

**SLICING: A NEW APPROACH FOR PRIVACY  
PRESERVING DATA PUBLISHING**

A PROJECT REPORT

*Submitted by*

BL.EN.U4CSE09002

P. ABINAYA

BL.EN.U4CSE09075

K.LAKSHMAN MADHAV

BL.EN.U4CSE08061

M. NAGA HITESH

*In partial fulfillment for the award of the degree  
Of*

**BACHELOR OF TECHNOLOGY**

IN

COMPUTER SCIENCE AND ENGINEERING



AMRITA SCHOOL OF ENGINEERING, BANGALORE

AMRITA VISHWA VIDYAPEETHAM

BANGALORE 560 035

MAY – 2013

**AMRITA VISHWA VIDYAPEETHAM**  
**AMRITA SCHOOL OF ENGINEERING, BANGALORE, 560035**



**BONAFIDE CERTIFICATE**

This is to certify that the project report entitled “ **SLICING: A NEW APPROACH FOR PRIVACY PRESERVING DATA PUBLISHING** ” submitted by

**BL.EN.U4CSE09002**

**P.ABINAYA**

**BL.EN.U4CSE09075**

**K.LAKSHMAN MADHAV**

**BL.EN.U4CSE08061**

**M.NAGA HITESH**

in partial fulfillment of the requirements for the award of the **Degree Bachelor of Technology** in “**COMPUTER SCIENCE AND ENGINEERING**” is a bonafide record of the work carried out under our guidance and supervision at Amrita School of Engineering, Bangalore.

**Ms. Sreevidya B**  
Internal Supervisor  
Lecturer  
CSE Department

**Dr.T.S.B Sudarshan**  
Chairperson  
CSE Department

This project report was evaluated by us on .....

INTERNAL EXAMINER

EXTERNAL EXAMINER

## ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people, whose constant guidance and encouragement crowned our effort with success.

We express our whole hearted gratitude towards **Mata Amritanandamayi Devi**, who showered us all blessings to complete our task.

We express our gratitude to **Dr. Rakesh S.G**, Associate Dean, for providing us with necessary facilities to carry out this project.

We wish to place on record our grateful thanks to the Head of the Department of Computer Science and Engineering, **Dr. T.S.B Sudarshan**, for his constant support and encouragement.

We are glad to mention the invaluable help and support in motivating us in successfully completing the project provided by our guide **Ms. Sreevidya B**, Lecturer, Department of Computer Science and Engineering, who constantly guided us in every step we take.

We wish to thank all the Teaching Staff and Non Teaching Staff of Computer Science Department for providing us all the support whenever needed.

Last but not the least we would like to thank our parents and all our friends without whose co-operation, the completion of project would not have been possible.

## ABSTRACT

Privacy preservation has emerged in the recent half decade as one of the more intriguing aspects of data mining. This is due to both rising concerns about rights violation using data mining and to the emergence of important markets for this type of applications. This area of research is rapidly maturing. Unfortunately, recent studies all point to one major deficiency: the lack of a well-defined way of modeling the privacy retained by a privacy preserving data mining algorithm.

Various organizations, such as hospitals, medical administrations and insurance companies, have collected a large amount of data over years. However, gold nuggets in these data are unlikely to be discovered if the data is locked in data custodians' storage. A major risk of releasing data for public research is revealing the private information of individuals in data. Disclosure-control is a traditional approach for privacy-preserving data releasing. Most of them concentrate on maintaining statistics of data invariant. Disclosure-control methodology revives in the data mining community due to privacy concerns in powerful data mining processes. The data mining community aims to build strong privacy-preserving models and design efficient, optimal and scalable heuristic solutions.

Several anonymization techniques, such as generalization and bucketization, have been designed for privacy preserving microdata publishing. Recent work has shown that generalization loses considerable amount of information, especially for high-dimensional data. Bucketization, on the other hand, does not prevent membership disclosure and does not apply for data that do not have a clear separation between quasi-identifying attributes and sensitive attributes. In this project, we present a novel technique called slicing, which partitions the data both horizontally and vertically. We show that slicing preserves better data utility than generalization and can be used for membership disclosure protection. Another important advantage of slicing is that it can handle high-dimensional data.

## TABLE OF CONTENTS

1. INTRODUCTION .....	2
1.1 Privacy Preserving Data Publishing .....	2
1.2 Phases of Privacy-Preserving Data Publishing .....	2
1.3 Privacy Models .....	4
1.4 Purpose & Objective .....	4
1.5 Anonymization Techniques .....	6
1.6 Generalization .....	6
1.7 Bucketization .....	8
1.8 Slicing .....	9
2.LITERATURE REVIEW .....	12
2.1 Motivation .....	12
2.2 Models of Data Publishing.....	13
2.3 The Anonymization Approach .....	13
2.4 Privacy Models .....	15
2.5 Record Linkage .....	16
2.6 Attribute Linkage .....	17
2.7 Table Linkage .....	19
2.8 Anonymization Operations .....	19
2.9 Privacy Vs Utility .....	20
3. REQUIREMENT SPECIFICATION .....	25
3.1 Introduction.....	25
3.1.1 Purpose.....	25
3.1.2 Scope.....	25
3.1.3 Definitions, Acronyms and Abbreviations.....	25
3.1.4 References.....	25
3.1.5 Overview .....	26
3.2 Functionality .....	26
3.2.1 Authentication.....	26
3.2.2 Admin Login .....	26
3.2.3 Employee Login.....	26

3.2.4 Anonymization Techniques .....	26
3.3 Usability .....	27
3.3.1 User Friendly.....	27
3.4 Reliability .....	27
3.5 Performance .....	27
3.6 Supportability .....	28
3.6.1 Microsoft.....	28
3.6.2 MySQL.....	28
3.7 Design Constraints .....	28
3.7.1 Windows 7 .....	28
3.8 Interfaces .....	28
3.8.1 User Interface .....	28
3.8.2 Hardware Interface.....	29
3.8.3 Software Interface .....	29
4. SYSTEM DESIGN .....	31
4.1 Process Model .....	31
4.2 System Architecture .....	33
4.3 Activity Diagram.....	34
4.4 Sequence Diagram .....	35
4.5 Data Flow Diagram .....	36
4.6 Use Case Diagram.....	37
4.7 Block Diagram .....	38
5. SYSTEM IMPLEMENTATION .....	40
5.1 Implementation Details .....	40
5.1.1 Coding for GUI .....	40
5.1.2 Coding for Anonymization Techniques .....	40
5.2 Modules .....	40
5.2.1 Registration .....	28
5.2.2 Employee Validation & Managing Users .....	41
5.2.3 Generalization .....	41
5.2.4 Bucketization.....	42
5.2.5 Slicing .....	42
5.2.6 Employee Details .....	43

5.3 Comparison with Generalization .....	43
5.4 Comparison with Bucketization .....	44
5.5 Attribute Disclosure Protection .....	45
5.6 Membership Disclosure Protection .....	46
6. TESTING .....	48
6.1 Introduction .....	48
6.2 Hardware Testing .....	48
6.3 Software Testing .....	48
6.4 Testing Methods .....	50
6.5 Testing Process .....	50
6.5.1 Unit Testing .....	51
6.5.2 Integration Testing .....	51
6.5.3 System Testing .....	52
6.6 Test Cases .....	53
6.6.1 Testing Module 1 .....	53
6.6.2 Testing Module 2 .....	55
7. RESULTS AND ANALYSIS .....	58
7.1 Index Page .....	58
7.2 Employee Registration .....	59
7.3 Admin Menu .....	60
7.3.1 Non Anonymized Data .....	62
7.3.2 Generalization .....	63
7.3.3 Bucketization .....	64
7.3.4 Slicing .....	65
7.4 Employee Menu .....	66
8. CONCLUSION .....	69
9. FUTURE ENHANCEMENTS .....	71
10. REFERENCES .....	73

## LIST OF FIGURES

<b>Figure No.</b>	<b>Description</b>	<b>Page No.</b>
1.2	Data Collection and Data Publishing	3
2.3	Record Linkage [Sweeny's example]	14
4.2	Architecture Diagram	34
4.3	Activity Diagram	34
4.4	Sequence Diagram	35
4.5.1	Level 0 DFD	36
4.5.2	Level 1 DFD	36
4.6	Use Case Diagram	37
4.7	Block Diagram	38
6.6.1.1	Result of Testing for Registration(Email Id)	54
6.6.1.2	Result of Testing for Registration(Unique User Id)	54
6.6.2.1	Result of Testing for Admin Validation	56
7.1	Index Page	58
7.2	Employee Registration Form	59
7.3.1	Admin Menu	60
7.3.2	View All users	61
7.3.3	Update Users	61
7.3.4	Delete users	62
7.3.1.1	Non Anonymized Data	63
7.3.2.1	Generalized Data	64
7.3.3.1	Bucketized Data	65
7.3.4.1	Sliced Data	66
7.4.1	Employee Menu	67
7.4.2	Employee Self Details	67



## LIST OF TABLES

<b>Table No.</b>	<b>Description</b>	<b>Page No.</b>
1.6.1	Raw Medical Data Set	7
1.6.2	Generalized Data Set	7
1.7	Bucketized Data Set	9
1.8	Sliced Data Set	10
2.5	Examples Illustrating Various Attacks	17
5.3	Generalization	44
5.4	Bucketization	45
5.5	Slicing	45
6.6.1	Test case for Registration	53
6.6.2	Test case for Admin Validation	55

# **Chapter 1**

# **INTRODUCTION**

# 1. INTRODUCTION

## 1.1 Privacy Preserving Data Publishing

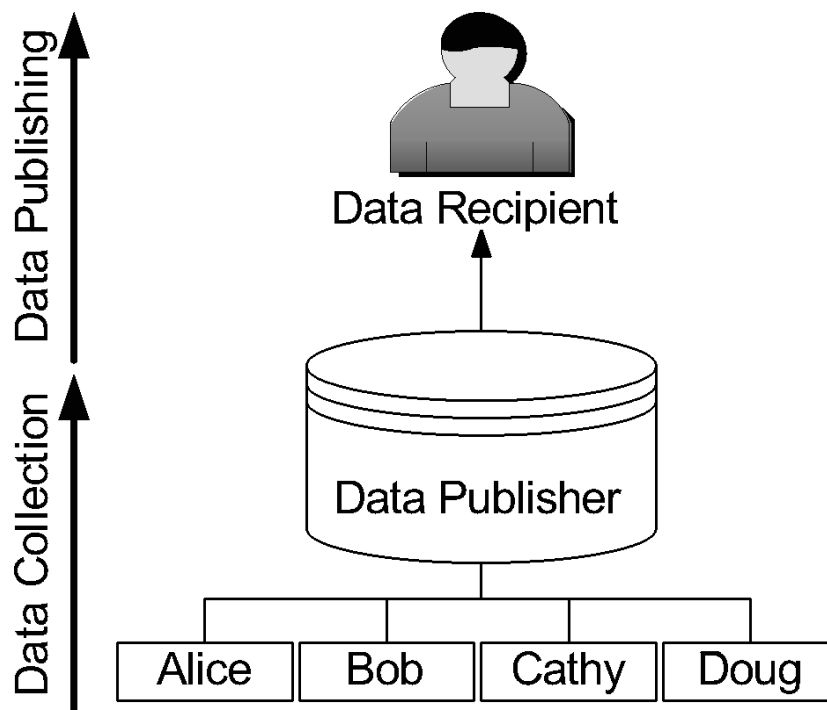
Detailed person-specific data in its original form often contains sensitive information about individuals, and publishing such data immediately violates individual privacy. The current practice primarily relies on policies and guidelines to restrict the types of publishable data and on agreements on the use and storage of sensitive data. The limitation of this approach is that it either distorts data excessively or requires a trust level that is impractically high in many data-sharing scenarios.

For example, contracts and agreements cannot guarantee that sensitive data will not be carelessly misplaced and end up in the wrong hands. A task of the utmost importance is to develop methods and tools for publishing data in a more hostile environment, so that the published data remains practically useful while individual privacy is preserved. This undertaking is called privacy-preserving data publishing (PPDP). In the past few years, research communities have responded to this challenge and proposed many approaches. While the research field is still rapidly developing, it is a good time to discuss the assumptions and desirable properties for PPDP, clarify the differences and requirements that distinguish PPDP from other related problems, and systematically summarize and evaluate different approaches to PPDP.

## 1.2 Phases of Privacy-Preserving Data Publishing

A typical scenario for data collection and publishing is described in Figure 1.2. In the data collection phase, the data publisher collects data from record owners (e.g., Alice and Bob). In the data publishing phase, the data publisher releases the collected data to a data miner or to the public, called the data recipient, who will then conduct data mining on the published data. In this survey, data mining has a broad sense, not necessarily restricted to pattern mining or model building. For example, a hospital collects data from patients and publishes the patient records to

an external medical center. In this example, the hospital is the data publisher, patients are record owners, and the ACM Computing Surveys medical center is the data recipient. The data mining conducted at the medical center could be anything from a simple count of the number of men with diabetes to a sophisticated cluster analysis. There are two models of data publishers. In the untrusted model, the data publisher is not trusted and may attempt to identify sensitive information from record owners. Various cryptographic solutions; anonymous communications; and statistical methods were proposed to collect records anonymously from their owners without revealing the owners' identity. In the trusted model, the data publisher is trustworthy and record owners are willing to provide their personal information to the data publisher; however, the trust is not transitive to the data recipient. In this ,we assume the trusted model of data publishers and consider privacy issues in the data publishing phase.



**Figure 1.2 Data collection and Data Publishing.**

### 1.3 Privacy Models

We can broadly classify privacy models into two categories based on their attack principles. The first category considers that a privacy threat occurs when an attacker is able to link a record owner to a record in a published data table, to a sensitive attribute in a published data table, or to the published data table itself. We call these record linkage, attribute linkage, and table linkage, respectively.

- Record linkage: attacker aims at linking a row to a person
- Attribute linkage: attacker wants to link (sensitive) attribute to person
- Table linkage: attacker wants to know whether data of a specific person is present in a table

The second category aims at achieving the uninformative principle: The published table should provide the attacker with little additional information beyond the background knowledge. If the attacker has a large variation between the prior and posterior beliefs, we call it the probabilistic attack.

### 1.4 Purpose & Objective

Privacy Preserving publishing of microdata has been studied extensively in recent years. Microdata contains records each of which contains information about an individual entity, such as a person, a household, or an organization. Several microdata anonymization techniques have been proposed. The most popular ones are generalization for k-anonymity and bucketization for l-diversity . In both approaches, attributes are partitioned into three categories: 1) some attributes are identifiers that can uniquely identify an individual, such as Name or Social Security Number; 2) some attributes are Quasi Identifiers (QI), which the adversary may already know (possibly from other publicly available databases) and which, when taken together, can potentially identify an individual, e.g., Birthdate, Sex, and Zipcode; 3) some attributes are Sensitive Attributes (SAs), which are unknown to the adversary and are considered sensitive, such as Disease and Salary.

We intend to develop a new anonymization tool defined from a technique called "Slicing" for privacy preserving publication, which is applicable to any monotonic anonymization principle. Slicing subsumes traditional generalization as a special case. It produces an anonymized relation that achieves the same privacy guarantee (as conventional generalization), but permits a much more accurate reconstruction of the original data distribution. Furthermore, Slicing offers a simple and rigorous solution to anonymized marginal publication, which was previously a difficult issue with conventional generalization. SLICING APPROACH utilizes existing algorithms of simple generalization to perform Slicing. It would be interesting to study whether it is possible to obtain better Slicing directly, without resorting to simple generalization. Previously only static microdata is considered for Slicing, but now we have extended it to real time dynamic data as well as support for republication with adhered changes from Slicing.

Slicing overcomes the limitations of generalization and bucketization and preserves better utility while protecting against privacy threats. We illustrate how to use slicing to prevent attribute disclosure and membership disclosure. Our experiments show that slicing preserves better data utility than generalization and is more effective than bucketization in workloads involving the sensitive attribute.

The general methodology proposed by this work is that: before anonymizing the data, one can analyze the data characteristics and use these characteristics in data anonymization.

The rationale is that one can design better data anonymization techniques when we know the data better. Attribute correlations can be used for privacy attacks and slicing succeeds in maintaining a balance to prevent such attacks yet maintaining data utility.

## 1.5 Anonymization Techniques

There are mainly three types of anonymization techniques

- Generalization
- Bucketization
- Slicing

In both generalization and bucketization, one first removes identifiers from the data and then partitions tuples into buckets. The two techniques differ in the next step. Generalization transforms the QI-values in each bucket into “less specific but semantically consistent” values so that tuples in the same bucket cannot be distinguished by their QI values. In bucketization, one separates the SAs from the QIs by randomly permuting the SA values in each bucket. The anonymized data consists of a set of buckets with permuted sensitive attribute values.

## 1.6 Generalization

A common way for k-anonymisation is achieved by generalisation — an attribute value is generalised according to its attribute domain hierarchy. For example, date of birth D/M/Y is replaced by M/Y. All attribute domains are in hierarchical structures. Domains with fewer values are more general than domains with more values for an attribute. The most general domain contains only one value. For example, date of birth in D/M/Y is a lower level domain, and date of birth in Y is a higher level domain. The most general level of date of birth domain contains value unknown ‘\*’. Numerical attributes are in a hierarchical structure too. That is {value, interval, \*}. Intervals can be determined by users or a machine learning algorithm, say, a discretisation method. As illustrated in Figure 1, 10 year interval level in birth domain is more general than year level.

For example, Table 1.6.1 shows a simplified medical data set. It does not contain personal identification attributes, such as name, address, and medical care card number. However, the unique combinations of gender, age and postcode still reveal sensitive information of individuals. For instance, the first record is unique

in these three attributes, and the patient is potentially identifiable. As a result, depression condition of the patient may be revealed by the unique combination.

	Quasi identifier			Other attributes	Sensitive attributes
	Gender	Age	Postcode		Diagnosis
1	male	25	4350	... ..	Depression
2	male	22	4352	... ..	Flu
3	male	28	4353	... ..	Flu
4	female	34	4352	... ..	Depression
5	female	38	4350	... ..	alcohol addiction
6	Male	42	4351	... ..	alcohol addiction
7	male	42	4350	... ..	alcohol addiction
8	male	45	4351	... ..	alcohol addiction
9	male	45	4350	... ..	alcohol addiction

**Table 1.6.1 Raw Medical Data Set**

To avoid privacy breaching, Table 1.6.1 can be modified to Table 1.6.2. In Table 1.6.2, age is grouped into intervals, and postcodes are clustered into large areas. Symbol ‘\*’ denotes any digit. A record in the quasi-identifier is identical to at least 3 other records, and therefore, no individual is identifiable.

	Quasi identifier			Other attributes	Sensitive attributes
	Gender	Age	Postcode		Diagnosis
1	Male	20-29	435*	... ..	Depression
2	Male	20-29	435*	... ..	Flu
3	Male	20-29	435*	... ..	Flu
4	Female	30-39	435*	... ..	Depression
5	Female	30-39	435*	... ..	alcohol addiction
6	Female	30-39	435*	... ..	alcohol addiction
7	Male	40-49	435*	... ..	alcohol addiction
8	Male	40-49	435*	... ..	alcohol addiction
9	Male	40-49	435*	... ..	alcohol addiction

**Table 1.6.2 Generalized Data Set**

Generalization may reveal sensitive information under the following two types of attacks. Homogeneity attack to a k-anonymity table: Bob and Tom are two hostile neighbours. Bob knows that Tom goes to hospital recently and tries to find out the disease Tom suffers. Bob finds the 4-anonymous table as in Table 1.6.2. He knows that Tom is 42 years’ old and lives in the suburb with postcode 4350. Tom



must be record 2, 10, 11, or 12. All four patients are alcohol addiction sufferers. Bob knows for sure that Tom suffers alcohol addiction. Therefore, homogeneous values in the sensitive attribute of a  $k$ -anonymous group leak private information. Background knowledge attack to a  $k$ -anonymity table: Bob and Alice are friends and Bob does not want Alice to know his medical condition. Alice knows Bob goes to hospital, but does not know what the medical problem is. She finds the 4-anonymous table containing Bob's record. Bob is 25 years old and lives in suburb with postcode 4352. Bob's record must be record 1, 2, 3 or 4. Based on the table, Alice does not know whether Bob suffers depression or flu. However, she knows Bob did not have a flu for a long time. So, Alice knows nearly for sure that Bob suffers depression. Therefore,  $k$ -anonymity does not protect individuals from a background knowledge attack.

## 1.7 Bucketization

In bucketization, we partition the tuples in table into buckets, and then to separate the sensitive attribute from the non-sensitive ones by randomly permuting the sensitive attribute values within each bucket. The sanitized data then consists of the buckets with permuted sensitive values. we use bucketization as the method of constructing the published data from the original table  $T$ , although all our results hold for full-domain generalization as well. We now specify our notion of bucketization more formally. Partition the tuples into buckets (i.e., horizontally partition the table  $T$  according to some scheme), and within each bucket, we apply an independent random permutation to the column containing  $S$ -values. The resulting set of buckets, denoted by  $B$ , is then published.

While bucketization has better data utility than generalization, it has several limitations. First, bucketization does not prevent membership disclosure. Because bucketization publishes the  $QI$  values in their original forms, an adversary can find out whether an individual has a record in the published data or not. As shown in, 87 percent of the individuals in the United States can be uniquely identified using only three attributes (Birthdate, Sex, and Zipcode). A microdata (e.g., census data) usually contains many other attributes besides those three attributes.

This means that the membership information of most individuals can be inferred from the bucketized table.

Second, bucketization requires a clear separation between QIs and SAs. However, in many data sets, it is unclear which attributes are QIs and which are SAs. Third, by separating the sensitive attribute from the QI attributes, bucketization breaks the attribute correlations between the QIs and the SAs.

Table 1.7 shows the bucketization where sensitive attributes are separated and randomly shuffled.

	Quasi identifier			Other attributes	Sensitive attributes
	Gender	Age	Postcode		Diagnosis
1	Male	25	4350	... ..	Flu
2	Male	22	4352	... ..	Depression
3	Male	28	4353	... ..	Depression
4	Female	34	4352	... ..	Alcohol addiction
5	Female	38	4350	... ..	Flu
6	female	35	4350	... ..	alcohol addiction
7	male	42	4351	... ..	Flu
8	male	45	4351	... ..	Depression
9	male	45	4350	... ..	alcohol addiction

**Table 1.7 Bucketized Data set**

## 1.8 Slicing

To overcome the problems with existing techniques, we introduce a novel data anonymization technique called slicing. Slicing partitions the data set both vertically and horizontally. Vertical partitioning is done by grouping attributes into columns based on the correlations among the attributes. Each column contains a subset of attributes that are highly correlated. Horizontal partitioning is done by grouping tuples into buckets. Finally, within each bucket, values in each column are randomly permuted (or sorted) to break the linking between different columns. The basic idea of slicing is to break the association cross columns, but to

preserve the association within each column. This reduces the dimensionality of the data and preserves better utility than generalization and bucketization.

Slicing preserves utility because it groups highly correlated attributes together, and preserves the correlations between such attributes. Slicing protects privacy because it breaks the associations between uncorrelated attributes, which are infrequent and thus identifying. Note that when the data set contains QIs and one SA, bucketization has to break their correlation; slicing, on the other hand, can group some QI attributes with the SA, preserving attribute correlations with the sensitive attribute.

The key intuition that slicing provides privacy protection is that the slicing process ensures that for any tuple, there are generally multiple matching buckets. Slicing first partitions attributes into columns. Each column contains a subset of attributes. Slicing also partition tuples into buckets. Each bucket contains a subset of tuples. This horizontally partitions the table. Within each bucket, values in each column are randomly permuted to break the linking between different columns. Table 1.8 shows how the correlated attributes are grouped together and how the associations cross columns are broken and associations within column are preserved.

(Age,Sex)	(Zipcode,Disease)
(22,M)	(47905,flu)
(22,F)	(47906,dysp.)
(33,F)	(47905,bron.)
(52,F)	(47906,flu)
(54,M)	(47304,gast.)
(60,M)	(47302,flu)
(60,M)	(47302,dysp.)
(64,F)	(47304,dysp.)

**Table 1.8 Sliced Data Set**

# **Chapter 2**

# **LITERATURE REVIEW**

## 2. LITERATURE REVIEW

### 2.1 Motivation

The collection of digital information by governments, corporations, and individuals has created tremendous opportunities for knowledge-based decision making. Driven by mutual benefits, or by regulations that require certain data to be published, there is a demand for the exchange and publication of data among various parties. For example, licensed hospitals in California are required to submit specific demographic data on every patient discharged from their facility. In June 2004, the Information Technology Advisory Committee released a report entitled *Revolutionizing Health Care Through Information Technology*. One of its key points was to establish a nationwide system of electronic medical records that encourages sharing of medical knowledge through computer-assisted clinical decision support. Data publishing is equally ubiquitous in other domains. For example, Netflix, a popular online movie rental service, recently published a data set containing movie ratings of 500,000 subscribers, in a drive to improve the accuracy of movie recommendations based on personal preferences; AOL published a release of query logs but quickly removed it due to the reidentification of a searcher.

Detailed person-specific data in its original form often contains sensitive information about individuals, and publishing such data immediately violates individual privacy. The current practice primarily relies on policies and guidelines to restrict the types of publishable data and on agreements on the use and storage of sensitive data. The limitation of this approach is that it either distorts data excessively or requires a trust level that is impractically high in many data-sharing scenarios. For example, contracts and agreements cannot guarantee that sensitive data will not be carelessly misplaced and end up in the wrong hands.

A task of the utmost importance is to develop methods and tools for publishing data in a more hostile environment, so that the published data remains practically

useful while individual privacy is preserved. This undertaking is called privacy-preserving data publishing (PPDP). In the past few years, research communities have responded to this challenge and proposed many approaches. While the research field is still rapidly developing, it is a good time to discuss the assumptions and desirable properties for PPDP, clarify the differences and requirements that distinguish PPDP from other related problems, and systematically summarize and evaluate different approaches to PPDP.

## 2.2 Models of Data Publishing

There are two models of data publishers.

In the untrusted model, the data publisher is not trusted and may attempt to identify sensitive information from record owners. Various cryptographic solutions; anonymous communications and statistical methods were proposed to collect records anonymously from their owners without revealing the owner's identity.

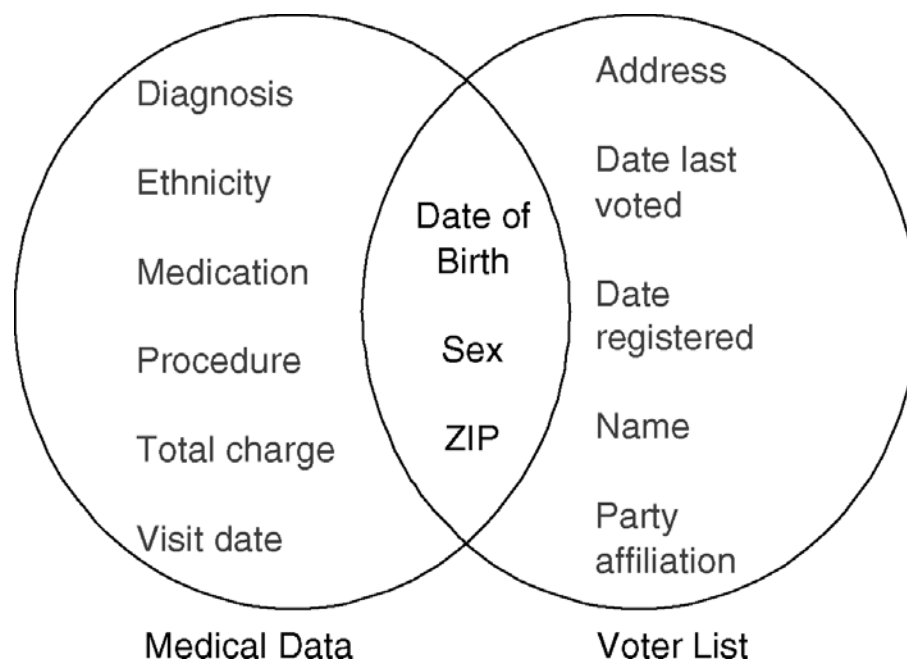
In the trusted model, the data publisher is trustworthy and record owners are willing to provide their personal information to the data publisher; however, the trust is not transitive to the data recipient. In this survey, we assume the trusted model of data publishers and consider privacy issues in the data publishing phase.

## 2.3 The Anonymization Approach

In the most basic form of PPDP, the data publisher has a table of the form  $D(\text{Explicit Identifier, Quasi Identifier, Sensitive Attributes, Non-Sensitive Attributes})$ , where Explicit Identifier is a set of attributes, such as name and social security number (SSN), containing information that explicitly identifies record owners; Quasi Identifier (QID) is a set of attributes that could potentially identify record owners; Sensitive Attributes consists of sensitive person-specific information such as disease, salary, and disability status; and Non-Sensitive Attributes contains all attributes that do not fall into the previous three categories .

The four sets of attributes are disjoint. Most works assume that each record in the table represents a distinct record owner.

Anonymization refers to the PPDP approach that seeks to hide the identity and/or the sensitive data of record owners, assuming that sensitive data must be retained for data analysis. Clearly, explicit identifiers of record owners must be removed. Even with all explicit identifiers being removed, Sweeney showed a real-life privacy threat to William Weld, former governor of the state of Massachusetts. In Sweeney's example shown in Figure 2.3, an individual's name in a public voter list was linked with his record in a published medical database through the combination of zip code, date of birth, and sex, as shown in Figure 2.3. Each of these attributes does not uniquely identify a record owner, but their combination, called the quasi identifier, often singles out a unique or a small number of record owners. According to Sweeney, 87% of the U.S. population had reported characteristics that likely made them unique based on only such quasi-identifiers.



**Figure 2.3 Record Linkage [Sweeney's example]**

In the above sweeney's example, the owner of a record is reidentified by linking his quasi identifier. To perform such linking attacks, the attacker needs two pieces

of prior knowledge: the victim's record in the released data and the quasi-identifier of the victim. Such knowledge can be obtained by observation. For example, the attacker noticed that his boss was hospitalized, and therefore knew that his boss's medical record would appear in the released patient database. Also, it was not difficult for the attacker to obtain his boss's zip code, date of birth, and sex, which could serve as the quasi-identifier in linking attacks. To prevent linking attacks, the data publisher provides an anonymous table,  $T(\text{QID}, \text{Sensitive Attributes}, \text{Non-Sensitive Attributes})$ , QID is an anonymous version of the original QID obtained by applying anonymization operations to the attributes in QID in the original table  $D$ . Anonymization operations hide some detailed information so that several records become indistinguishable with respect to QID. Consequently, if a person is linked to a record through QID, that person is also linked to all other records that have the same value for QID, making the linking ambiguous. Alternatively, anonymization operations could generate synthetic data table  $T$  based on the statistical properties of the original table  $D$ , or add noise to the original table  $D$ . The anonymization problem is to produce an anonymous  $T$  that satisfies a given privacy requirement determined by the chosen privacy model and to retain as much data utility as possible. An information metric is used to measure the utility of an anonymous table. Note that the Non-Sensitive Attributes are published if they are important to the data mining task.

## 2.4 Privacy Models

We can broadly classify privacy models into two categories based on their attack principles.

The first category considers that a privacy threat occurs when an attacker is able to link a record owner to a record in a published data table, to a sensitive attribute in a published data table, or to the published data table itself. We call these record linkage, attribute linkage, and table linkage, respectively. In all three types of linkages, we assume that the attacker knows the QID of the victim. In record and attribute linkages, we further assume that the attacker knows that the victim's record is in the released table, and seeks to identify the victim's record and/or



sensitive information from the table. In table linkage, the attack seeks to determine the presence or absence of the victim's record in the released table. A data table is considered to be privacy preserving if it can effectively prevent the attacker from successfully performing these linkages.

The second category aims at achieving the uninformative principle; The published table should provide the attacker with little additional information beyond the background knowledge. If the attacker has a large variation between the prior and posterior beliefs, we call it the probabilistic attack. Many privacy models in this family do not explicitly classify attributes in a data table into QID and Sensitive Attributes, but some of them could also thwart the sensitive linkages in the first category, so the two categories overlap.

## 2.5 Record Linkage

In the attack of record linkage, some value  $qid$  on Quasi Identifiers identifies a small number of records in the released table  $T$ , called a group. If the victim's Quasi Identifier matches the value  $qid$ , the victim is vulnerable to being linked to the small number of records in the group. In this case, the attacker faces only a small number of possibilities for the victim's record, and with the help of additional knowledge, there is a chance that the attacker could uniquely identify the victim's record from the group. For example, Suppose that a hospital wants to publish the patient records in Table 2.5 (a) to a research center. Suppose that the research center has access to the external table 2.5(b) and knows that every person with a record in Table 2.5(b) has a record in Table 2.5(a). Joining the two tables on the common attributes Job, Sex, and Age may link the identity of a person to his/her Disease. For example, Doug, a male lawyer who is 38 years old, is identified as an HIV patient by  $qid = (\text{Lawyer, Male, 38})$  after the join.

(a) Patient table				(b) External table			
Job	Sex	Age	Disease	Name	Job	Sex	Age
Engineer	Male	35	Hepatitis	Alice	Writer	Female	30
Engineer	Male	38	Hepatitis	Bob	Engineer	Male	35
Lawyer	Male	38	HIV	Cathy	Writer	Female	30
Writer	Female	30	Flu	Doug	Lawyer	Male	38
Writer	Female	30	HIV	Emily	Dancer	Female	30
Dancer	Female	30	HIV	Fred	Engineer	Male	38
Dancer	Female	30	HIV	Gladys	Dancer	Female	30
				Henry	Lawyer	Male	39
				Irene	Dancer	Female	32

(c) 3-anonymous patient table			
Job	Sex	Age	Disease
Professional	Male	[35-40)	Hepatitis
Professional	Male	[35-40)	Hepatitis
Professional	Male	[35-40)	HIV
Artist	Female	[30-35)	Flu
Artist	Female	[30-35)	HIV
Artist	Female	[30-35)	HIV
Artist	Female	[30-35)	HIV

**Table 2.5 Examples Illustrating Various Attacks**

In k-Anonymity, To prevent record linkage through QID, Sweeney proposed the notion of k-anonymity: if one record in the table has some value  $qid$ , at least  $k-1$  other records also have the value  $qid$ . In other words, the minimum group size on QID is at least  $k$ . A table satisfying this requirement is called k-anonymous Table 2.5(c) is a 3-anonymous table. In a k-anonymous table, each record is indistinguishable from at least  $k-1$  other records with respect to QID. Consequently, the probability of linking a victim to a specific record through QID is at most  $1/k$ .

## 2.6 Attribute Linkage

In the attack of attribute linkage, the attacker may not precisely identify the record of the target victim, but could infer his/her sensitive values from the published data  $T$ , based on the set of sensitive values associated to the group that the victim belongs to. In case some sensitive values predominate in a group, a successful inference becomes relatively easy even if k-anonymity is satisfied. Clifton [2000] suggested eliminating attribute linkages by limiting the released data size. Limiting data size may not be desirable if data records such as HIV patient data, are valuable and are difficult to obtain. Several other approaches have been

proposed to address this type of threat. The general idea is to diminish the correlation between QID attributes and sensitive attributes.

For example From Table 2.5(a), an attacker can infer that all female dancers at age 30 have HIV, i.e., (Dancer, Female, 30)  $\rightarrow$  HIV with 100% confidence. Applying this knowledge to Table 2.1(b), the attacker can infer that Emily has HIV with 100% confidence provided that Emily comes from the same population in Table 2.5(a).

Machanavajjhala proposed the diversity principle, called  $\ell$ -diversity, to prevent attribute linkage. The  $\ell$ -diversity requires every qid group to contain at least  $\ell$  “well-represented” sensitive values. There are several instantiations of this principle, which differ in the definition of being well-represented. The simplest understanding of “well-represented” is to ensure that there are at least  $\ell$ -distinct values for the sensitive attribute in each qid group. This distinct  $\ell$ -diversity privacy model (also known as p-sensitive k-anonymity) automatically satisfies k-anonymity, where  $k = \ell$ , because each qid group contains at least  $\ell$ -records.

Distinct  $\ell$ -diversity cannot prevent probabilistic inference attacks because some sensitive values are naturally more frequent than others in a group, enabling an attacker to conclude that a record in the group is very likely to have those values. For example, Flu is more common than HIV. This motivates the following two stronger notions of  $\ell$ -diversity.

One limitation of entropy  $\ell$ -diversity is that it does not provide a probability based risk measure, which tends to be more intuitive to the human data publisher. For example, in Table 2.1(c), being entropy 1.8-diverse does not convey the risk level that the attacker has 75% probability of success to infer HIV where 3 out of the 4 record owners in the qid group have HIV. Also, it is difficult to specify different protection levels based on varied sensitivity and frequency of sensitive values.

## 2.7 Table Linkage

Both record linkage and attribute linkage assume that the attacker already knows the victim's record is in the released table  $T$ . However, in some cases, the presence (or the absence) of the victim's record in  $T$  already reveals the victim's sensitive information. Suppose a hospital releases a data table with a particular type of disease. Identifying the presence of the victim's record in the table is already damaging. A table linkage occurs if an attacker can confidently infer the presence or the absence of the victim's record in the released table. The following example illustrates the privacy threat of a table linkage.

## 2.8 Anonymization Operations

Typically, the original table does not satisfy a specified privacy requirement and the table must be modified before being published. The modification is done by applying a sequence of anonymization operations to the table. An anonymization operation comes in several flavors: generalization, suppression, anatomization, permutation, and perturbation. Generalization and suppression replace values of specific description, typically the QID attributes, with less specific description. Anatomization and permutation deassociate the correlation between QID and sensitive attributes by grouping and shuffling sensitive values in a qid group. Perturbation distorts the data by adding noise, aggregating values, swapping values, or generating synthetic data based on some statistical properties of the original data.

Each generalization or suppression operation hides some details in QID. For a categorical attribute, a specific value can be replaced with a general value according to a For a numerical attribute, exact values can be replaced with an interval that covers exact values. If a taxonomy of intervals is given, the situation is similar to categorical attributes. More often, however, no predetermined taxonomy is given for a numerical attribute. Different classes of anonymization operations have different implications on privacy protection, data utility, and search space. But they all result in a less precise but consistent representation of

the original data. A generalization replaces some values with a parent value in the taxonomy of an attribute. The reverse operation of generalization is called specialization. A suppression replaces some values with a special value, indicating that the replaced values are not disclosed. The reverse operation of suppression is called disclosure.

Anatomization does not modify the quasi-identifier or the sensitive attribute, but deassociates the relationship between the two. Precisely, the method releases the data on QID and the data on the sensitive attribute in two separate tables: a quasi-identifier table (QIT) contains the QID attributes, a sensitive table (ST) contains the sensitive attributes, and both QIT and ST have one common attribute, GroupID. All records in the same group will have the same value on GroupID in both tables, and therefore are linked to the sensitive values in the group in the exact same way. If a group has  $\ell$ -distinct sensitive values and each distinct value occurs exactly once in the group, then the probability of linking a record to a sensitive value by Group ID is  $1/\ell$ . The attribute linkage attack can be distorted by increasing  $\ell$ .

Perturbation has a long history in statistical disclosure control due to its simplicity, efficiency, and ability to preserve statistical information. The general idea is to replace the original data values with some synthetic data values, so that the statistical information computed from the perturbed data does not differ significantly from the statistical information computed from the original data. The perturbed data records do not correspond to real-world record owners, so the attacker cannot perform the sensitive linkages or recover sensitive information from the published data.

## 2.9 Privacy Vs Utility

We cannot directly compare privacy and utility, it is neither reasonable nor feasible, because privacy and utility have very different characteristics, as discussed below.

In reality, the same piece of information can have very different impacts on privacy and utility. More specifically, for different kinds of knowledge, having the adversary and the researcher learn exactly the same knowledge can be beneficial in some cases and detrimental in other cases. For example, suppose that it is learned from the published data that people living near a small town have a much higher rate of getting cancer (say, 50%) than that among the general population. Learning this piece of information can impact both privacy and utility. On the one hand, this piece of information breaches the privacy of the people in this small town. For example, when they go to purchase health/life insurance, it can adversely affect their ability of getting insurance. On the other hand, by publishing this piece of information, people can investigate the causes of the problem (e.g., find some sources of pollution) and deal with the problem (e.g., by removing the pollution sources or taking precautions). In this case, such aggregate information results in more utility gain than privacy loss as it benefits the society on the whole, even for non-participants.

Suppose that, in another case, it is learned from the published data that an individual has a 50% probability of having cancer because the individual's record belongs to an equivalence class containing two records one of which has cancer. Such specific information has no utility value to researchers but causes privacy loss. Again, the information gain by the researcher and the adversary are the same, but the utility gain and the privacy loss are very different.

The above arguments leads to the first characteristic of privacy and utility: **“specific knowledge (that about a small group of individuals) has a larger impact on privacy, while aggregate information (that about a large group of individuals) has a larger impact on utility.”** In other words, privacy loss occurs when the adversary learns more information about specific individuals from the anonymized data. But data utility increases when information about larger-size populations is learned.

Another important reason why privacy and utility cannot be directly compared is as follows. For privacy protection, it is safe to publish the data only when every record satisfies the privacy parameter (i.e., every individual has a bounded privacy loss). This implies that privacy is an individual concept in that each individual's privacy is enforced separately. This characteristic is different from utility gain. When multiple pieces of knowledge are learned from the anonymized data, data utility adds up. This implies that utility is an aggregate concept in that utility accumulates when more useful information is learned from the data. The above arguments lead to the second characteristic of privacy and utility: **“privacy is an individual concept and should be measured separately for every individual while utility is an aggregate concept and should be measured accumulatively for all useful knowledge.”** This characteristic immediately implies the following corollary on measuring privacy and utility: For privacy, the worst-case privacy loss should be measured. For utility, the aggregated utility should be measured. Hence it is possible to publish anonymized data even if for each individual, both the privacy loss and the utility gain are small, because the utility adds up.

Yet another difference between privacy and utility emerges when we consider the correctness of the information learned from the anonymized data. When the adversary learns some false information about an individual, the individual's privacy is breached even though the perception is incorrect. However, when the researcher learns some false information, data utility deteriorates because it may lead to false conclusions or even misleading public policies. In fact, some studies have overlooked this difference between privacy and utility. For example, the direct comparison methodology uses the trivially-anonymized data as the baseline for both privacy and utility. While the trivially-anonymized data is appropriate as the baseline for privacy, it is inappropriate to be used as the baseline for utility gain. This is the third characteristic of privacy and utility: **“Any information that deviates from the prior belief, false or correct, may cause privacy loss but only correct information contributes to utility.”** This characteristic implies the following corollary on measuring privacy and utility: Privacy should be measured against the trivially-anonymized data whereas utility should be measured using

the original data as the baseline. When the original data is used for measuring utility, we need to measure “utility loss”, instead of “utility gain”. An ideal (but unachievable) privacy-preserving method should result in zero privacy loss and zero utility loss.



# **Chapter 3**

# **REQUIREMENT SPECIFICATION**

## 3. REQUIREMENT SPECIFICATION

### 3.1 Introduction

The purpose of the supplementary specification is to highlight the Publishing Private Data using Slicing. Readers of this document will be the users who are going to use the new technique “Slicing” for publishing their private data.

#### 3.1.1 Purpose

This document provides a brief importance of the project and its usage in the present world .It provides the specification and the usage of the Slicing technique to publish Privacy Preserving Data and comparing with various other existing techniques.

#### 3.1.2 Scope

This document is associated with the Data Publication while preserving privacy and maximizing the Data Utility. It briefly describes the usage and functionality and also steps taken while developing the project.

#### 3.1.3 Definitions, Acronyms and Abbreviations

QI – Quasi Identifiers

SA – Sensitive Attributes

SRS – Supplementary Specification

#### 3.1.4 References

Slicing: A New Approach for Privacy Preserving Data Publishing

By *Tiancheng Li, Ninghui Li, Senior Member, IEEE, Jian Zhang, Member, IEEE, and Ian Molloy* Volume No. 24, March 2012.

Privacy-Preserving Data Publishing: By *BENJAMIN C. M. FUNG, Concordia University, Montreal.*

### **3.1.5 Overview**

The SRS first gives the overview of the Privacy Preserving Data Publishing and later describes the functionality of Slicing in detail.

## **3.2 Functionality**

### **3.2.1 Authentication**

The User logs into his account by authenticating his username and password.

### **3.2.2 Admin Login**

- Validate Users
- Delete Users
- Update Data sets
- View Original Data
- Publish Anonymized data
- Logout

### **3.2.3 Employee Login**

- Register
- View self data
- Logout

### **3.2.4 Anonymization Techniques**

- **Generalization**

When the Dataset is loaded, Some Attribute values in the table are Generalized i.e less specific but consistent values hiding the original form of data.

- **Bucketization**

When the Dataset is loaded, Tuples are partitioned into buckets and SA's are separated from QI's and randomly permuted. Thus the correlations are broken between columns.

- **Slicing**

When the Dataset is loaded, Groups the Correlated Attributes into one column and tuples are partitioned into buckets where values in each column are permuted. Thus attribute correlations are preserved and associations with each column are preserved.

### **3.3 Usability**

#### **3.3.1 User Friendly**

The Interface is very user friendly. The User can get accustomed to its functionality in its first usage.

### **3.4 Reliability**

Due to the usage of advanced efficient database, Application is Reliable.

- Availability – When the user logs in, the software is available entirely to that user and in no way shares its resources with any other user.
- Mean Time Between Failures (MTBF) – Usually around 1 hour depending upon the severity of the failure.
- Mean Time To Repair (MTTR) – System will be out of operation for maximum of 1 hour in case of failure.
- Accuracy – This is very accurate as it calculates Correlation measures accurately and group attributes accordingly.
- Bugs or Defect rate – Minor bugs will only cause a minor disruption.

### **3.5 Performance**

The performance depends upon No. of users accessing this software and size of database.

- Response time for a transaction– 1 minute
- Capacity Only one user at a time
- Degradation modes– This automatically logs out when system goes to sleep.
- Resource utilization– It uses maximum hard disk memory of 2GB.

### **3.6 Supportability**

This Application is made on windows platform. So, supports in majority of systems.

#### **3.6.1 Microsoft**

This software needs a Microsoft licensed editions.

#### **3.6.2 MySQL**

This software uses MySQL for database.

### **3.7 Design Constraints**

This system runs best on windows platform.

#### **3.7.1 Windows 7**

Performance of the software is very good in case of windows 7 and performance will be affected negligibly in case of other platforms.

### **3.8 Interfaces**

#### **3.8.1 User Interface**

We used HTML for user to interact with the system by using update, logout and also to manage the details.

### **3.8.2 Hardware Interface**

Processor – Intel Pentium-III

Speed – 1.1 GHz

RAM – 256 MB (min)

HDD – 20 GB

Keyboard – Standard Windows Keyboard

Mouse – Two or Three Button Mouse

Monitor – SVGA

Web interface: any compatible.

### **3.8.3 Software Interface**

Operating System – Windows XP /vista/7

IDE – Net Beans IDE 7.0

Application Server – Tomcat 5.0/6.x

Front End – HTML

Scripts – JavaScript

Server side Script – JSP

Database Connectivity – MySQL

# **Chapter 4**

# **SYSTEM DESIGN**

## 4. SYSTEM DESIGN

### 4.1 Process Model

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

Stages in SDLC:

- ◆ Requirement Gathering
- ◆ Analysis
- ◆ Designing
- ◆ Coding
- ◆ Testing
- ◆ Maintenance

Requirements Gathering:

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.

Analysis Stage:

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches. The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software



product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

#### Designing Stage:

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

#### Development (Coding) Stage:

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.

#### Integration & Test Stage:

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test

environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.

#### Maintenance:

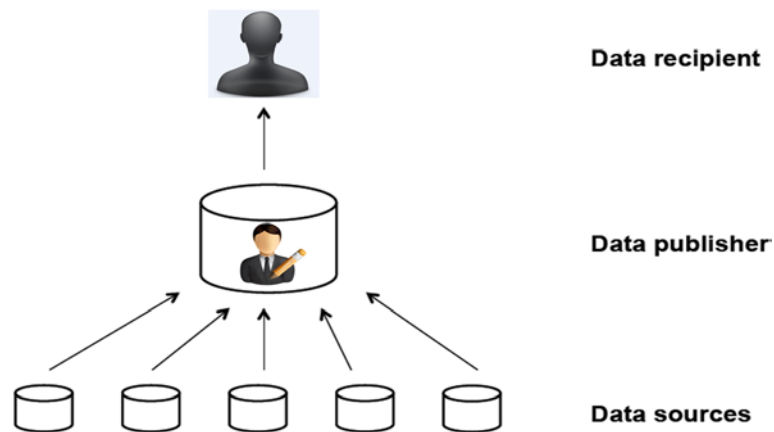
Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will under go training on that particular assigned category.

For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

## **4.2 System Architecture**

A typical architecture of the system is described in Figure 4.2. In the datacollection phase, the data publisher collects data from record owners. In the data publishing phase, the data publisher releases the collected data to a data miner or to the public, called the data recipient, who will then conduct data mining on the published data.

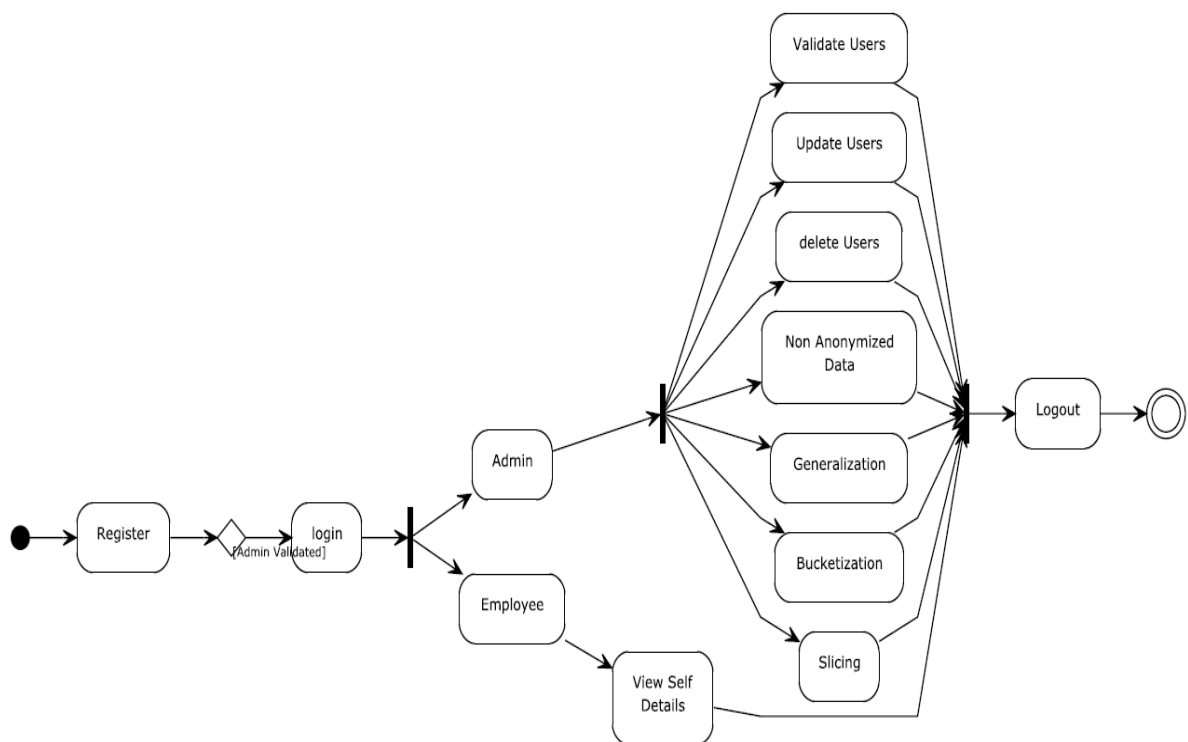
In our system, A corporate company wants to publish its corporate expenses to an external agency. Admin collects the data from all his employees and publish the data to the external research centre by applying anonymization techniques to preserve individual privacy.



**Figure 4.2 Architecture Diagram**

### 4.3 Activity Diagram

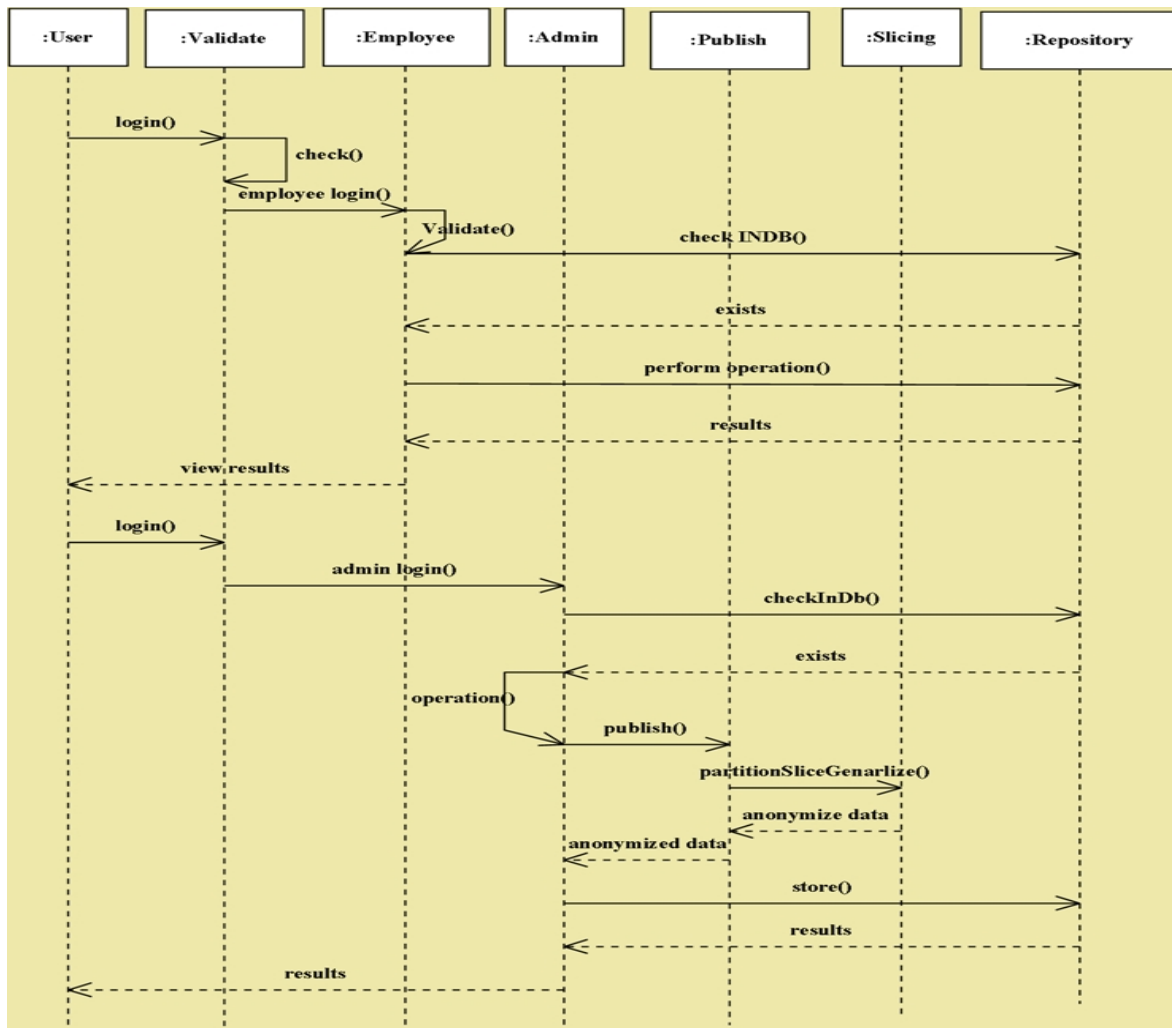
Activity Diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, Activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall control.



**Figure 4.3 Activity Diagram**

## 4.4 Sequence Diagram

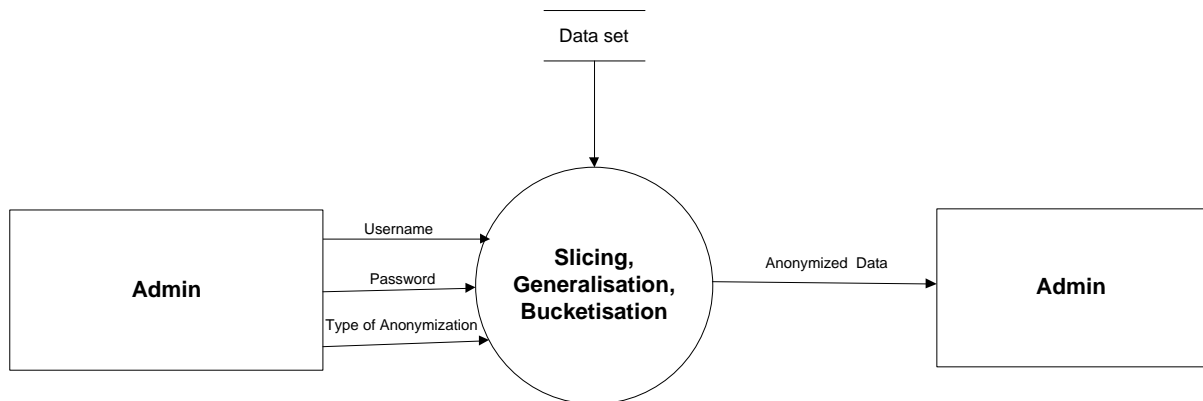
A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.



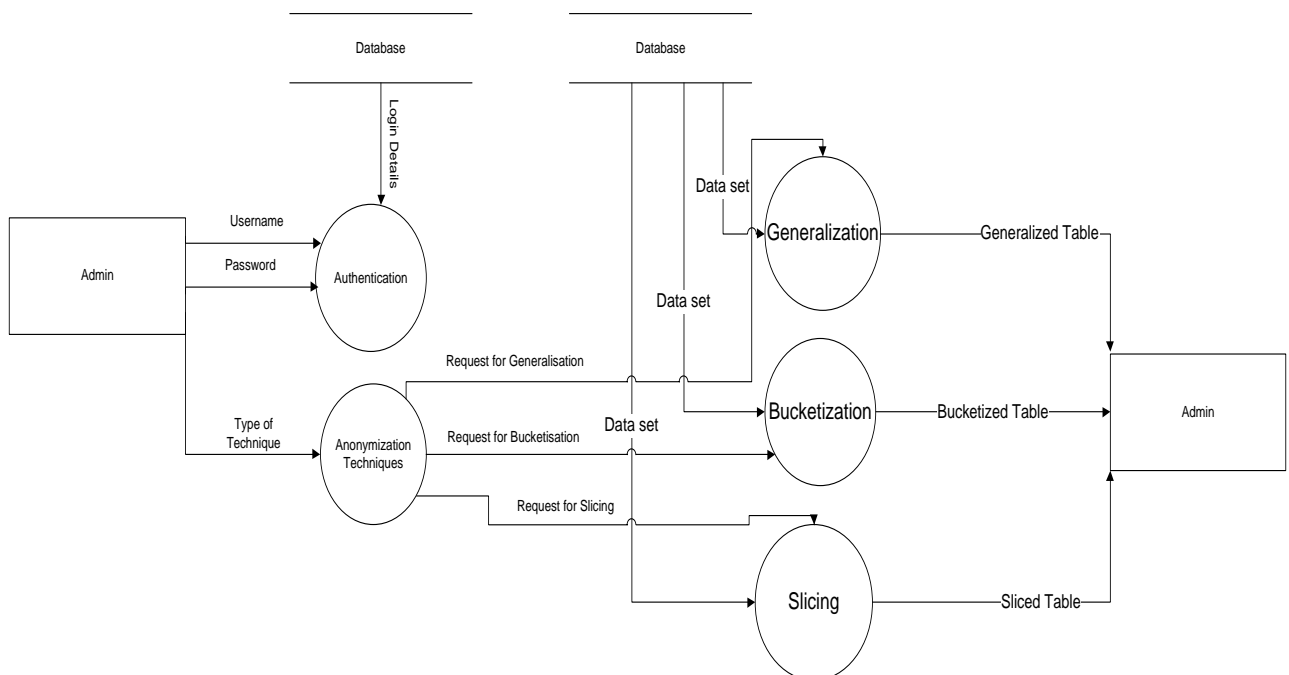
**Figure 4.4** Sequence Diagram

## 4.5 Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are preliminary step used to create an overview of the system which can later be elaborated. It shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored.



**Figure 4.5.1 Level 0 DFD**



**Figure 4.5.2 Level 1 DFD**

## 4.6 Use Case Diagram

A use case diagram is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

Use case diagram consists of

- The actors, usually individuals involved with the system defined according to their roles.
- The use cases, which are the specific roles played by the actors within and around the system.
- The relationships between and among the actors and the use cases.

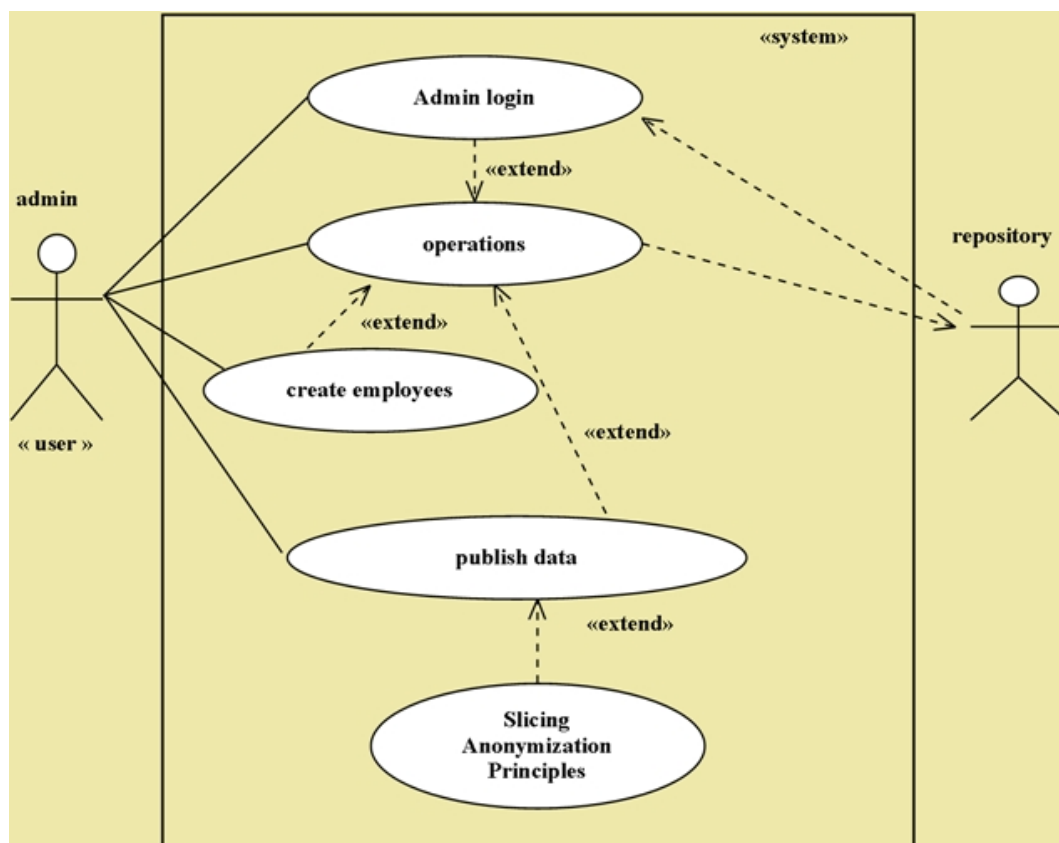
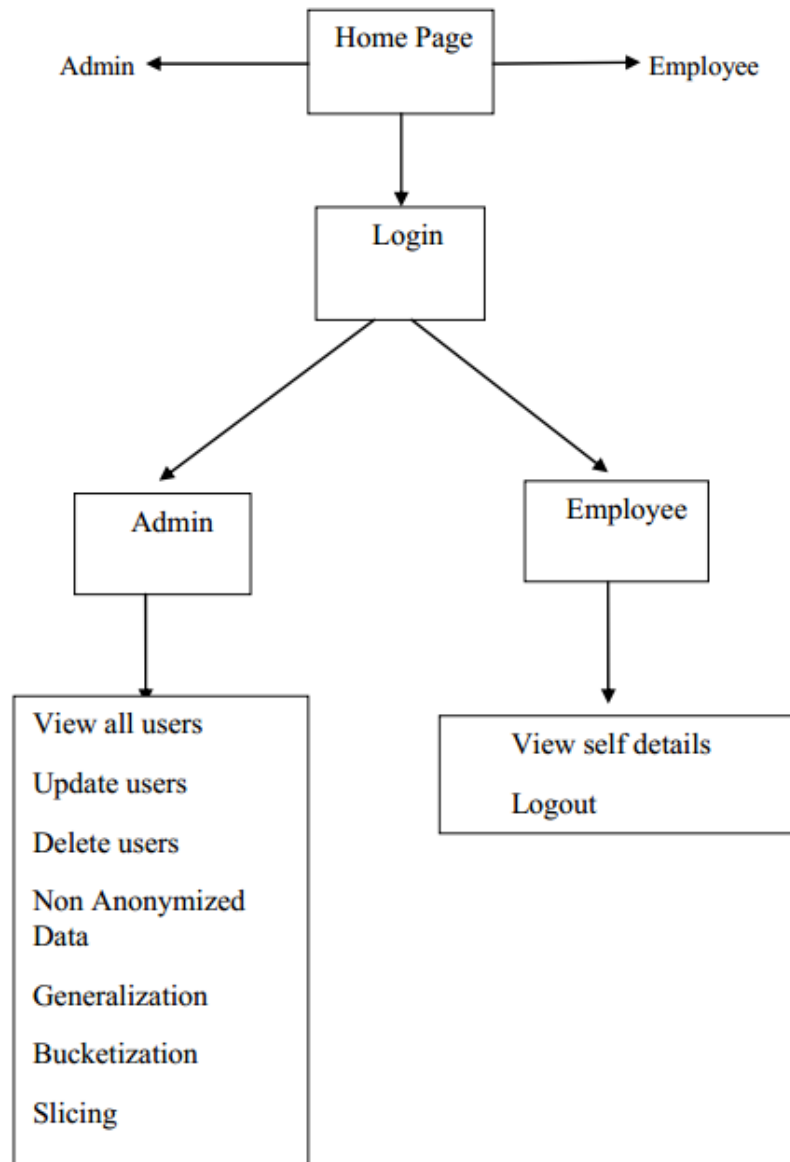


Figure 4.6 Use Case Diagram

## 4.7 Block Diagram

Block diagram gives the overall structure of the application.



**Figure 4.7 Block Diagram**

# **Chapter 5**

## **SYSTEM IMPLEMENTATION**



## **5. SYSTEM IMPLEMENTATION**

### **5.1 Implementation Details**

Coding of this project is done using java, JSP, MySQL and HTML. Coding process of the project can be divided mainly into two: Coding for GUI and Coding for Database connectivity and Anonymization Techniques.

Our Scenario is a corporate company wants to publish its corporate expenses to an external research agency where the data contains individual person specific data of all employees which if published in original form breaches privacy. So, we have to apply anonymization techniques on Quasi Identifiers i.e. salary, zip code, Email Id, Gender. So that the adversary cannot find out the sensitive attribute i.e. Designation and salary of a person.

#### **5.1.1 Coding for GUI**

We used HTML and CSS to create Front end where the user interacts with HTML forms and buttons and Java scripts to validate the forms and display error messages accordingly.

#### **5.1.2 Coding for Anonymization Techniques**

We used JDBC and servlets for connecting to database and MySQL to extract dataset, to get records from database and to store records in database. We used JSP pages to implement java code for anonymization and to include HTML pages. We used Apache Tomcat server for connecting JSP pages.

### **5.2 Modules**

The modules implemented in this project are

- Registration
- Employee Validation & Managing Users

- Generalization
- Bucketization
- Slicing
- Employee Details

### **5.2.1 Registration**

The new employee registers with the company by registering his details like Name, User Id, Address, Designation, Email Id e.t.c. through HTML form.

A JavaScript validation is given to form for correct input of details. An alert will be displayed when incorrect input is given. When user clicks on register, we check in database whether the user Id is unique or already exists, If exists, we display an alert saying try alternate user Id. If the user Id doesn't already exist in database then A query will store the details in a database.

### **5.2.2 Employee Validation & Managing Users**

The Admin validates the employee who registers newly by making the status as 'yes', until admin validation the status will be 'no' by default and the employee cant login unless the status is 'yes'.

The Admin also manages the users by Updating their details and deleting the users. An appropriate MySQL query will update or delete the records. The Admin can view all the users present in the company and their details.

### **5.2.3 Generalization**

In generalization, we remove the identifiers and we generalize the values i.e. less specific but consistent values.

In our Scenario, we applied generalization on employee salary, email id, zip code and gender .We generalized the salary of employees into two ranges such that all employees will fall under those two ranges. We generalized the last digit of zip code such that few people have same generalized zip code. We generalized the gender by keeping ‘\*’ such a way that no individual is linked to his record.

Now the table satisfies the k-anonymity and attribute linkage and record linkage is protected.

#### **5.2.4 Bucketization**

In bucketization, we remove the identifiers and the Sensitive Attributes are clearly separated from Quasi Identifiers and are randomly permuted breaking the correlations.

In our scenario, The sensitive Attribute i.e. designation is separated from other quasi identifiers and the designations of employees are randomly shuffled. It prevents attribute linkage and record linkage but doesn't prevent Membership Disclosure.

#### **5.2.5 Slicing**

Slicing partitions the data set both vertically and horizontally. Vertical partitioning is done by grouping attributes into columns based on the correlations among the attributes. Each column contains a subset of attributes that are highly correlated. Horizontal partitioning is done by grouping tuples into buckets. Finally, within each bucket, values in each column are randomly permuted (or sorted) to break the linking between different columns. The basic idea of slicing is to break the association cross columns, but to preserve the association within each column.

In our scenario, Designation and salary are correlated attributes which we group together for better data utility and we randomly permute both salary and designation for better data privacy.

### **5.2.6 Employee Details**

After admin validation, the employee can login with his credential and view his self details like his designation, his address, his zip code e.t.c.

## **5.3 Comparison with Generalization**

It has been shown that generalization for k-anonymity losses considerable amount of information, especially for high-dimensional data. This is due to the following three reasons. First, generalization for k-anonymity suffers from the curse of dimensionality. In order for generalization to be effective, records in the same bucket must be close to each other so that generalizing the records would not lose too much information. However, in high dimensional data, most data points have similar distances with each other, forcing a great amount of generalization to satisfy k-anonymity even for relatively small k's. Second, in order to perform data analysis or data mining tasks on the generalized table, the data analyst has to make the uniform distribution assumption that every value in a generalized interval/set is equally possible, as no other distribution assumption can be justified. This significantly reduces the data utility of the generalized data. Third, because each attribute is generalized separately, correlations between different attributes are lost. In order to study attribute correlations on the generalized table, the data analyst has to assume that every possible combination of attribute values is equally possible. This is an inherent problem of generalization that prevents effective analysis of attribute correlations.

Where as slicing breaks the association cross columns, but preserves the association within each column. This reduces the dimensionality of the data and preserves better utility than generalization. Slicing preserves utility because it groups highly correlated attributes together, and preserves the correlations

between such attributes. Slicing protects privacy because it breaks the associations between uncorrelated attributes, which are infrequent and thus identifying. Note that when the data set contains QIs and one SA, bucketization has to break their correlation; slicing, on the other hand, can group some QI attributes with the SA, preserving attribute correlations with the sensitive attribute.

Age	Sex	Zipcode	Disease
[20-52]	*	4790*	dyspepsia
[20-52]	*	4790*	flu
[20-52]	*	4790*	flu
[20-52]	*	4790*	bronchitis
[54-64]	*	4730*	flu
[54-64]	*	4730*	dyspepsia
[54-64]	*	4730*	dyspepsia
[54-64]	*	4730*	gastritis

**Table 5.3      Generalization**

## 5.4 Comparison with Bucketization

To compare slicing with bucketization, we first note that bucketization can be viewed as a special case of slicing, where there are exactly two columns: one column contains only the SA, and the other contains all the QIs. The advantages of slicing over bucketization can be understood as follows: First, by partitioning attributes into more than two columns, slicing can be used to prevent membership disclosure. Our empirical evaluation on a real data set shows that bucketization does not prevent membership disclosure.

Second, unlike bucketization, which requires a clear separation of QI attributes and the sensitive attribute, slicing can be used without such a separation. For data set such as the census data, one often cannot clearly separate QIs from SAs because there is no single external public database that one can use to determine which attributes the adversary already knows. Slicing can be useful for such data. Finally, by allowing a column to contain both some QI attributes and the sensitive attribute, attribute correlations between the sensitive attribute and the QI attributes are preserved. Attribute correlations are important utility in data publishing.

Age	Sex	Zipcode	Disease
22	M	47906	flu
22	F	47906	dyspepsia
33	F	47905	bronchitis
52	F	47905	flu
54	M	47302	gastritis
60	M	47302	flu
60	M	47304	dyspepsia
64	F	47304	dyspepsia

**Table 5.4      Bucketization**

## 5.5 Attribute Disclosure Protection

In this section, we show how slicing is used to prevent attribute disclosure, based on the privacy requirement of  $\ell$ -diversity. We first give an example illustrating how slicing satisfies  $\ell$ -diversity where the sensitive attribute is “Disease.” The sliced table shown in Table 5.5 satisfies 2-diversity. Consider tuple  $t_1$  with QI values (22;M; 47906). In order to determine  $t_1$ ’s sensitive value, one has to examine  $t_1$ ’s matching buckets. By examining the first column (Age; Sex) in Table 5.5, we know that  $t_1$  must be in the first bucket B1 because there are no matches of (22;M) in bucket B2. Therefore, one can conclude that  $t_1$  cannot be in bucket B2 and  $t_1$  must be in bucket B1. Then, by examining the Zip code attribute of the second column (Zipcode; Disease) in bucket B1, we know that the column value for  $t_1$  must be either (47906; dyspepsia) or (47906; flu) because they are the only values that match  $t_1$ ’s zip code 47906. Note that the other two column values have zip code 47905. Without additional knowledge, both dyspepsia and flu are equally possible to be the sensitive value of  $t_1$ . Therefore, the probability of learning the correct sensitive value of  $t_1$  is bounded by 0.5. Similarly, we can verify that 2-diversity is satisfied for all other tuples in Table 5.5.

(Age,Sex)	(Zipcode,Disease)
(22,M)	(47905,flu)
(22,F)	(47906,dysp.)
(33,F)	(47905,bron.)
(52,F)	(47906,flu)
(54,M)	(47304,gast.)
(60,M)	(47302,flu)
(60,M)	(47302,dysp.)
(64,F)	(47304,dysp.)

**Table 5.5      Slicing**

## 5.6 Membership Disclosure Protection

In this section, we analyze how slicing can provide membership disclosure protection. Let us first examine how an adversary can infer membership information from bucketization. Because bucketization releases each tuple's combination of QI values in their original form and most individuals can be uniquely identified using the QI values, the adversary can determine the membership of an individual in the original data by examining whether the individual's combination of QI values occurs in the released data.

Slicing offers protection against membership disclosure because QI attributes are partitioned into different columns and correlations among different columns within each bucket are broken. Consider the sliced table in Table 5.5. The table has two columns. The first bucket is resulted from four tuples; we call them the original tuples. The bucket matches altogether  $4^2 = 16$  tuples, 4 original tuples and 12 that do not appear in the original table. We call these 12 tuples as fake tuples. Given any tuple, if it has no matching bucket in the sliced table, then we know for sure that the tuple is not in the original table. However, even if a tuple has one or more matching bucket, one cannot tell whether the tuple is in the original table, because it could be a fake tuple.

# **Chapter 6**

# **TESTING**



## 6. TESTING

### 6.1 Introduction

Software Testing is the process of executing a program or system with the intent of finding errors. It also involves any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. Any errors created within the system development life cycle can lead to many faults within the software, which will eventually lead to failure of the software. Software bugs will almost always exist in any software module with moderate size: not because programmers are careless or irresponsible, but because the complexity of software is generally intractable the more faults or defects that are detected and corrected the more reliable the system will be. Since the project includes mainly software components the testing is done mainly for the software components. However, basic hardware testing is done as well.

### 6.2 Hardware Testing

The hardware used here is an android mobile phone with a speaker and a mic integrated with the system.

- **Testing the component:** The component required for the application is tested.

### 6.3 Software Testing

Software testing is a process used to assess the quality of computer software. Software testing is an empirical technical investigation conducted to provide stakeholders with information about the quality of the product or service under the test, with respect to the context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding software bugs. As computers and software are used in critical applications, the outcome of a bug can be severe. Bugs can cause huge losses.

Software testing involves an activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. Once the source code has been generated, software must be tested to uncover as many errors as possible before delivery to the customer.

The software faults occur through the following processes: A programmer makes an error (mistake), which results in a defect (fault, bug) in the software source code. If this defective code is executed then in certain situations the system will produce wrong results, causing failure. But note that not all defects will result in failures.

For example, defects in dead code will not result in failures. A defect can turn into a failure when the environment is changed. Examples of these changes in environment include software being used in a new hardware platform, alteration in source data or interaction with different software.

There are many approaches to software testing. Reviews, walk through or inspection are considered as static testing, whereas actually running the program within a given set of test cases in a given deployment stage is referred to as dynamic testing.

Software testing is used in association with

- Verification
- Validation

Verification checks if the result agrees with the specification and ensures the product is designed to deliver all functionality to the customer. Validation ensures that functionality, as defined in requirements, is the intended behavior of the product. Validation typically involves actual testing and takes place after verifications are completed.

## 6.4 Testing Methods

Software testing methods are traditionally divided into Black box testing and White box testing. The two approaches are used to describe the point of view that an engineer takes when designing test cases.

- **Black box testing** treats the software as a black-box without any understanding of the internal behavior. The black-box approach is a testing method in which test data are derived from the specified functional requirements without regard to the final program structure. Only the functionality of the software module is of concern. Black-box testing also mainly refers to functional testing, a testing method emphasized on executing the functions and examination of their input and output data. Thus, the tester inputs the data and only sees the output from the test object. This level of testing usually requires thorough test cases to be provided to the tester who then can simply verify that for a given input, the output value is the same as expected value specified in the test case. Black box testing methods include: equivalence partitioning, traceability matrix, boundary value analysis, all-pairs testing, fuzz testing, model based testing etc.
- **White box testing**, is when the tester has access to the internal data structures, code, algorithms. Software is viewed as a white-box, or glass-box in white-box testing, as the structure and flow of the software under test are visible to the tester. White box testing methods includes creating tests to satisfy some code coverage criteria. For example, the test designer can create tests to cause all statements in the program to be executed at least once. Other examples of white box testing are mutation testing and fault injection methods.

## 6.5 Testing Process

A common practice of software testing is performed by an independent group of testers after the functionality is developed before it is shipped to the customer.

This practice often results in the testing phase being used as project buffer to compensate for project delays,, thereby compromising the time devoted to testing. Another practice is to start software testing at the same moment the project starts and it is a continuous process until the project finishes.

Testing can be done in the following levels:

- Unit testing tests the minimal software components or module.
- Integration testing exposes defects in the interfaces and interaction between integrated components.
- System testing tests are completely integrated systems to verify that it meets its requirements.

### **6.5.1 Unit Testing**

In computer programming, unit testing is a procedure used to validate that individual units of source code are working properly. Unit testing extensively makes use of white box testing techniques to ensure complete coverage and maximum error detection. Unit testing deals with testing a unit as a whole. A unit is the smallest testable part of an application. This would test the interaction of many functions but confine the test within one unit. While in object oriented programming, the smallest unit is a method, which may belong to a base/super class, abstract class or derived/ child class. The exact scope of a unit is left to interpretation.

### **6.5.2 Integration Testing**

Integration testing is the phase of software testing in which individual software modules are combined and tested as a group. It follows unit testing and precedes system testing.

Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integration system ready for system testing.

The purpose of integration testing is to verify functional, performance and reliability requirements placed on major design items. These “design items”, i.e. groups of units, are exercised through their interfaces using black box testing, success and error cases being simulated via appropriate parameters and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test that all components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, that is unit testing.

### 6.5.3 System Testing

System testing of software and hardware is testing conducted on a complete, integrated system to evaluate the system compliance with its specified requirements. System testing falls within scope of black box testing and as such, should require no knowledge of the inner design of the code or logic.

As a rule, system testing takes, as its input, all of the “integrated” software components that have successfully passed integration testing and also software system itself integrated with an applicable hardware system(s). The purpose of the integration testing is to detect any inconsistencies between the software units that are integrated together or between any of the assemblages and the hardware. System testing is more limiting type of testing; it seeks to detect defects both within the “inter-assemblages”.

## 6.6 Test Cases

A test case in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not.

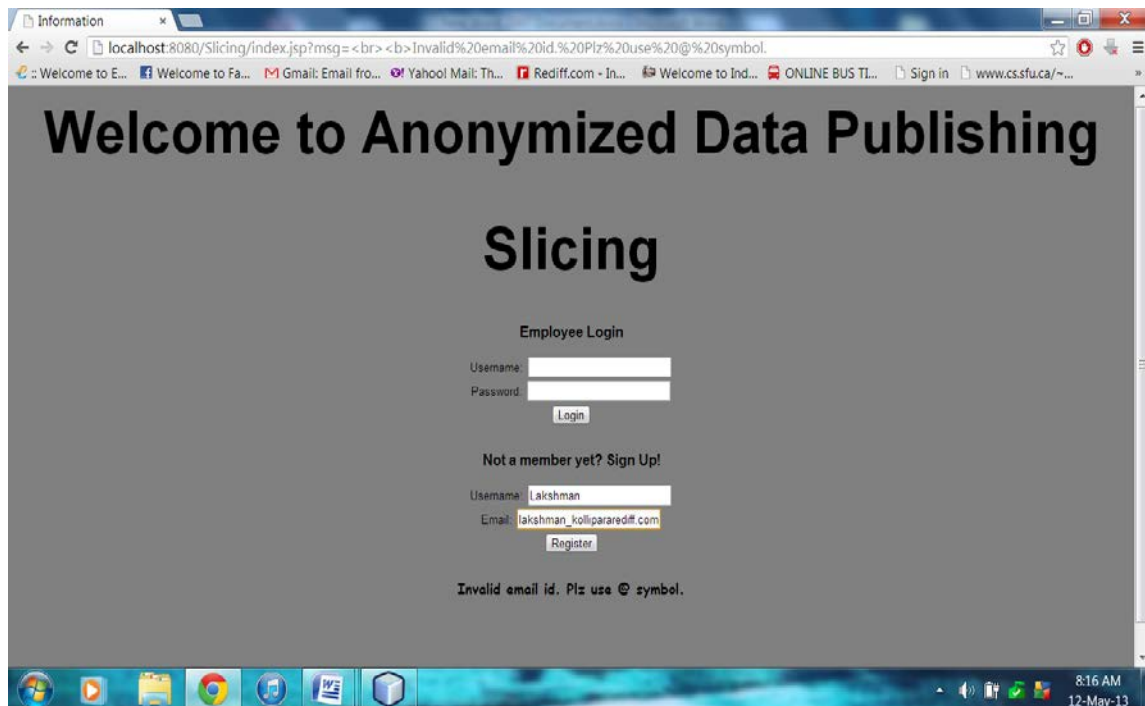
### 6.6.1 Testing Module 1

The first module is testing for Registration.

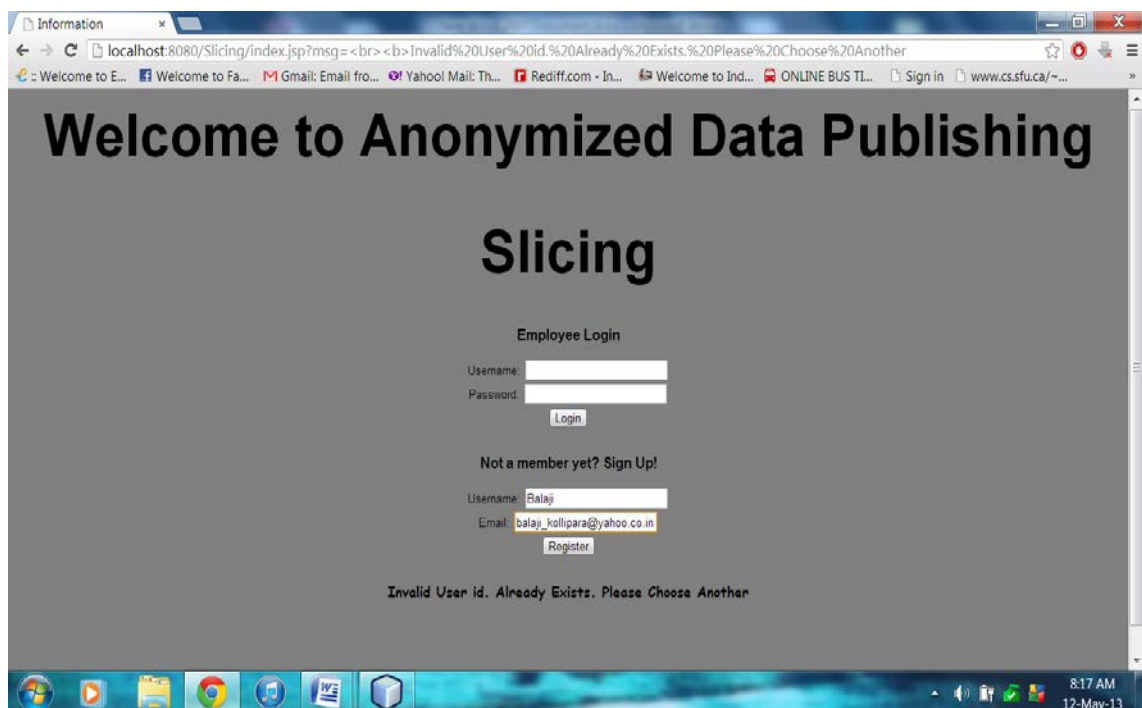
1. **Functionality Tested** : Testing the validation of registration form
2. **Test Execution Procedure** : Fill the form with credentials.
3. **Test Results Checking Method** : A message will be displayed when incorrect input is given

S I: No:	Condition to be tested	Expected Result	Result
1	E mail Id	If doesn't contain '@' or '.com' gives a message: Invalid Email Id	Pass
2	Contact number	Should contain 10 integers only	Pass
3	All fields should be filled	Prompts an alert when any field is empty	Pass
4	User Id & Username must be unique	Checks in database and If already exists, displays a message: Try another User Id	Pass

**Table 6.6.1 Test Case for Registration**



**Figure 6.6.1.1** Result of Testing for Registration(Email Id)



**Figure 6.6.1.2** Result of Testing for Registration(Unique User Id)

### 6.6.2 Testing Module 2

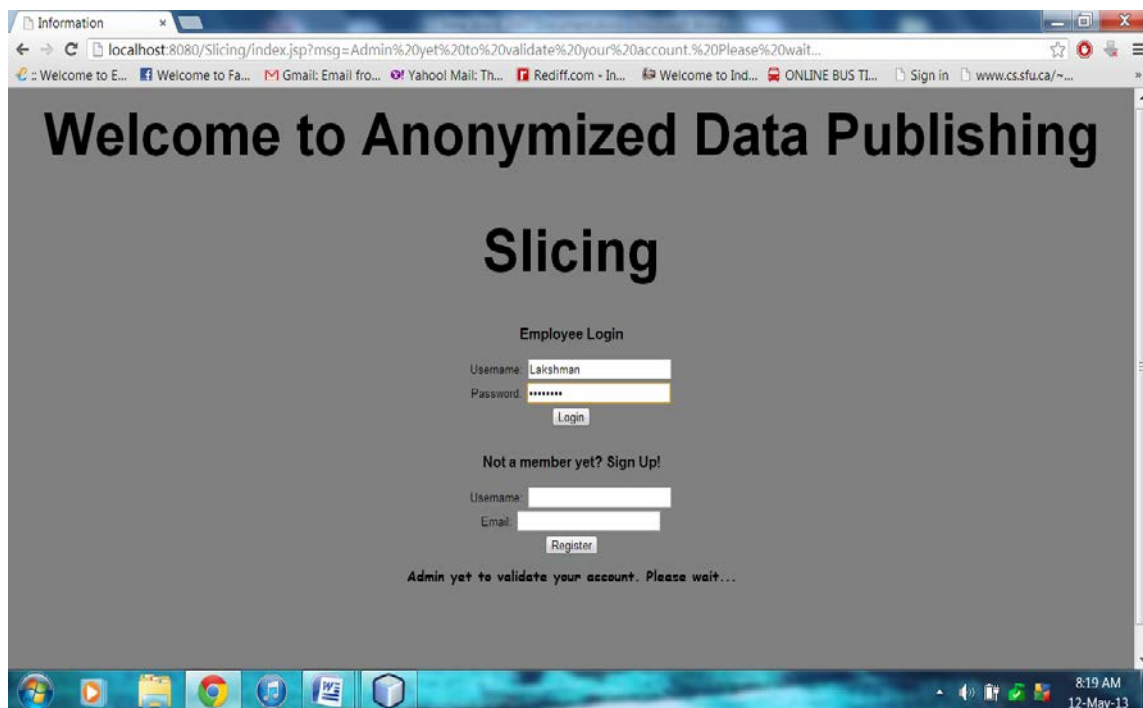
The Second module is testing for Validation of user

- 1. Functionality Tested** : The user is validated by admin or not
- 2. Test Execution Procedure** : The admin must make the status of user as 'yes'
- 3. Test Results Checking Method** : User can't login unless admin validates the employee by making status = 'yes'.

Sl: No:	Condition to be tested	Expected Result	Result
1	Admin validated employee	User can login	Pass
2	Admin didn't validate the employee	User can't login. A message will be displayed: Admin yet to validate	Pass

**Table 6.6.2 Test Case for Admin Validation**





**Figure 6.6.2.1**      **Result of Testing for Admin validation**

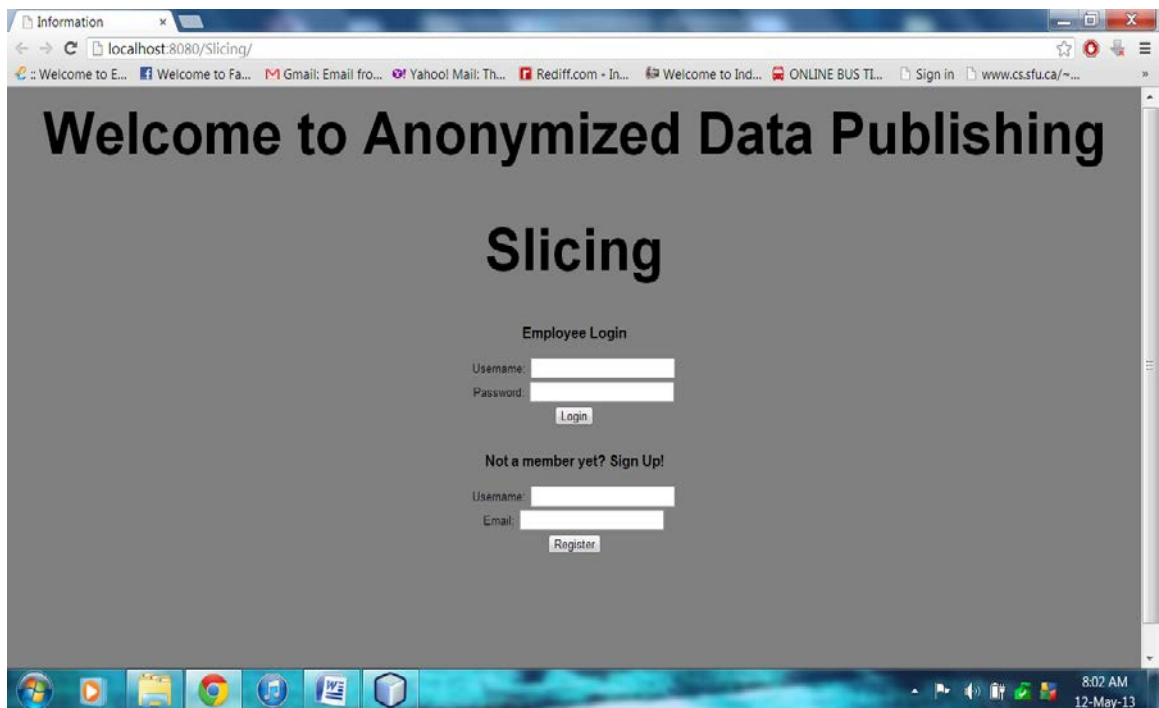
# **Chapter 7**

## **RESULT AND ANALYSIS**

## 7. RESULT AND ANALYSIS

### 7.1 Index Page

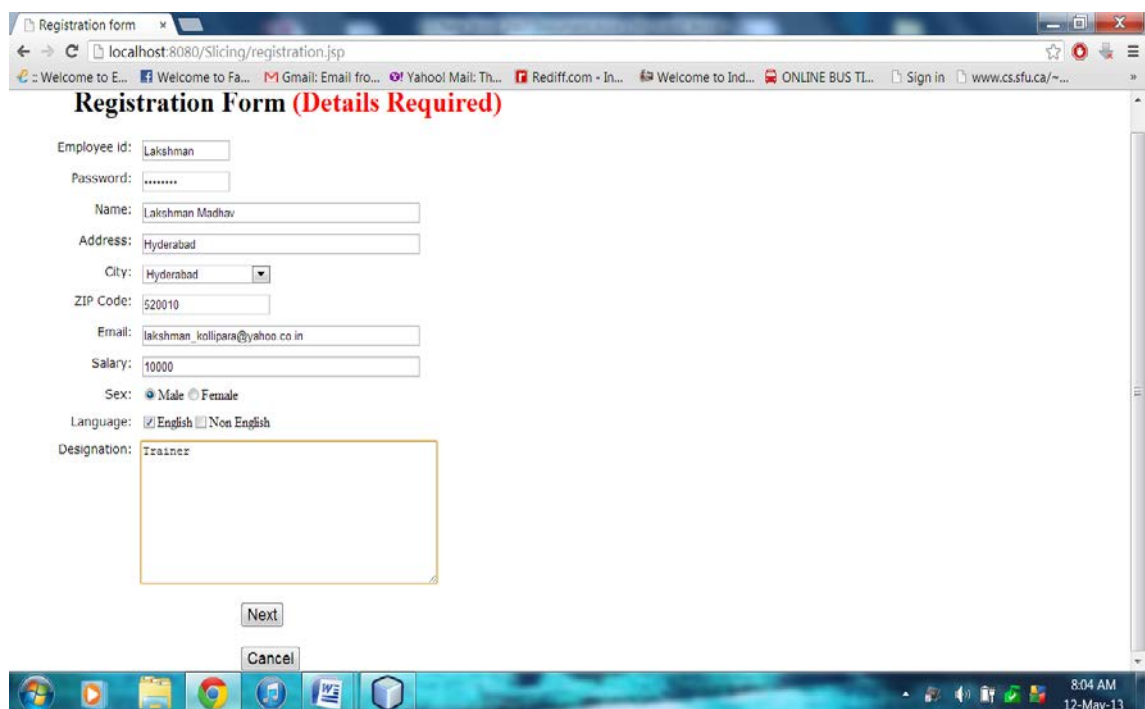
The Admin and Employee can login with their respective credentials and access their home pages. The new Employee can register by entering unique user id and email id and clicking on Register button.



**Figure 7.1** Index Page

## 7.2 Employee Registration

The new employee registers with the organization by entering his details like Name, User Id, Address, Designation, Email Id etc. An alert will be displayed when incorrect input is given. When user clicks on Register, we check in database whether the user Id is unique or already exists. If already exists, we display an error message.



The screenshot shows a web browser window with the title "Registration form". The address bar shows "localhost:8080/Sliding/registration.jsp". The page content includes a heading "Registration Form (Details Required)" in red. Below the heading is a registration form with the following fields and values:

- Employee Id: Lakshman
- Password: \*\*\*\*\*
- Name: Lakshman Madhav
- Address: Hyderabad
- City: Hyderabad (dropdown menu)
- ZIP Code: 520010
- Email: lakshman\_kolipara@yahoo.co.in
- Salary: 10000
- Sex: ☒ Male ☐ Female
- Language: ☒ English ☐ Non English
- Designation: Trainer (text area)

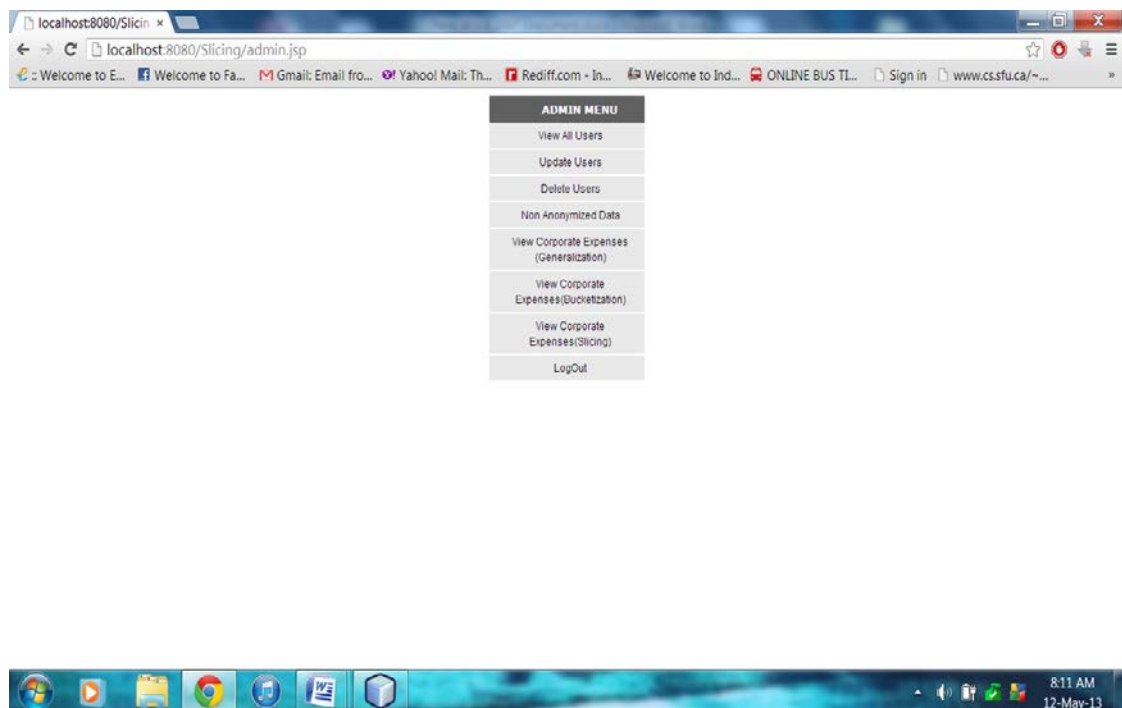
At the bottom of the form are two buttons: "Next" and "Cancel". The Windows taskbar at the bottom shows the time as 8:04 AM on 12-May-13.

**Figure 7.2 Employee Registration Form**

### 7.3 Admin Menu

When admin logs into his account, admin menu is provided. Admin can view all users and manage users like updating the user details and deleting a user. Admin can also apply the anonymization techniques such as generalization, bucketization, slicing on the dataset.

Figure 7.3.1 shows the operations of Admin, Figure 7.3.2 shows the all users present in the organization, Figure 7.3.3 shows the update of user details and Figure 7.3.4 shows the deletion of users by Admin.



**Figure 7.3.1 Admin Menu**

USER ID	EMPLOYEE NAME	SALARY	STATUS
Aabha	Aabha	30000	yes
Aabharana	Aabharana	35000	yes
Aabheer	Aabheer	15000	yes
Aadarsh	Aadarsh	10000	yes
Balachandra	Balachandra	12000	yes
Balagopal	Balagopal	20000	yes
Balagovind	Balagovind	25000	yes
Balaji	Balaji	14000	yes
Bolu	Bolu	85000	yes
Bani	Bani	40000	yes
Banmala	Banmala	80000	yes
Banni	Banni	50000	yes
Bansari	Bansari	65000	yes
Barsha	Barsha	70000	yes
Barsha	Barsha	55000	yes
Bhanu	Bhanu	90000	yes
Bharath	Bharath	60000	yes
Bharya	Bharath	75000	yes

Figure 7.3.2 View All Users

User ID :

User Name :

Salary :

Status :

[VIEW TABLE](#)

[ADMIN MENU](#)

Figure 7.3.3 Update Users

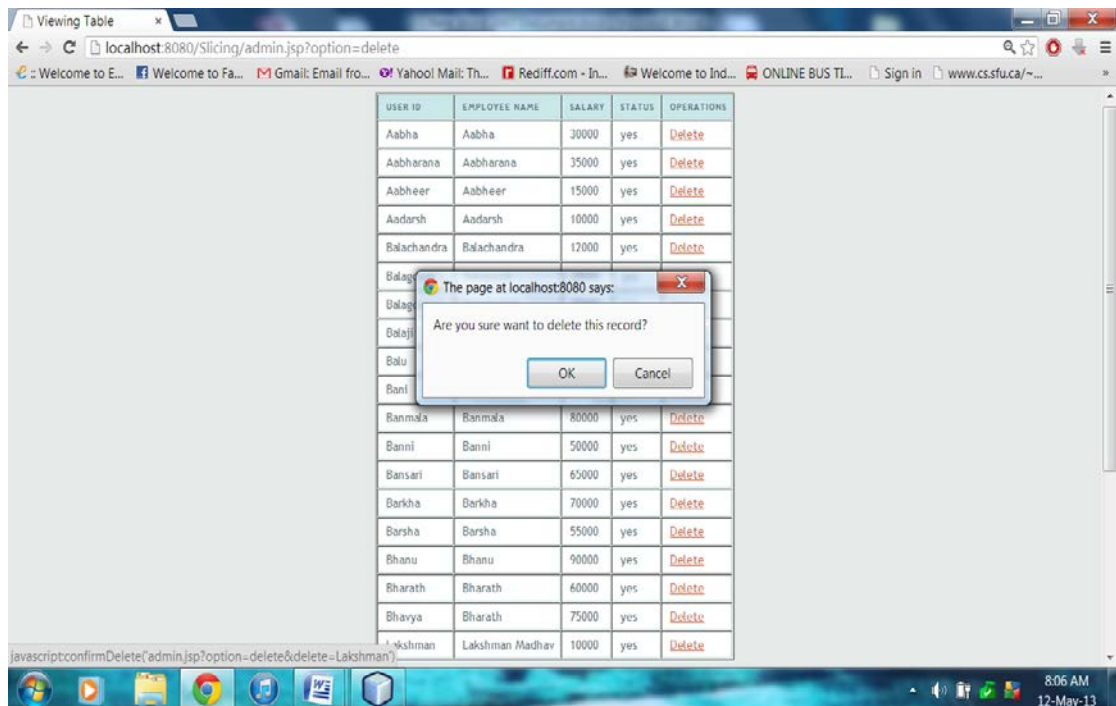


Figure 7.3.4 Delete Users

### 7.3.1 Non Anonymized Data

Figure 7.3.1 gives the data to be published before applying any anonymization techniques. If we publish this data in original form, Individual privacy is lost. So we apply the anonymization techniques on data to be published.



EMPLOYEE NAME	EMPLOYEE SALARY	EMAIL	ZIP	GENDER	DESIGNATION
Aadarsh	10000	Aadarsh@yahoo.com	560033	Male	Tester
Balachandra	12000	Balachandra@yahoo.com	560034	Male	Tester
Balaji	14000	Balaji@yahoo.com	560036	Male	Tester
Aabhveer	15000	Aabhveer@gmail.com	560032	Male	Trainer
Balagopal	20000	Balagopal@yahoo.com	560031	Male	Trainer
Balagovind	25000	Balagovind@gmail.com	560035	Male	Trainer
Aabhha	30000	Aabhha@gmail.com	560031	Female	Developer
Aabhharana	35000	Aabhharana@yahoo.com	560032	Female	Developer
Bani	40000	Bani@gmail.com	560035	Female	Developer
Banni	50000	Banni@yahoo.com	560044	Female	Team Leader
Barsha	55000	Barsha@gmail.com	560042	Female	Team Leader
Bharath	60000	Bharath@gmail.com	560045	Male	Team Leader
Bansari	65000	Bansari@gmail.com	560041	Female	HR Manager
Barkha	70000	Barkha@yahoo.com	560043	Female	HR Manager
Bharath	75000	Bharath@gmail.com	560045	Female	HR Manager
Banmala	80000	Banmala@yahoo.com	560041	Female	Pr. Manager
Balu	85000	Balu@gmail.com	560045	Male	Pr. Manager
Bhanu	90000	Bhanu@gmail.com	560041	Male	Pr. Manager
Total Company Expenses		831000.0			

**Figure 7.3.1.1 Non Anonymized Data**

### 7.3.2 Generalization

In generalization, we remove the identifiers and we generalize the values i.e. less specific but consistent values. In our Scenario, we applied generalization on employee salary, email id, zip code and gender. We generalized the salary of employees into two ranges such that all employees will fall under those two ranges. We generalized the last digit of zip code such that few people have same generalized zip code. We generalized the gender by keeping ‘\*’ such a way that no individual is linked to his record. Figure 7.3.2 shows the Generalized Table

Now the table satisfies the k-anonymity and attribute linkage and record linkage is protected and the probability of linking a person to his sensitive attribute is never more than 1/3.



EMPLOYEE DESIGNATION	EMPLOYEE SALARY	EMAIL	ZIP	GENDER
Tester	[10000.00-45000.0]	*****@yahoo.com	56003*	*
Tester	[10000.00-45000.0]	*****@yahoo.com	56003*	*
Tester	[10000.00-45000.0]	*****@yahoo.com	56003*	*
Trainer	[10000.00-45000.0]	*****@gmail.com	56003*	*
Trainer	[10000.00-45000.0]	*****@yahoo.com	56003*	*
Trainer	[10000.00-45000.0]	*****@gmail.com	56003*	*
Developer	[10000.00-45000.0]	*****@gmail.com	56003*	*
Developer	[10000.00-45000.0]	*****@yahoo.com	56003*	*
Developer	[10000.00-45000.0]	*****@gmail.com	56003*	*
Team Leader	[45000.0-90000.00]	*****@yahoo.com	56004*	*
Team Leader	[45000.0-90000.00]	*****@gmail.com	56004*	*
Team Leader	[45000.0-90000.00]	*****@gmail.com	56004*	*
HR Manager	[45000.0-90000.00]	*****@gmail.com	56004*	*
HR Manager	[45000.0-90000.00]	*****@yahoo.com	56004*	*
HR Manager	[45000.0-90000.00]	*****@gmail.com	56004*	*
Pr. Manager	[45000.0-90000.00]	*****@yahoo.com	56004*	*
Pr. Manager	[45000.0-90000.00]	*****@gmail.com	56004*	*
Pr. Manager	[45000.0-90000.00]	*****@gmail.com	56004*	*
Total Company Expenses		831000.0		

**Figure 7.3.2.1 Generalized Data**

### 7.3.3 Bucketization

In Bucketization, we remove the identifiers and the Sensitive Attributes are clearly separated from Quasi Identifiers and are randomly permuted breaking the correlations. In our scenario, the sensitive Attribute i.e. designation is separated from other quasi identifiers and the designations of employees are randomly shuffled. It prevents attribute linkage and record linkage but doesn't prevent Membership Disclosure because the Quasi Identifiers are present in their original form. Figure 7.3.3 shows the Bucketized Table.

Bucketization offers better data utility than Generalization because all the values are in their original forms. The probability of linking a person to his sensitive attribute is never more than  $1/6$ .

EMPLOYEE DESIGNATION	EMPLOYEE SALARY	EMAIL	ZIP	GENDER
Trainer	[10000.00-45000.0]	Aadarsh@yahoo.com	560033	Male
Trainer	[10000.00-45000.0]	Balachandra@yahoo.com	560034	Male
Developer	[10000.00-45000.0]	Balaji@yahoo.com	560036	Male
Developer	[10000.00-45000.0]	Aabheer@gmail.com	560032	Male
Tester	[10000.00-45000.0]	Balagopal@yahoo.com	560031	Male
Developer	[10000.00-45000.0]	Balagovind@gmail.com	560035	Male
Trainer	[10000.00-45000.0]	Aashu@gmail.com	560031	Female
Team Leader	[10000.00-45000.0]	Aasharana@yahoo.com	560032	Female
Pr. Manager	[10000.00-45000.0]	Bani@gmail.com	560035	Female
Pr. Manager	[45000.0-90000.00]	Banni@yahoo.com	560044	Female
HR Manager	[45000.0-90000.00]	Barsha@gmail.com	560042	Female
Tester	[45000.0-90000.00]	Bharath@gmail.com	560045	Male
Tester	[45000.0-90000.00]	Bansari@gmail.com	560041	Female
Pr. Manager	[45000.0-90000.00]	Barikha@yahoo.com	560043	Female
HR Manager	[45000.0-90000.00]	Bhavya@gmail.com	560045	Female
Team Leader	[45000.0-90000.00]	Banmalai@yahoo.com	560041	Female
Team Leader	[45000.0-90000.00]	Batu@gmail.com	560045	Male
HR Manager	[45000.0-90000.00]	Bhanu@gmail.com	560041	Male
Total Company Expenses		831000.0		

**Figure 7.3.3.1 Bucketized Data**

### 7.3.4 Slicing

Slicing partitions the data set both vertically and horizontally. Vertical partitioning is done by grouping attributes into columns based on the correlations among the attributes. Each column contains a subset of attributes that are highly correlated. Horizontal partitioning is done by grouping tuples into buckets. Finally, within each bucket, values in each column are randomly permuted (or sorted) to break the linking between different columns. The basic idea of slicing is to break the association cross columns, but to preserve the association within each column. In our scenario, Designation and Salary are correlated attributes which we group together for better data utility and we randomly permute both salary and designation together for better data privacy.

Slicing offers better data utility than Generalization and Bucketization because the highly correlated attributes are grouped together. Slicing offers better data privacy than above

techniques by randomly permutating the grouped attributes, which results in less probability of 1/108 for linking a person to his sensitive attribute.

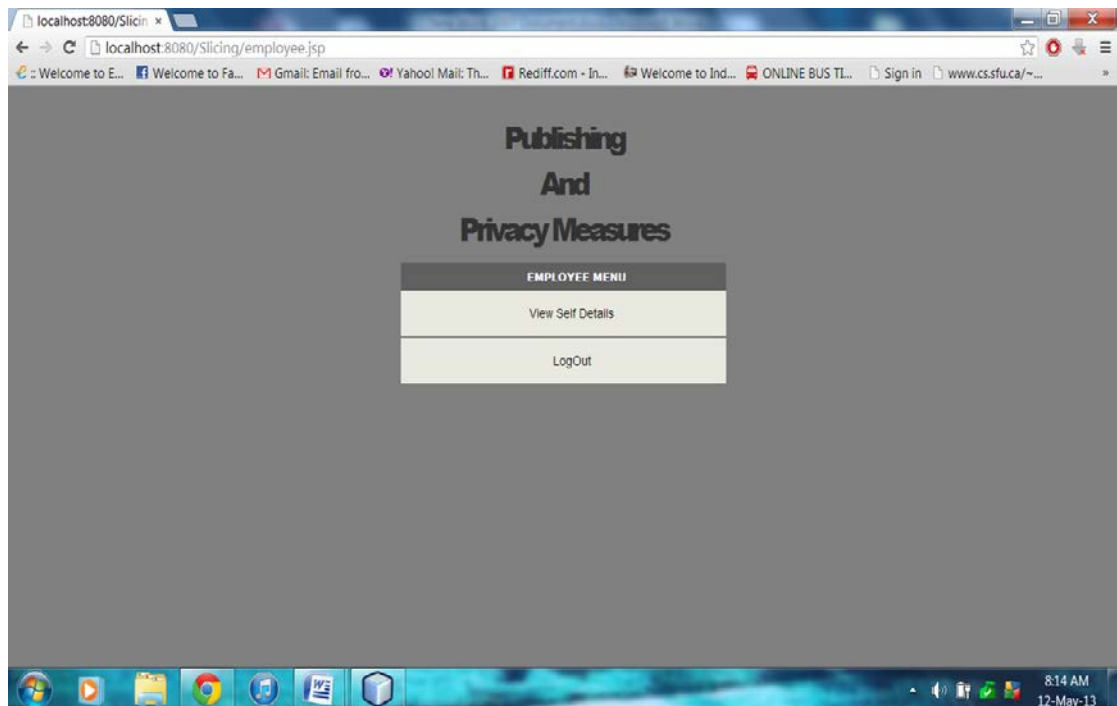


EMPLOYEE DESIGNATION	EMPLOYEE SALARY	EMAIL	ZIP	GENDER
HR Manager	52500.0-87500.0	*****@gmail.com	56003*	Female
Team Leader	37500.0-62500.0	*****@yahoo.com	56003*	Male
Trainer	18750.0-31250.0	*****@gmail.com	56003*	Male
Tester	9000.0-15000.0	*****@yahoo.com	56003*	Female
Developer	30000.0-50000.0	*****@yahoo.com	56003*	Male
Pr. Manager	63750.0-106250.0	*****@yahoo.com	56003*	Male
Team Leader	41250.0-68750.0	*****@gmail.com	56003*	Female
Tester	10500.0-17500.0	*****@gmail.com	56003*	Male
HR Manager	56250.0-93750.0	*****@yahoo.com	56003*	Male
Pr. Manager	67500.0-112500.0	*****@gmail.com	56004*	Male
Pr. Manager	60000.0-100000.0	*****@yahoo.com	56004*	Female
Tester	7500.0-12500.0	*****@gmail.com	56004*	Female
Trainer	15000.0-25000.0	*****@gmail.com	56004*	Female
Team Leader	45000.0-75000.0	*****@yahoo.com	56004*	Female
Developer	26250.0-43750.0	*****@yahoo.com	56004*	Female
HR Manager	48750.0-81250.0	*****@gmail.com	56004*	Male
Developer	22500.0-37500.0	*****@gmail.com	56004*	Female
Trainer	11250.0-18750.0	*****@gmail.com	56004*	Male
Total Company Expenses		\$11000.0		

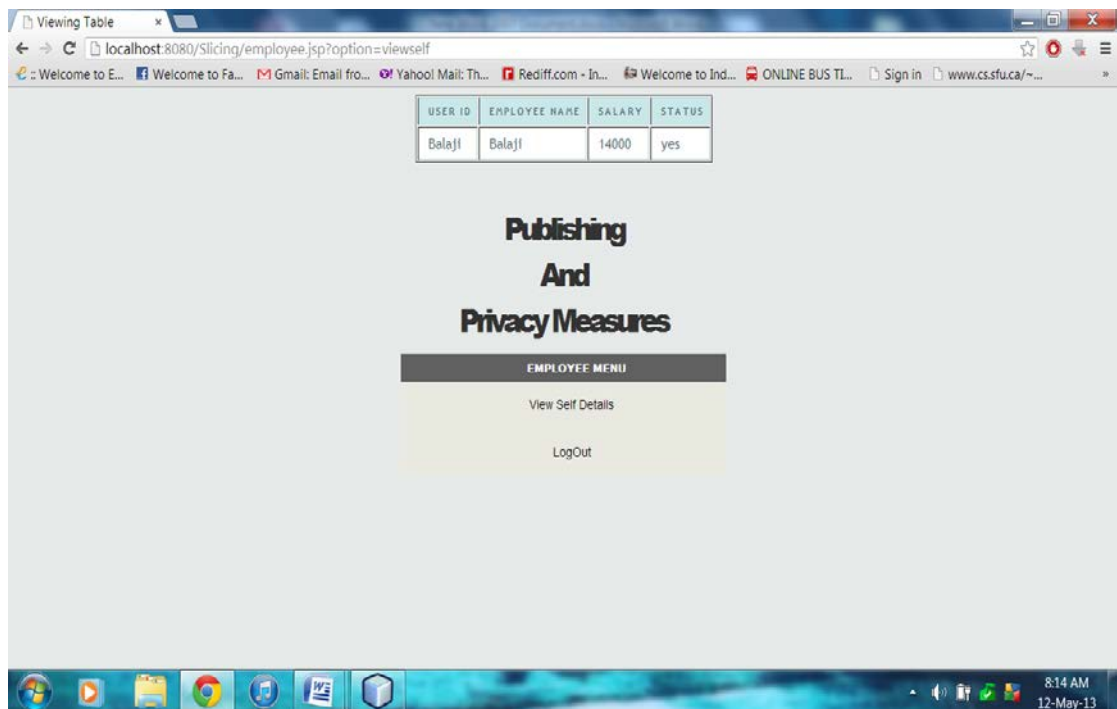
**Figure 7.3.4.1 Sliced Data**

## 7.4 Employee Menu

When employee logs into his account, an employee menu is provided where employee can view his self details and logout.



**Figure 7.4.1 Employee Menu**



**Figure 7.4.2 Employee Self Details**

# **Chapter 8**

# **CONCLUSION**

## 8. CONCLUSION

This paper presents a new approach called slicing for privacy preserving microdata publishing. Slicing overcomes the limitations of generalization and bucketization and preserves better utility while protecting against privacy threats. We illustrate how to use slicing to prevent attribute disclosure and membership disclosure. Our experiments show that slicing preserves better data utility than generalization and is more effective than bucketization in workloads involving the sensitive attribute.

The general methodology proposed by this work is that: before anonymizing the data, one can analyze the data characteristics and use these characteristics in data anonymization.

The rationale is that one can design better data anonymization techniques when we know the data better. Attribute correlations can be used for privacy attacks and slicing succeeds in maintaining a balance to prevent such attacks yet maintaining data utility.

Slicing is a promising technique for handling high-dimensional data. By partitioning attributes into columns, we protect privacy by breaking the association of uncorrelated attributes and preserve data utility by preserving the association between highly-correlated attributes. For example, slicing can be used for anonymizing transaction databases, which has been studied recently.

# **Chapter 9**

## **FUTURE ENHANCEMENTS**

## 9. FUTURE ENHANCEMENTS

This work motivates several directions for future research. First, in this paper, we consider slicing where each attribute is in exactly one column. An extension is the notion of overlapping slicing, which duplicates an attribute in more than one column. This releases more attribute correlations. This could provide better data utility, but the privacy implications need to be carefully studied and understood. It is interesting to study the trade-off between privacy and utility. Second, we plan to study membership disclosure protection in more details. Our experiments show that random grouping is not very effective. Third, slicing is a promising technique for handling high-dimensional data. By partitioning attributes into columns, we protect privacy by breaking the association of uncorrelated attributes and preserve data utility by preserving the association between highly correlated attributes. For example, slicing can be used for anonymizing transaction databases, which were being studied recently.

Finally, while a number of anonymization techniques have been designed, it remains an open problem on how to use the anonymized data. In our experiments, we randomly generate the associations between column values of a bucket. This may lose data utility. Another direction is to design data mining tasks using the anonymized data computed by various anonymization techniques.



# **Chapter 10**

## **REFERENCES**

## 10. REFERENCES

### Papers Referred:

Benjamin C. M. Fung, KE Wang, Rui Chen and Philips YU, "Privacy-Preserving Data Publishing: A Survey of Recent Developments" ACM Computing Surveys, Vol. 42, No. 4, Article 14, Publication date: June 2010.

David J. Martin , Daniel Kifer, Ashwin Machanavajjhala, Johannes Gehrke and Joseph Y. Halpern, "Worst-Case Background Knowledge for Privacy-Preserving Data Publishing"

L. Sweeney, " k-anonymity: a model for protecting privacy" International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, 10(5):557–570, 2002.

Neha V. Mogre, Prof. Girish Agarwal and Prof. Pragati Patil, "A Review On Data Anonymization Technique For Data Publishing" International Journal of Engineering Research & Technology (IJERT) Vol. 1 Issue 10, December- 2012

Tiancheng Li and Ninghui Li, "On the Tradeoff Between Privacy and Utility in Data Publishing" Purdue University, West Lafayette, IN 47907-2086, CERIAS Tech Report 2009-17

Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer and Muthuramakrishnan Venkitasubramaniam, "ℓ-Diversity: Privacy Beyond k-Anonymity"

P. Samarati and L. Sweeney, "Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization" Technical report, CMU, SRI, 1998.

Dinur and K. Nissim, "Revealing Information while Preserving Privacy," Proc. ACM Symp. Principles of Database Systems (PODS), pp. 202-210, 2003.

R.C.-W. Wong, A.W.-C. Fu, K. Wang, and J. Pei, "Minimality Attack in Privacy Preserving Data Publishing," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 543-554, 2007.

Tiancheng Li, Ninghui Li, Jian Zhang and Ian Molloy “Slicing: A New Approach for Privacy Preserving Data Publishing” IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 24, NO. 3, MARCH 2012

### **Presentations Referred**

“Data Privacy:A Bottleneck to Data Mining” by Ke Wang, School of Computing Science, Simon Fraser University, [www.cs.sfu.ca/~wangk](http://www.cs.sfu.ca/~wangk)

### **Books Referred:**

B. C. M. Fung, K. Wang, A.W.-C. Fu, and P. S. Yu. “Introduction to Privacy Preserving Data Publishing: Concepts and Techniques”, Data Mining and Knowledge Discovery Series. Chapman & Hall/CRC, August 2010. ISBN:9781420091489

Pratik Patel and Karl Moss, “Java database programming with JDBC” 2<sup>nd</sup> Edition, CORIOLIS GROUP BOOKS

Patrick Naughton, “The JAVA Handbook”, 2<sup>nd</sup> Edition, Tata McGraw-hill.

### **Websites Referred**

<http://java.sun.com>

<http://w3schools.com>

<http://roseindia.com>

<http://jsptut.com>

<http://www.tutorialspoint.com/servlets/>

<http://dev.mysql.com/doc/refman/5.0/en/tutorial.html>