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 → 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* → *reg, cnt, reg* → *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)

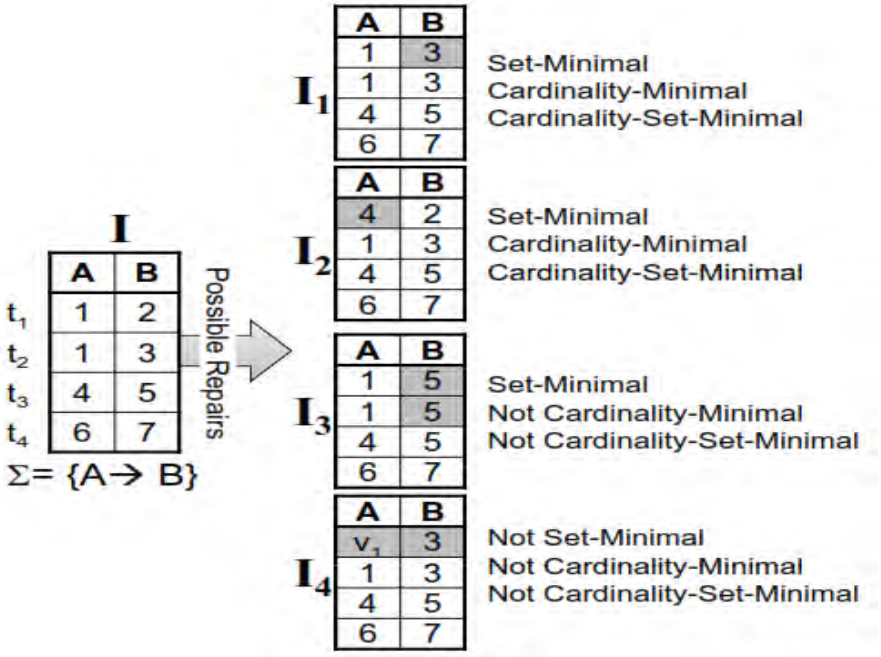

Figure 4. Example of various types of repairs. (George, et al., 2010)

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 →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* → *Manager defined over Project*,
- fd_2 : *Name* → *Phone defined over Employee*,
- *fk* : *Project [Manger]* ⊆ *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 ⊥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 ⊥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,