PJP/2012/FTMK/(1D)/S01002

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

CHARACTERISATION OF DATA SET FEATURES FOR STORAGE SPACE OPTIMISATION USING FUNCTIONAL DEPENDENCY

PENYELIDIK:
DR. NURUL AKMAR EMRAN
DR. NORASWALIZA ABDULLAH
NUZAIMAH MUSTAFA

FAKULTI TEKNOLOGI MAKLUMAT DAN KOMUNIKASI

2013

TABLE OF CONTENTS

ABSTRACT

Within data intensive applications, data volumes often be large enough for storage space requirements to become an issue that must be dealt by data centre providers. The growth of data volumes calls for a way to manage storage space efficiently. One way to manage data storage space is through space optimisation. In order to optimise space, data centre providers need to choose space optimisation method(s) that is useful for the data sets being stored. However, studies on the characteristics of data sets that will be useful for space optimisation is limited even though such information is crucial in designing space optimisation strategy. We argue that, if we could determine the characteristics of data sets that are useful (or less useful) for space optimisation, data centre providers could make guided decision in implementing their space optimisation strategy. This research focuses on investigating the characteristics of data sets for space optimisation using functional dependency technique. The contribution of this research is the result of the experiment and the analysis conducted against real data sets for a space optimisation techniques just mentioned. This research concludes with the characteristics of data set features discovered within the microbial genomics data sets.

ACKNOWLEDGEMENT

Praise to Allah s.w.t for the strength, patience and endurance to complete this research. We would like to acknowledge Universiti Teknikal Malaysia Melaka for the financial assistance granted to pursue this research, the Faculty of Information and Commuication Technology and the Centre for Research and Innovation Management (CRIM). Without these bodies, the achievement of the objectives set for this research is not possible.

Dr. Nurul Akmar Emran

Dr Noraswaliza Abdullah

Nuzaimah Mustafa

LIST OF TABLES

LIST OF FIGURES

FIGURE **TITLE** **PAGE**

LIST OF ABBREVIATIONS

FDs	-	Functional Dependencies
IND	-	Inclusion Functional Dependencies
AFD	-	Approximate Functional Dependencies
CFD	-	Conditional Functional Dependencies
DQ	-	Data quality

CHAPTER 1

INTRODUCTION

1.1 Background

One prominent concern in the establishment of green data centers is to decrease carbon footprint and operating costs (e.g. cooling systems for data centers) by reducing the amount of physical data storages required. Scientific applications which rely on large of data volumes require physical data storages that are not only impractically large to maintain, but also contribute to inefficient power consumption.Within the context of scien- tific applications that require access to scientific databases, data volumes often be large enough for storage space requirements to become an issue that must be dealt by scientific data center providers. Expanding database storage is an option that data center providers could take in order to address the space issue, however this option leads to an increase in the amount of physical data storages (data servers) required. As more data servers are added, more electrical power is needed to run the additional data servers and to cooling-off those servers. The issue concerning data centers has been raised in a recent estimation which stated that the worlds data centers currently consume about 330 billion kWh of electricity every year, which is almost equal to the entire electricity demand of the UK (Horn & Cook, 2011). In addition, power consumption that exceeds 100 billion kWh generate approximately 40, 568, 000 tons of CO2 emissions (Hazelhurst, 2008), (Kang, et al., 1990), (Kumar, 1992). Thus, in establishing successful green data centers, adding more data servers is not an interesting option to choose in dealing with the storage space issue as this option leads to undesirable increase in power consumption and in CO2 emissions.

1.2 The Proxy-based Approach

Within the context of applications that require access to databases, data volumes often be large enough for storage space requirements to become an issue that must be dealt by data center providers. Expanding database storage is an option that data center providers could take in order to address the space issue, however this option leads to an increase in the amount of physical data storages (data servers) required.

One way to reduce storage space requirement is by optimising the available database space. In fact, the need to optimise space is not new, as tools and techniques for this purpose provided by enterprise data storage vendors (such as Oracle and DB2) have been available in the market for about a decade. At the relational table level, data compression tools, for example, apply a repeated values removal technique to gain free space (Lai, 2008). In addition, data deduplication techniques remove duplicate records in the table to gain storage space (Freeman, 2007). The idea behind these space optimisation solutions is to exploit the presence of overlaps (of values or records) within tables. Both of these techniques are performed at the level of whole tables. A key (though often unstated) assumption behind these optimisation techniques is that all columns can be exploited for space optimisation. Because of this assumption, knowledge of semantics of applications (i.e., how the columns are used) is ignored and as the consequence, data center providers need to bear unnecessary query processing overhead for frequent compression (and decompression) of heavily queried data.

The key lesson learnt from space optimisation techniques that are available in the market to date is that, space optimisation techniques that achieve space saving at both schema level and whole tables level are limited. In addition, space optimisation techniques that consider knowledge of semantics of applications have not been studied in depth. Because of these limitations, the two techniques described above unfortunately do not fully support solving the storage space issue faced by data center providers, where knowledge of how database is used must be considered for space optimisation. Therefore, an alternative space optimisation technique is proposed to address the limitations of the existing techniques. This new, alternative technique is crucial to support data center providers in dealing with high storage space requirements.

In this research, we propose a space optimisation technique called the *proxy- based approach*. The proposed technique will be designed by exploiting the functional dependencies discovered within the database where, smaller alternatives called *proxies* will be used to substitute the information (in form of set of values) that are removed from the database. For example, Figure 2 shows a possible substitution made in a table (Table R) by a proxy attribute B for attribute D, an attribute which is removed from the table (shown as shaded column) where functional dependency between B and D (denoted as B → D) is present.

Table R

A	B	D
001	X	a
002	X	a
003	Y	b
004	Y	b
005	Y	b

A substitution table

B →	D
(X) →	a
(Y) →	b

Figure 1: An example of substitution made by proxy attribute B for attribute D

Basically, the proxy-based approach method offers space saving through database schema modification, in particular by dropping attributes from the schema under con sideration. The removal of the attributes, of course, will cause information loss and consequently will affect the queries that rely on those attributes. However, if the missing information can be retrieved from other attribute(s), the queries could still be computed using the smaller database. We use the term 'proxies' for attributes that substitute other attributes in the schema, which is inspired by proxies in other contexts with similar roles (e.g., in voting, a proxy is a person authorised to act on behalf of another (Petrik, 2009)). We identified the proxies based on functional dependency relationship that can be observed among attributes in relational tables. An understanding of the space-accuracy trade-offs that the proxies could offer is required to facilitate the decisions in selecting which attributes can be deleted from the universe schema. Therefore, answering the following questions regarding proxies are crucial before we can decide on its applicability:

- How do proxies contribute to space saving?
- How do we select the attributes to drop from the schema?
- What determines the amount of space saving that can be offered by proxies?

The idea behind the technique we propose is to achieve space saving through both database schema modification and exploitation of the presence of overlaps. Specifically, space saving through schema modification is achieved by dropping some attributes from the schema. If some attributes are dropped from the schema, the amount of space saved is roughly determined by the number of attributes being dropped and the number of tuples the table contains. For example, consider a table which consists of 100 tuples, with several attributes in its schema. If we drop an attribute from the schema, then the amount of space saved is 100 units of instances[1] (which is of course, is convertible to disk storage unit in bytes).

The question that arises is whether all attributes in the schema are droppable. To answer this question we need to understand the semantics of the application. As for the microbial genomics application, we need to understand how the data set is used in answering data set requests for the analyses. In particular, we need to know how attributes in the schema of the microbial database tables are used.

Nevertheless, before we can validate the usefulness of this alternative technique, studies on the characteristics of data sets that will be useful for space optimisation is needed. This information is crucial in designing space optimisation strategy for data centre providers that need to deal with storage space constraints. Moreover, substituting the values of the column which are missing (as the result of dropping the table columns from the schema is crucial) in order to determine the practicality of the approach. Therefore, in this research, the known functional dependency theory will be applied to predict the missing values in the data sets. In the next section, the types of functinal dependency will be presented.

1.3 Functional Dependency

The major roles of dependencies are involved in designing of database, quality management of data and knowledge representation. Basically, the dependencies are used in normalization of database and applied in database design to deserve the quality of data.

[1] We regard each cell in a common relational table as an instance

Dependencies in knowledge discovery are mined from available data from a database. This extraction process is known as dependency discovery where the objective is to find all the dependencies in available data. Types of dependencies are functional dependency (FDs), Inclusion Dependency (INDs), Approximate Functional Dependency (AFD) and conditional Functional Dependency (CFDs).

Table 1: Types of dependencies

Dependency	Definition
Functional Dependencies (FDs)	A functional dependency (FDs) describes a relationship between attributes in a single relation. An attribute is functionally dependent on another if we can use the value of one attribute to determine the value of another. (Liu, et al., 2012)
Approximate Functional Dependencies (AFDs)	An Approximate Functional Dependency (AFDs) is define as approximate satisfaction of a normal FD f : X →Y. (Liu, et al., 2012)
Conditional Functional Dependencies (CFDs)	A Conditional Functional Dependency is an expansion of FDs by supporting patterns of semantically associated constants, and also used in cleaning of relational data. (Liu, et al., 2012)
Inclusion Dependencies (INDs)	An Inclusion Functional Dependency (INDs) one of the valuable dependency since it helping the developer to define what data must be duplicated in what relations in a database. (Liu, et al., 2012)

The statement X->Y is the same for most of the FDs and AFDs. The difference only can be seen through the satisfaction level. The statement X->Y must satisfy for all the tuple of relation in FDs while AFDs shows small part of tuples to be violate in FD statement. On the side, CFDs use different statement (X-> Y,S) and the satisfaction is based on the tuples that match the tableau. The CFD can equivalent to FD if the tableau have one and only pattern tuple with "-" values.

One of the important uses of discovered dependencies is to improve the data quality. The primary function of implementing dependency in a database is to permit the data quality of the database. Missing values or errors in data sets can be recognised by analysing the discovered dependencies that hold among the attributes. Finally, this will help to evaluate the quality of data. Data errors or missing values cause negative effect in many application domains for example in bioinformatics. Basically, missing values occurs in bioinformatics for various reasons such as incomplete resolution, image corruption and due to presence of foreign particle or dust in a sample. This kind of missing values may cause irregularity in analysis of biological data for example to determine the function, domain or taxonomy of a certain species. Recently many researchers focus to improve data quality of a database by discovering dependencies among the data set attributes. (Liu, et al., 2012).

Among the four types of dependencies, functional dependency has the main key function in the determination of missing data. FDs also guarantee the accuracy of missing data prediction compared to the other dependencies. Beside this, the FDs used to discover the attributes to analyse space reduction in the database storage.

Therefore, the major focus in this research is implementing functional dependency to learn the characteristics of data set attributes (called as proxies) in preparation of missing values prediction for microbial genomics data sets. The perception of functional dependency is one of the primary dependencies which is important in designing and developing of a database. In contrast of design the database using FDs, properties of FDs studies as well. FDs may consider as integrity constraints that determine semantics of data. Data quality problem may arise due to violations of FDs in a sample datasets. Hence this missing data prediction may help to solve the data quality problem as well as to reduce the storage space.

1.4 Problem statement

In implementing storage space optimisation using the proxy-based approach, we need to understand the characteristics of data sets that will be of useful to utilise the proxies. In this research, we address the problem of: 'How can we determine the characteristics of data sets that will be make proxies useful in terms of space saving?'

1.5 Research Questions

The following are the research questions that we set to answer in order to deal with the problem as mentioned in Section 1.4:

1. How FDs can be used to predict the missing data?
2. What are the requirements to prepare the data sets for missing data prediction?
3. What are the characteristics good proxies?

1.6 Aims and Objective

This research aims to define the characteristics of proxies and to determine whether it is useful and implementable in practice. The following are the primary research objectives:

1. To identify the types of dependencies from the literature
2. To analyse properties of FDs that can offer missing data prediction
3. To discover FDs that are useful for missing values prediction.

1.7 Research Contribution

Studies on the characteristics of data sets that will be useful for space optimisation is needed is crucial in designing space optimisation strategy for data centre providers that need to deal with storage space constraints. By understanding the characteristics of data sets that will contribute to gaining spaces, databased designer can make informed decision regarding to data centers capacity planning. The contribution of this research is the result of the experiment and analysis conducted against real data sets for space optimisation techniques using proxies.

CHAPTER 2

LITERATURE REVIEW

2.1 Background

In this chapter, we provide a literature review on data dependency with the aim to learn the different forms of dependencies in preparing the methods to predict missing values in data sets. By learning the features and properties of FDs in the literature, an understanding of the different dependencies can be achieved.

2.2 Application of Functional Dependency in different domain

Data quality, concerning completeness of data sets is not a new problem; researchers has been started the studies since 1980's. Some of the researchers use FDs to detect missing data in a sample datasets. (Liu, et al., 2012).

A functional dependency states that if in a relation two rows agree on the value of a set of attributes X then they must agree on the value of a set of attributes Y. The dependency is written as $X \rightarrow Y$. For example, in a relation such as Buyers (Name, Address, City, Nation, Age, Product), there is a functional dependency City → Nation, because for each row the value of the attribute City identifies the value of attribute Nation. Cleaning works of data focus more on removing duplicates or dealing with syntactic errors. (George, et al., 2010).

Dependencies have very important roles in designing of database, quality management of data and knowledge representation. Application of dependencies can be normally in observed in database design (through normalisation data normalisation) to preserve data consistency. Functional Dependency (FD) for instance is applied, checking data of Disease and Symptom columns in a medical database. If Pneumonia is a value of disease and fever is a value of symptom and if every patient has a fever, then fever is said to be associated with pneumonia. If the relationship continues for every pair of symptom and disease values, then disease functionally determines symptom. Additionally, discovered of dependency from existing data will be used in determining whether data sets in databases correct and also to check the semantics of data of an existing database. The primary role of dependency application in database is to check the quality of data in the database. (Li, et al., 2012).

2.2.1 Methods for FDs discovery

The methods proposed in discovery of functional dependency are either top-down approach or bottom-up approach. Candidates of FD were generated level-by-level and then checking of candidates of FD's satisfaction against the relation or its partitions is performed in top-down approach. Bottom-up approach is started with tuples comparison to get agree-sets or difference-sets then only candidate FD were generated. This is followed by checking them against the agree-sets or difference-sets for satisfaction (Li, et al., 2012).

It has been discovered that the large databases been violated where an underlying set of constraints and data inconsistent through data integration systems. Data inconsistency has been attacked in different ways and there were different steps taken to deal with this data inconsistency. The first step is trying to extract the most reliable answer to query posed to an inconsistent database. The second step is by minimally modifying repairing an inconsistent database; the modification can be done through deleting or inserting tuples or value. The last step is by producing a nucleus, which is a condensed representation of all repairs that can be used for consistent query answering. But the main focus of the researcher here is to repair the database that violates a set of functional dependencies by modifying attribute values. V-repairs been introduced by the researcher to repair an inconsistent database with respect to functional dependencies. V-repairs basically database that have variables representing incomplete information. This V-repair reproduce two types of changes made to the original database: changing a constant to another constant whenever there is enough information for doing so, and changing a constant to a variable whenever we cannot suggest a constant for an incorrect value. (Kolahi & Lakshmanan, 2009).

	name	*cnt*	*prov*	*reg*	*arCode*	*phone*
t_1	Smith	CAN	BC	Van	604	1234567
t_2	Adams	CAN	BC	Van	604	7654321
t_3	Simpson	CAN	BC	Van	604	3456789
t_4	Rice	CAN	AB	Vic	604	9876543

(a)

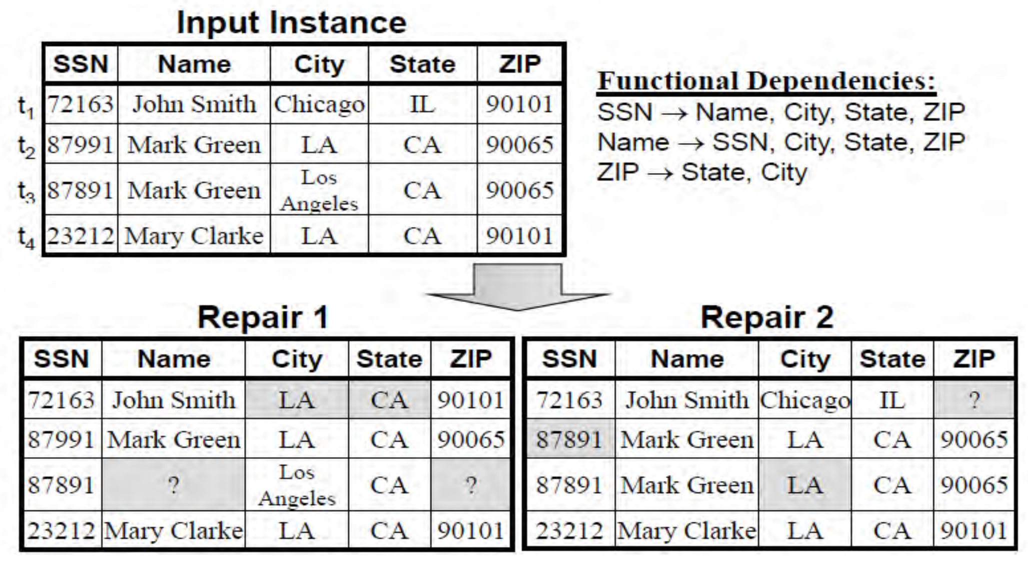
	name	*cnt*	*prov*	*reg*	*arCode*	*phone*
t_1	Smith	CAN	BC	Van	604	1234567
t_2	Adams	CAN	BC	Van	604	7654321
t_3	Simpson	CAN	BC	Van	604	3456789
t_4	Rice	v_1	AB	Vic	604	9876543

(b)

Figure 2 (a) A database instance violating $\sum = \{cnt, arCode \rightarrow reg, cnt, reg \rightarrow prov\}$. (b) An optimum V-repair (Kolahi & Lakshmanan, 2009)

Figure 2(a) shows a database instance over name, country (cnt), province/state (prov), region (reg), area code (arCode) and phone. However the database instance in Figure 2(a) violates the functional dependencies $\sum$ = {cnt, arCode → reg, cnt, reg → prov}. Figure 2(b) shows two necessary value modifications to solve the repair the violations. One, researcher change the value of reg ‘Man” to the correct value of ‘Van” and in the other is change the value ‘CAN’ with variable v_1. This shows that to achieve an optimum repair, the best option is to change the value of country to something else. The semantics is that v_1 stands for a value outside the active domain of cnt. (Kolahi & Lakshmanan, 2009).

Functional dependency abusing is very common and may arise in the context of data integration or Web data extraction. Functional dependency also known as Integrity constraints, encode data semantics. Hence, FD violations show variation from the expected semantics, which is caused due to data quality problems. Figure 3 shows a sample database and a set of FDs, where some of the values have been violated (e.g., tuples t_2 and t_3 violate ZIP→City, tuples t_2 and t_3 violate Name →SSN,City, and tuples t_1 and t_4 violate ZIP → State,City). (George, et al., 2010).



Input Instance

	SSN	Name	City	State	ZIP
t_1	72163	John Smith	Chicago	IL	90101
t_2	87991	Mark Green	LA	CA	90065
t_3	87891	Mark Green	Los Angeles	CA	90065
t_4	23212	Mary Clarke	LA	CA	90101

Functional Dependencies:
SSN → Name, City, State, ZIP
Name → SSN, City, State, ZIP
ZIP → State, City

Repair 1

SSN	Name	City	State	ZIP
72163	John Smith	LA	CA	90101
87991	Mark Green	LA	CA	90065
87891	?	Los Angeles	CA	?
23212	Mary Clarke	LA	CA	90101

Repair 2

SSN	Name	City	State	ZIP
72163	John Smith	Chicago	IL	?
87891	Mark Green	LA	CA	90065
87891	Mark Green	LA	CA	90065
23212	Mary Clarke	LA	CA	90101

Figure 3. An example of an unclean database and possible repairs. (George, et al., 2010)

Basically, there are many ways to modify a table which is satisfies all the required FDs. One of the way is to delete the wrong tuples (ideally, delete the fewest possible such tuples) such that the remainder satisfies all the FDs. For example, the researcher, “repair” the relation instance in Figure 3 by deleting t_1 and t_3. But, if delete the whole tuples may arise new problem where loss of “clean” data if only one of its attribute value is wrong. However the researcher modifies the selected attribute values. Figure 3 show two possible ways to repairs obtained from attribute modifications; and the questions marks specify that an attribute value can be modified to one o several values in order to satisfy the FDs. In between, the researcher also mentions that the existing methods do not identify the needs of the following criteria such as Interactive data cleaning, data integration, and uncertain query answering. (George, et al., 2010)

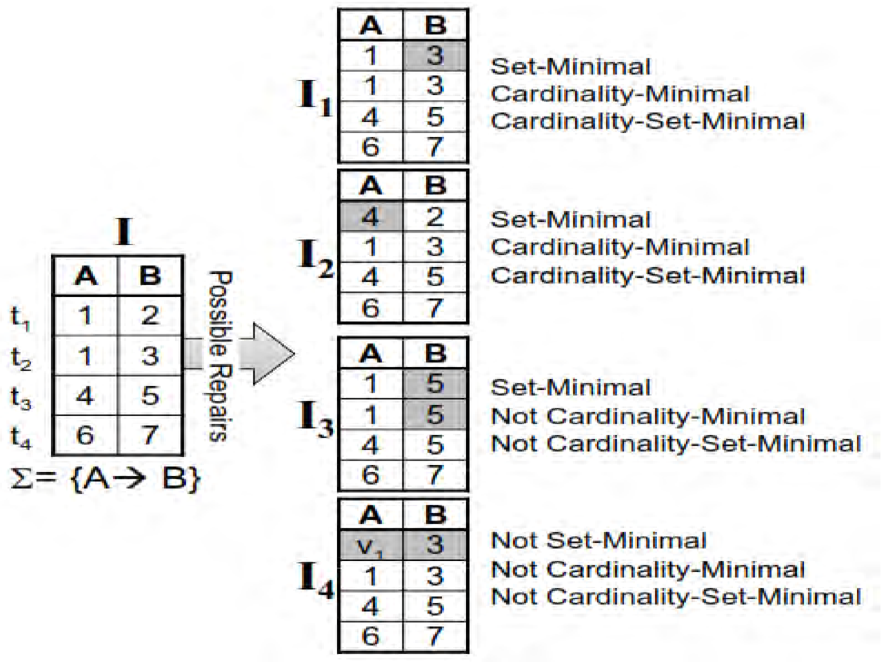


Figure 4. Example of various types of repairs. (George, et al., 2010)

Figure 4shows, few types of repairs have been proposed by the researcher in order to correct the wrong value in violation of functional dependencies. Repairs I_1 and I_2 are cardinality-minimal because no other repair has fewer changed cells. Clearly, I_1 and I_2 are also cardinality-set-minimal and set minimal. I_3 is set-minimal because reverting any of the changed cells to the values in *I* will violate A → B. On the other hand, I_3 is not cardinality-set-minimal (or cardinality-minimal) because changing t_1 [B] to 3 and reverting

t_2 [B] to 3 gives a repair of *I*. I_4 is not set-minimal because I_4 satisfies A $\rightarrow$B even after reverting t_1 [A] to 1. (George, et al., 2010).

The researchers focus analysis on semantic error detection in order to verify accuracy of the stored information. Data constraints and functional dependencies are the main issues in relational database. Apiletti and colleagues has proposed means of association rule mining to discover the data constraints and functional dependencies using.

Syntactic anomalies can be divided into few categories where it is occur due it incompleteness (lack of attribute values), inaccuracy (presence of error and outliers), lexical errors, domain format errors and irregularity (Apiletti, et al., 2006).

Semantic anomalies where there are discrepancy, due to a conflict between some attribute values, ambiguity, due to the presence of synonyms, homonyms or abbreviations, redundancy due to the presence of duplicate information, inconsistency due to an integrity constraint violation or functional constraint violation, invalidity due to the presence of tuples that do not display anomalies of the classes above but still do not represent valid entities (Apiletti, et al., 2006).

Association rules were applied to biological data cleaning for detecting outlier and duplicates, and to Gene Ontology to find relationships among terms of the ontology levels. But at the same time, it is not used to find constraints or dependencies. Using association rules, can find the causality relationship among the attribute values. Hence, analyse the support and confidence of each rule to detect the data constraints and functional dependencies. (Apiletti, et al., 2006).

Molinaro and Greco (2010) found that there are some problems in repairing and querying a database in the presence of functional dependencies and foreign key constraints. An attributes of a particular that present on right-hand side of FDs cannot appear on the left-hand side called canonical (FDs). Researchers proposed semantics of constraint satisfaction for databases which contain null and unknown values for the tuple insertions and updates. (Molinaro & Greco, 2010).

(a) Research

Name	*Manager*
p1	John
p2	Bob
p3	carl

(b) Employee

Name	*Phone*
John	123
Bob	111

Figure 5. Sample inconsistent databases (Molinaro and Greco 2010).

Project

Name	*Manager*
p1	#1
p2	carl

Employee

Name	*Phone*
John	123
bob	111
carl	$\perp_1$

Figure 6. Sample consistent databases (Molinaro and Greco 2010).

Suppose to have the following set of constraints (functional dependencies and foreign key constraints):

- fd_1 : *Name* $\rightarrow$ *Manager defined over Project*,
- fd_2 : *Name* $\rightarrow$ *Phone defined over Employee*,
- *fk* : *Project [Manger]* $\subseteq$ *Employee [Name]*.

Figure 5 shows an inconsistency database where there's occurrence of violation on both fd_1 and *fk*: for same research two different managers *p1* and *carl*, present in research relation, but not in employee table. Figure 6shows repairing of database. (Molinaro & Greco, 2010).

In Figure 6where #1 is an unknown value whose domain is *{john, bob}* whereas $\perp$1 is (labelled) null value. The FD fd_1 satisfied through introduction of unknown value *#1* which shows that the p1 gas a unique manager either *john* or *bob*. The *fk* in first tuple of the relation not violated because of *p1*, anybody in here, is in the employee relation too. The consistency of the original database w.r.t. *fk* is restored by inserting the manager *carl* into the employee relation. (Molinaro & Greco, 2010).

Null value was introduced in the Figure 6for the phone number of *carl* because of the information is missing. Here, we do not know whether the telephone number of carl does not exist or exists but is not known. Thus, neither the ‘‘nonexistent” (a value does not exist) nor the ‘‘unknown” (a value exists but is not known) interpretation of the null is applicable in this situation. Thus, both unknown and null values express incomplete information, even though unknown values are ‘‘more informative than” null values. (Molinaro & Greco, 2010).

From the database of Figure 2.5, the consistent answer to the query asking for the manager of *p2* is *carl,* because this answer can be obtained from every possible world of the repaired database. Clearly, there is no consistent answer to the query asking for the manager of *p1*, whereas the consistent answer to the query asking for the telephone number of *p2* 's manager is $\perp$1, that means that we have no information about it. (Molinaro & Greco, 2010).

In addition, Yao, J.Hamilton and J.Butz, n.d. had proposed a new method for discovery of functional dependency called FD_mine. This new approach will help to decrease the size of data set as well to detect the number of FDs present. Beside this, this algorithm will also prevent the data set from lost its information. This FD_Mine algorithm is based on level-wise searching. For example the results from level k will be used in next level which is level k+1. At first, all the FDs X->Y where X and Y are the single attributes were stored in FD_SET F_1. Thus, the candidates in this set refer to L_1. Candidates X_i X_j of L_2 was generated from F_1 and L_1. Second level, FDs are detected from X_i X_j -> Y and stored in FD_SET F_2. And then, F_1, F_2, L_1, and L_2 utilised to produce the L_3 candidates and so on till there's no remaining of candidates. (i.e., $L_k = \phi$ ($k \leq n-1$)). (Yao, et al., n.d.)

2.3 Data Incompleteness problem: Missing values

Missing values in a sample datasets is not a new problem faced by the scientist due to its negative impacts on scientific analysis results. In bioinformatics database management, it is important to get complete and correct datasets. This is because in future this datasets will be used for further research such as experimental analysis or development of model. Many field such as computer science, statistics, economics, and bioinformatics are concerned for good data quality. The focus of this research is on the missing values problem faced by microbial genomics domain. Microbial genomics is the study of microbe's genomes, it sequences, functions and structures. Bioinformatics can be divided into few different domains for instance genomics, proteomics, RNA and DNA, gene expression, and phylogenetics.

The following are the studies in which missing values are key factor in several application domains:

- In gene expression microarray data, missing values frequently create problems. Because missing data, can delay the downstream analysis such as gene clustering, distance calculation between gene networks. Tuikkala et al., proposed an imputation method to produce complete datasets. (Tuikkala, et al., 2008)

- Phylogenetics is evolutionary relation study among a group of organisms which is discovered through sequencing data and morphological data matrices. Missing values cause problem in phylogenetics analysis in terms of taxonomy and characters of organisms. Hence the overall classification among the organisms is not accurate and complete. J.Wiens and C.Morrill conduct new approach to determine the effect of missing data in phylogenetic analysis. They did the analysis in terms of simulation and empirical studies. (J.Wiens & C.Morrill, 2011).

- In genomics, missing values cause problem when the data matrix cannot be represented in memory. In addition it is also possible to produce biases in terms of results from scientific analysis. For example in Single-nucleotide polymorphism (SNP) identification missing values may cause calculation imbalance and complicated for statistical analyses. Therefore, Li et al. implement an approach called Bayesian Association with Missing Data (BAMD) to detect the SNP interactions without any effects from missing data. (Li, et al., 2012)

Since the missing data can cause negative effects to various field, it must be handled in a proper way where it can give best and accurate results in the analysis.

2.4 Conclusions

Basically, this chapter provides background about functional dependencies dealing with the missing data prediction in the database and statistics in different application domains. Here the analysis on FDs was conducted from different aspects for example in bioinformatics domain. Beside this, the FDs provide important roles in prediction of missing data also been surveyed through the literature studies.

CHAPTER 3

MATERIALS AND METHODS

3.1 Background

This chapter describes about the methodology and materials that is used in this research. The very first step in this research illustrate about the general method and data set we used for the analysis and why we choose it. And there are also details about the TANE algorithm that we used to obtain the FDs between the attributes. In addition, it also followed by conclusion of the chapter.

3.2 Research Methodology

As shown in Figure 3.1, first step in this research is to check for the data available in the Comprehensive Microbial Resources (CMR) and to download sample data sets. CMR is a freely available website to show information about complete prokaryotic genome. As well, this CMR database make easier by making availability of all the organisms information, it also giving analysis of comparison between the genomes of the different organisms. CMR also contains genome tools, searches for genes, genomes, sequence; comparative tool which for comparison of multiple genomes. The tools could be more useful because it's providing graphical displays of genomes, biochemical pathways of genome as well. The data were stored in a database called Omniome database. There are more than 20 tables in the Omniome database scheme. From there, Taxon table downloaded and used for missing data analysis in this research. In particular Taxon table has been selected in this research since it has missing values in it.

And the second step is verifying presence of the missing values in the Taxon data set. Taxan data set were viewed in Microsoft Office Excel and each column and row of the table checked for missing values appearances. Statistic analysis was done on percentage of missing data in taxon table. Step three is to prepare the dataset for FDs discovery. Datasets must be separated into sub-tables and followed by reduction of missing data columns and rows. Here the Taxon main table is spliced up into seven categories since it has 12 attributes. For ease of reference, attributes were presented as alphabets as shown in Table 2.

Table 2. List of attributes in Taxon

Attributes	Represented by
U_id	A
Taxon_id	B
Kingdom	C
Genus	D
Species	E
Strain	F
Intermediate_rank_1	G
Intermediate_rank_2	H
Intermediate_rank_3	I
Intermediate_rank_4	J
Intermediate_rank_5	K
Intermediate_rank_6	L

The first five column (attributes) A to E is remain unchanged for all the seven tables while the balance seven attributes were spliced into seven table as follows: AE_F, AE_G, AE_H, AE_I, AE_J, AE_K and AE_L. The attributes A, B C, D and E are never changed because it has been found that those attributes does not have any missing values. Hence these attributes remain the same to analyse the presence of FDs for the missing value analysis for the other attributes. And the schemas of the sub-tables from Taxon are as follows:

i. AE_F = (A, B, C, D, E)

ii. AE_G = (A, B, C, D, G)

iii. AE_H = (A, B, C, D, H)

iv. AE_I = (A, B, C, D, I)

v. AE_J = (A, B, C, D, J)

vi. AE_K = (A, B, C, D, K)

vii. AE_L = (A, B, C, D, L)

Fourth step is to genere test table by data cleaning the taxon table into sub-table. After data cleaning process, the data sets saved as comma separated values file to be used as input in TANE. Sample input table data set is shown in Appendix A. Followed by step five, TANE algorithm is used to detect the FDs in the test table which are generated before. TANE algorithm was developed by Huhtala and colleageus. (Huhtala, et al., 1999). Step six will be carried out experiment to obseve the missing values in test table and the original complete table. Results from the experiment is used further for discussion of proxies for space requirement analysis and aslo to recommend the characteristics of proxies for missing values prediction.

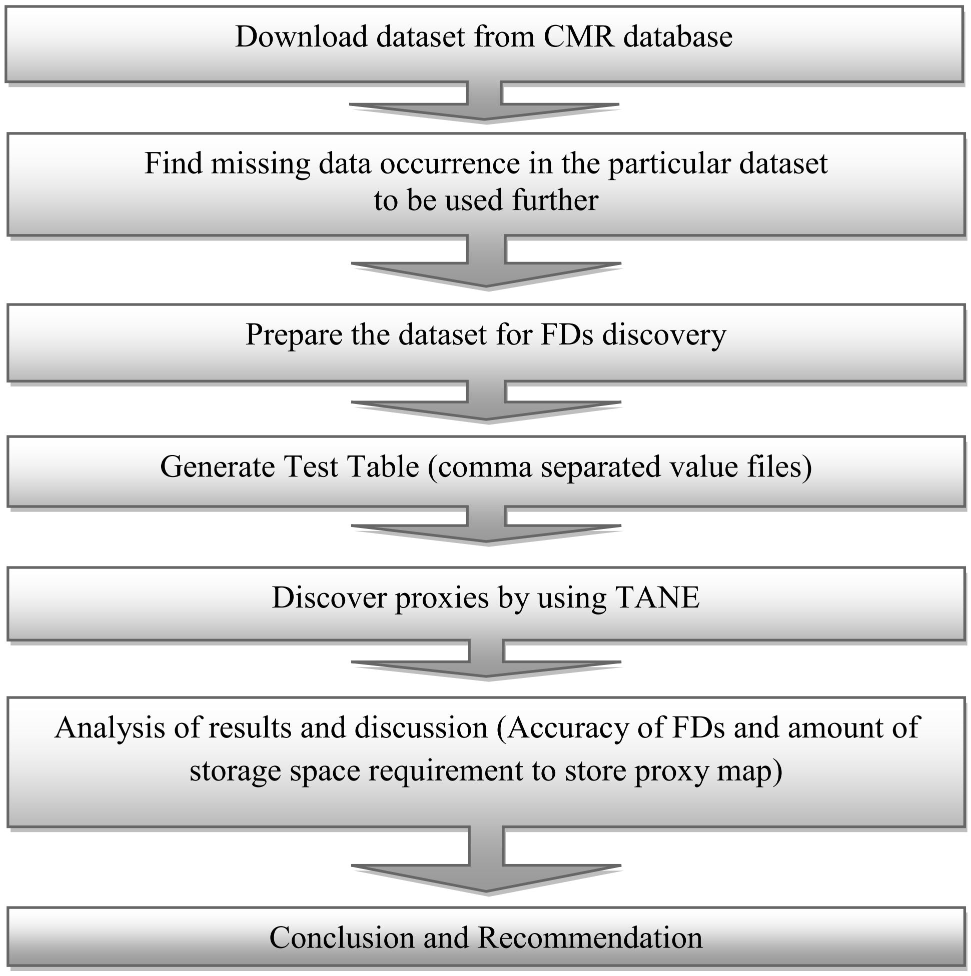

Download dataset from CMR database
Find missing data occurrence in the particular dataset to be used further
Prepare the dataset for FDs discovery
Generate Test Table (comma separated value files)
Discover proxies by using TANE
Analysis of results and discussion (Accuracy of FDs and amount of storage space requirement to store proxy map)
Conclusion and Recommendation

Figure 7. Flow chart of overall methodology

3.3 Data source of Microbial Genomics data sets

As mentioned earlier in the methodology part, CMR database were chosen to get the sample data set. Specifically microbial genomics database chose because most of the diseases caused by the microbes called as pathogen. Hence there are many microbes has been identified by the scientists in their research. Though, it is not properly managed to be used in future; for example to obtain a vaccine or drug to cure a particular disease. Data incompleteness may arise from this improper management of the database. Therefore, sample data set were taken from CMR to analyse the presence of FDs which can be used in missing values prediction. Microbial genomics is the study of genome of microbes where it determines the whole DNA sequence of the microbes. Along with this, the genes will determine the functions and pathways of the microbes.

Chromosomes are made of nucleotides sequence which is called as gene that encode specific product such RNA or protein molecule. Basically, gene contains biological information for instance roles in cellular pathways and the location on a chromosome for each specific species. The main characteristic of the genome is the taxonomic classification (phylogenetic) including organism's domains, phylum, class, order, family, genus, species and strain. The reactions pathways are involving compounds such as reactants, and enzymes to catalyse the reaction.

Fundamentally, genomics is the study of the organisation of genome's molecule, its content and the gene that they encode. It is divided into three categories such as structural genomics, functional genomics, and comparative genomics. Structural genomics is the study of the physical structure of an organism's genomes. The major objective is to resolve and explore the DNA sequence of the genome. Functional genomics is the analysis to verify the genomes functions. The function is determined by the proteins that encode the genome. The third category is comparative genomics to analyse the differences and similarities in the genomes from different organisms. This analysis will help the researchers to identify the conserved region in a particular genome and differentiate function and regulation patterns.

DNA sequencing can be done using Sanger method. Whole-genome shotgun sequencing is one of the simplest ways to analyse the microbial genomes. Here, fragments of gene that has been produced were sequenced individually and computer is used to align them to get a complete genome. The whole-genome shotgun is divided into four stages as follows: library construction, random sequencing, fragment alignment and gap closure, and editing.

Researchers have complete sequencing of many bacterial genomes and make comparison between one another as well. This output will help us in identification and determination of structure of genome, microbial physiology, phylogeny, and also the pathogen that cause a disease. Identification of those criteria directly will help in producing new vaccines and drugs for the disease treatment. At last, the function of genome can be identified by annotation, where the DNA chips were used to study the mRNA synthesis and the organism's protein content. The extensive contribution of the genomes comparison is the understanding of prokaryotic evolution and assists to assume the genes that are responsible for different cellular processes.

3.2.1 Description of the semantics of Taxon table attributes

The taxon table holds the information about each genome filled into the omniome database. Table db_data and taxon_link has linkage of genome information with taxon table. Taxon table's data was taken from NCBI. Taxon table has 14 attributes and 723 rows of tuples. The attributes of Taxon table are:

```
Taxon = (u_id, taxon_id, kingdom, genus, species, comment, strain,
intermediate_rank_1,    intermediate_rank_2,    intermediate_rank_3,
intermediate_rank_4,    intermediate_rank_5,    intermediate_rank_6,
short_name)
```

Fundamentally, kingdom in biology is known as taxonomic rank which is the top rank or three-domain system. Kingdoms are divided into three main domains such as bacteria, archaea, and eukarya. Classification level of the organisms during 1970's was increase due to importance of molecular level comparisons of the genes is the key factor besides genetic similarity and the physical appearance and behaviour.

In biology, genus (plural: genera) is the low-level taxonomic rank which is used to classify the living and fossil organisms. Biodiversity studies especially fossil studies of a species cannot always be identified and genera and families basically have lengthy ranges than species which is determined by using genera and higher taxonomic level for instance families.

Essentially species is a group of organisms that is capable of interbreeding and reproducing good offspring. Normally species that shared common ancestors were placed in one genus based on some similarities. The similarities are comparison of physical attributes, for example their DNA sequences.

Strain also known as low-level taxonomic rank used in some of the biological field. A strain is a genetic variant or a subtype of a micro-organism for instance virus, bacterium or fungus. “Flu strain” is an example of the influenza or “flu” virus. Intermediate ranking is about subdivision of the kingdom to get more specific gene for further use such as to produce vaccine.

3.2.2 Observation of missing values in Taxon table

Out of 14 attributes 1 attribute (column: comment) is completely empty. There are total 9399 tuples in the Taxon table. And 875 rows of tuples were missing in this table. Statistics shows that about 9.31% data were missing. This missing data may cause some problem during further analysis. For example loss of the specific gene or strain may cause inconsistencies in production of vaccine. In term of phylogenetic analysis also produced invariance results for organism classification. Statistics of the missing data is calculated based on the characteristics of Taxon table as shown in Table 3.

Table 3. Statistics of missing data in Taxon table

Characteristics	**Total**	**Percentage of missing values (%)**
Attributes used	13	-
Missing attribute (fully empty)	1	-
Tuples (include with missing values)	Total number of cells = number of tuple(s) x number of attribute(s) Total number of cells = 723 x 13 = 9399	-
Missing values	875	$\frac{875}{9399}$ x 100 =9.31

3.4 TANE Algorithm for discovery of FDs

TANE is an available algorithm where presented by Huhtala et al. (1999) to discover FDs that is not limited to small amount of datasets even for large number of datasets. This algorithm is divided into few parts such as TANE main algorithm, generating levels, computing dependencies, pruning the lattice, computing partitions, and approximate dependencies. Fundamentally, TANE is partition based algorithm where set of rows are partitioning their attributes which makes discovery of FDs faster and efficient. Besides FDs, the partition also used to detect the AFDs with efficiently. “To find all valid minimal non-trivial dependencies, TANE searches the set containment lattice in a level wise manner". (Huhtala, et al., 1999). The further details of the algorithm were explained in section 3.3.1. Advantages of TANE are:

1. Fast even for a large number of tuples.
2. Not only FDs, discovery of approximate functional dependencies easy and efficient and the erroneous or exceptional rows can be identified easily
3. Space can be pruned effectively and how the partitions and dependencies can be computed efficiently.

The following are the steps involved in TANE algorithm process:

i. The installation of the data set must be done: For example the file (data set) name is AE_F.orig. Save this file in "original" folder.

ii. Than open "description" folder and edit/create AE_F.dsc file to the variables.

iii. Create new data set by using select.perl command

```
% cd descriptions

%../bin/select.perl AE_F.dsc
```

(to produce AE_F.dat file in description folder)

iv. To get the output from the TANE: we use the following command

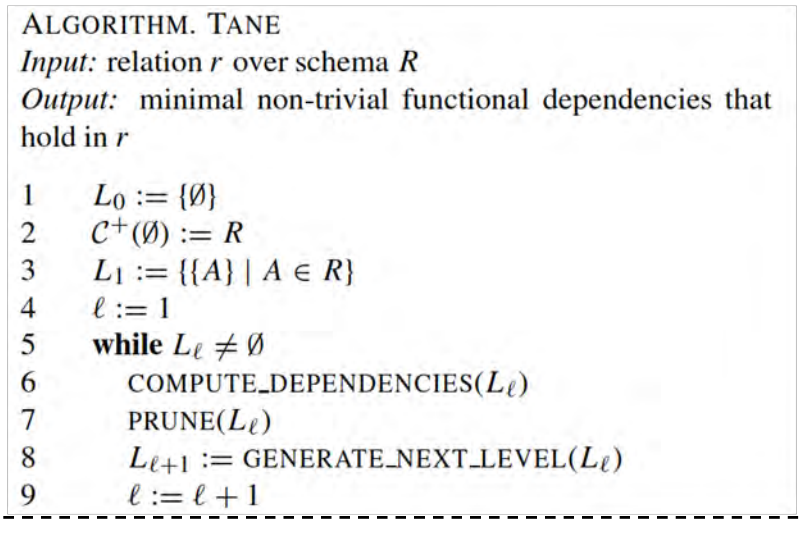
```
%bin/taneg3 <# of attributes> <# of records> <# of attributes> data/AE_F.dat 0.1 &>

TaxonAF01.txt
```

(the output file is in .txt format)

3.3.1 TANE Algorithm categories

Huhtala et al., (1999) developed TANE algorithm for prediction of FDs. It is divided into six subparts such as main TANE algorithm, generating levels, computing dependencies, pruning the lattice, computing partitions, and approximate dependencies.

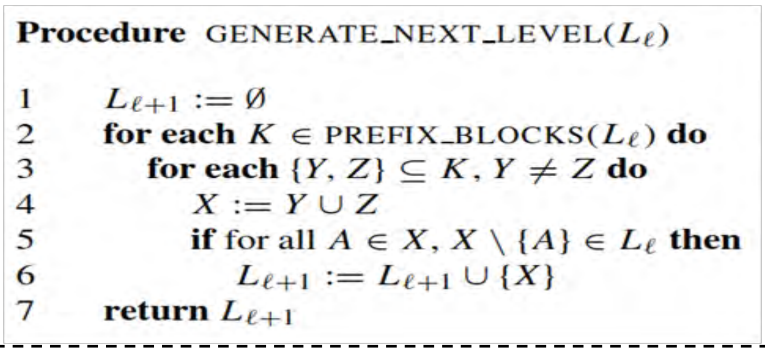
Figure 8shows the main TANE algorithm's procedure. The computation in TANE will begins with $L_1 = \{\{A\} \mid A \in R\}$ and work out L_2 from L_1 and L_3 from L_2. The step 6 COMPUTE_DEPENDENCIES (L_ℓ) is to find the least dependencies with the left hand side in $L_{\ell-1}$. Next the PRUNE (L_ℓ) will search for the space. And then, the last step GENERATE_NEXT_LEVEL (L_ℓ) produces next level from the current level. (Huhtala, et al., 1999).



```
ALGORITHM. TANE
Input: relation $r$ over schema $R$
Output: minimal non-trivial functional dependencies that
hold in $r$

1   $L_0 := \{\emptyset\}$
2   $C^+(\emptyset) := R$
3   $L_1 := \{\{A\} \mid A \in R\}$
4   $\ell := 1$
5   while $L_\ell \neq \emptyset$
6       COMPUTE_DEPENDENCIES($L_\ell$)
7       PRUNE($L_\ell$)
8       $L_{\ell+1}$ := GENERATE_NEXT_LEVEL($L_\ell$)
9       $\ell := \ell + 1$
```

Figure 8. TANE main algorithm (Adapted from Huhtala et al., 1999)

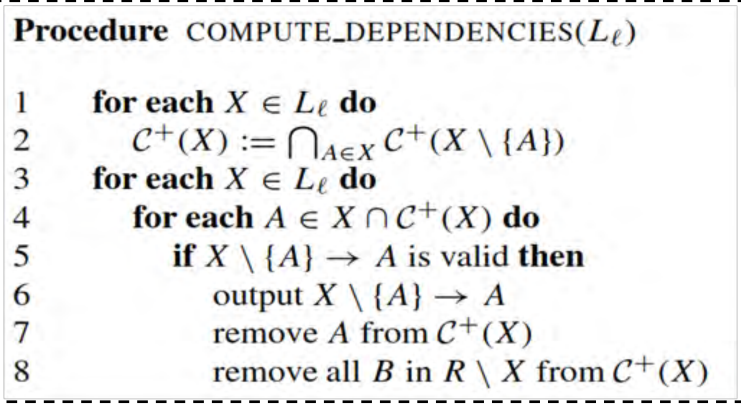
Figure 9 shows the subsequent algorithm from the main TANE algorithm; the generating level algorithm. Here the GENERATE_NEXT_LEVEL is computing the $L_{\ell+1}$ from L_ℓ. PREFIX_BLOCKS (L_ℓ) is to sort the list of attributes with same prefix block. (Huhtala, et al., 1999).



```
Procedure GENERATE_NEXT_LEVEL($L_\ell$)

$L_{\ell+1} := \emptyset$
for each $K \in$ PREFIX_BLOCKS($L_\ell$) do
    for each $\{Y, Z\} \subseteq K, Y \neq Z$ do
        $X := Y \cup Z$
        if for all $A \in X, X \setminus \{A\} \in L_\ell$ then
            $L_{\ell+1} := L_{\ell+1} \cup \{X\}$
return $L_{\ell+1}$
```

Figure 9. Generating levels algorithm (Adapted from Huhtala et al., 1999)

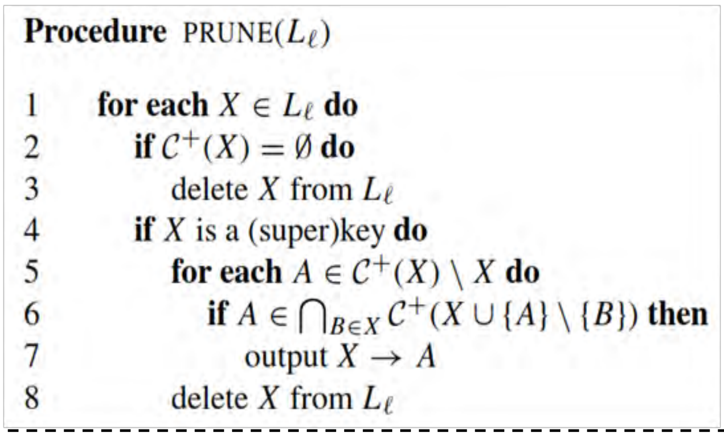
After generating levels algorithm, COMPUTING_DEPENDENCIES is the next step of TANE as in Figure 10. The output is to obtain minimal dependencies from this procedure. (Huhtala, et al., 1999).



```
Procedure COMPUTE_DEPENDENCIES($L_\ell$)

for each $X \in L_\ell$ do
    $\mathcal{C}^+(X) := \bigcap_{A \in X} \mathcal{C}^+(X \setminus \{A\})$
for each $X \in L_\ell$ do
    for each $A \in X \cap \mathcal{C}^+(X)$ do
        if $X \setminus \{A\} \rightarrow A$ is valid then
            output $X \setminus \{A\} \rightarrow A$
            remove $A$ from $\mathcal{C}^+(X)$
            remove all $B$ in $R \setminus X$ from $\mathcal{C}^+(X)$
```

Figure 10. Computing dependencies algorithm (Adapted from Huhtala et al., 1999)

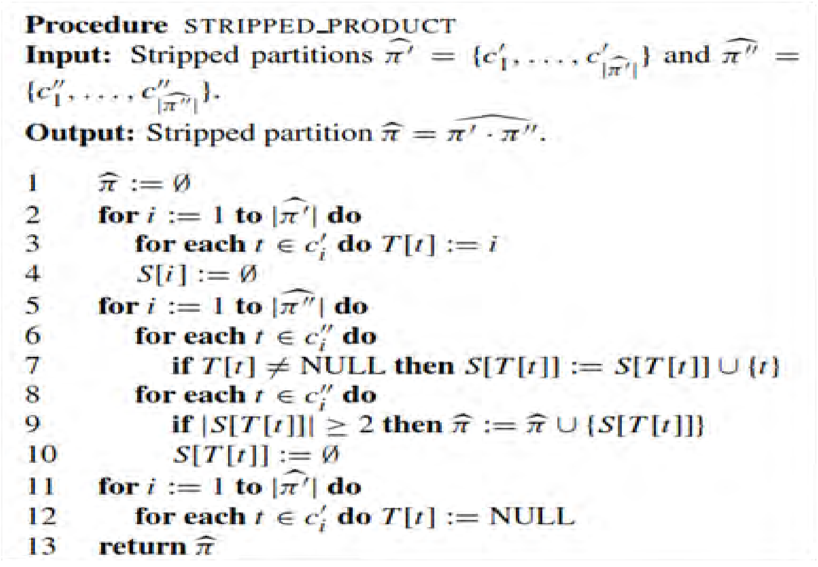
Procedure of pruning in TANE algorithm was given in Figure 11. Essentially, this pruning procedure contains two parts; Rhs candidates pruning and key pruning. This pruning step is used to detect the dependencies without missing it. (Huhtala, et al., 1999).



```
Procedure PRUNE($L_\ell$)

for each $X \in L_\ell$ do
    if $C^+(X) = \emptyset$ do
        delete $X$ from $L_\ell$
    if $X$ is a (super)key do
        for each $A \in C^+(X) \setminus X$ do
            if $A \in \bigcap_{B \in X} C^+(X \cup \{A\} \setminus \{B\})$ then
                output $X \rightarrow A$
        delete $X$ from $L_\ell$
```

Figure 11. Pruning the lattice algorithm (Adapted from Huhtala et al., 1999)

The *e* value in TANE algorithm is calculated by stripped partitions procedure as in Figure 12. An initialisation of table T to all NULL made through this procedure as an assumption. The same table can be utilised repeatedly without re-initialisation because before out the procedure resets to all NULL.

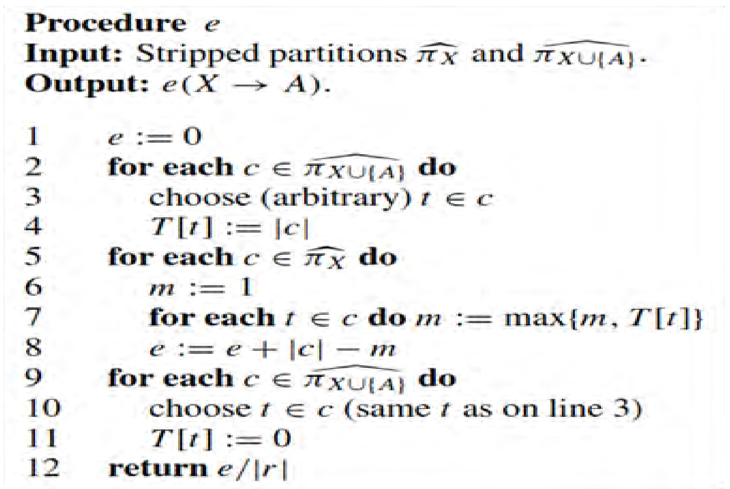


```
Procedure STRIPPED_PRODUCT
Input: Stripped partitions $\widehat{\pi}' = \{c'_1, \ldots, c'_{|\widehat{\pi}'|}\}$ and $\widehat{\pi}'' = \{c''_1, \ldots, c''_{|\widehat{\pi}''|}\}$.
Output: Stripped partition $\widehat{\pi} = \widehat{\pi' \cdot \pi''}$.

$\widehat{\pi} := \emptyset$
for $i := 1$ to $|\widehat{\pi}'|$ do
    for each $t \in c'_i$ do $T[t] := i$
    $S[i] := \emptyset$
for $i := 1$ to $|\widehat{\pi}''|$ do
    for each $t \in c''_i$ do
        if $T[t] \neq$ NULL then $S[T[t]] := S[T[t]] \cup \{t\}$
    for each $t \in c''_i$ do
        if $|S[T[t]]| \geq 2$ then $\widehat{\pi} := \widehat{\pi} \cup \{S[T[t]]\}$
        $S[T[t]] := \emptyset$
for $i := 1$ to $|\widehat{\pi}'|$ do
    for each $t \in c'_i$ do $T[t] :=$ NULL
return $\widehat{\pi}$
```

Figure 12. Computing partitions algorithm (Adapted from Huhtala et al., 1999)

The Figure 13 shows the approximate dependencies procedure. This procedure is obtained from modification of TANE algorithm to find all approximate dependencies.

Beside this, it not only use to find the minimal approximate dependencies but with smaller error.



Procedure e
Input: Stripped partitions $\widehat{\pi_X}$ and $\widehat{\pi_{X\cup\{A\}}}$.
Output: $e(X \rightarrow A)$.

```
1    $e := 0$
2    for each $c \in \widehat{\pi_{X\cup\{A\}}}$ do
3        choose (arbitrary) $t \in c$
4        $T[t] := |c|$
5    for each $c \in \widehat{\pi_X}$ do
6        $m := 1$
7        for each $t \in c$ do $m := \max\{m, T[t]\}$
8        $e := e + |c| - m$
9    for each $c \in \widehat{\pi_{X\cup\{A\}}}$ do
10       choose $t \in c$ (same $t$ as on line 3)
11       $T[t] := 0$
12   return $e/|r|$
```

Figure 13. Approximate Dependencies algorithm (Adapted from Huhtala et al., 1999)

3.5 The method in preparing analysis of space requirement

Proxy based approach is designed for storage space optimisation. In this research, we adopt proxy-based approach to study the requirement to predict missing values and types of proxies which are contribute in save the space. Basically, the candidates proxies were identified from the output produced from TANE algorithm, where the G3 errors are very low or zero. If attribute shows very low or zero error, than it can be replaced to droppable attribute to predict the missing values.

Pivot table can be used to count the number of instances in taxon proxy tables in terms of space saving. Pivot table make it easy to arrange and summarise the complicated data and drill down on details. Hence using this pivot table function in MS Excel, we calculate the number of relationship between one tuple to another either one to one or one to many.

3.5.1 Proxy based approach for space optimisation

The rising of data volumes in many application domains, makes raise a problem to maintain such large data storages. Though, the storage space is reducible if the space of storage was optimised. These space optimisations not only contribute to save the space, but also decreasing the carbon footprint and the cost of operation. Beside this, it also optimises query response time. Additionally, this space optimisation might make easy the job of administering which are basically requires new infrastructure, utilities like power and cooling increase, extra floor space and extra staff. (Emran, et al., 2012).

Therefore, Emran, Abdullah, and Isa (2012) produced an approach called Proxy-based approach which can generate space optimisation through modification of database schema. This can be done by deleting the attributes from the particular schema. The term 'proxies' were used by the researchers, is to replace the attribute with another attribute in the schema. The functional dependency relationship is used to recognise the proxies among the attributes in a relational table.

Basically, the space saving is obtained through some modification in the schema by dropping some of the attributes. And then, the total saved space is approximately verified by the number of attributes has been dropped and the tuples number in the table remain. However, the droppable attribute and the proxy must have relationship in terms of missing data. Hence, the functional dependency relationship is obtained between the attributes in the relational tables. Proxies for the delete able attributes been found through discover of the relations among attributes in the tables where there is presence of FD.

This proxy-based technique apply algorithm which will get back the removed overlaps from the meta-data, when the query is submitted against the compressed tables.

For instance, *a* and *b* are the droppable attribute. A proxy map consists of the following mappings:

$$a \rightarrow \{1,2,3,4\},$$
$$b \rightarrow \{5,6,7,8\},$$

where the numbers are the proxy values and the arrow shows relationship mapping. From here, the researchers have identified two types of proxy maps as follows:

i. A pure relational table:

   This structure shows each value in a droppable attribute is matched to exactly one value of the proxy. The schema structure of the table is: <droppableAttr, proxy>. (Table 4)

ii. A multi-valued table:

   In this table, each value of droppable attribute is matched to a set of proxy values. The schema structure of the table is: <droppableAttr, proxy>. (Table 5)

Table 4. A Proxy map in pure relational table (Emran, Abdullah, and Isa 2012)

A	B
a	1
a	2
a	3
a	4
b	5
b	6
b	7
b	8

Table 5. A Proxy map in a multi-valued table (Emran, Abdullah, and Isa 2012)

A	B
a	1,2,3,4
b	5,6,7,8

The example also shows that, the size of proxy map in the multi-valued table is smaller than the proxy map in pure relational table. As a result from the example, the storage space can be saved by minimizing the proxy map in a multi-valued table structure as shown in table 2.2. In the example above, the multi-valued table saves of 6 instances.

3.6 Conclusions

To conclude, the method has been applied according to research methodology described above sections. The steps must be followed correctly to obtain precise results to be analysed later. Results obtained from TANE algorithm saved as .txt file analysed and discussed in the next chapter. The results were analysed according to accuracy or low G3 errors.

CHAPTER 4

RESULTS

4.1 Background

This chapter presents the results obtained from the steps performed in the methodology which is described in the previous chapter. The input table used for this research is in comma separated values file than saved as .txt file. The output also saved in the same format to make easy to view the results. There are two types of results have been obtained from the analysis:

i. Discovery of candidate proxies and its G3 values
ii. Amount of space required to store proxy information

4.2 Proxy discovery from Taxon sub-tables

The TANE algorithm produced the output which can be viewed in notepad or WordPad. Basically, the TANE algorithm has ten ranges which is start from 0.10 to 1.00. Since we have total seven tables, each table can produce ten outputs. Figure 14 – Figure 23 show the raw results for table AE_F produced by TANE algorithm and the other results were shown in the Appendix B. Then, these results are analysed according to G3 ranges as described in this sections above. Section 4.2.1 to 4.2.7 shows the summary of the output from TANE. The highlighted (yellow colour) attributes are the FD-based proxies for each proxy attribute that being analysed in each section. They are F, G, H, I, J, K, and L. This identification is basically done according to the G3 errors values produced by the TANE algorithm.

```
Level == 1  #candidates == 6      avg.elements == 8     (1710/208)
Level == 2  #candidates == 6      avg.elements == 3     (397/118)
Level == 3  #candidates == 1      avg.elements == 2     (31/14)

========================================================================
Parameters (approximate dependencies):
No. of tuples           == 720
No. of attributes       == 6
Stop level              == 6
Data                    == data/TaxonAF.dat
Percentage of all tuples == 0.10 %
==> G3 threshold        == 72 max. rows removed
========================================================================

-> 3   (50 / 0.07)
1 -> 2   (key)
1 -> 3   (key)
1 -> 4   (key)
1 -> 5   (key)
1 -> 6   (key)
2 -> 1   (approximate key: 18 / 0.03)
2 -> 3   (approximate key: 18 / 0.03)
2 -> 4   (approximate key: 18 / 0.03)
2 -> 5   (approximate key: 18 / 0.03)
2 -> 6   (approximate key: 18 / 0.03)
6 -> 1   (approximate key: 4 / 0.01)
6 -> 2   (approximate key: 4 / 0.01)
6 -> 3   (approximate key: 4 / 0.01)
6 -> 4   (approximate key: 4 / 0.01)
6 -> 5   (approximate key: 4 / 0.01)

========================================================================
total_no_of_candidates == 13
prune_key            == 9
prune_key_sub        == 1
prune_key_second     == 0
prune_rhs            == 1
prune_rhs_sub        == 0
prune_rhs_second     == 3
========================================================================
```

Figure 14. Output of TANE algorithms for table AE_F on 0.10 G3 range

```
Level == 1  #candidates == 6      avg.elements == 8     (1710/208)
Level == 2  #candidates == 6      avg.elements == 3     (397/118)
Level == 3  #candidates == 1      avg.elements == 2     (31/14)

=====================================================================
Parameters (approximate dependencies):
No. of tuples           == 720
No. of attributes       == 6
Stop level              == 6
Data                    == data/TaxonAF.dat
Percentage of all tuples == 0.20 %
==> G3 threshold        == 144 max. rows removed
=====================================================================

-> 3   (50 / 0.07)
1 -> 2   (key)
1 -> 3   (key)
1 -> 4   (key)
1 -> 5   (key)
1 -> 6   (key)
2 -> 1   (approximate key: 18 / 0.03)
2 -> 3   (approximate key: 18 / 0.03)
2 -> 4   (approximate key: 18 / 0.03)
2 -> 5   (approximate key: 18 / 0.03)
2 -> 6   (approximate key: 18 / 0.03)
6 -> 1   (approximate key: 4 / 0.01)
6 -> 2   (approximate key: 4 / 0.01)
6 -> 3   (approximate key: 4 / 0.01)
6 -> 4   (approximate key: 4 / 0.01)
6 -> 5   (approximate key: 4 / 0.01)
5 -> 4   (81 / 0.11)

=====================================================================
total_no_of_candidates == 13
prune_key            == 9
prune_key_sub        == 1
prune_key_second     == 0
prune_rhs            == 2
prune_rhs_sub        == 0
prune_rhs_second     == 3
=====================================================================
```

Figure 15. Output of TANE algorithms for table AE_F on 0.20 G3 range

```
Level == 1  #candidates == 6      avg.elements == 8     (1710/208)
Level == 2  #candidates == 6      avg.elements == 3     (397/118)
Level == 3  #candidates == 1      avg.elements == 2     (31/14)

===========================================================================
Parameters (approximate dependencies):
No. of tuples           == 720
No. of attributes        == 6
Stop level              == 6
Data                  == data/TaxonAF.dat
Percentage of all tuples == 0.30 %
==> G3 threshold         == 216 max. rows removed
===========================================================================

-> 3   (50 / 0.07)
1 -> 2   (key)
1 -> 3   (key)
1 -> 4   (key)
1 -> 5   (key)
1 -> 6   (key)
2 -> 1   (approximate key: 18 / 0.03)
2 -> 3   (approximate key: 18 / 0.03)
2 -> 4   (approximate key: 18 / 0.03)
2 -> 5   (approximate key: 18 / 0.03)
2 -> 6   (approximate key: 18 / 0.03)
6 -> 1   (approximate key: 4 / 0.01)
6 -> 2   (approximate key: 4 / 0.01)
6 -> 3   (approximate key: 4 / 0.01)
6 -> 4   (approximate key: 4 / 0.01)
6 -> 5   (approximate key: 4 / 0.01)
5 -> 4   (81 / 0.11)

===========================================================================
total_no_of_candidates == 13
prune_key             == 9
prune_key_sub          == 1
prune_key_second       == 0
prune_rhs             == 2
prune_rhs_sub          == 0
prune_rhs_second       == 3
===========================================================================
```

Figure 16. Output of TANE algorithms for table AE_F on 0.30 G3 range

```
Level == 1  #candidates == 6      avg.elements == 8     (1710/208)
Level == 2  #candidates == 6      avg.elements == 3     (397/118)
Level == 3  #candidates == 1      avg.elements == 2     (31/14)

=================================================================
Parameters (approximate dependencies):
No. of tuples          == 720
No. of attributes      == 6
Stop level             == 6
Data                   == data/TaxonAF.dat
Percentage of all tuples == 0.40 %
==> G3 threshold       == 288 max. rows removed
=================================================================

-> 3   (50 / 0.07)
1 -> 2   (key)
1 -> 3   (key)
1 -> 4   (key)
1 -> 5   (key)
1 -> 6   (key)
2 -> 1   (approximate key: 18 / 0.03)
2 -> 3   (approximate key: 18 / 0.03)
2 -> 4   (approximate key: 18 / 0.03)
2 -> 5   (approximate key: 18 / 0.03)
2 -> 6   (approximate key: 18 / 0.03)
6 -> 1   (approximate key: 4 / 0.01)
6 -> 2   (approximate key: 4 / 0.01)
6 -> 3   (approximate key: 4 / 0.01)
6 -> 4   (approximate key: 4 / 0.01)
6 -> 5   (approximate key: 4 / 0.01)
5 -> 4   (81 / 0.11)

=================================================================
total_no_of_candidates == 13
prune_key            == 10
prune_key_sub        == 1
prune_key_second     == 0
prune_rhs            == 2
prune_rhs_sub        == 0
prune_rhs_second     == 3
=================================================================
```

Figure 17. Output of TANE algorithms for table AE_F on 0.40 G3 range

```
Level == 1  #candidates == 6       avg.elements == 8     (1710/208)
Level == 2  #candidates == 6       avg.elements == 3     (397/118)

=====================================================================
Parameters (approximate dependencies):
No. of tuples           == 720
No. of attributes        == 6
Stop level              == 6
Data                    == data/TaxonAF.dat
Percentage of all tuples == 0.50 %
==> G3 threshold         == 360 max. rows removed
=====================================================================


-> 3   (50 / 0.07)
1 -> 2   (key)
1 -> 3   (key)
1 -> 4   (key)
1 -> 5   (key)
1 -> 6   (key)
2 -> 1   (approximate key: 18 / 0.03)
2 -> 3   (approximate key: 18 / 0.03)
2 -> 4   (approximate key: 18 / 0.03)
2 -> 5   (approximate key: 18 / 0.03)
2 -> 6   (approximate key: 18 / 0.03)
5 -> 1   (approximate key: 300 / 0.42)
5 -> 2   (approximate key: 300 / 0.42)
5 -> 3   (approximate key: 300 / 0.42)
5 -> 4   (approximate key: 300 / 0.42)
5 -> 6   (approximate key: 300 / 0.42)
6 -> 1   (approximate key: 4 / 0.01)
6 -> 2   (approximate key: 4 / 0.01)
6 -> 3   (approximate key: 4 / 0.01)
6 -> 4   (approximate key: 4 / 0.01)
6 -> 5   (approximate key: 4 / 0.01)
4 -> 5   (322 / 0.45)

=====================================================================
total_no_of_candidates == 12
prune_key              == 10
prune_key_sub          == 0
prune_key_second       == 0
prune_rhs              == 3
prune_rhs_sub          == 0
prune_rhs_second       == 3
=====================================================================
```

Figure 18. Output of TANE algorithms for table AE_F on 0.50 G3 range

```
Level == 1  #candidates == 6      avg.elements == 8     (1710/208)
Level == 2  #candidates == 6      avg.elements == 3     (397/118)

==================================================================
Parameters (approximate dependencies):
No. of tuples           == 720
No. of attributes        == 6
Stop level              == 6
Data                 == data/TaxonAF.dat
Percentage of all tuples == 0.60 %
==> G3 threshold        == 432 max. rows removed
==================================================================

-> 3   (50 / 0.07)
1 -> 2   (key)
1 -> 3   (key)
1 -> 4   (key)
1 -> 5   (key)
1 -> 6   (key)
2 -> 1   (approximate key: 18 / 0.03)
2 -> 3   (approximate key: 18 / 0.03)
2 -> 4   (approximate key: 18 / 0.03)
2 -> 5   (approximate key: 18 / 0.03)
2 -> 6   (approximate key: 18 / 0.03)
5 -> 1   (approximate key: 300 / 0.42)
5 -> 2   (approximate key: 300 / 0.42)
5 -> 3   (approximate key: 300 / 0.42)
5 -> 4   (approximate key: 300 / 0.42)
5 -> 6   (approximate key: 300 / 0.42)
6 -> 1   (approximate key: 4 / 0.01)
6 -> 2   (approximate key: 4 / 0.01)
6 -> 3   (approximate key: 4 / 0.01)
6 -> 4   (approximate key: 4 / 0.01)
6 -> 5   (approximate key: 4 / 0.01)
4 -> 5   (322 / 0.45)

==================================================================
total_no_of_candidates == 12
prune_key            == 10
prune_key_sub         == 0
prune_key_second      == 0
prune_rhs            == 3
prune_rhs_sub         == 0
prune_rhs_second      == 3
==================================================================
```

Figure 19. Output of TANE algorithms for table AE_F on 0.60 G3 range

```
Level == 1  #candidates == 6      avg.elements == 8     (1710/208)
Level == 2  #candidates == 6      avg.elements == 3     (397/118)

====================================================================
Parameters (approximate dependencies):
No. of tuples           == 720
No. of attributes        == 6
Stop level              == 6
Data                  == data/TaxonAF.dat
Percentage of all tuples == 0.70 %
==> G3 threshold         == 503 max. rows removed
====================================================================

-> 3   (50 / 0.07)
1 -> 2   (key)
1 -> 3   (key)
1 -> 4   (key)
1 -> 5   (key)
1 -> 6   (key)
2 -> 1   (approximate key: 18 / 0.03)
2 -> 3   (approximate key: 18 / 0.03)
2 -> 4   (approximate key: 18 / 0.03)
2 -> 5   (approximate key: 18 / 0.03)
2 -> 6   (approximate key: 18 / 0.03)
4 -> 1   (approximate key: 463 / 0.64)
4 -> 2   (approximate key: 463 / 0.64)
4 -> 3   (approximate key: 463 / 0.64)
4 -> 5   (approximate key: 463 / 0.64)
4 -> 6   (approximate key: 463 / 0.64)
5 -> 1   (approximate key: 300 / 0.42)
5 -> 2   (approximate key: 300 / 0.42)
5 -> 3   (approximate key: 300 / 0.42)
5 -> 4   (approximate key: 300 / 0.42)
5 -> 6   (approximate key: 300 / 0.42)
6 -> 1   (approximate key: 4 / 0.01)
6 -> 2   (approximate key: 4 / 0.01)
6 -> 3   (approximate key: 4 / 0.01)
6 -> 4   (approximate key: 4 / 0.01)
6 -> 5   (approximate key: 4 / 0.01)

====================================================================
total_no_of_candidates == 12
prune_key             == 11
prune_key_sub         == 0
prune_key_second      == 0
prune_rhs             == 5
prune_rhs_sub         == 0
prune_rhs_second      == 1
=============================================================
```

Figure 20. Output of TANE algorithms for table AE_F on 0.70 G3 range

```
Level == 1  #candidates == 6      avg.elements == 8     (1710/208)
Level == 2  #candidates == 6      avg.elements == 3     (397/118)

==============================================================================
Parameters (approximate dependencies):
No. of tuples           == 720
No. of attributes       == 6
Stop level              == 6
Data                    == data/TaxonAF.dat
Percentage of all tuples == 0.80 %
==> G3 threshold        == 576 max. rows removed
==============================================================================


-> 3   (50 / 0.07)
1 -> 2   (key)
1 -> 3   (key)
1 -> 4   (key)
1 -> 5   (key)
1 -> 6   (key)
2 -> 1   (approximate key: 18 / 0.03)
2 -> 3   (approximate key: 18 / 0.03)
2 -> 4   (approximate key: 18 / 0.03)
2 -> 5   (approximate key: 18 / 0.03)
2 -> 6   (approximate key: 18 / 0.03)
4 -> 1   (approximate key: 463 / 0.64)
4 -> 2   (approximate key: 463 / 0.64)
4 -> 3   (approximate key: 463 / 0.64)
4 -> 5   (approximate key: 463 / 0.64)
4 -> 6   (approximate key: 463 / 0.64)
5 -> 1   (approximate key: 300 / 0.42)
5 -> 2   (approximate key: 300 / 0.42)
5 -> 3   (approximate key: 300 / 0.42)
5 -> 4   (approximate key: 300 / 0.42)
5 -> 6   (approximate key: 300 / 0.42)
6 -> 1   (approximate key: 4 / 0.01)
6 -> 2   (approximate key: 4 / 0.01)
6 -> 3   (approximate key: 4 / 0.01)
6 -> 4   (approximate key: 4 / 0.01)
6 -> 5   (approximate key: 4 / 0.01)

==============================================================================
total_no_of_candidates == 12
prune_key              == 11
prune_key_sub          == 0
prune_key_second       == 0
prune_rhs              == 5
prune_rhs_sub          == 0
prune_rhs_second       == 1
==========================================================================
```

Figure 21. Output of TANE algorithms for table AE_F on 0.80 G3 range

```
Level == 1  #candidates == 6      avg.elements == 8     (1710/208)
Level == 2  #candidates == 6      avg.elements == 3     (397/118)

=====================================================================
Parameters (approximate dependencies):
No. of tuples           == 720
No. of attributes        == 6
Stop level              == 6
Data                  == data/TaxonAF.dat
Percentage of all tuples == 0.90 %
==> G3 threshold         == 648 max. rows removed
=====================================================================

-> 3   (50 / 0.07)
1 -> 2   (key)
1 -> 3   (key)
1 -> 4   (key)
1 -> 5   (key)
1 -> 6   (key)
2 -> 1   (approximate key: 18 / 0.03)
2 -> 3   (approximate key: 18 / 0.03)
2 -> 4   (approximate key: 18 / 0.03)
2 -> 5   (approximate key: 18 / 0.03)
2 -> 6   (approximate key: 18 / 0.03)
4 -> 1   (approximate key: 463 / 0.64)
4 -> 2   (approximate key: 463 / 0.64)
4 -> 3   (approximate key: 463 / 0.64)
4 -> 5   (approximate key: 463 / 0.64)
4 -> 6   (approximate key: 463 / 0.64)
5 -> 1   (approximate key: 300 / 0.42)
5 -> 2   (approximate key: 300 / 0.42)
5 -> 3   (approximate key: 300 / 0.42)
5 -> 4   (approximate key: 300 / 0.42)
5 -> 6   (approximate key: 300 / 0.42)
6 -> 1   (approximate key: 4 / 0.01)
6 -> 2   (approximate key: 4 / 0.01)
6 -> 3   (approximate key: 4 / 0.01)
6 -> 4   (approximate key: 4 / 0.01)
6 -> 5   (approximate key: 4 / 0.01)

=====================================================================
total_no_of_candidates == 12
prune_key             == 11
prune_key_sub         == 0
prune_key_second      == 0
prune_rhs             == 5
prune_rhs_sub         == 0
prune_rhs_second      == 1
=================================================================
```

Figure 22. Output of TANE algorithms for table AE_F on 0.90 G3 range

```
Level == 1  #candidates == 6      avg.elements == 8     (1710/208)

==================================================================
Parameters (approximate dependencies):
No. of tuples           == 720
No. of attributes       == 6
Stop level              == 6
Data                    == data/TaxonAF.dat
Percentage of all tuples == 1.00 %
==> G3 threshold        == 720 max. rows removed
==================================================================

-> 1   (720 / 1.00)
-> 2   (715 / 0.99)
-> 3   (50 / 0.07)
-> 4   (684 / 0.95)
-> 5   (666 / 0.93)
-> 6   (717 / 1.00)

==================================================================
total_no_of_candidates == 6
prune_key             == 6
prune_key_sub         == 0
prune_key_second      == 0
prune_rhs             == 5
prune_rhs_sub         == 0
prune_rhs_second      == 0
==================================================================
```

Figure 23. Output of TANE algorithms for table AE_F on 1.00 G3 range

4.2.1 Summary output of table AE_F

The results of FDs discovery from AE_F table is summarised according to the range of G3.

The tables are shown as follows:

Table 6. FDs discoveries in AE_F table with G3 ranges of 0.10 to 1.00

(a) G3 range is 0.10

G3 (0.10%)		
0	0.01	0.03
A -> B	F->A	B->A
A ->C	F->B	B->C
A ->D	F->C	B->D
A ->E	F->D	B->E
A ->F	F->E	B->F

(b) G3 range is 0.20

G3 (0.20%)			
0	0.01	0.03	0.11
A -> B	F->A	B->A	E -> D
A ->C	F->B	B->C	
A ->D	F->C	B->D	
A ->E	F->D	B->E	
A ->F	F->E	B->F	

(c) G3 range is 0.30

G3 (0.30%)				
0	0.01	0.03	0.07	0.11
A -> B	F->A	B->A	A,B,D,E,F -> C	E -> D
A ->C	F->B	B->C		
A ->D	F->C	B->D		
A ->E	F->D	B->E		
A ->F	F->E	B->F		

(d) G3 range is 0.40

G3 (0.40%)				
0	0.01	0.03	0.07	0.11
A -> B	F->A	B->A	A,B,D,E,F -> C	E -> D
A ->C	F->B	B->C		
A ->D	F->C	B->D		
A ->E	F->D	B->E		
A ->F	F->E	B->F		

(e) G3 range is 0.50

G3 (0.50%)					
0	0.01	0.03	0.07	0.42	0.45
A -> B	F->A	B->A	A,B,D,E,F -> C	E -> A	D -> E
A ->C	F->B	B->C		E -> B	
A ->D	F->C	B->D		E -> C	
A ->E	F->D	B->E		E -> D	
A ->F	F->E	B->F		E -> F	

(f) G3 range is 0.60

G3 (0.60%)					
0	0.01	0.03	0.07	0.42	0.45
A -> B	F->A	B->A	A,B,D,E,F -> C	E -> A	D -> E
A ->C	F->B	B->C		E -> B	
A ->D	F->C	B->D		E -> C	
A ->E	F->D	B->E		E -> D	
A ->F	F->E	B->F		E -> F	

(g) G3 range is 0.70

G3 (0.70%)					
0	0.01	0.03	0.07	0.42	0.64
A -> B	F->A	B->A	A,B,D,E,F -> C	E -> A	D -> A
A ->C	F->B	B->C		E -> B	D -> B
A ->D	F->C	B->D		E -> C	D -> C
A ->E	F->D	B->E		E -> D	D -> E
A ->F	F->E	B->F		E -> F	D -> F

(h) G3 range is 0.80

G3 (0.80%)					
0	0.01	0.03	0.07	0.42	0.64
A -> B	F->A	B->A	A,B,D,E,F -> C	E -> A	D -> A
A ->C	F->B	B->C		E -> B	D -> B
A ->D	F->C	B->D		E -> C	D -> C
A ->E	F->D	B->E		E -> D	D -> E
A ->F	F->E	B->F		E -> F	D -> F

(i) G3 range is 0.90

G3 (0.90%)					
0	0.01	0.03	0.07	0.42	0.64
A -> B	F->A	B->A	A,B,D,E,F -> C	E -> A	D -> A
A ->C	F->B	B->C		E -> B	D -> B
A ->D	F->C	B->D		E -> C	D -> C
A ->E	F->D	B->E		E -> D	D -> E
A ->F	F->E	B->F		E -> F	D -> F

(j) G3 range is 1.00

G3 (1.0%)				
0.07	0.93	0.95	0.99	1.00
A,B,D,E,F -> C	A,B,C,D,F -> E	A,B,C,E,F -> D	A,C,D,E,F-> B	B,C,D,E,F ->A
				A,B,C,D,E ->F

Table 7. Overall FD accuracy and proxy table size analysis for table AE_F

Proxy candidates	**G3**	**FDs Accuracy Percentage (%)**	**Proxy table size**
A -> F	0	100	1442
B->F	0.03	97	1442
C->F	-	-	-
D->F	0.64	36	1442
E->F	0.42	48	1442
AB->F	-	-	-
AC->F	-	-	-
AD->F	-	-	-
AE->F	-	-	-
BC->F	-	-	-
BD->F	-	-	-
BE->F	-	-	-
CD->F	-	-	-
CE->F	-	-	-
DE->F	-	-	-
ABC->F	-	-	-
ABD->F	-	-	-
ABE->F	-	-	-
BCD->F	-	-	-
BCE->F	-	-	-
CDE->F	-	-	-
ABCD->F	-	-	-
ABCE->F	-	-	-
BCDE->F	-	-	-
ABCDE->F	1.00	0	4326

From the results table 7, it is found that, the presence of FDs on attribute A -> F, B -> F, D -> F and E -> F. The FD for A->F is 100% accurate where the G3 value is 0. And B -> F shows about 97% of accurate prediction of FDs. The other two FD predictions in between D -> F and E -> F does not showing precise prediction. Because D->F produced G3 error of 0.42 which is about only 58% FD accuracy. Whereas E->F produced G3 error of 0.64, about 36% FD accuracy. If the G3 value is near to zero than the prediction of FDs is 100% accurate.

4.2.2 Summary output of table AE_G

The AE_G table also produced ten outputs through the TANE algorithm. The analyses of the output for FD discovery from table AE_G are shown in Table 8.

Table 8. FDs discoveries in AE_G table with G3 ranges of 0.10 to 1.00

(a) G3 range is 0.10

G3 (0.10%)			
0	0.01	0.02	0.07
A -> B	D -> G	B->A	A,B,D,E,G -> C
A ->C		B->C	E -> G
A ->D		B->D	
A ->E		B->E	
A ->G		B->G	

(b) G3 range is 0.20

G3 (0.20%)				
0	0.01	0.02	0.07	0.11
A -> B	D -> G	B->A	A,B,D,E,G -> C	E -> D
A ->C		B->C	E -> G	
A ->D		B->D		
A ->E		B->E		
A ->G		B->G		

(c) G3 range is 0.30

G3 (0.30%)				
0	0.01	0.02	0.07	0.11
A -> B	D -> G	B->A	A,B,D,E,G -> C	E -> D
A ->C		B->C	E -> G	
A ->D		B->D		
A ->E		B->E		
A ->G		B->G		

(d) G3 range is 0.40

G3 (0.40%)				
0	0.01	0.02	0.07	0.11
A -> B	D -> G	B->A	A,B,D,E,G -> C	E -> D
A ->C		B->C	E -> G	
A ->D		B->D		
A ->E		B->E		
A ->G		B->G		

(e) G3 range is 0.50

G3 (0.50%)					
0	0.01	0.02	0.07	0.41	0.45
A -> B	D -> G	B->A	A,B,D,E,G -> C	E -> A	D -> E
A ->C		B->C		E -> B	
A ->D		B->D		E -> C	
A ->E		B->E		E -> D	
A ->G		B->G		E -> G	

(f) G3 range is 0.60

G3 (0.60%)					
0	0.02	0.07	0.41	0.45	0.53
A -> B	B->A	A,B,D,E,G -> C	E -> A	D -> E	A,B,C,D,E -> G
A ->C	B->C		E -> B		
A ->D	B->D		E -> C		
A ->E	B->E		E -> D		
A ->G	B->G		E -> G		

(g) G3 range is 0.70

G3 (0.70%)					
0	0.02	0.07	0.41	0.53	0.64
A -> B	B->A	A,B,D,E,G -> C	E -> A	A,B,C,D,E -> G	D -> A
A ->C	B->C		E -> B		D -> B
A ->D	B->D		E -> C		D -> C
A ->E	B->E		E -> D		D -> E
A ->G	B->G		E -> G		D -> G

(h) G3 range is 0.80

G3 (0.80%)					
0	0.02	0.07	0.41	0.53	0.64
A -> B	B->A	A,B,D,E,G -> C	E -> A	A,B,C,D,E -> G	D -> A
A ->C	B->C		E -> B		D -> B
A ->D	B->D		E -> C		D -> C
A ->E	B->E		E -> D		D -> E
A ->G	B->G		E -> G		D -> G

(i) G3 range is 0.90

G3 (0.90%)					
0	0.02	0.07	0.41	0.53	0.64
A -> B	B->A	A,B,D,E,G -> C	E -> A	A,B,C,D,E -> G	D -> A
A ->C	B->C		E -> B		D -> B
A ->D	B->D		E -> C		D -> C
A ->E	B->E		E -> D		D -> E
A ->G	B->G		E -> G		D -> G

(j) G3 range is 1.00

G3 (1.00%)					
0.07	0.53	0.93	0.95	0.99	1.00
A,B,D,E,G -> C	A,B,C,D,E -> G	A,B,C,D,G -> E	A,B,C,E,G -> D	A,C,D,E,G -> B	B,C,D,E,G -> A

Table 9. Overall FD accuracy and proxy table size analysis for table AE_G

Proxy Candidates	**G3**	**FDs Accuracy Percentage (%)**	**Proxy table size**
A ->G	0	100	1446
B->G	0.02	98	1446
C->G	-	-	-
D->G	0.64	36	1446
E->G	0.07 | 0.41	93 | 59	1446
AB->G	-	-	-
AC->G	-	-	-
AD->G	-	-	-
AE->G	-	-	-
BC->G	-	-	-
BD->G	-	-	-
BE->G	-	-	-
CD->G	-	-	-
CE->G	-	-	-
DE->G	-	-	-
ABC->G	-	-	-
ABD->G	-	-	-
ABE->G	-	-	-
BCD->G	-	-	-
BCE->G	-	-	-
CDE->G	-	-	-
ABCD->G	-	-	-
ABCE->G	-	-	-
BCDE->G	-	-	-
ABCDE->G	0.53	47	4338

Table 10 shows analysis of result of table AE_G from TANE algorithm. Based on this table some observation are can be made. There are presence of FDs in between A-> G, B->G, D-> G and E -> G. Even though there are presence of FDs, refer to it's G3 error values which is 0 or nearly zero it is acceptable as accurate FDs. Hence here, A->G and B->G are acceptable as accurate FD occurrence because of G3 error for A->G is 0 (100% accurate) and 0.02 G3 error for B->G (98% accurate). Besides, D-> G and E -> G showing worst case scenarios since both produced higher percentage of G3 error. D->G produced 0.64 G3 error values which are only 36% of accurate FD occurrence. It is different case for E->G because TANE has produced both best case scenario and worst case scenario. For the best case scenario, when the G3 ranges used in TANE is from 0.10 to 0.40 it produces 0.07 G3 errors which are about 93% of FD existence accuracy. On the other hand, worst case, produces 0.41 G3 error from the ranges of 0.50 to 0.90. Therefore, we have to ignore the low G3 error of 0.07 and take into account of the higher value of G3 which is 0.41. The worst case of E->G shows very low percentage (only 59%) of FD accuracy. Though we can say that A, B, D, and E are the proxy candidates for G, only A and B acceptable as good candidates as both produce very low G3 error which is near to zero.

4.2.3 Summary output of table AE_H

Table 10 shows summary of the output for AE_H table for FD discovery.

Table 10. FDs discoveries in AE_F table with G3 ranges of 0.10 to 1.00

(a) G3 range is 0.10

G3 (0.10%)				
0	0.01	0.02	0.04	0.07
A -> B	D->H	B->A	E,H->D	A,B,D,E,H -> C
A ->C		B->C		
A ->D		B->D		
A ->E		B->E		
A ->H		B->H		

(b) G3 range is 0.20

G3 (0.20%)					
0	0.01	0.02	0.07	0.10	0.11
A -> B	D->H	B->A	A,B,D,E,H -> C	E ->H	E->D
A ->C		B->C			
A ->D		B->D			
A ->E		B->E			
A ->H		B->H			

(c) G3 range is 0.30

G3 (0.30%)					
0	0.01	0.02	0.07	0.10	0.11
A -> B	D->H	B->A	A,B,D,E,H -> C	E ->H	E->D
A ->C		B->C			
A ->D		B->D			
A ->E		B->E			
A ->H		B->H			

(d) G3 range is 0.40

G3 (0.40%)					
0	0.01	0.02	0.07	0.10	0.11
A -> B	D->H	B->A	A,B,D,E,H -> C	E ->H	E->D
A ->C		B->C			
A ->D		B->D			
A ->E		B->E			
A ->H		B->H			

(e) G3 range is 0.50

G3 (0.50%)					
0	0.01	0.02	0.07	0.41	0.45
A -> B	D->H	B->A	A,B,D,E,H -> C	E ->A	D->E
A ->C		B->C		E ->B	
A ->D		B->D		E ->C	
A ->E		B->E		E ->D	
A ->H		B->H		E ->H	

(f) G3 range is 0.60

G3 (0.60%)					
0	0.01	0.02	0.07	0.41	0.45
A -> B	D->H	B->A	A,B,D,E,H -> C	E ->A	D->E
A ->C		B->C		E ->B	
A ->D		B->D		E ->C	
A ->E		B->E		E ->D	
A ->H		B->H		E ->H	

(g) G3 range is 0.70

G3 (0.70%)					
0	0.02	0.07	0.41	0.61	0.64
A -> B	B->A	A,B,D,E,H -> C	E ->A	H->D	D->A
A ->C	B->C		E ->B		D->B
A ->D	B->D		E ->C		D->C
A ->E	B->E		E ->D		D->E
A ->H	B->H		E ->H		D->H

(h) G3 range is 0.80

G3 (0.80%)					
0	0.02	0.07	0.41	0.64	0.77
A -> B	B->A	A,B,D,E,H -> C	E ->A	D->A	A,B,C,D,E -> H
A ->C	B->C		E ->B	D->B	
A ->D	B->D		E ->C	D->C	
A ->E	B->E		E ->D	D->E	
A ->H	B->H		E ->H	D->H	

(i) G3 range is 0.90

G3 (0.90%)					
0	0.02	0.07	0.41	0.64	0.77
A -> B	B->A	A,B,D,E,H -> C	E ->A	D->A	A,B,C,D,E -> H
A ->C	B->C		E ->B	D->B	
A ->D	B->D		E ->C	D->C	
A ->E	B->E		E ->D	D->E	
A ->H	B->H		E ->H	D->H	

(j) G3 range is 1.00

G3 (1.00%)					
0.07	0.77	0.93	0.95	0.99	1.00
A,B,D,E,H -> C	A,B,C,D,E -> H	A,B,C,D,H -> E	A,B,C,E,H -> D	A,C,D,E,H -> B	B,C,D,E,H -> A

Table 11. Overall FD accuracy and proxy table size analysis for table AE_H

Proxy	**G3**	**FDs Accuracy Percentage (%)**	**Proxy table size**
A ->H	0	100	1446
B->H	0.02	98	1446
C->H	-	-	-
D->H	0.01 | 0.64	99 | 36	1446
E->H	0.10 | 0.41	90 | 59	1446
AB->H	-	-	-
AC->H	-	-	-
AD->H	-	-	-
AE->H	-	-	-
BC->H	-	-	-
BD->H	-	-	-
BE->H	-	-	-
CD->H	-	-	-
CE->H	-	-	-
DE->H	-	-	-
ABC->H	-	-	-
ABD->H	-	-	-
ABE->H	-	-	-
BCD->H	-	-	-
BCE->H	-	-	-
CDE->H	-	-	-
ABCD->H	-	-	-
ABCE->H	-	-	-
BCDE->H	-	-	-
ABCDE->H	0.77	33	4338

As you can see from the Table 11, not all the attributes from the input table can be a proxy for the attribute under observation which is H. For example, A, B, D and E can be considered as candidates proxies for attribute H. From the table, we can say that, those candidates' proxies not 100% accurate but have very low G3 error. Therefore, attribute A is showing 0 G3 errors which 100% accurate existence of FD. While B showing 0.02 (98% accuracy) G3 errors which is very low or near to zero.

On the other hand, attribute D and E shows two different scenarios. Both had produced low G3 error and higher G3 error which can be categorised into best case scenario and worst case scenario. As usual, in this case, the low G3 error must be ignored because when the ranges of G3 are varies it is not produce a stable G3 error value. Hence, the higher G3 error must take to be analysed. When the ranges of G3 for D are 0.10 to 0.50 it produces 0.01 very low errors. However, this error value is not stable when the G3 range varies from 0.60 to 0.90 producing high G3 error with 0.64 values. For E, TANE produces 0.10 G3 errors when the range is from 0.10 to 0.40; when range is increase from 0.50 to 0.90 the G3 error level increase up to 0.41. Thus, D and E are not good candidate proxies since both produce high G3 errors.

4.2.4 Summary output of table AE_I

Here are summary of FD discovery from the outputs of table AE_F (Table 12).

Table 12. FDs discoveries in AE_F table with G3 ranges of 0.10 to 1.00

(a) G3 range is 0.10

G3 (0.10%)			
0	0.02	0.03	0.07
A -> B	D->I	B->A	A,B,D,E,I -> C
A ->C		B->C	
A ->D		B->D	
A ->E		B->E	
A ->I		B->I	
		E,I->D	

(b) G3 range is 0.20

G3 (0.20%)				
0	0.02	0.03	0.07	0.11
A -> B	D->I	B->A	A,B,D,E,I -> C	E->D
A ->C		B->C		E->I
A ->D		B->D		
A ->E		B->E		
A ->I		B->I		

(c) G3 range is 0.30

G3 (0.30%)				
0	0.02	0.03	0.07	0.11
A -> B	D->I	B->A	A,B,D,E,I -> C	E->D
A ->C		B->C		E->I
A ->D		B->D		
A ->E		B->E		
A ->I		B->I		

(d) G3 range is 0.40

G3 (0.40%)					
0	0.02	0.03	0.07	0.11	0.39
A -> B	D->I	B->A	A,B,D,E,I -> C	E->D	I->D
A ->C		B->C		E->I	
A ->D		B->D			
A ->E		B->E			
A ->I		B->I			

(e) G3 range is 0.50

G3 (0.50%)						
0	0.02	0.03	0.07	0.39	0.42	0.45
A -> B	D->I	B->A	A,B,D,E,I -> C	I->D	E->A	D->E
A ->C		B->C			E->B	
A ->D		B->D			E->C	
A ->E		B->E			E->D	
A ->I		B->I			E->I	

(f) G3 range is 0.60

G3 (0.60%)						
0	0.02	0.03	0.07	0.39	0.42	0.45
A -> B	D->I	B->A	A,B,D,E,I -> C	I->D	E->A	D->E
A ->C		B->C			E->B	
A ->D		B->D			E->C	
A ->E		B->E			E->D	
A ->I		B->I			E->I	

(g) G3 range is 0.70

G3 (0.70%)						
0	0.03	0.07	0.39	0.42	0.64	0.70
A -> B	B->A	A,B,D,E,I -> C	I->D	E->A	D->A	I->E
A ->C	B->C			E->B	D->B	
A ->D	B->D			E->C	D->C	
A ->E	B->E			E->D	D->E	
A ->I	B->I			E->I	D->I	

(h) G3 range is 0.80

G3 (0.80%)						
0	0.03	0.07	0.39	0.42	0.64	0.70
A -> B	B->A	A,B,D,E,I -> C	I->D	E->A	D->A	I->E
A ->C	B->C			E->B	D->B	
A ->D	B->D			E->C	D->C	
A ->E	B->E			E->D	D->E	
A ->I	B->I			E->I	D->I	

(i) G3 range is 0.90

G3 (0.90%)					
0	0.03	0.07	0.42	0.64	0.86
A -> B	B->A	A,B,D,E,I -> C	E->A	D->A	I->A
A ->C	B->C		E->B	D->B	I->B
A ->D	B->D		E->C	D->C	I->C
A ->E	B->E		E->D	D->E	I->D
A ->I	B->I		E->I	D->I	I->E

(j) G3 range is 1.00

G3 (1.00%)					
0.07	0.91	0.93	0.95	0.99	1.00
A,B,D,E,I -> C	A,B,C,D,E->I	A,B,C,D,I->E	A,B,C,E,I->D	A,C,D,E,I->B	B,C,D,E,I->A

Table 13. Overall FD accuracy and proxy table size analysis for table AE_I

Proxy	G3	FDs Accuracy Percentage (%)	Proxy table size
A ->I	0	100	1436
B->I	0.03	97	1436
C->I	-	-	-
D->I	0.02 | 0.64	98 | 36	1436
E->I	0.11 | 0.42	89 | 58	1436
AB->I	-	-	-
AC->I	-	-	-
AD->I	-	-	-
AE->I	-	-	-
BC->I	-	-	-
BD->I	-	-	-
BE->I	-	-	-
CD->I	-	-	-
CE->I	-	-	-
DE->I	-	-	-
ABC->I	-	-	-
ABD->I	-	-	-
ABE->I	-	-	-
BCD->I	-	-	-
BCE->I	-	-	-
CDE->I	-	-	-
ABCD->I	-	-	-
ABCE->I	-	-	-
BCDE->I	-	-	-
ABCDE->I	0.91	9	4308

As table AE_H, in table AE_I (Table 13) attribute A, B, D and E are can be considered as candidates proxies for attribute F. Even though, there are four attribute consider as candidate proxies only attribute A and B shows low G3 errors while D and E shows both low and high G3 errors. Attribute A shows 100% of accurate FD occurrence with 0 G3 errors. Besides, attribute B shows 0.03 G3 errors (nearly 0) with 97% FD accuracy.

Attribute D and E shows best case scenario with low G3 errors and worst case scenario with high G3 errors. From TANE, D->I produces 0.02 (98% accuracy) G3 errors which is very low errors in between ranges of G3 are 0.10 to 0.60. Meanwhile, it shows higher G3 errors, 0.64 (36% accuracy) when there are changes in ranges of G3 from 0.70 to 0.90. Moreover, E->I shows 0.11 (89% accuracy) G3 errors on ranges of G3 are 0.10 to 0.40. It produces high G3 error, 0.42 when the G3 ranges varies from 0.50 to 0.90 which showing only 58% of FD existence. Therefore, we have to consider the worst case scenario in this particular situation.

4.2.5 Summary output of table AE_J

The following Table 14 shows the summary of FD discoveries from output from table AE_F.

Table 14. FDs discoveries in AE_F table with G3 ranges of 0.10 to 1.00

(a) G3 range is 0.10

G3 (0.10%)			
0	0.02	0.07	0.10
A -> B	B->A	A,B,D,E,J -> C	E->J
A ->C	B->C		
A ->D	B->D		
A ->E	B->E		
A ->J	B->J		
	D->J		

(b) G3 range is 0.20

G3 (0.20%)			
0	0.02	0.07	0.10
A -> B	B->A	A,B,D,E,J -> C	E->D
A ->C	B->C		E->J
A ->D	B->D		
A ->E	B->E		
A ->J	B->J		
	D->J		

(c) G3 range is 0.30

G3 (0.30%)				
0	0.02	0.07	0.10	0.25
A -> B	B->A	A,B,D,E,J -> C	E->D	J->D
A ->C	B->C		E->J	
A ->D	B->D			
A ->E	B->E			

A ->J	B->J			
	D->J			

(d) G3 range is 0.40

G3 (0.40%)				
0	0.02	0.07	0.25	0.39
A -> B	B->A	A,B,D,E,J -> C	E->D	E->A
A ->C	B->C			E->B
A ->D	B->D			E->C
A ->E	B->E			E->D
A ->J	B->J			E->J
	D->J			

(e) G3 range is 0.50

G3 (0.50%)					
0	0.02	0.07	0.25	0.39	0.46
A -> B	B->A	A,B,D,E,J -> C	E->D	E->A	D->E
A ->C	B->C			E->B	
A ->D	B->D			E->C	
A ->E	B->E			E->D	
A ->J	B->J			E->J	
	D->J				

(f) G3 range is 0.60

G3 (0.60%)					
0	0.02	0.07	0.25	0.39	0.46
A -> B	B->A	A,B,D,E,J -> C	E->D	E->A	D->E
A ->C	B->C			E->B	
A ->D	B->D			E->C	
A ->E	B->E			E->D	
A ->J	B->J			E->J	
	D->J				

(g) G3 range is 0.70

G3 (0.70%)						
0	0.02	0.07	0.25	0.39	0.62	0.63
A -> B	B->A	A,B,D,E,J -> C	J->D	E->A	J->E	D->A
A ->C	B->C			E->B		D->B
A ->D	B->D			E->C		D->C
A ->E	B->E			E->D		D->E
A ->J	B->J			E->J		D->J

(h) G3 range is 0.80

G3 (0.80%)					
0	0.02	0.07	0.39	0.63	0.76
A -> B	B->A	A,B,D,E,J -> C	E->A	D->A	J->A
A ->C	B->C		E->B	D->B	J->B
A ->D	B->D		E->C	D->C	J->C

A ->E	B->E		E->D	D->E	J->D
A ->J	B->J		E->J	D->J	J->E

(i) G3 range is 0.90

G3 (0.90%)					
0	0.02	0.07	0.39	0.63	0.76
A -> B	B->A	A,B,D,E,J -> C	E->A	D->A	J->A
A ->C	B->C		E->B	D->B	J->B
A ->D	B->D		E->C	D->C	J->C
A ->E	B->E		E->D	D->E	J->D
A ->J	B->J		E->J	D->J	J->E

(j) G3 range is 1.00

G3 (1.00%)				
0.07	0.90	0.94	0.95	1.00
A,B,D,E,J -> C	A,B,C,D,E -> J	A,B,C,D,J -> E	A,B,C,E,J -> D	B,C,D,E,J -> A
				A,C,D,E,J -> B

Table 15. Overall FD accuracy and proxy table size analysis for table AE_J

Proxy	G3	FDs Accuracy Percentage (%)	Proxy table size
A ->J	0	100	1314
B->J	0.02	98	1314
C->J	-	-	-
D->J	0.02 | 0.63	98 | 37	1314
E->J	0.10 | 0.39	90 | 61	1314
AB->J	-	-	-
AC->J	-	-	-
AD->J	-	-	-
AE->J	-	-	-
BC->J	-	-	-
BD->J	-	-	-
BE->J	-	-	-
CD->J	-	-	-
CE->J	-	-	-
DE->J	-	-	-
ABC->J	-	-	-
ABD->J	-	-	-
ABE->J	-	-	-
BCD->J	-	-	-
BCE->J	-	-	-
CDE->J	-	-	-
ABCD->J	-	-	-
ABCE->J	-	-	-
BCDE->J	-	-	-
ABCDE->J	0.90	10	3942

Table 15 shows attribute A, B, D and E are to be candidate proxies for attribute J. Through best case scenarios, attribute A and B are to be good candidate proxies since both showing zero and nearly zero G3 errors. Since attribute A is the key for table AE-J, than it shows zero G3 error which is 100% accurate FD. Attribute B too shows nearly zero G3 error (0.02), 98% of accurate FD presence.

Best case scenario for D->J produces 0.02 G3 errors when the ranges are in between 0.10 to 0.60; while the ranges increase from 0.70 to 0.90 than the G3 error increase as well up to 0.63 (37% accuracy) which is very high showing worst case situation. On the other hand, similar things happen to E->J where it produces 0.10 G3 errors when the ranges up to 0.30; if the G3 ranges increase from 0.40 to 0.90, accuracy of FD become lower to 61% (0.39 G3 errors).

4.2.6 Summary output of table AE_K

Table 16 shows output from TANE algorithm for table AE_K which are summarised according to G3 ranges.

Table 16. FDs discoveries in AE_F table with G3 ranges of 0.10 to 1.00

(a) G3 range is 0.10

G3 (0.10%)						
0	0.02	0.03	0.04	0.06	0.09	0.10
A -> B	B->A	K->D	D->K	A,B,D,E,K -> C	E->D	E->K
A ->C	B->C					
A ->D	B->D					
A ->E	B->E					
A ->K	B->K					

(b) G3 range is 0.20

G3 (0.20%)						
0	0.02	0.03	0.04	0.06	0.09	0.10
A -> B	B->A	K->D	D->K	A,B,D,E,K -> C	E->D	E->K
A ->C	B->C					
A ->D	B->D					
A ->E	B->E					
A ->K	B->K					

(c) G3 range is 0.30

G3 (0.30%)						
0	0.02	0.03	0.04	0.06	0.09	0.10
A -> B	B->A	K->D	D->K	A,B,D,E,K -> C	E->D	E->K
A ->C	B->C					
A ->D	B->D					
A ->E	B->E					
A ->K	B->K					

(d) G3 range is 0.40

G3 (0.40%)						
0	0.02	0.03	0.04	0.06	0.37	0.39
A -> B	B->A	K->D	D->K	A,B,D,E,K -> C	E->A	D->E
A ->C	B->C				E->B	K->E
A ->D	B->D				E->C	
A ->E	B->E				E->D	
A ->K	B->K				E->K	

(e) G3 range is 0.50

G3 (0.50%)						
0	0.02	0.03	0.04	0.06	0.37	0.39
A -> B	B->A	K->D	D->K	A,B,D,E,K -> C	E->A	D->E
A ->C	B->C				E->B	K->E
A ->D	B->D				E->C	
A ->E	B->E				E->D	
A ->K	B->K				E->K	

(f) G3 range is 0.60

G3 (0.60%)					
0	0.02	0.06	0.37	0.57	0.58
A -> B	B->A	A,B,D,E,K -> C	E->A	K->A	D->A
A ->C	B->C		E->B	K->B	D->B
A ->D	B->D		E->C	K->C	D->C
A ->E	B->E		E->D	K->D	D->E
A ->K	B->K		E->K	K->E	D->K

(g) G3 range is 0.70

G3 (0.70%)					
0	0.02	0.06	0.37	0.57	0.58
A -> B	B->A	A,B,D,E,K -> C	E->A	K->A	D->A
A ->C	B->C		E->B	K->B	D->B
A ->D	B->D		E->C	K->C	D->C
A ->E	B->E		E->D	K->D	D->E
A ->K	B->K		E->K	K->E	D->K

(h) G3 range is 0.80

G3 (0.80%)					
0	0.02	0.06	0.37	0.57	0.58
A -> B	B->A	A,B,D,E,K -> C	E->A	K->A	D->A
A ->C	B->C		E->B	K->B	D->B
A ->D	B->D		E->C	K->C	D->C
A ->E	B->E		E->D	K->D	D->E
A ->K	B->K		E->K	K->E	D->K

(i) G3 range is 0.90

G3 (0.90%)					
0	0.02	0.06	0.37	0.57	0.58
A -> B	B->A	A,B,D,E,K -> C	E->A	K->A	D->A
A ->C	B->C		E->B	K->B	D->B
A ->D	B->D		E->C	K->C	D->C
A ->E	B->E		E->D	K->D	D->E
A ->K	B->K		E->K	K->E	D->K

(j) G3 range is 1.00

G3 (1.00%)			
0.06	0.93	0.95	1.00
A,B,D,E,K -> C	A,B,C,D,K -> E	A,B,C,E,K -> D	B,C,D,E,K -> A
		A,B,C,D,E -> K	A,C,D,E,K -> B

Table 17. Overall FD accuracy and proxy table size analysis for table AE_K

Proxy	**G3**	**FDs Accuracy Percentage (%)**	**Proxy table size**
A ->K	0	100	1004
B->K	0.02	98	1004
C->K	-	-	-
D->K	0.04 | 0.58	96 | 42	1004
E->K	0.10 | 0.37	90 | 63	1004
AB->K	-	-	-
AC->K	-	-	-
AD->K	-	-	-
AE->K	-	-	-
BC->K	-	-	-
BD->K	-	-	-
BE->K	-	-	-
CD->K	-	-	-
CE->K	-	-	-
DE->K	-	-	-
ABC->K	-	-	-
ABD->K	-	-	-
ABE->K	-	-	-
BCD->K	-	-	-
BCE->K	-	-	-
CDE->K	-	-	-
ABCD->K	-	-	-
ABCE->K	-	-	-
BCDE->K	-	-	-
ABCDE->K	0.95	5	3012

Table 17 shows that not all the attribute can be considered as candidate proxies for attribute K. Some attributes even though not 100% accurate and having very low G3 error than it can be acceptable as candidate proxies. Therefore, here the attribute A, B, D and E can be considered as candidate proxies for attribute K. G3 error that produced by TANE algorithm for A->K is 0 which is 100% accurate FD presence. For B->K it produces nearly zero error as well with 0.02 which is 98% FD accuracy.

As in previous table, there are best case and worst case scenarios in table 5.6.1. Only worst case scenario considered in this situation since the G3 errors is not stable if changes made on ranges of G3. Hence, D->K shows 42% of accuracy with 0.58 G3 errors. And E->K produces 0.37 G3 errors with 63% of accuracy of FD.

4.2.7 Summary output of table AE_L

This Table 18 shows the FD discovery summary for output of table AE_L in a simplified form.

Table 18. FDs discoveries in AE_F table with G3 ranges of 0.10 to 1.00

(a) G3 range is 0.10

G3 (0.10%)		
0	0.01	0.05
A -> B	B->A	E->D
A ->C	B->C	
A ->D	B->D	
A ->E	B->E	
A ->L	B->L	
	L->D	
	A,B,D,E,L -> C	

(b) G3 range is 0.20

G3 (0.20%)				
0	0.01	0.05	0.15	0.19
A -> B	B->A	E->D	E->L	D->L
A ->C	B->C			
A ->D	B->D			
A ->E	B->E			
A ->L	B->L			
	L->D			
	A,B,D,E,L -> C			

(c) G3 range is 0.30

G3 (0.30%)				
0	0.01	0.19	0.24	0.29
A -> B	B->A	D->L	E->A	L->E
A ->C	B->C		E->B	
A ->D	B->D		E->C	
A ->E	B->E		E->D	
A ->L	B->L		E->L	
	L->D			
	A,B,D,E,L -> C			

(d) G3 range is 0.40

G3 (0.40%)					
0	0.01	0.19	0.24	0.35	0.37
A -> B	B->A	D->L	E->A	L->A	D->E
A ->C	B->C		E->B	L->B	
A ->D	B->D		E->C	L->C	
A ->E	B->E		E->D	L->D	
A ->L	B->L		E->L	L->E	
	A,B,D,E,L -> C				

(e) G3 range is 0.50

G3 (0.50%)				
0	0.01	0.24	0.35	0.46
A -> B	B->A	E->A	L->A	D->A
A ->C	B->C	E->B	L->B	D->B
A ->D	B->D	E->C	L->C	D->C
A ->E	B->E	E->D	L->D	D->E
A ->L	B->L	E->L	L->E	D->L
	A,B,D,E,L -> C			

(f) G3 range is 0.60

G3 (0.60%)				
0	0.01	0.24	0.35	0.46
A -> B	B->A	E->A	L->A	D->A
A ->C	B->C	E->B	L->B	D->B
A ->D	B->D	E->C	L->C	D->C
A ->E	B->E	E->D	L->D	D->E
A ->L	B->L	E->L	L->E	D->L
	A,B,D,E,L -> C			

(g) G3 range is 0.70

G3 (0.70%)				
0	0.01	0.24	0.35	0.46
A -> B	B->A	E->A	L->A	D->A
A ->C	B->C	E->B	L->B	D->B
A ->D	B->D	E->C	L->C	D->C
A ->E	B->E	E->D	L->D	D->E
A ->L	B->L	E->L	L->E	D->L
	A,B,D,E,L -> C			

(h) G3 range is 0.80

G3 (0.80%)				
0	0.01	0.24	0.35	0.46
A -> B	B->A	E->A	L->A	D->A
A ->C	B->C	E->B	L->B	D->B
A ->D	B->D	E->C	L->C	D->C
A ->E	B->E	E->D	L->D	D->E
A ->L	B->L	E->L	L->E	D->L
	A,B,D,E,L -> C			

(i) G3 range is 0.90

G3 (0.90%)				
0	0.01	0.24	0.35	0.46
A -> B	B->A	E->A	L->A	D->A
A ->C	B->C	E->B	L->B	D->B
A ->D	B->D	E->C	L->C	D->C
A ->E	B->E	E->D	L->D	D->E
A ->L	B->L	E->L	L->E	D->L
	A,B,D,E,L -> C			

(j) G3 range is 1.00

G3 (1.00%)					
0.01	0.91	0.92	0.93	0.99	1.00
A,B,D,E,L -> C	A,B,C,E,L -> D	A,B,C,D,E -> L	A,B,C,D,L -> E	A,C,D,E,L -> B	B,C,D,E,L -> A

Table 19. Overall FD accuracy and proxy table size analysis for table AE_L

Proxy	G3	FDs Accuracy Percentage (%)	Proxy table size
A ->L	0	100	280
B->L	0.01	99	280
C->L	-	-	-
D->L	0.19 | 0.46	81 | 54	280
E->L	0.15 | 0.24	85 | 76	280
AB->L	-	-	-
AC->L	-	-	-
AD->L	-	-	-
AE->L	-	-	-
BC->L	-	-	-
BD->L	-	-	-
BE->L	-	-	-
CD->L	-	-	-
CE->L	-	-	-
DE->L	-	-	-
ABC->L	-	-	-
ABD->L	-	-	-
ABE->L	-	-	-
BCD->L	-	-	-
BCE->L	-	-	-
CDE->L	-	-	-
ABCD->L	-	-	-
ABCE->L	-	-	-
BCDE->L	-	-	-
ABCDE->L	0.92	8	840

In Table 19, attribute A, B, D and E are considered as candidate proxies. Though, all this can't become good candidate proxies, since they show different G3 errors. A good candidate proxy has zero G3 error with 100% accuracy FD attribute A. Candidate proxy B->L shows nearly zero error (0.01 G3 errors) with 99% of FD presence can be considered as good candidate proxy.

Like in the previous case, even though D and E show two different G3 errors, we have to consider the worst case scenario. Thus, D->L and E->L produce 0.46 (54% FD accuracy) and 0.24 (76% FD accuracy) G3 error respectively.

4.3 Summary of Space requirement results

Here we have shown one of the results from FD-based proxies' characteristic which is percentage of space requirement for the data storage. The space requirement analysis results shown further are total of seven sub-tables which have been separated from Taxon main table. The proxy map size was obtained by using pivot table function in MS Excel. This pivot function directly calculates, the total number of instances present in an attribute (column) and also help us to analyse whether droppable proxy attribute showing one to one relationship or one to many relationship.

4.3.1 Multi-valued table for Table AE_F

Table 20. Multi-table scheme of table AE_F (total instances)

Strain (F)	U_id (A)	Size of Proxy Map	Percentage of Space Requirement (%)
719	721	1440	$\frac{1440}{4326}$ x 100 = 33.29

Strain (F)	Taxon_id (B)	Size of Proxy Map	Percentage of Space Requirement (%)
719	721	1440	$\frac{1440}{4326}$ x 100 = 33.29

Strain (F)	Genus (D)	Size of Proxy Map	Percentage of Space Requirement (%)
719	721	1440	$\frac{1440}{4326}$ x 100 = 33.29

Strain (F)	Species (E)	Size of Proxy Map	Percentage of Space Requirement

			(%)
719	721	1440	$\frac{1440}{4326}$ x 100 = 33.29

4.3.2 Multi-valued table for Table AE_G

Table 21. Multi-table scheme of table AE_G (total instances)

Intermediate_rank_1 (G)	U_id (A)	Size of Proxy Map	Percentage of Space Requirement (%)
24	723	747	$\frac{747}{4338}$ x 100 = 17.22

Intermediate_rank_1 (G)	Taxon_id (B)	Size of Proxy Map	Percentage of Space Requirement (%)
24	723	747	$\frac{747}{4338}$ x 100 = 17.22

Intermediate_rank_1 (G)	Genus (D)	Size of Proxy Map	Percentage of Space Requirement (%)
24	723	747	$\frac{747}{4338}$ x 100 = 17.22

Intermediate_rank_1 (G)	Species (E)	Size of Proxy Map	Percentage of Space Requirement (%)
24	723	747	$\frac{747}{4338}$ x 100 = 17.22

4.3.3 Multi-valued table for Table AE_H

Table 22. Multi-table scheme of table AE_H (total instances)

Intermediate_rank_2	U_id (A)	Size of	Percentage of Space

(H)		Proxy Map	Requirement (%)
53	723	776	$\frac{776}{4338}$ x 100 = 17.89

Intermediate_rank_2 (H)	Taxon_id (B)	Size of Proxy Map	Percentage of Space Requirement (%)
53	723	776	$\frac{776}{4338}$ x 100 = 17.89

Intermediate_rank_2 (H)	Genus (D)	Size of Proxy Map	Percentage of Space Requirement (%)
53	723	776	$\frac{776}{4338}$ x 100 = 17.89

Intermediate_rank_2 (H)	Species (E)	Size of Proxy Map	Percentage of Space Requirement (%)
53	723	776	$\frac{776}{4338}$ x 100 = 17.89

4.3.4 Multi-valued table for Table AE_I

Table 23. Multi-table scheme of table AE_I (total instances)

Intermediate_rank_3 (I)	U_id (A)	Size of Proxy Map	Percentage of Space Requirement (%)

101	718	819	$\frac{819}{4308}$ x 100 = 19.01

Intermediate_rank_3 (I)	Taxon_id (B)	Size of Proxy Map	Percentage of Space Requirement (%)
101	718	819	$\frac{819}{4308}$ x 100 = 19.01

Intermediate_rank_3 (I)	Genus (D)	Size of Proxy Map	Percentage of Space Requirement (%)
101	718	819	$\frac{819}{4308}$ x 100 = 19.01

Intermediate_rank_3 (I)	Species (E)	Size of Proxy Map	Percentage of Space Requirement (%)
101	718	819	$\frac{819}{4308}$ x 100 = 19.01

4.3.5 Multi-valued table for Table AE_J

Table 24. Multi-table scheme of table AE_J (total instances)

Intermediate_rank_4 (J)	U_id (A)	Size of Proxy Map	Percentage of Space Requirement (%)
157	657	814	$\frac{814}{3942}$ x 100 = 20.65

Intermediate_rank_4 (J)	Taxon_id (B)	Size of Proxy Map	Percentage of Space Requirement (%)
157	657	814	$\frac{814}{3942}$ x 100 = 20.65

Intermediate_rank_4 (J)	Genus (D)	Size of Proxy Map	Percentage of Space Requirement (%)
157	657	814	$\frac{814}{3942}$ x 100 = 20.65

Intermediate_rank_4 (J)	Species (E)	Size of Proxy Map	Percentage of Space Requirement (%)
157	657	814	$\frac{814}{3942}$ x 100 = 20.65

4.3.6 Multi-valued table for Table AE_K

Table 25. Multi-table scheme of table AE_K (total instances)

Intermediate_rank_5 (K)	U_id (A)	Size of Proxy Map	Percentage of Space Requirement (%)
217	502	719	$\frac{719}{3012}$ x 100 = 23.87

Intermediate_rank_5 (K)	Taxon_id (B)	Size of Proxy Map	Percentage of Space Requirement (%)
217	502	719	$\frac{719}{3012}$ x 100 = 23.87

Intermediate_rank_5 (K)	Genus (D)	Size of Proxy Map	Percentage of Space Requirement (%)
217	502	719	$\frac{719}{3012}$ x 100 = 23.87

Intermediate_rank_5 (K)	Species (E)	Size of Proxy Map	Percentage of Space Requirement (%)
217	502	719	$\frac{719}{3012}$ x 100 = 23.87

4.3.7 Multi-valued table for Table AE_L

Table 26. Multi-table scheme of table AE_L (total instances)

Intermediate_rank_6 (L)	U_id (A)	Size of Proxy Map	Percentage of Space Requirement (%)
92	140	232	$\frac{232}{840}$ x 100 = 27.62

Intermediate_rank_6 (L)	Taxon_id (B)	Size of Proxy Map	Percentage of Space Requirement (%)
92	140	232	$\frac{232}{840}$ x 100 = 27.62

Intermediate_rank_6 (L)	Genus (D)	Size of Proxy Map	Percentage of Space Requirement (%)
92	140	232	$\frac{232}{840}$ x 100 = 27.62

Intermediate_rank_6 (L)	Species (E)	Size of Proxy Map	Percentage of Space Requirement (%)
92	140	232	$\frac{232}{840}$ x 100 = 27.62

4.4 Conclusions

This chapter provides us with summary of results from TANE algorithms. It has been divided into ranges categories to make ease of to predict the accuracy of FDs discovery. From here we can say that, if the produced G3 value is 0 or nearly zero, than the accuracy of FDs are good or ~100%. In the next chapter we are going to discuss about the results has presented in this chapter.

CHAPTER 5

RESULTS ANALYSIS AND DISCUSSIONS

5.1 Background

This chapter presents a discussion about the results obtained from the TANE algorithm in chapter 4. There are 70 outputs has been produced by TANE algorithm for seven sub-tables of Taxon. Since TANE algorithm contains G3 ranges from 0.10 to 1.00, each table to be produced 10 outputs.

5.2 Analysis of FD accuracy for candidate proxy in Taxon sub-tables

Table 27. Overall summary of FD accuracy percentage for candidate proxies.

(Proxy) Attribute	U_id (A)	Taxon_id (B)	Genus (D)	Species (E)
	% FD Accuracy	% FD Accuracy	% FD Accuracy	% FD Accuracy
F	100	97	36	58
G	100	98	36	59
H	100	98	36	59
I	100	97	36	58
J	100	98	37	61
K	100	98	42	63
L	100	99	54	76
Average	**100**	**98**	**40**	**62**

Table 27 shows the accuracy of all FD-based proxy candidates which are considered in this case studies. From the table we can say that, attributes A and B shows zero and nearly zero errors respectively. As an average, attribute A shows 100% of FD accuracy and B shows

98% of FD accuracy. Therefore, attributes A and B from the microbial data sets are non defective. But the other attributes D and E are shows higher G3 errors with low FD accuracy percentages, hence these are defective proxy candidates. Other than this, attribute C does not imply any dependency since the relationship not determines any error values. The proof is as presented in Table 28. This is because, the values in the attributes are cannot droppable, hence they not dependent each other.

Table 28. Proxy candidates that do not shows any accuracy in FD prediction

Proxy candidates	**G3**	**FDs Accuracy Percentage (%)**
C->L	-	-
AB->L	-	-
AC->L	-	-
AD->L	-	-
AE->L	-	-
BC->L	-	-
BD->L	-	-
BE->L	-	-
CD->L	-	-
CE->L	-	-
DE->L	-	-
ABC->L	-	-
ABD->L	-	-
ABE->L	-	-
BCD->L	-	-
BCE->L	-	-
CDE->L	-	-
ABCD->L	-	-
ABCE->L	-	-
BCDE->L	-	-

Figure 24 to 30 shows information about the G3 errors and the FD accuracy for all the seven sub-tables from Taxon. From the figures, we can conclude that if lower the G3 errors, than the higher the percentage of FD accuracy. From the seven figures, as an overall observation can conclude attributes A and B are showing very low G3 errors with higher percentage of FD accuracy. They were accepted as good candidate proxies.

The attribute A shows zero error, this is because it is the key attribute to the Taxon table. In addition, attribute B can be a good proxy other than A (key attribute), because averagely it showing 98% FD accuracy. Therefore, attribute B is able to save the data from any loss since it becomes second important key attribute to the Taxon data set. However, attributes D and E are not good proxy candidates as their FD accuracy percentage very low which is far from best case scenario.

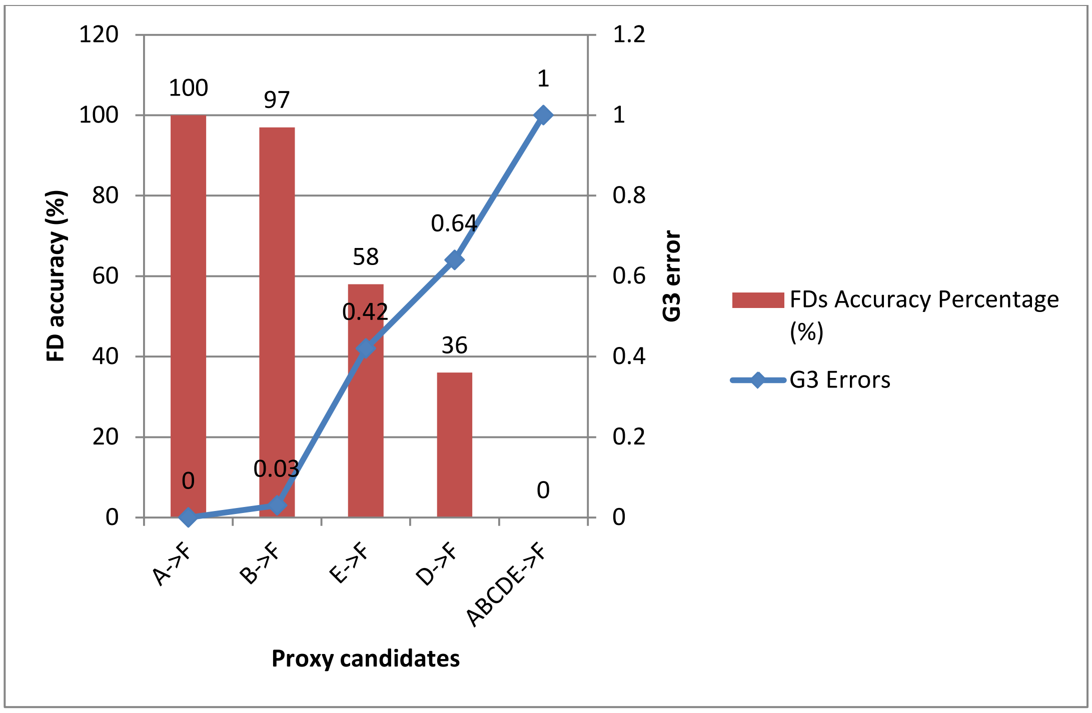


Figure 24. FD accuracy percentage and G3 errors table AE_F

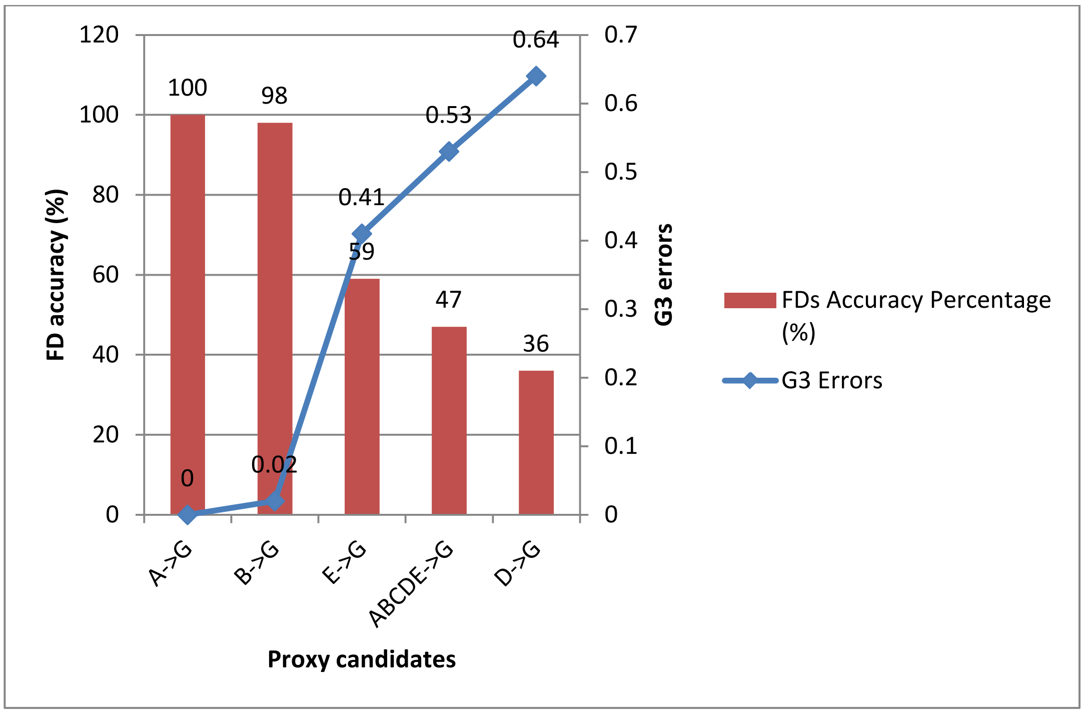


Figure 25. FD accuracy percentage and G3 errors for table AE_G

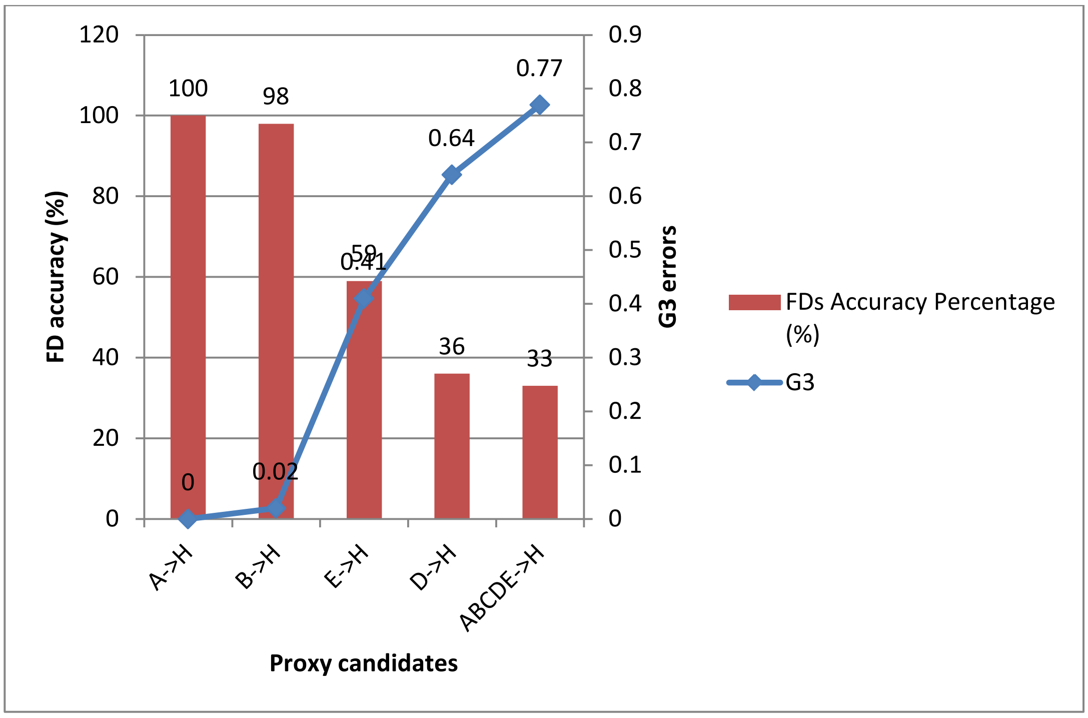


Figure 26. Proxy H FD accuracy percentage and G3 errors table AE_H

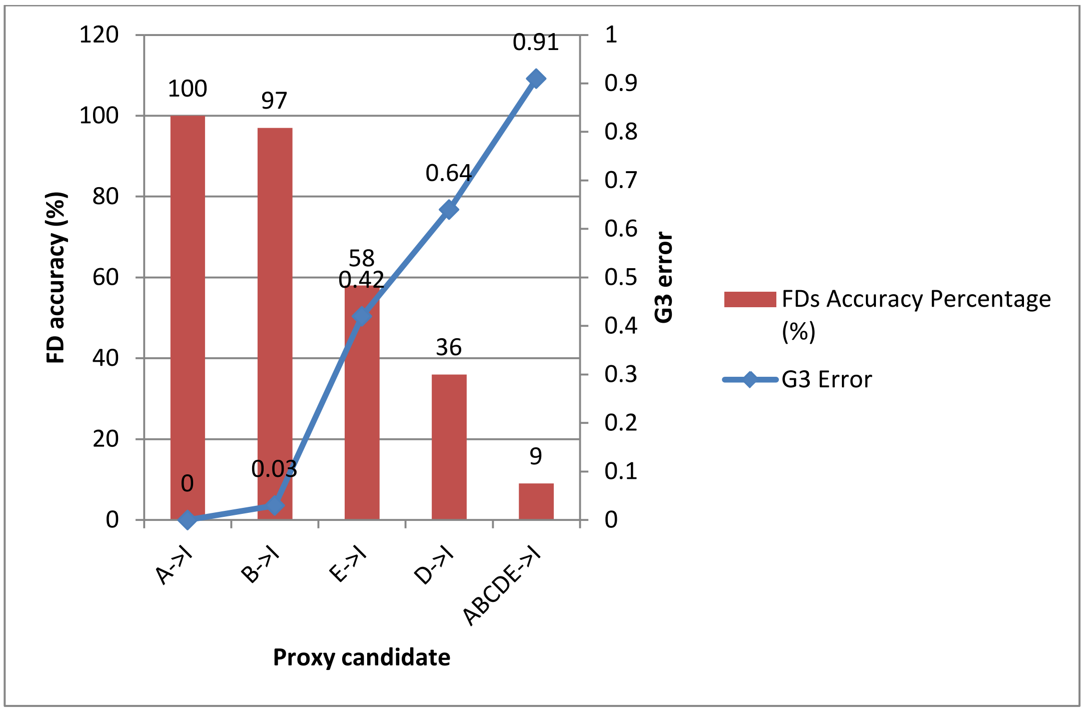


Figure 27. FD accuracy percentage and G3 errors table AE_I

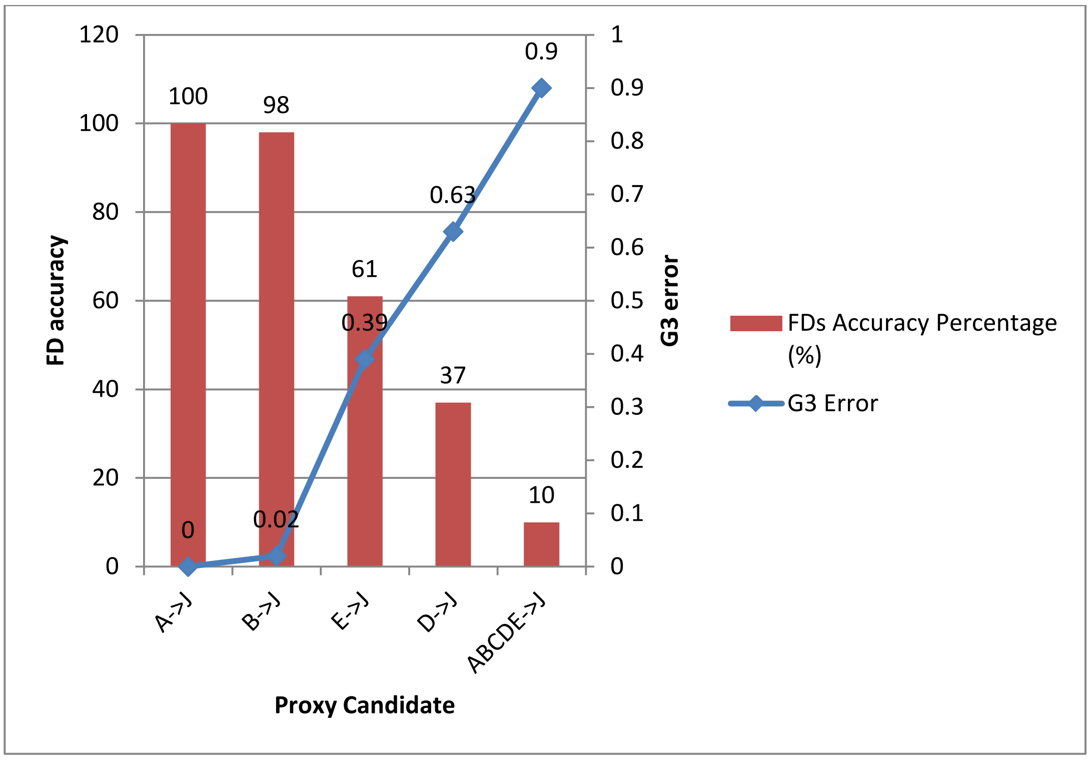


Figure 28. FD accuracy percentage and G3 errors table AE_J

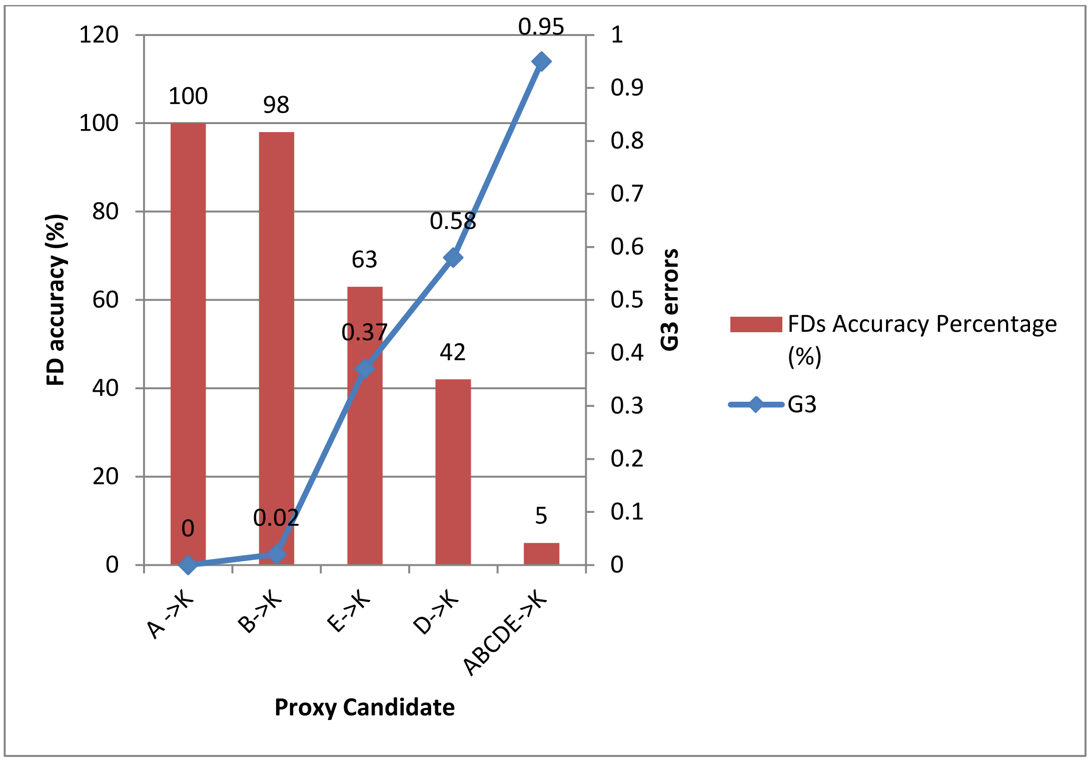


Figure 29. FD accuracy percentage and G3 errors table AE_K

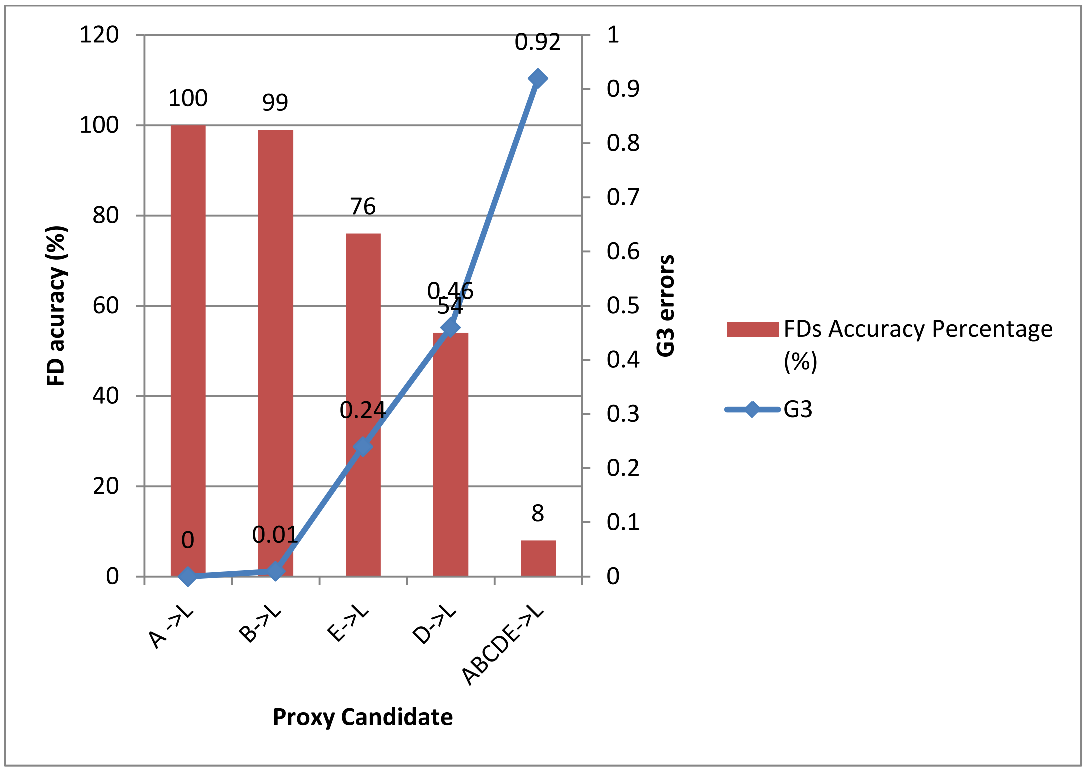


Figure 30. FD accuracy percentage and G3 errors table AE_L

5.3 Space Requirement Analysis

From this case study, we can say that a good characteristic of a proxy candidates is small amount of space required for proxy table. The amount of space required by each table has been summarised in Table 30 and is illustrated further Figure 31 in a form of a bar chart. As we mentioned earlier, multi-valued table scheme only required small amount of the space for storage. The formula used to calculate for the space requirement is as follows:

$$\textit{Percentage of space requirement} = \frac{size\ of\ proxy\ map}{size\ of\ subtable}\ x\ 100$$

Size of proxy map = number of instances calculated in between two attributes

Size of sub-table = number of values (data) found in a particular table

Table 29. Percentage of proxy table space requirement

Table	Multi-valued table (number of instances)	Size of sub-table	Domain size in Pivot table for Proxy	Percentage of Space Requirement (%)
AE_G	747	4338	24	17.22
AE_H	776	4338	53	17.89
AE_I	819	4308	101	19.01
AE_J	814	3942	157	20.65
AE_K	719	3012	217	23.87
AE_L	232	840	92	27.62
AE_F	1440	4326	719	33.29

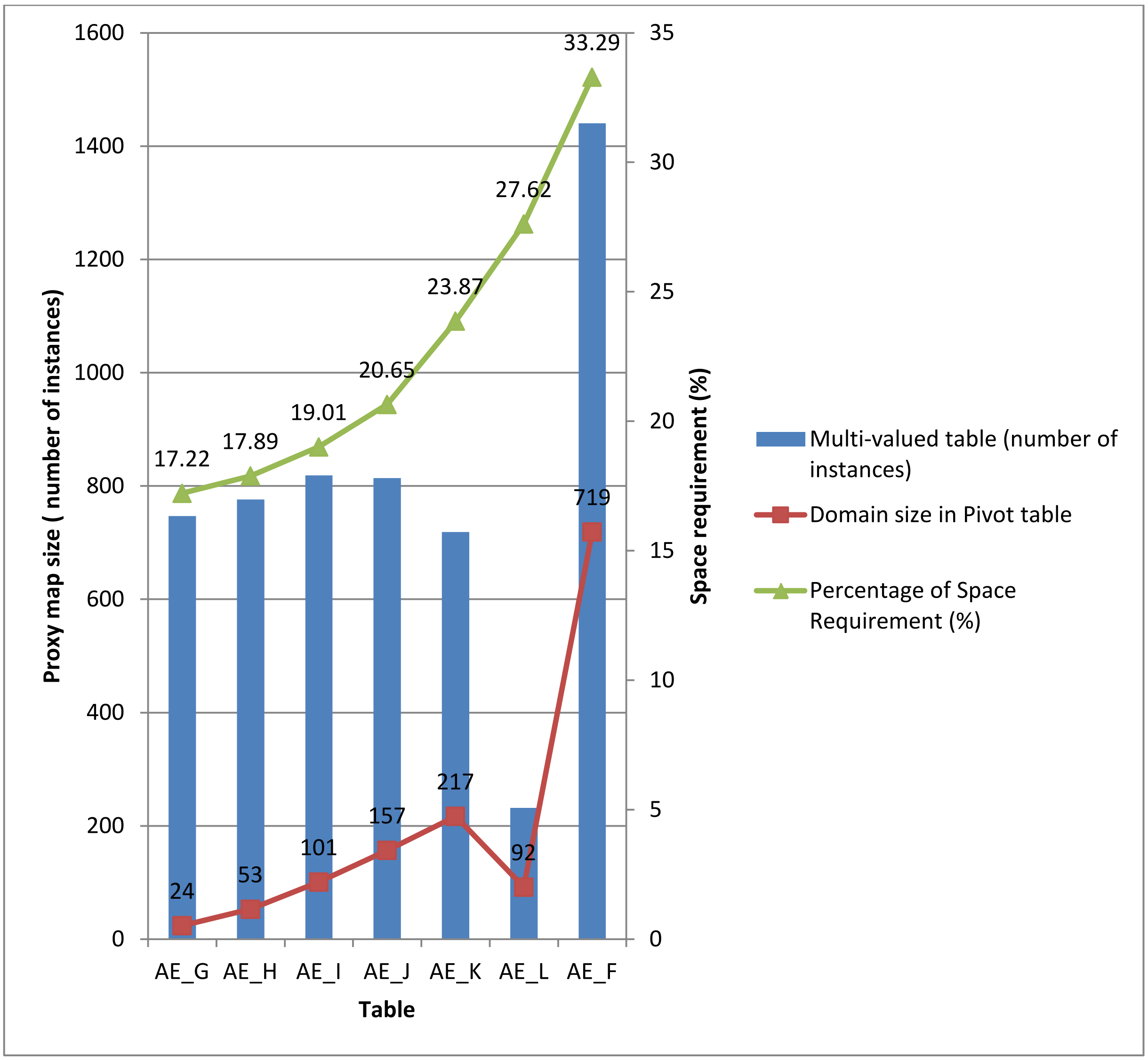


Figure 31. Total space required by all proxies in Taxon sub-tables

However based on Figure 31, we can conclude that, the higher is the number of instances of multi-valued table, the higher is the space requirement percentage. Here table AE_F required the largest amount of space (33.29%) to store proxies information. Even though the multi-valued table scheme used can produced less space requirement as compared to the pure relational scheme, the size of the attribute domain influences the amount of space required to stire the proxy maps. Hence, this is the worst case scenario, where the space requirement for this table is the highest as compared to other tables.

On the other hand, table AE_G shows the best case scenario with lowest proxy map size that requires about 17.22% of space. This can be explained by the lowest domain size which is 24 that demands small amount of space for storing the proxy maps.

The domain size for proxy H in this table is 53 which higher than proxy G. Table AE_H required the second smallest amount of space, which ic about 17.89% of space for storage. Table AE_I needed about 19.01% of space; table AE_J needs 20.65% of space; table AE_K required about 23.87% of space. Table AE_L plots the second highest space requirement about 27.62%.

5.4 Conclusions

To conclude, the analysis results shows that not all FD-based proxy candidates are defective. This can be proved by the G3 error values shown in the Figure 24-30. In addition, it also support for the determination of space requirement analysis, where the number of repeating tuple pairs has been reduced. Therefore, table AE_G shows best case scenario since it demonstrate very low space requirement. As a result, proxy candidates in table AE_G are the good proxies where they helping to provide small space for storage. However, only table AE_F needed higher space since its domain size is the highest. When the domain size is increasing, then requirement of space for storage increases as well.

CHAPTER 6

CONCLUSIONS

In this research we set to analyse the characteristics of proxies and the requirements to predict missing values in data set. To achieve the objectives we use functional dependency analysis to find the candidate proxies for attributes that have large number of null values. By having good proxies, we can predict the missing values with some acceptable levels of accuracy.

We identified one important requirement to implement proxies in handling missing values prediction which is additional space requirement. This space requirement is needed to store the proxy maps in particular in handling missing values problems. Even though the results of the analyses are based on case studies of biological domains, it may be used in other application domains.

From the results, we can conclude that, attributes A and B (in Table AE_G) are good proxy candidates as the possess both high accuracy and proxy map space requirement charactersitics. The characteristic of good proxy shows high accuracy in FDs discovery (low G3 errors) and low storage space requirement. Attribute A shows 100% of FD accuracy as this is the key attribute for the Taxon table. Attribute B is also a good proxy with high FD accuracy. Thus attribute B can used as the alternative of the droppable attribute to substitute the missing values in the sample data set. The charactristic of poor proxies can be seen from proxy candidates D and E's performance. They produced unstable G3 errors when there are changes made on the G3 ranges. Attribute D and E do not act as good proxies because their FD prediction accuracy is very low. Thus they should not be used in missing values prediction.

Table AE_G require small amount of space as compared to the other tables, since it has good proxy candidates with higher FD accuracy existence. However, even though AE_F has good proxies, requires higher space for proxy maps storage as the domain size of attribute F is bigger as compared to other proxy candidates'.

In the future, the implementation of proxies in missing values prediction analysis will be performed in order to apply the outcome of this research. Exploration of the characteristics of proxies should be expanded in other domain data sets in order to compare and verify the findings in this research.

REFERENCES

Anon., 2002. Microbial Genomics. In: *Microbiology.* 5 ed. s.l.:The McGraw-Hill Companies.

Apiletti, D., Bruno, G., Ficarra, E. & Baralis, E., 2006. Data Cleaning and Semantic Improvement in Biological Databases.. *Journal of Integrative Bioinformatics.*

D.Peterson, J. et al., 2001. The Comprehensive Microbial Resource. *Nucleic Acids Research,* November, Volume 29, pp. 123-125.

Davidsen, T. et al., 2009. The comprehensive microbial resource. *Nucleic Acids Research,* November.pp. 1-6.

Emran, N. A., Abdullah, N. & Isa, M. N. M., 2012. *Storage Space Optimisation for Green Data Center.* Malaysia, s.n.

Freeman, L., 2007. Looking Beyond the Hype: Evaluating Data Deduplication Solutions. *http://www.techrepublic.com/whitepapers/looking-beyond-the-hype-evaluating-data-deduplication-solutions/1294015.*

George, B., Ilyas, I. F. & Golab, L., 2010. *Sampling the Repairs of Functional Dependency Violations under Hard Constraints..* Singaapore, s.n., pp. 197-207.

Giannella, C. M., Dalkilic, M. M., Groth, D. P. & Robertson, E. L., 2002. Improving Query Evaluation with Approximate Functional Dependency Based Decompositions. *LNCS 2405,* pp. 26-41.

Hazelhurst, S., 2008. *Scientific computing using virtual high-performance computing: A case study using the amazon elastic computing cloud.* s.l., ACM.

Herrero, J., D´ıaz-Uriarte, R. & Dopazo, J., 2003. Gene expression data preprocessing. *Journal of Bioinformatics Vol. 19 no. 5,* pp. 655-656.

Horn, G. & Cook, J. V., 2011. *How dirty is your data? a look at the energy choices that power cloud,* s.l.: Greenpeace International.

Huhtala, Y., Karkkainen, J., Porkka, P. & Toivonen, H., 1999. TANE: An efficeint algorithm for discovering Functional and Approximate Dependencies. *The Computer Journal,* March, 31st, Volume 2, pp. 100-111.

J.Wiens, J. & C.Morrill, M., 2011. Missing Data in Phylogenetic Analysis: Reconciling Results from Simulations and Empirical Data.. *Systematic Biology Advance Access. Vol.60.,* pp. 1-13.

Jombart, T., 2012. Analysing genome-wide SNP data using adegenet.. pp. 1-37.

Kang, J. H. K. et al., 1990. *Feature-oriented domain analysis (FODA) feasibility study,* s.l.: s.n.

Kolahi, S. & Lakshmanan, L. V. S., 2009. On Approximating Optimum Repairs for Functional Dependency Violations.. *ACM,* pp. 53-62.

Kumar, V., 1992. Algorithm for constraints-satisfaction problems: A survey. *AI Magazine*, pp. 32-44.

Lai, E., 2008. Oracle Pushes Compression as Cheaper Database Scale-Up Method. *Computerworld White Paper*.

Liu, J., Li, J., Liu, C. & Chen, Y., 2012. Discover Dependencies from Data—A Review.. *IEEE transactions on knowledge and data engineering, vol. 24, no. 2.,* pp. 251-264.

Li, Z. et al., 2012. Simultaneous SNP Identification In Association Studies With Missing Data.. *The Annals of Applied Statistics, Vol. 6, No. 2,* p. 432–456.

Markowitz, V. M. et al., 2005. *The Integrated Microbial Genomes (IMG) System: A Case Study in Biological Data Management.* Trondheim, Norway, VLDB.

Molinaro, C. & Greco, S., 2010. Polynomial time queries over inconsistent databases with functional dependencies and foreign keys.. *Data & Knowledge Engineering 69,* p. 709–722.

Petrik, K., 2009. *Participation and e-democracy how to utilize web 2.0 for policy decision-making.* s.l., Digital Government Society of North America.

Pevsner, J., 2009. *Bioinformatics and Functional Genomics.* 2nd Edition ed. United States of America: A John Wiley & Sons, Insc..

Tuikkala, J., Elo, L. L., Nevalainen, O. S. & Aittokallio, T., 2008. Missing value imputation improves clustering and interpretation of gene expression microarray data... *BMC Bioinformatics, 9:202,* pp. 1-14.

Wiens, J. J., 2006. Missing data and the design of phylogenetic analyses.. *Journal of Biomedical Informatics 39,* p. 34–42.

Yao, H., J.Hamilton, H. & J.Butz, C., n.d. FD_Mine: Discovering Functional Dependencies in a Database Using Equivalences.

Yeh, P. Z. & Puri, C. A., 2010. *Discovering Conditional Functional Dependencies to Detect Data Inconsistencies.* Singapore, s.n., pp. 1-7.

APPENDICES

APPENDIX A

Sample input file for TANE algorithm (comma separated values file)

```
207,243159,Bacteria,Acidithiobacillus,ferrooxidans,Acidithiobacillus ferrooxidans
590,351607,Bacteria,Acidothermus,cellulolyticus,Acidothermus
202,240017,Bacteria,Actinomyces,naeslundii,Actinomyces
405,290397,Bacteria,Anaeromyxobacter,dehalogenans,Anaeromyxobacter
199,212042,Bacteria,Anaplasma,phagocytophilum,phagocytophilum group
326,290340,Bacteria,Arthrobacter,aurescens,Arthrobacter
654,1667,Bacteria,Arthrobacter,sp.,Arthrobacter
407,322098,Bacteria,Aster,yellows,Candidatus Phytoplasma asteris
803,553184,Bacteria,Atopobium,rimae,Atopobium
275,288681,Bacteria,Bacillus,cereus,group
250,283166,Bacteria,Bartonella,henselae,Bartonellaceae
236,283165,Bacteria,Bartonella,quintana,Bartonellaceae
187,264462,Bacteria,Bdellovibrio,bacteriovorus,Bdellovibrio bacteriovorus
734,94624,Bacteria,Bordetella,petrii,Alcaligenaceae
622,339670,Bacteria,Burkholderia,ambifaria,Burkholderia cepacia complex
584,398577,Bacteria,Burkholderia,ambifaria,Burkholderia cepacia complex
425,331271,Bacteria,Burkholderia,cenocepacia,Burkholderia cepacia complex
541,331272,Bacteria,Burkholderia,cenocepacia,Burkholderia cepacia complex
551,395019,Bacteria,Burkholderia,multivorans,Burkholderia cepacia complex
563,395019,Bacteria,Burkholderia,multivorans,Burkholderia cepacia complex
761,357348,Bacteria,Burkholderia,pseudomallei,pseudomallei
665,357347,Bacteria,Burkholderia,pseudomallei,pseudomallei group
400,320372,Bacteria,Burkholderia,pseudomallei,pseudomallei group
507,320373,Bacteria,Burkholderia,pseudomallei,pseudomallei group
146,272560,Bacteria,Burkholderia,pseudomallei,pseudomallei group
529,269482,Bacteria,Burkholderia,vietnamiensis,Burkholderia cepacia complex
31,192222,Bacteria,Campylobacter,jejuni,Campylobacter jejuni
201,195099,Bacteria,Campylobacter,jejuni,Campylobacter jejuni
334,291272,Bacteria,Candidatus,Blochmannia,Candidatus Blochmannia
200,246194,Bacteria,Carboxydothermus,hydrogenoformans,Carboxydothermus hydrogenoformans
369,340177,Bacteria,Chlorobium,chlorochromatii,Chlorobium
628,443906,Bacteria,Clavibacter,michiganensis,Clavibacter
764,31964,Bacteria,Clavibacter,michiganensis,Clavibacter
799,553204,Bacteria,Corynebacterium,amycolatum,Corynebacterium
191,257309,Bacteria,Corynebacterium,diphtheriae,Corynebacterium
120,196164,Bacteria,Corynebacterium,efficiens,Corynebacterium
307,196627,Bacteria,Corynebacterium,glutamicum,Corynebacterium
308,306537,Bacteria,Corynebacterium,jeikeium,Corynebacterium
797,553207,Bacteria,Corynebacterium,matruchotii,Corynebacterium
798,553206,Bacteria,Corynebacterium,tuberculostearicum,Corynebacterium
336,255470,Bacteria,Dehalococcoides,sp.,Dehalococcoides
203,246195,Bacteria,Dichelobacter,nodosus,Dichelobacter nodosus
197,205920,Bacteria,Ehrlichia,chaffeensis,Ehrlichia chaffeensis
472,362663,Bacteria,Escherichia,coli,Escherichia coli
206,59374,Bacteria,Fibrobacter,succinogenes,Fibrobacter succinogenes subsp. succinog
440,326424,Bacteria,Frankia,alni,Frankia
409,106370,Bacteria,Frankia,sp.,Frankia
698,1855,Bacteria,Frankia,sp.,Frankia
175,114,Bacteria,Gemmata,obscuriglobus,Gemmata obscuriglobus
168,233412,Bacteria,Haemophilus,ducreyi,Haemophilus ducreyi
309,281310,Bacteria,Haemophilus,influenzae,Pasteurellaceae
8,71421,Bacteria,Haemophilus,influenzae,Haemophilus influenzae
155,235279,Bacteria,Helicobacter,hepaticus,Helicobacter hepaticus
5,85962,Bacteria,Helicobacter,pylori,Helicobacter pylori
24,85963,Bacteria,Helicobacter,pylori,Helicobacter pylori
```

```
173,81032,Bacteria,Hyphomonas,neptunium,Hyphomonas
684,266940,Bacteria,Kineococcus,radiotolerans,Kineococcus
259,272624,Bacteria,Legionella,pneumophila,Legionellaceae
235,281090,Bacteria,Leifsonia,xyli,Micrococcineae
214,267377,Archaea,Methanococcus,maripaludis,Methanococcus maripaludis
172,243233,Bacteria,Methylococcus,capsulatus,Methylococcus capsulatus
383,264732,Bacteria,Moorella,thermoacetica,Moorella
765,561007,Bacteria,Mycobacterium,abscessus,Mycobacterium
193,174277,Bacteria,Mycobacterium,avium,Mycobacterium
616,410289,Bacteria,Mycobacterium,bovis,Mycobacterium
611,350054,Bacteria,Mycobacterium,gilvum,Mycobacterium
209,246196,Bacteria,Mycobacterium,smegmatis,Mycobacterium
600,164757,Bacteria,Mycobacterium,sp.,Mycobacterium
649,189918,Bacteria,Mycobacterium,sp.,Mycobacterium
464,164756,Bacteria,Mycobacterium,sp.,Mycobacterium
656,336982,Bacteria,Mycobacterium,tuberculosis,Mycobacterium
618,419947,Bacteria,Mycobacterium,tuberculosis,Mycobacterium
603,350058,Bacteria,Mycobacterium,vanbaalenii,Mycobacterium
311,262722,Bacteria,Mycoplasma,hyopneumoniae,Mycoplasmataceae
213,246197,Bacteria,Myxococcus,xanthus,Myxococcus
349,348780,Archaea,Natronomonas,pharaonis,Halobacteriaceae
198,222891,Bacteria,Neorickettsia,sennetsu,Neorickettsia sennetsu
271,247156,Bacteria,Nocardia,farcinica,Nocardia
576,35761,Bacteria,Nocardioides,sp.,Nocardioides
672,357244,Bacteria,Orientia,tsutsugamushi,Orientia
385,319225,Bacteria,Pelodictyon,luteolum,Pelodictyon
169,243265,Bacteria,Photorhabdus,luminescens,Photorhabdus luminescens
188,100379,Bacteria,Phytoplasma,asteris,16SrI (Aster yellows group)
176,246198,Bacteria,Prevotella,intermedia,Prevotella intermedia
212,264731,Bacteria,Prevotella,ruminicola,Prevotella ruminicola
170,167539,Bacteria,Prochlorococcus,marinus,Prochlorococcus marinus subsp. marinus
314,59920,Bacteria,Prochlorococcus,marinus,Prochlorococcus
239,267747,Bacteria,Propionibacterium,acnes,Propionibacterium
787,553199,Bacteria,Propionibacterium,acnes,Propionibacterium
785,553197,Bacteria,Propionibacterium,sp.,Propionibacterium
598,290318,Bacteria,Prosthecochloris,vibrioformis,Chlorobium
36,208964,Bacteria,Pseudomonas,aeruginosa,Pseudomonas aeruginosa
118,220664,Bacteria,Pseudomonas,fluorescens,Pseudomonas fluorescens
119,205922,Bacteria,Pseudomonas,fluorescens,Pseudomonas fluorescens
55,160488,Bacteria,Pseudomonas,putida,Pseudomonas putida
89,223283,Bacteria,Pseudomonas,syringae,Pseudomonas syringae group genomosp. 3
315,205918,Bacteria,Pseudomonas,syringae,Pseudomonadales
190,103985,Bacteria,Pseudomonas,syringae,Pseudomonas savastanoi
509,288705,Bacteria,Renibacterium,salmoninarum,Renibacterium
422,347834,Bacteria,Rhizobium,etli,Rhizobium
613,216596,Bacteria,Rhizobium,leguminosarum,Rhizobium
340,272943,Bacteria,Rhodobacter,sphaeroides,Rhodobacteraceae
786,596309,Bacteria,Rhodococcus,erythropolis,Rhodococcus
474,101510,Bacteria,Rhodococcus,sp.,Rhodococcus
189,258594,Bacteria,Rhodopseudomonas,palustris,Rhodopseudomonas palustris
725,293614,Bacteria,Rickettsia,akari,Rickettsia
692,391896,Bacteria,Rickettsia,bellii,Rickettsia
432,336407,Bacteria,Rickettsia,bellii,Rickettsia
719,293613,Bacteria,Rickettsia,canadensis,Rickettsia
316,315456,Bacteria,Rickettsia,felis,Rickettsia
724,416276,Bacteria,Rickettsia,massiliae,Rickettsia
740,452659,Bacteria,Rickettsia,rickettsii,Rickettsia
693,392021,Bacteria,Rickettsia,rickettsii,Rickettsia
245,257363,Bacteria,Rickettsia,typhi,Rickettsia
783,553201,Bacteria,Rothia,mucilaginosa,Rothia
```

```
469,266117,Bacteria,Rubrobacter,xylanophilus,Rubrobacter
174,246200,Bacteria,Ruegeria,pomeroyi,Silicibacter pomeroyi
651,405948,Bacteria,Saccharopolyspora,erythraea,Saccharopolyspora
325,309807,Bacteria,Salinibacter,ruber,Sphingobacteriales
700,391037,Bacteria,Salinispora,arenicola,Salinispora
612,369723,Bacteria,Salinispora,tropica,Salinispora
295,321314,Bacteria,Salmonella,enterica,Salmonella enterica subsp. enterica serovar Choleraesuis
159,209261,Bacteria,Salmonella,enterica,Salmonella enterica
763,454169,Bacteria,Salmonella,enterica,Salmonella enterica subsp. enterica serovar Heidelberg
731,272994,Bacteria,Salmonella,enterica,Salmonella enterica subsp. enterica serovar Paratyphi B
51,211586,Bacteria,Shewanella,oneidensis,Shewanella oneidensis MR-1
160,198215,Bacteria,Shigella,flexneri,Shigella flexneri
127,198214,Bacteria,Shigella,flexneri,Shigella flexneri 2a
663,366394,Bacteria,Sinorhizobium,medicae,Sinorhizobium
317,342451,Bacteria,Staphylococcus,saprophyticus,Staphylococcaceae
306,170187,Bacteria,Streptococcus,pneumoniae,Streptococcaceae
134,218496,Bacteria,Tropheryma,whipplei,Tropheryma
133,203267,Bacteria,Tropheryma,whipplei,Tropheryma
166,196600,Bacteria,Vibrio,vulnificus,Vibrio vulnificus
234,80849,Bacteria,Wolbachia,pipientis,Wolbachia
321,314565,Bacteria,Xanthomonas,campestris,Xanthomonadaceae
218,229193,Bacteria,Yersinia,pestis,Yersinia pestis
248,273123,Bacteria,Yersinia,pseudotuberculosis,Enterobacteriaceae
261,264203,Bacteria,Zymomonas,mobilis,Sphingomonadaceae
```

APPENDIX B1

Sample Multi-valued table output of pivot table for table AE_G (G to A)

Row Labels Intermediate_rank_1 (G)	Count of u_id (A)
1. Actinobacteria	56
2. Aquificae	1
3. Bacteroidetes	17
4. Chlamydiae	13
5. Chlorobi	4
6. Chloroflexi	6
7. Crenarchaeota	13
8. Cyanobacteria	30
9. Deinococcus-Thermus	4
10. dsDNA viruses, no RNA stage	2
11. Euryarchaeota	33
12. Fibrobacteres	1
13. Fibrobacteres/Acidobacteria group	1
14. Firmicutes	164
15. Fusobacteria	2
16. Korarchaeota	1
17. Nanoarchaeota	1
18. Planctomycetes	2
19. Proteobacteria	341
20. Spirochaetes	15
21. ssRNA + strand viruses, no DNA stage	1
22. Tenericutes	8
23. Thermomicrobia	1
24. Thermotogae	6
Grand Total	**723**

APPENDIX B2

Sample Multi-valued table output of pivot table for table AE_G (G to B)

Row Labels Intermediate_rank_1 (G)	Count of taxon_id (B)
1. Actinobacteria	56
2. Aquificae	1
3. Bacteroidetes	17
4. Chlamydiae	13
5. Chlorobi	4
6. Chloroflexi	6
7. Crenarchaeota	13
8. Cyanobacteria	30
9. Deinococcus-Thermus	4
10. dsDNA viruses, no RNA stage	2
11. Euryarchaeota	33
12. Fibrobacteres	1
13. Fibrobacteres/Acidobacteria group	1
14. Firmicutes	164
15. Fusobacteria	2
16. Korarchaeota	1
17. Nanoarchaeota	1
18. Planctomycetes	2
19. Proteobacteria	341
20. Spirochaetes	15
21. ssRNA + strand viruses, no DNA stage	1
22. Tenericutes	8
23. Thermomicrobia	1
24. Thermotogae	6
Grand Total	**723**

APPENDIX B3

Sample Multi-valued table output of pivot table for table AE_G (G to D)

Row Labels Intermediate_rank_1 (G)	Count of Genus (D)
1. Actinobacteria	56
2. Aquificae	1
3. Bacteroidetes	17
4. Chlamydiae	13
5. Chlorobi	4
6. Chloroflexi	6
7. Crenarchaeota	13
8. Cyanobacteria	30
9. Deinococcus-Thermus	4
10. dsDNA viruses, no RNA stage	2
11. Euryarchaeota	33
12. Fibrobacteres	1
13. Fibrobacteres/Acidobacteria group	1
14. Firmicutes	164
15. Fusobacteria	2
16. Korarchaeota	1
17. Nanoarchaeota	1
18. Planctomycetes	2
19. Proteobacteria	341
20. Spirochaetes	15
21. ssRNA + strand viruses, no DNA stage	1
22. Tenericutes	8
23. Thermomicrobia	1
24. Thermotogae	6
Grand Total	**723**

APPENDIX B4

Sample Multi-valued table output of pivot table for table AE_G (G to E)

Row Labels Intermediate_rank_1 (G)	Count of Species (E)
1. Actinobacteria	56
2. Aquificae	1
3. Bacteroidetes	17
4. Chlamydiae	13
5. Chlorobi	4
6. Chloroflexi	6
7. Crenarchaeota	13
8. Cyanobacteria	30
9. Deinococcus-Thermus	4
10. dsDNA viruses, no RNA stage	2
11. Euryarchaeota	33
12. Fibrobacteres	1
13. Fibrobacteres/Acidobacteria group	1
14. Firmicutes	164
15. Fusobacteria	2
16. Korarchaeota	1
17. Nanoarchaeota	1
18. Planctomycetes	2
19. Proteobacteria	341
20. Spirochaetes	15
21. ssRNA + strand viruses, no DNA stage	1
22. Tenericutes	8
23. Thermomicrobia	1
24. Thermotogae	6
Grand Total	**723**