes)	$R = \sum_{j=1}^{t+d} \text{Size}(A_j)$	$R^1 = t + d + \sum_{j=t+1}^{t+d} \text{Size}(A_j)$
Blocking Factor (Record/Block)	$bfr = \lfloor B/R \rfloor$	$bfr^1 = \lfloor B/R^1 \rfloor$
Number of Blocks (Blocks)	$b = \lceil N/bfr \rceil$	$b^1 = \lceil N/bfr^1 \rceil$

5.4 Comparisons between the KRDB and the Original Database

This section provides comparisons between the KRDB and the original database in terms of number of blocks, linear search, binary search and sorting. These comparisons are performed for the following two case studies. The results of the comparisons showed that there is vast improvement in terms of number of blocks, linear search and sorting. And slight improvement in terms of binary search [56][57].

In this section, the terms vast, slight and fair improvement can be used as follows:

- Vast: if Reduction $\geq 70\%$,
- Slight: if Reduction $\leq 40\%$,
- Fair: otherwise.

5.4.1 Case Study I – One Data Attribute

Based on the first example which has been stated in section 3.3.5.1, consider the following two examples which compare the original database with its corresponding KRDB. These examples illustrate the savings in block accesses (CPU time) and storage space that are attainable when a KRDB is used (see Table 5.2).

Example 5.1:

The fixed-length records of the original database in Table 3.3 have a record size of $R = 4 + 20 + 1 + 2 + 3 + 4 = 34$ bytes. Suppose that this table contains $N = 30,000$ records on a disk with a block size of $B = 1024$ bytes. The blocking factor for the table would be $bfr = \lfloor B / R \rfloor = \lfloor 1024 / 34 \rfloor = 30$ record/block.

- The number of blocks needed for this table is $b = \lceil N / bfr \rceil = \lceil 30,000 / 30 \rceil = 1000$ blocks.
- A linear search on this table would need $b = 1000$ block accesses.
- To perform a binary search on this table, in case it is an ordered table, would need approximately $\lceil \log_2 b \rceil = \lceil \log_2 1000 \rceil = 10$ block accesses.
- If a table has to be sorted, we would have to add the cost of the sort, which would need approximately $\lceil b \log_2 b \rceil = \lceil 1000 \log_2 1000 \rceil = 9966$ block accesses.

Example 5.2:

Consider the KRDB D^1 in Table 3.4, which its fixed-length records have a record size of $R^1 = 3 + 1 + 4 = 8$ bytes. Suppose that this table also contains $N = 30,000$ records on a disk with a block size of $B = 1024$ bytes. The blocking factor for the table would be $bfr^1 = \lfloor B / R^1 \rfloor = \lfloor 1024 / 8 \rfloor = 128$ record/block.

- The number of blocks needed for this table is $b^1 = \lceil N / bfr^1 \rceil = \lceil 30,000 / 128 \rceil = 235$ blocks. This is a vast improvement over the number of blocks needed for the original database D , which required 1000 blocks.
- A linear search on this table would need $b^1 = 235$ block accesses. This is a vast improvement over the 1000 block accesses needed for a linear search on the original database D (see Figure 5.1(a)).

- A binary search on this table, would need approximately $\lceil \log_2 b' \rceil = \lceil \log_2 235 \rceil = 8$ block accesses. This is a slight improvement over the 10 block accesses needed for a binary search on the original database D (see Figure 5.1(b)).
- The cost for sorting this table would need approximately $\lceil b' \log_2 b' \rceil = \lceil 235 \log_2 235 \rceil = 1851$ block accesses. This is a vast improvement over the 9966 block accesses needed for sorting the original database D (see Figure 5.1(c)).

Table 5.2: Case Study I - The KRDB D^1 VS the Original Database D

	Original Database D	KRDB D^1	Reduction (%)	Improvement
Record Size (bytes)	$R = 34$	$R^1 = 8$		
Blocking Factor (Record/Block)	$bfr = 30$	$bfr^1 = 128$		
Number of Blocks (Blocks)	$b = 1000$	$b^1 = 235$	76.5 %	Vast
Linear Search (Block Accesses)	$b = 1000$	$b^1 = 235$	76.5 %	Vast
Binary Search (Block Accesses)	$\lceil \log_2 b \rceil = 10$	$\lceil \log_2 b' \rceil = 8$	20.0 %	Slight
Sorting (Block Accesses)	$\lceil b \log_2 b \rceil = 9966$	$\lceil b' \log_2 b' \rceil = 1851$	81.4 %	Vast

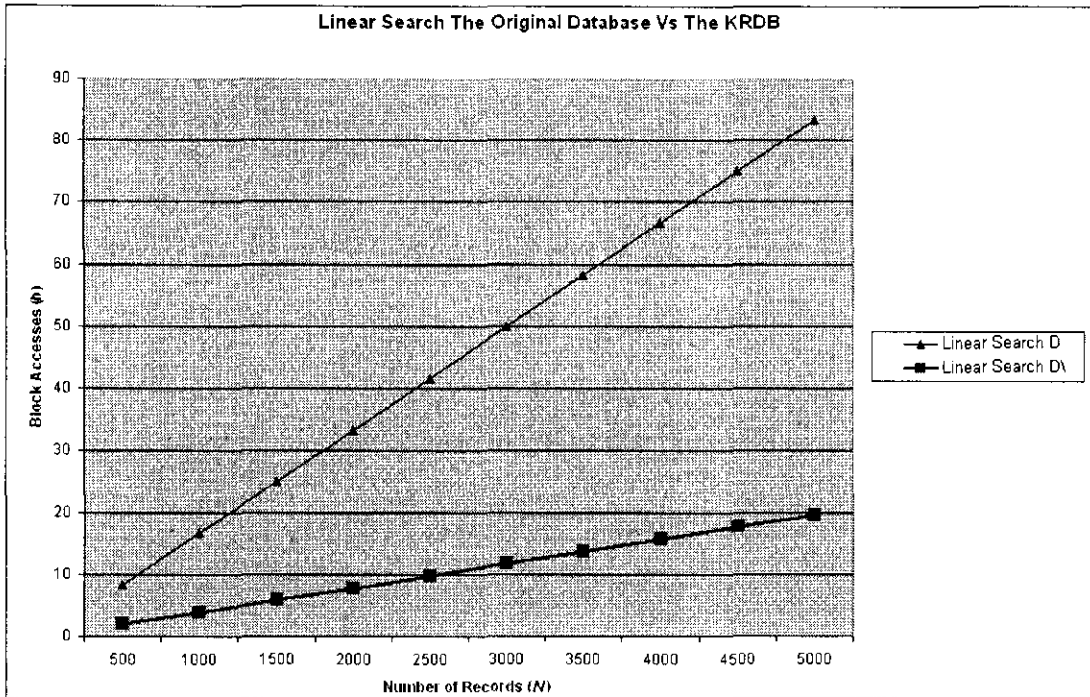


Figure 5.1(a): Case Study I - Linear Search: the Original Database Vs the KRDB

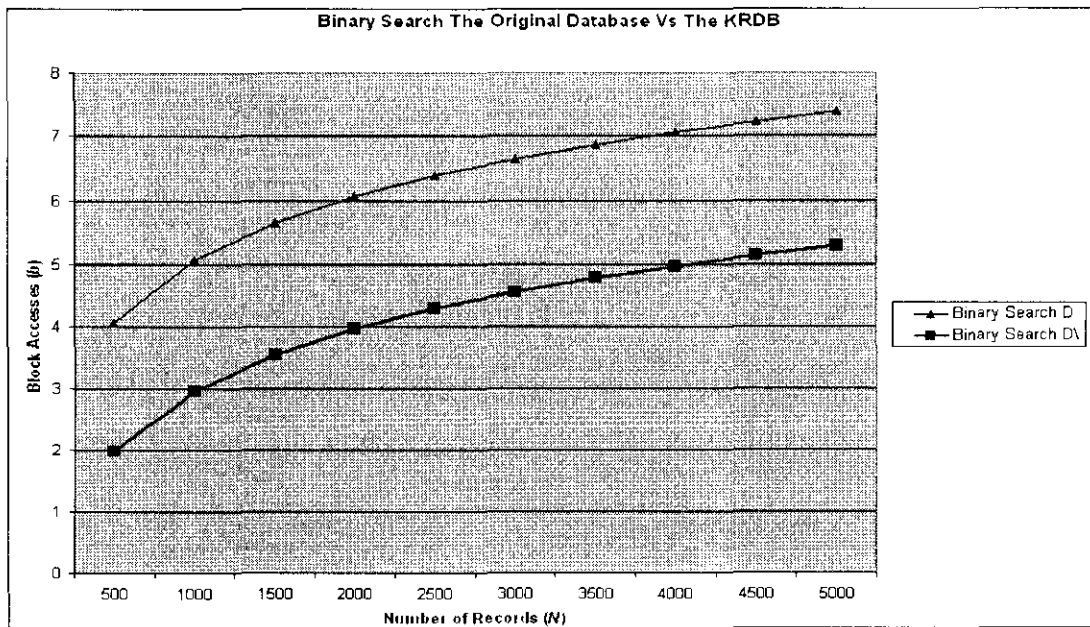


Figure 5.1(b): Case Study I - Binary Search: the Original Database Vs the KRDB

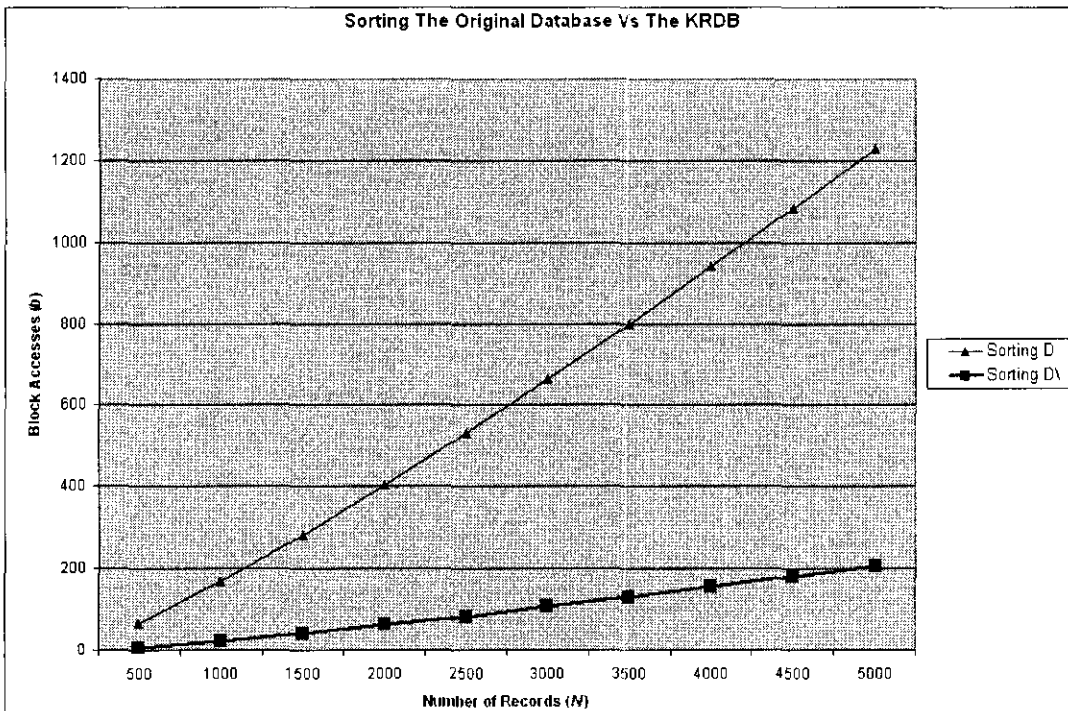


Figure 5.1(c): Case Study I - Sorting: the Original Database Vs the KRDB

5.4.2 Case Study II – More than One Data Attributes

Based on the second example which has been stated in section 3.3.5.2, consider the following two examples which compare the original database with its corresponding KRDB. These examples illustrate the savings in block accesses (CPU time) and storage space that are attainable when a KRDB is used (see Table 5.3).

Example 5.3:

The fixed-length records of the original database in Table 3.6 have a record size of $R = 4 + 20 + 6 + 4 + 4 + 4 = 42$ bytes. Suppose that this table contains $N = 30,000$ records on a disk with a block size of $B = 1024$ bytes. The blocking factor for the table would be $bfr = \lfloor B / R \rfloor = \lfloor 1024 / 42 \rfloor = 24$ record/block.

- The number of blocks needed for this table is $b = \lceil N / bfr \rceil = \lceil 30,000 / 24 \rceil = 1250$ blocks.

- A linear search on this table would need $b = 1250$ block accesses.
- To perform a binary search on this table, in case it is an ordered table, would need approximately $\lceil \log_2 b \rceil = \lceil \log_2 1250 \rceil = 11$ block accesses.
- If a table has to be sorted, we would have to add the cost of the sort, which would need approximately $\lceil b \log_2 b \rceil = \lceil 1250 \log_2 1250 \rceil = 12860$ block accesses.

Example 5.4:

Consider the KRDB D^1 in Table 3.7, which its fixed-length records have a record size of $R^1 = 2 + 2 + 4 + 4 = 12$ bytes. Suppose that this table also contains $N = 30,000$ records on a disk with a block size of $B = 1024$ bytes. The blocking factor for the table would be $bfr^1 = \lfloor B / R^1 \rfloor = \lfloor 1024 / 12 \rfloor = 85$ record/block.

- The number of blocks needed for this table is $b^1 = \lceil N / bfr^1 \rceil = \lceil 30,000 / 85 \rceil = 353$ blocks. This is a vast improvement over the number of blocks needed for the original database D , which required 1250 blocks.
- A linear search on this table would need $b^1 = 353$ block accesses. This is a vast improvement over the 1250 block accesses needed for a linear search on the original database D (see Figure 5.2(a)).
- A binary search on this table, would need approximately $\lceil \log_2 b^1 \rceil = \lceil \log_2 353 \rceil = 9$ block accesses. This is a slight improvement over the 11 block accesses needed for a binary search on the original database D (see Figure 5.2(b)).
- The cost for sorting this table would need approximately $\lceil b^1 \log_2 b^1 \rceil = \lceil 353 \log_2 353 \rceil = 2988$ block accesses. This is a vast improvement over the 12860 block accesses needed for sorting the original database D (see Figure 5.2(c))

Table 5.3: Case Study II - The KRDB D^1 VS the Original Database D

	Original Database D	KRDB D^1	Reduction (%)	Improvement
Record Size (bytes)	$R = 42$	$R^1 = 12$		
Blocking Factor (Record/Block)	$bfr = 24$	$bfr^1 = 85$		
Number of Blocks (Blocks)	$b = 1250$	$b^1 = 353$	71.8 %	Vast
Linear Search (Block Accesses)	$b = 1250$	$b^1 = 353$	71.8 %	Vast
Binary Search (Block Accesses)	$\lceil \log_2 b \rceil = 11$	$\lceil \log_2 b^1 \rceil = 9$	18.2 %	Slight
Sorting (Block Accesses)	$\lceil b \log_2 b \rceil = 12860$	$\lceil b^1 \log_2 b^1 \rceil = 2988$	76.8 %	Vast

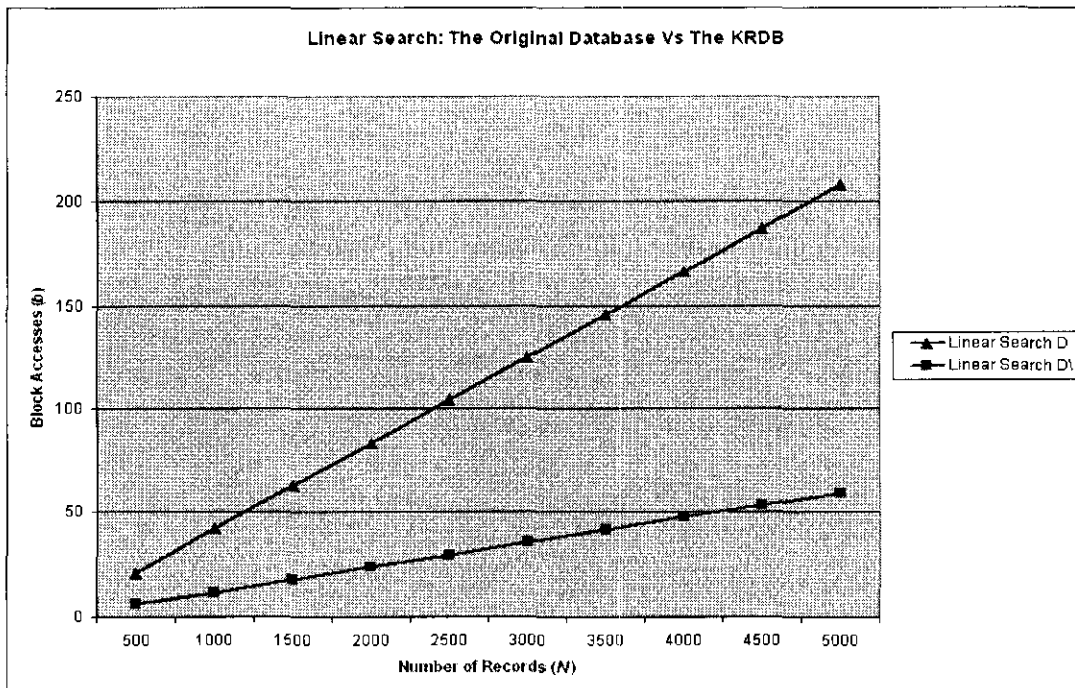


Figure 5.2(a): Case Study II - Linear Search: the Original Database Vs the KRDB

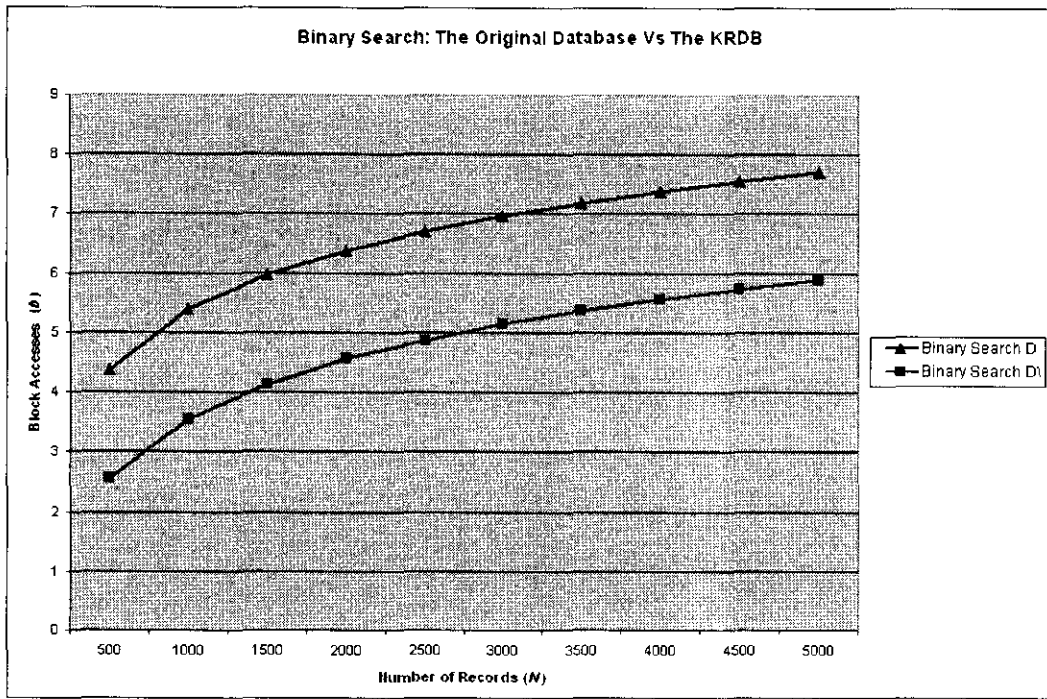


Figure 5.2(b): Case Study II - Binary Search: the Original Database Vs the KRDB

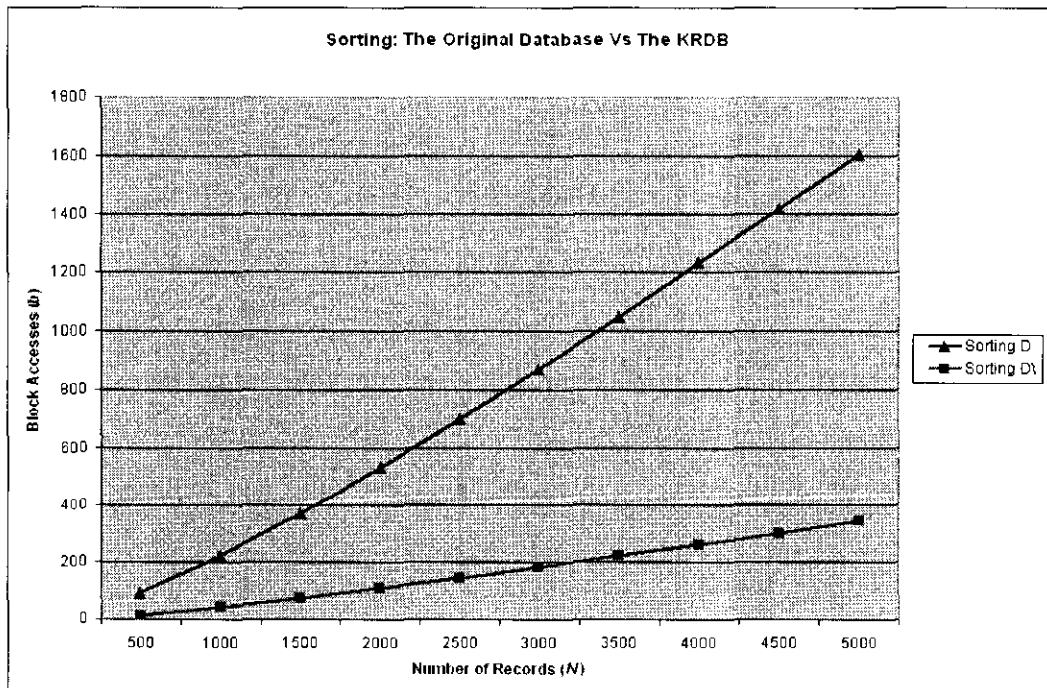


Figure 5.2(c): Case Study II - Sorting: the Original Database Vs the KRDB

5.5 Summary

In this chapter, cost estimation for our proposed scheme (KRAS) was performed, and it illustrated the savings in block accesses (CPU time) and storage space that are attainable when a KRDB is used. Comparisons were made between the KRDB and the original database in terms of number of blocks, linear search, binary search and sorting. These comparisons were performed for two case studies which have been presented in section 3.3.5. The results of the comparisons showed that there is vast improvement in terms of number of blocks, linear search and sorting. And slight improvement in terms of binary search. All the schemes proposed by previous works depend directly on the original database, while our proposed scheme (KRAS) depends directly on the KRDB, which saves block accesses (CPU time) and storage space compared to the original database.

CHAPTER 6

STATISTICAL ANALYSIS FOR THE KEY REPRESENTATION DATABASE AND THE ORIGINAL DATABASE

6.1 Introduction

This chapter provides statistical analysis and comparative study to compare between means and variances of the original database and the KRDB populations. Comparisons between the KRDB and the original database are provided in terms of record size, number of blocks, linear search, binary search and sorting to examine whether the KRDB is better than the original database or not. The statistical analysis uses t-test and F-test to evaluate the differences in means and variances, respectively, between the two populations. The size of the sample drawn from each population is 25. The null hypothesis, that there will be no significant difference between the two populations' means/variances, against the alternative hypothesis, that there will be a significant difference between the two populations' means/variances are tested. The results of the tests showed that the differences between means/variances are statistically significant, except in one case, namely the difference between the variances in terms of binary search.

6.2 Comparisons between the KRDB and the Original Database

Statistical analysis and comparative study were provided to compare between means and variances of the original database and the KRDB populations. The comparisons were made in terms of record size, number of blocks, linear search, binary search and sorting to examine whether the KRDB is better than the original database or not. The

t-test and F-test were used to evaluate the differences in means and variances, respectively, between the two populations [58]-[63].

6.2.1 Record Size: the Original Database Vs the KRDB

Table 6.1 shows record sizes of the original databases (R s) in comparison to their KRDBs (R^1 s). The size of each sample is 25. R and R^1 can be computed as follows:

$$R = \sum_{j=1}^{t+d} \text{Size}(A_j)$$

$$R^1 = t + d + \sum_{j=t+1}^{t+d} \text{Size}(A_j)$$

where:

t = the number of category attributes

d = the number of data attributes

A_j = the attribute number j in the original database

Table 6.1: Record Size: R Vs R^1

Trial Runs	Original DB Record Size R	KRDB Record Size R^1
1	34	8
2	42	12
3	57	15
4	66	13

Table 6.1 – Continued

Trial Runs	Original DB Record Size R	KRDB Record Size R^1
5	87	9
6	59	7
7	57	8
8	69	12
9	64	13
10	56	16
11	74	8
12	77	9
13	66	22
14	63	11
15	94	11
16	96	10
17	110	15
18	104	15
19	139	14
20	115	15
21	78	14
22	75	14
23	85	15

Table 6.1 – Continued

Trial Runs	Original DB Record Size R	KRDB Record Size R^1
24	110	11
25	100	15
\bar{X}_i	79.08	12.48
S_i^2	614.74	11.51

The first trial run in Table 6.1 shows:

- The record size of the original database which has been stated in Table 3.3 ($R = 4 + 20 + 1 + 2 + 3 + 4 = 34$ bytes).
- The record size of the key representation database (KRDB) which has been stated in Table 3.4 ($R^1 = 3 + 1 + 4 = 8$ bytes).

6.2.1.1 Comparing the Means in terms of Record Size

In order to compare the record size means of the two types of databases, namely the original database and the KRDB, hypothesis was tested. The null hypothesis $H_0: \mu_1 = \mu_2$, says that there will be no significant difference between the two populations' means, and the alternative hypothesis $H_a: \mu_1 \neq \mu_2$, says that there will be a significant difference between the two populations' means, with a significant level of $\alpha = 0.05$.

The pooled variance S_p^2 can be computed as follows:

$$S_p^2 = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{(n_1 + n_2 - 2)}$$

$$= \frac{24 * 614.74 + 24 * 11.51}{48} = 313.13$$

The standard error of the difference between means $S_{\bar{X}_1 - \bar{X}_2}$ can be computed as follows:

$$S_{\bar{X}_1 - \bar{X}_2} = S_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}$$

$$= \sqrt{313.13 * \left(\frac{1}{25} + \frac{1}{25}\right)} = 5.005$$

The calculated value t_c can be computed as follows:

$$t_c = \frac{\bar{X}_1 - \bar{X}_2}{S_{\bar{X}_1 - \bar{X}_2}}$$

$$= \frac{79.08 - 12.48}{5.005} = 13.31$$

The critical value t_t , obtained from a table of t-values, corresponding to the significant level of $\alpha = 0.05$ with $df = n_1 + n_2 - 2 = 48$ degrees of freedom, is:

$$t_t = t_{1 - \frac{\alpha}{2}, n_1 + n_2 - 2} = t_{0.975, 48} = 2.009$$

☞ With $\alpha = 0.05$, reject H_0 if $t_c > t_t$.

By comparing the calculated value $t_c = 13.31$ with the tabled value $t_t = 2.009$, the null hypothesis H_0 should be rejected, and it can be concluded that there is a statistically significant difference between the means of the KRDB and the original database.

6.2.1.2 Comparing the Variances in terms of Record Size

Furthermore, in order to compare record size variances between the original database and the KRDB, another hypothesis was tested. The null hypothesis $H_0: \sigma_1^2 = \sigma_2^2$, says that there will be no significant difference between the two populations' variances,

and the alternative hypothesis $H_a: \sigma_1^2 \neq \sigma_2^2$, says that there will be a significant difference between the two populations' variances, with a significant level of $\alpha = 0.05$.

The calculated value F_c can be computed as follows:

$$F_c = \frac{S_1^2}{S_2^2}$$

$$= \frac{614.74}{11.51} = 53.41$$

The critical values F_{t_1} and F_{t_2} , obtained from a table of F-values, corresponding to the significant level of $\alpha = 0.05$ with $df_1 = n_1 - 1 = 24$ and $df_2 = n_2 - 1 = 24$ degrees of freedom, are:

$$F_{t_1} = F_{(n_1-1, n_2-1)}^{1-\frac{\alpha}{2}} = F_{(24,24)}^{0.975} = 2.2693$$

$$F_{t_2} = F_{(n_1-1, n_2-1)}^{\frac{\alpha}{2}} = F_{(24,24)}^{0.025} = \frac{1}{F_{(24,24)}^{0.975}} = \frac{1}{2.2693} = 0.4407$$

☞ With $\alpha = 0.05$, reject H_0 if $F_c < F_{t_2}$ or $F_c \geq F_{t_1}$

By comparing the calculated value $F_c = 53.41$ with the tabled values $F_{t_1} = 2.2693$ and $F_{t_2} = 0.4407$, the null hypothesis H_0 should be rejected, it can be concluded that there is a statistically significant difference between the variances of the KRDB and the original database.

6.2.2 Number of Blocks: the Original Database Vs the KRDB

Besides record size, number of blocks was also compared between the original database and the KRDB. Table 6.2 shows the result of the comparison. The size of

each sample is 25. b , which is the number of blocks of the original database, and b^1 , which is number of blocks of the corresponding KRDB, can be computed as follows:

$$b = \lceil N / bfr \rceil$$

$$b^1 = \lceil N / bfr^1 \rceil$$

where:

$$bfr = \lfloor B / R \rfloor, bfr^1 = \lfloor B / R' \rfloor,$$

$$R = \sum_{j=1}^{t+d} \text{Size}(A_j), R^1 = t + d + \sum_{j=t+1}^{t+d} \text{Size}(A_j)$$

B = the block size,

N = the database size.

t = the number of category attributes

d = the number of data attributes

Table 6.2: Number of Blocks: b Vs b^1

Trial Runs	Original DB Number of Blocks b	KRDB Number of Blocks b^1
1	1000	235
2	1250	353
3	1765	442
4	2000	385

Table 6.2 – Continued

Trial Runs	Original DB Number of Blocks b	KRDB Number of Blocks b^1
5	2728	266
6	1765	206
7	1765	235
8	2143	353
9	1875	385
10	1667	469
11	2308	235
12	2308	266
13	2000	653
14	1875	323
15	3000	323
16	3000	295
17	3334	442
18	3334	442
19	4286	411
20	3750	442
21	2308	411
22	2308	411
23	2500	442

Table 6.2 – Continued

Trial Runs	Original DB Number of Blocks b	KRDB Number of Blocks b^1
24	3334	323
25	3000	442
\bar{X}_i	2,424.12	367.60
S_i^2	632,509.28	10,099.00

In the first trial run in Table 6.2, $b = 1000$ blocks and $b^1 = 235$ blocks, namely the number of blocks of the original database and the KRDB, respectively, can be computed as follows:

Suppose that the database contains $N = 30,000$ records on a disk with a block size of $B = 1024$ bytes. From the first trial run in Table 6.1, $R = 34$ and $R^1 = 8$. Then, bfr and bfr^1 can be computed as follows:

$$bfr = \lfloor B / R \rfloor = \lfloor 1024 / 34 \rfloor = 30 \text{ record/block}$$

$$bfr^1 = \lfloor B / R^1 \rfloor = \lfloor 1024 / 8 \rfloor = 128 \text{ record/block.}$$

Then, b and b^1 can be computed as follows:

$$b = \lceil N / bfr \rceil = \lceil 30,000 / 30 \rceil = 1000 \text{ blocks}$$

$$b^1 = \lceil N / bfr^1 \rceil = \lceil 30,000 / 128 \rceil = 235 \text{ blocks.}$$

6.2.2.1 Comparing the Means in terms of Number of Blocks

Again, the number of blocks means of the two databases was compared. The null hypothesis $H_0: \mu_1 = \mu_2$, says that there will be no significant difference between the

two populations' means, and the alternative hypothesis $H_a: \mu_1 \neq \mu_2$, says that there will be a significant difference between the two populations' means, with a significant level of $\alpha = 0.05$.

The pooled variance S_p^2 can be computed as follows:

$$S_p^2 = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{(n_1 + n_2 - 2)}$$

$$= \frac{24 * 632,509.28 + 24 * 10,099.00}{48} = 321,304.14$$

The standard error of the difference between means $S_{\bar{X}_1 - \bar{X}_2}$ can be computed as follows:

$$S_{\bar{X}_1 - \bar{X}_2} = S_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}$$

$$= \sqrt{321,304.14 * \left(\frac{1}{25} + \frac{1}{25}\right)} = 160.33$$

The calculated value t_c can be computed as follows:

$$t_c = \frac{\bar{X}_1 - \bar{X}_2}{S_{\bar{X}_1 - \bar{X}_2}}$$

$$= \frac{2,424.12 - 367.60}{160.33} = 12.77$$

The critical value t_t , obtained from a table of t-values, corresponding to the significant level of $\alpha = 0.05$ with $df = n_1 + n_2 - 2 = 48$ degrees of freedom, is:

$$t_t = t_{1 - \frac{\alpha}{2}, n_1 + n_2 - 2} = t_{0.975, 48} = 2.009$$

☞ With $\alpha = 0.05$, reject H_0 if $t_c > t_t$.

By comparing the calculated value $t_c = 12.77$ with the tabled value $t_c = 2.009$, the null hypothesis H_0 should be rejected, and it can be concluded that there is a statistically significant difference between the means of the KRDB and the original database.

6.2.2.2 Comparing the Variances in terms of Number of Blocks

Moreover, the null hypothesis $H_0: \sigma_1^2 = \sigma_2^2$, that there will be no significant difference between the two populations' variances, and the alternative hypothesis $H_a: \sigma_1^2 \neq \sigma_2^2$, that there will be a significant difference between the two populations' variances, were tested with a significant level of $\alpha = 0.05$.

The calculated value F_c can be computed as follows:

$$F_c = \frac{S_1^2}{S_2^2} = \frac{632,509.28}{10,099.00} = 62.63$$

The critical values F_{t_1} and F_{t_2} , obtained from a table of F-values, corresponding to the significant level of $\alpha = 0.05$ with $df_1 = n_1 - 1 = 24$ and $df_2 = n_2 - 1 = 24$ degrees of freedom, are:

$$F_{t_1} = F_{(n_1-1, n_2-1)}^{1-\frac{\alpha}{2}} = F_{(24,24)}^{0.975} = 2.2693$$

$$F_{t_2} = F_{(n_1-1, n_2-1)}^{\frac{\alpha}{2}} = F_{(24,24)}^{0.025} = \frac{1}{F_{(24,24)}^{0.975}} = \frac{1}{2.2693} = 0.4407$$

☞ With $\alpha = 0.05$, reject H_0 if $F_c < F_{t_2}$ or $F_c \geq F_{t_1}$

By comparing the calculated value $F_c = 62.63$ with the tabled values $F_{t_1} = 2.2693$ and $F_{t_2} = 0.4407$, the null hypothesis H_0 should be rejected, and it can

be concluded that there is a statistically significant difference between the variances of the KRDB and the original database.

6.2.3 Linear Search: the Original Database Vs the KRDB

As shown in Table 6.3, the results of comparisons between a linear search on the original databases and its corresponding KRDBs are presented. The size of each sample is 25. The linear search needs as follows:

- On the original database b block accesses.
- On the KRDB b^1 block accesses.

Table 6.3: Linear Search: Original DB Vs KRDB

Trial Runs	Original DB Linear Search	KRDB Linear Search
1	1000	235
2	1250	353
3	1765	442
4	2000	385
5	2728	266
6	1765	206
7	1765	235

Table 6.3 – Continued

Trial Runs	Original DB Linear Search	KRDB Linear Search
8	2143	353
9	1875	385
10	1667	469
11	2308	235
12	2308	266
13	2000	653
14	1875	323
15	3000	323
16	3000	295
17	3334	442
18	3334	442
19	4286	411
20	3750	442
21	2308	411
22	2308	411
23	2500	442

Table 6.3 – Continued

Trial Runs	Original DB Linear Search	KRDB Linear Search
24	3334	323
25	3000	442
\bar{X}_i	2,424.12	367.60
S_i^2	632,509.28	10,099.00

In the first trial run in Table 6.3:

- The linear search of the original database would need b block accesses. From the first trial run in Table 6.2: $b = 1000$.
- The linear search of the KRDB would need b^1 block accesses. From the first trial run in Table 6.2: $b^1 = 235$.

6.2.3.1 Comparing the Means in terms of Linear Search

The null hypothesis $H_0: \mu_1 = \mu_2$, that there will be no significant difference between the two populations' means, and the alternative hypothesis $H_a: \mu_1 \neq \mu_2$, that there will be a significant difference between the two populations' means, were then tested with a significant level of $\alpha = 0.05$.

The pooled variance S_p^2 can be computed as follows:

$$S_p^2 = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{(n_1 + n_2 - 2)}$$

$$= \frac{24 * 632,509.28 + 24 * 10,099.00}{48} = 321,304.14$$

The standard error of the difference between means $S_{\bar{X}_1 - \bar{X}_2}$ can be computed as follows:

$$S_{\bar{X}_1 - \bar{X}_2} = S_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}$$

$$= \sqrt{321,304.14 * \left(\frac{1}{25} + \frac{1}{25}\right)} = 160.33$$

The calculated value t_c can be computed as follows:

$$t_c = \frac{\bar{X}_1 - \bar{X}_2}{S_{\bar{X}_1 - \bar{X}_2}}$$

$$= \frac{2,424.12 - 367.60}{160.33} = 12.77$$

The critical value t_t , obtained from a table of t-values, corresponding to the significant level of $\alpha = 0.05$ with $df = n_1 + n_2 - 2 = 48$ degrees of freedom, is:

$$t_t = t_{1 - \frac{\alpha}{2}, n_1 + n_2 - 2} = t_{0.975, 48} = 2.009$$

☞ With $\alpha = 0.05$, reject H_0 if $t_c > t_t$.

By comparing the calculated value $t_c = 12.77$ with the tabled value $t_t = 2.009$, the null hypothesis H_0 should be rejected, and it can be concluded that there is a statistically significant difference between the means of the KRDB and the original database.

6.2.3.2 Comparing the Variances in terms of Linear Search

Another hypothesis was then tested for comparing the variances, in terms of linear search that needs to be performed in order to find relevant records from the original database and the KRDB. The null hypothesis $H_0: \sigma_1^2 = \sigma_2^2$, says that there will be no

significant difference between the two populations' variances, and the alternative hypothesis $H_a: \sigma_1^2 \neq \sigma_2^2$, says that there will be a significant difference between the two populations' variances, with a significant level of $\alpha = 0.05$.

The calculated value F_c can be computed as follows:

$$F_c = \frac{S_1^2}{S_2^2} = \frac{632,509.28}{10,099.00} = 62.63$$

The critical values F_{t_1} and F_{t_2} , obtained from a table of F-values, corresponding to the significant level of $\alpha = 0.05$ with $df_1 = n_1 - 1 = 24$ and $df_2 = n_2 - 1 = 24$ degrees of freedom, are:

$$F_{t_1} = F_{(n_1-1, n_2-1)}^{1-\frac{\alpha}{2}} = F_{(24,24)}^{0.975} = 2.2693$$

$$F_{t_2} = F_{(n_1-1, n_2-1)}^{\frac{\alpha}{2}} = F_{(24,24)}^{0.025} = \frac{1}{F_{(24,24)}^{0.975}} = \frac{1}{2.2693} = 0.4407$$

☞ With $\alpha = 0.05$, reject H_0 if $F_c < F_{t_2}$ or $F_c \geq F_{t_1}$

By comparing the calculated value $F_c = 62.63$ with the tabled values $F_{t_1} = 2.2693$ and $F_{t_2} = 0.4407$, the null hypothesis H_0 should be rejected, and it can be concluded that there is a statistically significant difference between the variances of the KRDB and the original database.

6.2.4 Binary Search: the Original Database Vs the KRDB

As shown in Table 6.4, the results of performing binary search on the original databases and its corresponding KRDBs are presented. The size of each sample is 25. The binary search needs as follows:

- On the original database, approximately, $\lceil \log_2 b \rceil$ block accesses.
- On the KRDB, approximately, $\lceil \log_2 b' \rceil$ block accesses.

Table 6.4: Binary Search: Original DB Vs KRDB

Trial Runs	Original DB Binary Search	KRDB Binary Search
1	10	8
2	11	9
3	11	9
4	11	9
5	12	9
6	11	8
7	11	8
8	12	8
9	11	9
10	11	9
11	12	8
12	12	9
13	11	10
14	11	9
15	12	9
16	12	9

Table 6.4 – Continued

Trial Runs	Original DB Binary Search	KRDB Binary Search
17	12	9
18	12	9
19	13	9
20	12	9
21	12	9
22	12	9
23	12	9
24	12	9
25	12	9
\bar{X}_i	11.60	8.88
S_i^2	0.42	0.19

In the first trial run in Table 6.4:

- The binary search of the original database would need $\lceil \log_2 b \rceil = \lceil \log_2 1000 \rceil = 10$ block accesses. From the first trial run in Table 6.2: $b = 1000$.
- The binary search of the KRDB would need $\lceil \log_2 b' \rceil = \lceil \log_2 235 \rceil = 8$ block accesses. From the first trial run in Table 6.2: $b' = 235$.

6.2.4.1 Comparing the Means in terms of Binary Search

The null hypothesis $H_0: \mu_1 = \mu_2$, that there will be no significant difference between the two populations' means, and the alternative hypothesis $H_a: \mu_1 \neq \mu_2$, that there will

be a significant difference between the two populations' means, were tested with a significant level of $\alpha = 0.05$.

The pooled variance S_p^2 can be computed as follows:

$$S_p^2 = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{(n_1 + n_2 - 2)}$$

$$= \frac{24 * 0.42 + 24 * 0.19}{48} = 0.31$$

The standard error of the difference between means $S_{\bar{X}_1 - \bar{X}_2}$ is computed can be follows:

$$S_{\bar{X}_1 - \bar{X}_2} = S_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}$$

$$= \sqrt{0.31 * \left(\frac{1}{25} + \frac{1}{25}\right)} = 0.157$$

The calculated value t_c can be computed as follows:

$$t_c = \frac{\bar{X}_1 - \bar{X}_2}{S_{\bar{X}_1 - \bar{X}_2}}$$

$$= \frac{11.60 - 8.88}{0.157} = 17.32$$

The critical value t_t , obtained from a table of t-values, corresponding to the significant level of $\alpha = 0.05$ with $df = n_1 + n_2 - 2 = 48$ degrees of freedom, is:

$$t_t = t_{1 - \frac{\alpha}{2}, n_1 + n_2 - 2} = t_{0.975, 48} = 2.009$$

☞ With $\alpha = 0.05$, reject H_0 if $t_c > t_t$.

By comparing the calculated value $t_c = 17.32$ with the tabled value $t_t = 2.009$, the null hypothesis H_0 should be rejected, and it can be concluded that there is a statistically significant difference between the means of the KRDB and the original database.

6.2.4.2 Comparing the Variances in terms of Binary Search

The null hypothesis $H_0: \sigma_1^2 = \sigma_2^2$, that there will be no significant difference between the two populations' variances, against the alternative hypothesis $H_a: \sigma_1^2 \neq \sigma_2^2$, that there will be a significant difference between the two populations' variances, were tested with a significant level of $\alpha = 0.05$.

The calculated value F_c can be computed as follows:

$$F_c = \frac{S_1^2}{S_2^2}$$

$$= \frac{0.42}{0.19} = 2.2105$$

The critical values F_{t_1} and F_{t_2} , obtained from a table of F-values, corresponding to the significant level of $\alpha = 0.05$ with $df_1 = n_1 - 1 = 24$ and $df_2 = n_2 - 1 = 24$ degrees of freedom, are:

$$F_{t_1} = F_{(n_1-1, n_2-1)}^{1-\frac{\alpha}{2}} = F_{(24,24)}^{0.975} = 2.2693$$

$$F_{t_2} = F_{(n_1-1, n_2-1)}^{\frac{\alpha}{2}} = F_{(24,24)}^{0.025} = \frac{1}{F_{(24,24)}^{0.975}} = \frac{1}{2.2693} = 0.4407$$

☞ With $\alpha = 0.05$, reject H_0 if $F_c < F_{t_2}$ or $F_c \geq F_{t_1}$

By comparing the calculated value $F_c = 2.2105$ with the tabled values $F_{t_1} = 2.2693$ and $F_{t_2} = 0.4407$, the null hypothesis H_0 should be accepted, and it can

be concluded that there is no a statistically significant difference between the variances of the KRDB and the original database.

6.2.5 Sorting: the Original Database Vs the KRDB

As shown in Table 6.5, the results of sorting original databases and its corresponding KRDBs are presented. The size of each sample is 25. Sorting needs as follows:

- For the original database, approximately, $\lceil b \log_2 b \rceil$ block accesses.
- For the KRDB, approximately, $\lceil b' \log_2 b' \rceil$ block accesses.

Table 6.5: Sorting: Original DB Vs KRDB

Trial Runs	Sorting Original DB	Sorting KRDB
1	9,966	1,851
2	12,860	2,988
3	19,037	3,885
4	21,932	3,307
5	31,137	2,143
6	19,037	1,584
7	19,037	1,851
8	23,714	1,851
9	20,387	3,307
10	17,842	4,162
11	25,786	1,851
12	25,786	2,143

Table 6.5 – Continued

Trial Runs	Sorting Original DB	Sorting KRDB
13	21,932	6,107
14	20,387	2,693
15	34,653	2,693
16	34,653	2,421
17	39,018	3,885
18	39,018	3,885
19	51,713	3,569
20	44,523	3,885
21	25,786	3,569
22	25,786	3,569
23	28,220	3,885
24	39,018	2,693
25	34,653	3,885
\bar{X}_i	27,435.24	3,151.96
S_i^2	102,515,708.86	1,014,098.37

In the first trial run in Table 6.5:

- Sorting the original database would need $\lceil b \log_2 b \rceil = \lceil 1000 \log_2 1000 \rceil = 9966$ block accesses. From the first trial run in Table 6.2: $b = 1000$.

- Sorting the KRDB would need $\lceil b' \log_2 b' \rceil = \lceil 235 \log_2 235 \rceil = 1851$ block accesses. From the first trial run in Table 6.2: $b^1 = 235$.

6.2.5.1 Comparing the Means in terms of Sorting

The null hypothesis $H_0: \mu_1 = \mu_2$, that there will be no significant difference between the two populations' means, and the alternative hypothesis $H_a: \mu_1 \neq \mu_2$, that there will be a significant difference between the two populations' means, were tested with a significant level of $\alpha = 0.05$.

The pooled variance S_p^2 can be computed as follows:

$$S_p^2 = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{(n_1 + n_2 - 2)}$$

$$= \frac{24 * 102,515,708.86 + 24 * 1,014,098.37}{48} = 51,764,903.615$$

The standard error of the difference between means $S_{\bar{X}_1 - \bar{X}_2}$ can be computed as follows:

$$S_{\bar{X}_1 - \bar{X}_2} = S_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}$$

$$= \sqrt{51,764,903.615 * \left(\frac{1}{25} + \frac{1}{25} \right)} = 2,034.99$$

The calculated value t_c can be computed as follows:

$$t_c = \frac{\bar{X}_1 - \bar{X}_2}{S_{\bar{X}_1 - \bar{X}_2}}$$

$$= \frac{27,435.24 - 3,151.96}{2,034.99} = 11.93$$

The critical value t_t , obtained from a table of t-values, corresponding to the significant level of $\alpha = 0.05$ with $df = n_1 + n_2 - 2 = 48$ degrees of freedom, is:

$$t_t = t_{1-\frac{\alpha}{2}, n_1+n_2-2} = t_{0.975, 48} = 2.009$$

☞ With $\alpha = 0.05$, reject H_0 if $t_c > t_t$.

By comparing the calculated value $t_c = 11.93$ with the tabled value $t_t = 2.009$, the null hypothesis H_0 should be rejected, and it can be concluded that there is a statistically significant difference between the means of the KRDB and the original database.

6.2.5.2 Comparing the Variances in terms of Sorting

The null hypothesis $H_0: \sigma_1^2 = \sigma_2^2$, that there will be no significant difference between the two populations' variances, against the alternative hypothesis $H_a: \sigma_1^2 \neq \sigma_2^2$, that there will be a significant difference between the two populations' variances, were tested with a significant level of $\alpha = 0.05$.

The calculated value F_c can be computed as follows:

$$F_c = \frac{S_1^2}{S_2^2}$$

$$= \frac{102,515,708.86}{1,014,098.37} = 101.09$$

The critical values F_{t_1} and F_{t_2} , obtained from a table of F-values, corresponding to the significant level of $\alpha = 0.05$ with $df_1 = n_1 - 1 = 24$ and $df_2 = n_2 - 1 = 24$ degrees of freedom, are:

$$F_{t_1} = F_{(n_1-1, n_2-1)}^{1-\frac{\alpha}{2}} = F_{(24,24)}^{0.975} = 2.2693$$

$$F_{t_2} = F_{(n_1-1, n_2-1)}^{\frac{\alpha}{2}} = F_{(24,24)}^{0.025} = \frac{1}{F_{(24,24)}^{0.975}} = \frac{1}{2.2693} = 0.4407$$

☞ With $\alpha = 0.05$, reject H_0 if $F_c < F_{t_2}$ or $F_c \geq F_{t_1}$

By comparing the calculated value $F_c = 101.09$ with the tabled values $F_{t_1} = 2.2693$ and $F_{t_2} = 0.4407$, the null hypothesis H_0 should be rejected, and it can be concluded that there is a statistically significant difference between the variances of the KRDB and the original database.

6.3 Summary

In this chapter, a statistical analysis was performed to compare between means and variances of the original database and the KRDB populations. Comparisons were made between the KRDB and the original database in terms of record size, number of blocks, linear search, binary search and sorting to examine whether the KRDB is better than the original database or not. This analysis used t-test and F-test to evaluate the differences in means and variances, respectively, between the two populations. The sample size for each population was 25. The null hypothesis, that there will be no significant difference between the two populations' means/variances, against the alternative hypothesis, that there will be a significant difference between the two populations' means/variances were tested. The results of the tests showed that the differences between means/variances are statistically significant, except in one case, namely the difference between the variances in terms of binary search. Table 6.6 summarizes the statistical analysis results.

Table 6.6: Statistical Analysis Results

Characteristics		Statistical Test Result	Conclusion	
1	Record Size	Means	H_0 Rejected	Significant difference
		Variances	H_0 Rejected	Significant difference
2	Number of Blocks	Means	H_0 Rejected	Significant difference
		Variances	H_0 Rejected	Significant difference
3	Linear Search	Means	H_0 Rejected	Significant difference
		Variances	H_0 Rejected	Significant difference
4	Binary Search	Means	H_0 Rejected	Significant difference
		Variances	H_0 Accepted	No significant difference
5	Sorting	Means	H_0 Rejected	Significant difference
		Variances	H_0 Rejected	Significant difference

CHAPTER 7

CONCLUSION AND FUTURE DIRECTIONS

7.1 Research Summary

This research mainly introduces a new scheme for auditing online and dynamic SDBs to guarantee the security of online and dynamic SDBs, and to repel the attacks of snoopers (malicious users) to the confidentiality of the individuals. In addition, it is introduced to conveniently specify the illegal queries which could lead to disclosing the SDB. Since some of the major drawbacks of an auditing strategy is its excessive CPU time and storage requirements to process the retrieval of relevant records from the SDB, the main purpose of introducing this scheme is to reduce CPU time and storage space during query processing.

The proposed scheme is called the key representation auditing scheme (KRAS). The core idea of this scheme is to convert the original database, which is in both string and numerical representations, into a key representation database (KRDB). Also, this scheme involves converting each new user query from string representation into a key representation query (KRQ) and storing it in the Audit Query table (AQ table). Hence, the first objective which is to develop a new scheme for auditing online and dynamic SDBs has been achieved.

Three audit stages are also proposed in order to protect the confidentiality of the individuals. The three audit stages could prevent the SDB threats such as individual trackers, general trackers, double trackers, insertion and deletion attacks and all the other types of threats which can occur by stitching the answerable queries.

Also, efficient algorithms for these stages are presented, namely the First Stage Algorithm (FSA), the Second Stage Algorithm (SSA) and the Third Stage Algorithm

(TSA). These algorithms enable the Key Representation Auditor (KRA) to conveniently specify the illegal queries which could lead to disclose the SDB. Hence, the second objective which is to guarantee the security of online and dynamic SDBs has been achieved.

Since the new scheme does not resort to estimate the value of the new query response according to the distribution of the previous answered queries, the third objective which is to provide precise and accurate responses has been achieved.

Cost estimation for the proposed scheme is performed. It is illustrated that there is savings in block accesses (CPU time) and storage space are attainable when a KRDB is used. Cost estimation comparisons between the KRDB and the original database in terms of number of blocks, linear search, binary search and sorting were also provided. The results of the comparisons show that there is a vast improvement in terms of the number of blocks, linear search and sorting. And there is a slight improvement in terms of the binary search. All the schemes proposed by previous works depend directly on the original database, while the proposed scheme (KRAS) depends directly on the KRDB, which saves block accesses (CPU time) and storage space compared to the original database.

In addition, a statistical analysis and a comparative study to compare between the means and variances of the original database and the KRDB populations is also presented. The statistical analysis tests were made in terms of record size, number of blocks, linear search, binary search and sorting to examine whether the KRDB is better than the original database or not. T-test and F-test were used to evaluate the differences in means and variances, respectively, between the two populations. The null hypothesis, that there will be no significant difference between the two populations' means/variances, against the alternative hypothesis, that there will be a significant difference between the two populations' means/variances was tested. The results of the tests show that the differences, between means/variances, are statistically significant. Hence, the fourth objective which is to reduce CPU time and storage space during query processing has been also achieved.

Finally, the implementation of the new scheme was performed, and the components of the proposed system were discussed. In addition, the graphical user interface and its logical interaction were also developed. Moreover, by applying the three audit stages, the proposed system is capable of conveniently specifying whether the user query is answerable or not.

7.2 Research Contributions

This section presents the contributions of this study which is discussed in the earlier chapters of this thesis:

- The new scheme guarantees the security of online and dynamic SDBs. The three audit stages could prevent the SDB threats such as individual trackers, general trackers, double trackers and insertion and deletion attacks. Moreover, it could prevent another new three types of threats which can occur by stitching two answerable queries using two different category attributes, hiding an unanswerable key representation query (KRQ), which satisfies the first stage conditions (FSCs) with one of the previous KRQs, inside the parts of the user query, and hiding a repeated unanswerable KRQ, which does not satisfy the third stage condition (TSC), inside the parts of the new user query.
- The new scheme provides precise and accurate responses, while most of the previous works resort to estimate the value of the new response according to the distribution of the previous answered queries.
- The new scheme, which depends directly on the key representation database (KRDB), saves CPU time and storage space compared to the original database. All schemes proposed by previous works depend directly on the original database.

7.3 Future Directions

Since the proposed scheme, namely the key representation auditing scheme (KRAS), includes only auditing count and sum statistical queries, for future work, the followings can be enhanced:

- Online and dynamic auditing of other statistical queries, including auditing of average, min, max and median queries.
- Auditing of combinations of these queries.
- Optimization of the algorithms.

7.4 Research Conclusions

In conclusion, the outcome of this research has provided an enhanced approach for efficient block accesses and space reduction audit scheme for statistical databases. It has fulfilled the research objectives which are to develop a new audit scheme for auditing online and dynamic SDBs, to guarantee the security of online and dynamic SDBs by preventing illegal queries which could lead to disclosing the SDB, to provide precise and accurate responses and to reduce CPU time and storage space during query processing.

PUBLICATIONS

1. Asim A. Elshiekh and P. D. D. Dominic, "A New Auditing Scheme for Securing Statistical Databases," IEEE, International Symposium on Information Technology 2008 (ITSIM 08), KLCC, Kuala Lumpur, Malaysia, 26-29 August, 2008, pp. 234-238.
2. Asim A. Elshiekh and P. D. D. Dominic, "The Key Representation Auditing Scheme for Securing Statistical Databases," IEEE, Proceedings of 2008 Student Conference on Research and Development (SCOReD 2008), Johor, Malaysia, 26-27 Nov, 2008, pp. 238.1-238.4.
3. Asim A. Elshiekh and P. D. D. Dominic, "Three Audit Stages for Securing Statistical Databases," 2009 International Conference on Information Management and Engineering (ICIME 2009), 2009, pp. 283-286,
4. Asim A. Elshiekh and P. D. D. Dominic, "Cost Estimation for the Key Representation Auditing Scheme," 2009 International Conference on Computer Design and Applications (ICCD2009), Nanyang Executive Centre, Singapore, 15-17 May, 2009.
5. Asim A. Elshiekh and P. D. D. Dominic, "Efficient Algorithms for the Key Representation Auditing Scheme," International Journal of Computer and Electrical Engineering (IJCEE), Vol. 1, No. 3, August 2009, pp. 340-349.
6. Asim A. Elshiekh and P. D. D. Dominic, "Statistical Analysis for the Key Representation Database and the Original Database," IEEE, International Symposium on Information Technology 2010 (ITSIM 2010), KLCC, Kuala Lumpur, Malaysia, 15-17 June, 2010.
7. Asim A. Elshiekh and P. D. D. Dominic, "Performance Comparison between the Key Representation Database and the Original Database," International Journal of

Multimedia and Security (IJMIS), INDERSCIENCE Publishers, (Accepted July 2010).

8. Asim A. Elshiekh and P. D. D. Dominic, “Statistical Analysis for Comparison of the Key Representation Database with the Original Database,” International Journal of Business Information Systems (IJBIS), INDERSCIENCE Publishers, (Accepted July 2010).

REFERENCES

- [1] S. Castano, M. Fugini, G. Martella and P. Samarati, Database Security, 1st ed, Addison-Wesley, 1995, pp. 291-341.
- [2] T. Connolly and C. Begg, Database Systems: A practical Approach to Design, Implementation and Management, 3rd ed, Addison-Wesley, 2002, pp. 604-649.
- [3] R. Elmasri and S. Navathe, Fundamentals of Database Management Systems, 5th ed, Addison-Wesley, 2007, pp. 463-591.
- [4] J. Wortmann and N. Adam, "Security-Control Methods for Statistics Databases: A Comparative Study," ACM Computing Surveys, Vol. 21(4), Dec. 1989, pp. 515-554.
- [5] S. Shieh and C. Lin, "Information Protection in Dynamic Statistical Databases," Invited paper for Encyclopedia of Computer Science and Technology, 1999.
- [6] S. Shieh and C. Lin, "Auditing User Queries in Dynamic Statistical Databases," Information Science, Vol. 113(1-2), January 1999, pp. 131-146.
- [7] D. Denning, Cryptography and Data Security, Addison-Wesley Publishing Company, Inc., 1982, pp. 313-387.
- [8] R. Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems, 2nd ed, Wiley, April 2008, pp. 172-179.
- [9] E. Unger, S. McNulty and P. Conell, "Natural Change in Dynamic Databases as a Deterrent to Compromise by Trackers," Digital Object Identifier 10.1109/CSAC.1990.143760, 1990, pp. 116-124.
- [10] F. Malvestuto, M. Mezzini and M. Moscarini, "Auditing Sum-Queries to Make a Statistical Database Secure," ACM Transactions on Information and System

Security (TISSEC), Volume 9, Issue 1 (February 2006), ISSN:1094-9224, 2006, pp. 31-60.

- [11] J. Domingo-Ferrer and J. Maria, "Current Directions in Statistical Data Protection", Proceedings of the Statistical Data Protection (SDP98), 1998, pp. 105-112.
- [12] F. Malvestuto and M. Moscarini, "Computational Issues Connected with the Protection of Sensitive Statistics by Auditing Sum-Queries," In Proc. Of IEEE Scientific and Statistical Database Management, 1998, pp. 134-144.
- [13] R. Huebner, "Automated Mechanisms for Controlling Inference in Database Systems," Project Proposal, January, 2004.
- [14] R. Ahlswede and H. Aydinian, "On Security of Statistical Databases," Digital Object Identifier: 10.1109/ISIT.2006.261767, 2006, pp. 506-508.
- [15] F. Chin and G. Ozsoyoglu, "Auditing and Inference Control in Statistical Databases," IEEE Trans. on Softw. Eng., (Apr. 1982), pp. 574-582.
- [16] G. Duncan, S. Fienberg, R. Krishnan, R. Padman and S. Roehrig, "Disclosure Limitation Methods and Information Loss for Tabular Data," In Confidentiality, Disclosure and Data Access, P. Doyle, J. Lane, J. Theeuwes, L. Zayatz, Eds. Elsevier, New York, 2001, pp. 135–166.
- [17] F. Malvestuto and M. Mezzini, "On the Hardness of Protecting Sensitive Information in a Statistical Database," In Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, Vol. XIV, 2001, pp. 504–509.
- [18] F. Malvestuto and M. Mezzini, "A Linear Algorithm for Finding the Invariant Edges of an Edge-weighted Graph," SIAM J. Computing 31, 2002, pp. 1438–1455.

- [19] F. Malvestuto and M. Mezzini, "Auditing Sum-Queries," In Proceedings of the International Conference on Database Theory, Lecture Notes in Computer Sciences, 2003, pp. 504–509.
- [20] F. Malvestuto and M. Mezzini, "Privacy Preserving and Data Mining in an On-line Statistical Database of Additive Type," In Proceedings of the International Conference on Privacy in Statistical Databases, Barcelona, 2004.
- [21] F. Malvestuto and M. Moscarini, "An Audit Expert for Large Statistical Databases," In Statistical Data Protection, EUROSTAT, 1999, pp. 29–43.
- [22] F. Malvestuto and M. Moscarini, "Privacy in Multidimensional Databases," In Multidimensional Databases, M. Rafanelli, Ed., Idea Group Pub., Hershey, PA, 2003, pp. 310–360.
- [23] L. Wang, D. Wijekera and S. Jajodia, "Cardinality-based Inference Control in Datacubes," *J. Comp. Security* 12, 2004, pp. 655–692.
- [24] L. Willenborg and T. Dewaal, "Elements of Statistical Disclosure. Lecture Notes in Statistics," 155. Springer-Verlag, New York, 2000.
- [25] N. Zhang, W. ZHAO and J. CHEN, "Cardinality-based Inference Control in OLAP Systems: An Information Theoretic Approach," In Proceedings of the ACM International Workshop on Data Warehousing and OLAP (DOLAP 2004), 2004, pp. 59–64.
- [26] K. Kenthapadi, N. Mishra and K. Nissim, "Simulatable Auditing," In Proc. of ACM Principles of Database Systems (PODS), 2005, pp. 118-127.
- [27] I. Dinur and K. Nissim, "Revealing Information while Preserving Privacy," In Symposium on Principles of Database Systems (PODS), 2003, pp. 202-210.
- [28] C. Dwork and K. Nissim, "Privacy-preserving Data Mining on Vertically Partitioned Databases," In Proceedings of the 24th Annual International

Cryptology Conference (CRYPTO 2004), Santa Barbara, CA, August 2004, pp. 528-544.

- [29] A. Evfimievski, J. Gehrke and R. Srikant, "Limiting Privacy Breaches in Privacy Preserving Data Mining," In Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, ACM Press, 2003, pp. 211-222.
- [30] Y. Li, L. Wang, X. Wang and S. Jajodia, "Auditing Interval-based Inference," In Proceedings of the 14th International Conference on Advanced Information Systems Engineering, 2002, pp. 553-567.
- [31] S. Nabar, B. Marthi, K. Kenthapadi, N. Mishra and R. Motwani, "Towards Robustness in Query Auditing," Proceedings of the 32nd International Conference on Very Large Databases (VLDB), 2006, pp. 151-162.
- [32] R. Agrawal and R. Srikant, "Privacy-preserving Data Mining," In Proc. of ACM SIGMOD, 2000, pp. 439-450.
- [33] R. Agrawal, R. Srikant and D. Thomas, "Privacy Preserving OLAP," In Proc. of ACM SIGMOD, 2005, pp. 251-262.
- [34] J. Biskup and P. Bonatti, "Controlled Query Evaluation for Known Policies by Combining Lying and Refusal," Annals of Mathematics and Artificial Intelligence, 2004, 40(1-2):37-62.
- [35] A. Blum, C. Dwork, F. McSherry and K. Nissim, "Practical Privacy: the SuLQ Framework," In Symposium on Principles of Database Systems (PODS), 2005, pp. 128-138.
- [36] S. Chawla, C. Dwork, F. McSherry, A. Smith and H. Wee, "Toward Privacy in Public Databases," In the Theory of Cryptography Conference (TCC), 2005, pp. 363-385.

- [37] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov and M. Naor, "Our Data, Ourselves: Privacy via Distributed Noise Generation," *Advances in Cryptography (EUROCRYPT 2006)*, 2006, pp.486-503.
- [38] C. Dwork, F. McSherry, K. Nissim and A. Smith, "Calibrating Noise to Sensitivity in Private Data Analysis," In the third Theory of Cryptography Conference (TCC), 2006, pp. 265-284.
- [39] O. Goldreich, *Foundations of Cryptography, Volumes I and II*, Cambridge University Press, 2004.
- [40] <http://www.stat.cmu.edu/~hwainer/bertinoro.htm>, CS-Statistics Workshop On Privacy and Confidentiality, 2005.
- [41] N. Mishra and M. Sandler, "Privacy via Pseudorandom Sketches," *Proc. 25th ACM Symp., Principles of Database Systems (PODS)*, 2006, pp. 143-152.
- [42] M. McLeish, "An Information Theoretic Approach to Statistical Databases and Their Security: A Preliminary Report," In *Proceedings of the 2nd International Workshop on Statistical Database Management*, 1983, pp. 355-359.
- [43] J. Kam and J. Ullman, "A Model of Statistical Databases and Their Security," *ACM Trans. Database Syst.* 2, 1, 1977, pp. 1-10.
- [44] J. Kleinberg, C. Papadimitriou and P. Raghavan, "Auditing Boolean Attributes," *Journal of Computer and System Sciences*, 2003, pp. 6:244-253.
- [45] R. Agrawal, R. Bayardo, C. Faloutsos, J. Kiernan, R. Rantzau and R. Srikant, "Auditing Compliance with a Hippocratic Database," In *Proceedings of the 30th International Conference on Very Large Databases (VLDB)*, 2004, pp. 516-527.
- [46] R. Agrawal, J. Kiernan, R. Srikant and Y. Xu, "Hippocratic Databases," In *28th Int'l Conference on Very Large Databases*, Hong Kong, China, 2002, pp. 143-154.

- [47] L. Cranor, M. Langheinrich, M. Marchiori, M. Pressler-Marshall and J. Reagle, "The Platform for Privacy Preferences 1.0 (P3P1.0) Specification," W3C Recommendation, April 2002.
- [48] J. Chen, D. DeWitt, F. Tian and Y. Wang, "NiagaraCQ: A Scalable Continuous Query System for Internet Databases," In ACM SIGMOD Conference on Management of Data, Dallas, Texas, 2000, pp. 379-390.
- [49] R. Dechter, Constraint Processing, Morgan Kaufman Publishers, 2003.
- [50] K. LeFevre, R. Agrawal, V. Ercegovac, R. Ramakrishnan, Y. Xu and D. DeWitt, "Limiting Disclosure in Hippocratic Databases," In 30th Int'l Conf. on Very Large Data Bases, Toronto, Canada, 2004, pp. 108-119.
- [51] A. Nanda and D. Bursleson, Oracle Privacy Security Auditing, Rampant, 2003.
- [52] R. Ramakrishnan and J. Gehrke, Database Management Systems, McGraw-Hill, 2000.
- [53] A. Elshiekh and P. Dominic, "A New Auditing Scheme for Securing Statistical Databases," International symposium on information technology 2008 (ITSIM 08), KLCC, Kuala Lumpur, Malaysia, 26-29 August, 2008, pp. 234-238.
- [54] A. Elshiekh and P. Dominic, "The Key Representation Auditing Scheme for Securing Statistical Databases," Proceedings of 2008 Student Conference on Research and Development (SCORED 2008), Johor, Malaysia, 26-27 Nov, 2008, pp. 238.1-238.4.
- [55] A. Elshiekh and P. Dominic, "Three Audit Stages for Securing Statistical Databases," 2009 International Conference on Information Management and Engineering (ICIME 2009), 2009, pp. 283-286.
- [56] A. Elshiekh and P. Dominic, "Cost Estimation for the Key Representation Auditing Scheme," 2009 International Conference on Computer Design and

Applications (ICCD2009), Nanyang Executive Centre, Singapore, 10.1109/ICSPS.2009.135, 15-17 May, 2009, pp. 469-473.

- [57] A. Elshiekh and P. Dominic, "Efficient Algorithms for the Key Representation Auditing Scheme," *International Journal of Computer and Electrical Engineering (IJCEE)*, Vol. 1, No. 3, August 2009, pp. 340-349.
- [58] S. Washington, M. Karlaftis and F. Mannering, *Statistical and Econometric Methods for Transportation Data Analysis*, Chapman Hall/CRC, 2003, pp. 23-59.
- [59] D. Morien, *Business Statistics*, Thomson, 2007, pp. 150-177.
- [60] A. Glenberg and M. Andrzejewski, *Learning from Data: An Introduction to Statistical Reasoning*, 3rd ed, Taylor & Francis Group, LLC., 2008, pp. 263-295.
- [61] T. Urdan, *Statistics in plain English*, 2nd ed, Lawrence Erlbaum Associates, Inc., 2005, pp. 89-100.
- [62] R. Ott and M. Longnecker, *An Introduction to Statistical Methods and Data Analysis*, 6th ed, Brooks/Cole, Cengage Learning, 2008, pp. 360-401.
- [63] D. Anderson, D. Sweeney and T. Williams, *Statistics for Business and Economics*, 10th ed, Thomson South-Western, 2009, pp. 338-392.

APPENDIX A

SUBTRACTING P CELLS

A.1 Consider Example 4.1 in Section 4.2.1

$$q_r^1 = q_1^1 - q_2^1 = 2^{**.*} - \overline{212}.*$$

- Here the first cell and the fourth cell should be excluded, because these cells are the common cells between q_1^1 and q_2^1 . So, the first and the fourth cells in q_r^1 should be 2 and *, respectively.
- The second cell (Dept = {1, 2, 3}) (refer to Table 3.4):

$$q_1^1: * \text{ implies } \{1, 2, 3\}$$

$$q_2^1: \overline{1} \text{ implies } \{2, 3\}$$

$$\text{Then, } * - \overline{1} = \{1, 2, 3\} - \{2, 3\} = \{1\} = 1$$

- The third cell (Level = {1, 2, 3}):

$$q_1^1: * \text{ implies } \{1, 2, 3\}$$

$$q_2^1: \overline{2} \text{ implies } \{1, 3\}$$

$$\text{Then, } * - \overline{2} = \{1, 2, 3\} - \{1, 3\} = \{2\} = 2$$

- Accordingly, $q_r^1 = q_1^1 - q_2^1 = 2^{**.*} - \overline{212}.* = 212.* = \{4\}$

- The answer of the result query q_r^1 , referring to Table 3.3 and Table 3.4, is the fourth record. That means the fourth record, namely Saria's record, can be disclosed if the query q_2^1 is permitted.

A.2 Consider Example 4.2 in Section 4.2.1

$$q_r^1 = q_1^1 - q_2^1 = *3*.* - **1.*$$

- Here the first and the fourth cells should be excluded. So, the first and the fourth cells in q_r^1 should be * and *.
- The second cell (Dept = {1, 2, 3}) (refer to Table 3.4):

$$q_1^1: 3 \text{ implies } \{3\}$$

$$q_2^1: * \text{ implies } \{1, 2, 3\}$$

$$\text{Then, } 3 - * = \{3\} - \{1, 2, 3\} = \Phi$$

☞ In this case assume without loss of generality that

$$\text{If } X_{bj} - X_{aj} = \Phi, \text{ then let } X_{rj} = X_{bj}.$$

$$\text{Consequently, } 3 - * = \{3\} - \{1, 2, 3\} = \{3\} = 3$$

- The third cell (Level = {1, 2, 3}):

$$q_1^1: * \text{ implies } \{1, 2, 3\}$$

$$q_2^1: 1 \text{ implies } \{1\}$$

$$\text{Then, } * - 1 = \{1, 2, 3\} - \{1\} = \{2, 3\} = \bar{1}$$

- Accordingly, $q_r^1 = q_1^1 - q_2^1 = *3*.* - **1.* = *3\bar{1}.* = \{5\}$

- The answer of the result query q_r^1 , referring to Table 3.3 and Table 3.4, is the fifth record. That means the fifth record, namely Samy's record, can be disclosed if the query q_2^1 is permitted.

A.3 Consider Example 4.3 in Section 4.2.1

$$q_r^1 = q_1^1 - q_2^1 = 11*.* - **1.*$$

- Here the fourth cell should be excluded. So, the fourth cell in q_r^1 should be *.
- The first cell (Gender = {1, 2}) (refer to Table 4.4):

$$q_1^1: 1 \text{ implies } \{1\}$$

$$q_2^1: * \text{ implies } \{1, 2\}$$

$$\text{Then, } 1 - * = \{1\} - \{1, 2\} = \Phi$$

☞ In this case assume, as mentioned earlier, without loss of generality that

$$\text{If } X_{bj} - X_{aj} = \Phi, \text{ then let } X_{rj} = X_{bj}.$$

$$\text{Consequently, } 1 - * = \{1\} - \{1, 2\} = \{1\} = 1$$

- The second cell (Dept = {1, 2, 3}):

$$q_1^1: 1 \text{ implies } \{1\}$$

$$q_2^1: * \text{ implies } \{1, 2, 3\}$$

$$\text{Then, } 1 - * = \{1\} - \{1, 2, 3\} = \Phi$$

☞ In this case assume, as mentioned earlier, without loss of generality that

If $\mathbf{X}_{bj} - \mathbf{X}_{aj} = \Phi$, then let $\mathbf{X}_{rj} = \mathbf{X}_{bj}$.

Consequently, $1 - * = \{1\} - \{1, 2, 3\} = \{1\} = 1$

- The third cell (Level = {1, 2, 3}):

q_1^1 : * implies {1, 2, 3}

q_2^1 : 1 implies {1}

Then, $* - 1 = \{1, 2, 3\} - \{1\} = \{2, 3\} = \bar{1}$

- Accordingly, $q_r^1 = q_1^1 - q_2^1 = 11*.* - **1.* = 11\bar{1}.* = \{1, 7, 12\}$
- The answer of the result query q_r^1 , referring to Table 3.3 and Table 3.4, is the first, seventh and 12th records. That means permitting the query q_2^1 would not disclose the SDB.

A.4 Consider Example 4.4 in Section 4.2.1

$$q_r^1 = q_1^1 - q_2^1 = *1*.* - \bar{2}1*.*$$

- Here the second, the third and the fourth cells should be excluded. So, the second, the third and the fourth cells in q_r^1 should be 1, * and *, respectively.
- The first cell (Gender = {1, 2}) (refer to Table 3.7):

q_1^1 : * implies {1, 2}

q_2^1 : $\bar{2}$ implies {1}

Then, $* - \bar{2} = \{1, 2\} - \{1\} = \{2\} = 2$

- Accordingly, $q_r^1 = q_1^1 - q_2^1 = *1*.* - \bar{2}1*.* = 21*.* = \{2\}$

- The answer of the result query q_r^1 , referring to Table 3.6 and Table 3.7, is the second record. That means the second record, namely Sara's record, can be disclosed if the query q_2^1 is permitted.

A.5 Consider Example 4.5 in Section 4.2.1

$$q_r^1 = q_1^1 - q_2^1 = **.21.* - *2.21.*$$

- Here the first, the third and the fourth cells should be excluded. So, the first, the third and the fourth cells in q_r^1 should be *, 21 and *, respectively.
- The second cell (Dept = {1, 2}) (refer to Table 3.7):

$$q_1^1: * \text{ implies } \{1, 2\}$$

$$q_2^1: 2 \text{ implies } \{2\}$$

$$\text{Then, } * - 2 = \{1, 2\} - \{2\} = \{1\} = \bar{2}$$

- Accordingly, $q_r^1 = q_1^1 - q_2^1 = **.21.* - *2.21.* = *\bar{2}.21.* = \{8\}$
- The answer of the result query q_r^1 , referring to Table 3.6 and Table 3.7, is the 8th record. That means the 8th record, namely Nasir's record, can be disclosed if the query q_2^1 is permitted.

APPENDIX B

DEVELOPMENT OF KEY REPRESENTATION AUDITING SOFTWARE

B.1 The Proposed Key Representation Auditing Software

The proposed system can be set as a core of auditing online and dynamic SDBs, and it will play a vital role to guarantee the security of SDBs among the three audit stages which discussed previously in chapter four.

The database of the proposed system contains various tables with relationships among them. It stores all the required data such as the original data about the individuals, the key representation data and the answered key representation queries. It does provide the system with all the needed data in order to obtain the final decision and to specify the illegal queries which could lead to disclosing the SDB.

Basically the inputs for this system can be identified in two major groups. Firstly, the original information about individuals, including category and data attributes, such as name, gender, department, salary, etc. Secondly, the user query which may consists of one part or more than one part. The system converts the original data and the user query into key representation data and key representation query, respectively. When a user submits a query to the system, the three audit stages will be applied to decide whether the query is answerable or not.

Considering the first example which was stated earlier in section 3.3.5.1, the next section gives a glimpse of the input screens of the proposed system as well as the system output.

B.2 The Graphical User Interface

From the main screen, the SDB user has three different options available. The first option is storing the basic information of employees using the Employee Data button. The second option is allowing the user to enter his query using the User Query button. The final option is closing the system using the Exit button. Figure B.1 depicts the main screen of the system.

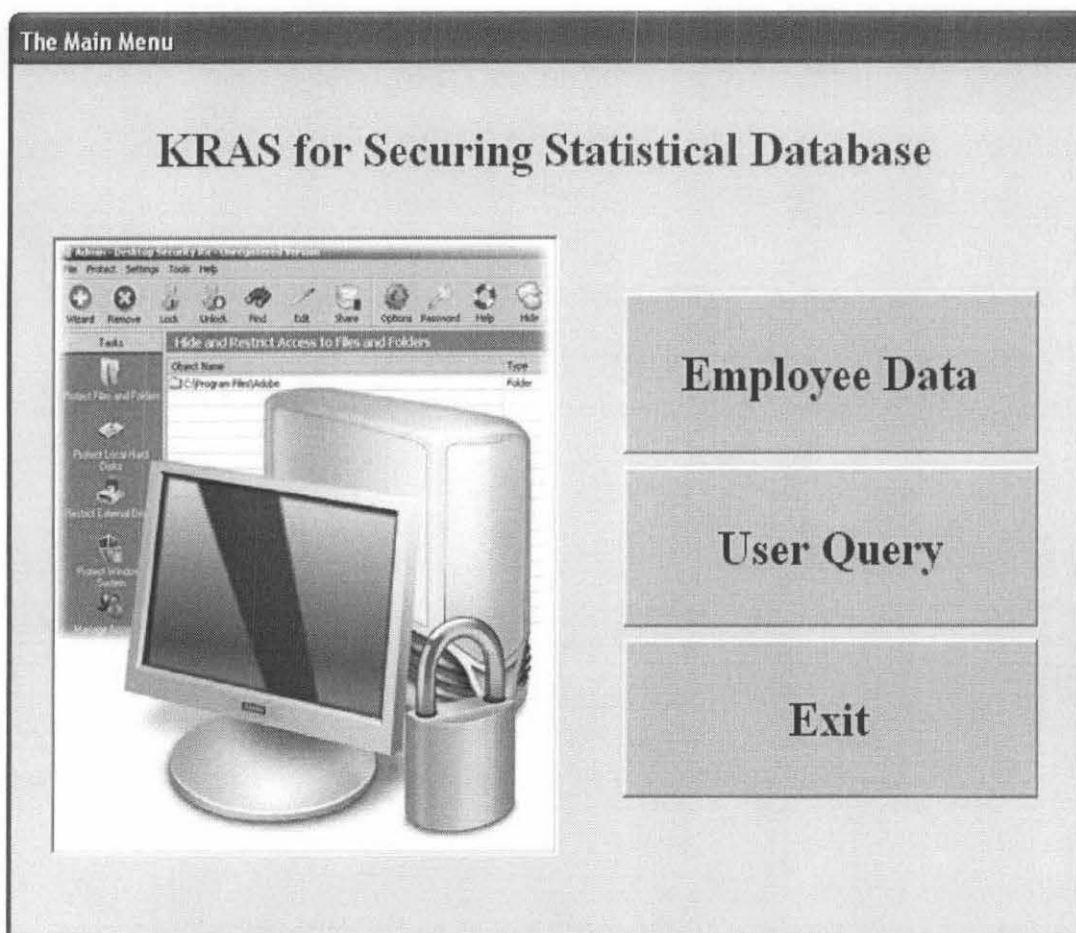


Figure B.1: The Main Screen

B.2.1 Employee Data Screen

From the main screen, when the Employee Data button has been pressed, the employee data screen will immediately appear, as shown below in Figure B.2, which contains fields for entering the employee data. The SDB user can add more employees or manipulate the data using the buttons Add, Save, Cancel, Search, Edit and Delete. Whereas, the buttons Employee Data and KRDB Data are only allowable for the SDB administrator to show him the original data (see Figure B.3) and the key representation data (see Figure B.4), respectively.

The screenshot shows a window titled "Employee Data" with the following fields and buttons:

Employee Name:	<input type="text" value="Adil"/>
Gender:	<input type="text" value="Male"/>
Department:	<input type="text" value="CS"/>
Level:	<input type="text" value="MSc"/>
Salary:	<input type="text" value="200"/>

Add	Save	Cancel	Close
Search	Edit	Delete	
Employee Data	KRDB Data		

Figure B.2: Employee Data Screen

Browse Employees Data

No	Name	Gender	Dept	Level	Salary
1	Adil	Male	CS	MSc	200
2	Omer	Male	EE	MSc	150
3	Sara	Female	EE	MSc	150
4	Saria	Female	CS	MSc	150
5	Samy	Male	PE	MSc	180
6	Zainab	Female	PE	BSc	220
7	Gasim	Male	CS	MSc	100
8	Ahmed	Male	EE	MSc	180
9	Fatima	Female	CS	PhD	30
10	Nasir	Male	PE	BSc	200
11	Mahasin	Female	EE	MSc	250
12	Khalid	Male	CS	PhD	30

Close

Figure B.3: Browse Employees Data Screen

The screenshot shows a window titled "Browse KRDB Data". Inside the window is a table with two columns: "No" and "Record's Key". The table contains 12 rows of data. The row with "No" 8 and "Record's Key" 122.180 is highlighted with a dark background. Below the table is a "Close" button.

No	Record's Key
1	112.200
2	122.150
3	222.150
4	212.150
5	132.180
6	231.220
7	112.100
8	122.180
9	213.30
10	131.200
11	222.250
12	113.30

Close

Figure B.4: Browse KRDB Data Screen

B.2.2 User Query Screen

From the main screen, when the User Query button has been pressed, the user query screen will immediately appear, as shown below in Figure B.5. This screen contains all the category and data attributes, and it allows the user to determine all the conditions of his query. Also, it allows the user to add new parts for the query using the OR button or remove some parts from the query using the X button. For each category attribute, there are two types of Not Sign (NS). The first type is the green not sign (green NS), namely the not sign for the specific category attribute. The corresponding values of the green NS are either 0 if it is not checked or 1 if it is checked by the user. The second type is the red not sign (red NS), namely the not sign for at least two category attributes. The corresponding values of the red NS are either 0 if it is not checked or 2 if it is checked by the user.

The screenshot shows a window titled "User Query" containing a table with four columns: Gender, Dept, Level, and Salary. Each column has a "Not" checkbox and a "Not" dropdown menu. The first row has "Not" checkboxes for Gender (unchecked), Dept (unchecked), and Level (unchecked), with a dropdown menu for each. The second row has "Not" checkboxes for Gender (unchecked), Dept (checked), and Level (unchecked), with a dropdown menu for each. The Salary column has a dropdown menu with "[]" selected, and two input fields containing "100" and "500". There are "OR" buttons at the end of each row and an "X" button at the start of the second row. Below the table are five buttons: "New Query", "Query Result", "Exit", "Browse Result", and "Browse AQ table".

	Gender	Dept	Level	Salary	
	<input type="checkbox"/> Not <input type="checkbox"/> Not Male	<input type="checkbox"/> Not <input type="checkbox"/> Not PE	<input type="checkbox"/> Not <input type="checkbox"/> Not ^	<input type="checkbox"/> Not <input type="checkbox"/> Not ^	OR
X	<input type="checkbox"/> Not <input type="checkbox"/> Not ^	<input checked="" type="checkbox"/> Not <input type="checkbox"/> Not CS	<input type="checkbox"/> Not <input type="checkbox"/> Not ^	<input type="checkbox"/> Not <input type="checkbox"/> Not [] 100 500	OR

Buttons: New Query, Query Result, Exit, Browse Result, Browse AQ table

Figure B.5: User Query Screen

By hitting the Query Result button, the system will apply the three audit stages to decide whether the query could lead to disclosing the SDB or not. If the current user query is answerable, namely it could not lead to disclosing the SDB, the result of the query will immediately appear (see Figure B.6). But if the current user query is not answerable, namely it could lead to disclosing the SDB, a message will appear to show the user that this query is prevented (see Figure B.7).

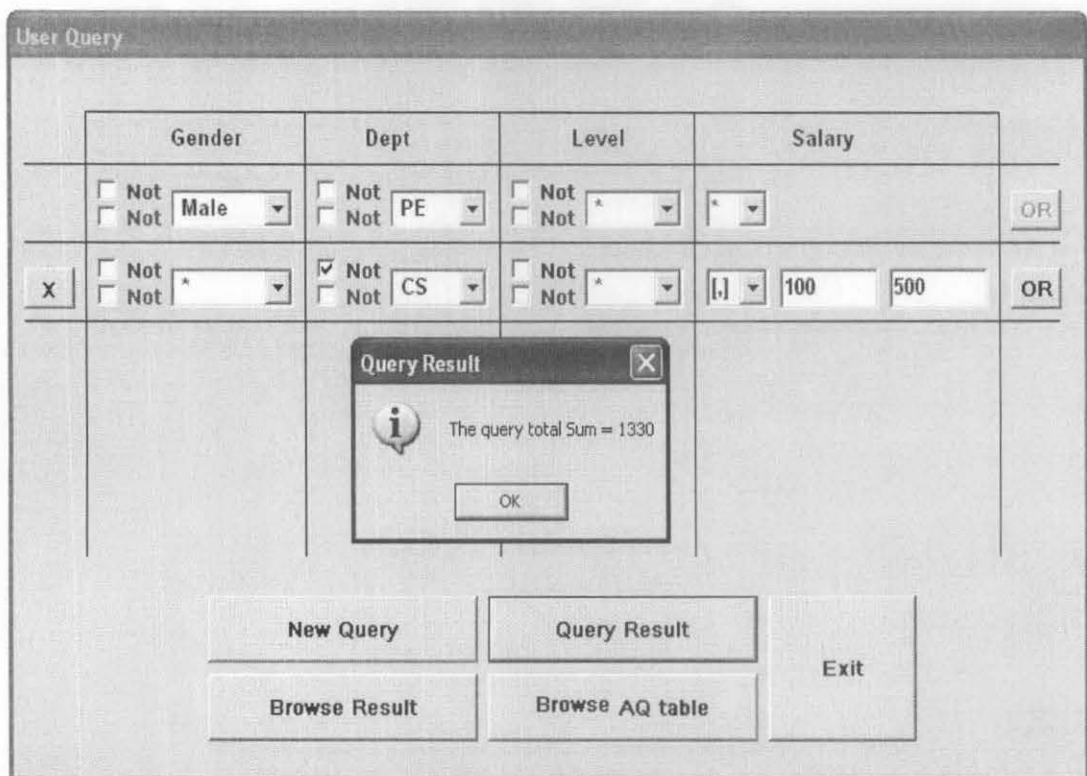


Figure B.6: Permitted Query Result

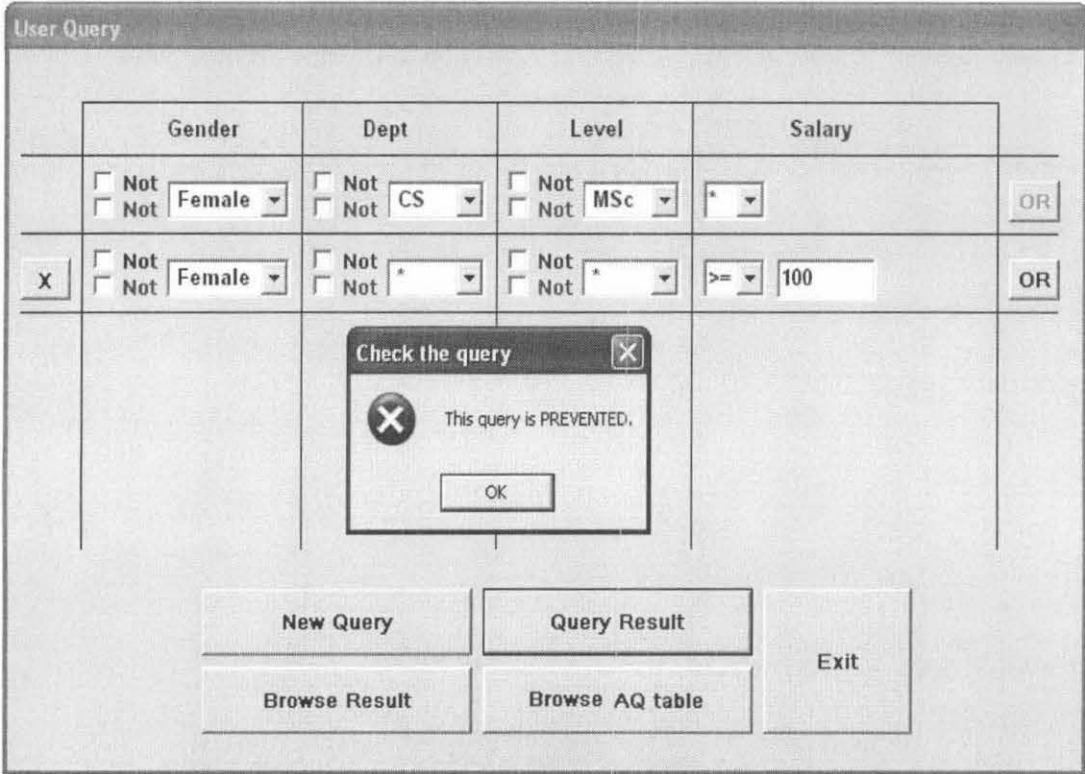


Figure B.7: Prevented Query Message

The buttons Browse Result and Browse AQ Table are only allowable for the SDB administrator, to show him the result of the current query (see Figure B.8) and the answered key representation queries (see Figure B.9), respectively.

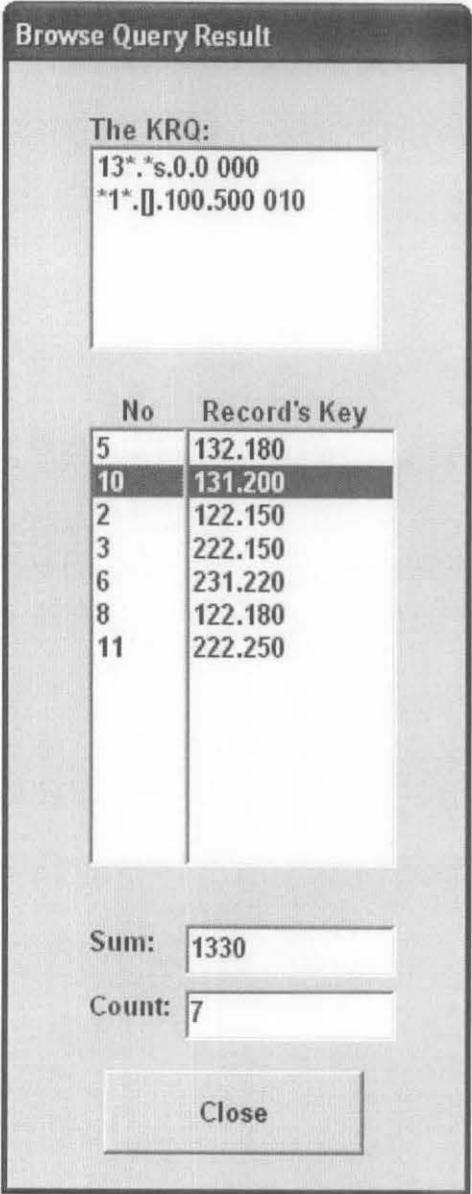


Figure B.8: Browse Query Result Screen

Browse AQ Table			
KRQ	NS	q	L q
1 ^{**} .s.0.0	000	7	7
11 [*] .s.0.0	000	3	3
2 ^{**} .s.0.0	000	5	5
11 [*] .s.0.0	220	9	9
1 [*] 2.<=,500.0	001	2	2
11 [*] .s.0.0	010	4	4
13 [*] .s.0.0	000	2	2
1 [] .].100.500	010	7	7

Close

Figure B.9: Browse AQ Table Screen

APPENDIX C

THE PROGRAM CODE

C.1 Main Form

```
Private Sub cmdEmployeeData_Click()
```

```
    frmEmployeeForm.Show
```

```
End Sub
```

```
Private Sub cmdExit_Click()
```

```
    End
```

```
End Sub
```

```
Private Sub cmdUserQuery_Click()
```

```
    frmUserQuery.Show
```

```
End Sub
```

C.2 Employee Data Form

```
Dim Found As Boolean
```

```
Dim Answer As String
```

```
Dim CurrentRecNo As Integer
```

```
Private Sub Form_Load()
```

```
    Me.cmdSave.Enabled = False
```

```
    Me.cmdCancel.Enabled = False
```

```
Me.cmdEdit.Enabled = False
Me.cmdDelete.Enabled = False
Me.cmdAdd.Enabled = True
Me.cmdClose.Enabled = True
Me.cmdSearch.Enabled = True
Me.txtName.Text = ""
EraseTextBoxes
```

```
End Sub
```

```
Private Sub cmdAdd_Click()
```

```
Me.cmdSave.Enabled = True
Me.cmdCancel.Enabled = True
Me.cmdEdit.Enabled = False
Me.cmdDelete.Enabled = False
Me.cmdAdd.Enabled = False
Me.cmdClose.Enabled = False
Me.cmdSearch.Enabled = False
Me.txtName.Text = ""
EraseTextBoxes
Me.txtName.SetFocus
```

```
End Sub
```

```
Private Sub cmdBrowseEmployee_Click()
```

```
On Error GoTo Err_Handler
```

```
frmBrowseEmployees.lstRecNo.Clear
frmBrowseEmployees.lstName.Clear
frmBrowseEmployees.lstGender.Clear
```



```

frmBrowseEmployees.lstDept.Clear
frmBrowseEmployees.lstLevel.Clear
frmBrowseEmployees.lstSalary.Clear
With Me.datEmployeeData.Recordset
    .MoveFirst
    Do While Not .EOF
        frmBrowseEmployees.lstRecNo.AddItem .Fields!recNo
        frmBrowseEmployees.lstName.AddItem .Fields!empName
        frmBrowseEmployees.lstGender.AddItem .Fields!Gender
        frmBrowseEmployees.lstDept.AddItem .Fields!Dept
        frmBrowseEmployees.lstLevel.AddItem .Fields!Level
        frmBrowseEmployees.lstSalary.AddItem .Fields!Salary
        .MoveNext
    Loop
End With
frmBrowseEmployees.Show
Exit Sub
Err_Handler:
    If Err = 3021 Then
        MsgBox "Sorry Employee Table is empty now.", vbInformation, "Employee
Table"
    End If
End Sub
Private Sub cmdBrowseKRDB_Click()
On Error GoTo Err_Handler
    frmBrowseKRDB.lstRecNo.Clear

```

```

frmBrowseKRDB.lstCategoryKeys.Clear
With Me.datKRDB.Recordset
    .MoveFirst
    Do While Not .EOF
        frmBrowseKRDB.lstRecNo.AddItem .Fields!recNo
        frmBrowseKRDB.lstCategoryKeys.AddItem .Fields!ca0 & .Fields!ca1 &
.Fields!ca2 & "." & .Fields!Salary
        .MoveNext
    Loop
End With
frmBrowseKRDB.Show
Err_Handler:
    If Err = 3021 Then
        MsgBox "Sorry Key Representation Database Table is empty now.",
vbInformation, "KRDB Table"
    End If
End Sub
Private Sub cmdBrwoseAQtbl_Click()
On Error GoTo Err_Handler
    With Me.datAQtbl.Recordset
        .MoveFirst
        Do While Not .EOF
            frmBrowse_AQ_tbl.lstKRQ.AddItem .Fields!ca0 & .Fields!ca1 & .Fields!ca2
& "." _
            & .Fields!Op & "." & .Fields!V1 & "." & .Fields!V2
            frmBrowse_AQ_tbl.lstNS.AddItem .Fields!NS0 & .Fields!NS1 & .Fields!NS2
            frmBrowse_AQ_tbl.lst_q_set_size.AddItem .Fields!q_set_size

```

```

        frmBrowse_AQ_tbl.lst_L_q_set_size.AddItem .Fields!L_q_set_size
        .MoveNext

    Loop

End With

frmBrowse_AQ_tbl.Show

Exit Sub

Err_Handler:

    If Err = 3021 Then

        MsgBox "Sorry Audit Query Table is empty now.", vbInformation, "AQ Table"

    End If

End Sub

Private Sub cmdCancel_Click()

    Me.txtName.Text = ""

    EraseTextBoxes

    MsgBox "Insertion, modification or deletion is cancelled.", vbInformation,
" labelling"

    Me.cmdSave.Enabled = False

    Me.cmdCancel.Enabled = False

    Me.cmdEdit.Enabled = False

    Me.cmdDelete.Enabled = False

    Me.cmdAdd.Enabled = True

    Me.cmdClose.Enabled = True

    Me.cmdSearch.Enabled = True

    Me.cmdAdd.SetFocus

End Sub

Private Sub cmdClose_Click()

```

Unload Me

End Sub

Private Sub cmdDelete_Click()

Dim myCriteria As String

Dim kr As KRDB_rec

myCriteria = "recNo=" & CurrentRecNo

If Found = True Then

Answer = MsgBox("Are you sure?", vbYesNo + vbDefaultButton2 +
vbQuestion, "Deletion")

If Answer = vbYes Then

Me.datEmployeeData.Recordset.Delete

With Me.datKRDB.Recordset

.FindFirst (myCriteria)

If Not .NoMatch Then

kr.ca(0) = .Fields!ca0

kr.ca(1) = .Fields!ca1

kr.ca(2) = .Fields!ca2

kr.Sal = .Fields!Salary

.Delete

End If

End With

MsgBox "Record is deleted.", vbInformation, "Deletion"

Found = False

End If

Else

MsgBox "This name is not found.", vbCritical, "Not found"

```

End If

Call Update_L_q_Set_Size_AQtbl(kr, "Del")

Me.cmdSave.Enabled = False

Me.cmdCancel.Enabled = False

Me.cmdEdit.Enabled = False

Me.cmdDelete.Enabled = False

Me.cmdAdd.Enabled = True

Me.cmdClose.Enabled = True

Me.cmdSearch.Enabled = True

Me.txtName.Text = ""

EraseTextBoxes

Me.cmdAdd.SetFocus

End Sub

Private Sub cmdEdit_Click()

    Dim myCriteria As String

    Dim Old_rec As KRDB_rec

    Dim New_rec As KRDB_rec

    myCriteria = "recNo=" & CurrentRecNo

    If Found Then

        If Trim(Me.txtName) <> "" And Trim(Me.txtSalary) <> "" And
Trim(Me.cmbGender) <> "" And Trim(Me.cmbDept) <> "" And Trim(Me.cmbLevel)
<> "" Then

            If IsNumeric(Me.txtName) Or Not IsNumeric(Me.txtSalary) Then

                MsgBox "Type mismatch.", vbCritical, "Type error"

                Me.txtName.SetFocus

            Else

```

With Me.datEmployeeData.Recordset

.FindFirst (myCriteria)

.Edit

.Fields!empName = Me.txtName

.Fields!Gender = Me.cmbGender

.Fields!Dept = Me.cmbDept

.Fields!Level = Me.cmbLevel

.Fields!Salary = Me.txtSalary

.Update

End With

With Me.datKRDB.Recordset

.FindFirst (myCriteria)

If Not .NoMatch Then

Old_rec.ca(0) = .Fields!ca0

Old_rec.ca(1) = .Fields!ca1

Old_rec.ca(2) = .Fields!ca2

Old_rec.Sal = .Fields!Salary

.Edit

.Fields!Salary = Me.txtSalary

Select Case Me.cmbGender

Case "Male"

.Fields!ca0 = "1"

Case "Female"

.Fields!ca0 = "2"

End Select

```

Select Case Me.cmbDept
    Case "CS"
        .Fields!ca1 = "1"
    Case "EE"
        .Fields!ca1 = "2"
    Case "PE"
        .Fields!ca1 = "3"
End Select

Select Case Me.cmbLevel
    Case "BSc"
        .Fields!ca2 = "1"
    Case "MSc"
        .Fields!ca2 = "2"
    Case "PhD"
        .Fields!ca2 = "3"
End Select

New_rec.ca(0) = .Fields!ca0
New_rec.ca(1) = .Fields!ca1
New_rec.ca(2) = .Fields!ca2
New_rec.Sal = .Fields!Salary
If New_rec.ca(0) <> Old_rec.ca(0) Or _
    New_rec.ca(1) <> Old_rec.ca(1) Or _
    New_rec.ca(2) <> Old_rec.ca(2) Or _
    New_rec.Sal <> Old_rec.Sal Then
    Call Update_L_q_Set_Size_AQtbl(Old_rec, "Del")

```

```

    Call Update_L_q_Set_Size_AQtbl(New_rec, "Add")
    MsgBox "Data is modified", vbInformation, "Edit Data"
Else
    MsgBox "Data is not changed", vbCritical, "Edit Data"
    Me.txtName.Text = ""
    EraseTextBoxes
    Me.cmdSave.Enabled = False
    Me.cmdCancel.Enabled = False
    Me.cmdEdit.Enabled = False
    Me.cmdDelete.Enabled = False
    Me.cmdAdd.Enabled = True
    Me.cmdClose.Enabled = True
    Me.cmdSearch.Enabled = True
    Me.cmdAdd.SetFocus
    .CancelUpdate
    Exit Sub
End If
.Update
End If
End With
End If
Else
    MsgBox "Data is incomplete.", vbCritical, "Incomplete Data"
    Me.txtName.SetFocus
    Exit Sub

```



```

    End If

Else

    MsgBox "This name is not found.", vbCritical, "Not Found"

End If

Me.txtName.Text = ""

EraseTextBoxes

Me.cmdSave.Enabled = False

Me.cmdCancel.Enabled = False

Me.cmdEdit.Enabled = False

Me.cmdDelete.Enabled = False

Me.cmdAdd.Enabled = True

Me.cmdClose.Enabled = True

Me.cmdSearch.Enabled = True

Me.cmdAdd.SetFocus

End Sub

Private Sub cmdSave_Click()

    On Error Resume Next

    Dim myRecNo As Integer

    Dim kr As KRDB_rec

    If Trim(Me.txtName) <> "" And Trim(Me.txtSalary) <> "" And
Trim(Me.cmbGender) <> "" And Trim(Me.cmbDept) <> "" And Trim(Me.cmbLevel)
<> "" Then

        If IsNumeric(Me.txtName) Or Not IsNumeric(Me.txtSalary) Then

            MsgBox "Type mismatch.", vbCritical, "Type error"

            Me.txtName.SetFocus

        Else

```

With Me.datEmployeeData.Recordset

.MoveLast

If .RecordCount = 0 Then

 myRecNo = .RecordCount + 1

Else

 myRecNo = .Fields!recNo + 1

End If

.AddNew

.Fields!recNo = myRecNo

.Fields!empName = Me.txtName

.Fields!Gender = Me.cmbGender

.Fields!Dept = Me.cmbDept

.Fields!Level = Me.cmbLevel

.Fields!Salary = Me.txtSalary

.Update

End With

With Me.datKRDB.Recordset

.AddNew

.Fields!recNo = myRecNo

.Fields!Salary = Me.txtSalary

Select Case Me.cmbGender

 Case "Male"

 .Fields!ca0 = "1"

 Case "Female"

 .Fields!ca0 = "2"

```

End Select

Select Case Me.cmbDept

Case "CS"
    .Fields!ca1 = "1"

Case "EE"
    .Fields!ca1 = "2"

Case "PE"
    .Fields!ca1 = "3"

End Select

Select Case Me.cmbLevel

Case "BSc"
    .Fields!ca2 = "1"

Case "MSc"
    .Fields!ca2 = "2"

Case "PhD"
    .Fields!ca2 = "3"

End Select

kr.ca(0) = .Fields!ca0
kr.ca(1) = .Fields!ca1
kr.ca(2) = .Fields!ca2
kr.Sal = .Fields!Salary

.Update

End With

Call Update_L_q_Set_Size_AQtbl(kr, "Add")

MsgBox "Data is saved", vbInformation, "Save Data"

```

```

    Me.txtName.Text = ""
    EraseTextBoxes
    Me.cmdSave.Enabled = False
    Me.cmdCancel.Enabled = False
    Me.cmdEdit.Enabled = False
    Me.cmdDelete.Enabled = False
    Me.cmdAdd.Enabled = True
    Me.cmdClose.Enabled = True
    Me.cmdSearch.Enabled = True
    Me.cmdAdd.SetFocus
End If
Else
    MsgBox "Data is incomplete.", vbCritical, "Incomplete Data"
    Me.txtName.SetFocus
Exit Sub
End If
End Sub
Private Sub cmdSearch_Click()
    Dim myName As String
    myName = InputBox("Enter Employee name you want:", "Search Name")
    If SearchName(myName) = True Then
        Me.cmdSave.Enabled = False
        Me.cmdCancel.Enabled = True
        Me.cmdEdit.Enabled = True
        Me.cmdDelete.Enabled = True

```

```
Me.cmdAdd.Enabled = False
Me.cmdClose.Enabled = False
Me.cmdSearch.Enabled = False
Me.txtName.SetFocus
Else
Me.cmdSave.Enabled = False
Me.cmdCancel.Enabled = False
Me.cmdEdit.Enabled = False
Me.cmdDelete.Enabled = False
Me.cmdAdd.Enabled = True
Me.cmdClose.Enabled = True
Me.cmdSearch.Enabled = True
Me.cmdAdd.SetFocus
End If
End Sub
```

C.3 Browse Employee Data Form

```
Private Sub cmdClose_Click()
    Unload Me
End Sub
Private Sub cmdRefresh_Click()
    On Error Resume Next
    Me.lstRecNo.Clear
    Me.lstName.Clear
```

```

Me.lstGender.Clear

Me.lstDept.Clear

Me.lstLevel.Clear

Me.lstSalary.Clear

With Me.datEmployeeData.Recordset
    .MoveFirst

    Do While Not .EOF

        Me.lstRecNo.AddItem .Fields!recNo

        Me.lstName.AddItem .Fields!empName

        Me.lstGender.AddItem .Fields!Gender

        Me.lstDept.AddItem .Fields!Dept

        Me.lstLevel.AddItem .Fields!Level

        Me.lstSalary.AddItem .Fields!Salary

        .MoveNext

    Loop

End With

End Sub

Private Sub lstDept_Click()

    Me.lstRecNo.Selected(Me.lstDept.ListIndex) = True

    Me.lstName.Selected(Me.lstDept.ListIndex) = True

    Me.lstGender.Selected(Me.lstDept.ListIndex) = True

    Me.lstLevel.Selected(Me.lstDept.ListIndex) = True

    Me.lstSalary.Selected(Me.lstDept.ListIndex) = True

End Sub

Private Sub lstGender_Click()

```

```
Me.lstRecNo.Selected(Me.lstGender.ListIndex) = True
Me.lstName.Selected(Me.lstGender.ListIndex) = True
Me.lstDept.Selected(Me.lstGender.ListIndex) = True
Me.lstLevel.Selected(Me.lstGender.ListIndex) = True
Me.lstSalary.Selected(Me.lstGender.ListIndex) = True
End Sub
```

```
Private Sub lstLevel_Click()
    Me.lstRecNo.Selected(Me.lstLevel.ListIndex) = True
    Me.lstName.Selected(Me.lstLevel.ListIndex) = True
    Me.lstGender.Selected(Me.lstLevel.ListIndex) = True
    Me.lstDept.Selected(Me.lstLevel.ListIndex) = True
    Me.lstSalary.Selected(Me.lstLevel.ListIndex) = True
End Sub
```

```
Private Sub lstName_Click()
    Me.lstRecNo.Selected(Me.lstName.ListIndex) = True
    Me.lstGender.Selected(Me.lstName.ListIndex) = True
    Me.lstDept.Selected(Me.lstName.ListIndex) = True
    Me.lstLevel.Selected(Me.lstName.ListIndex) = True
    Me.lstSalary.Selected(Me.lstName.ListIndex) = True
End Sub
```

```
Private Sub lstRecNo_Click()
    Me.lstName.Selected(Me.lstRecNo.ListIndex) = True
    Me.lstGender.Selected(Me.lstRecNo.ListIndex) = True
    Me.lstDept.Selected(Me.lstRecNo.ListIndex) = True
    Me.lstLevel.Selected(Me.lstRecNo.ListIndex) = True
```

```
Me.lstSalary.Selected(Me.lstRecNo.ListIndex) = True
End Sub
```

```
Private Sub lstSalary_Click()
    Me.lstRecNo.Selected(Me.lstSalary.ListIndex) = True
    Me.lstName.Selected(Me.lstSalary.ListIndex) = True
    Me.lstGender.Selected(Me.lstSalary.ListIndex) = True
    Me.lstDept.Selected(Me.lstSalary.ListIndex) = True
    Me.lstLevel.Selected(Me.lstSalary.ListIndex) = True
End Sub
```

C.4 Browse KRDB Form

```
Private Sub cmdClose_Click()
```

```
    Unload Me
```

```
End Sub
```

```
Private Sub cmdRefresh_Click()
```

```
    On Error Resume Next
```

```
    Me.lstRecNo.Clear
```

```
    Me.lstCategoryKeys.Clear
```

```
    With Me.datKRDB.Recordset
```

```
        .MoveFirst
```

```
        Do While Not .EOF
```

```
            Me.lstRecNo.AddItem .Fields!recNo
```

```
            Me.lstCategoryKeys.AddItem .Fields!ca0 & .Fields!ca1 & .Fields!ca2 & "." & .Fields!Salary
```

```
            .MoveNext
```



```

        Loop
    End With
End Sub

Private Sub lstCategoryKeys_Click()
    Me.lstRecNo.Selected(Me.lstCategoryKeys.ListIndex) = True
End Sub

Private Sub lstRecNo_Click()
    Me.lstCategoryKeys.Selected(Me.lstRecNo.ListIndex) = True
End Sub

```

C.5 Browse AQ Table Form

```

Private Sub cmdClose_Click(Index As Integer)
    Unload Me
End Sub

Private Sub lst_L_q_set_size_Click()
    Me.lstKRQ.Selected(Me.lst_L_q_set_size.ListIndex) = True
    Me.lstNS.Selected(Me.lst_L_q_set_size.ListIndex) = True
    Me.lst_q_set_size.Selected(Me.lst_L_q_set_size.ListIndex) = True
End Sub

Private Sub lst_q_set_size_Click()
    Me.lstKRQ.Selected(Me.lst_q_set_size.ListIndex) = True
    Me.lstNS.Selected(Me.lst_q_set_size.ListIndex) = True
    Me.lst_L_q_set_size.Selected(Me.lst_q_set_size.ListIndex) = True
End Sub

```

```
Private Sub lstKRQ_Click()  
    Me.lstNS.Selected(Me.lstKRQ.ListIndex) = True  
    Me.lst_q_set_size.Selected(Me.lstKRQ.ListIndex) = True  
    Me.lst_L_q_set_size.Selected(Me.lstKRQ.ListIndex) = True  
End Sub
```

```
Private Sub lstNS_Click()  
    Me.lstKRQ.Selected(Me.lstNS.ListIndex) = True  
    Me.lst_q_set_size.Selected(Me.lstKRQ.ListIndex) = True  
    Me.lst_L_q_set_size.Selected(Me.lstNS.ListIndex) = True  
End Sub
```

C.6 User Query Form

```
Private Sub chk1_NS0_1_Click()  
    If Me.chk1_NS0_1.Value = 1 Then  
        Me.chk1_NS0_2.Value = 0  
    End If  
End Sub
```

```
Private Sub chk1_NS0_2_Click()  
    If Me.chk1_NS0_2.Value = 1 Then  
        Me.chk1_NS0_1.Value = 0  
    End If  
End Sub
```

```
Private Sub chk1_NS1_1_Click()  
    If Me.chk1_NS1_1.Value = 1 Then
```

```

        Me.chk1_NS1_2.Value = 0
    End If
End Sub
Private Sub chk1_NS1_2_Click()
    If Me.chk1_NS1_2.Value = 1 Then
        Me.chk1_NS1_1.Value = 0
    End If
End Sub
Private Sub chk1_NS2_1_Click()
    If Me.chk1_NS2_1.Value = 1 Then
        Me.chk1_NS2_2.Value = 0
    End If
End Sub
Private Sub chk1_NS2_2_Click()
    If Me.chk1_NS2_2.Value = 1 Then
        Me.chk1_NS2_1.Value = 0
    End If
End Sub
Private Sub chk2_NS0_1_Click()
    If Me.chk2_NS0_1.Value = 1 Then
        Me.chk2_NS0_2.Value = 0
    End If
End Sub
Private Sub chk2_NS0_2_Click()
    If Me.chk2_NS0_2.Value = 1 Then

```

```

        Me.chk2_NS0_1.Value = 0
    End If
End Sub
Private Sub chk2_NS1_1_Click()
    If Me.chk2_NS1_1.Value = 1 Then
        Me.chk2_NS1_2.Value = 0
    End If
End Sub
Private Sub chk2_NS1_2_Click()
    If Me.chk2_NS1_2.Value = 1 Then
        Me.chk2_NS1_1.Value = 0
    End If
End Sub
Private Sub chk2_NS2_1_Click()
    If Me.chk2_NS2_1.Value = 1 Then
        Me.chk2_NS2_2.Value = 0
    End If
End Sub
Private Sub chk2_NS2_2_Click()
    If Me.chk2_NS2_2.Value = 1 Then
        Me.chk2_NS2_1.Value = 0
    End If
End Sub
Private Sub chk3_NS0_1_Click()
    If Me.chk3_NS0_1.Value = 1 Then

```

```

        Me.chk3_NS0_2.Value = 0
    End If
End Sub
Private Sub chk3_NS0_2_Click()
    If Me.chk3_NS0_2.Value = 1 Then
        Me.chk3_NS0_1.Value = 0
    End If
End Sub
Private Sub chk3_NS1_1_Click()
    If Me.chk3_NS1_1.Value = 1 Then
        Me.chk3_NS1_2.Value = 0
    End If
End Sub
Private Sub chk3_NS1_2_Click()
    If Me.chk3_NS1_2.Value = 1 Then
        Me.chk3_NS1_1.Value = 0
    End If
End Sub
Private Sub chk3_NS2_1_Click()
    If Me.chk3_NS2_1.Value = 1 Then
        Me.chk3_NS2_2.Value = 0
    End If
End Sub
Private Sub chk3_NS2_2_Click()
    If Me.chk3_NS2_2.Value = 1 Then

```

```

        Me.chk3_NS2_1.Value = 0
    End If
End Sub
Private Sub chk4_NS0_1_Click()
    If Me.chk4_NS0_1.Value = 1 Then
        Me.chk4_NS0_2.Value = 0
    End If
End Sub
Private Sub chk4_NS0_2_Click()
    If Me.chk4_NS0_2.Value = 1 Then
        Me.chk4_NS0_1.Value = 0
    End If
End Sub
Private Sub chk4_NS1_1_Click()
    If Me.chk4_NS1_1.Value = 1 Then
        Me.chk4_NS1_2.Value = 0
    End If
End Sub
Private Sub chk4_NS1_2_Click()
    If Me.chk4_NS1_2.Value = 1 Then
        Me.chk4_NS1_1.Value = 0
    End If
End Sub
Private Sub chk4_NS2_1_Click()
    If Me.chk4_NS2_1.Value = 1 Then

```

```

        Me.chk4_NS2_2.Value = 0
    End If
End Sub
Private Sub chk4_NS2_2_Click()
    If Me.chk4_NS2_2.Value = 1 Then
        Me.chk4_NS2_1.Value = 0
    End If
End Sub
Private Sub chk5_NS0_1_Click()
    If Me.chk5_NS0_1.Value = 1 Then
        Me.chk5_NS0_2.Value = 0
    End If
End Sub
Private Sub chk5_NS0_2_Click()
    If Me.chk5_NS0_2.Value = 1 Then
        Me.chk5_NS0_1.Value = 0
    End If
End Sub
Private Sub chk5_NS1_1_Click()
    If Me.chk5_NS1_1.Value = 1 Then
        Me.chk5_NS1_2.Value = 0
    End If
End Sub
Private Sub chk5_NS1_2_Click()
    If Me.chk5_NS1_2.Value = 1 Then

```

```

        Me.chk5_NS1_1.Value = 0
    End If
End Sub
Private Sub chk5_NS2_1_Click()
    If Me.chk5_NS2_1.Value = 1 Then
        Me.chk5_NS2_2.Value = 0
    End If
End Sub
Private Sub chk5_NS2_2_Click()
    If Me.chk5_NS2_2.Value = 1 Then
        Me.chk5_NS2_1.Value = 0
    End If
End Sub
Private Sub cmb1_Dept_Click()
    If Me.cmb1_Dept.List(Me.cmb1_Dept.ListIndex) = "*" Then
        Me.chk1_NS1_1.Value = 0
        Me.chk1_NS1_2.Value = 0
    End If
End Sub
Private Sub cmb1_Gender_Click()
    If Me.cmb1_Gender.List(Me.cmb1_Gender.ListIndex) = "*" Then
        Me.chk1_NS0_1.Value = 0
        Me.chk1_NS0_2.Value = 0
    End If
End Sub

```



```

Private Sub cmb1_Level_Click()
    If Me.cmb1_Level.List(Me.cmb1_Level.ListIndex) = "*" Then
        Me.chk1_NS2_1.Value = 0
        Me.chk1_NS2_2.Value = 0
    End If
End Sub

Private Sub cmb1_Op_Click()
    Me.txt1_V1.Text = ""
    Me.txt1_V2.Text = ""
    If Me.cmb1_Op.List(Me.cmb1_Op.ListIndex) = "*" Then
        Me.txt1_V1.Visible = False
        Me.txt1_V2.Visible = False
    ElseIf Me.cmb1_Op.List(Me.cmb1_Op.ListIndex) = "=" Or _
        Me.cmb1_Op.List(Me.cmb1_Op.ListIndex) = "<" Or _
        Me.cmb1_Op.List(Me.cmb1_Op.ListIndex) = ">" Or _
        Me.cmb1_Op.List(Me.cmb1_Op.ListIndex) = ">=" Or _
        Me.cmb1_Op.List(Me.cmb1_Op.ListIndex) = "<" Or _
        Me.cmb1_Op.List(Me.cmb1_Op.ListIndex) = "<=" Then
        Me.txt1_V1.Visible = True
        Me.txt1_V2.Visible = False
    ElseIf Me.cmb1_Op.List(Me.cmb1_Op.ListIndex) = "[,]" Or _
        Me.cmb1_Op.List(Me.cmb1_Op.ListIndex) = "(,]" Or _
        Me.cmb1_Op.List(Me.cmb1_Op.ListIndex) = "[,)" Or _
        Me.cmb1_Op.List(Me.cmb1_Op.ListIndex) = "(,)" Then
        Me.txt1_V1.Visible = True

```

```

        Me.txt1_V2.Visible = True
    End If
End Sub
Private Sub cmd1_Or_Click()
    If Part1_Validation = False Then
        Exit Sub
    End If
    Parts_ctr = Parts_ctr + 1
    Fill_krq0
    Me.txt2_V1.Visible = False
    Me.txt2_V2.Visible = False
    Me.chk2_NS0_1.Value = 0
    Me.chk2_NS0_2.Value = 0
    Me.chk2_NS1_1.Value = 0
    Me.chk2_NS1_2.Value = 0
    Me.chk2_NS2_1.Value = 0
    Me.chk2_NS2_2.Value = 0
    Me.chk2_NS0_1.Visible = True
    Me.chk2_NS0_2.Visible = True
    Me.chk2_NS1_1.Visible = True
    Me.chk2_NS1_2.Visible = True
    Me.chk2_NS2_1.Visible = True
    Me.chk2_NS2_2.Visible = True
    Me.cmd2_Cancel.Visible = True
    Me.cmb2_Gender.Visible = True

```

```

Me.cmb2_Dept.Visible = True
Me.cmb2_Level.Visible = True
Me.cmb2_Op.Visible = True
Me.cmd2_Or.Visible = True
Me.h_L4.Visible = True
Me.chk2_NS0_1 = 0
Me.chk2_NS0_2 = 0
Me.cmb2_Gender.Text = ""
Me.chk2_NS1_1 = 0
Me.chk2_NS1_2 = 0
Me.cmb2_Dept.Text = ""
Me.chk2_NS2_1 = 0
Me.chk2_NS2_2 = 0
Me.cmb2_Level.Text = ""
Me.cmb2_Op.Text = ""
Me.txt2_V1.Text = ""
Me.txt2_V2.Text = ""
Me.cmb2_Gender.SetFocus
Me.cmd1_Or.Enabled = False
Me.cmdBrowseResult.Enabled = False
End Sub
Private Sub cmd2_Cancel_Click()
    Dim Answer As String
    If Parts_ctr > 2 Then
        If Parts_ctr = 3 Then

```

```
MsgBox "You have to remove (Part 3) before removing this part.", vbCritical, "Removing the current part"
```

```
Exit Sub
```

```
End If
```

```
If Parts_ctr = 4 Then
```

```
MsgBox "You have to remove (Part 4 and Part 3) before removing this part.", vbCritical, "Removing the current part"
```

```
Exit Sub
```

```
End If
```

```
If Parts_ctr = 5 Then
```

```
MsgBox "You have to remove (Part 5, Part 4, and Part 3) before removing this part.", vbCritical, "Removing the current part"
```

```
Exit Sub
```

```
End If
```

```
End If
```

```
Answer = MsgBox("Are you sure you want to remove this part (Part 2) from your query?", vbYesNo + vbQuestion, "Removing the current part")
```

```
If Answer = vbYes Then
```

```
Parts_ctr = Parts_ctr - 1
```

```
Me.chk2_NS0_1.Visible = False
```

```
Me.chk2_NS0_2.Visible = False
```

```
Me.chk2_NS1_1.Visible = False
```

```
Me.chk2_NS1_2.Visible = False
```

```
Me.chk2_NS2_1.Visible = False
```

```
Me.chk2_NS2_2.Visible = False
```

```
Me.cmd1_Or.Enabled = True
```

```
Me.cmdBrowseResult.Enabled = False
```

```

    Me.cmd1_Or.SetFocus
    Me.cmd2_Cancel.Visible = False
    Me.cmb2_Gender.Visible = False
    Me.cmb2_Dept.Visible = False
    Me.cmb2_Level.Visible = False
    Me.cmb2_Op.Visible = False
    Me.txt2_V1.Visible = False
    Me.txt2_V2.Visible = False
    Me.cmd2_Or.Visible = False
    Me.h_L4.Visible = False
Else
    Exit Sub
End If
End Sub
Private Sub cmdBrowseResult_Click()
    frmBrowseQueryResult.Show
    Me.cmdBrowseResult.Enabled = False
End Sub
Private Sub cmdClear_Click()
    Me.Cls
End Sub
Private Sub cmdBrwoseAQtbl_Click()
On Error GoTo Err_Handler
    With Me.datAQ_table.Recordset
        .MoveFirst

```

```

Do While Not .EOF

    frmBrowse_AQ_tbl.lstKRQ.AddItem .Fields!ca0 & .Fields!ca1 & .Fields!ca2
    & "." _
        & .Fields!Op & "." & .Fields!V1 & "." & .Fields!V2

    frmBrowse_AQ_tbl.lstNS.AddItem .Fields!NS0 & .Fields!NS1 & .Fields!NS2

    frmBrowse_AQ_tbl.lst_q_set_size.AddItem .Fields!q_set_size

    frmBrowse_AQ_tbl.lst_L_q_set_size.AddItem .Fields!L_q_set_size

    .MoveNext

Loop

End With

frmBrowse_AQ_tbl.Show

Exit Sub

Err_Handler:

If Err = 3021 Then

    MsgBox "Sorry Audit Query Table is empty now.", vbInformation, "AQ Table"

End If

End Sub

Private Sub cmdExit_Click()

    Unload Me

End Sub

Private Sub cmdNewQuery_Click()

    Me.txt1_V1.Visible = False

    Me.txt1_V2.Visible = False

    Me.cmdBrowseResult.Enabled = False

    Me.cmd1_Or.Enabled = True

    Parts_ctr = 1

```

Me.chk1_NS0_1.Value = 0
Me.chk1_NS0_2.Value = 0
Me.chk1_NS1_1.Value = 0
Me.chk1_NS1_2.Value = 0
Me.chk1_NS2_1.Value = 0
Me.chk1_NS2_2.Value = 0
Me.chk2_NS0_1.Visible = False
Me.chk2_NS0_2.Visible = False
Me.chk2_NS1_1.Visible = False
Me.chk2_NS1_2.Visible = False
Me.chk2_NS2_1.Visible = False
Me.chk2_NS2_2.Visible = False
Me.cmd2_Cancel.Visible = False
Me.cmb2_Gender.Visible = False
Me.cmb2_Dept.Visible = False
Me.cmb2_Level.Visible = False
Me.cmb2_Op.Visible = False
Me.txt2_V1.Visible = False
Me.txt2_V2.Visible = False
Me.cmd2_Or.Visible = False
Me.chk3_NS0_1.Visible = False
Me.chk3_NS0_2.Visible = False
Me.chk3_NS1_1.Visible = False
Me.chk3_NS1_2.Visible = False
Me.chk3_NS2_1.Visible = False

Me.chk3_NS2_2.Visible = False
Me.cmd3_Cancel.Visible = False
Me.cmb3_Gender.Visible = False
Me.cmb3_Dept.Visible = False
Me.cmb3_Level.Visible = False
Me.cmb3_Op.Visible = False
Me.txt3_V1.Visible = False
Me.txt3_V2.Visible = False
Me.cmd3_Or.Visible = False
Me.chk4_NS0_1.Visible = False
Me.chk4_NS0_2.Visible = False
Me.chk4_NS1_1.Visible = False
Me.chk4_NS1_2.Visible = False
Me.chk4_NS2_1.Visible = False
Me.chk4_NS2_2.Visible = False
Me.cmd4_Cancel.Visible = False
Me.cmb4_Gender.Visible = False
Me.cmb4_Dept.Visible = False
Me.cmb4_Level.Visible = False
Me.cmb4_Op.Visible = False
Me.txt4_V1.Visible = False
Me.txt4_V2.Visible = False
Me.cmd4_Or.Visible = False
Me.chk5_NS0_1.Visible = False
Me.chk5_NS0_2.Visible = False

Me.chk5_NS1_1.Visible = False
Me.chk5_NS1_2.Visible = False
Me.chk5_NS2_1.Visible = False
Me.chk5_NS2_2.Visible = False
Me.cmd5_Cancel.Visible = False
Me.cmb5_Gender.Visible = False
Me.cmb5_Dept.Visible = False
Me.cmb5_Level.Visible = False
Me.cmb5_Op.Visible = False
Me.txt5_V1.Visible = False
Me.txt5_V2.Visible = False
Me.h_L4.Visible = False
Me.h_L5.Visible = False
Me.h_L6.Visible = False
Me.h_L7.Visible = False
Me.chk1_NS0_1 = 0
Me.chk1_NS0_2 = 0
Me.cmb1_Gender.Text = ""
Me.chk1_NS1_1 = 0
Me.chk1_NS1_2 = 0
Me.cmb1_Dept.Text = ""
Me.chk1_NS2_1 = 0
Me.chk1_NS2_2 = 0
Me.cmb1_Level.Text = ""
Me.cmb1_Op.Text = ""

```

Me.txt1_V1.Text = ""
Me.txt1_V2.Text = ""
Me.cmb1_Gender.SetFocus
For x = 0 To no_of_parts - 1
    For y = 0 To 2
        krq(x).ca(y) = ""
        krq(x).NS(y) = 0
    Next y
    krq(x).Op = ""
    krq(x).V1 = 0
    krq(x).V2 = 0
    Part_Size(x) = 0
Next x
End Sub
Private Sub cmdQueryResult_Click()
    Dim Part_Index As Integer
    For x = 0 To (no_of_parts - 1)
        For y = 0 To 2
            krq(x).ca(y) = ""
            krq(x).NS(y) = 0
        Next y
        krq(x).Op = ""
        krq(x).V1 = 0
        krq(x).V2 = 0
        Part_Size(x) = 0
    
```

```

Next x

frmBrowseQueryResult.lstKRQ.Clear

frmBrowseQueryResult.lstRecNo.Clear

frmBrowseQueryResult.lstCategoryKeys.Clear

If Parts_ctr = 1 Then
    If Part1_Validation = False Then
        Me.cmb1_Gender.SetFocus
        Exit Sub
    End If
    Fill_krq0
ElseIf Parts_ctr = 2 Then
    If Part1_Validation = False Then
        Me.cmb1_Gender.SetFocus
        Exit Sub
    End If
    If Part2_Validation = False Then
        Me.cmb2_Gender.SetFocus
        Exit Sub
    End If
    Fill_krq0
    Fill_krq1
ElseIf Parts_ctr = 3 Then
    If Part1_Validation = False Then
        Me.cmb1_Gender.SetFocus
        Exit Sub

```

```
End If

If Part2_Validation = False Then
    Me.cmb2_Gender.SetFocus
    Exit Sub
End If

If Part3_Validation = False Then
    Me.cmb3_Gender.SetFocus
    Exit Sub
End If

Fill_krq0
Fill_krq1
Fill_krq2

ElseIf Parts_ctr = 4 Then
    If Part1_Validation = False Then
        Me.cmb1_Gender.SetFocus
        Exit Sub
    End If
    If Part2_Validation = False Then
        Me.cmb2_Gender.SetFocus
        Exit Sub
    End If
    If Part3_Validation = False Then
        Me.cmb3_Gender.SetFocus
        Exit Sub
    End If
```

```
If Part4_Validation = False Then
    Me.cmb4_Gender.SetFocus
    Exit Sub
End If
Fill_krq0
Fill_krq1
Fill_krq2
Fill_krq3
ElseIf Parts_ctr = 5 Then
    If Part1_Validation = False Then
        Me.cmb1_Gender.SetFocus
        Exit Sub
    End If
    If Part2_Validation = False Then
        Me.cmb2_Gender.SetFocus
        Exit Sub
    End If
    If Part3_Validation = False Then
        Me.cmb3_Gender.SetFocus
        Exit Sub
    End If
    If Part4_Validation = False Then
        Me.cmb4_Gender.SetFocus
        Exit Sub
    End If
```

```

If Part5_Validation = False Then
    Me.cmb5_Gender.SetFocus
    Exit Sub
End If
Fill_krq0
Fill_krq1
Fill_krq2
Fill_krq3
Fill_krq4
End If
If Query_Validation = False Then
    Me.cmb1_Gender.SetFocus
    Exit Sub
End If
Intersection = 0
q_Sum = 0
q_Count = 0
j = 0
For Part_Index = 0 To (Parts_ctr - 1)
    Call Select_Recs(krq(Part_Index), Part_Index)
Next Part_Index
frmBrowseQueryResult.txtSum = q_Sum
frmBrowseQueryResult.txtCount = q_Count
Call Check_the_Query
Me.cmdBrowseResult.Enabled = True

```

End Sub

C.7 Browse Query Result Form

```
Private Sub cmdClose_Click()
```

```
    Unload Me
```

```
End Sub
```

```
Private Sub lstCategoryKeys_Click()
```

```
    Me.lstRecNo.Selected(Me.lstCategoryKeys.ListIndex) = True
```

```
End Sub
```

```
Private Sub lstRecNo_Click()
```

```
    Me.lstCategoryKeys.Selected(Me.lstRecNo.ListIndex) = True
```

```
End Sub
```