



An Operator for Data Error Detection

By

Ms. Amithra Ruck

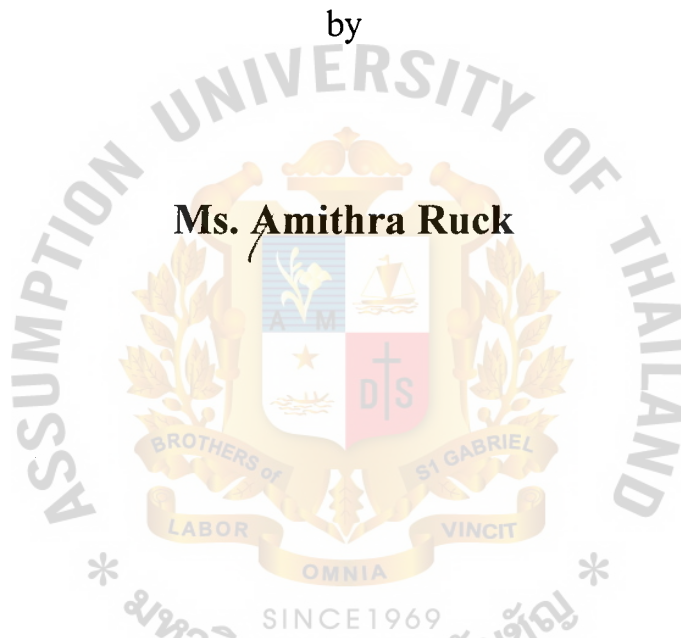
Submitted in Partial Fulfillment of
the Requirements for the Degree of
Master of Science
in Information Technology
Assumption University

September, 2000

An Operator for Data Error Detection

by

Ms. Amithra Ruck



**Submitted in Partial Fulfillment of
the Requirements for the Degree of
Master of Science
in Information Technology
Assumption University**

September, 2000

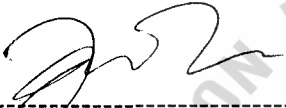
The Faculty of Science and Technology

Thesis Approval

| | |
|----------------|--------------------------------------|
| Thesis Title | An Operator for Data Error Detection |
| By | Amithra Ruck |
| Thesis Advisor | Dr.Jirapun Daengdej |

The Department of IT, the Faculty of Science and Technology of Assumption University had approved this final report of the **twelve** credits course. **IT7000 Master Thesis**, submitted in partial fulfillment of the requirements for the degree of Master of Science in Information Technology.

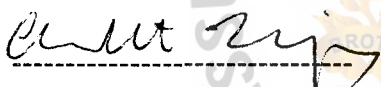
Approval Committee:




(Dr. Jirapun Daengdej)
Advisor



(Dr. Thotsapon Sortrakul)
Committee



(Dr. Thiraphong Charoenkhunwiwat)
Committee




(Asst.Prof.Dr. Somnuk Keretho)
Representative of Ministry
of University Affairs

Faculty Approval:



(Dr. Thotsapon Sortrakul)
Director



(Asst.Prof.Dr. Pratit Santiprabhob)
Dean

July/ 2000

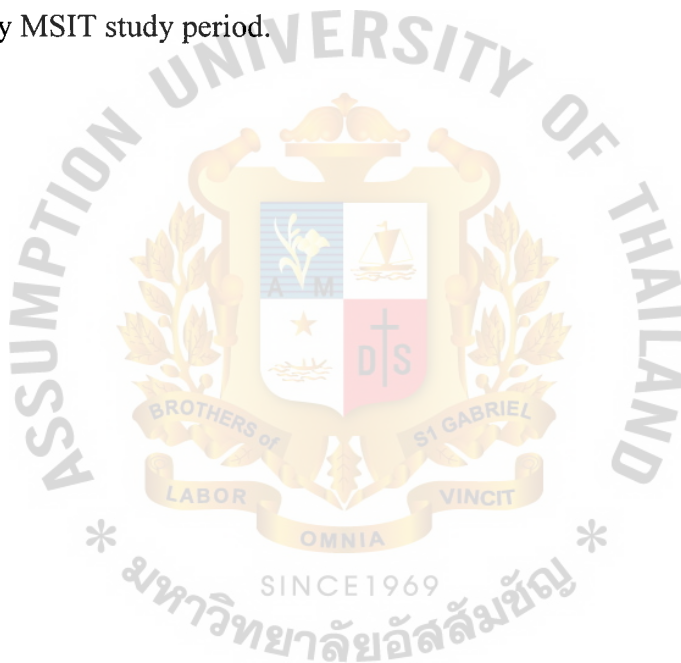
Month/Year

ACKNOWLEDGMENTS

I desire to express my most sincere appreciation and thanks to Dr. Jirapun Daengdej, my thesis advisor, for the many useful comments, invaluable advice and encouragement during the development step and later during the writing step. I am grateful to Dr. Thotsapon Sortrakul and Dr. Thiraphong Charoenkhunwiwat for giving good suggestions to improve my thesis.

Further, I would like to thank my friends for their cheerful and friendly nature.

Finally, very special thanks are due to my family for their encouragement and support during my MSIT study period.



ABSTRACT

This thesis provides the SQL-like operators for handling data error detection and discusses the important of data quality in KDD processes.

The proposed error detection method uses domain knowledge that is prior defined by the domain expert. Additionally, we also proposed the interactive user interface, in which the user can always certain that the result of error detection is complete because the users can closely monitor the process.

The main advantage of this approach is to prevent users from having to remember and use long and complicated SQL statement in order to do the same task.



St. Gabriel's Library
TABLE OF CONTENTS

| | |
|--|----|
| ACKNOWLEDGEMENTS | i |
| ABSTRACT | ii |
| LIST OF FIGURES | v |
| LIST OF TABLES | vi |
| CHAPTER 1 INTRODUCTION | 1 |
| 1.1 Introduction | 1 |
| 1.2 Objectives and Scope of Thesis | 2 |
| 1.3 Outline of the Thesis | 2 |
| CHAPTER 2 KNOWLEDGE DISCOVERY IN DATABASE | 3 |
| 2.1 The KDD Process | 3 |
| 2.2 Data Quality Problems | 5 |
| 2.3 Cleaning Approaches in KDD | 7 |
| 2.3.1 ESTEEM | 7 |
| 2.3.2 The Basic Sorted-Neighborhood Method | 7 |
| CHAPTER 3 LITERATURE REVIEW | 9 |
| 3.1 Cooperative Query Answers (CQA) | 9 |
| 3.1.1 Fundamentals of the CQA | 9 |
| 3.1.2 Domain Knowledge | 9 |
| 3.1.3 Cooperative Operators (CoSQL) | 11 |
| 3.1.4 Query Relaxation | 12 |
| 3.2 Data Extraction: Interfacing with a Database | 13 |

| | |
|---|----|
| 3.3 A Data Mining Query Language (DMQL) | 14 |
| CHAPTER 4 THE PROPOSED OPERATOR | 16 |
| 4.1 Required Environment | 16 |
| 4.2 Different Type of Domain Knowledge | 16 |
| 4.2.1 Hierarchy | 17 |
| 4.2.2 Rules | 17 |
| 4.3 Syntax of Data Error Detection Operator | 18 |
| 4.4 Control Flow Chart of Data Error Detection Operator | 20 |
| 4.5 Examples of Data Error Detection | 21 |
| 4.6 The Interactive User Interface | 25 |
| CHAPTER 5 EXPERIMENTAL AND EVALUATION | 30 |
| 5.1 Environment of Examination | 31 |
| 5.2 A Development Database | 31 |
| 5.3 Experiment Methods | 32 |
| 5.4 Result Analysis | 34 |
| CHAPTER 6 DRAWBACKS AND FUTURE RESEARCH | 36 |
| CHAPTER 7 CONCLUSIONS | 37 |
| BIBLIOGRAPHY | 38 |

LIST OF FIGURES

| | | |
|-------------|---|----|
| Figure 2-1 | An Overview of the KDD process | 5 |
| Figure 3-1 | Type Abstract Hierarchy (runway_length) | 10 |
| Figure 3-2 | Proposed method for mining from SQL queries | 14 |
| Figure 4-1 | Data Error Detection Control Flow Chart | 20 |
| Figure 4-2 | Domain Knowledge – RegionDomain | 23 |
| Figure 4-3 | The interactive user interface | 25 |
| Figure 4-4 | User input DETECT Command | 27 |
| Figure 4-5 | The modified SQL command and result will be shown | 28 |
| Figure 4-6 | Show suggestion for Student ID attribute | 28 |
| Figure 4-7 | Show suggestion for GPA attribute | 28 |
| Figure 4-8 | Show suggestion for SubDistrict attribute | 29 |
| Figure 4-9 | Show suggestion for District attribute | 29 |
| Figure 4-10 | Show suggestion for Province attribute | 29 |
| Figure 4-11 | Show complete updated message | 29 |
| Figure 5-1 | Data error detection time in various conditions | 35 |

LIST OF TABLES

| | | |
|-----------|---|----|
| Table 3-1 | Relaxation Range for the Approximate Operator | 12 |
| Table 4-1 | Domain Knowledge – StdRule | 22 |
| Table 5-1 | Summary of Experiment Cases | 32 |
| Table 5-2 | Summary of Experiments and Results | 34 |



CHAPTER 1 INTRODUCTION

1.1 Introduction

The decade of the 1990s has brought a growing data glut problem to the worlds of science, business, and government. Our capabilities for collecting and storing data of all kinds have far outpaced our abilities to analyze, summarize, and extract “knowledge” from this data.

Traditional methods of data analysis base mainly on the human dealing directly with the data. While database technology has provided us with the basic tools for the efficient storage and lookup of large data sets. Unfortunately, the issues of how to help humans understand and analyze large data sets remain a difficult and unsolved problem. In order to deal with the large data sets, a new generation of intelligent tools for automated data mining and knowledge is needed [1].

Knowledge Discovery and Data Mining is an area of common interest to researches in several fields, including machine learning, statistics, database, and data visualization. The continuing rapid growth of on-line data has created a need and an opportunity for extracting knowledge from databases. Responding to this need, researchers and application developers have created knowledge discovery applications for many areas of businesses and sciences.

KDD aims at the discovery of useful information from large collections of data. The large databases contain many errors such as *noisy*, *erroneous missing* or *irrelevant data*. Whenever these errors exist in a corporate data, it can easily lead to problems such as shipping or receiving wrong goods. The problem can also occur with data analysis by leading to the wrong result, regardless of how good the analysis algorithm is. Unfortunately, the cleaning process is very costly and can consume large

amount of time. The main objective of data-quality improvement is to minimize costs, problems, and missed opportunities caused by non-quality data.

1.2 Objective and Scope of Thesis

In this thesis we consider the data cleaning process, which is one step in KDD processes. The proposed SQL-like operators can be used with a large database, and similar to the conventional SQL operators, the operators allow fast and efficient data retrieval and manipulation.

1.3 Outline of the Thesis

The rest of this thesis is organized as follows. Chapter 2 will discuss the KDD processes and explain the important of data quality in current businesses. Then, chapter 3 will discuss the literature review. Additionally, chapter 4 will describe the philosophy of the proposed operators for data error detection. Chapter 5 will demonstrate the experimental and evaluation process, and the outcome of the proposed operators. Chapter 6 will discuss drawbacks of the proposed operators together with future research. This concluded by conclusion of the thesis direction in chapter 7.

CHAPTER 2 KNOWLEDGE DISCOVERY IN DATABASE

KDD or data mining is the iterative process for discovering interesting knowledge, such as patterns, associations, changes, anomalies and significant structures, from large amounts of data stored in database, data warehouses, or other information repositories. However, it is widely known that the process of KDD requires interaction with the user [10, 11].

2.1 The KDD Process

In Figure 2-1, the KDD process consists of several steps and iteration between them, sequence of the KDD process is:

First is developing an understanding of the application domain and the relevant prior knowledge and identifying the goal of the KDD process from the customer's viewpoint.

Second is creating a target data set: selecting and extracting a data set, or focusing on a subset of variables or data samples, on which discovery is to be performed.

Third is data cleaning and preprocessing. Basic operations include removing noise if appropriate, collecting the necessary information to model or account for noise, deciding on strategies for handling missing data fields, and accounting for time-sequence information and known changes. The errors could come from many sources of error when merging multiple databases [3,12].

Fourth is data reduction and projection: finding useful features to represent the data depending on the goal of the task. With dimensionality reduction or transformation methods, the effective number of variables under consideration can be reduced, or invariant representations for the data can be found.

Fifth is matching the goals of the KDD process (step1) to a particular data-mining method. Examples of data mining method are summarization, classification, regression, clustering, and so on.

Sixth is exploratory analysis and model and hypothesis selection: choosing the data-mining algorithm(s) and selection method(s) to be used for searching for data patterns. This process includes deciding which models and parameters might be appropriate and matching a particular data-mining method with the overall criteria of the KDD process.

Seventh is data mining: searching for patterns of interest in a particular representational form or a set of such representations, including classification rules or trees, regression, and clustering. The user can significantly aid the data-mining method by correctly performing the preceding steps.

Eighth is interpreting mined patterns, possibly returning to any of steps 1 through 7 for further iteration. This step can also involve visualization of the extracted patterns and models or visualization of the data given the extracted models.

Ninth is acting on the discovered knowledge: using the knowledge directly, incorporating the knowledge into another system for further action, or simply documenting it and reporting it to interested parties. This process also includes checking for and resolving potential conflicts with previously believed (or extracted) knowledge.

In Figure 2-1, the KDD process can involve significant iteration and can contain loops between any two steps.

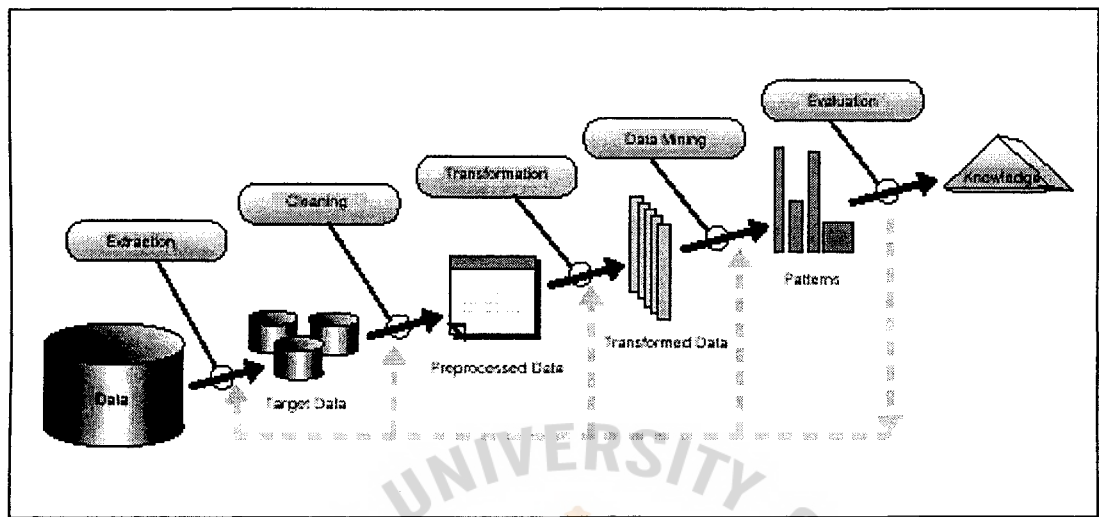


Figure 2-1: An Overview of the KDD process

2.2 Data Quality Problems

Currently, data or information has become one of the most valuable corporate assets. It is the foundation of knowledge and it is used and reused in various business intelligence applications to support sophisticated analysis and decision-making processes which enable company to be more competitive. However, the value of data is mainly concerned upon its *quality*. It is important to notice that the quality of data can influence quality of decisions. Regardless of algorithms for data analysis or deriving of knowledge, quality of the result depends solely on quality of data. This issue has been widely known as “*Garbage in, Garbage out*” [23].

As the size and complexity of databases grow, two things begin to raise ["Quality Unbound", by Kamran Parsaye and Mark Chignell]:

- We rely more and more on the database which we will soon become fully dependent on it.
- As the number of errors in the data increases, these errors become harder to trace.

Almost anyone who works with database is well informed of the great number of errors that can occur in database. These errors are the results of a range of different factors. In many cases, these errors are caused by faulty data entry. In other cases, errors are made intentionally, such as in cases of fraud. Errors are also sometimes caused by the failure of software or hardware. (“WizRule: A New Approach to Data Cleansing” by Abraham Meidan).

The problem of merging multiple databases of information especially concerning with common entities is frequently encountered in KDD and decision support applications in large commercial and government organization [12]. Instances of this problem appearing in the literature have been called *record linkage* [19], the *semantic integration* problem [20] or the *instance identification* problem [21]. Further, recently, the *data cleaning* problem is regarded as a crucial first step the KDD process [22].

There are many software applications for data cleaning. The results of them are suspected-errors, or at least cases to be examined. These products perform one or more of the following functions:

- **Rule discover:** discover data relationships and rules which describe meaning of the data.
- **Analysis/audit:** analyze data against a set of business rules to discover inconsistencies.

- **Data cleansing/scrubbing:** analyze and standardize data, identify duplicates, and transform data to a correct format or probably, at least, correct values.
- **Defect prevention:** enforce integrity rules at the source of the data.

2.3 Cleaning Approaches in KDD

Data cleaning is a very important step in the KDD process. There are many researchers proposing the approaches for data cleaning. Examples of these approaches appearing in the literature are following.

2.3.1 *ESTEEM (Elimination of Suspicious Training Examples with Error on the Model)*

Jonh has proposed ESTEEM [3] which is a method of data cleaning that uses a data mining algorithm to find patterns in a database, then “*improves the self-esteem*” of the data mining algorithm by removing records from the training database that do not fit the discovered patterns, and retraining.

2.3.2 *The Basic Sorted-Neighborhood Method*

Mauricio A. and Salvatore J. have proposed the basic “Sorted-Neighborhood Method” for solving merge/purge problem [12]. This method can be summarized in three phases:

1. **Create Keys:** Compute a key for each record in the list by extracting relevant fields or portions of fields. The choice of the key depends upon an “error model” that may be viewed as knowledge intensive and domain-specific. The effectiveness of the sorted-neighborhood method highly depends on a properly

chosen key with the intent that common but erroneous data will have closely matching keys.

2. **Sort Data:** Sort the records in the data list using the key of step1.
3. **Merge:** Move a fixed size window through the sequential list of records limiting the comparisons for matching records to those records in the window. If the size of the window is w records, then every new record entering the window is compared with the previous $w-1$ records to find “matching” records. The first record in the window slides out of the window.

This method is expensive due to the sorting phase, as well as the need to search in large windows for high accuracy.



CHAPTER 3 LITERATURE REVIEW

3.1 Cooperative Query Answers (CQA)

Traditional query answering systems require users to understand the overall database schema, what they want to ask, and only return *exact* answers to user's query. This is usually very difficult for inexperienced users.

3.1.1 Fundamentals of the CQA

In conventional relational database, if required data is missing, an exact answer is unavailable, or a query is not well-formed with respect to the schema, the database just returns a *null* answer or an *error*. CQA [13,14,15,16] techniques have been developed to overcome these drawbacks where different approaches are used to broaden the scope of the user query.

The goal of cooperative database research is to create information systems which can provide approximate answers to a user's query when exact answers are unavailable and summaries answer when answer sets are too large [13,14].

There are three main components which are used by the CQA based systems when intelligently constructing queries for the users. They are *domain knowledge*, *new SQL operators*, and *query relaxation techniques*.

3.1.2 Domain Knowledge

Most domain knowledge that has been used by CQA-based system is contained in hierarchical structure [13,14,16]. However, different names have been used to refer to the hierarchies (e.g., Concept Hierarchy [17] and Type Abstraction Hierarchy (TAH) [13]).

CQA based systems generally use a number of concept hierarchies to represent different types of domain knowledge. Higher levels of the hierarchy provide more abstract data representation than lower levels. Generalization (moving up in the hierarchy), specialization (moving down the hierarchy), and association (moving between hierarchies) are the three key operations in deriving cooperative query answers for the users. Figure 3-1 shows an example of a concept hierarchy used in the CoBase project at UCLA [16].

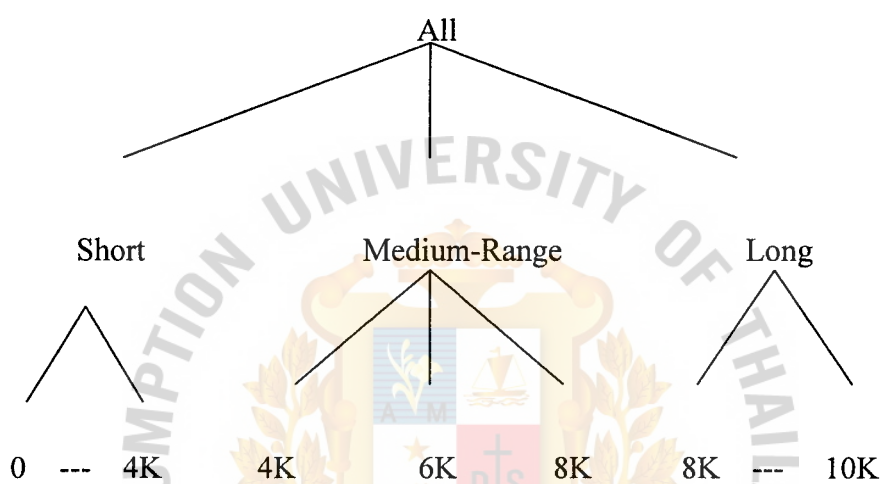


Figure 3-1: Type Abstract Hierarchy (runway_length)

In order to answer the querying, if no exactly answer can be found, the system will proceed to relax the range of specified field in query by using the abstract hierarchy.

Assume that the following is the query before relaxation:

```
SELECT aport_name, runway_length_ft
FROM Airports
WHERE runway_length_ft > 7500
```

In relaxation process, the system will move up in the hierarchy. Then it found that `runway_length_ft = 7500` is under *Medium-Range* group. And then it will move down again to seek the range of `runway_length` which is located in *Medium-Range* group.

As the result, the query will be relaxed to the following:

```
SELECT aport_name, runway_length_ft
FROM Airports
WHERE runway_length_ft >= 4000
```

3.1.3 Cooperative Operators (CoSQL)

In order for the DBMSs to support CQA requirement, W. Chu [13] has proposed a number of CoSQL such as APPROXIMATE, BETWEEN, WITHIN, NEAR-TO, SIMILAR-TO, BASED-ON, RELAXATION-ORDER and ANSWER-SET, as used in CoBase project at UCLA.

For example:

```
SELECT aport_name, runway_length_ft, runway_width_ft
FROM Airports
WHERE runway_length_ft BETWEEN(6000, 7000)
```

```
SELECT aport_name, latitude, logitude
FROM Airports, GEOLOG
```

WHERE `aport_name` NEAR-TO 'Bizerte'
AND `country_state_name` = 'Tunisia'

3.1.4 Query Relaxation

In order to perform partial match retrieval, most of CQA-based systems use the concept hierarchy as their knowledge structure to guide query relaxation. The high level nodes in the hierarchy represent general knowledge of the domain, while the lower level nodes represent more specific knowledge. The following example describes process of relaxation of CoBase[13]. Consider the domain knowledge as shown in Figure 3-1 and the default relaxation range as shown in Table 3-1.

Assume that a user wants to list all the airports with the runway length greater than 7500 feet and runway width greater than 100 feet. If no answer can be found, then in order to find partial match records, the system has to relax the runway length condition first.

Table 3-1: Relaxation Range for the Approximate Operator.

| Relation Name | Attribute Name | Approximate Range |
|---------------|-----------------|-------------------|
| Airports | Runway_width_ft | 10 |
| Airports | Parking_sq_ft | 100000 |

The following is the corresponding CoSQL query:

```
SELECT aport_name, runway_length_ft, runway_width_ft  
FROM Airports  
WHERE runway_length_ft > 7500 AND runway_width_ft >100  
WITH RELAXATION-ORDER (runway_length_ft, runway_width_ft)
```

Based on the runway_length hierarchy and the relaxation order, the query is relaxed to:

```
SELECT aport_name, runway_length_ft, runway_width_ft
FROM Airports
WHERE runway_length_ft >= 4000 AND runway_width_ft > 100
```

There is still no answer can be found, the system will proceed to relax the range of runway width by a pre-specified value as shown in Table 3-1. As the result the query is relaxed to:

```
SELECT aport_name, runway_length_ft, runway_width_ft
FROM Airports
WHERE runway_length_ft >= 4000 AND runway_width_ft > 80
```

3.2 Data Extraction: Interfacing with a Database

Data Extraction is one phase in KDD process. The current process of extraction requires users to manually extract data from a database, save it to a flat file, and then have a mining algorithm process that file. John [3] argued that a mining algorithm should directly queried the RDBMS rather than directly manipulating data records in a file. He proposed the SQL Interface Protocol (SIP), which is one such framework for interaction between a mining algorithm and a database. The data continues to reside entirely within the DBMS, but the query interface to the database gives the data mining algorithm all the information it needs to discover patterns.

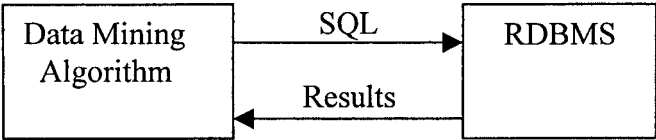


Figure 3-2: Proposed method for mining from SQL queries [3].

The following is the example of SIP:

```
SELECT  $\phi$ ,  $f(*)$   
FROM  $D$   
WHERE  $\psi$   
GROUP BY  $\phi$ 
```

Where f is an aggregation operator (e.g., COUNT, SUM, AVG, STDDEV, MIN, MAX), ϕ is a list of attribute names, ψ is a list of attribute names with associated constraints, and D is the name of the relation.

3.3 A Data Mining Query Language (DMQL)

DMQL language has been implemented in DBMiner system for interactive mining of multiple-level knowledge in relational database[18]. DMQL adopts an SQL-like syntax to facilitate high level data mining and natural integration with relational query language, SQL. The syntax of DMQL for mining different kind of rules are:

<DMQL> ::=

```
USE DATABASE <database_name>

{USE HIERARCHY <hierarchy_name> FOR <attribute>}

<rule_spec>

RELATED TO <attr_or_agg_list>

FROM <relation(s)>

[WHERE <condition>]

[ORDER BY <order_list>]

{WITH [<kinds_of>] THRESHOLD = <threshold_value>

    [FOR <attribute(s)>]}
```

Assume that the user want to find the general characteristics of the graduate students in computer science in relevance to attributes gpa, birth_place and address, for the students who were born in Canada.

The data mining query example is presented in DMQL as follow.

```
USE DATABASE University

FIND Charateristic rule

RELATED TO gpa, birth_place, address, COUNT(*)%

FROM Student

WHERE status="graduate" AND major="cs"

    AND birth_place="Canada"

WITH noise THRESHOLD=0.05
```

CHAPTER 4 THE PROPOSED OPERATORS

4.1 Required Environment

We have the following philosophical considerations which will serve as guidelines in our design of a data error detection operators.

1. The set of data that need to be error detected should be specified. Since a user may be interested in any portion of data in a database.
2. Domain knowledge base should be available for data error detection. Since the processes of detection need some knowledge in order to compare and examine the level of data accuracy.
3. The interactive user interface should be included in data error detection. This resultant from the fact that this process must allow the user to input a new set of values instead of the incorrect data values.

Based on the above considerations, we propose a set of data error detection operators which have been designed to detect errors in relational databases. The required environment consist of three major primitives: (1) the set of data, (2) the domain knowledge, and (3) the interactive user interface.

4.2 Different Types of Domain Knowledge

Humans use various types of knowledge when solving their problems [7]. This knowledge can be represented in a computer by using many different knowledge representation schemes. This thesis discusses two types of knowledge schemes:

4.2.1 Hierarchy

Recall the fact that different names have been used to refer to the hierarchies (e.g., Concept Hierarchy [17] and Type Abstraction Hierarchy (TAH) [13]). In most cases, the domain knowledge used by CQA based systems are contained in hierarchical structure.

The hierarchy represents objects at different levels of abstraction. For example, in Figure 3-1, the Medium-Range (i.e, from 4,000 to 8,000ft.) in the hierarchy for runway_length is a more abstract representation than a specific runway length in the same hierarchy (e.g., 6,000ft.). A higher-level and more abstract object representation corresponds to multiple lower-levels and more specialized object representations.

A query can be modified by relaxing the query conditions via such operations as generalization (moving up the hierarchy) and specialization (moving down the hierarchy), e.g., from 6000ft to Medium-Range to [4000ft, 8000ft].

4.2.2 Rules

As the result of knowledge discovery process, logical relationships are generated. The logical relationships are usually represented as rules. The simplest types of rules express conditional or association relationship.

A conditional rule is a statement of the form:

IF condition1

THEN condition 2

Another type of rule, *affinity* analysis (or *association* analysis), is the patterns or conditions that describe how various items “group together” or “happen together” within a series of events or transactions. An affinity rule has the form:

WHEN item1

ALSO item2

In additional, the logical conditions and associations have been combined for better results (“Data Mining with OLAP Affinities”, Parsaye, K.). A dimensional affinity has the form:

CONFIDENCE=%

IF condition1

WHEN item1

ALSO item2

4.3 Syntax of the Data Error Detection Operator

The data error detection operator will be defined in the following grammar, where words in capital characters represent keywords, and “{ }” represent 0 or more occurrence of statement, as shown below.

```

DETECT <attribute_list>

FROM <relation_name>

{WHERE <condition_list>}

USE DATABASE <database_name>

USE KNOWLEDGE <knowledge_name> FOR <attribute_list>

{MAXIMUM = <max_result_record>}

```

1. "DETECT" is the core keyword that used to specify a set of attributes <attribute_list> that need to be error detected. A comma "," is used to separate each attribute name.
2. "FROM" is used to collect the relevant data from the specific relation name <relation_name>.
3. "WHERE" is optional statement to specify the collection of data that need to be error detected. Only the data that satisfy the conditions in <condition_list> will be error detected.
4. "USE DATABASE" is used to specify the target database <database_name>.
5. "USE KNOWLEDGE" is statement to assign the knowledge base <knowledge_name> to a particular set of attributes <attribute_list>. A comma "," is used to separate each attribute name.

6. "MAXIMUM" is optional statement to specify the maximum record number of error detection result <max_result_record>. Otherwise, the system will detect all records and also show all invalid records in database. Recall that data size use in most data mining is very large. The reason for having this type of operator is to provide more flexibility to user in order to concentrate on just a small set of data.

4.4 Control Flow Chart of Data Error Detection Operator

The control flow chart of data error detection operator is shown in Figure 4-1.

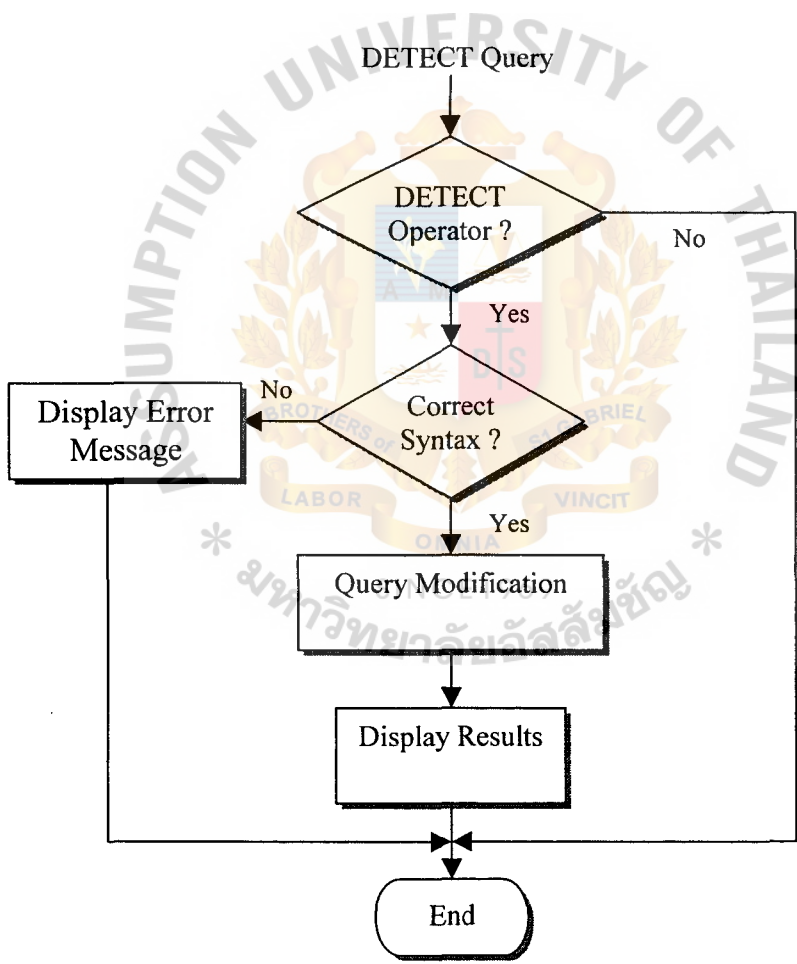


Figure 4-1 Data Error Detection Control Flow Chart

When users like to detect errors, they will have to enter 'DETECT' command follows by other keywords that previously shown. Thus the query is presented to system, the system first examines whether the query is 'DETECT' query or not. After pass syntax checking the query will be modified under the direction of knowledge discovery from knowledge base. Finally, the modified query is executed and the results are displayed to user.

4.5 Examples of Data Error Detection

The examples shown below base on a university database with the following schema.

Student(StdID,FirstName,LastName,BirthDate,FacultyID,GPA,addrNum,Soi,
Road,SubDistrict,District,Province,ZipCode,IssuedOn,ExpiredOn)
Faculty(FacultyID,FacultyName)

Example1: Assume that users want to detect errors from attribute StdID and GPA for all students.

The following is the corresponding SQL query:

```
DETECT StdID, GPA  
  
FROM Student  
  
USE DATABASE University  
  
USE KNOWLEDGE StdRule FOR StdID, GPA
```

The data error detection will be first syntax checked. Then the algorithm for finding rules is executed by considering knowledge StdRule, Table 4-1, which

specified in the query. All attributes that occur in attributes list will be examined with the related rules. Finally, the modified query is executed and the results are displayed to user.

The following is the modified SQL query:

```
SELECT stdID AS Pri_index, stdID INTO table_name1 FROM student
WHERE NOT( SUBSTRING(StdID,1,2)=SUBSTRING(IssuedOn,1,2) AND SUBSTRING
(StdID,3,2)=FacultyID )
```

```
SELECT stdID AS Pri_index, GPA INTO table_name2 FROM student
WHERE NOT( GPA >= 0 AND GPA <=4 )
```

```
SELECT student.stdID AS StudentID, table_name1.stdID , table_name2.GPA
INTO TStudent FROM student, table_name1 , table_name2
WHERE student.stdID *= table_name1.Pri_index
AND student.stdID *= table_name2.Pri_index
AND student.stdid NOT IN
( SELECT student.stdID FROM student WHERE
    student.stdid NOT IN ( SELECT pri_index from table_name1 ) AND
    student.stdid NOT IN ( SELECT pri_index from table_name2 ) )
```

Table 4-1: Domain Knowledge – StdRule.

| DB Name | Relation Name | Attribute Name | Rules |
|------------|------------------|-------------------|---|
| University | Student | StdID | SubString(StdID,1,2) = SubString (IssuedOn,1,2), SubString(StdID,3,2) = FacultyID |
| University | Student | GPA | GPA >= 0 AND GPA <= 4 |
| University | Student | SubDistrict | FOUNDIN RegionDomain |
| University | Student | District | FOUNDIN RegionDomain |
| University | Student | Province | FOUNDIN RegionDomain |

St. Gabriel's Library

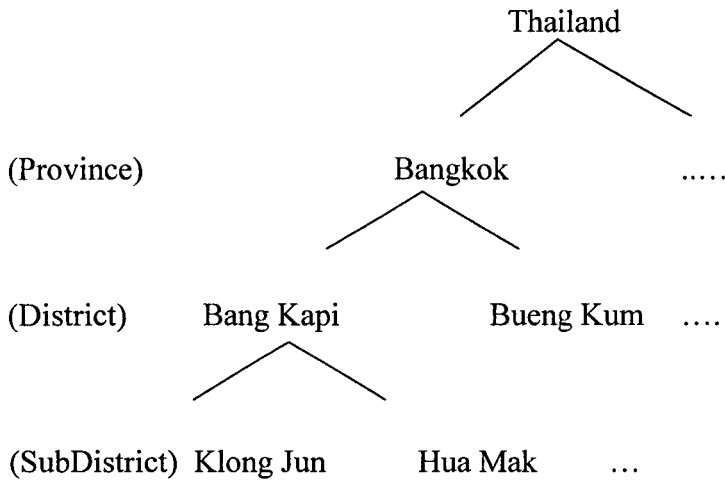


Figure 4-2: Domain Knowledge – RegionDomain.

Example2: Assume that users want to detect errors in student's address (SubDistrict, District, Province) for all students which locate in Bangkok and the result should be shown only the first 1000 invalid records.

The following is the corresponding SQL query:

```
DETECT SubDistrict, District, Province
FROM Student
WHERE Province = 'Bangkok'
USE DATABASE University
USE KNOWLEDGE StdRule FOR SubDistrict,District,Province
MAXIMUM = 1000
```

The data error detection will be first syntax checked and then executed the algorithm for finding rules. In this example, the system will use two knowledge bases,

StdRule and RegionDomain shown in Figure 4-2. The rule “FOUNDIN RegionDomain” mean value of all specified attributes must be related together which in turn will be examined by using RegionDomain knowledge. Such as, if the SubDistrict attribute is “Klong Jun”, the District attribute must be “Bang Kapi” and also the Province attribute must be “Bangkok”.

The following is the modified SQL query:

```
SELECT stdID AS Pri_index, SubDistrict, District, Province INTO table_name1
FROM Student
WHERE Province='Bangkok'
AND stdID NOT IN
( SELECT Student.stdID
FROM Student, RegionDomain
WHERE Student.SubDistrict = RegionDomain.SubDistrict
AND Student.District = RegionDomain.District
AND Student.Province = RegionDomain.Province )
```

```
SELECT stdID AS Pri_index, SubDistrict, District, Province INTO table_name2
FROM Student
WHERE Province='Bangkok'
AND stdID NOT IN
( SELECT Student.stdID
FROM Student, RegionDomain
WHERE Student.SubDistrict = RegionDomain.SubDistrict
AND Student.District = RegionDomain.District
AND Student.Province = RegionDomain.Province )
```

```
SELECT stdID AS Pri_index, SubDistrict, District, Province INTO table_name3
FROM Student
WHERE Province='Bangkok'
AND stdID NOT IN
( SELECT Student.stdID
FROM Student, RegionDomain
WHERE Student.SubDistrict = RegionDomain.SubDistrict
AND Student.District = RegionDomain.District
AND Student.Province = RegionDomain.Province )
```

```
SELECT Student.stdID AS StudentID, table_name1.SubDistrict ,
table_name2.District , table_name3.Province INTO TStudent
FROM Student, table_name1 , table_name2 , table_name3
WHERE Student.stdID *= table_name1.Pri_index
```

```

AND Student.stdID *= table_name2.Pri_index
AND Student.stdID *= table_name3.Pri_index
AND Student.stdid NOT IN
( SELECT Student.stdID FROM Student WHERE
  Student.stdid NOT IN ( SELECT pri_index FROM table_name1 ) AND
  Student.stdid NOT IN ( SELECT pri_index FROM table_name2 ) AND
  Student.stdid NOT IN ( SELECT pri_index FROM table_name3 ) )

```

4.6 The Interactive User Interface

Recall that data mining process itself normally consists of a number of iterate process, yet complex. There are many researchers argued that the KDD system should be seen as interactive tools, not as automatic analysis system [10, 11]. This way the user can always certain that the result of analysis is complete because the users can closely monitor the KDD process.

As a result of thesis, an interactive user interface is developed. As shows in Figure 4-3.

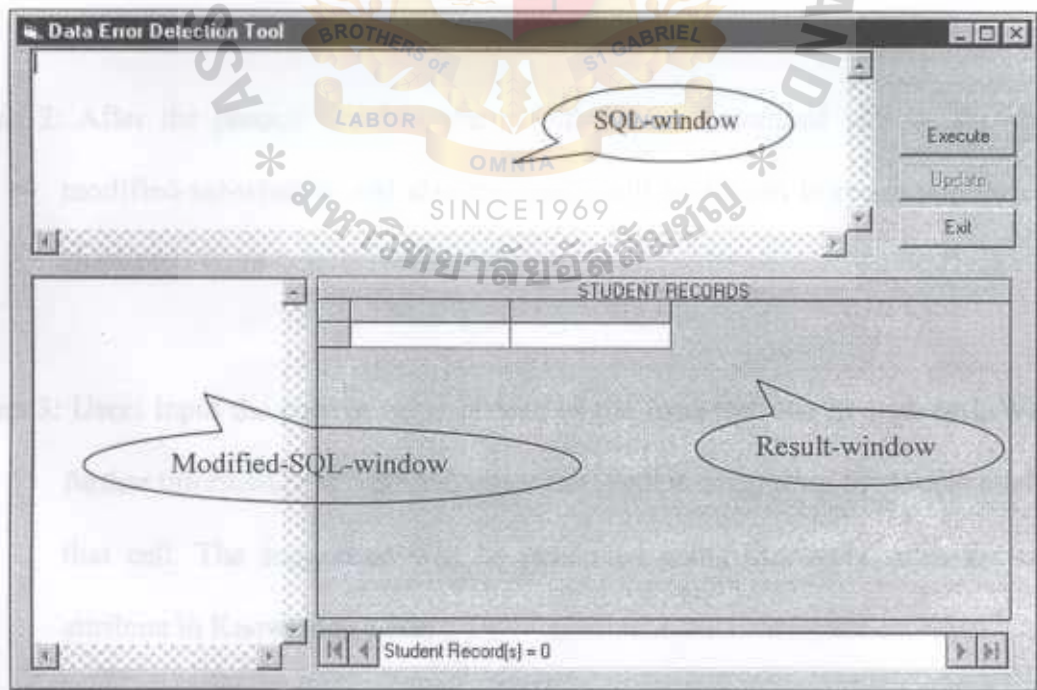


Figure 4-3: The interactive user interface.

St. Gabriel's Library

The interface is separated into 3 parts.

The first is a Query-window, this window is an input window that will be prompt to receive SQL command from the user.

The second is Modified-SQL-window, this window shows final SQL commands that will be sent to DBMS.

The third is Result-window, when users entry the command and execute it, the results will be shown in this window. Users can examine the error in each record and input new values instead, whenever the incorrect value is found.

The following paragraphs show steps of data error detection.

Step 1: User input DETECT command in query-window and then press the Execute button. As shows in Figure 4-4.

Step 2: After the process finishes, the modified SQL command will be shown in modified-sql-window and also the result will be shown in result-window. As shows in Figure 4-5.

Step 3: Users input the correct value instead of the incorrect one in each cell. When further information is required, users can request suggestion by double click in that cell. The suggestion will be generated using discovery rules for each attribute in Knowledge Base.

Step 4: After they finish examines and changes all invalid values, press the Update button. As the result, the correct value will be updated to database, and the complete message will be shown after completing the update.

Figure 4-6 – 4-11 show the suggestion that appear when user double click in cell that contains invalid value. The appeared suggestion depend on the attribute contains in the cell.



Figure 4-4: User input DETECT Command.

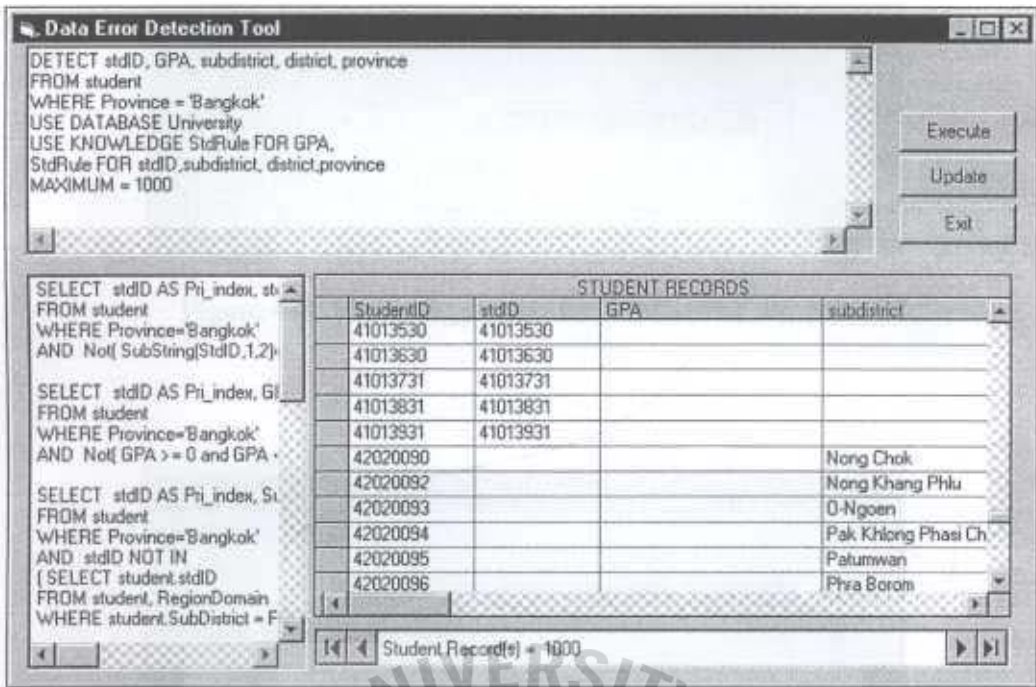


Figure 4-5: The modified SQL command and result will be shown.

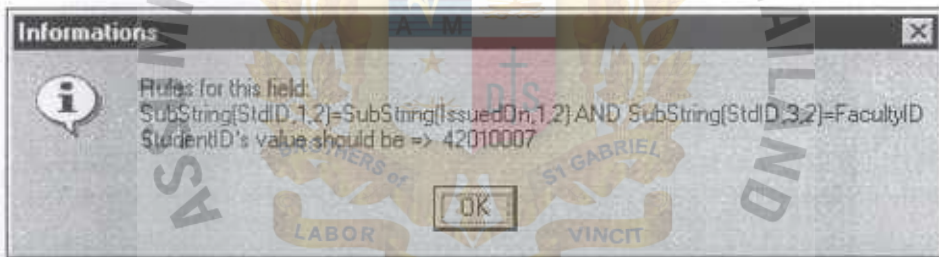


Figure 4-6: Show suggestion for Student ID attribute.

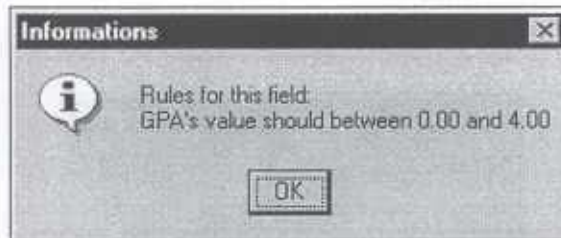


Figure 4-7: Show suggestion for GPA attribute.



Figure 4-8: Show suggestion for SubDistrict attribute.



Figure 4-9: Show suggestion for District attribute.

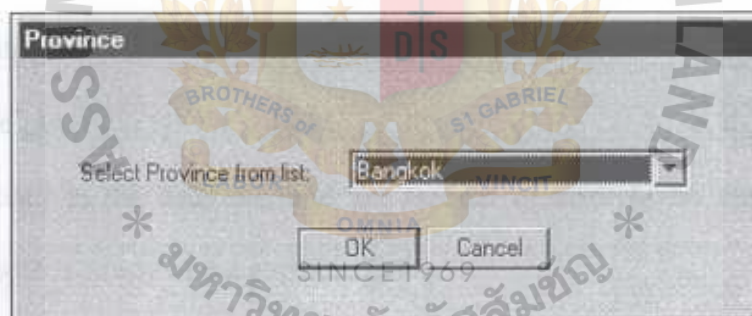


Figure 4-10: Show suggestion for Province attribute.



Figure 4-11: Show complete updated message.

CHAPTER 5 EXPERIMENTAL AND EVALUATION

This chapter presents the performance evaluation based on measuring the execution of a set of queries on the tested development domain. Since, the evaluation on this type of contribution is difficult. A method of similar researches has been explored especially how their systems are evaluated.

In the literature, CoBase system at UCLA [13], the performance measure includes response time for query relaxation, association and explanation, and the quality of answers. Their evaluation performs on the CoBase tested developed with a number of queries. The response time depends on the type of queries (e.g., size of joins, number of joins) as well as the amount of relaxation, association and explanation required to produce an answer. The quality of the answer depends on the amount of relaxation and association involved. Additionally, Merge/Purge system [22] used tested databases that were generated automatically by a database generator. This database generator allows a user to set the size of the database, the percentage of duplicate record in the database, and the amount of error to be introduced in the duplicated records in any of the attribute fields.

In this thesis we will focus on measuring the quality of answers and response time.

Based on the evaluation methods discussed above, in order to measure the quality of answers; the followings are the environment of testing.

- A tested development database with the amount of error to be introduced in the records in any of the attributed fields.
- Test cases are a number of error detection queries with various conditions.

- The results for each test case that shows expected data in database after executing error detection query.

Finally, we can examine percentage of accuracy from the expected results and the real results. Accuracy is measured as the percentage of the number of invalid records correctly found by the process. Errors are measured as the percentage of records claimed to be error, which in fact are not actual errors.

5.1 Environment of Examination

In this thesis, we examine the accuracy and performance of proposed data error detection tool under the following environments.

Hardware Specification:

- CELERON 500 MHz
- SDRAM 128 MB 100 MH
- 4.3 GB HDD

Software Specification:

- Data error detection Tool is developed from Microsoft Visual Basic (VB) version 6.0
- Microsoft SQL Server version 6.5
- Windows NT Server version 4.0 (SP4)

5.2 A Development Database

Database used to examine data error detection tool was generated in controlled environment that is we control size of database and percentage of invalid record in

that database. Each record generated consists of the following fields, some of which can be empty: student id, first name, last name, birth date, faculty id, gpa, address number, soi, road, subdistrict, district, province, zip code, issue on and expired on.

5.3 Experiment Methods

There are many parameters that could effect the accuracy and performance of data error detection such as number of attributes to be error detected, size of database, etc. To examine all related parameters, the experiments are performed as following.

We examine data error detection tool under various conditions and size of database that show in Table 5-1.

Table 5-1: Summary of Experiment Cases

| Experiment | No of Records | No of Attribute Considered | Amount of Invalid Records (%) |
|------------|---------------|-------------------------------|----------------------------------|
| 1 | 500 | 1 | 10 |
| 2 | 500 | 2 | 20 |
| 3 | 500 | 5 | 30 |
| 4 | 1,000 | 1 | 10 |
| 5 | 1,000 | 2 | 20 |
| 6 | 1,000 | 5 | 30 |
| 7 | 2,000 | 1 | 10 |
| 8 | 2,000 | 2 | 20 |
| 9 | 2,000 | 5 | 30 |
| 10 | 4,000 | 1 | 10 |
| 11 | 4,000 | 2 | 20 |
| 12 | 4,000 | 5 | 30 |

The purpose of these experiments was to determine accuracy of the data error detection tool and time used for error detection under different size of database.

Additionally the experiments have been conducted three times over each database size for different number of attributes. The first time we detect errors from only one attribute, the second we detect errors from two attributes and the last time we detect errors from five attributes. The purpose of these experiments was to determine the effect of varying number of attribute. The results of data error detection show in Table 5-2. where:

- Actual Invalid :number of invalid records that is controlled when creates database.
- Found Invalid :number of invalid records that is result of search process.
- Process Time :time used to detect errors that is separated to “Search” and “Update” time.
- Search Time :time used to search invalid record using knowledge that recovery from knowledge base.
- Update Time :time used to update correct data that user input into database.

Table 5-2: Summary of Experiments and Results

| EXP. | DB Size | Actual Invalid (Records) | Found Invalid (Records) | Process Time (Seconds) | | |
|------|---------|-----------------------------|----------------------------|---------------------------|--------|-------|
| | | | | Search | Update | Total |
| 1 | 500 | 50 | 50 | 0.75 | 0.75 | 1.5 |
| 2 | 500 | 100 | 100 | 1.0 | 1.25 | 2.25 |
| 3 | 500 | 150 | 150 | 5.25 | 2.0 | 7.25 |
| 4 | 1,000 | 100 | 100 | 0.75 | 1.25 | 2.0 |
| 5 | 1,000 | 200 | 200 | 1.25 | 2.25 | 3.5 |
| 6 | 1,000 | 300 | 300 | 5.25 | 3.75 | 9.0 |
| 7 | 2,000 | 200 | 200 | 1.0 | 2.0 | 3.0 |
| 8 | 2,000 | 400 | 400 | 1.75 | 4.5 | 6.25 |
| 9 | 2,000 | 600 | 600 | 9.25 | 8.0 | 17.25 |
| 10 | 4,000 | 400 | 400 | 1.5 | 4.25 | 5.75 |
| 11 | 4,000 | 800 | 800 | 3.0 | 9.25 | 12.25 |
| 12 | 4,000 | 1,200 | 1,200 | 14.25 | 15.75 | 30.0 |

5.4 Result Analysis

In regard to the result of data error detection, we found that the proposed data error detection tool can work correctly with high accuracy level. It found all invalid data on the database as shows in Figure 5-1. In other words, the number of found invalid records is equal to the number of actual invalid records for all experiments.

Additionally, we also found that there are two factors that effect the performance of data error detection. These factors are size of database and number of attribute to be error detected.

Figure 5-1 shows data error detection time (summation of search time and update time) in each database size and varying number of error detected attributes.

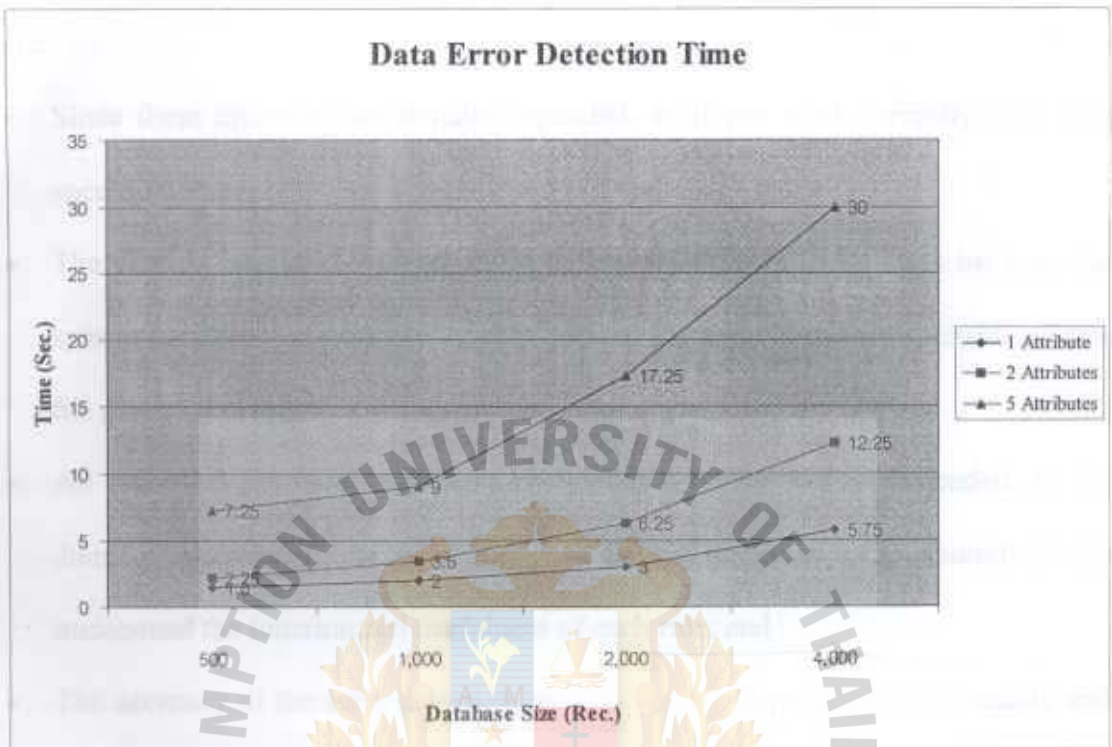


Figure 5-1: Data error detection time in various conditions.

As the result we can conclude that performance of data error detection is depend on two factors:

- Databases size, if database size increases time to detect errors increases.
- Number of attribute to be error detected, if number of attribute in SQL increases time to process increases.

CHAPTER 6 DRAWBACKS AND FUTURE RESEARCH

There are some possible drawbacks of the proposed data error detection operator, they are:

- Since these operators are domain-dependent, so it can work correctly only in a specific domain.
- The domain expert should correctly define the domain knowledge because the system assumes that the knowledge is correct. The system cannot by itself validate the given knowledge.
- All rules that are used in domain knowledge are also domain-dependent, so the domain experts who have authority to define the knowledge should clearly understand the function and usefulness of each rule; and
- The accuracy of the error detection result is mainly depended on the quality and sufficient amount of the domain knowledge.

However, the proposed data error detection operator can be enhanced to increase its performance by providing tool for domain experts to add the knowledge base. Additional, it will be increased usefulness if DETECT operator is standard command for widely database.

CHAPTER 7 CONCLUSIONS

In this thesis, we briefly describe the KDD process and the functionality for each of its step. We have focused on the cleaning process that is one important step due to the quality of the derived knowledge is highly depended on the quality of the data.

This thesis proposed a method for handling data error detection based on ideas draw from the benefit of using SQL-like operators. The error detection method uses domain knowledge that is prior defined by the domain expert. Additionally, we also proposed the interactive user interface, in which the user can always certain that the result of error detection is complete because the users can closely monitor the process.

The main advantage of this approach is to prevent users from having to remember and use long and complicated SQL statement in order to do the same task.

However, experimental results have shown that there are two factors that effect the performance of data error detection. These factors are **size** of database and **number of attribute** to be error detected. If database size is increased, then time to detect errors is also increased. In addition, if number of attribute in SQL is incremented, then time to process is also increased. However, these issues of time complexity also occur with conventional SQL.

BIBLIOGRAPHY

- [1] Ronald J. Brachman and Tej Anand. The Process of Knowledge Discovery in Databases. In *Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, Advance in Knowledge Discovery and Data Mining*, pages 37-57. AAAI Press/MIT Press, 1996.
- [2] T. Imielinski and H Mannila. A Database Perspective on Knowledge Discovery, *communications of the ACM*, 39(11), November 1996.
- [3] George H. John. Enhancements to the Data Mining Process. PhD thesis, Department of Computer Science, Stanford University, USA, 1997.
- [4] U.M. Fayyad, G. Piatetsky-Shapiro, P.Smyth, R. Uthurusamy (eds.). *Advance in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.
- [5] G. Piatetsky-Shapiro and W. J. Frawley (eds.), *Knowledge Discovery in Databases*. AAAI/MIT Press, 1991.
- [6] Heikki Mannila. *Inductive Databases and Condensed Representatins for Data Mining*, MIT Press, 1997.
- [7] Aamodt, A. and Plaza, E. Case-Based Reasoning: Foundatinal Issues, Methodological Variatins, and System Approaches, *AICom-Artificial Intelligence Communications*, Vol. 7, No. 1, 1994.
- [8] Han, J., Huang, Y., Cercone, N., and Fu, Y. Intelligent Query Answering by Knowledge Discovery Techniques, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 3, pp. 373-390, 1996.
- [9] Chu, W. W., Merzbacher, M. A., and Berkovich, L. The Design and Implementation of CoBase, In *Proceedings of ACM SIGMOD'93*, Washington D. C., 1993.

- [10] M. Klemettinen, H. Mannila, H. Toivonen. A Data Mining Methodology and Its Application to Semi-Automatic Knowledge Acquisition.
- [11] H. mannila. Methods and problems in data mining. In *the proceeding of International Conference on Database Theory*, Delphi, Greece, January 1997, F. Afrati and P. Kolaitis (ed.), Springer-Verlag.
- [12] Mauricio a. Herna'ndez, Salvatore J. Stolfo. Real World Data is Dirty: Data Cleansing and The Merge/Purge Problem. *Data Mining and Knowledge Discovery*, 2, 9-37 (1998). Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.
- [13] Wesley W. Chu, Hua Yang, Kuanang Chiang, Michael Minock, Gladys Chow, Chris Larson. CoBase: A Scalable and Extensible Cooperative Information System. *Journal of Intelligent Information System*, 6, 1996.
- [14] Michael J. Minock, Wesley W.Chu. Explanation for Cooperative Information System. In *Proceeding of 9th Intelligent Symposium on Methodologies for Intelligent Systems*, June, 1996.
- [15] Gilles Fouque', Wesley W.Chu, Henrich Yau. A Case-Baed Reasoning Approach for Associative Query Answering.
- [16] Wesley W.Chu, Qiming Chen, Matthew Merzbacher. CoBase: A Cooperative Database System.
- [17] J. Han, Y. Huang, N. Cercone, and Y. Fu. Intellignet Query Answering by Knowledge Discovery Techniques. *IEEE Transactions on Knowledge and Data Engineering*, 8(3):373-390, 1996.
- [18] Jiawei Han, YongjianFu, Wei Wang, Krzysztof Koperski, Osmar Zaiane. DMQL: A Data Mining Query Language for Relational Database.

- [19] Fellegi, I. And Sunter, A. A Theory for Record Linkage. *American Statistical Association Journal*, pages 1183-1210, December 1969.
- [20] ACM. SIGMOD record, December 1991.
- [21] Wang, Y. R. and Madnick, S. E. The Inter-Database Instance Identification Problem in Integration Autonomous Systems. In *Proceedings of the Sixth International Conference on Data Engineering*, February 1989.
- [22] Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(3), Fall 1996.
- [23] Thomas C. Redman. Data Quality for the Information Age. ARTECH HOUSE Inc., 1996.



