

University of Windsor

Scholarship at UWindor

Electronic Theses and Dissertations

Theses, Dissertations, and Major Papers

2001

Code generator for integrating warehouse data sources.

Yi. Liu

University of Windsor

Follow this and additional works at: <https://scholar.uwindsor.ca/etd>

Recommended Citation

Liu, Yi., "Code generator for integrating warehouse data sources." (2001). *Electronic Theses and Dissertations*. 1457.

<https://scholar.uwindsor.ca/etd/1457>

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI[®]

NOTE TO USERS

Page(s) not included in the original manuscript and are unavailable from the author or university. The manuscript was microfilmed as received.

78 - 81

This reproduction is the best copy available.

UMI[®]

**CODE GENERATOR FOR INTEGRATING
WAREHOUSE DATA SOURCES**

BY

LIU YI

A Thesis

**Submitted to the Faculty of Graduate Studies and Research
through the School of Computer Science in
Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor**

Windsor, Ontario, Canada

2001



**National Library
of Canada**

**Acquisitions and
Bibliographic Services**

**395 Wellington Street
Ottawa ON K1A 0N4
Canada**

**Bibliothèque nationale
du Canada**

**Acquisitions et
services bibliographiques**

**395, rue Wellington
Ottawa ON K1A 0N4
Canada**

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-62244-4

Canada

938501

Liu Yi 2000

© All Right Reserved

Abstract

Data warehouse is a large integrated database organized around major subjects of an enterprise for the purpose of decision support querying. Many enterprises are creating their own data warehouse systems from scratch in different varying formats, making the issue of building a more efficient, more reliable, cost-effective and easy-to-use data warehouse system important. Building a code generator for creating a program that automatically integrates different data sources to target data warehouse is one solution.

Thus, understanding approaches for integrating warehouse data sources is a key to the success of data warehouse code generation. Integrating warehouse data sources involves building a code generator program for creating both data warehouse and metadata using novel techniques for extracting and cleaning data. Many types of data sources like e-commerce databases, web databases and knowledge bases can also be accommodated in the code generator in the future.

There is little or no literature showing the use of the newest integration techniques in code generator for data warehouse data integration. This thesis aims at employing new techniques for both data integration and code generation, in building a code generation tool for data warehouse data integration.

To my parents.
To my wife and son
To my teachers,
To my friends

...

Acknowledgments

I take this opportunity to thank my advisor Dr. Ezeife. This work could not have been accomplished without her solid help in theory and practice. Thanks for her precious time on consultation during this work, her continuous encouragement and intensive support throughout my graduate studies. Thanks to Dr. Schmitt and Dr. Zamani for comments, questions, and criticisms on this thesis.

I would like to thank my family for their support. I specially thank my wife, Du Ying, for love, courage, determination and support. Thank Du for devoting so much time to housework during this period, thank my boy, Liu Hua, who brought me joy that enabled me to study hard because he could take good care of himself, thank my father and mother who gave me lots of courage and financial support.

Table of Contents

Abstract	IV
Acknowledgment	VI
1 Introduction	1
1.1 Data Warehouse	2
1.2 Data Warehouse System	2
1.2.1 Data in Data Warehouse System	4
1.2.2 Programs in Data Warehouse System	5
1.3 Data/Schema Integration	7
1.3.1 Schema Integration	8
1.3.2 Data Integration	8
1.4 Code Generator	9
1.4.1 Code Generator and Compiler	10
1.4.2 Code Generation for Data Warehouse Integration	12
1.4.3 Code Generation Approaches	13
1.5 Thesis Problem and Contributions	14
1.6 Outline of the Thesis	15
2: Previous/Related Work	16
2.1 Schema Integration	16
2.2 Data Integration Systems	21
2.2.1 TSIMMIS	23
2.2.2 SIMS	25
2.2.3 Infomaster	27
2.2.4 Dynamic View	31
2.3 Some Code Generator Systems	32
2.3.1 SAS	33
2.3.2 OBLOG	34
2.3.3 GENTLE	36
3: WODD Code Generation Techniques	39

3.1 Schema/Data Integration Techniques	39
3.1.1 Definitions	40
3.1.2 Rule Transformation Algorithm	42
3.1.2 Example	44
3.2 Code Generator Technique	48
3.2.1 Example	49
4: Implementation	56
4.1 Project Aims and Restrictions	56
4.2 System Architecture	57
4.3 User Interface and Components	60
4.4 WODD programs	62
4.5 Example	65
4.6 Target Programs for Warehouse Data Integration	71
4.6.1 Data Extraction	72
4.6.2 Data cleaning	73
4.6.3 Data loading	75
5: Conclusions and Future work	76
5.1 Conclusions	76
5.2 Future Work	77
Bibliography	78
Appendix A: User Manual	82
Appendix B: Project Codes	103
Vita Auctoris	240

List of Figures

Figure 1.1: Data Warehouse System Architectures	3
Figure 1.2: Compilation Process	11
Figure 1.3: Data Warehouse Code Generator	13
Figure 2.1: Original Schema	17
Figure 2.2: Choose “Topic” for “Keyword”	17
Figure 2.3: Make Publisher into an Entity	18
Figure 2.4: Superimposition of Schema	18
Figure 2.5: Creation of a Subset Relationship	19
Figure 2.6: An Example of Integration	19
Figure 2.7: The architecture of TSIMMIS	23
Figure 2.8: Example of TSIMMIS data model	23
Figure 2.9: Example of TSIMMIS query language	24
Figure 2.10: Example of SIMS data model	25
Figure 2.11: Example of query language	26
Figure 2.12: Architecture of Infomaster	27
Figure 2.13: Mercedes car table and transferring constraints	28
Figure 2.14: Common (target) table	29
Figure 2.15: Vertical and horizontal integration	30
Figure 2.16: Three source databases	31
Figure 2.17: traditional and dynamic query	32
Figure 2.18: Example of target program and its hierarchy	34
Figure 2.19: Rules example for documentation generation	35
Figure 2.20: GENTLE expression	36
Figure 2.21: An example of expression rule in a stack	37
Figure 2.22: evaluation of stack based code generator	37
Figure 3.1: Interface base, site relation diagrams	41
Figure 3.2: Relation definition	42
Figure 3.3: Template combines with rule based code generation approach	49
Figure 3.4: empty template	50
Figure 3.5: component in template before assigning Variable in \$ Sign	51
Figure 3.6: Template with two components	53
Figure 3.7: output program for extracting Windsor star information	55
Figure 4.1: System Architecture	58
Figure 4.2: System Components	61
Figure 4.3 Java Class Definitions	62
Figure 4.4 Basic composition components	63
Figure 4.5 Example of mainComponent for Pro*C	63
Figure 4.6: Banking Testing Files	64
Figure 4.7: Graphic files	64
Figure 4.8: Source Database Architecture	66
Figure 4.9: Data Warehouse Architecture	67

Figure A1.1: DW table generator window	85
Figure A1.2: Inputting DW definition manually	86
Figure A1.3: Display DW definition	86
Figure A1.4: Input all for fact tables	87
Figure A1.5: Input all for banking DW tables	88
Figure A1.6: DW definition in <i>target.txt</i> text file	89
Figure A1.7: DW Input from file	89
Figure A2.1: Source generator	90
Figure A2.2: Input source definition manually	91
Figure A 2.3 Display SDB definition	92
Figure A2.4: Display all of SDB definition	92
Figure A2.5: SDB definition source.txt text file	93
Figure A2.6: Input SDB definition from file	94
Figure A3.1: Integration rule generator window	95
Figure A3.2: Inputting mapping rule	96
Figure A3.3: measurement integration rule	97
Figure A3.4: Mapping rule display	98
Figure A3.5: Structure mapping rule	98
Figure A3.6: Mapping relation text file	99
Figure A3.7: Rule Input from file	100
Figure A3.8: Display the rule	100
Figure A4.1: Target codes execution and display	101

1 Introduction

Many individuals, companies and governments store enormous amounts of information and data already available in traditional database system, text file and WWW information sources. Possible usage of this information for decision support remains limited as long as information and data are stored separately, with no easy means for combining data from different sources. *Data warehouse data integration is a solution* for integrating different source data into a unique format with meaningful information for decision support systems (DSS). But this solution remains difficult during the data warehouse creation and data integration due to several reasons [In99]. The first complication is *distribution* because not every data can be obtained from a single information source. Useful information might be broken into fragments that are distributed among distinct sources. The second complication is *heterogeneity*. Different sources might require different access methods, like HTML pages, relational query languages like SQL or others. Moreover, there might be semantic mismatches between different sources. The same concept might be represented by different words. Even worse, the same word might refer to different concepts. Thirdly, querying unintegrated data sources and integrating the results during querying has the limitation that local data sources may be in use by local transactions, making the process of querying different sources extremely slow. The fourth complication is *instability* because new information sources appear every day while others disappear. Existing information sources change the format of their data, or change their content. The first and the second complications can be addressed by data warehouse data integration. However, the third complication can only be more efficiently handled with the use of a data integration code generator.

This thesis's purpose is to suggest a method that combines the newest data integration techniques and code generator techniques, to provide users a more uniform tool for integrating distributed information sources by generating data warehouse application programs in a database target language such as Pro*C. An extended architecture of the warehouse code generator system that accommodates extension to other target programming languages is also proposed.

1.1 Data Warehouse

A *data warehouse (DW)* is a single, complete and consistent store of data obtained from a variety of data sources and made available to end users in a way they can understand and query for business decisions making [De99(b)]. Data warehouse is a *subject oriented, integrated, nonvolatile, time collection* of data in support of business decision making. Data warehouse is *subject oriented* because it is oriented around the major subject areas of a company, e.g., savings account and checking account might be two subject areas of a banking system. Data warehouse is *integrated* because the different operational applications might be running on different computers and the data might be stored in different databases, e.g. savings account database and checking account database are two different databases running on the machines of different bank departments. Data warehouse is *nonvolatile* means that data warehouse stores historical data and does not have to be updated once the data is loaded into data warehouse. Data warehouse is *time variant* because a data warehouse is designed to store historical data over long periods of time. The structure of the data warehouse always contains some elements of time to answer the question like “What is customer 2000’s balance every year for the last 10 years?”. Data warehouse is *summarized* because the operational data needs to be aggregated for business decision making. Different from traditional relational database, relational data warehouse is *not well normalized* and it is *large*. It is used to store business information for many years.

1.2 Data Warehouse System

A *data warehouse system (DWS)* [De99(b)] consists of data sources, data integration program, operational data store, data transformation program, data warehouse, DW metadata and DW application system. Typical DWS architecture is shown in Figure 1.1

Data sources are the sources from which the data are integrated into operational data store. These data can be in different formats, structures, names and measurements, e.g., the customer ID in savings account database is represented as CID in string data

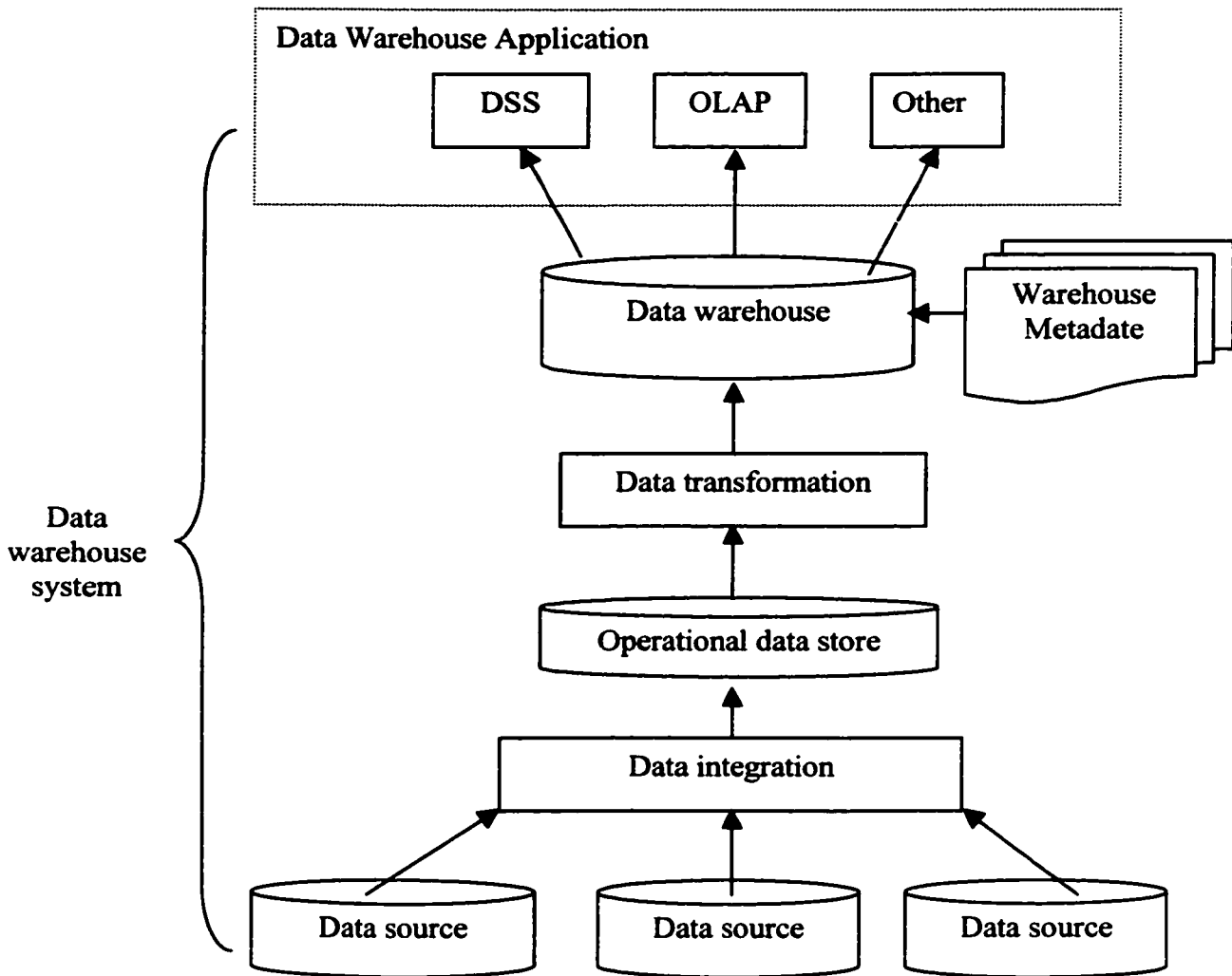


Figure 1.1: Data Warehouse System Architectures

type and in checking account database represented as ID in integer data type. A typical data source that is a traditional database is called *source database* (SDB). *Data integration program* is a set of programs that translate different formats of source data into the unique format of operational data store's data, remove redundancy data and non

useful data, and append data into operational data store. *Operational data store (ODS)* has the same table structure as data warehouse but the data in ODS need to be validated (unified) before being loaded into the data warehouse. Different from the data in data warehouse, data in ODS can be inserted, updated and removed. Data are stored in ODS period time before loading into data warehouse. *Data transformation program* is a set of programs that are used for calculating aggregated values from ODS's data, updating warehouse metadata and moving valid ODS data into data warehouse. *Data warehouse* as defined in Section 1.1 is the data store that stores the data integrated and transformed from data sources for a long period. *Warehouse Metadata* is "data about data" [BZ96]. Warehouse Metadata describes the data in data warehouse such as tables in DW, number of data in each table, last update time etc. *Warehouse application* is a set of systems for warehouse data mining, querying for business decision making.

1.2.1 Data in Data Warehouse System

From the role of data in DWS, data can be internal data, external data and metadata. *Internal Data* is generated by the operational transaction system inside of the enterprise [Ke98], e.g., a banking DWS accepts the transaction information from ATM of this bank's data sources (information from inside of DWS). *External data* is captured from *outside of the enterprise*, e.g., foreign currency exchange rate is from the Canadian government (information from outside of DWS). In DWS, managing data and supporting operations for the data warehouse requires a variety of warehouse metadata [BZ96].

From the type of data in DWS, the data can be structured data, semi-structured data or unstructured data. *Structured data* consists of a group of data which has metadata describing it, e.g., relational source database, relational data warehouse. *Semi-structured data* is the group of data which has no metadata directly available [Ha96]. The basic characteristic of semi-structured data is that they are "self-describing, which means that information generally associated with the data, metadata is specified directly within the data themselves [Bu97], e.g. web sources/HTML file can represent data by <label, value>. *Unstructured data* is the data that metadata is never available, e.g. text files.

1.2.2 Programs in Data Warehouse System

In order to accomplish the task of data movement from data source to DW, two distinct processes – data integration and data transformation are needed [IWG97]. In most of DWS, *data warehouse data integration* (DWDI) includes both the integration and transformation processes. The entire process is made up of data extracting, data cleaning and data loading.

Data integration processes are used to retrieve and combine data from different data sources and to minimize the redundancy in order to support business requirements adequately [BS97]. Data integration in a DWS includes data extracting and data cleaning.

Data extracting is responsible for retrieving data from different data sources and translating the different data formats into unique format [De99(1)]. Typically, the most important *parts* of data extracting for a DWS data integration includes:

- (1) Integrating different measurements. Assume one SDB has chosen to store distance in centimeters and another stores it in inches. When the distance information arrive DW, it needs to be measured in the same unit.
- (2) Integrating different attribute structures. One SDB represents a person's name by last name and first name separately, and another represents it as full name that combines last name and first name in one field.
- (3) Integrating different data types. One SDB represents a SIN number in integer and another source represents it as string. When arriving DW, it needs to be translated in the same data type.
- (4) Integrating different naming conventions: One SDB represents an address as location and another represents it as place.
- (5) Integrating differences in encoding: Database designers have chosen to encode the field of gender in different ways, one represents gender as "M" or "F" and another databases may represent gender as "male" and "female". Whatever source gender comes, it must arrive in the data warehouse in a consistent integrated gender.

Data extraction represents a critical success factor for data warehouse creation [BS97]. There are several needs for data extraction:

Data conversion transfers data from one format to another based on the possible differences among data sources and DW. *Data calculation* is based on the application of the business rules, e.g., the current balance in bank system equals the last balance plus interests.

Data cleaning is a program for removing redundancy from data and dirty data (non-useful data), validating useful data. All data in DWS need to be cleaned and validated [GW98]. The most important *parts* of data cleaning for a DWS data integration includes

(1) Validating means the data has to be correct. Once error or non-useful data slips into data warehouse, it will remain there forever unless some external event occurs. For instance, a data from CID of customer table can be 999030031 and in another table of same database, the data referring the *same person* maybe 999030032 that is errors number. Before putting into DW, the ID must be changed to correct one.

(2) Integrating redundant data (also called cleaning). Some data that have same meaning may appear in more than one data source. *Cleaning* requires that the same piece of data must be referred only once in DW. For example, the tuple for customer (999030031) in customer table of savings account database and the tuple for customer (W999030031) in customer table of checking account database represent exactly same person's information. This customer's personal information only can have one tuple in the customer table of DW.

Data are recorded repeatedly and often in different ways in an organization. As a result, the same information may be referred to in different places. Data warehouse, with its emphasis on data integrity, requires that redundant data be cleaned if it is to be uniform. There are two ways [En99] for data cleaning:

(1) *Data cleaning in data source*: improving the quality of data within the existing data

source. This means standardizing nonstandard data value and domain. Filling in missing data, correcting wrong data and consolidating duplicate occurrences.

(2) *Data cleaning during data extraction*: the objective of data extraction is not just to transfer source data into DW data. The objective also includes improving the existing data quality, filling in missing value and new field.

(3) *Data cleaning in ODS*: the data stored in ODS are not just to store source data temporary. The data also includes improving the existing data quality, unifying data encoding, filling in missing data and removing redundant data.

Data transformation processes, usually referred to data loading, are used to load data from ODS to DW, which has several parts [GW98].

(1) *Data aggregation*: That means that some data in DW are aggregated (e.g., average values). If proper data are aggregated, the user can focus on using the data, not its credibility or consistency in the future.

(2) *Loading changed (new) data*: This entails moving just the changed data in order to reduce the amount of data that has to be moved, which includes both newly obtained transaction information and the changes from the source system.

(3) *Indexing newly loaded records*: Indexing in data warehouse can be used for speeding query performance.

(4) *Updating DW metadata*: This makes sure the DW metadata truly describes the current status of the DW.

1.3 Data/Schema Integration

In the integration field, both schema integration and data integration work are used for integrating different data sources into unified format in data storage. All of the approaches will be discussed in Chapter 2 with example.

1.3.1 Schema integration

Schema integration is defined as the activity of integrating the schemas of existing or proposed databases into a global, unified schema. Schema integration, as defined here, occurs in two contexts [BLN86]:

(1) View integration (in database design), which produces a global conceptual description of a proposed database.

(2) Database integration (in distributed database management), which produces the global schema of a collection of database. This global schema is a virtual view of all databases taken together in a distributed database environment.

1.3.2 Data integration

Mainly, there are three groups of approaches in the field of database (DB) and DW data integration, which are structured approach, semantic approach and virtual approach.

Structured approach is characterized by a self-describing model where each data item has an associated descriptive label e.g., <label value>. Semantic information is effectively encoded in the rule that does the integration. Using structured approach for the data integration is flexible - generality and conciseness of a self-describing model makes the “structured approach” a good candidate for the integration of widely heterogeneous and semi-structured information sources. A set of rules that define view of the data and the set of functions that are invoked to translate data from one format to another is also involved. The schema-less nature of modeled objects is particularly useful when a client does not know in advance the labels or structure of the objects of a source. In traditional data model, a client must be aware of the schema in order to pose a query. With this approach, a client can discover the structure of the information as queries are posed.

Semantic approach has the following characteristics: For each source, conceptual schema must be available. Semantic information is encoded in the schema. A common data model as the basis for describing sharable information must be available. Using semantic approach has some advantages, which include: The schema nature of conventional OO models together with classification aggregation and generalization primitives allow us to organize extensional knowledge and to give a high level abstraction view of information. The adoption of a schema permits to check consistency of instances with respect to their descriptions, and thus to preserve the 'quality' of data. Semantic information encoded into schema permits to efficiently extract information, e.g. to perform query optimization (we will not discuss the query optimization in this thesis). A relevant effort has been devoted to developing OO standard: CORBA for object exchanging among heterogeneous systems.

Virtual approach is a good approach for data warehouse data integration. Different from other approaches, virtual approach is based on source's schema information to create one to one (source to target) relation pair, e.g., customer's name in DW table can be represented by the combination of first name and last name in the source table (called its rule). There is no schema model in virtual data integration approach and source data are integrated only by the integration rules.

1.4 Code Generator

The conventional handing approach for building a DWS is tedious, expensive and time consuming because it often involves coding, adapting, and debugging many programs. Once completed, the DWS is run with SDB and warehouse application to see how well it performs. Because the workload while system testing is different from the actual workload, some of the hand designed and coded features of the DWS have to be sub-optimal [WRT77]. At this point, DW designers have to face two unpleasant options: either leave the data warehouse or redesign/re-code the data warehouse in another round of system development life cycle (SDLC) by more testing. Re-designing has the additional unpleasant side effect that some functions of the DW may change and be re-

coded, which will waste money and also time consuming. To avoid these problems, one solution is using code generator to create DWS. Using code generator can create standard DWS programs based on well known approach, which guarantees not only new DWS can be created fast and correctly, but also updated efficiently when new data sources or application are added into this DWS. Generally, *automatic code generation* is a way to ensure consistency between design and implementation stages of SDLC. The advantages of code generator approach include *reducing* amount of coding and programming time, *improving* the data extraction rules, and *facilitating* data transformation process as such code program is used to transform different data sources.

Code generator [BDB+94] generates computer code or program from object and information descriptions. For *DWDI code generator*, it accepts description about data source, desired DW information and relation rule information between data source and DW. The output of DWDI code generator is a set of DWS programs for DW creation and integration.

More detail, code generation products (a set of programs) are used for data conversion projects, and for building an enterprise wide DWS, when there is a significant amount of data integration and transformation to be done involving a variety of different flat files, non relational, and relational data sources. Code generators create data integration (data extracting and data cleaning) and transformation (data loading) programs based on data source schema and DW data definition. Code generator in a data warehouse will generate a program that translates and transfers data from the source system to a DW based on each group of related data sources. This code generator reduces the need for an organization to write its own DW creating, data extracting, data cleaning and data loading programs.

1.4.1 Code Generator and Compiler

As known, compiler is also used for generating computer programs, which is similar to a code generator. What is the different between a compiler and a code generator? A

compiler is a software unit that translates a program from a input program written by a language to machine code, so that it can be executed directly on the computer. A compiler is used for translating high level programming language such as Java to low level computer language such as executable codes. The language that compiler translates is called the source language while the executable language is the target language. The process of compilation takes place in several phases, the most important of which are shown in Figure 1.2.

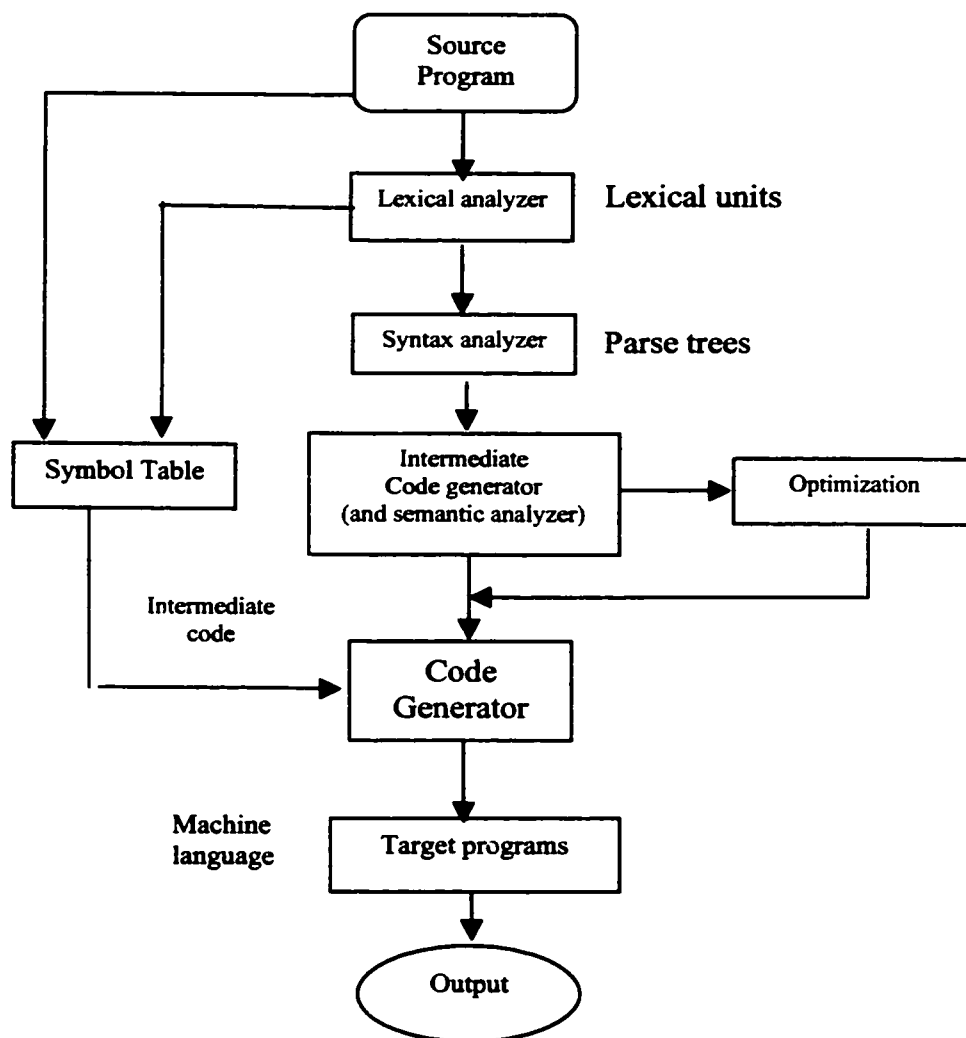


Figure 1.2: Compilation Process

In Figure 1.2, the *lexical analyzer* gathers the characters of the source program into lexical units. The lexical units of a program are identifiers, special words, operators, and punctuation symbols. The *syntax analyzer* takes the lexical units from the lexical analyzer and uses them to construct hierarchical structure to represent the syntactic structure of the program. The *intermediate code generator* produces a program in a different language, at an intermediate level between the source program and the final output of the compiler, the machine language program. The *optimization*, which improves programs by making them smaller or faster or both, is often an optional part of compilation. The *code generator* translates the optimized intermediate code version of the program into an equivalent machine language program. The *code generator* translates the optimized intermediate code version of the program into an equivalent machine language program.

1.4.2 Code Generation for Data Warehouse Data Integration

Many approaches can be used to develop DWS code generator. Figure 1.3 is an example of a code generator based on the virtual data integration approach and template driven code generator approach, which takes source data, source schema and the data integration rules and program templates as input. The output of the code generator is a set of target programs such as Pro*C programs for data warehouse data integration, According to the discussion in Section 1.2.2. These output programs should includes:

- *DW creating program*: DW creating program creates DW based on the user requirement and business need.
- *Data extraction program*: Data extraction program extracts data from different data sources, translates different format data into uniform format data based on data integration rule and also transfers the data into ODS.
- *Data cleaning program* remove the dirty data and duplicate data from ODS.
- *Data loading program* inserts data into data warehouse. Update DW index and DW metadata.

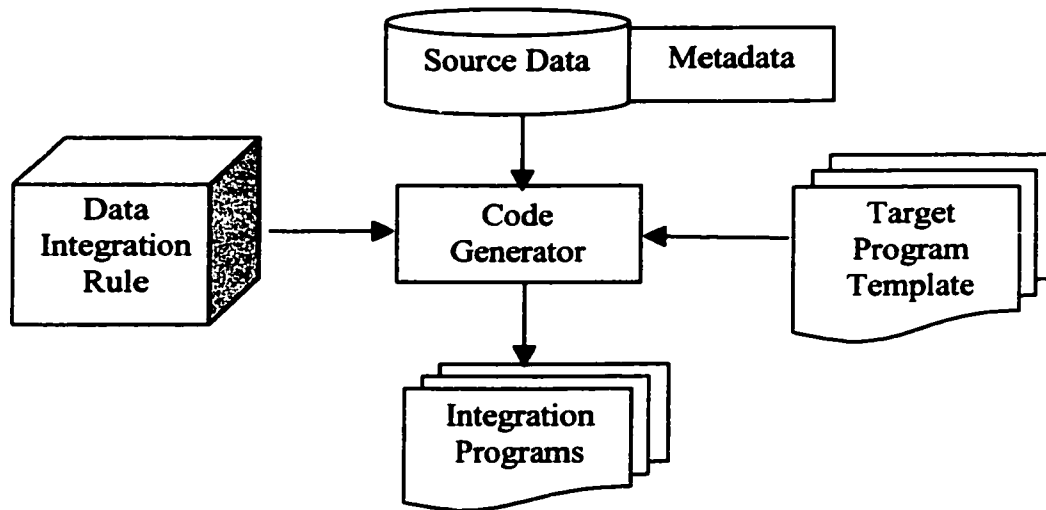


Figure 1.3 Data Warehouse Code Generator

1.4.3 Code Generation Approaches

There are many different code generation approaches today. For example, A *stack based code generator* generates code [KJ92] from source language or information to target language using stack variable memory. This is the most common way in current code generator and compiler. A *rule based code generator* allows the integration of source language or user-specific code by specification rules. The main benefits of rule based code generator includes (1) Enabling organizations to incorporate their knowledge and best practices in the generation rules. (2) Allowing automatic implementation of the models within the desired architectures. (3) Any specification can be implemented in different target environments without changes. (4) Architecture independence. *Pattern-based code generation* generates domain specific application code using patterns, e.g. using UML to generate codes. *Forms-based code generator* is an easy-to-use visual design interface versatile code generator. More code generation approaches include *web based code generator*, *template-driven code generator* and *accumulator based code generator*.

1.5 Thesis Problem and Contributions

As discussed in Section 1.4, code generator is a very useful tool for data warehouse creation, maintenance, data integration and system expansion because of its powers in creating the program codes for integrating different SDB into DW. It can save large labor time, development cost as well as increase the system's reliability. But the problem is:

- Most existing code generation systems try to translate the older language to newer generation language and do not try to generate DWDI programs, e.g., OBLOG is a code generation system which translates COBOL to Java programming language.
- Existing very few DWDI code generation systems create the DWDI programs only for special language based on special data storage. SAS has a DWDI code generation system that only can create SAS programs based on SAS database.
- Existing literatures have not tried to combine the newest integration techniques and new code generation idea into data DWDI code generator.

The proposed system has to be extendable to distributed DW system, web DW system in the future. It can further be extended to output multi target language in order to allow users to choose different target languages in the future. Further extensions include integrating newer data integration approaches and newer code generation approaches in order to make the system flexible. System can be made to operate on different platforms, e.g., on both Unix and personal computer platforms.

How to develop a data warehouse system in particular, data integration, successfully and effectively is a new and important topic. In this thesis, *parts* of problems described above will be addressed in order to establish a basis for future research and development. Specific contributions of this thesis are:

- Finding newer techniques for data integration, which are good for the DWDI.
- Finding code generation approach that is easy to generate many programs and also

can be extended to distributed environment in the future.

- Finding an algorithm that can combine the two techniques in (1) and (2) above for DWDI.
- Making the system extendable for multi target language output and other types of data sources.
- Using this algorithm to build the code generator for data warehouse data integration.

1.6 Outline of the Thesis Proposal Document

The organization of the rest of this document is as follows. Chapter 2 reviews previous work on the schema and data integration approach, code generator approach. Chapter 3 presents a data integration approach, and code generator approach and discusses why they are good for DWDI code generation. Chapter 4 introduces the implementation of *code generator for integration warehouse data* that is based on chosen data integration approach and code generation approaches. An example with a banking system is also demonstrated. In Chapter 5, the conclusions and future work are discussed.

2 Previous/Related Work

This chapter reviews earlier works on schema/data integration and code generation. Section 2.1 briefly reviews previous projects on schema integration and in Section 2.2 more detailed introduction of the structured data integration approach, semantic data integration approach, virtual data integration approach and some other approaches is given. In section 2.3, we will focus on reviewing the code generator approaches and related work in classification of rule-based code generator, stack-based code generator and template-based code generator.

2.1 Schema integration

Schema integration is defined as the activity of integrating the schemas of existing or proposed databases into a global, unified schema, Schema integration, as defined here, occurs in two contexts [BLN86]

- (1) View integration (in database design) for producing a global conceptual description of a proposed database.
- (2) Database integration (in distributed database management) for producing the global schema of a collection of database. This global schema is a virtual view of all databases taken together in a distributed database environment.

An example for integration from [BLN86] shows the structure of a database for large applications (organizations) which is too complex to be modeled by a single designer in a single view. In Figure 2.1, the meaning of "Topics" in the first schema is the same as that of "Keyword" in the second schema. And also "publication" in the second schema is a more abstract concept than "Book" in the first schema. That is, "publication" includes additional things such as proceedings, journals, monographs etc. Because the "topics" and "Keyword" correspond to the same concept, since the schema have to be merged, the names should be unified into a single name. Now change the schema into Figure 2.2. Next look at Figure 2.2, another difference is Publisher is present in the two schemas with different types: It is an entity in the first schema and an attribute in the second. In this case, transform the attribute Publisher into an entity in the second schema

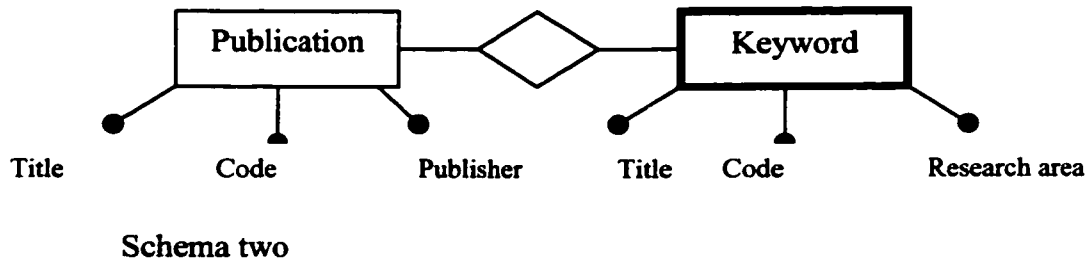
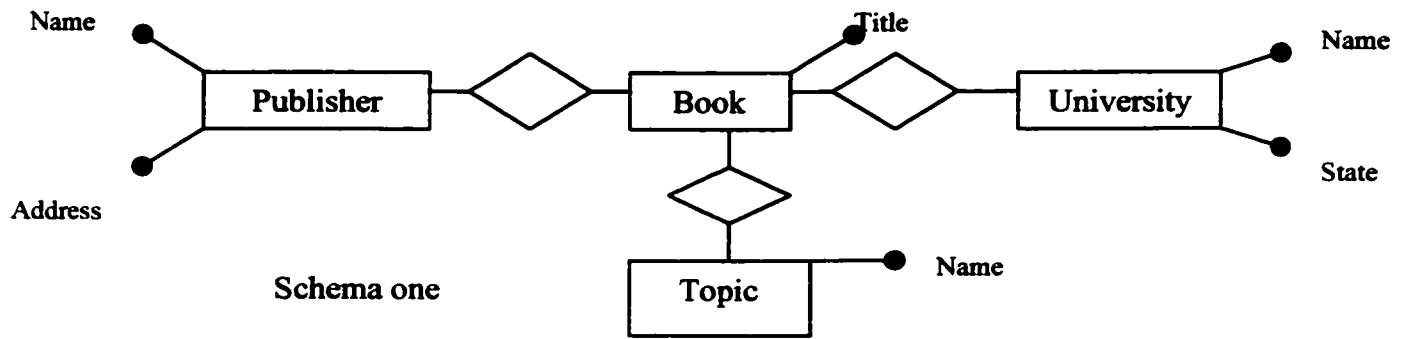


Figure 2.1: Original Schema

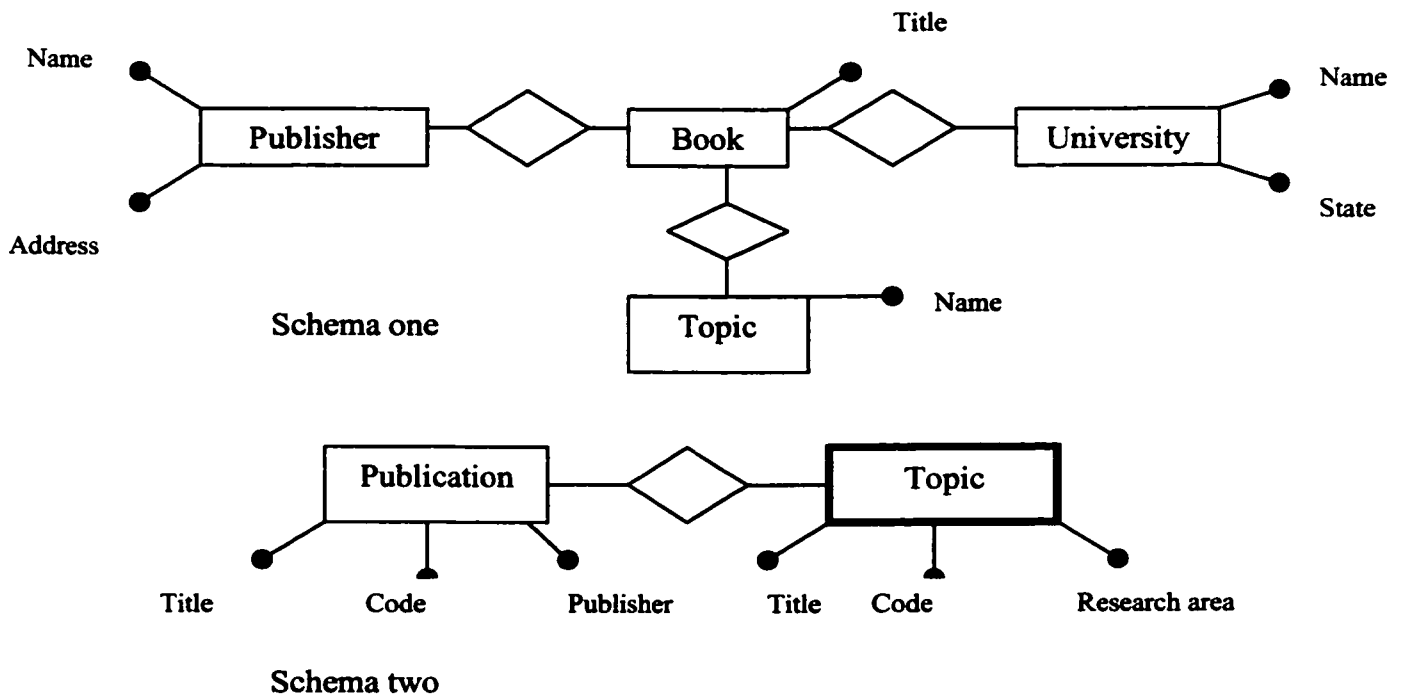


Figure 2.2: Choose "Topic" for "Keyword"

and add a new attribute, Name, to it. The new figure will look like Figure 2.3.

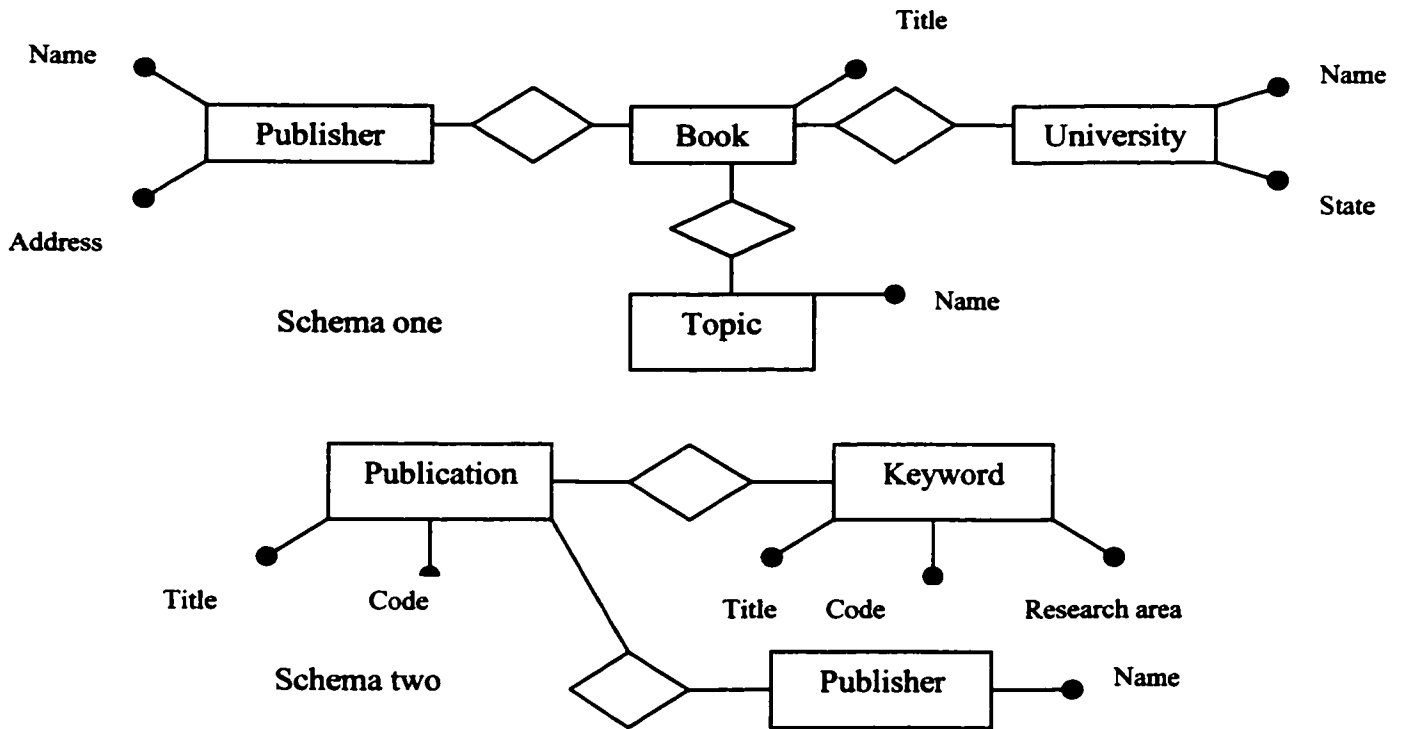


Figure 2.3: Make Publisher into an Entity

The next step is to superimpose the two schemas, producing the representation in Figure 2.4

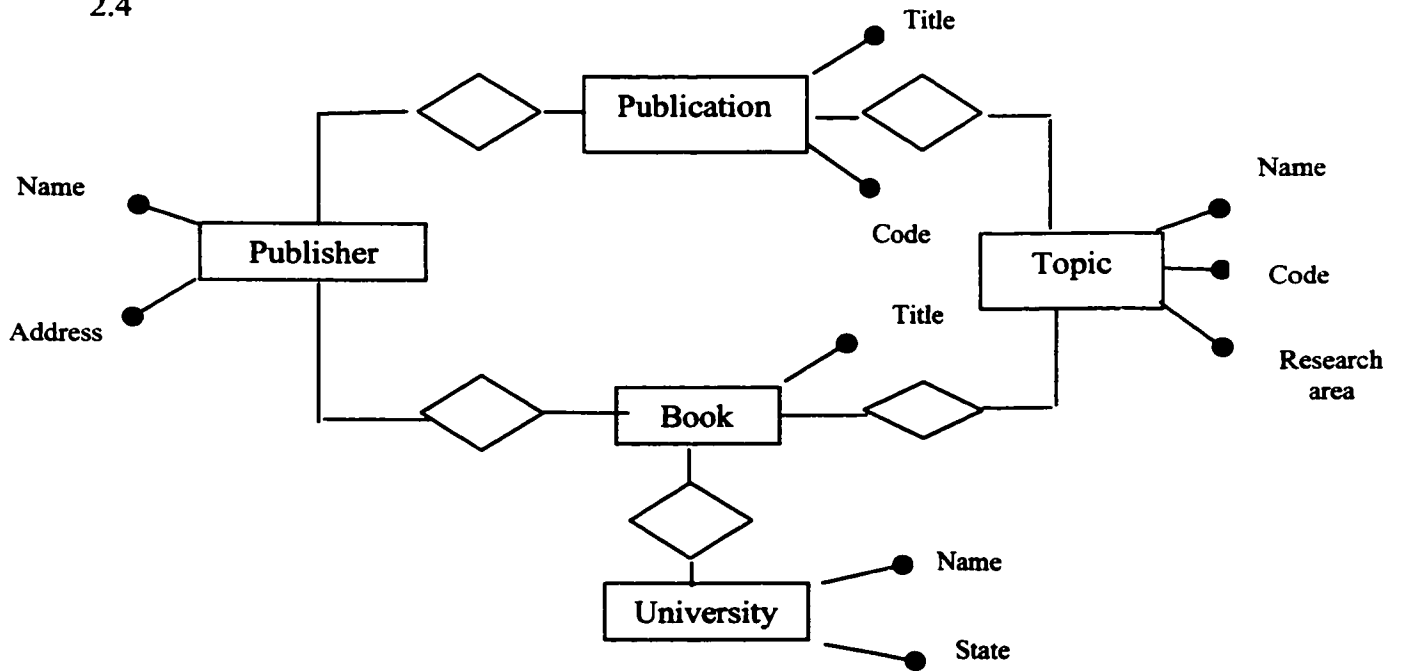


Figure 2.4: Superimposition of Schema

Because of the properties that relate concepts belong to different schemas, which were “hidden” previously. The sub-relationship between the concepts Book and Publication has to be added to the merged schema. This is the case with the subset relationship to the merged schema, producing the result shown in Figure 2.6

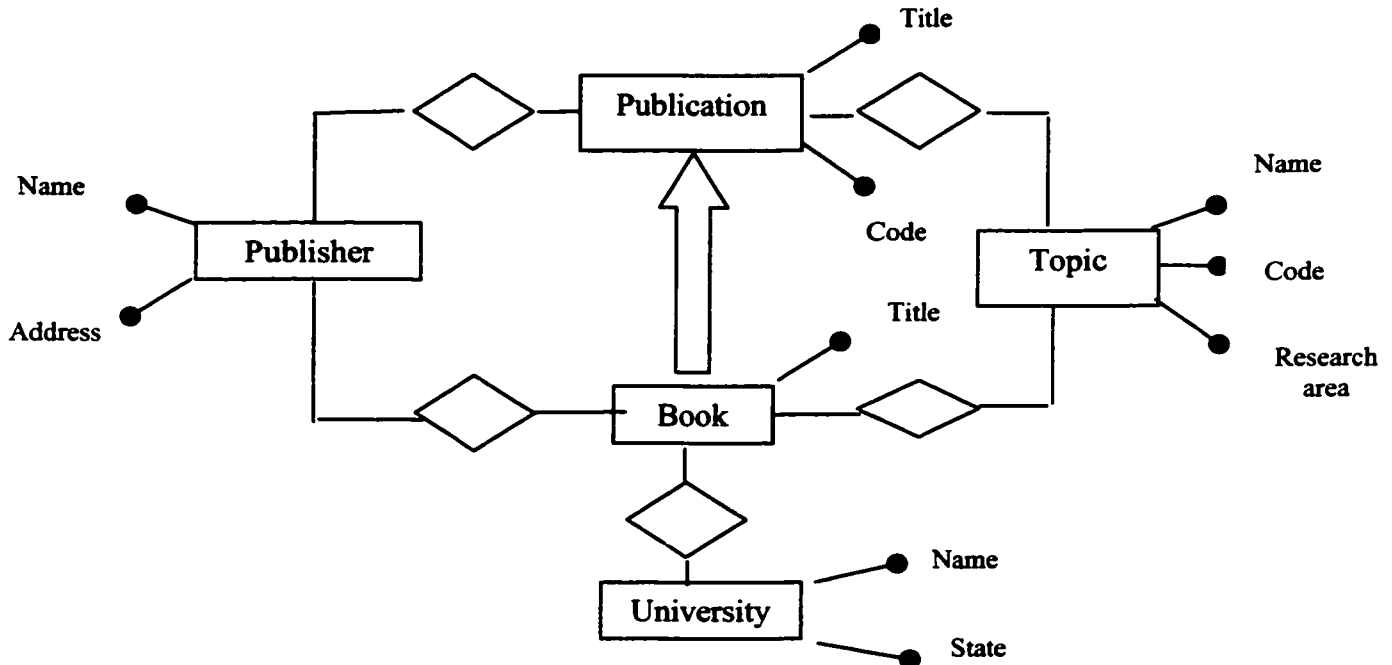


Figure 2.5: Creation of a Subset Relationship

The last is trying to restructure the schema by dropping the properties (relationships and attributes) of Book that is common to Publication, the final schema is shown in Figure 2.7

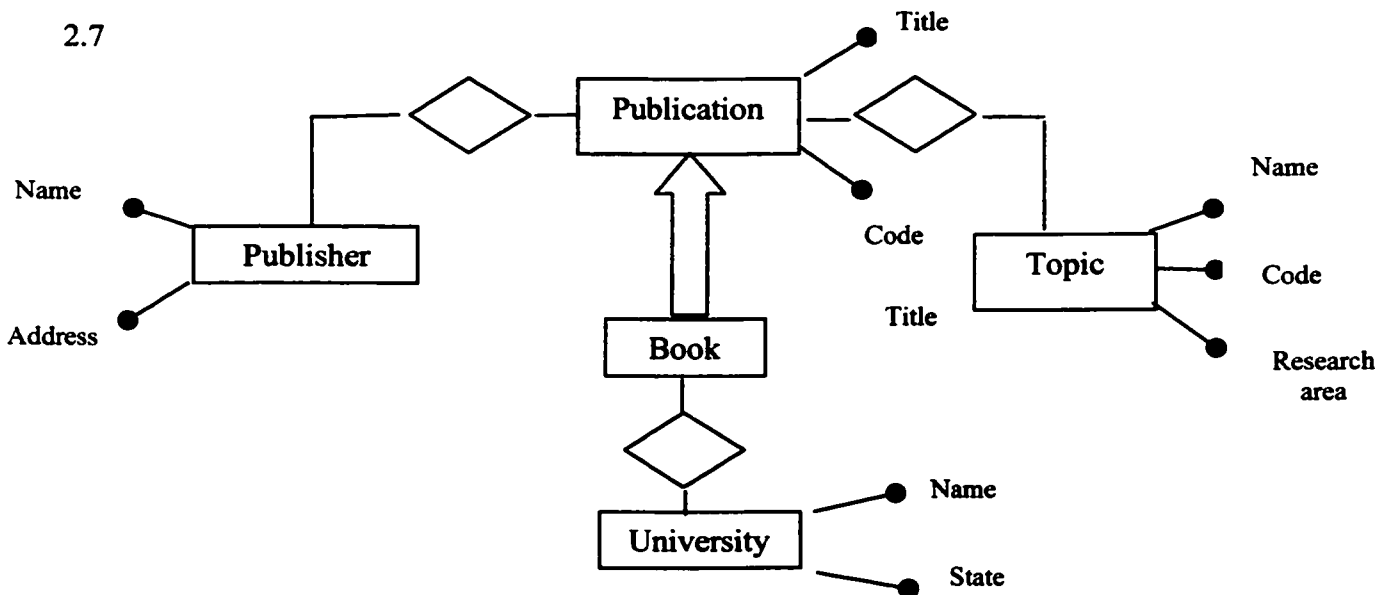


Figure 2.6: An Example of Integration

Summary of the algorithm for the schema integration, which includes the several steps is:

- **Pre-integration:** An analysis of schemas is carried out before integration to decide upon some integration policy. This governs the choice of schemas to be integrated, the order of integration and a possible assignment of preferences to entire schemas or portions of schemas. In this step, collection of additional information relevant to integration is made.
- **Conforming the schemas:** once conflicts are detected, an effort is made to resolve them so that the merging of various schemas is possible. Automatic conflict resolution is generally not feasible; close integration with designers and users is required before compromises can be achieved in any real-life integration activity
- **Merging and restructuring** now the schema is ready to be superimposed, giving rise to some intermediate integrated schemas. The intermediate results are analyzed and if, necessary restructured in order to achieve several desirables.

Discussion: Schema integration is important for DWDI code generation because DWDI code generation needs to retrieve schema description from data sources, translate (but not merge) them into DW schema. The products of DWDI code generator are a set of programs that are based on the data sources and DW schema to integrate source data into DW tables. However, the schema integration from [BLN86] uses merging and restructuring to create a unique database schema, which will not satisfy user interesting DW structure. DW structure is different from SDB structure, e.g., DW can use star structure (one fact table and many dimension tables around this fact table to describe each attribute of this fact table) to organize its tables. This approach is also around the data source's schemas to create a new schema, not around the major subject of an enterprise to create a DW schema for business decision making. As discussed in Section 1.2.1, DW source data can be internal data (need schema integration), external data (foreign exchange rate from outside of DWS), DW metadata (more attribute than database schema) and some constant values. External data, DW metadata and constant value are outside of database schema. Because of these limitations, some concept, such as replace the "Topic" by "Book" (Integrate different naming conventions), will be used in DWDI code generator but some ideas will be retrieved from other data integration approaches.

2.2 Data Integration Systems

Following the classification of integration systems proposed by [Hu97], some virtual approaches appeared first in the early 80s. For example, the MOMIS, a virtual approach, was first proposed in multi-database models [BBC+00]. More recently, systems have been developed based on the use of description logic [LRO96] such as CLASSIC [BBM+89]. All of the virtual approaches are based on a model of query decomposition, which sends sub-query to source databases, and merges the answers through an integrated view that holds a subset of the complete answer implied by the underlying database. This approach is useful only if all sources are traditional databases.

For schema integration, a top-down approach is used: in essence a global schema encompassing all relevant information is created, and data held in the source database is expressed as views over this global schema [U197].

The GARLIC project [CHS+94] built upon complex wrapper architecture to describe the local sources with an OO language (GDL), and on the definition of Garlic complex objects to manually unify the local sources to define a global schema. The SIMS project [AKH96] creates a global schema definition based on the use of description logic (i.e. the LOOM language) for describing information sources. The use of a global schema allows both GARLIC and SIMS projects to support every possible user queries on the schema instead of a predefined subset of them.

Information Manifold System [LRO96], as the MOMIS project, provides a source independent, query independent mediator [BB99]. The input schema of an Information manifold system is a set of descriptions of the sources; so, given a query, the system will create a plan for answering the query using the source. The algorithms to decide the useful information sources and to generate the query plan are provided. With respect to the input schema generation, it is completely modeled by the user.

Infomaster system [CGL+98] provides integrated access to multiple distributed heterogeneous information sources giving the illusion of a centralized, homogeneous information system. It is based on a global schema, completely modeled by the user, and a core system that dynamically determines an efficient plan to answer the user queries by using translation rules that harmonize heterogeneous sources.

Another proposal based on the Description Logic and Reasoning techniques is described in [Ra98], where a declarative approach (semantic approach) is used. The framework provides inter-model assertions to define inter-relationships between concepts in different sources. The inter-model assertions may define both at intentional (similarly to our terminological relationships) and extensional levels. In the proposal, the definition of the global schema (called Enterprise Model) is a manual task.

On the other hand, other projects that are based on a structured approach [CMH+94] (for example, TSIMMIS) follow a structural approach and use a self-describing model (OEM-object exchange model) to represent the data object and pattern matching techniques to perform a predefined set of queries based on a query template. The semantic knowledge is effectively encoded in the MSL (Mediator Specification Language) rule enforcing source integration at the mediator level. Although the generality and conciseness of OEM and MSL make this approach a good candidate for the integration of widely heterogeneous and semi-structured information sources, a major drawback in such an approach is that dynamically adding sources is an expensive task. In fact, new TSIMMIS sources not only must be wrapped, but the mediators that use them have to be redefined and their MSL definitions recompiled [Ca95]. The administrator of the system must figure out whether and how to use the new sources.

Some data integration projects are typically based on structured data integration approach while others are based on semantic data integration approach or virtual data integration approach. TSIMMIS is a data integration system that is based on structured data integration approach, SIMS is a system of semantic data integration approach and Infomaster is a virtual data integration approach system.

2.2.1 TSIMMIS

[QRS+95] introduce the TSIMMIS. TSIMMIS is a structured data integration system by the definition of [Hu97], which provides integrated access via wrappers/translators such that *wrappers convert data into a common model; mediators combine, integrate or refine the data from common model of the wrapper*. The wrappers also provide a common query language for information [MPQ97]. Applications can access data directly through the wrapper but they can also go through mediators as Figure 2.8.

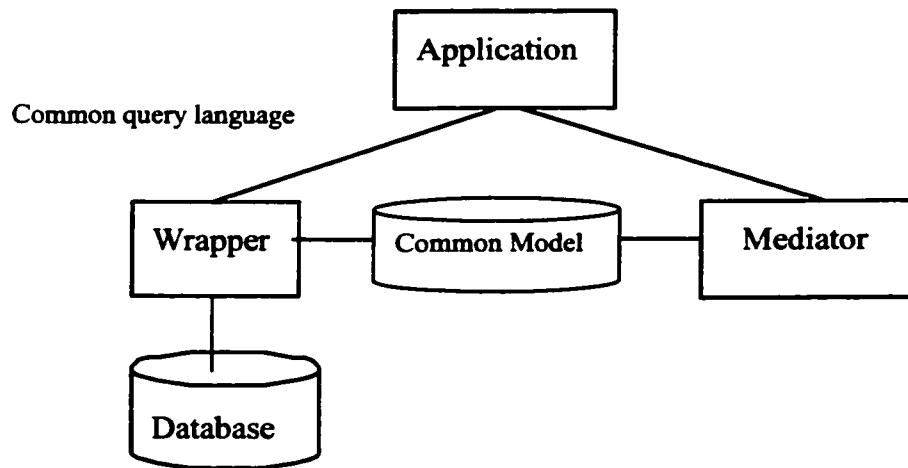


Figure 2.7: The Architecture of TSIMMIS

In Figure 2.8, each source has a wrapper that logically converts the underlying data objects to a common information/data model (called OEM-Object exchange model. Figure 2.9 is an example of a data model) and a query - like language (called MSL-Mediator specification language in figure 2.10) that extracts information from the data model.

Id	Label	Type	Value
Ob1	person	set	{sub1, sub2, sub3, sub4, sub5}
Sub1:	last_name,	string,	'Smith'
Sbu2:	first_name,	string,	'Jone'
Sub3:	role,	string,	'Faculty'
Sub4:	department,	string,	'cs'
Sub5:	telephone,	string,	'519-2563000-3003'

Figure 2.8: Example of TSIMMIS Data Model

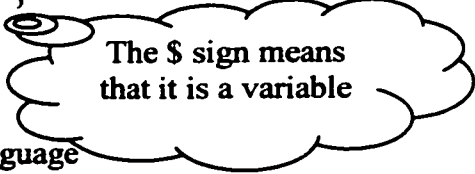
Query

(Q1) Query ::= *0: - <0 person {< last_name \$LN > } >

(Q2) Query ::= *0: - <0 person {< last_name \$LN > } >

{< first_name \$FN > } >

(Q3) Query ::= *0: - <0 person {<role \$R> } >



The \$ sign means that it is a variable

Figure 2.9: Example of TSIMMIS Query Language

From Figure 2.9, the data description including data type, name is in the data model, so it is structured data model. The data integration based on this structured model is the structured data integration approach. Figure 2.10 shows a query language. In fact, it is a set of query templates stored in the wrapper. Any query entered by the user must match one or more of queries from the template, e.g., assume that a user input query is Q: Q person {<last_name 'smith'> <role student>}. Divide the whole user input query Q into two sub-queries. In the sub-query σ_1 : σ_1 person {<last_name 'smith'>} can be found from query template (Q1 in Figure 2.10) and in the sub-query Ω_2 : Ω_2 person {<role 'Faculty'>} can be filtered from data model (Q3 in Figure 2.10) based on the last name. In TSIMMIS, there are three types of supported queries [BC97], which includes (1) directly supported query – user queries with a syntax that is similar to the query in template, (2) logical supported queries - two queries are logically equivalent if they give the same answer set in the same context. And (3) indirectly supported queries - a query is not supported by a template directly but can be decomposed in a template-supported query and a filter query, which is the same as the example above.

Discussion: TSIMMIS is a typical structured data integration approach for database data integration, which is based on a structured model and a set of queries written by a query language to integration data. This approach is flexible because the data and schema are described in the structured data model, which is independent of source schema and data. It is good for already known business data integration because both label and data pairs <label, data> are put in the structured model. The limitation is that it is impossible to extend to all business in the world because the model can be very huge if many

businesses of the world are included. If any new business appears, the model also has to be updated for both label and data. Because relationships used by DWDI code generator are always different each time, when creating new DWS or updating an existing DWS, creating and keeping a huge model is unnecessary and it is also impossible to cover all of the business in the world. The limitations make this approach not a good candidate for DWDI code generator.

2.2.2 SIMS

SIMS uses semantic data model to integrate information from various information sources. There are two basic components used in SIMS according to [AKH96].

(1) Data model: which is used to describe the information stored in the sources and the relationship among these sources. Figure 2.4 is an example of this data model that describe the ports and port locations, shadow circles are real data sources and bright circles are conceptual data models that link the different data sources by semantic. Any data source that links to another sources must be through conceptual data model, e.g., A and B are both *sea port* because both of them link to sea port conceptual model. A and C are *port* because both of them link to port conceptual model through sea port and air port. This data model represents semantic data model because any data source linked through some meaningful conceptual model.

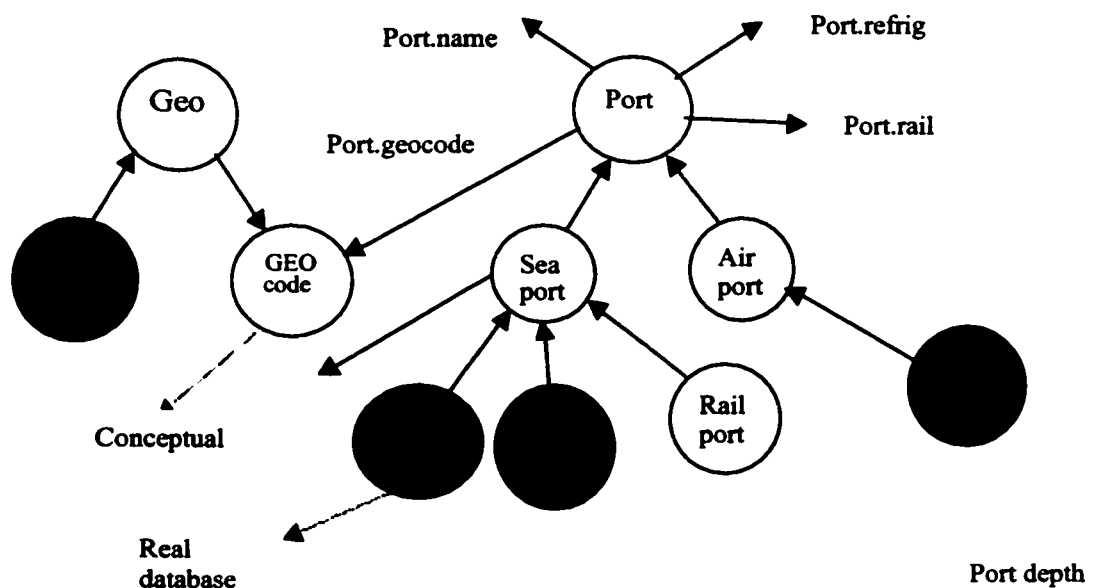


Figure 2.10: Example of SIMS Data Model

(2) Query: which is used to construct a sequence of queries (called Loom in SIMS) to individual information sources. It queries data based on the data model, e.g., Figure 2.5 is a query requiring a rail port in Germany, the query will start from the name (line 1), rail port (line 2) because from data model, rail port is seaport following the port search geologic location –Germany (line 3-7 in Figure 2.5).

1. (retrieve (? name)
2. (:and (rail_port ?port)
3. (port.geocode ?port ? geocode)
4. (prot.name ?port ? name)
5. (geoloc ? geoloc)
6. (geoloc.country_name ? geoloc "Germany")
7. (geoloc.geocode ? geoloc ? geocode)))

Figure 2.11: Example of Query Language

Arens in [AC99] describe the algorithm of SIMS as follows:

- Determine which information sources contain the data relevant to the database classes used in formulating a given query (select information sources).
- For those classes mentioned in the query which appear to have no matching information source, determine if any knowledge encoded in the domain model (such as relationship to other classes) permits reformulation in a way that will enable suitable information sources to be identified (query plan).
- Use knowledge about databases to optimize the plan.
- In general, provide a uniform way to describe information sources to the system, so that data in them is accessible.

Discussion: Semantic data integration approach is popular since 1990' with Internet development. This approach is based on a knowledge model to maintain the relations of data among data sources and mainly for database and web data integration. Because the relations in knowledge model link each pair of data that have same meanings (or one include another), it is very good candidate for semi-structured data integration, e.g., web page information. The idea of finding each semantically similar (same or including) data

pair is very useful for the DWDI code generation because designing a DWDI code generation system needs to create relations for each pair of attributes that have similar meanings (even if the data in each pair have different attribute names). But the limitation of semantic approach is that it does not discuss integrating different data structures and encoding (See Section 1.2.2). Another is this approach asks to create a knowledge model that keeps all relationship pairs for future use. Because relationships used by DWDI code generator are always different each time, when creating new DWS or updating an existing DWS, creating and keeping a knowledge base is unnecessary and it is also impossible to cover all of the business in the world. This limitation makes this approach not a good candidate for DWDI code generator because DWDI code generator should be used to create programs for any business.

2.2.3 Infomaster

Infomaster is an information integration tool that solves problems based on the virtual data integration approach [GGK95]. In virtual data integration approach, there is no predefined data model and any query is based on a set of query rules. Infomaster designs a strategy for answering the query and performs translations to convert source data to a common form, which is shown in Figure 2.5. This approach is try to translates different format data sources into a common format based on correspondent rule.

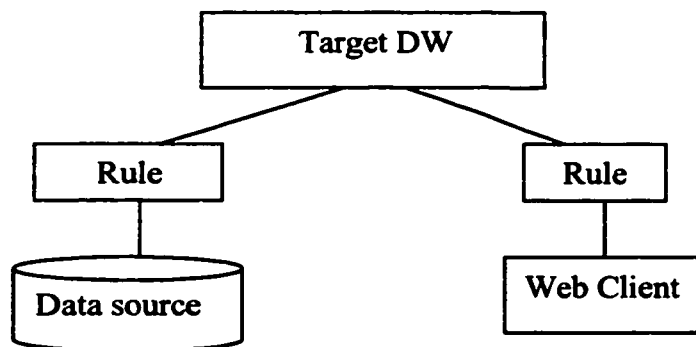


Figure 2.12: Architecture of Infomaster

Infomaster uses rules and constraints to describe information sources and translation into common form (target DW). Every translation will be based on these rules and constraints.

An example from [GKD97], supposing there is source with the transferring rules, Mercedes car table (Figure 2.7) and the target *common table* (Figure 2.8). The purpose of this demonstration is to integrate different data from different tables to the common (target) table, e.g., rule 1 say Mercedes sport car is a Mercedes car with 2 doors. So in target table, the Mercedes sport car will be 2 doors and 2 seats (from rule 4) after the data are extracted from source table.

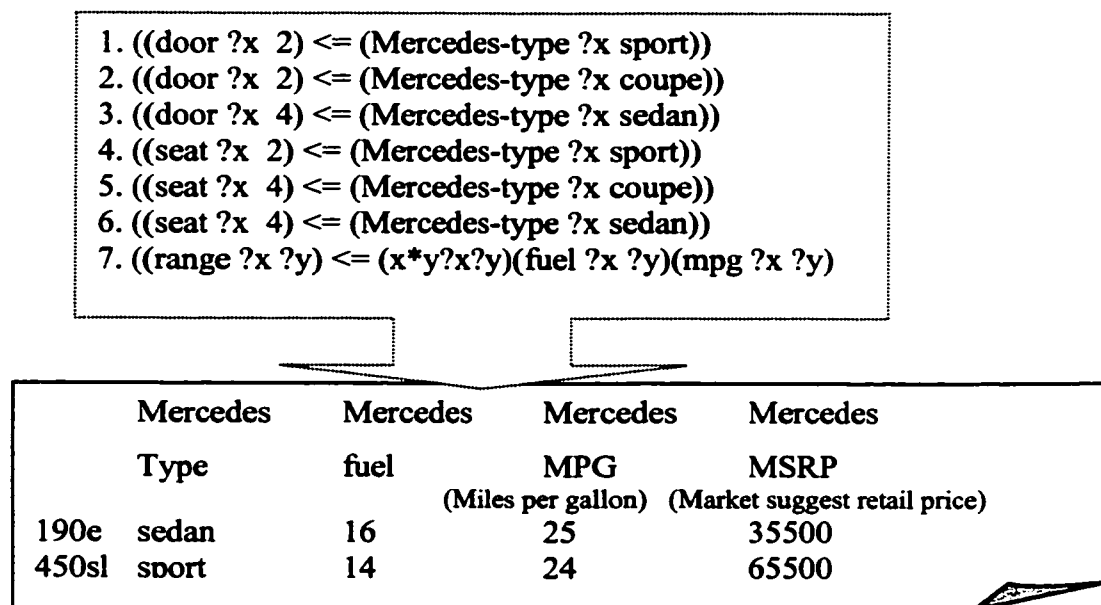


Figure 2.13: Mercedes Car Table and Transferring Constraints

In Mercedes car source table (Figure 2.7), the attributes appear different from the target table (Figure 2.9). Except the name and Mrsp, the type, fuel and Mpg are not in target table. Searching target table, the name can be got from the table's name—Mercedes. The range can be obtained from fuel * Mpg. There is no information about doors and seats from source table. In this situation, additional information in the rule is required, e.g., from the first rule (<= (door ?x 2) (Mercedes-type ?x sport)) and the forth rule (<= (seat ?x 2) (Mercedes-type ?x sport)) in Figure 2.7, because the Mercedes car

type is *sport*, there are two doors and two seats corresponding to *sport* car. Again, from the third rule ($\leq (\text{door } ?x \ 4) (\text{Mercedes-type } ?x \ \text{sedan})$) and the sixth rule ($\leq (\text{seat } ?x \ 4) (\text{Mercedes-type } ?x \ \text{sedan})$) in Figure 2.7, because the Mercedes car type is *sedan*, there are four doors and four seats corresponding to *sedan* car. Filling the information offered by rules into the target table, the integrated common table in Figure 2.8 is obtained.

Model	makes	doors	seats	range	MSRP (Market suggest retail price)
190e	mercedes	4	4	400	35500
450sl	mercedes	2	2	336	65500

Figure 2.14 Common (target) Table

The algorithm of Infomaster consists of *query planning* and *query execution*. The *query planning algorithm* takes a query Q , a collection Δ of rules and integrity constraints (like definitions) as inputs. The output of the algorithm is a query process plan suitable for input to the plan execution algorithm. The *plan execution algorithm* takes a query plan as input. It retrieves data from the available databases and merges this as described in the plan. The output is a table of answers to the original query. In the query planning algorithm, the step includes reduction, abduction, conjunctive and disjunctive minimization and grouping [DG97]:

- (1) *Reduction*: In this step, the system rewrites each atom in the query using the definition of the associated predicate. It then repeats the process until it obtains an expression in terms of base relations (which, by definition, have no definitions of their own.) Termination is assured due to the non-recursive nature of the definitions.
- (2) *Abduction*: Given an expression ϕ in terms of base relations and a set of definitions, the abduction process produces the set R of all consistent and minimal conjunctions of retrievable atoms that can be shown from the definitions to be contained in ϕ .
- (3) *Conjunctive Minimization*: In this step, eliminating any redundant conjunct, i.e. one that can be shown to contain the remaining conjuncts.

(4) *Disjunctive Minimization* In this step, dropping any disjunction that can be shown to be contained in the union of the remaining disjunction.

(5) *Grouping*. Finally, the conjuncts within each conjunction are grouped so that the atoms in each group all share a common provider.

Discussion: Virtual data integration approach is for DW data integration. Different from schema integration, structured data integration and semantic data integration, virtual data integration approach retrieves source information from each source one by one, converts the information into target formats information and put them into target DW. The main idea is only to create the relationships between each data source and target DW, do not think about the relations among these data sources. It can be called integrating vertically and others are horizontal. See Figure 2.15

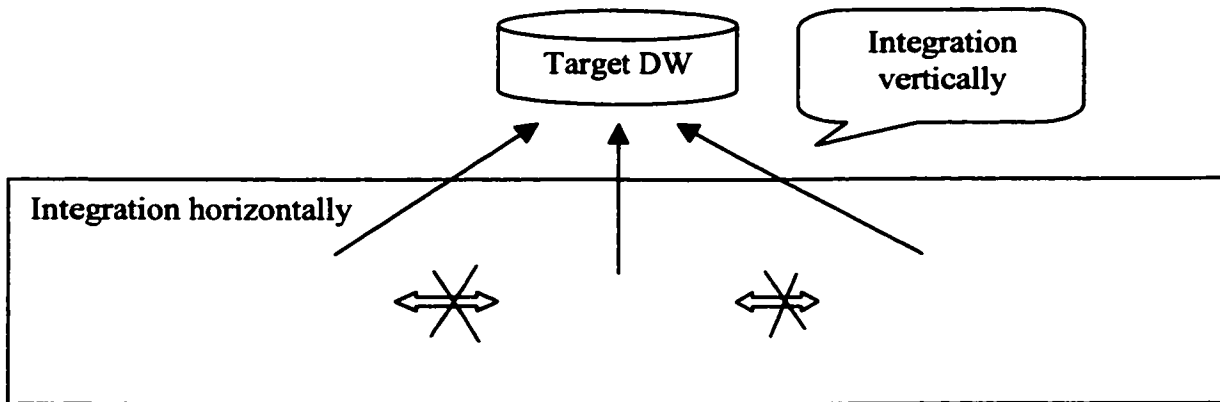


Figure 2.15: Vertical and Horizontal Integration

This approach presents an idea about finding each pair of relations between each source data and target DW data, e.g., one pair is $customer.ID \leftarrow savingCustomer.CID$. Here, $customer.ID$ is an attribute of DW customer table and $savingCustomer.CID$ is an attribute of customer table in savings account database. It means that different from schema integration, structured integration and semantic integration, the target DW table must exist first in virtual data integration approach in order to create the relation pairs between DW and SDB. DW table can not be created after integration. Virtual data integration approach does not need to create any model to store these relationships. This approach

only uses these relations to retrieve data from source to target DW one by one. It is very good when adding a new data source, removing a data source or updating a data source because it is only referred to create, remove and delete new relationships between this new data source and target DW (not referred to any other data sources). One to one relation will create many redundant data in target DW; e.g., a customer 999030031 is in savings account database and it is also in checking account database too. Because there are two relation pairs here, one is between DW and savings account and another is between DW and checking account, both pairs will load their own data into target DW. Because DWDI programs include a “data cleaning” program, this problem can be solved during data cleaning. It satisfies the basic needs for the code generator (see Section 1.4.2). It will be a good candidate for the DWDI code generator when only integrating schema using this approach to create the DWDI programs.

2.2.4 Dynamic Query approach

[Mi98] suggests using dynamic view to integrate different data sources. A dynamic view is defined by a query language called *SchemaSQL*. In *SchemaSQL*, attribute, relation and database variables are collectively referred to as schema variables. Assume there are 3 databases as shown in Figure 2.9:

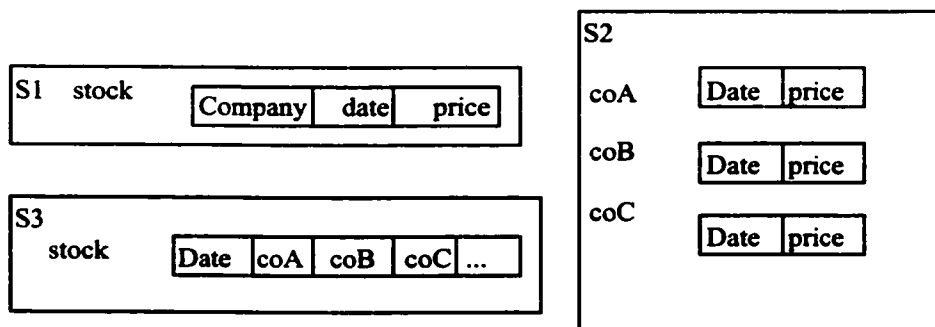


Figure 2.16: Three Source Databases

Using the tool "schemaSQL", in which set of the database, relation, attribute and tuple are variables called schema variables that can appear in the SQL query statement. So Dynamic View is the schema that is defined by the output of query. By traditional way, the integration by view from s2 to s1 will be the top chart below:

```
create view v1 (co, date, price ) as
( select coA, date, price from coA
  union
  select coB, date, price from coB
  union
  select coC, date, price from coC)

Create view v2 (co, date, price) as
select R, T.date, T.price
  from s2->R, R T
create view v3 (co, date, price) as
select A, T.date, T.A
  from s3::stock->A, s3::stock T
```

Figure 2.17: Traditional and Dynamic Query

And it is no way to integrate it from s3 to s1 by traditional way. The bottom chart is integrating by dynamic view (R is relation variable, T is tuple variable, A is attribute variable).

Discussion: Dynamic query data integration is an integration approach based on a query language called dynamic SQL. Discussing this approach gives idea on how the output programs of the DWDI code generator can handle data integration when the output target program language is dynamic. Because DWDI code generator will generate non-dynamic language, the attribute relations must be assigned into the output programs. Naturally, this approach will not be candidate for thesis project.

2.3 Some Code Generator Systems

There are many code generator projects. Some projects are stack based code generators e.g., GENTLE [Ge97] is a stack based code generator. Some projects are rule based code

generator that allow generating code by specification rule, e.g., OBLOG is a rule based code generator for transforming older computer languages programs into some new generation computer language programs [Ko92]. Pattern-based code generation using the approach that generates domain specific application code, e.g., using UML to create programs. Forms-based code generator use easy-to-use visual interface to enter the source information. Other types of code generators include web based code generator, template-driven code generator e.g., Template Manager, accumulator based code generator and so on. Of course, some projects combine several approaches to create program code.

In data warehouse's code generator projects, SAS Warehouse Administrator [SWA99] is a data warehouse code generator, which is a table-driven code generator.

2.3.1 SAS

SAS/Warehouse Administrator [SWA99] is a customizable solution that offers a single point of control, making it easier to respond to the ever-changing needs of the business community. It (1) integrates extraction, transformation and loading tools for building and managing data warehouses. (2) Provides a framework for effective warehouse management through metadata. With graphical user interface, SAS can simplify the visualization, navigation and maintenance of the data warehouse. SAS integrates data source using a set of processes, e.g., *Address Standardization Add-In* routine demonstrates how to use an add-in tool to standardize an address in a data warehouse environment. The algorithm (1) Breaks down address strings to tokens. (2) Standardizes tokens. (3) Categorizes and rearranges tokens. (4) Computes scores for evaluating the extent of standardization. (5) Outputs tokens classified as unknown by the routine to a separate data set for further investigation. (6) Constructs standardized addresses. As a result, this algorithm converts different formats of the same address to a single standardized form. For example, the two addresses: Apart 7, 123 Northwest Main Street and 123 NW Main Str, Apartment 7 would both be standardized to 123 NW MAIN ST APT 7. Another example called *Data Step Mapping Add-In program* demonstrates a user-written one-to-one data mapping process by extracting information by metadata.

Discussion: The limitation of SAS Warehouse Administrator is only setting up and managing SAS data warehouse with SAS data and no evidence shows new data integration approaches used. It can not be used for Oracle database so that it is not extendable. It can not be flexible to generate any other language program for data warehouse data integration, which will not satisfy to fill the gap of thesis problem discussed in Section 1.4. SAS is not a good candidate for DWDI code generation project.

2.3.2 OBLOG

OBLOG [Ko92] is a rule based code generator for transforming computer languages from older to newer. In order to specify and execute the generation rule, a script language is used. This language is used for defining rules to extract and manipulate information. This tool provides flexible user-extensibility to several common tasks performed in a development project, like code and documentation generation, impact analysis, repository maintenance and model transformations etc. A class is shown in Figure 2.11

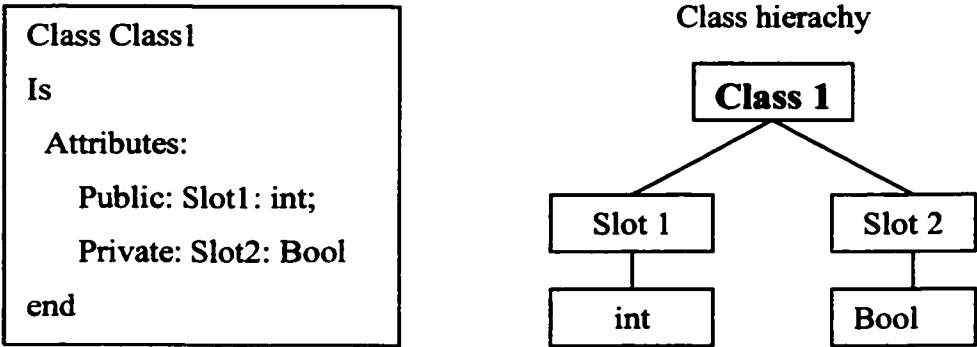


Figure 2.18: Example of Target Program and Its Hierarchy

When the code documentation is created, the rule language in the Figure 2.12 to issue how the documentation will be the same as in this rule language. From line 1 to line 13, the rule defines the class header and templates for each slot. Lines 13 to 17 define the

first slot and lines 18 to 23 define the second slot. The limitation of this code generator approach is that for a generated language, a new language, so called rule-defining language, is introduced. This approach is flexible for generating code but the users have to learn a new language. It is not easy to use.

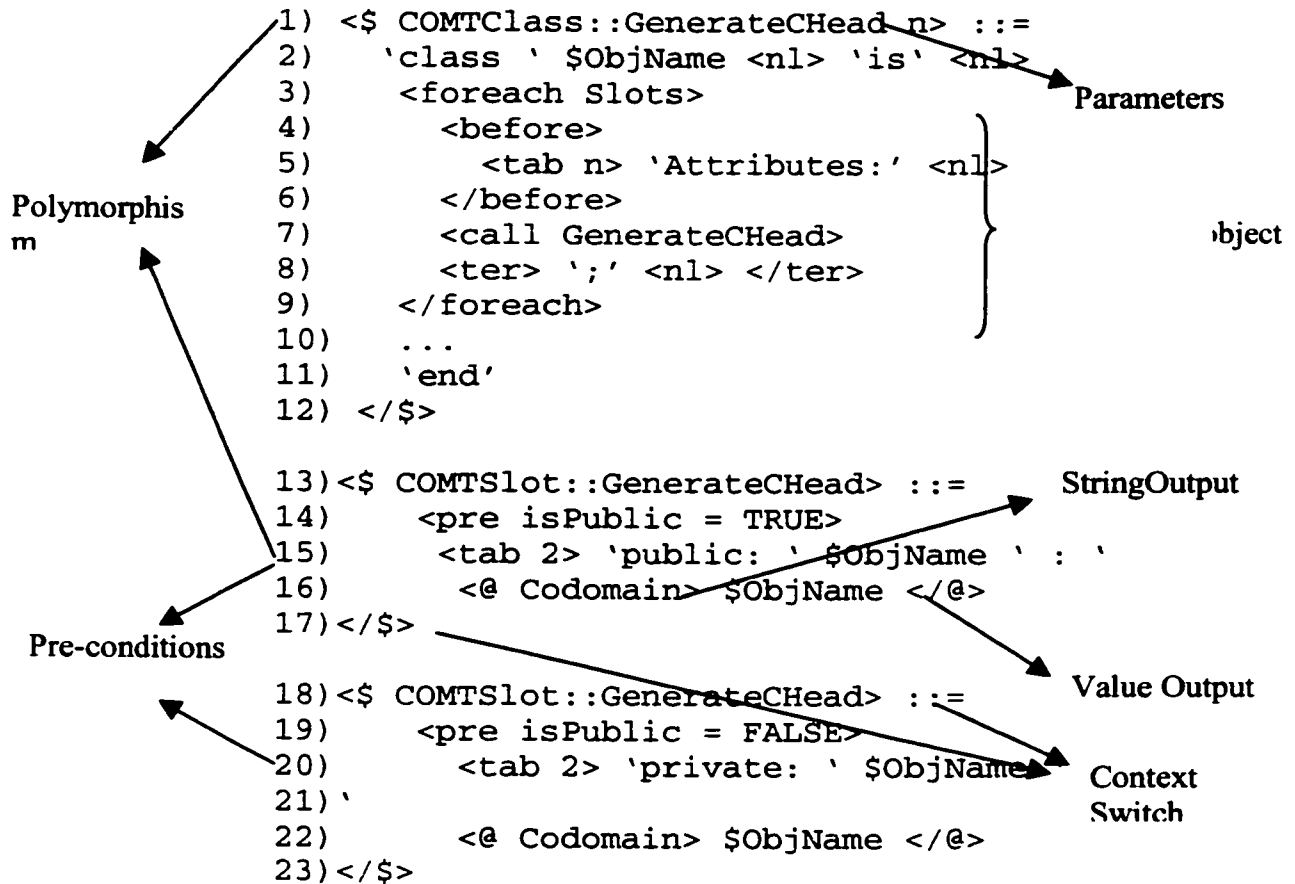


Figure 2.19: Rules Example for Code Generation

Discussion: Rule based code generator is an extendable approach, which can generate many kinds of output language. The pair of \$ sign and /\$ sign frames each part of output programs and the single \$ and @ sign assigned the variable's position in the output program. This program frame can be written in Java-like language to generate the Java program, it also can be write in C like language to generate the C program. This advantage makes this approach a good candidate for DWDI code generator. This

approach's limitation is all the output programs have to be in a single program frame (only variable are different), which make this approach less flexible because DWDI code generator can create different programs based on the real world business situation. For example, some DW need to remove the redundant data but some only need to integrate the different encoding during data cleaning. It is unnecessary to put any non-useful program block into output programs.

2.2.3 GENTLE

GENTLE [Ge97] is a stack based code generator and expressions are defined in Figure 2.13, part of the expressions, for example the assignment statement, plus operation, minus operation and variable, are defined

```
'type' Statement
assign (Variable, Expression) //Assign the result of expression to the variable
'type' Expression
plus (Expression, Expression)
minus (Expression, Expression)
var (Variable)
'action' PLUS
'action' MINUS
'action' PUSH(Variable)
'action' POP(Variable)
```

Figure 2.20: GENTLE expression

In Figure 2.20, the instructions modify the stack of the computer. If the stack has the form ... X Y, Then PLUS replaces the top two elements by their sum $Z = X+Y$, i.e. the stack becomes ... Z. Similarly, MINUS replaces X and Y by $Z = X-Y$. If K is the value of a variable V, then PUSH (V) puts K onto the stack. If K is the value on top of the stack, then POP (V) removes it from the stack and stores it in V. Assume two

predicates *Encode* and *StackCode* that emit instructions for statements and expressions, e.g., suppose *V* is a variable and *X* and *Y* are expression, If generating code for assign (*V*, *X*), this way has to emit code for the expression *X* that computes its value on top of the stack. Then, use the POP instruction to store it in *V*, which is in rule 1 of Figure 2.13. For the predicate *StackCode* in rule 1, it processes the 'plus', 'minus' and 'var' for expression in rule 2, 3 and 4 in Figure 2.13 recursively: (this is defined recursively and *X* and *Y* in rule 2 and 3 can be another expression).

'Rule 1' Encode (assign (V, X)): StackCode (X) POP (V)	'Rule 3' StackCode(minus(X,Y)): StackCode(X) StackCode(Y) MINUS
'Rule 2' StackCode(plus(X,Y)): StackCode(X) StackCode(Y) PLUS	'Rule 4' StackCode(var(V)): PUSH (V)

Figure 2.21: An Example of Expression Rule in a Stack

For example, assume a statement *assign ("Result", plus ("A", minus ("B", "C")))* in the left side of Figure 3.7 (here, A, B and C are variables). When using the rules of Figure 3.6, the generated codes by the stack process will look like the middle of Figure 3.7 and after executed, results in the following stack configurations will be the right side of Figure 3.7. Now, the new code is generated. Of course, this is just a simple example. In real situation, it will be much more complex.

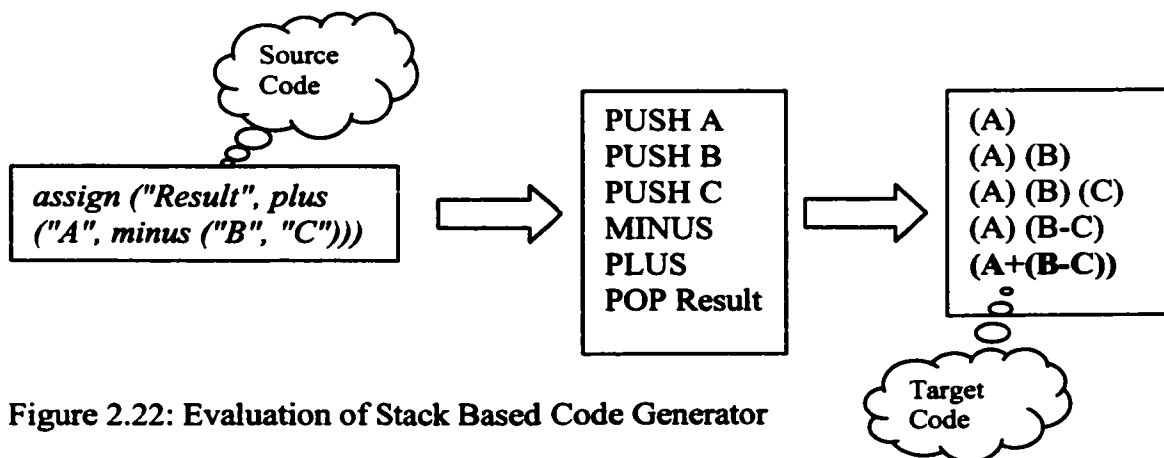


Figure 2.22: Evaluation of Stack Based Code Generator

Discussion: Stack based code generator has a long history [JT77]. It first appeared with compiler, Today, Most code generators in compiler still use this kind of approach to generate target code from higher level language's program to lower level language program. The advantage of stack based code generator is very flexible. It also has many well-known algorithms for this kind of code generator. But it is hard to output different language program (less extendable). Any new output language has to be redesigned and redeveloped for whole system. Because of this limitation, stack based code generator is very good candidate for compiler (one kind of input language program gets one kind of output programs) but not good for DWDI code generator as discussed in Section 1.4.

3. WODD Code Generation Technique

As discussed in Chapter 1, DWDI code generator is very important for DWS development. This chapter will briefly review data integration approaches, code generation approaches and then detail discuss this approach chosen for Data Warehouse, Object-oriented Database and Data mining (WODD) code generator project with a brief example.

There are two steps in WODD code generator project. In the first step, WODD project uses SDB schema description and DW definition to generate the DWDI programs. Most integration in this step is for schema integration. In the second step, using the programs that were generated from code generator extract, clean and load data from SDB to DW. This step is mostly involved in data integration.

3.1 Schema/Data Integration Techniques

As discussed in Section 2.1, schema integration is important for DWDI code generation. However, the schema integration from [BLN86] using the merging and restructure to create a unique database schema, which will not satisfy user interesting DW structure. This approach is also around the data source's schemas to create a new schema, not around the major subject of an enterprise to create a DWS for business decision making. Because the limitations, some concept will be used in WODD code generator project but some ideas will be retrieved from other data integration approaches.

There are three major approaches for data integration as discussed in Chapter 2, structured data integration approach, semantic data integration approach and virtual data integration approach.

Structured data integration approach is based on a structured model and a set of query written by a query language to integration data. This approach is flexible and good for already known business data integration, but it is difficult to extend to all business in the world. This limitation make this approach is not a good candidate for DWDI code

generator.

Semantic data integration approach is based on a knowledge model to maintain the relations of data among data sources. Because these relations in knowledge model link each pair of data that have similar meanings, it is very good candidate for semi-structured data integration. This advantage can make DWDI code generation more extendable to integrate non-database data source in future. But the limitation discussed in Section 2.2.2 for DWDI makes this approach is not a good candidate for DWDI code generator. But the idea about relation pair should include in WODD project.

Virtual approach is a vertical data integration approach as discussed in Section 2.2.3. This approach presents an idea about finding each pair of relations between each data source and target DW. Because it is very extendable and flexible for DWDI code generator, if using it for schema to find same meaning (same as schema integration see Section 2.1) and create the relation pair (same as semantic data integration), it will be a good candidate for the DWDI code generator.

3.1.1 Definitions

In order to integrate the sources to data warehouse by virtual approach, some term have to be defined first [DGe97]].

- (1) An *atom expression* is an expression of the form $p(\tau_1, \dots, \tau_n)$.
- (2) An *expression* is ether an atom expression, an expression of $\phi \vee \psi$ or $\phi \wedge \psi$.
- (3) A *definition* is an expression of the form $\phi \equiv \psi$, where ϕ is an atom expression and ψ is an expression. ϕ is called the head of the definition, and ψ is called the body. In the virtual data integration approach, definition are required to be safe, e.g., all variables appearing in the head must appear in the body. Variables appeared in the body (but not in the head) are assumed to be quantified existentially. The bodies of definition are also required to be uniform with respect to the variables in the head. The disjunction in every disjunction must contain the same head variables. For example:

$$\text{grandparent}(X, Z) \equiv \text{parent}(X, Y) \wedge \text{parent}(Y, Z).$$

Based on these basic definitions above, some relation definitions that are used in the virtual data integration approach are introduced. In virtual data integration approach, the user interface and the available information sources are modeled by a set of relations, which is a tool that will be used for finding the one to one relation (also called rule) for DWDI [DGe97].

- (1) *Interface relations* are the forms of users defined target DW table. Interface relations conceptualize the interaction between target DW table and SDB.
- (2) *Site relations* are the data available from a data source table. Site relations represent the data that are actually stored in the available data sources.
- (3) *Base relations* are for expressing both interface relations and site relations. Base relations are used as means to describe both interface and site relations and are crucial in order to simplify adding new information sources and accommodating the changes in content of existing ones.

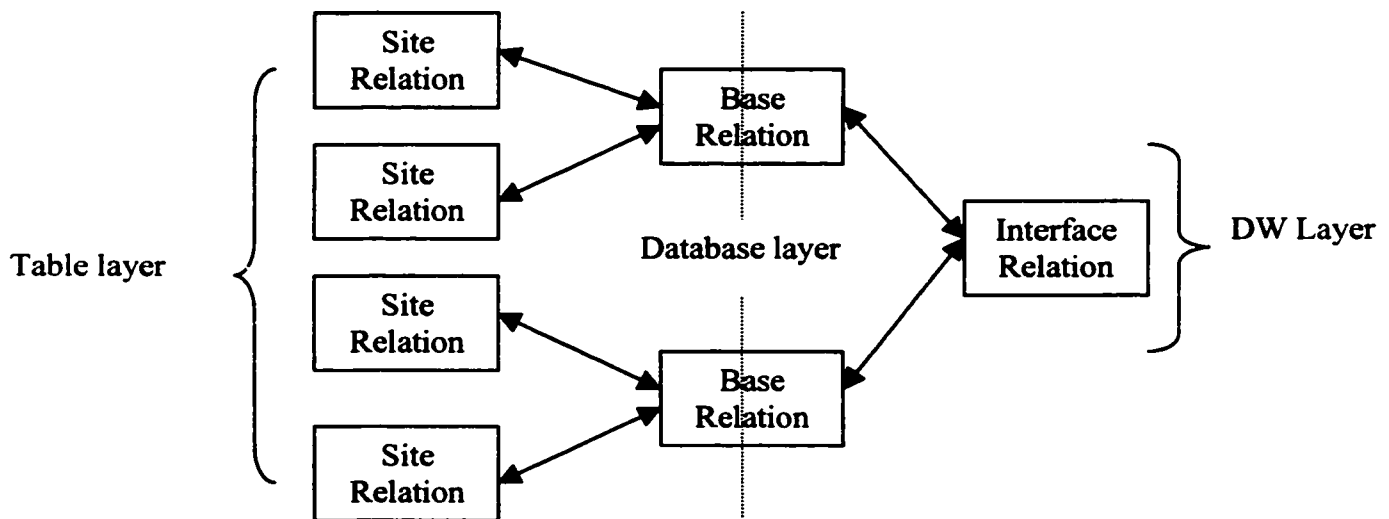


Figure 3.1: Interface Base, Site Relation Diagrams

In Figure 3.1, site relation describes the relationship between tables and database. Base relation describes the relationship between SDB and DW. This architecture is flexible when a DW corresponds to unknown number of source databases in base relation

(flexible to increase source or delete sources) and source tables are different among these sources (because it isolates the source and its table from other sources).

Based on the relation characters to create all of the *definitions*, we assume in the *base relation*, the *head of definition* (left side) is target DW and the *body of the definition* (right side) is the source database. In the *site relation*, the *head of definition* is *base relation* and *body of the definition* is the source table of that database.

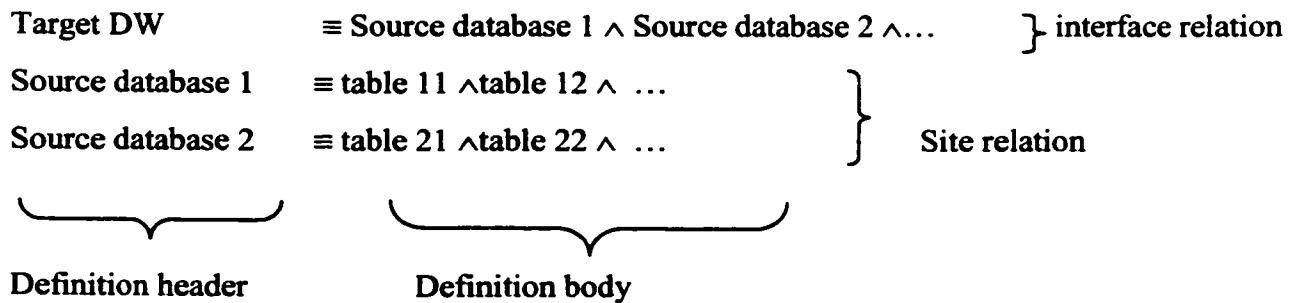


Figure 3.2 Relation Definition

In Figure 3.2, the base relation is used to simplify adding new information sources and accommodating the changes in content of existing ones. Both interface relations and site relations are represented in terms of base relations. Base relation is for flexibility and the base relation will not be present in the final integration rule and limitation because the data warehouse data integration only takes care of DW and SDB tables.

3.1.2 Rule Transformation Algorithm

Written in the way of the *definitions* (head + body) in basic definition and the *relations* can have several characters [DGe97].

(1) The word relations can be defined in terms of the other relations, e.g., the grandparent relation ‘grandparent’ in terms of the parent relation ‘parent’ as shown below:

$$\text{grandparent}(X, Z) \equiv \text{parent}(X, Y) \wedge \text{parent}(Y, Z)$$

(2) Describe the contents of databases in the same way that *views* (definition 9) are defined. For example, assume that there is a database *DI* that stores those tuples of the unary relation *r* satisfying the unary relation *s*, we can capture this information by defining the site relation *r1* and writing the following definition for *r1*

$$r1(X) = r(X) \wedge s(X)$$

(3) Partial information can be expressed by inventing new base relations and using these new relations in the definitions of other relations, e.g., we can express the fact that *r2* is contained in *r* by inventing a gensym predicate *g* and writing the following definition

$$r2(X) = r(X) \wedge g(X)$$

(4) By writing integrity constraints, we can describe what tuples are not present in a relation, e.g., we can express the disjointness of relations *p* and *q* as follows

$$\neg(p(X) \wedge q(X))$$

The expressiveness of infomaster's description language allows to define a wide range of rules. One advantage of this approach is that different horizontal fragments can be described independently of other horizontal fragment, since each fragment can be described in separate rules. For example, in a banking system, one source using full name and another source table using first name and last name, suppose in the target table, the attribute represented by name. Two independent rules can be created. $Name \leq fullname$ and $Name \leq first\ name \parallel last\ name$. This algorithm, in fact, offers one attribute to one attribute integration relation, which is the key in the virtual data integration approach.

Query processing consists of query planning and query execution. The *query planning algorithm* takes a query *Q*, a collection Δ of rules and integrity constraints (like definitions) as inputs. The output of the algorithm is a query processing plan suitable for input to the plan execution algorithm. The *plan execution algorithm* takes a query plan as input. It retrieves data from the available databases and merges this as described in the plan. The output is a table of answers to the original query. Query processing in the Infomaster system is a three step process. Assume the user asks a query *q*. This query is expressed in terms of interface relations. In a first step, query *q* is rewritten into a query in terms of base relations. This step is called reduction. In the second step, the

descriptions of the site relations have to be used to translate the rewritten query into a query in terms of site relations. This second step is called abduction. The query in terms of site relations is an executable query plan, because it only refers to data that is actually available from the information sources. However, the generated query plan might be inefficient. Using the descriptions of the site relations, the query plan can be optimized.

We also can give a more detailed algorithm for query planning. There are five steps in the query planning algorithm.

(1) *Reduction*: In this step, the system rewrites each atom in the query using the definition of the associated predicate. It then repeats the process until it obtains an expression in terms of base relations (which, by definition, have no definitions of their own.). Termination is assured due to the non-recursive nature of the definitions.

(2) *Abduction*: Given an expression φ in terms of base relations and a set of definitions, the abduction process produces the set R of all consistent and minimal conjunctions of retrievable atoms that can be shown from the definitions to be contained in φ .

(3) *Conjunctive Minimization*: In this step, eliminating any redundant conjunct, i.e. one that can be shown to contain the remaining conjuncts.

(4) *Disjunctive Minimization*: In this step, dropping any disjunction that can be shown contained in the union of the remaining disjunction.

(5) *Grouping*. Finally, the conjuncts within each conjunction are grouped so that the atoms in each group all share a common provider.

For simplicity, these steps were considered as taking place sequentially. In the implementation, some of the steps are interleaved. For example, conjunctive minimization is interleaved with abduction. This saves time by cutting further work on a conjunction once an inconsistency has been detected.

3.1.3 Example

Assume an example about used car. The newspaper Toronto Star information table and the Windsor Star information table both are contained in a used car classified database. Moreover, assume that both NISSAN and HONDA dealer provide data on the average

market value of their cars for a given model, year, and mileage in Dealer database. This example will describe interface relation, site relation and base relation used to link interface relation and site relation.

Site relation TorontoStar and WindsorStar model the information got from the used car classified in Toronto Star table and Windsor Star table. The relation shown as

$\text{Classifies} \equiv \text{TorontoStar} \vee \text{WindsorStar}$

Or

$\text{Classifies} (\text{Manufacturer}; \text{Model}; \text{Year}; \text{Mileage}; \text{price}) \equiv$
 $\text{TorontoStar} (\text{Manufacturer}; \text{Model}; \text{Year}; \text{Mileage}; \text{price}) \vee$
 $\text{WindsorStar} (\text{Manufacturer}; \text{Model}; \text{Year}; \text{Mileage}; \text{price})$

Site relation NISSAN table and HONDA table represent the information available from Dealer database. The example contains two further relation that provides the current exchange rate from US\$ into Canadian \$ and information about mile to kilometer. This relation can be represented as (note: the rate is an external data)

$\text{Dealer} \equiv \text{NISSAN} \vee \text{HONDA}$

Or

$\text{Dealer} (\text{Manufacturer}; \text{Model}; \text{Year}; \text{Mileage}; \text{price}) \equiv$
 $\text{NISSAN} (\text{Model}; \text{Year}; \text{mile}; \text{Value in US\$}) \vee$
 $\text{HONDA} (\text{Model}; \text{Year}; \text{km}; \text{Value in Canadian \$}) \wedge$

$\text{mile} = \text{km} / 1.6$

$\text{Value US \$} = \text{Value in Canadian \$} * \text{Rate}$

Or

$\text{Dealer} (\text{Manufacturer}; \text{Model}; \text{Year}; \text{Mileage}; \text{price}) \equiv$
 $\text{NISSAN} (\text{Model}; \text{Year}; \text{km} / 1.6; \text{Value in US\$}) \vee$
 $\text{HONDA} (\text{Model}; \text{Year}; \text{km}; \text{Value in Canadian \$} * \text{Rate})$

For base relation, The base relation Classified represents classified ads in TorontoStar and WindsorStar, which is same as site relation because they have same attribute number. But for the base relation Price represents the market value information on car models, which add one more constant value into NISSAN and HONDA tables.

Price ("NISSAN" Model; Year; km / 1.6; Value in US\$) \equiv
 NISSAN (Model; Year; km / 1.6; Value in US\$)

And

Price ("HONDA"; Model; Year; km; Value in Canadian \$ * Rate) \equiv
 HONDA (Model; Year; km; Value in Canadian \$ * Rate)

So the interface relation "Car" represents the target DW table. In this example, there are basically two kinds of data sources: Classified database and general information on car models from Dealer database.

Car \equiv Classified \vee dealer

Or

Cars (Manufacturer; Model; Year; Mileage; price; Value) \equiv
 Classifieds (Manufacturer; Model; Year; Mileage; price) \vee
 Dealer (Manufacturer; Model; Year; Mileage; Value)

In this interface relation, the schema (Manufacturer; Model; Year; Mileage; price; Value) is real target but only a picture for both databases.

Finally, Add relation from interface to site relation, the relation from data source to target will be looked like

Cars (Manufacturer; Model; Year; Mileage; price; Value) \equiv
 TorontoStar (Manufacturer; Model; Year; Mileage; price) \vee
 WindsorStar (Manufacturer; Model; Year; Mileage; price) \vee
 NISSAN (Model; Year; km / 1.6; Value in US\$) \vee
 HONDA (Model; Year; km; Value in Canadian \$ * Rate)

Or

Cars (Manufacturer; Model; Year; Mileage; price; Value) \equiv
TorontoStar (Manufacturer; Model; Year; Mileage; price) \vee
WindsorStar (Manufacturer; Model; Year; Mileage; price) \vee
Price ("NISSAN"; Model; Year; km / 1.6; Value in US\$) \vee
Price ("HONDA"; Model; Year; km; Value in Canadian \$ * Rate)

This finally relation says that in order to get the Car, we can extract Toronto Star first, Windsor Star second, and then Price for NISSAN and finally extract the price for HONDA

In the last, consider the relation vertically, using the pair in semantic data integration approach, a set of integration rule for the DW schema integration can be show as below (Note, car.model means the model from car table. Only demo the Cars.Manufacturer and Cars.Mileage)

Cars.Manufacturer \blacktriangleleft TorontoStar.Manufacturer
Cars.Manufacturer \blacktriangleleft WindsorStar.Manufacturer
Cars.Manufacturer \blacktriangleleft "NISSAN"
Cars.Manufacturer \blacktriangleleft "HONDA"
...
...
...
Cars.Mileage \blacktriangleleft TorontoStar.Mileage
Cars.Mileage \blacktriangleleft WindsorStar.Mileage
Cars.Mileage \blacktriangleleft NISSAN.km / 1.6
Cars.Mileage \blacktriangleleft HONDA.km

Using this set of pairs (also called *integration rules*), put them into code generator to generate the code.

3.2 Code Generator Technique

As discussed in Section 2.3, there are many code generation approaches, and each of them has advantage or limitation.

Stack based code generator approach is very flexible. But it is less extendable to other output programming language. Also, stack based code generator asks programs as input, which is different from DWDI code generator defined in Section 1.4.2. Because this limitation, stack based code generator is not good for WODD code generator project.

Rule base code generator is an extendable approach, which can generate many kinds of output language. This advantage make this approach can be a good candidate for DWDI code generator. This approach's limitation is less flexible.

Template based code generation approach is flexible. Template is similar to blank paper that many components can be written into this paper and become an "article". In Template based code generation approach, a text file is a template and several small segment of program as components. Template code is good for object orient program generation because each component is a class definition. But for WODD code generator project, same piece of program can have different number/type variable, e.g., in a banking DWDI system, for customer table, the data extracting program need variable such as costomerID and in branch table, it need such as branchID. one component must create more than one programs. Using template base code generator can not solve this problem.

If combining the template based code generation approach and rule base code generation together, it will fill the gap discussed in Section 1.4. Assuming the segment of program generated by rule based code generator is a component of template, the code generator will be extendable (from rule based generation approach) and flexible (from template code generation approach). For example, after putting the components, DBconnection, keyUnify and removeRedundancy components into template called cleaning.pc (Pro*C template) and set the different variable such as customerID, name etc

(name them propriety temporary in this thesis) into \$ sign position of components as Figure 3.3, the data cleaning output program will be ready.

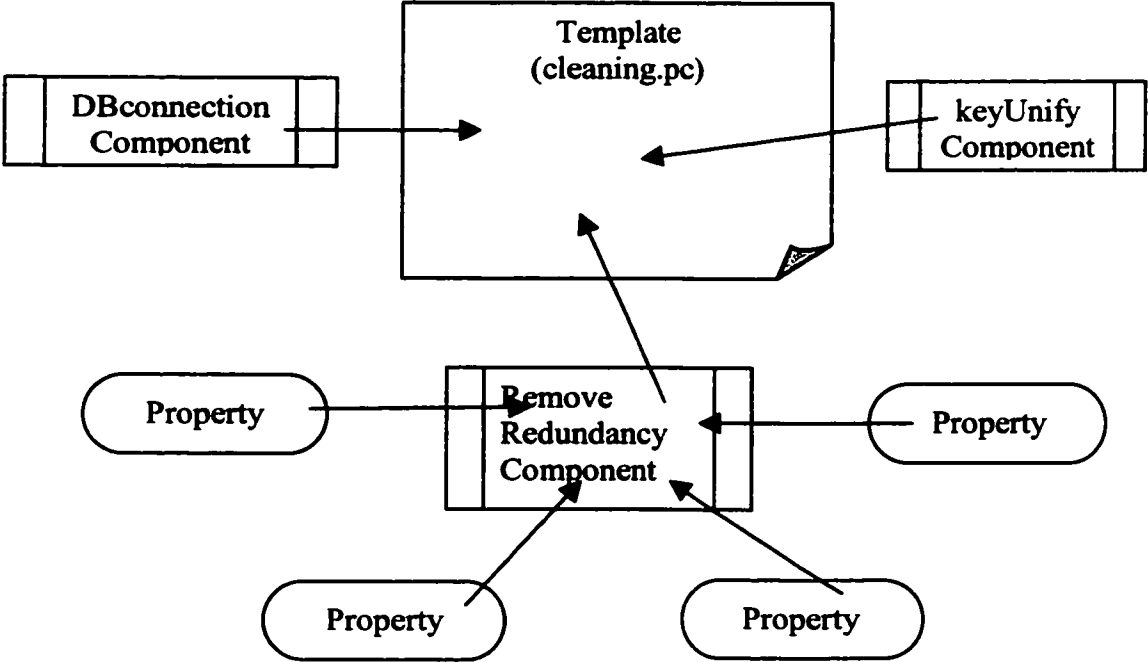


Figure 3.3: Template Combines with Rule Based Code Generation Approach

3.2.1 Example

If thinking the code generator is a black box, the input of this box will be (1) target DW description (2) transformation rule. The output of this box is a set of computer programs. Assume we try to extract some data from WindsorStar to DW car table in Pro*C

computing language. The set of integration rules has already been ready as shown in Section 3.1.2

```
Cars.Manufacturer ◀ TorontoStar.Manufacturer
Cars.Manufacturer ◀ WindsorStar.Manufacturer
    Cars.Manufacturer ◀ "NISSAN"
    Cars.Manufacturer ◀ "HONDA"
    ...
Cars.Mileage ◀ TorontoStar.Mileage
Cars.Mileage ◀ WindsorStar.Mileage
    Cars.Mileage ◀ NISSAN.km / 1.6
    Cars.Mileage ◀ HONDA.km
```

First, an empty extracting template called 'extracting.pc' is created as shown in Figure 3.4. There are at least two components needed for this template: a main component that connects to an Oracle databases, and several extracting function components that consists of the frame of Pro*C programs without property definition.

```
/* Extracting.pc */
```

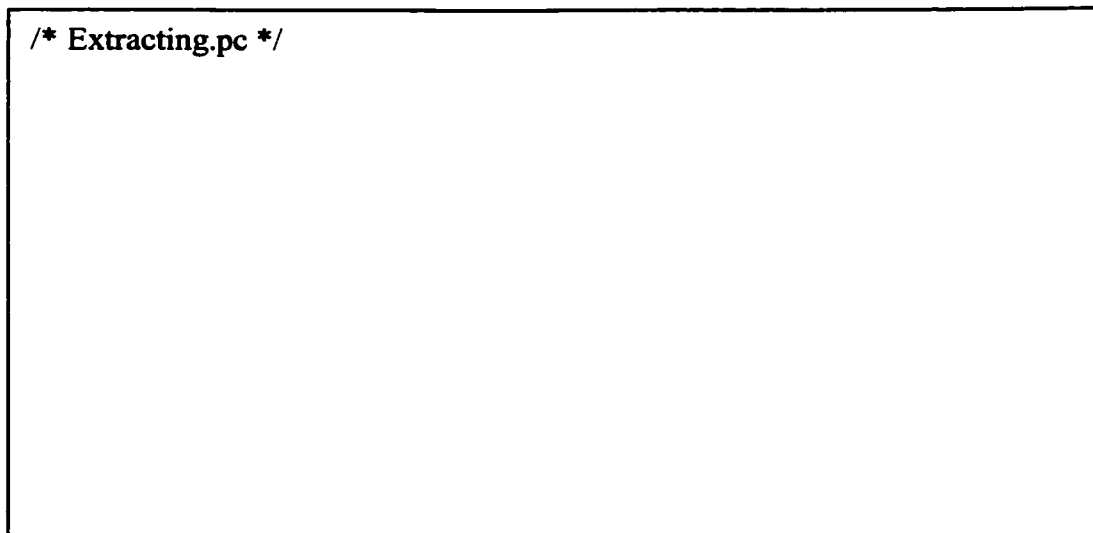


Figure 3.4: Empty Template

Next, insert main component into the template 'extracting.pc' to connect an Oracle database with user id "scott" and password "tiger". The dollar sign in these components are properties to issue what kind of variable should put here later. See figure 3.5

```

/* Extracting.pc */

/*****
/*
/*          Main Component          */
/*
/*          *****/
/*****

#include <stdio.h>
#include <string.h>

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
char *username="scott\n";
char *password="tiger\n";
EXEC SQL END DECLARE SECTION;

void $ff ();

void main()
{

EXEC SQL CONNECT :username identified by :password;

if (sqlca.sqlcode < 0)
{
printf("\n%s",sqlca.sqlerrm.sqlerrmc);
EXEC SQL ROLLBACK WORK RELEASE;
exit(1);
}
else
printf("Login Oracle for DW table creation successfully\n");

$ff ();

EXEC SQL COMMIT WORK RELEASE;
exit(0);
}

```

Figure 3.5: Component in Template before Assigning Variable in \$ Sign

There are two SDB (four tables) that will be integrated. Because of virtual integration approach, the relation can be one to one (One-source-table to target-table independently). So there are four extracting functions here. One function will extract the data from SDB WindsorStar table to DW car table and other three functions will extract data from SDB TorontoStar table, NISSAN table and HONDA table to DW car table. For each function, one source attribute corresponds to one target attribute, which is exactly same as integration rules. First, add one component into template as shown Figure 3.6

```

/* Extracting.pc */

/*****
/*
/*          Main Component          */
/*
/*****

#include <stdio.h>
#include <string.h>

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
char *username="scott\n";
char *password="tiger\n";
EXEC SQL END DECLARE SECTION;

void $ff ();

void main()
{

EXEC SQL CONNECT :username identified by :password;

if (sqlca.sqlcode < 0)
{
printf("\n%s",sqlca.sqlerrm.sqlerrmc);
EXEC SQL ROLLBACK WORK RELEASE;
exit(1);
}
else
printf("Login Oracle for DW table creation successfully\n");

$ff ();

EXEC SQL COMMIT WORK RELEASE;
exit(0);
}

/*****
      One extracting component
*****/

void $f ()
{
EXEC SQL BEGIN DECLARE SECTION;
  $dd $vv ;
EXEC SQL END DECLARE SECTION;

EXEC SQL DECLARE $C CURSOR FOR SELECT
  $vv ,
  $v
FROM $t ;

EXEC SQL OPEN $C ;
EXEC SQL WHENEVER NOT FOUND DO break;          (Continue)

```

```

while(sqlca.sqlcode==0)
{
  EXEC SQL FETCH $C INTO
    : $vv ,
    : $v ;

  EXEC SQL INSERT INTO $ODSt
  (
    $oo ,
    $o
  )
  VALUES
  (
    : $cc ,
    : $c
  );
}
EXEC SQL CLOSE $C ;
}

```

Figure 3.6: Template with Two Components

There are many dollar signs in the template now. Each dollar sign represents a kind of variable property, which defines what kind of variable property should be put here.

\$ff: repeatable function name

\$f: non repeatable function name

\$dd: repeatable data type name

\$vv: repeatable variable name

\$v: non repeatable variable name

\$t: non repeatable table name

\$C: non repeatable cursor name

\$cc: repeatable value name

\$c: non repeatable value name

(Note: repeatable means when adding variable in the \$ position, keep the \$ sign for next insertion. non repeatable is when adding variable at the \$ sign position, not keep the \$ sign)

Set each property about WindsorStar into the property position and the \$ff named extractingWindsorStar. The template will become Figure 3.7

```

/* Extracting.pc */

/*****
/*
/*          Main Component
/*
/*
/*****

#include <stdio.h>
#include <string.h>

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
char *username="scott\n";
char *password="tiger\n";
EXEC SQL END DECLARE SECTION;

void extractingWindsorStar ();
void $ff ();

void main()
{

EXEC SQL CONNECT :username identified by :password;

if (sqlca.sqlcode < 0)
{
printf("\n%s", sqlca.sqlerrm.sqlerrmc);
EXEC SQL ROLLBACK WORK RELEASE;
exit(1);
}
else
printf("Login Oracle for DW table creation successfully\n");

extractingWindsorStar ();
$ff ();

EXEC SQL COMMIT WORK RELEASE;
exit(0);
}

/*****
Generated from extracting template
*****/

void extractingWindsorStar ()
{
EXEC SQL BEGIN DECLARE SECTION;
Int model;
Int mileage;
Float price;
EXEC SQL END DECLARE SECTION;

```

```

EXEC SQL DECLARE WindsorStarCursor CURSOR FOR SELECT
    Model,
    Mileage,
    price
FROM WindsorStar ;

EXEC SQL OPEN WindsorStarCursor;
EXEC SQL WHENEVER NOT FOUND DO break;

while(sqlca.sqlcode==0)
{
    EXEC SQL FETCH %C INTO
        Model,
        Mileage,
        price

    EXEC SQL INSERT INTO car
    (
        model,
        mileage,
        price
    )
    VALUES
    (
        : model,
        : mileage,
        : price
    );
}
EXEC SQL CLOSE WindsorStarCursor;
}

```

Figure 3.7: Output Program for Extracting WindsorStar Information

Last, insert other components for TorontoStar table, NISSAN table and HONDA table. Set property for these components. Thus, the whole extracting program is generated.

4. Implementation

WODD code generator is a system that creates a group of output programs such as Pro*C programs, which can be used for creating DW and ODS tables, extracting different format data from different SDB into ODS, cleaning the dirty data in ODS, e.g., removing redundant data, loading data from ODS into DW tables if the data is not existed the DW.

In this chapter, we will discuss the WODD project aims and system environment, present the WODD project architecture and finally, demonstrate an example. For more detail operation, please see *user manual* in Appendix A.

4.1 Project aims and Restrictions

The aim of project is to implement a code generation system for DWDI based on *rule based data integration approach* and *template based code generation approach*. We suggest that this project follow these restrictions below:

Location: assume this code generator is to solve a problem when the source and target are in the same location or machine.

Source: assume this code generator is to solve a problem when the sources include only relational database (structured data sources).

Generation: assume this code generator is to solve a problem based on general architecture of data warehouse (reference Figure 1.1).

Production: assume this code generator only generates one language.

Extension: assume this code generator is to solve a problem that can be extended into distributed or web system in the future and multi generated language output. Extensions to accommodate other different data source like HTML, XML and textual data in the future should be possible too.

4.2 System Architecture

As shown in Figure 4.1, The implementation of *code generator for integrating data warehouse* focuses on four different major areas: (1) data warehouse (target) creation, (2) data integration, (3) code generation and (4) data procession.

Data warehouse table generator is a segment for data warehouse creation, which can generate a set of programs with certain output language such as Pro*C. it can be used for creating data warehouse tables and operational data store tables. This segment accepts user-input information about target table name, attribute and data types, checks the information's validation and finally, generates the data warehouse table creation programs.

Source table Generator is a segment for extracting source database definition such as database name (also user ID and password), table name, attribute and data type. Source database description can be input by manual, from source description file or JDBC extracting automatically.

Data integration rule generator is a segment for assigning data integration mapping rule for the future data integration purpose. This generator accepts operational data store table definitions from data warehouse table generator, source table definition from source generator and the mapping relationship between source database and operational data store. This segment will generate a set of mapping rules, which are similar to the rule based data integration approach to create (1 to 1) integration rule. For example, in banking system, the attribute *name* in the target *customer* table comes from the combination of *last name* and *first name* in source *checkingCustomer* table. The rules can be represented as

Customer..name ← checkingCustomer.lastName || :checkingCustomer.firstName

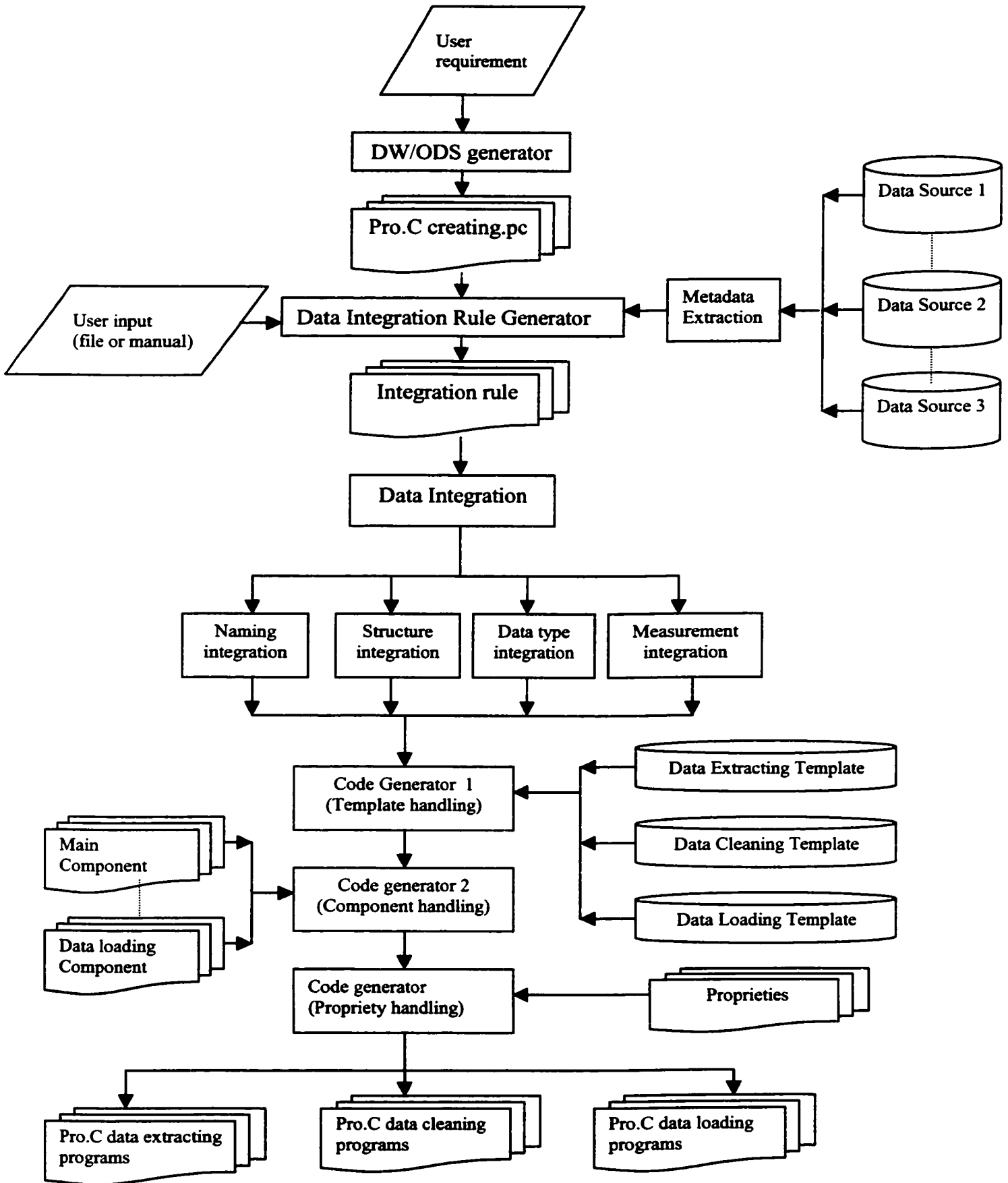


Figure 4.1: System Architecture

Which can use for integrating different data structure between source and target attribute. Note, based on the *rule based data integration approach*, the customer name will be extracted from both *checkingCustomer* and *savingCustomer* table. In target customer table, same customer will appear twice

Another example for measurement integration is a rule that issues the *balance* represented by US dollar in source *saving* table and *bal* by Canadian dollar in target *fact* table. The rule can be:

$$\text{fact.bal} \leftarrow \text{saving.balance} * 0,76$$

After collecting the whole set of data integration rules, the integration rule generator transfers the whole set of integration rules into each attribute to integration data sources.

In the implementation, there are two types of integration. One type of integration is only related on database/table definition and their metadata, which is called *schema integration*. For example, integrating different data structure, integrating different data type, integrating different naming conversion and integrating different measurement are only based on the schema information. Another type of integration e.g., integrating different encoding, remove redundancy data would work for database data, which is called data integration.

In WODD implementation, *data integration segment* accepts the data integration rule from data integration rule generator and integrates these attributes based on this set of data integration rule. Any schema integration will be done here.

Code generator is a segment for generating the target DWDI programs. It accepts *variable propriety* information from data integrating segment to create the target code. The *Code generator* uses *template based code generation approach* combining *rule based code generation approach*. In this segment, a template that relates on target language will be created first, such as creation template (called *creating.pc*), data

extraction template (called *extracting.pc*), data cleaning template (*cleaning.pc*) and data loading template (*loading.pc*). A set of components consisting of language rules will be assigned into the templates. For example, put database connector into the template for linking the database. Finally, assign *variable propriety* into code generation rule (also called components). The output of code generator is a group of programs that can be used for handling data integration.

Target programs is a set of programs generated from code generator for creating data warehouse tables, extracting data from source database with different format to ODS, cleaning redundancy data and dirty data in ODS and loading data from ODS into DW.

4.3 User Interfaces and Background Processes

This implementation as shown in Figure 4.2 consists of three types of processes. *Front processes* with *graphic user interface* (GUI) to accept information from outside, to generate new information and display them back to the user. *Background processes* without GUI accept information from front processes, generate new information and output the information for future procession. *Target processes* are for integrating warehouse data. In this implementation, WODD can be used for multi-user system and personal computer. In multi-user system, front processes design by JBuilder 3.5 under Sun Solaria. Background processes were designed and coded on Java under UNIX. Pro*C were compiled on SGI. In personal computer, they work with Java 2.1 under Window98 and Oracle 8i.

There are six major GUIs in WODD project: *Start WODD GUI* is an interface to express the system's beginning. *Data warehouse table generator GUI* is an interface for accepting user interesting DW definition. *Source database generator GUI* is an interface that accepts the source database definition by user input, file or JDBC extracting. *Data Integration rule generator GUI* is an interface that accepts user assignment of the relations among attributes of different source database and tables. This GUI is designed

using dynamic coding technique. It accepts the target/source table definitions based on the information from DW table generator and the source database generator, checks the integrating validation in order to guarantee the rule correction. Finally, it outputs the integration rules. *Execution/Display GUI* is a GUI for displaying the result programs based on the dynamic programming technique and compiling, running the target data integration programs automatically. Any target program under certain language can be displayed from this interface. *End WODD GUI* is a GUI to tell user the target programs have already been done. There are more interactive windows for information input and output such as template name input and help information output. For more detail, please check Appendix A - user manual.

There are two major background segments in WODD, data integration is a segment that takes the rule as input, and generated the variable propriety as output. Code generator is another background segment that generates the target code based on the template and rule based code generation approach. The components that will be added into the template and the properties will be modified each component to define the exact function of the components.

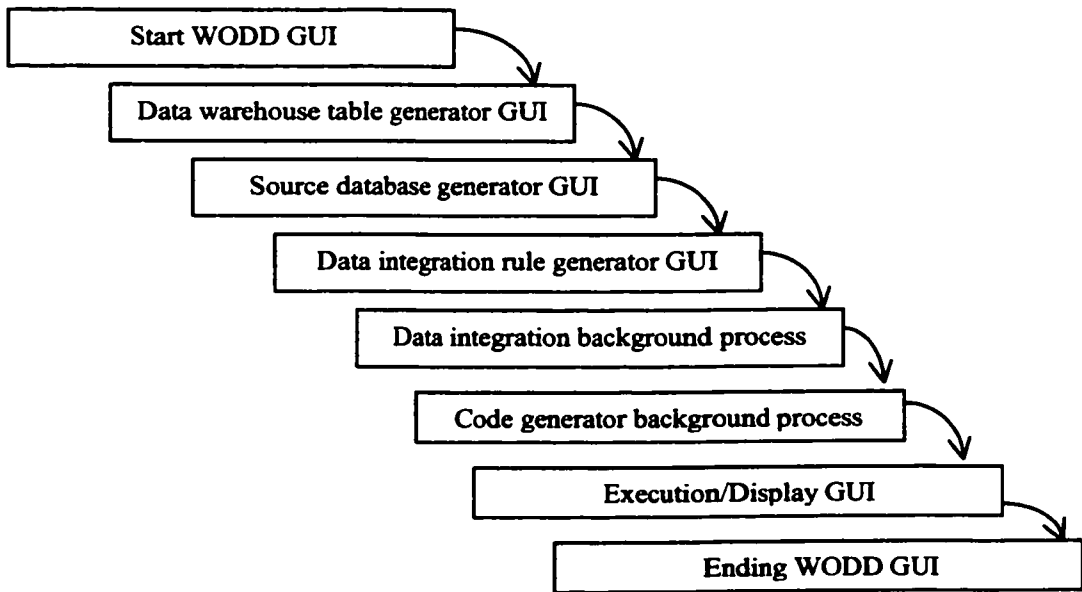


Figure 4.2 System Components

4.4 WODD Programs

WODD system consists of 19 Java programs, 12 Components, 5 graphics and 3 banking testing text file. After compiling and running WODD, WODD will generate 96 class files and 4 target (Pro*C or other language) files. For detail program codes, please reference to Appendix B-*WODD codes*.

WODD has 19 Java programs. Each program has its purpose. As the list in Figure 4.3, the WODD starts from WIG.java and ends in endWIG.java for ending of WODD (from top to down).

Program Name	Program Size	Program Purpose
<i>Wig</i>		Batch file for going through WODD
<i>WIG.java</i>	101 lines	Start GUI
TableGenerator.java	39 lines	Input Target table definition
Frame1.java	536 lines	Target table generation GUI
SourceGenerator.java	48 lines	Input source DB definition
Frame4.java	443 lines	Source table GUI
LinkingGenerator.java	204 lines	Using to link different SDB
Frame6.java	752 lines	Linking DB information GUI
SDBconnector.java	106 lines	Using for auto extracting SDB definition
RuleGenerator.java	57 lines	Start to generate mapping rule
Frame2.java	2210 lines	Rule mapping GUI
DataIntegration.java	762 lines	Integration schema based on cases
MainGenerator.java	1154 lines	Generate the fact for data warehouse
CodeGenerator.java	1409 lines	Generate other programs for DW
Execution.java	102 lines	Compiler and executing target programs
Frame5.java	240 lines	Running GUI for the target
Display.java	56 lines	Display the target programs
Frame3.java	191 lines	The GUI for display
EndWIG.java	188 lines	Finish WODD system

Figure 4.3 Java Class Definitions

WODD has 12 different components, which can be defined by programmer when new target data warehouse data integration programs are added. Each component has particular name and purpose, which is very important to create a useful target programs.

Component Name	Component Purpose	Belonging
CreatingComponent	Creating DW tables	Creation program
InitMetadataComponent	Initialize metadata contents	Creation program
MetadataComponent	Creating metadata table	Creating program
TimeComponent	Creating time dimension table	Creating program
MainComponent	Creating program header	All programs
ExtractingComponent	Creating extracting program	Extracting program
LinkingComponent	Linking different source databases	Extracting program
FactComponent	Managing fact table of data warehouse	Extracting program
CleaningComponent	Using for generating cleaning program	Cleaning program
KeyCleaningComponent	Cleaning PF attributes	Cleaning program
NonKeyCleaningComponent	Cleaning non key attributes	Cleaning program
LoadingComponent	Generating loading program	Loading program

Figure 4.4 Basic composition components

```

#include <stdio.h>
#include <string.h>

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
char *username=" $UID \t\n";
char *password=" $PWD \t\n";
EXEC SQL END DECLARE SECTION;

void $ff ();

void main()
{
EXEC SQL CONNECT :username identified by :password;

if (sqlca.sqlcode < 0)
{
printf("\n%s",sqlca.sqlerrm.sqlerrmc);
EXEC SQL ROLLBACK WORK RELEASE;
exit(1);
}
else
printf("Login Oracle successfully\n");

$ff ();

EXEC SQL COMMIT WORK RELEASE;
exit(0);
}

```

Figure 4.5 Example of mainComponent for Pro*C

For example, the mainComponent is for creating the program header. In most high level programming language such as Pro*C, C++ and JAVA need header to define the prototype, JDBC segment and so on. But some script languages such as SQL, PL/SQL don't need header section. Based on different target language, the mainComponent can be prototype, database linking for high level language or just empty for lower level language. For example in Pro*C, the mainComponent can as Figure 4.5.

In WODD, we tested this system by creating a small data warehouse system called banking system (see section 4.5 - An example) and generated Pro*C target programs and PL/SQL programs. For more target programs code, please see section 4-generated target programs in Appendix B. Banking system database structure has been introduced in section 4.5. There are 3 different files for the testing even if all the parameters described in the file can be inputted by manual. These files include

Banking test file	Purpose
Target.txt	Data warehouse definition
Source.txt	Source database definition
Rule.txt	Mapping rule definition

Figure 4.6: Banking Testing Files

There are 5 graphic file for WODD GUI, which can make the system more colorful. These files are not very necessary for WODD. Without theses file, WODD can run under any platform too.

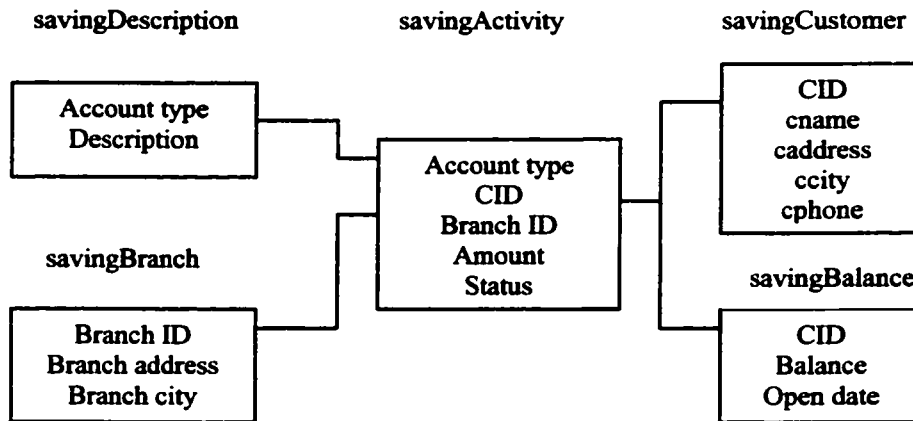
Graphic file
Plant.gif
Boy.gif
Buttefly.gif
Windsor.gif
Angry.gif

Figure 4.7: Graphic files

4.5 Example

Assume that a *banking system* consists of two databases--*saving account database* and *checking account database* as shown in Figure 4.8.

Saving account database, tables and data example:



CID(int)	CNAME(char[20])	CADDRESS(char[20])	CCITY(char[10])	CPHONE(char[10])
999030031	Liu Yi	260 Randolph Place	Windsor	256-8338
100000000	Liu Hua	33 Isabella Street	Toronto	666-8888
100324235	Du Ying	438 Niagara Street	Windsor	771-9867

savingCustomer table

CID(int)	BALANCE(float)	OPEN-DATE (char[10])
999030031	1000	01-01-1999
100000000	5000	01-01-2000
100324235	2000	10-10-1999

savingBalance table

BRANCH-ID(int)	BRANCH-ADDRESS(char[20])	BRANCH-CITY(char[10])
666	4 Bloor Street	Toronto
888	42 Ouellete Ave	Windsor

savingBranch table

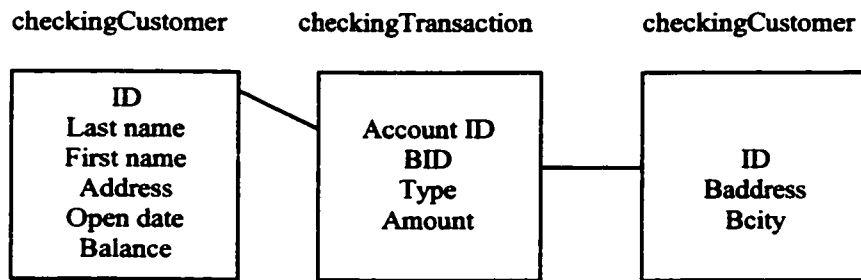
TYPE(char[3])	DESCRIPTION(char[10])
DP	Deposit
TR	Transfer
WD	Withdraw

savingDescription table

TYPE(char[3])	CID (int)	BRANCH-ID(int)	AMOUNT(float)	STATUS(char[1])
DP	100324235	666	2000	Y
TR	100000000	888	3000	Y
WD	999030031	888	1000	Y

savingActivity table

Checking account database, its tables and data examples



ID(char[8])	LASTNAME (char[10])	FIRSTNAME (char[10])	ADDRESS (char[25])	OPEN_DATE (char[10])	BANLANCE (float)
W999030031	Liu	Yi	260 Randolph Place, Windsor	01/01/1999	1000
W100000000	Liu	Hua	33 Isabella Street	01/01/2000	8000
W100324235	Du	Ying	438 Niagara Street, Windsor	10/10/1999	2000

checkingCustomer table

ID(int)	BADDRESS(char[20])	BCITY(char[10])
G666	4 Bloor Street	Toronto
G888	42 Ouellete Ave	Windsor

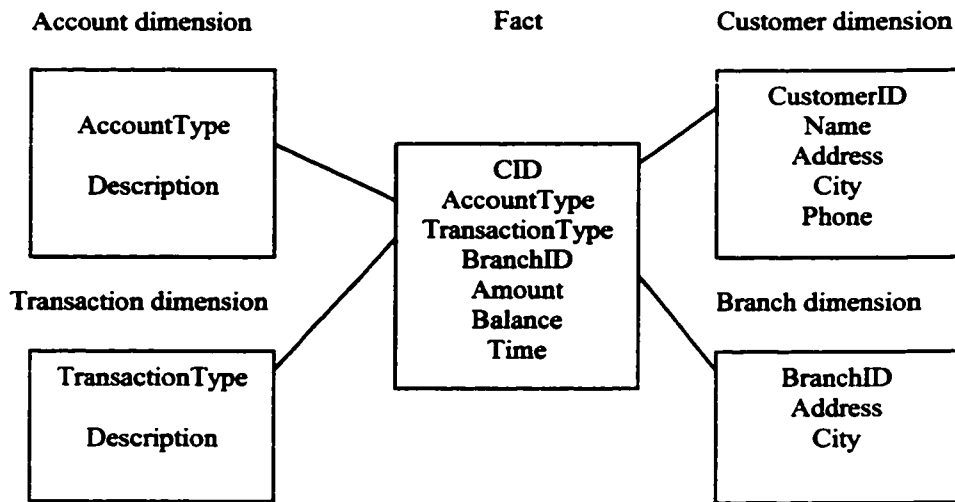
checkingCustomer table

ACCOUNTID(char[8])	BID(char[5])	TYPE(char[10])	AMOUNT(float)
W100324235	G888	WD	2000
W100000000	G888	TR	3000
W999030069	G666	DEP	1000

checkingTransaction table

Figure 4.8 Source Database Architecture

In WODD project, The purpose is to integrate the source databases shown in Figure 4.8 into the target data warehouse shown in Figure 4.9.



CID	AccountType	TransactionType	BranchID	Time	Amount	Balance
W999030031	S	DW	666	18082000123456	1000	3200
W100000000	S	DP	666	20001010123456	3000	8000
W100324235	C	DP	999	26121999123456	2000	0

Fact table

CustomerID	Name	Address	City	Phone
W999030031	Liu Yi	260 Randolph Place	Windsor	256-8338
W100324235	Du Ying	438 Niagara Street	Windsor	771-9867
W100000000	Liu Hua	33 Isabella Street	Toronto	666-8888

Customer-dimension table

BranchID	Address	City
G666	4 Bloor Street	Toronto
G888	42 Ouellete Ave	Windsor

Branch-dimension table

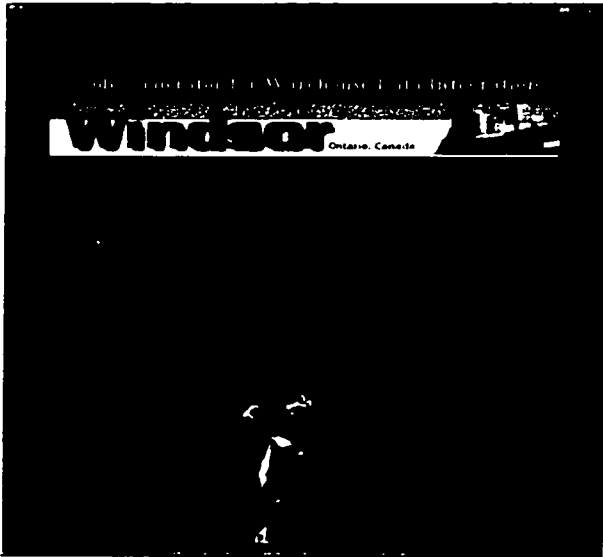
TransactionType	Description
DW	Withdraw
DP	Deposit
TF	Transfer

Transaction-dimension table

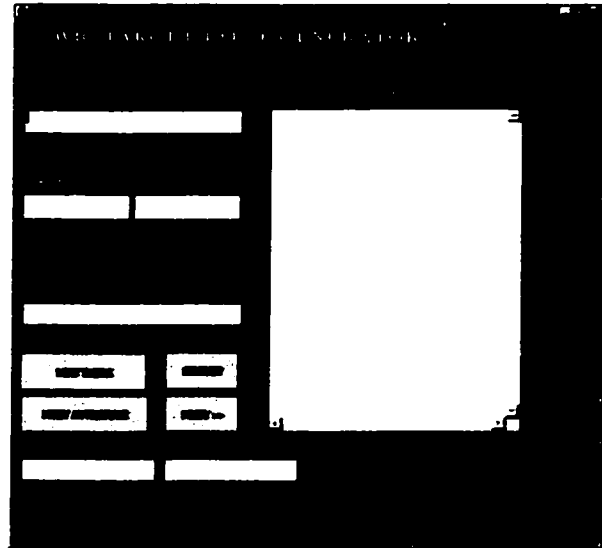
AccountType	Description
S	Saving
C	Checking

Account-dimension table

Figure 4.9 Data Warehouse Architecture



Start WODD GUI

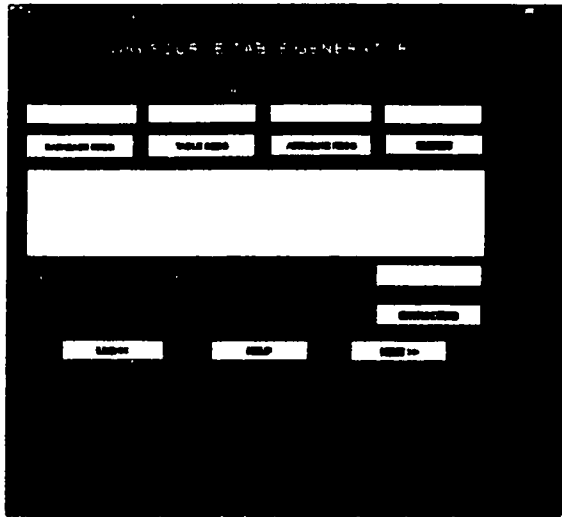


Data warehouse table generator

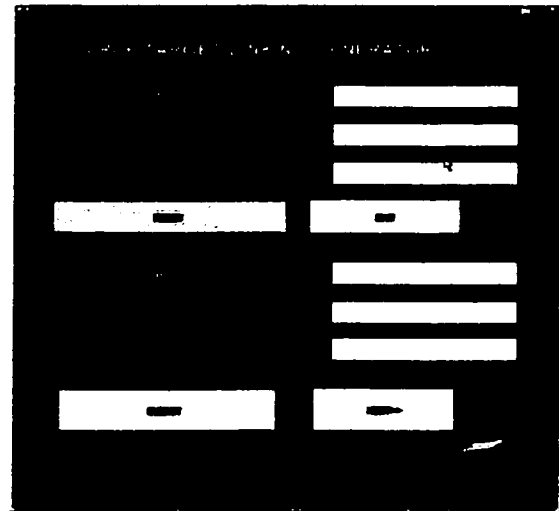
Running WODD system, start program by typing 'WODD' after '%' prompt in multi-user platform or '>' prompt in personal computer. *Start WODD GUI* is displayed. Clicking on 'Start WODD' button, WODD starts and *data warehouse table generator GUI* is activated.

Data warehouse table generator GUI is for generating a set of target data warehouse tables based on the user interests and business needs. In banking system, user wants to create data warehouse as discussed in Figure 4.4. After inputting target table definition from file or by manual, click 'submit' button, the table descriptions will appear on the right side of windows. Click the 'NEXT' button, *source database generator GUI* appears.

Source database generator GUI is for user to input the source definition by manual, from file or using JDBC connection. Source database can be single or multi databases with different database names, user ID and password. After inputting source database and tables definitions, system starts to generate the mapping rules that describe the relations between target DW and SDB, which will be done by *data integration rule generator GUI*.



Source Generator



Linking Generator

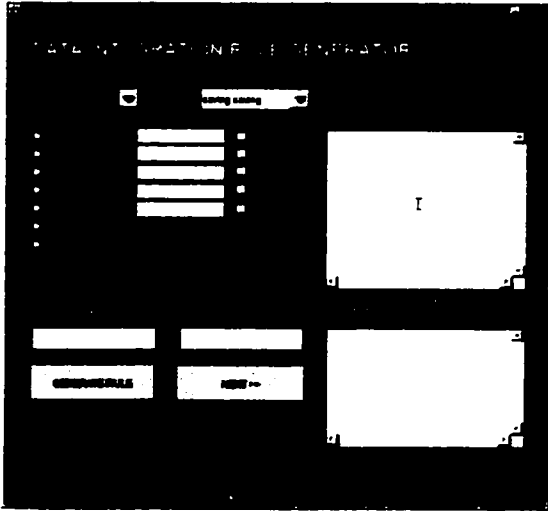
In *data integration rule generator GUI*, inputting the corresponding information and constraints between operational data store and source database tables. For example, the combination of `firstName` and `lastName` in source `checkingCustomer` table corresponds to the name in ODS `customer` table. The constraints is `fistrName || :lastName`. After checking all of the mapping relations, The set of integration rule in banking system will be created such as

```

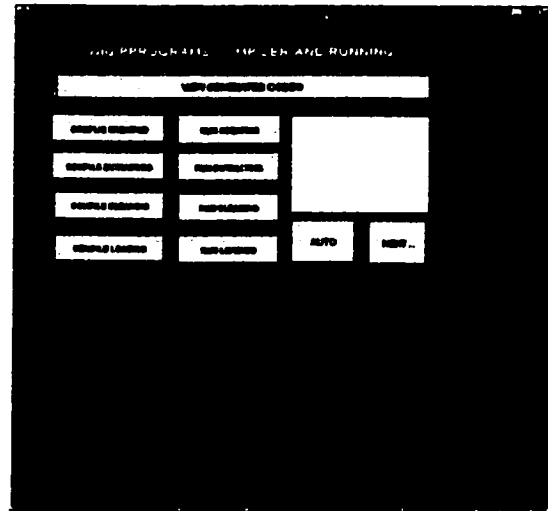
fact.cid <== saving.cid without constraint
fact.accounttype <== 'saving'
fact.transtype <== saving.type without constraint
fact.branchid <== branch.branchid without constraint
fact.amount <== saving.amt without constraint
fact.time <== to_char(sysdate)
fact.balance <== balance.balance without constraint
fact.cid <== ckecking.accid without constraint
fact.accounttype <== 'checking'
fact.transtype <== ckecking.type without constraint
fact.branchid <== brancid.id without constraint
fact.amount <== ckecking.amt without constraint
fact.time <== to_char(sysdate)
fact.balance <== checkcustomer.bal without constraint
customer.customerid <== customer.cid without constraint
customer.name <== customer.cname without constraint
customer.address <== customer.caddress without constraint
customer.city <== customer.ccity without constraint
customer.phone <== customer.cphone without constraint
customer.customerid <== checkingcustomer.id without constraint
customer.name <== checkingcustomer.lastname with constraint lastname || :firstname
customer.name <== checkingcustomer.firstname with constraint lastname || :firstname

```

customer.address	<==	checkingcustomer.addr without constraint
customer.city	<==	checkingcustomer.city without constraint
branch.branchid	<==	branch.branchid without constraint
branch.address	<==	branch.address without constraint
branch.city	<==	branch.city without constraint
branch.branchid	<==	customerBranch.branchid without constraint
branch.address	<==	customerBranch.baddress without constraint
branch.city	<==	customerBranch.bcity without constraint

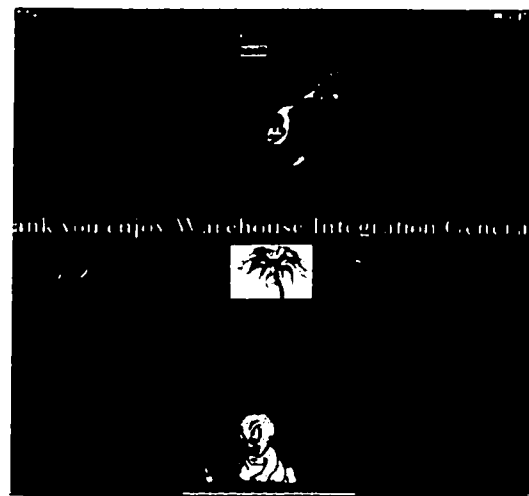
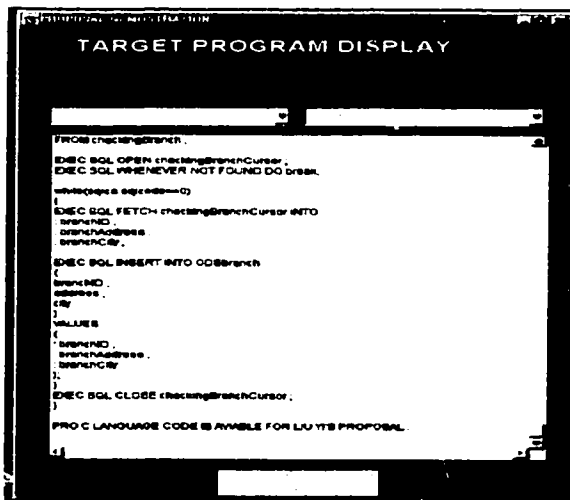


Rule Generator



Execution

These rules will go to background subsystem data integration and code generator to generate output programs and *Execution/display GUI* will appear. In *Execution/display GUI*, the generated code can be viewed or compiling/running the target programs. The last step is *End WODD GUI* for exiting WODD system.



4.6 Target Programs for Warehouse Data Integration

As discussed in Section 4.1, the target product of WODD code generator project is a set of data integration and transformation programs, which can be Pro*C programs. These programs may include `creating.pc` for creating target data warehouse, `extracting.pc` for extracting data from SDB to ODS, `cleaning.pc` for validating the data and removing the redundancy data from ODS, `loading.pc` for loading the cleaned data from ODS to DW and update the DW metadata as discussed in Section 1.2.2.

What is Pro*C, Pro*C is a pre-compiler that takes SQL statements embedded in a C program and converts it to standard C code. After pre-compile this code, the result is a C that can be compiled and used to build applications that access an Oracle database. One of the primary reasons to use Pro*C is to have the capability of utilizing SQL statements in a high-level language. Pro*C allow users to create highly customized applications, especially, for these applications which require the procedural processing power of C or must be done on a regular basis; some applications that run in the background without the need for user interaction. Besides, it's convenient, easy to use interface that lets the applications to access Oracle directly. Another reason using Pro*C is that Pro*C increases programmer productivity in a number of ways including shorter, more compact code; syntactic and semantic checking of source code at pre-compile time. Also tight integration with SQL, PL/SQL, and the Oracle RDBMS; and automatic handling of several tasks such as default cursor management and data type conversions. In WODD project, DW normally contains huge number of long historical data, which has to use the language that can invoke Oracle database more fast and efficient to process the huge number of data in both SDB and DW.

In WODD system, the target programs that are used for data extraction, data cleaning and loading following the discussion of DWS programs in Section 1.2.2

4.6.1 Data extraction:

The extracting program, first, calculates each data of each attribute of SDB for measurement integration based on the constraints of integration rule. The calculated data is a basis for future data integration, e.g., the amount in saving account database is represented by US dollar and in target data warehouse is Canadian dollar. Extracting program first calculates all '*amount*' in saving account database by multiplying the exchange rate in order to translate the US dollar to Canadian dollar.

And then, the extracting program will integrate different data type based on the relation pair of integration rule. Convert the attribute's data type to unique data type is a basis for guarantee these attributes can be combined for structure integration. Because the number data type is the most efficient when querying a DW, extracting program first try to convert data type to integer if possible. If not, try to convert data type following the order of float, char, string (varchar2), string (varchar) and date. For example, CID in savingCustomer table is integer 999030031 and ID in checkingCustomer is string data type "W999030031". This data type integration first to convert W999030031 to integer. Obviously, this conversion is impossible. So try to convert 999030031 to string data type "999030031". And now, both W999030031 and 999030031 are in same data type.

The last is the structure integration. When integration rules include the sentence such as firstName || lastName, it tell us that the data in two attributes will be combined together, e.g., the firstName is "Liu" and the lastName is "Hua" in checkingCustomer table, after structure integration, it will become "Liu Hua".

Finally, the extracting program will load the integrated data into ODS one source by source for future data cleaning. There will be big duplicating data in ODS, e.g., the tuple <"W999030031", "H. Liu", "33 Isabella Street", "Toronto"> from savingCustomer table and the tuple <"999030031", "Hua", "Liu", "33 Isabella Street", "Toronto"> from checkingCustomer are duplicating data.

4.6.2 Data cleaning

As introduced in Section 1.2.2, data cleaning can be done in SDB (before extracting), when data extracting (during extracting) or in ODS (after extracting). In WODD project, data cleaning will be done in ODS because virtual data integration is based on one to one relation and each relation will extract its own data from SDB to ODS independently. Data will not be validate and also many redundant data exist in ODS.

Cleaning programs first makes the data validation for each dimension table. Same as example above, <“1000000”, “H. Liu”, “33 Isabella Street”, “Toronto”> and <“W10000000”, “Hua Liu”, “33 Isabella Street”, “Toronto”> are not validated because “Hua Liu” and “H. Liu” are different. And also “W100000000” and “100000000” too. In order to validate these data, the cleaning program must check the attribute to see it is primer key or not. If only relating non-primary key attributes, validates the dimension table data. If it relates the primary key, not only need to validate dimension table but also fact table because any primary key in dimension table must in fact table according to DW theory.

- For non-key attribute validation: Only validates the dimension table’s data, e.g., convert “H. Liu” to “Hua Liu”. If more than one non-key attribute need validating, repeat this step.
- For key attribute validation: first validates the dimension table’s data, e.g., convert “999030031” to “W999030031” and then validates the fact table’s attributes because the key in dimension table always link to one attribute in fact table and that attribute in fact also need invalidation. Foe example, in ODS customer table, the turples for Hua Liu before and after validation are

Before validation

<“W100000000”, “H. Liu”, “33 Isabella Street”, “Toronto”>
<“100000000”, “Hua”, "Liu”, “33 Isabella Street”, “Toronto”>



After validation

<"W100000000", "Hua Liu", "33 Isabella Street", "Toronto">

<"W100000000", "Hua Liu", "33 Isabella Street", "Toronto">

After data validation, the two tuples in customer table of ODS will be same each other. And also the key "100000000" in fact table will be converted into "W100000000"

There are two questions need to be answer during the data validation. Question 1 is how to determine which two tuples are same even if they look like different. To answer this question, just simply to see all of the attributes that do not need validated. In the previous example, <"33 Isabella Street", "Toronto"> doesn't need validated. If any tuple with non-validation attribute is same as <"33 Isabella Street", "Toronto">, we can say they are same tuple and rest of the attributes except <"33 Isabella Street", "Toronto"> needs to be validated. For confidential reason, only if there are more than 50% of total attributes that are non- validation attributes, we can use it to determine the two tuples are same or not. For example: <"W100000000", "H. Liu", "401 Sunset", "Windsor"> and <"100000000", "Hua Liu", "33 Isabella Street", "Windsor"> are not same tuple because $\frac{3}{4}$ attributes need validation (non- validation attributes is only 25%). The second question is how to validating the number when the attribute need to be validated, e.g., "H. Liu" converts to "Hua Liu" or "Hua Liu" converts to "H. Liu". In the cleaning program, we follow the idea - keeping information as more as possible. Using the MAXUIN data of each group data with same non-validation data of each *column* (in the term of attribute in database), which means find the max data for each same tuple group data, convert other data of this column into this max value. Because one group data only have one max value, we don't worry about there is different value after validation. For example, before validation, the two same tuples are (50% same, sown as italic letters)

<"W100000000", "H. Liu", "33 Isabella Street", "Toronto">
<"100000000", "Hua Liu", "33 Isabella Street", "Toronto">

After validating the non key attribute, the two tuple will be (75% same now)

<"W100000000", "*Hua. Liu*", "33 Isabella Street", "Toronto">
<"100000000", "*Hua Liu*", "33 Isabella Street", "Toronto">

After validating the key attribute, the two tuples will be (100% same now) and also validating the data in fact table too. Here, in string data type, "W100000000" is bigger than "100000000" in ASCII code.

<*"W100000000"*, "*Hua. Liu*", "33 Isabella Street", "Toronto">
<*"W100000000"*, "*Hua Liu*", "33 Isabella Street", "Toronto">

After validation, the cleaning program will remove the redundant tuples from ODS. There are two different redundancy, One is there are two tuples are same in ODS. Another is one tuple in ODS is same as one of the tuple in DW. In this case, we only keep single (can not be duplicate in ODS), newer information (not in DW) in ODS. Up to now, we can say that the data in ODS are cleaned.

4.6.3 Data loading

As discussed in Section 1.2.2, data loading program will load the cleaned data into DW, update the DW metadata to record what is the last loading, how many record in each DW table, how many new record in each DW table, and also where the data come from.

5 Conclusions And Future Work

This chapter presents conclusions in section 5.1 and future research directions in section 5.2.

5.1 Conclusions

In this thesis, code generator for data warehouse data integration based on virtual data integration approach combined with schema data integration approach and semantic data integration approach is presented. A combination of template driven based code generation approach and rule based code generation approach is first presented, which are used for generating the Pro*C programs for the data warehouse creation, data extracting, cleaning and loading.

The main contributions of this thesis are

- (1) Proposing system architecture for the code generation of DWDI project that can be good for any target programming language with high flexibility.
- (2) Determining a data integration approach for the data warehouse that is good for code generation project
- (3) Migrating visual code generation approach to background code generation process for the code generation of the data warehouse.
- (4) Finding a data integration approach to combine the virtual data integration, schema integration and semantic data integration together.
- (5) Finding a code generation approach to combine the rule based code generation approach and template based code generation approach to generate the code.
- (6) Finding an approach for combining the chosen data integration and code generator approach for the code generation of data warehouse data integration.
- (7) Proposing a software development cycle for the project of the data warehouse integrator, code generator.
- (8) Implementing the component of the project that integrates traditional data source into warehouse by generating Pro*C code.

5.2 Future Work

As we already discussed, this thesis focuses on the local data warehouse generation. That means all of the source database and the data warehouse is in the same machine and the output language only for Pro*C.

- *Distribution/web*: How to create a code generation for the distributed/web data warehouse system is still a hot topic because the databases in today's market are distributed. Data warehouse needs the ability to extract the data from different site of data sources. So does code generator.
- Incorporating more efficient data cleaning and schema integration techniques to the code generator project are also interesting future work.

NOTE TO USERS

Page(s) not included in the original manuscript and are unavailable from the author or university. The manuscript was microfilmed as received.

78 - 81

This reproduction is the best copy available.

UMI[®]

Appendix A: User Manual

This chapter introduces about how to use WODD code generator system based on a banking system example.

Step 1: DW Definition Planning

Before using WODD to create DWDI programs, you should review local machine and the database carefully. Every table on this local machine could be SDB table even if some tables and databases you may not use actually.

Good beginning is the most important for generating correct output programs. First, you should plan what kind of target DW will be created. There are some restrictions that must be satisfied before running this system. Every target DWS tables should follow DW definition.

- What tables should be generated? According DW definition, at least, the target DW should have a fact table, metadata table, timetable and at least one dimension table. An example is that we want to create banking DWS includes *fact*, *customer* and *branch tables*.
- Which data should be in metadata. By default, WODD will offer some DW metadata items such as last update date, table names and the total new record in each table. You can add more parts as you wish, for example, your can add the last new record information or something else. Because the DW we creating bases on Oracle database, some meatadata information has been already offered by Oracle RDBMS, E.g., the attribute for each table and their data type. we will ignore in WODD. But you still can add them later.
- What attributes in each table? You should make sure that every attribute in target DW comes from one or more attributes of SDB (or SDB name). Never try to create target

attribute that can not get from SDB. WODD only integrates the information that really exists in the SDB.

- How about data types for each attribute. The data type for each attribute in DW must be same as SDB data types or can be converted from SDB data type. In fact, some data type can be not converted. For example, if customer ID in SDB is character data type such as “W100000000”, In target DW customer table, we can not use integer data type as customer ID because character can not convert into integer. WODD offers a very strong data type checking. If anything wrong, the system will throw the error exception and WODD will be terminated. Sometimes, this system can guess some simple errors and correct them automatically, e.g., if inputting the target data type as *var.*, the system will guess it should be *varchar2* (oracle data type) and convert *var* data type to *varchar2*. But in some cases, WODD can not guess what you really want. For example, converting 'W100000000' to integer.

An example of planning is the banking SDB shown in Figure 4.3. Because in SDB, customerID in checkingCustomer is varchar2(20), which the data may like 'W100000000', and can not convert to integer. When planning target DW data type, you should use varchar2(20) as customerID data type. So the target DW could be as Figure 4.4

Step 2: Mapping Relation Planning

(1) Planning structure integration: Generally, Planning mapping relation should find every relation between each SDB attribute and each target DW attribute: For structure integration, it could be in three cases:

- *One to one relation.* Relation between one source attribute and one target attribute: e.g., city in SDB table correspondences to the city in target DW table.
- *Many to one relation.* Relation between several source attributes and one target attribute; e.g., first name and last name in SDB can be full name in target DW.

- *Many to many relation.* For multi source attributes to multi target attributes relation. They can be divided into several many to one relation based on the virtual data integration approach.
- (2) **Planning measurement conversion:** For example, *amount* in SDB is US dollar and in target DW may be Canadian dollar. You should list all of the conversion accurately. Otherwise, the result in target database will be non-meaningful.
- (3) **Other planning:** Rest of the integration relation such as data type, data redundancy integration will be considered by WODD automatically. But if you have time, try to do them too because it is good for you to fully understand what the data integration is and how to handle integration by hand using the thesis approach in order to know how data integration can be done in WODD code generator system.

Step 3: Start WODD

Congratulation, your can start WODD code generation system. Because this system is the platform independent, your can run this system under any machine and operating system.

- Under UNIX multi user system (SUN/Solar, Unix/SGI etc): simply type *wig* after %.
- Under PC/DOS or Window platform, type *wig.bat* after *c>* sign.

WODD code generation system will start to display a dark green window with a welcome information. This window is an interface that tells user they have started WODD code generation system. Click the button 'push me' on the busy cat, the system will go to the next step-*DW table generation* window.

Step 4: DW Table Generation

DW table generation window is an interface that accepts the user interesting DW definitions and output DW template. DW definitions include table name, attribute, data type and the primer key constraint. See Figure A1.1

DW TABLE GENERATOR

<p>REQUIRED TABLE</p> <div style="border: 1px solid black; height: 25px; width: 100%; margin-bottom: 10px;"></div> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%; text-align: center;">ATTRIBUTE NAME</td> <td style="width: 50%; text-align: center;">ATTRIBUTE TYPE LENGTH</td> </tr> <tr> <td style="text-align: center;"><div style="border: 1px solid black; height: 25px; width: 100%;"></div></td> <td style="text-align: center;"><div style="border: 1px solid black; height: 25px; width: 100%;"></div></td> </tr> </table> <p> <input checked="" type="radio"/> KEY <input type="radio"/> NON KEY </p> <p>INPUT FROM FILE</p> <div style="border: 1px solid black; height: 25px; width: 100%; margin-bottom: 10px;"></div> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%; text-align: center; border: 1px solid black; padding: 5px;">NEXT ATTRIBUTE</td> <td style="width: 50%; text-align: center; border: 1px solid black; padding: 5px;">SUMIT</td> </tr> <tr> <td style="width: 50%; text-align: center; border: 1px solid black; padding: 5px;">NEXT TABLE</td> <td style="width: 50%; text-align: center; border: 1px solid black; padding: 5px;">GO RULE</td> </tr> </table>	ATTRIBUTE NAME	ATTRIBUTE TYPE LENGTH	<div style="border: 1px solid black; height: 25px; width: 100%;"></div>	<div style="border: 1px solid black; height: 25px; width: 100%;"></div>	NEXT ATTRIBUTE	SUMIT	NEXT TABLE	GO RULE	<p>TABLE ATTRIBUTE TYPE</p> <div style="border: 1px solid black; height: 300px; width: 100%;"></div>
ATTRIBUTE NAME	ATTRIBUTE TYPE LENGTH								
<div style="border: 1px solid black; height: 25px; width: 100%;"></div>	<div style="border: 1px solid black; height: 25px; width: 100%;"></div>								
NEXT ATTRIBUTE	SUMIT								
NEXT TABLE	GO RULE								

Figure A1.1: DW table generator window

There are two ways to input DW definitions from DW table generation window: by manual or from file:

(1) By manual:

As Figure A1.1, there are four text fields for inputting user interesting DW definitions, which include table name, attribute, and data type/length and key constraints. In these text fields, required table name can be *fact* table and *dimension* tables. Attribute name text field can be used to input attribute names of each table. Attribute type/length can be used to input the attribute data type and length.

DW TABLE GENERATOR

REQUIRED TABLE

ATTRIBUTE NAME	ATTRIBUTE TYPE LENGTH
<input type="text" value="customerID"/>	<input type="text" value="Varchar2(20)"/>

KEY NON KEY

INPUT FROM FILE

TABLE ATTRIBUTE TYPE

Figure A1.2: Inputting DW definition manually

DW TABLE GENERATOR

REQUIRED TABLE

ATTRIBUTE NAME	ATTRIBUTE TYPE LENGTH
<input type="text" value="customerID"/>	<input type="text" value="Varchar2(20)"/>

KEY NON KEY

INPUT FROM FILE

TABLE ATTRIBUTE TYPE

Fact customerID varchar2(20) key

Figure A1.3: Display DW definition

In banking example, we planed DW table definitions in Step 1. After inputting *fact* table name and *customerID*, DW table generation window will look like Figure A1.2. Don't forget that *customerID* is a key attribute. After clicking "submit" button to submit this attribute definition to WODD system, the inputted attribute information will be displayed on the right side of the display area. See Figure A1.3

Next, input the non-key attribute – *amount* for the fact table and click "submit" button. The window will display the amount information of fact table on the right side of the window. See Figure A1.4

DW TABLE GENERATOR

REQUIRED TABLE

fact

ATTRIBUTE NAME	ATTRIBUTE TYPE LENGTH
amount	Number(10)

KEY NON KEY

INPUT FROM FILE

NEXT ATTRIBUTE

SUMIT

NEXT TABLE

GO RULE

TABLE ATTRIBUTE	TYPE
Fact customerID	varchar2(20) key
Fact amount	number(10) non key

Figure A1.4: Input all for fact table

Now, input other table definition one by one and final DW information should look like Figure A1.5. This is a very simply example for DWS. If an application is more complex, it can be very huge and time consuming. The better way for a bigger application is inputted by file.

DW TABLE GENERATOR

REQUIRED TABLE

ATTRIBUTE NAME	ATTRIBUTE TYPE LENGTH
-----------------------	------------------------------

KEY

 NON KEY

INPUT FROM FILE

<input type="button" value="NEXT ATTRIBUTE"/>	<input type="button" value="SUMIT"/>
<input type="button" value="NEXT TABLE"/>	<input type="button" value="GO RULE"/>

TABLE ATTRIBUTE	TYPE
fact	cid varchar2(20)
fact	accounttype varchar2(20)
fact	transtype varchar2(20)
fact	branchid varchar2(20)
fact	amount number(10)
fact	balance number(10)
fact	time varchar2(20)
customer	customerid varchar2(20)
customer	name varchar2(20)
customer	address varchar2(20)
customer	city varchar2(20)
customer	phone varchar2(20)
branch	branchid varchar2(20)
branch	address varchar2(20)
branch	city varchar2(20)

Figure A1.5: Input all definition for banking DW table

(2) From File:

In order to input the DW table definition from file, we should create a text file that contains DW table definitions such as *target.txt* using any text editor. Inputting definitions into text file must follow this format/order as

<table name, data type, length, key or non key>

In banking example, the DWS table definition can be as Figure 1.6

```

fact, cid, varchar2(20), null
fact, accounttype, varchar2(20), null
fact, transtype, varchar2(20), null
fact, branchid, varchar2(20), null
fact, amount, number(10), null
fact, balance, number(10), null
fact, time, varchar2(20), null
customer, customerid, varchar2(20), key
customer, name, varchar2(20), nonKey
customer, address, varchar2(20), null
customer, city, varchar2(20), null
customer, phone, varchar2(20), null
branch, branchid, varchar2(20), key
branch, address, varchar2(20), null
branch, city, varchar2(20), null

```

Figure A1.6: DW definition in *target.txt* text file

DW TABLE GENERATOR

REQUIRED TABLE

ATTRIBUTE NAME	ATTRIBUTE TYPE LENGTH
<input style="width: 90%; height: 20px;" type="text"/>	<input style="width: 90%; height: 20px;" type="text"/>

KEY NON KEY

INPUT FROM FILE

<input type="button" value="NEXT ATTRIBUTE"/>	<input type="button" value="SUMIT"/>
<input type="button" value="NEXT TABLE"/>	<input type="button" value="GO RULE"/>

TABLE ATTRIBUTE	TYPE

Figure A1.7: DW Input from file

Simply input the file's name as Figure A1.7, and press "submit" button. WODD will check the file name is right or not first. If the name is wrong, the system will ask user to do it again. When name is right, DW table definitions will display on the right side of the window shown as Figure A1.5

After inputting all of target DW definitions, click the "Next" button. There are some information such as "new template will be created" will be displayed in order to tell user that WODD starts to create a set of empty templates for the DW table creation, data cleaning and data loading. Then, the system will display a window called "Source database generator" as the Figure A2.1 below.

Step 5: SDB Generation Window

Source database Generator			
Database Name	database table	database attribute	database data type
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>			
Input from File	<input type="text"/>		
Extracting definition automatically	<input type="button" value="Extracting"/>		
<input type="button" value="Linking <<"/>	<input type="button" value="Help"/>	<input type="button" value="Next >>"/>	

Figure A 2.1 Source generator window

Same as DW generation window, SDB definition can be input into WODD from SDB generation window by manual, from text file or using JDBC extraction.

(1) By manual

As Figure A2.1, there are four text fields for inputting SDB definitions, which include database name, table name, attribute, data type/length. In these text fields, required database name can be business name (such as *saving* and *checking* in banking example). Table name is the SDB table name, Attribute name text field can be used to input attribute names of each table. Data type can be used to input the attribute data type and length.

In banking example, we review SDB in Step 1. After inputting database name - saving, table name - savingCustomer and attribute *customerID*, SDB table generation window will look like Figure A2.2. After clicking “submit” button to submit this attribute definition to WODD system, the inputted attribute information will be displayed on the right side of the display area. See Figure A2.3. repeat input all definition shown as Figure 2.4

Source database Generator			
Database Name	database table	database attribute	database data type
<input type="text" value="saving"/>	<input type="text" value="savingCustome"/>	<input type="text" value="customerID"/>	<input type="text" value="Number(10)"/>
<input type="text"/>			
Input from File	<input type="text"/>		
Extracting definition automatically	<input type="button" value="Extracting"/>		
<input type="button" value="Linking <<"/>	<input type="button" value="Help"/>	<input type="button" value="Next >>"/>	

Figure 2.2 Input source Definition manually

Source database Generator

Database Name	database table	database attribute	database data type
saving	savingCustome	customerID	Number(10)

Saving	savingCustomer	customerID	Number(10)
--------	----------------	------------	------------

Input from File

Extracting definition automatically

Linking <<

Help

Next >>

Figure A 2.3 Display SDB definition

Source database Generator

Database Name	database table	database attribute	database data type

Saving	saving	cid	number(10)
Saving	saving	branchid	varchar2(20)
Saving	saving	transtype	varchar2(20)
Saving	saving	amt	number(10)
Saving	saving	status	varchar2(10)
Saving	branch	branchid	varchar2(20)

Input from File

Extracting definition automatically

Linking <<

Help

Next >>

Figure A 2.4 Display all of SDB definition

(2) From File:

In order to input the SDB table definition from file, we should create a text file that contains SDB table definitions such as *source.txt* using any text editor. Inputting definitions into text file must follow this format/order as

<database name, table name, data type, length>

In banking example, the SDB table definition can be as Figure 2.5

```
saving, saving, cid, number (10)
saving, saving, branchid, varchar2 (20)
saving, saving, transtype, varchar2 (20)
saving, saving, amt, number (10)
saving, saving, status, varchar2 (10)
saving, branch, branchid, varchar2 (20)
saving, branch, address, varchar2 (20)
saving, branch, city, varchar2 (10)
saving, customer, cid, number (10)
saving, customer, cname, varchar2 (20)
saving, customer, caddress, varchar2 (20)
saving, customer, ccity, varchar2 (20)
saving, customer, cphone, varchar2 (10)
saving, balance, cid, number (10)
saving, balance, opendate, date
saving, balance, balance, number (10)
checking, cchecking, accid, varchar2 (20)
checking, cchecking, bid, varchar2 (20)
checking, cchecking, type, varchar2 (20)
checking, cchecking, amt, number (10)
checking, checkcustomer, acctid, varchar2 (20)
checking, checkcustomer, lastname, varchar2 (20)
checking, checkcustomer, firstname, varchar2 (20)
checking, checkcustomer, addr, varchar2 (20)
checking, checkcustomer, city, varchar2 (20)
checking, checkcustomer, open_date, number (10)
checking, checkcustomer, bal, number (10)
checking, branch, bid, varchar2 (20)
checking, branch, baddress, varchar2 (20)
checking, branch, bcity, varchar2 (20)
```

Figure A 2.5 SDB definition source.txt text file

Simply input the file's name as Figure A2.6, and press "submit" button. WODD will check the file name is right or not first. If the name is wrong, the system will ask user to do it again. When name is right, SDB table definitions will display on the middle of the window shown as Figure A2.3

After inputting all of target SDB definitions, click the "Next" button. Then, the system will display a window called "rule generator" as the Figure A3.1 to input the DW/SDW integration mapping rules.

Source database Generator			
Database Name	database table	database attribute	database data type
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>			
Input from File	<input type="text" value="Source.txt"/>		
Extracting definition automatically	<input type="button" value="Extracting"/>		
<input type="button" value="Linking <<"/>	<input type="button" value="Help"/>	<input type="button" value="Next >>"/>	

Figure A 2.6 Input SDB definition from file

Step 6 Rule Generation Window

There are two ways to generate relation pairs for WODD data integration - by manual or from file

(1) By manual

As shown in Figure A3.1, there are two lists in this window. One displays FACT and another display savingCustomer. Below fact is a set of attribute names such as customerID and amount, which belong to fact table. Below savingCustomer is another set of attribute names that belong to savingCustomer table. The left-hand list is the DW target table that generated from the table generator window. The left side list is the SDB tables that come from source database generation window.

DATA INTEGRATION RULE GENERATOR

FACT ▼ savingCustomer ▼ Integration rule

○customerID [] ○ CID

○amount [] ○ amt

... ...

Constraints input from file

[] []

Submit Generate

Warning

[]

Figure A3.1: Integration rule generator window

DATA INTEGRATION RULE GENERATOR

FACT ▼	savingCustomer ▼	Integration rule
○customerID <input style="width: 100px;" type="text"/>	○ CID	Fact.customerID ← saving.savingcustomer.CID
○amount <input style="width: 100px;" type="text"/>	○ amt	
Constraints <input style="width: 100px;" type="text"/>	input from file <input style="width: 100px;" type="text"/>	Warning <input style="width: 100px;" type="text"/>
Submit	Generate	

Figure A3.2: Inputting mapping rule

From the step 2, we know the first relation pair should be

Fact.customerID ← saving.savingcustomer.CID.

That means that the CID in savingcustomer table (SDB) will extract into customerID in fact (DW). Simply click each ID radio button on from both sides, press the button “submit”, the integration rule will display on the text area called “integration rule”. See Figure A3.2.

When there are different measure units between target DW attribute and SDB attribute such as US dollar in SDB side and Canada dollar in DW side, input the rate (external data) from the text field between DW attribute and SDB attribute’s name. For example, assume the *amount* in DW is Canadian dollar and *amt* is US dollar in SDB, input the exchange rate and calculation * 0.76, click both radio buttons and press “submit” button, the result will display on the text area on the right side as Figure A3.3.

DATA INTEGRATION RULE GENERATOR

FACT ▼	savingCustomer ▼	Integration rule
○customerID <input type="text"/>	○ CID	Fact.customerID ← saving.savingCustomer.CID Fact.amount ← saving.savingCustomer.amt * 0.76
○amount <input type="text" value="* 0.76"/>	○ amt	
Constraints <input type="text"/>	input from file <input type="text"/>	Warning
<input type="button" value="Submit"/>	<input type="button" value="Generate"/>	

Figure A3.3: measurement integration rule

After repeating this step to input all relations between savingCustomer and fact table, we can start to input the relation pairs between DW fact and SDB checkingCustomer. Click the list on the right side, choose the SDB called checkingCustomer. The right side of the radio group will display the attribute of this table. Like savingCustomer, you can choose the pair for each DW target attribute and checkingCustomer attribute. Click the “submit” button after finishing all.

After finish the fact table, we can start to choose the pairs between customer table (DW) and SDB. Simply click the list on the left side of the window, choose the customer from the list, the attribute belong to customer table will display now. Choosing the right side table checkingCustomer from right side list. The attribute of checkingCustomer will be display, which will be look like Figure A3.4.

DATA INTEGRATION RULE GENERATOR

<input type="text" value="customer"/>	▼	<input type="text" value="checkingCustomer"/>	Integration rule
○ customerID <input type="text"/>	○ CID	<pre>fact.cid <= saving.cid without constraint fact.accounttype <= . with constraint 'savings' fact.transtype <= saving.transtype without constraint</pre>	
○ name <input type="text"/>	○ firstname		
○ name <input type="text"/>	○ lastname		
Constraints <input type="text"/>	input from file <input type="text"/>	Warning	
<input type="button" value="Submit"/>	<input type="button" value="Generate"/>		

Figure A3.4: Mapping rule display

Now assume that the name combines the model in SDB is the name in DW (structure integration). In this case, we can input the structure relation in the text field called constraints. Simply type *name || model* in constraints text field (means combining the name and model together) as shown in Figure A3.5

DATA INTEGRATION RULE GENERATOR

<input type="text" value="customer"/>	▼	<input type="text" value="checkingCustomer"/>	Integration rule
○ customerID <input type="text"/>	○ CID	<pre>Customer.customerID ← checkingcustomer.lastname with constraint lastname :firstname</pre>	
○ name <input type="text"/>	○ firstname		
○ name <input type="text"/>	○ lastname		
Constraints <input type="text" value="Firstname : lastName"/>	input from file <input type="text"/>	Warning	
<input type="button" value="Submit"/>	<input type="button" value="Generate"/>		

Figure A3.5: Structure mapping rule

and press the submit button. The rule information will be display on the right side of text field. You can see the rule now with constraints like Firstname ||: lastName. Repeat inputting all of the relations for ckeckingCustomer, all of the relation pairs will be inputted and all of integration rule will be created too. Finally, press the button to generate the code.

(2) From file

Same as the table generation window, First you should create a text file that contains both target DW information and SDB. Like the Step 3, name a text file as rule.txt. The information organized in rule.txt should follow the order as <DW table name, DW attribute name, data type, SDB table name, SDB attribute, SDB data type and constraints>. For example, in banking DWS, the text information for both DW and SDB shown as Figure 3.6

```

fact,cid, fact,cid,varchar2(20),saving,saving,cid,number(10),null
fact,accounttype,varchar2(20),saving, , , , 'savings'
fact,transtype,varchar2(20),saving,saving,transtype,varchar2(20),null
fact,branchid,varchar2(20),saving,branch,branchid,varchar2(20),null
fact,amount,number(10),saving,saving,amt,number(10),null
fact,time,varchar2(20),saving, , , ,to_char(SYSDATE)
fact,balance,number(10),saving,balance,balance,number(10),null
fact,cid,varchar2(20),checking,checking,accid,varchar2(20),null
fact,accounttype,varchar2(20),checking, , , , 'checking'
fact,transtype,varchar2(20),checking,checking,type,varchar2(20),null
fact,branchid,varchar2(20),checking,checking,bid,varchar2(20),null
fact,amount,number(10),checking,checking,amt,number(10),null
fact,time,varchar2(20),checking, , , ,to_char(SYSDATE)
fact,balance,number(10),checking,checkingcustomer,bal,number(10),null
customer,customerid,varchar2(20),saving,customer,cid,number(10),null
customer,name,varchar2(20),saving,customer,cname,varchar2(20),null
customer,address,varchar2(20),saving,customer,caddress,varchar2(20),null
customer,city,varchar2(20),saving,customer,ccity,varchar2(20),null
customer,phone,varchar2(20),saving,customer,cphone,varchar2(20),null
customer,customerid,varchar2(20),checking,checkingcustomer,accid,varchar2(20),null
customer,name,varchar2(20),checking,checkingcustomer,lastname,varchar2(20),lastname || :firstname
customer,name,varchar2(20),checking,checkingcustomer,firstname,varchar2(20),lastname || :firstname
customer,address,varchar2(20),checking,checkingcustomer,addr,varchar2(20),null
customer,city,varchar2(20),checking,checkingcustomer,city,varchar2(20),null
customer,phone,varchar2(20),checking,checkingcustomer,phone,varchar2(20),null
branch,branchid,varchar2(20),saving,branch,branchid,varchar2(20),null
branch,address,varchar2(20),saving,branch,address,varchar2(20),null
branch,city,varchar2(20),saving,branch,city,varchar2(20),null
branch,branchid,varchar2(20),checking,checkingbranch,bid,varchar2(20),null
branch,address,varchar2(20),checking,checkingbranch,baddress,varchar2(20),null
branch,city,varchar2(20),checking,checkingbranch,bcity,varchar2(20),null

```

Figure A3.6: Mapping relation text file

After typing all of the DW and SDB information into *rule.txt* file, input the file name in the text field called input from file as Figure A3.7

DATA INTEGRATION RULE GENERATOR

FACT ▼ CARFACTORY ▼ Integration rule

○ PID ○ PID

○ name ○ name

Constraints input from file

 Rule.txt

Submit Generate

Warning

Figure A3.7: Rule Input from file

DATA INTEGRATION RULE GENERATOR

FACT ▼ CARFACTORY ▼ Integration rule

○ PID ○ PID

○ name ○ name

Constraints input from file

Submit Generate

Warning

fact, PID, varchar2(20),
catFactory, PID, number(10), non
constraint
fact, amount, varchar2(20),
catFactory, amount,

Figure A3.8: Display the rule

And now, you can click the button “submit”, the rule will display on the right side of the integration rule text area. See Figure A3.8

After finishing the rule generation, click the button “Generate”, the system will use the rules to generate the target code. Execution/display window appears.

Step 7 Execution/Display window

In order to make WODD working easily, WODD has a function called Execution/display window for compiling target program and running the target program to integrate SDB data into DW.

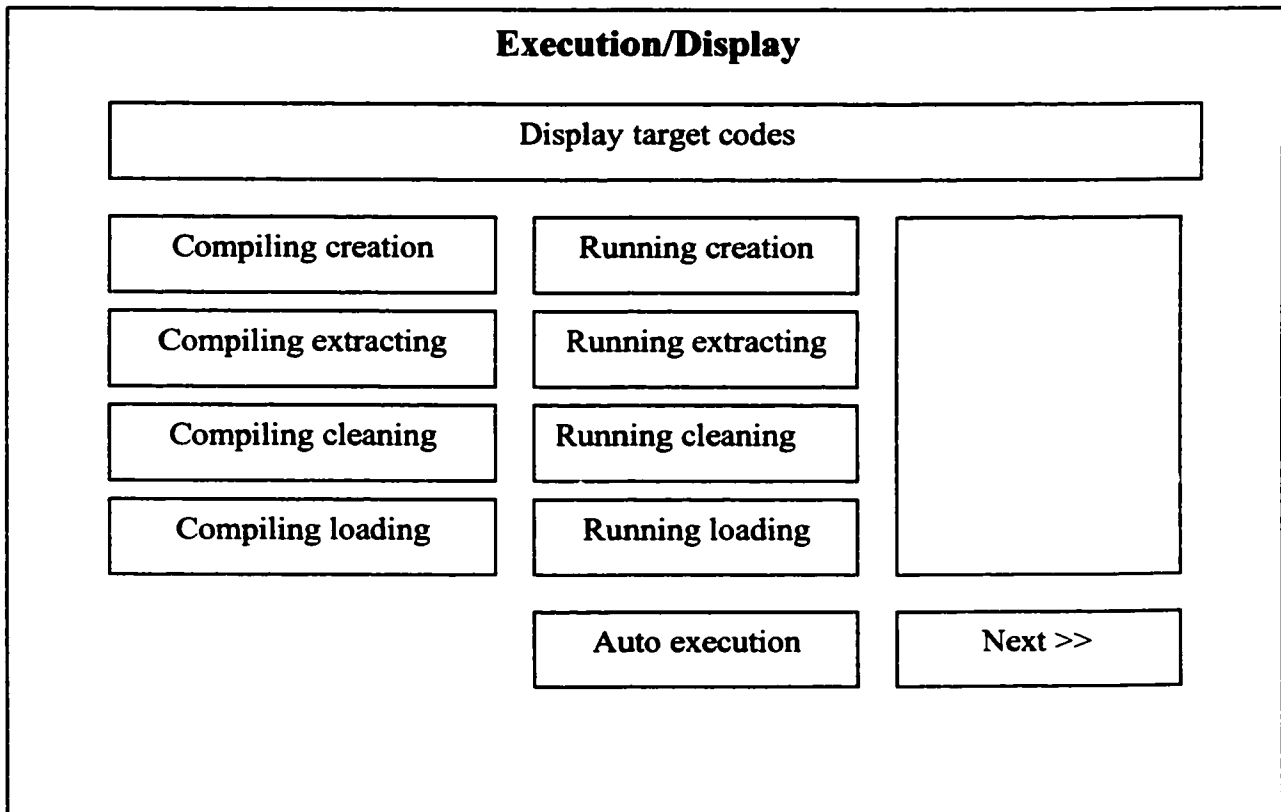


Figure A4.1: Target code execution and display

As shown in Figure A4.1, *display* button uses for displaying the target code generated by WODD. *Compiling* buttons use for compiling different target programs and *Running* buttons use for running the different target programs. A button called *auto execution* is for compiling and running all of target programs sequentially.

Type next button, WODD displays the *END WIG window*, which indicates that the WODD is ending and warehouse has been created successfully. Press *End WIG* button, the system will be quit and system end.

Appendix B: WODD codes

B.1 WODD programs

(1) WIG batch file

```
clear
echo "*****"
echo ""
echo ""
echo "    <WAREHOUSE DATA INTEGRATION CODE GENERATION>"
echo ""
echo ""
echo ""
echo ""
echo "--University of Windsor, School of Computer Science"
echo ""
echo "    Dr.    C.Ezeife"
echo "    Student Liu Yi"
echo ""
echo "    trail version 1.1"
echo ""
echo ""
echo ""
echo ""
echo "*****"
echo ""
echo ""
echo ""
echo "Remove old  Java class and pro*C files"
rm *.pc
rm *.class
rm *.sql
echo ""
echo ""
echo "Compiling <WAREHOUSE DATA INTEGRATION CODE GENERATION> java programs"
javac *.java
echo ""
echo "Running <WAREHOUSE DATA INTEGRATION CODE GENERATION> to create pro*C codes"
echo "For <WAREHOUSE DATA INTEGRATION>"
java WIG
echo ""
echo "Compiling creating.pc For Creating Target Data Warehouse"
make -f procl6.mk creating
echo ""
echo "Running creating.pc For Creating Target Data Warehouse"
creating
echo ""
echo "Removing creating pro*C program For saving space"
rm creating
rm *.c
rm *.o
echo ""
echo "Compiling extracting.pc For extracting data from Source  Database to Target Data Warehouse"
make -f procl6.mk extracting
echo ""
echo "Running extracting.pc For extracting data from Source  Database to Target Data Warehouse"
extracting
echo ""
echo "Removing creating pro*C program For saving space"
rm extracting
rm *.c
rm *.o
echo ""
echo "Compiling cleaning.pc For cleaning data from Operational Data Store"
make -f procl6.mk cleaning
echo "Running cleaning.pc For cleaning data from Operational Data Store"
```

```

cleaning
echo ""
echo "Removing creating pro*C program For saving space"
rm cleaning
rm *.c
rm *.o
echo ""
echo "Compiling loading.pc For loading data from Operational Data Store to Data Warehouse"
make -f procl6.mk loading
echo "Running loading.pc For loading data from Operational Data Store to Data Warehouse"
loading
echo "Removing loading pro*C program For saving space"
rm loading
rm *.c
rm *.o
echo ""
echo "Finish <WAREHOUSE DATA INTEGRATION CODE GENERATION>"
java endWIG
rm *.class

```

(2) WIG.java

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;

public class WIG extends JFrame
{
    public JPanel title =new JPanel();
    public JPanel control=new JPanel();
    public JButton button =new JButton();
    public JLabel label =new JLabel();
    public JLabel label0 =new JLabel();
    public GridBagLayout gb=new GridBagLayout();;
    public GridBagConstraints gbc=new GridBagConstraints();;

    public WIG()
    {
        super ("Warehouse Integration Generator");

        Container c=getContentPane();
        c.setLayout(new BorderLayout());

        Icon bug0=new ImageIcon("butterfly.gif");
        label0.setText("");
        label0.setIcon(bug0);
        label0.setHorizontalTextPosition(SwingConstants.CENTER);
        label0.setVerticalTextPosition(SwingConstants.BOTTOM);
        label0.setToolTipText("label0");
        label0.setFont(new Font("TimesRoman",Font.BOLD,26));
        title.add(label0);
        title.setBackground(SystemColor.desktop);
        c.setBackground(SystemColor.desktop);
        c.add(title,BorderLayout.NORTH);

        Icon bug=new ImageIcon("windsor.gif");
        label.setText("Code Generator for Warehouse Data Integration");
        label.setIcon(bug);
        label.setHorizontalTextPosition(SwingConstants.CENTER);
        label.setVerticalTextPosition(SwingConstants.TOP);
        label.setToolTipText("label");
        label.setFont(new Font("TimesRoman",Font.BOLD,26));
        label.setForeground(Color.yellow);
        title.add(label);
        title.setBackground(SystemColor.desktop);
        c.setBackground(SystemColor.desktop);
        c.add(title,BorderLayout.CENTER);

        control.setLayout(gb);
        gbc.weightx=1;
    }
}

```

```

gbc.weighty=1;
gbc.gridwidth=GridBagConstraints.REMAINDER;

Icon bug1=new ImageIcon("angry.gif");
Icon bug2=new ImageIcon("boy.gif");
button=new JButton("START WIG",bug1);
button.setRolloverIcon(bug2);
button.setFont(new Font("TimesRoman",Font.BOLD,18));
button.setForeground(Color.yellow);
button.setBackground(SystemColor.desktop);
gb.setConstraints(button,gbc);
control.setBackground(SystemColor.desktop);
control.add(button);

ButtonHandler buttonHandler=new ButtonHandler();
button.addActionListener(buttonHandler);
c.setBackground(SystemColor.desktop);
c.add(control, BorderLayout.SOUTH);

setSize(700,700);
show();
}

public class ButtonHandler implements ActionListener
{

public void actionPerformed (ActionEvent e)
{
setVisible(false);
tableGenerator tg=new tableGenerator();
}
}

public static void main(String[] args)
{
WIG app=new WIG();

app.addWindowListener
(
new WindowAdapter()
{
public void windowClosing(WindowEvent e)
{
System.exit(0);
}
}
);
}
}

```

(3) tableGenerator.java

```

import javax.swing.UIManager;

public class tableGenerator
{
boolean packFrame = false;

//Construct the application
public tableGenerator()
{
Frame1 frame = new Frame1();
//Validate frames that have preset sizes

```

```

//Pack frames that have useful preferred size info, e.g. from their layout
if (packFrame)
{
    frame.pack();
}
else
{
    frame.validate();
}
frame.setVisible(true);
}

//Main method
public void userQuery()
{
    try
    {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }

    new tableGenerator();
}
}

```

(4) Frame1.java

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import java.io.*;

public class Frame1 extends JFrame
{
    JPanel contentPane;
    Box box1;
    Label title = new Label();
    Label label1 = new Label();
    TextField textField1 = new TextField();
    Label label3 = new Label();
    TextField textField2 = new TextField();
    Button button1 = new Button();
    Button button2 = new Button();
    MenuBar menuBar1 = new MenuBar();
    MenuBar menuBar2 = new MenuBar();
    Button button3 = new Button();
    Label label4 = new Label();
    Label label2 = new Label();
    Button button4 = new Button();
    TextArea textArea1 = new TextArea();
    TextField textField3 = new TextField();
    TextField textField4 = new TextField();
    TextField textField5 = new TextField();
    TextField textField6 = new TextField();
    Panel panel1 = new Panel();
    JRadioButton jRadioButton1 = new JRadioButton();
    JRadioButton jRadioButton2 = new JRadioButton();
    Label label5 = new Label();
    String file="";

    Vector table          = new Vector();
    Vector attribute      = new Vector();
    Vector type           = new Vector();
    Vector check          = new Vector();
    Vector attributeTemp  = new Vector();
    Vector typeTemp       = new Vector();
}

```

```

Vector checkTemp    = new Vector();

//Construct the frame
public Frame1()
{
    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
    try
    {
        jbInit();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}

//Component initialization
private void jbInit() throws Exception
{
    title.setBounds(new Rectangle(17, 9, 612, 23));
    title.setFont(new java.awt.Font("Dialog", 1, 26));
    title.setForeground(Color.yellow);
    title.setLocale(new java.util.Locale("ar", "BH", ""));
    title.setText("    WIG TARGET TABLE GENERATOR");
    contentPane = (JPanel) this.getContentPane();
    box1 = Box.createHorizontalBox();
    contentPane.setLayout(null);
    this.setSize(new Dimension(700, 700));
    this.setTitle("PROPOSAL DEMOSTRATION");
    label1.setBounds(new Rectangle(13, 77, 259, 30));
    label1.setFont(new java.awt.Font("Dialog", 0, 18));
    label1.setText("REQUIRED TABLE NAME");
    label1.setForeground(Color.yellow);
    textField1.setBackground(SystemColor.white);
    textField1.setBounds(new Rectangle(11, 113, 259, 32));
    textField1.setFont(new java.awt.Font("Dialog", 0, 14));
    textField1.setText("");
    label3.setBounds(new Rectangle(13, 162, 258, 33));
    label3.setFont(new java.awt.Font("Dialog", 0, 18));
    label3.setForeground(Color.yellow);
    label3.setText("ATTRIBUTE    ATTRIBUTE");
    textField2.setBackground(SystemColor.white);
    textField2.setBounds(new Rectangle(10, 221, 129, 34));
    textField2.setFont(new java.awt.Font("Dialog", 0, 14));
    textField2.setText("");
    button1.setBackground(Color.lightGray);
    button1.setBounds(new Rectangle(11, 427, 152, 49));
    button1.setFont(new java.awt.Font("Dialog", 1, 12));
    button1.setLabel("NEW TABLE");
    button1.addActionListener
    (
        new java.awt.event.ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                button1_actionPerformed(e);
            }
        }
    );
    button2.setBackground(Color.lightGray);
    button2.setBounds(new Rectangle(12, 482, 153, 48));
    button2.setFont(new java.awt.Font("Dialog", 1, 12));
    button2.setLabel("NEXT ATTRIBUTE");
    button2.addActionListener
    (
        new java.awt.event.ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                button2_actionPerformed(e);
            }
        }
    );
}

```

```

    }
    }
};
menuBar2.setFont(new java.awt.Font("Dialog", 1, 12));
button3.setBackground(Color.lightGray);
button3.setBounds(new Rectangle(183, 482, 87, 47));
button3.setFont(new java.awt.Font("Dialog", 1, 12));
button3.setLabel("NEXT >>");
button3.addActionListener
(
    new java.awt.event.ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            button3_actionPerformed(e);
        }
    }
);
label4.setText("TABLE      ATTRIBUTE      TYPE");
label4.setForeground(Color.yellow);
label4.setFont(new java.awt.Font("Dialog", 0, 18));
label4.setBounds(new Rectangle(305, 77, 303, 33));
label2.setBounds(new Rectangle(14, 193, 270, 28));
label2.setFont(new java.awt.Font("Dialog", 0, 18));
label2.setForeground(Color.yellow);
label2.setText("NAME      TYPE LENGTH");
button4.setBackground(Color.lightGray);
button4.setBounds(new Rectangle(183, 426, 86, 48));
button4.setFont(new java.awt.Font("Dialog", 1, 12));
button4.setLabel("SUBMIT");
button4.addActionListener
(
    new java.awt.event.ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            button4_actionPerformed(e);
        }
    }
);
textArea1.setBackground(SystemColor.white);
textArea1.setBounds(new Rectangle(303, 111, 296, 416));
textArea1.setFont(new java.awt.Font("Dialog", 0, 16));
textField3.setBackground(SystemColor.white);
textField3.setBounds(new Rectangle(142, 221, 127, 34));
textField3.setFont(new java.awt.Font("Dialog", 0, 14));
textField3.setText("");
textField4.setBackground(SystemColor.white);
textField4.setBounds(new Rectangle(11, 313, 260, 81));
textField4.setFont(new java.awt.Font("Dialog", 0, 14));
textField4.setText("");
panel1.setBackground(SystemColor.desktop);
panel1.setBounds(new Rectangle(11, 313, 260, 81));
panel1.setLayout(null);
box1.setBounds(new Rectangle(12, 280, 260, 39));
jRadioButton1.setFont(new java.awt.Font("Dialog", 1, 16));
jRadioButton1.setForeground(Color.yellow);
jRadioButton1.setBackground(SystemColor.desktop);
jRadioButton1.setVisible(true);
jRadioButton1.setText(" KEY      ");
jRadioButton1.setSelected(false);
jRadioButton1.addActionListener
(
    new java.awt.event.ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            jRadioButton1_actionPerformed(e);
        }
    }
);

```



```

    }
  );
  jButton2.setFont(new java.awt.Font("Dialog", 1, 16));
  jButton2.setForeground(Color.yellow);
  jButton2.setBackground(SystemColor.desktop);
  jButton2.setVisible(true);
  jButton2.setText(" NON KEY");
  jButton2.setSelected(false);
  jButton2.addActionListener
  (
    new java.awt.event.ActionListener()
    {
      public void actionPerformed(ActionEvent e)
      {
        jButton2_actionPerformed(e);
      }
    }
  );

  label5.setBounds(new Rectangle(12, 320, 260, 30));
  label5.setFont(new java.awt.Font("Dialog", 1, 16));
  label5.setForeground(Color.yellow);
  label5.setText("INPUT USER QUERY FROM FILE");
  textField4.setBackground(SystemColor.white);
  textField4.setBounds(new Rectangle(11, 360, 260, 30));
  textField4.setFont(new java.awt.Font("Dialog", 0, 14));
  textField4.setText("");
  textField5.setBackground(SystemColor.white);
  textField5.setBounds(new Rectangle(11, 560, 160, 30));
  textField5.setFont(new java.awt.Font("Dialog", 0, 14));
  textField5.setText("");
  textField6.setBackground(SystemColor.white);
  textField6.setBounds(new Rectangle(180, 560, 160, 30));
  textField6.setFont(new java.awt.Font("Dialog", 0, 14));
  textField6.setText("");
  contentPane.setBackground(SystemColor.desktop);
  contentPane.add(label1, null);
  contentPane.add(textField1, null);
  contentPane.add(button3, null);
  contentPane.add(label4, null);
  contentPane.add(title, null);
  contentPane.add(textAreal, null);
  contentPane.add(button4, null);
  contentPane.add(label3, null);
  contentPane.add(textField2, null);
  contentPane.add(textField3, null);
  contentPane.add(textField4, null);
  contentPane.add(textField5, null);
  contentPane.add(textField6, null);

  contentPane.add(label2, null);
  contentPane.add(label5, null);
  contentPane.add(panell, null);
  contentPane.add(button1, null);
  contentPane.add(button2, null);
  contentPane.add(box1, null);
  box1.add(jButton1, null);
  box1.add(jButton2, null);
}

//Overridden so we can exit when window is closed
protected void processWindowEvent(WindowEvent e)
{
  super.processWindowEvent(e);
  if (e.getID() == WindowEvent.WINDOW_CLOSING)
  {
    System.exit(0);
  }
}

void jButton1_actionPerformed(ActionEvent e)

```

```

{
}

void jButton2_actionPerformed(ActionEvent e)
{
}

void jButton3_actionPerformed(ActionEvent e)
{
    setVisible(false);

    //useless now
    //linkingGenerator lg=new linkingGenerator();

    String userid = textField5.getText();
    String password = textField6.getText();

    int count = 0;
    String tableName = new String();

    codeGenerator cg=new codeGenerator();

    while(true)
    {
        if ( count >= table.size() )
            break;

        tableName = (String) table.elementAt(count);

        for (int i=0; i<table.size(); i++)
        {

            if ( ((String) table.elementAt(i)).equals(tableName) )
            {
                count=i;

                attributeTemp.addElement( (String) attribute.elementAt(i));
                typeTemp.addElement( (String) type.elementAt(i));
                checkTemp.addElement( (String) check.elementAt(i));
            }

        }

        cg.creatingGenerator(tableName, attributeTemp, typeTemp, userid, password);
        conversion();
        cg.cleaningGenerator(tableName, attributeTemp, typeTemp, checkTemp, userid, password);
        cg.loadingGenerator(tableName, attributeTemp, typeTemp, userid, password);

        //cleaning the temp buffer
        attributeTemp.removeAllElements();
        typeTemp.removeAllElements();
        checkTemp.removeAllElements();

        count++;
    }

    //add new and called
    sourceGenerator sg=new sourceGenerator(table, attribute, type);
}

/*****
When press the button 4 for submit
*****/
void jButton4_actionPerformed(ActionEvent e)
{
    String s4 = textField4.getText();

    //restore the text 4 field to null

```

```

textField4.setText("");

//checking from file to read the rule or input by manual
if (!s4.equals(""))
{
    JOptionPane.showMessageDialog(null, "Read from file "+s4+"\nPress OK will be continue");

    //get input from a file
    //prompt for a file name and return it
    String fileName = s4;

    //save the lines in the file into a vector
    inputFromFile(fileName);

    //display the file
    for (int i=0; i<table.size(); i++)
    {
        textArea1.append( (String) table.elementAt(i)+"\t\t"+
                          (String) attribute.elementAt(i)+"\t\t"+
                          (String) type.elementAt(i)+"\n");
    }
}
else if (!file.equals(""))
{
    JOptionPane.showMessageDialog(null, "Read from file "+file+"\nPress OK will be continue");

    //get input from a file
    //prompt for a file name and return it
    String fileName = file;

    //save the lines in the file into a vector
    inputFromFile(fileName);

    //display the file
    for (int i=0; i<table.size(); i++)
    {
        textArea1.append( (String) table.elementAt(i)+"\t\t"+
                          (String) attribute.elementAt(i)+"\t\t"+
                          (String) type.elementAt(i)+"\n");
    }
}
else
{
    String s1 = textField1.getText();
    String s2 = textField2.getText();
    String s3 = textField3.getText();
    textArea1.append(s1+"\t"+s2+"\t"+s3+"\n");
    table.addElement(s1);
    attribute.addElement(s2);
    type.addElement(s3);
    if (jRadioButton1.isSelected())
        check.addElement("key");
    else if (jRadioButton2.isSelected())
        check.addElement("nonKey");
    else
        check.addElement("null");

    jRadioButton1.setSelected(false);
    jRadioButton2.setSelected(false);
}
}

/*****
bottom for set tasble and variable name
*****/
void button1_actionPerformed(ActionEvent e)
{
    textField3.setText("");
}

```

```

    textField2.setText("");
    textField1.setText("");
}

/*****
  botton for set tasble and variable name
*****/
void button2_actionPerformed(ActionEvent e)
{
    textField3.setText("");
    textField2.setText("");
}

/*****
inputFromFile reads command from a file, insert each command
line into a vector, and returns data in this vector
*****/
private void inputFromFile(String fileName)
{
    LineNumberReader lineNumberReader;

    Vector inputVector = new Vector();
    String a_line = new String("");

    boolean more_line = true;

    try
    {
        FileReader fileReader = new FileReader(fileName);
        lineNumberReader = new LineNumberReader(fileReader);

        do
        {
            //reads 1 line at a time
            a_line = lineNumberReader.readLine().trim();

            if (a_line.length() == 0)
                continue;
            else
                //save this line as a element in the vector
                inputVector.addElement(a_line);
        }

        while (more_line);
        lineNumberReader.close();
    }
    catch (NullPointerException npe)
    {
        //if reach the EndOfFile flag
        if (inputVector.size() == 0)
            //the input file is empty
            //return ""
            inputVector.addElement("");

        more_line = false;
    }
    catch (IOException e)
    {
        //for other file I/O errors
        file= JOptionPane.showInputDialog("No such file\nPlease retype the file name");

        inputVector.removeAllElements();
    }

    //Read them into Vector
    String inString=new String();
    for (int i=0; i<inputVector.size(); i++)
    {
        inString= (String) inputVector.elementAt(i);
    }
}

```



```

import javax.swing.*;
import java.util.*;
import java.io.*;

public class sourceGenerator
{
    boolean packFrame = false;

    //Construct the application
    public sourceGenerator(Vector targetTable, Vector targetAttribute, Vector targetType)
    {
        Frame4 frame = new Frame4(targetTable, targetAttribute, targetType);

        //Validate frames that have preset sizes
        //Pack frames that have useful preferred size info, e.g. from their layout
        if (packFrame)
        {
            frame.pack();
        }
        else
        {
            frame.validate();
        }

        //Center the window
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        Dimension frameSize = frame.getSize();

        if (frameSize.height > screenSize.height)
        {
            frameSize.height = screenSize.height;
        }
        if (frameSize.width > screenSize.width)
        {
            frameSize.width = screenSize.width;
        }

        frame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height - frameSize.height)
/ 2);
        frame.setVisible(true);
    }
}

```

(6) Frame4.java

```

//Source Generator
import java.awt.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import java.io.*;

public class Frame4 extends JFrame
{
    JPanel contentPane;
    JLabel jLabel1 = new JLabel();
    JLabel jLabel2 = new JLabel();
    JTextField jTextField1 = new JTextField();
    JTextField jTextField2 = new JTextField();
    JTextField jTextField3 = new JTextField();
    JTextField jTextField4 = new JTextField();
    JButton jButton1 = new JButton();
    JButton jButton2 = new JButton();
    JButton jButton3 = new JButton();
    JButton jButton4 = new JButton();
    JTextArea jTextArea1 = new JTextArea();
    JLabel jLabel3 = new JLabel();
}

```

```

JLabel jLabel4 = new JLabel();
JTextField jTextField5 = new JTextField();
JLabel jLabel5 = new JLabel();
JButton jButton5 = new JButton();
JButton jButton6 = new JButton();
JButton jButton7 = new JButton();
JButton jButton8 = new JButton();
Vector targetTable = new Vector();
Vector targetAttribute = new Vector();
Vector targetType = new Vector();
Vector sourceDatabase = new Vector();
Vector sourceTable = new Vector();
Vector sourceAttribute = new Vector();
Vector sourceType = new Vector();
String file="";

//Construct the frame
public Frame4(Vector targetTable, Vector targetAttribute, Vector targetType)
{

    this.targetTable=targetTable;
    this.targetAttribute=targetAttribute;
    this.targetType=targetType;

    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
    try
    {
        jbInit();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}

//Component initialization
private void jbInit() throws Exception
{
    jLabel1.setFont(new java.awt.Font("Dialog", 1, 22));
    jLabel1.setForeground(Color.yellow);
    jLabel1.setText("WIG SOURCE TABLE GENERATOR");
    jLabel1.setBounds(new Rectangle(126, 16, 386, 46));
    contentPane = (JPanel) this.getContentPane();
    contentPane.setLayout(null);
    this.setSize(new Dimension(700, 700));
    this.setTitle("WIG SOURCE TABLE GENERATOR");
    jLabel2.setFont(new java.awt.Font("Dialog", 1, 16));
    jLabel2.setForeground(Color.yellow);
    jLabel2.setText("DATABASE NAME      TABLE NAME      ATTRIBUTE NAME      DATA TYPE");
    jLabel2.setBounds(new Rectangle(22, 87, 572, 17));
    jTextField1.setBounds(new Rectangle(22, 114, 142, 29));
    jTextField2.setBounds(new Rectangle(175, 114, 135, 28));
    jTextField3.setBounds(new Rectangle(325, 114, 128, 28));
    jTextField4.setBounds(new Rectangle(466, 115, 124, 28));
    jButton1.setFont(new java.awt.Font("Dialog", 1, 10));
    jButton1.setText("DATABASE REDO");
    jButton1.setBounds(new Rectangle(22, 155, 138, 34));
    jButton1.addActionListener(new java.awt.event.ActionListener() {

        public void actionPerformed(ActionEvent e) {
            jButton1_actionPerformed(e);
        }
    });
    jButton2.setFont(new java.awt.Font("Dialog", 1, 10));
    jButton2.setText("TABLE REDO");
    jButton2.setBounds(new Rectangle(175, 155, 134, 33));
    jButton2.addActionListener(new java.awt.event.ActionListener() {

        public void actionPerformed(ActionEvent e) {
            jButton2_actionPerformed(e);
        }
    });
}

```

```

});
jButton3.setFont(new java.awt.Font("Dialog", 1, 10));
jButton3.setText("ATTRIBUTE REDO");
jButton3.setBounds(new Rectangle(326, 156, 128, 31));
jButton3.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        jButton3_actionPerformed(e);
    }
});
jButton4.setText("SUBMIT");
jButton4.setBounds(new Rectangle(468, 155, 124, 30));
jButton4.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        jButton4_actionPerformed(e);
    }
});
contentPane.setBackground(SystemColor.desktop);
jTextAreal.setBounds(new Rectangle(25, 203, 568, 118));
jLabel3.setFont(new java.awt.Font("Dialog", 1, 16));
jLabel3.setForeground(Color.yellow);
jLabel3.setText("INPUT FROM FROM FILE");
jLabel3.setBounds(new Rectangle(30, 331, 416, 34));
jLabel4.setText("jLabel4");
jLabel4.setBounds(new Rectangle(719, 376, 41, 17));
jTextField5.setBounds(new Rectangle(457, 331, 134, 31));
jTextField5.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        jTextField5_actionPerformed(e);
    }
});
jLabel5.setFont(new java.awt.Font("Dialog", 1, 16));
jLabel5.setForeground(Color.yellow);
jLabel5.setText("EXTRACT SOURCE AUTOMATICALLY");
jLabel5.setBounds(new Rectangle(30, 383, 442, 32));
jButton5.setText("EXTRACTING");
jButton5.setBounds(new Rectangle(457, 385, 134, 31));
jButton5.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        jButton5_actionPerformed(e);
    }
});
jButton6.setText("Link<<");
jButton6.setBounds(new Rectangle(70, 433, 128, 29));
jButton6.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        jButton6_actionPerformed(e);
    }
});
jButton7.setText("HELP");
jButton7.setBounds(new Rectangle(254, 434, 123, 29));
jButton7.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        jButton7_actionPerformed(e);
    }
});
jButton8.setText("NEXT >>");
jButton8.setBounds(new Rectangle(427, 434, 122, 30));
jButton8.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        jButton8_actionPerformed(e);
    }
});
contentPane.add(jLabel2, null);
contentPane.add(jTextField1, null);

```



```

    contentPane.add(jTextArea1, null);
    contentPane.add(jLabel4, null);
    contentPane.add(jLabel3, null);
    contentPane.add(jTextField5, null);
    contentPane.add(jLabel1, null);
    contentPane.add(jLabel5, null);
    contentPane.add(jButton5, null);
    contentPane.add(jButton6, null);
    contentPane.add(jButton7, null);
    contentPane.add(jButton8, null);
    contentPane.add(jButton1, null);
    contentPane.add(jButton2, null);
    contentPane.add(jButton3, null);
    contentPane.add(jButton4, null);
    contentPane.add(jTextField2, null);
    contentPane.add(jTextField3, null);
    contentPane.add(jTextField4, null);
}

//Overridden so we can when window is closed
protected void processWindowEvent(WindowEvent e) {
    super.processWindowEvent(e);
    if (e.getID() == WindowEvent.WINDOW_CLOSING) {
        System.exit(0);
    }
}

void jButton1_actionPerformed(ActionEvent e)
{
    jTextField1.setText("");
    jTextField2.setText("");
    jTextField3.setText("");
    jTextField4.setText("");
}

void jButton2_actionPerformed(ActionEvent e)
{
    jTextField2.setText("");
    jTextField3.setText("");
    jTextField4.setText("");
}

void jButton3_actionPerformed(ActionEvent e)
{
    jTextField3.setText("");
    jTextField4.setText("");
}

void jButton4_actionPerformed(ActionEvent e)
{
    sourceDatabase.addElement(jTextField1.getText());
    sourceTable.addElement(jTextField2.getText());
    sourceAttribute.addElement(jTextField3.getText());
    sourceType.addElement(jTextField4.getText());

    jTextArea1.append( jTextField2.getText()+"\t"+
        jTextField2.getText()+"\t"+
        jTextField3.getText()+"\t"+
        jTextField4.getText()+"\n");
}

void jButton5_actionPerformed(ActionEvent e)
{
    //JDBC should insert here
    Vector name = new Vector();
    Vector attribute = new Vector();
    Vector type = new Vector();

```

```

SDBconnector d = new SDBconnector();

name = d.getTable_name();

for (int i=0; i<name.size(); i++)
{
    attribute = d.getAttribute((String)name.elementAt(i));
    type      = d.getType((String)name.elementAt(i));

    for (int j=0; j<attribute.size(); j++)
    {
        sourceTable.addElement((String)name.elementAt(i));
        sourceAttribute.addElement((String)attribute.elementAt(j));
        sourceType.addElement((String) type.elementAt(j));
    }
}

for (int i=0; i<sourceAttribute.size(); i++)
{
    jTextArea1.append("checking\t"+
        sourceTable.elementAt(i)+"\t"+
        sourceAttribute.elementAt(i)+"\t"+
        sourceType.elementAt(i)+"\n");
}

SDBconnection dd = new SDBconnection();

name = dd.getTable_name();

for (int i=0; i<name.size(); i++)
{
    attribute = dd.getAttribute((String)name.elementAt(i));
    type      = dd.getType((String)name.elementAt(i));

    for (int j=0; j<attribute.size(); j++)
    {
        sourceTable.addElement((String)name.elementAt(i));
        sourceAttribute.addElement((String)attribute.elementAt(j));
        sourceType.addElement((String) type.elementAt(j));
    }
}

for (int i=0; i<sourceAttribute.size(); i++)
{
    jTextArea1.append("savings\t"+
        sourceTable.elementAt(i)+"\t"+
        sourceAttribute.elementAt(i)+"\t"+
        sourceType.elementAt(i)+"\n");
}
}

void jTextField5_actionPerformed(ActionEvent e)
{
    //read from file
    String s4 = jTextField5.getText();

    //restore the text 4 field to null
    jTextField5.setText("");

    //checking from file to read the rule or input by manual
    if (!s4.equals(""))
    {
        JOptionPane.showMessageDialog(null, "Read from file "+s4+"\nPress OK will be continue");

        //get input from a file
        //prompt for a file name and return it
        String fileName = s4;

        //save the lines in the file into a vector
        inputFromFile(fileName);
    }
}

```

```

//display the file
for (int i=0; i<sourceTable.size(); i++)
{
    JTextArea1.append( sourceDatabase.elementAt(i)+"\t"+
                      sourceTable.elementAt(i)+"\t"+
                      sourceAttribute.elementAt(i)+"\t"+
                      sourceType.elementAt(i)+"\n");
}
}
else if (!file.equals(""))
{
    JOptionPane.showMessageDialog(null, "Read from file "+file+"\nPress OK will be continue");

//get input from a file
//prompt for a file name and return it
String fileName = file;

//save the lines in the file into a vector
inputFromFile(fileName);

//display the file
for (int i=0; i<sourceTable.size(); i++)
{
    JTextArea1.append( sourceDatabase.elementAt(i)+"\t"+
                      sourceTable.elementAt(i)+"\t"+
                      sourceAttribute.elementAt(i)+"\t"+
                      sourceType.elementAt(i)+"\n");
}
}
}

void jButton6_actionPerformed(ActionEvent e)
{
    setVisible(false);
    linkingGenerator lg=new linkingGenerator(targetTable, targetAttribute, targetType,
sourceDatabase, sourceTable, sourceAttribute, sourceType);
}

void jButton7_actionPerformed(ActionEvent e)
{
    JOptionPane.showMessageDialog(null, "Read from file \n or read by hand \n or using JDBC");
}

void jButton8_actionPerformed(ActionEvent e)
{
    setVisible(false);
    ruleGenerator rg = new ruleGenerator(targetTable, targetAttribute, targetType, sourceDatabase,
sourceTable, sourceAttribute, sourceType, "", "");
}

/*****
inputFromFile reads command from a file, insert each command
line into a vector, and returns data in this vector
*****/
private void inputFromFile(String fileName)
{
    LineNumberReader lineNumberReader;

    Vector inputVector = new Vector();
    String a_line = new String("");

    boolean more_line = true;

    try
    {
        FileReader fileReader = new FileReader(fileName);
        lineNumberReader = new LineNumberReader(fileReader);

    do
    {

```

```

//reads 1 line at a time
a_line = lineNumberReader.readLine().trim();

if (a_line.length() == 0)
    continue;
else
    //save this line as a element in the vector
    inputVector.addElement(a_line);
}

while (more_line);
    lineNumberReader.close();
}
catch (NullPointerException npe)
{
    //if reach the EndOfFile flag
    if (inputVector.size() ==0)
        //the input file is empty
        //return ""
        inputVector.addElement("");

    more_line = false;
}
catch (IOException e)
{
    //for other file I/O errors
    file= JOptionPane.showInputDialog("No such file\nPlease retype the file name");

    inputVector.removeAllElements();
}

//Read them into Vector
String inString=new String();
for (int i=0; i<inputVector.size(); i++)
{
    inString= (String) inputVector.elementAt(i);
    StringTokenizer st = new StringTokenizer(inString, ",");

    sourceDatabase.addElement( st.nextToken() );
    sourceTable.addElement( st.nextToken() );
    sourceAttribute.addElement( st.nextToken() );
    sourceType.addElement( st.nextToken() );

    if(st.hasMoreTokens())
    {
        sourceDatabase.removeAllElements();
        sourceTable.removeAllElements();
        sourceAttribute.removeAllElements();
        sourceType.removeAllElements();

        JOptionPane.showMessageDialog(null, "The contents of file is wrong!\nPlease check your file");
        inputVector.removeAllElements();
    }
}
}
}
}

```

(7) SDBconnection.java

```

import java.sql.*;

import java.util.Vector;
import java.io.*;

public class SDBconnection
{
    /*****

```

```

get all of the tables name from the local machine
*****/
public Vector getTableName()
{
    //select tables name and put it into output Vector
    String input="select * from tab";

    Vector output = new Vector();

    try
    {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        String url = "jdbc:odbc:savings";
        Connection con = DriverManager.getConnection(url, "demo", "demo");
        Statement statement = con.createStatement();
        ResultSet rs= statement.executeQuery(input);
        ResultSetMetaData rsmd=rs.getMetaData();
        while( rs.next())
            output.addElement(rs.getString(1));
    }
    catch (java.lang.Exception ex)
    {
        System.err.println("error on query");
        ex.printStackTrace();
    }

    return output;
}

/*****
get all Attribute of one tables from the local machine
*****/
public Vector getAttribute(String tableName)
{
    //select tables name and put it into output Vector
    String input="select * from "+tableName;

    //define some variables for attribute
    Vector attribute = new Vector();

    try
    {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        String url = "jdbc:odbc:savings";
        Connection con = DriverManager.getConnection(url, "demo", "demo");
        Statement statement = con.createStatement();
        ResultSet rs= statement.executeQuery(input);
        ResultSetMetaData rsmd=rs.getMetaData();

        int columnNumber=rsmd.getColumnCount();

        for(int n=1; n<=columnNumber; n++)
            attribute.addElement( rsmd.getColumnLabel(n) );
    }
    catch (java.lang.Exception ex)
    {
        System.err.println("error on getting attributes");
        ex.printStackTrace();
    }

    return attribute;
}

/*****
get all types of one tables from the local machine
*****/
public Vector getType(String tableName)
{
    //select tables name and put it into output Vector
    String input="select * from "+tableName;

```

```

Vector type =new Vector();

try
{
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
String url = "jdbc:odbc:savings";
Connection con = DriverManager.getConnection(url, "demo", "demo");
Statement statement = con.createStatement();
ResultSet rs= statement.executeQuery(input);
ResultSetMetaData rsmd=rs.getMetaData();

int columnNumber=rsmd.getColumnCount();

for(int n=1; n<=columnNumber; n++)
type.addElement( rsmd.getColumnTypeName(n) );
}
catch (java.lang.Exception ex)
{
System.err.println("error on getting attributes");
ex.printStackTrace();
}

return type;
}
}

import java.sql.*;
import java.util.Vector;
import java.io.*;

public class SDBconnector
{
/*****
get all of the tables name from the local machine
*****/
public Vector getTableName()
{
//select tables name and put it into output Vector
String input="select * from tab";

Vector output = new Vector();

try
{
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
String url = "jdbc:odbc:checking";
Connection con = DriverManager.getConnection(url, "scott", "tiger");
Statement statement = con.createStatement();
ResultSet rs= statement.executeQuery(input);
ResultSetMetaData rsmd=rs.getMetaData();
while( rs.next())
output.addElement(rs.getString(1));
}
catch (java.lang.Exception ex)
{
System.err.println("error on query");
ex.printStackTrace();
}

return output;
}

/*****
get all Attribute of one tables from the local machine
*****/
public Vector getAttribute(String tableName)
{
//select tables name and put it into output Vector

```

```

String input="select * from "+tableName;

//define some variables for attribute
Vector attribute = new Vector();

try
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String url = "jdbc:odbc:checking";
    Connection con = DriverManager.getConnection(url, "scott", "tiger");
    Statement statement = con.createStatement();
    ResultSet rs= statement.executeQuery(input);
    ResultSetMetaData rsmd=rs.getMetaData();

    int columnNumber=rsmd.getColumnCount();

    for(int n=1; n<=columnNumber; n++)
        attribute.addElement( rsmd.getColumnLabel(n) );
}
catch (java.lang.Exception ex)
{
    System.err.println("error on getting attributes");
    ex.printStackTrace();
}

return attribute;
}

/*****
get all types of one tables from the local machine
*****/
public Vector getType(String tableName)
{
    //select tables name and put it into output Vector
    String input="select * from "+tableName;

    Vector type =new Vector();

    try
    {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        String url = "jdbc:odbc:checking";
        Connection con = DriverManager.getConnection(url, "scott", "tiger");
        Statement statement = con.createStatement();
        ResultSet rs= statement.executeQuery(input);
        ResultSetMetaData rsmd=rs.getMetaData();

        int columnNumber=rsmd.getColumnCount();

        for(int n=1; n<=columnNumber; n++)
            type.addElement( rsmd.getColumnTypeName(n) );
    }
    catch (java.lang.Exception ex)
    {
        System.err.println("error on getting attributes");
        ex.printStackTrace();
    }

    return type;
}
}

```

(8) linkingGenerator.java

```

import javax.swing.UIManager;
import java.util.*;

public class linkingGenerator

```

```

{
    boolean packFrame = false;

    //Construct the application
    public linkingGenerator(Vector targetTable,
                           Vector targetAttribute,
                           Vector targetType,
                           Vector sourceDatabase,
                           Vector sourceTable,
                           Vector sourceAttribute,
                           Vector sourceType)
    {
        Frame6 frame = new Frame6( targetTable,
                                    targetAttribute,
                                    targetType,
                                    sourceDatabase,
                                    sourceTable,
                                    sourceAttribute,
                                    sourceType);

        //Validate frames that have preset sizes
        //Pack frames that have useful preferred size info, e.g. from their layout
        if (packFrame)
        {
            frame.pack();
        }
        else
        {
            frame.validate();
        }
        frame.setVisible(true);
    }
}

/*
//Main method
public static void main(String[] args)
{
    try
    {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    new linkingGenerator();
}
*/
}

```

(9) Frame6.java

```

import java.awt.*;

import java.awt.event.*;
import javax.swing.*;
import java.io.*;
import java.util.*;

public class Frame6 extends JFrame
{
    JPanel contentPane;
    JLabel jLabel1 = new JLabel();
    JLabel jLabel2 = new JLabel();
    JTextField jTextField1 = new JTextField();
    JLabel jLabel3 = new JLabel();
    JTextField jTextField2 = new JTextField();
    JLabel jLabel4 = new JLabel();
    JPasswordField jPasswordField1 = new JPasswordField();
    JButton jButton1 = new JButton();
}

```



```

JButton jButton2 = new JButton();
JLabel jLabel5 = new JLabel();
JTextField jTextField3 = new JTextField();
JLabel jLabel6 = new JLabel();
JTextField jTextField4 = new JTextField();
JLabel jLabel7 = new JLabel();
JButton jButton3 = new JButton();
JPasswordField jPasswordField2 = new JPasswordField();
JButton jButton4 = new JButton();
String sourceName = "", sourceUID = "", sourcePWD = "", targetName = "", targetUID = "", targetPWD
= "";
Vector targetTable =new Vector();
Vector targetAttribute=new Vector();
Vector targetType=new Vector();
Vector sourceDatabase =new Vector();
Vector sourceTable =new Vector();
Vector sourceAttribute =new Vector();
Vector sourceType=new Vector();

//Construct the frame
public Frame6(Vector targetTable,
              Vector targetAttribute,
              Vector targetType,
              Vector sourceDatabase,
              Vector sourceTable,
              Vector sourceAttribute,
              Vector sourceType)
{
    this.targetTable =targetTable;
    this.targetAttribute=targetAttribute;
    this.targetType=targetType;
    this.sourceDatabase=sourceDatabase;
    this.sourceTable=sourceTable;
    this.sourceAttribute=sourceAttribute;
    this.sourceType=sourceType;

    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
    try {
        jbInit();
    }
    catch(Exception e) {
        e.printStackTrace();
    }
}

//Component initialization
private void jbInit() throws Exception {
    jLabel1.setFont(new java.awt.Font("Dialog", 1, 24));
    jLabel1.setForeground(Color.yellow);
    jLabel1.setText("SOURCE/TARGET LINKING GENERATOR");
    jLabel1.setBounds(new Rectangle(56, 11, 536, 58));
    contentPane = (JPanel) this.getContentPane();
    contentPane.setLayout(null);
    this.setSize(new Dimension(656, 702));
    this.setTitle("Frame Title");
    contentPane.setBackground(SystemColor.desktop);
    jLabel2.setForeground(Color.yellow);
    jLabel2.setText("SOURCE DATABASE NAME");
    jLabel2.setBounds(new Rectangle(50, 82, 191, 40));
    jTextField1.setBounds(new Rectangle(400, 89, 240, 33));
    jLabel3.setForeground(Color.yellow);
    jLabel3.setText("SOURCE DATABASE USER ID");
    jLabel3.setBounds(new Rectangle(49, 144, 202, 34));
    jTextField2.setBounds(new Rectangle(400, 142, 240, 33));
    jLabel4.setForeground(Color.yellow);
    jLabel4.setText("SOURCE DATABASE PASSWORD");
    jLabel4.setBounds(new Rectangle(50, 201, 197, 39));
    jPasswordField1.setBounds(new Rectangle(400, 195, 240, 33));
    jButton1.setBackground(Color.lightGray);
    jButton1.setText("RESET");
    jButton1.setBounds(new Rectangle(48, 248, 298, 46));
}

```

```

jButton1.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        jButton1_actionPerformed(e);
    }
});
jButton2.setText(" EXIT ");
jButton2.setBounds(new Rectangle(371, 247, 195, 48));
jButton2.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        jButton2_actionPerformed(e);
    }
});
jLabel5.setForeground(Color.yellow);
jLabel5.setText("DATA WAREHOUSE NAME");
jLabel5.setBounds(new Rectangle(54, 330, 223, 42));
jTextField3.setForeground(Color.red);
jTextField3.setBounds(new Rectangle(400, 332, 240, 33));
jLabel6.setForeground(Color.yellow);
jLabel6.setText("DATA WAREHOUSE USER ID");
jLabel6.setBounds(new Rectangle(55, 382, 220, 44));
jTextField4.setForeground(Color.red);
jTextField4.setBounds(new Rectangle(400, 386, 240, 33));
jLabel7.setForeground(Color.yellow);
jLabel7.setText("DATA WAREHOUSE PASSWORD");
jLabel7.setBounds(new Rectangle(56, 434, 231, 40));
jButton3.setText("SUBMIT");
jButton3.setBounds(new Rectangle(56, 507, 277, 60));
jButton3.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        jButton3_actionPerformed(e);
    }
});
jPasswordField2.setForeground(Color.red);
jPasswordField2.setBounds(new Rectangle(400, 436, 240, 33));
jButton4.setText(" NEXT>> ");
jButton4.setBounds(new Rectangle(377, 506, 184, 61));
jButton4.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        jButton4_actionPerformed(e);
    }
});
contentPane.add(jLabel11, null);
contentPane.add(jLabel13, null);
contentPane.add(jLabel14, null);
contentPane.add(jLabel15, null);
contentPane.add(jLabel16, null);
contentPane.add(jLabel17, null);
contentPane.add(jLabel12, null);
contentPane.add(jButton1, null);
contentPane.add(jButton3, null);
contentPane.add(jButton2, null);
contentPane.add(jTextField1, null);
contentPane.add(jTextField2, null);
contentPane.add(jPasswordField1, null);
contentPane.add(jTextField3, null);
contentPane.add(jTextField4, null);
contentPane.add(jPasswordField2, null);
contentPane.add(jButton4, null);

resize(700,700);
show();
}

//Overridden so we can exit when window is closed
protected void processWindowEvent(WindowEvent e)
{
    super.processWindowEvent(e);
}

```

```

    if (e.getID() == WindowEvent.WINDOW_CLOSING)
    {
        System.exit(0);
    }
}

void jButton1_actionPerformed(ActionEvent e)
{
    jTextField1.setText("");
    jTextField2.setText("");
    jPasswordField1.setText("");
}

void jButton2_actionPerformed(ActionEvent e)
{
    System.exit(0);
}

void jButton4_actionPerformed(ActionEvent e)
{
    setVisible(false);
    ruleGenerator rg = new ruleGenerator(targetTable, targetAttribute, targetType, sourceDatabase,
sourceTable, sourceAttribute, sourceType, jTextField4.getText(), jPasswordField2.getText());
}

void jButton3_actionPerformed(ActionEvent e)
{
    int targetLength=targetTable.size();
    String targetTableArray[] =new String[targetLength];
    String targetAttributeArray[]=new String[targetLength];
    String targetTypeArray[] =new String[targetLength];

    int sourceLength=sourceTable.size();
    String sourceDatabaseArray[] =new String[sourceLength];
    String sourceTableArray[] =new String[sourceLength];
    String sourceAttributeArray[]=new String[sourceLength];
    String sourceTypeArray[] =new String[sourceLength];

    for (int i=0; i<targetLength; i++)
    {
        targetTableArray[i] = (String) targetTable.elementAt(i);
        targetAttributeArray[i]= (String) targetAttribute.elementAt(i);
        targetTypeArray[i] = (String) targetType.elementAt(i);
    }

    for (int i=0; i<sourceLength; i++)
    {
        sourceDatabaseArray[i] = (String) sourceDatabase.elementAt(i);
        sourceTableArray[i] = (String) sourceTable.elementAt(i);
        sourceAttributeArray[i]= (String) sourceAttribute.elementAt(i);
        sourceTypeArray[i] = (String) sourceType.elementAt(i);
    }

    sourceName= jTextField1.getText();
    sourceUID = jTextField2.getText();
    sourcePWD = jPasswordField1.getText();
    targetName= jTextField3.getText();
    targetUID = jTextField4.getText();
    targetPWD = jPasswordField2.getText();
    jTextField1.setText("");
    jTextField2.setText("");
    jPasswordField1.setText("");

    //Defining the function name for the hasPropriety $ff
    String functionName = sourceName+"Linking";

    //Defining the output file name. For pro.C,
    //it is file name plus dot pc
    String template = sourceName+"Linking.pc";

    /***Component setup***/

```

```

//Determining if the 'Create' is already in the template
//if this output file is aviable, add one more creating
//component into it, else, add main component and creating
//component into the template
append("blankComponent", "linkingComponent", template);

singleInsert(targetUID, "$targetUserID", template);
singleInsert(targetPWD, "$targetPassword", template);
singleInsert(sourceUID, "$sourceUserID", template);
singleInsert(sourcePWD, "$sourcePassword", template);

doubleInsert(functionName, "$ff", template);

Vector currentSourceDatabase = sourceDatabase;
Vector currentSourceTable = sourceTable;

int k=0, currentSize=currentSourceDatabase.size();
while(currentSize>0 && k<currentSize )
{
    if ( !((String)currentSourceDatabase.elementAt(k)).equals(sourceName) )
    {

        currentSourceDatabase.removeElementAt(k);
        currentSourceTable.removeElementAt(k);
    }
    else
    {
        k++;
    }
    currentSize=currentSourceDatabase.size();
}

while(currentSourceTable.size(>0)
{
    //set current source name
    String currentTable = (String) currentSourceTable.elementAt(0);
    doubleInsert("EXEC SQL CREATE TABLE "+
                targetUID+"."+currentTable+
                " AS SELECT * FROM "+
                sourceUID+"."+currentTable+";" , "$dd", template);

    //delete the current information from input vector
    //so, keep the input vector is un_integrated
    while (currentSourceTable.indexOf(currentTable)!=-1)
    {
        int position=(int) currentSourceTable.indexOf(currentTable);
        currentSourceTable.removeElementAt(position);
    }
}

remove("$dd", template);

targetTable.removeAllElements();
targetAttribute.removeAllElements();
targetType.removeAllElements();
for (int i=0; i<targetLength; i++)
{
    targetTable.addElement(targetTableArray[i]);
    targetAttribute.addElement(targetAttributeArray[i]);
    targetType.addElement(targetTypeArray[i]);
}

sourceDatabase.removeAllElements();
sourceTable.removeAllElements();
sourceAttribute.removeAllElements();
sourceType.removeAllElements();
for (int i=0; i<sourceLength; i++)
{
    sourceDatabase.addElement(sourceDatabaseArray[i]);
    sourceTable.addElement(sourceTableArray[i]);
}

```

```

        sourceAttribute.addElement(sourceAttributeArray[i]);
        sourceType.addElement(sourceTypeArray[i]);
    }
}

public String getSourceName()
{
    return sourceName;
}

public String getSourceUID()
{
    return sourceUID;
}

public String getSourcePWD()
{
    return sourcePWD;
}

public String getTargetName()
{
    return targetName;
}

public String getTargetUID()
{
    return targetUID;
}

public String getTargetPWD()
{
    return targetPWD;
}

void singleInsert(String content, String mark, String fileName)
{
    //Creating object line reader to read line by line
    LineNumberReader lineNumberReader;

    //Defining the target vector to fill the generated code
    Vector fileVector = new Vector();

    //Seperating the elements of one line by token
    Vector lineVector = new Vector();

    //Some temp parameters
    String line = new String("");
    String newLine = new String("");
    boolean moreLine = true;
    boolean hasPropriety = true;

    try
    {
        //Open tha target file template
        FileReader fileReader = new FileReader(fileName);

        //Reading the file line by line
        lineNumberReader = new LineNumberReader(fileReader);

        //Loop to read every line from this file tempalte
        do
        {
            //reading one line from file to 'line'
            line = lineNumberReader.readLine().trim();

            //Empty line or not
            if (line.length() == 0)
            {
                //Add empty to target vector
                fileVector.addElement(" ");
            }
        }
    }
}

```

```

        //Going back to reading next line
        continue;
    }
    else
    {
        //Symbol is for the search is any property in this line
        hasProperty = false;

        //Tokenize the line element
        lineVector = tokenize(line);

        //Checking every line to see the mark is inside or not
        for (int i=0; i<lineVector.size();i++)
        {
            if (((String)lineVector.elementAt(i)).equals(mark))
            {
                //When hasProperty is true, there is a property in this line
                hasProperty=true;

                //Insert into target vector
                lineVector.setElementAt(content,i);

                //Forming the whole line after the insert
                newLine="";
                for (int j=0; j<lineVector.size(); j++)
                    newLine=newLine+" "+(String)lineVector.elementAt(j);

                //when have property, add the new line into file but not old line
                fileVector.addElement(newLine);

                break;
            }
            //End if
        }
        //End for
    }
    //End else

    //If no property in this line, add the origin line into target vector
    if (hasProperty==false)
        fileVector.addElement(line);

}while(moreLine);

lineNumberReader.close();
}
catch (NullPointerException npe)
{
    moreLine = false;
}
catch (IOException e)
{
}

update(fileVector, fileName);
}

public void update(Vector fileVector, String fileName)
{
    FileOutputStream outputFile;

    try
    {
        outputFile = new FileOutputStream(fileName);
        PrintStream output = new PrintStream(outputFile);

        for (int i=0; i<fileVector.size(); i++)
            output.println(fileVector.elementAt(i));

        output.close();
    }
}

```

```

catch (NullPointerException npe)
{
    System.out.println("IOException found by LiuYi at 3");
}
catch (IOException e)
{
    System.out.println("IOException found by LiuYi at 4");
}
}

public boolean search(String mark, String fileName)
{
    LineNumberReader lineNumberReader;
    Vector fileVector = new Vector();
    Vector lineVector = new Vector();
    String line = new String("");
    boolean moreLine = true, result = false;

    try
    {
        FileReader fileReader = new FileReader(fileName);
        lineNumberReader = new LineNumberReader(fileReader);

        do
        {
            line = lineNumberReader.readLine().trim();

            lineVector = tokenize(line);

            for (int i=0; i<lineVector.size(); i++)
            {
                if (((String)lineVector.elementAt(i)).equals(mark))
                {
                    result=true;
                    break;
                }
            }
        }while(moreLine);

        lineNumberReader.close();
    }
    catch (NullPointerException npe)
    {
        moreLine = false;
    }
    catch (IOException e)
    {
    }

    return result;
}

public void append(String sourceFile, String targetFile, String procFile)
{
    LineNumberReader lineNumberReader,lineReader;
    Vector targetFileVector = new Vector();
    boolean moreLine = true;
    String line = new String("");

    try
    {
        FileReader targetFileReader = new FileReader(targetFile);
        lineNumberReader = new LineNumberReader(targetFileReader);

        do
        {
            line = lineNumberReader.readLine().trim();

```

```

        targetFileVector.addElement(line);
    }while(moreLine);

}
catch (NullPointerException npe)
{
    moreLine = false;
}
catch (IOException e)
{
    System.out.println("IOException found by LiuYi at 1");
}

moreLine=true;

try
{

    FileReader sourceFileReader = new FileReader(sourceFile);
    lineNumberReader = new LineNumberReader(sourceFileReader);
    do
    {
        line = lineNumberReader.readLine().trim();
        targetFileVector.addElement(line);
    }while(moreLine);

}
catch (NullPointerException npe)
{
    moreLine = false;
}
catch (IOException e)
{

    System.out.println("IOException found by LiuYi at 2");
}

update(targetFileVector, procFile);
}

public Vector tokenize(String inString)
{
    Vector tokenVector = new Vector();
    String tokenString = new String();
    StringTokenizer st = new StringTokenizer(inString);

    while (st.hasMoreTokens())
    {
        tokenString = st.nextToken();
        tokenVector.addElement(tokenString);
    }
    return tokenVector;
}

void doubleInsert(String content, String mark, String fileName)
{
    //Creating object line reader to read line by line
    LineNumberReader lineNumberReader;

    //Defining the target vector to fill the generated code
    Vector fileVector = new Vector();

    //Separating the elements of one line by token
    Vector lineVector = new Vector();

    //Some temp parameters
    String line = new String("");
    String newLine = new String("");
    boolean moreLine = true;

```



```

try
{
//Open the target file template
FileReader fileReader = new FileReader(fileName);

//Reading the file line by line
lineNumberReader = new LineNumberReader(fileReader);

//Loop to read every line from this file template
do
{
//reading one line from file to 'line'
line = lineNumberReader.readLine().trim();

//Empty line or not
if (line.length() == 0)
{
//Add empty to target vector
fileVector.addElement(" ");

//Going back to reading next line
continue;
}
else
{
//Tokenize the line element
lineVector = tokenize(line);

//check it is declare or not. if it is declare, not do anything
if (dollarNumber(lineVector)==1)
{

//Checking every line to see the mark is inside or not
for (int i=0; i<lineVector.size(); i++)
//check it is the mark we want replace
if (((String)lineVector.elementAt(i)).equals(mark))
{

//Insert into target vector
lineVector.setElementAt(content,i);

//Forming the whole line after the insert
newLine="";

for (int j=0; j<lineVector.size(); j++)
newLine=newLine+" "+(String)lineVector.elementAt(j);

//add new line
fileVector.addElement(newLine);

break;
} //End if
} //End if

//keep the repeatable line
fileVector.addElement(line);

} //End else
}while(!moreLine); //End do

lineNumberReader.close();
} //End try

catch (NullPointerException npe)
{
moreLine = false;
}
catch (IOException e)
{
}
}

```

```

update(fileVector, fileName);
}

public int dollarNumber(Vector input)
{
    int count=0;

    for(int i=0; i<input.size(); i++)
        if ( ((String)input.elementAt(i)).equals("$aa") ||
              ((String)input.elementAt(i)).equals("$dd") ||
              ((String)input.elementAt(i)).equals("$ff") ||
              ((String)input.elementAt(i)).equals("$vv") ||
              ((String)input.elementAt(i)).equals("$oo") ||
              ((String)input.elementAt(i)).equals("$ODSvv") ||
              ((String)input.elementAt(i)).equals("$ODSdd") ||
              ((String)input.elementAt(i)).equals("$DWvv") ||
              ((String)input.elementAt(i)).equals("$DWdd") ||
              ((String)input.elementAt(i)).equals("$cc") ||
              ((String)input.elementAt(i)).equals("$pp") ||
              ((String)input.elementAt(i)).equals("$ss") )
            count++;

    return count;
}

public void remove(String mark, String fileName)
{
    LineNumberReader lineNumberReader;
    Vector fileVector = new Vector();
    Vector lineVector = new Vector();
    String line = new String("");
    String newLine = new String("");
    boolean moreLine = true;
    boolean hasProprity = true;

    try
    {
        FileReader fileReader = new FileReader(fileName);
        lineNumberReader = new LineNumberReader(fileReader);

        do
        {
            line = lineNumberReader.readLine().trim();

            if (line.length() == 0)
            {
                fileVector.addElement(" ");
                continue;
            }
            else
            {
                hasProprity = false;

                lineVector = tokenize(line);

                for (int i=0; i<lineVector.size(); i++)
                {
                    if (((String)lineVector.elementAt(i)).equals(mark))
                    {
                        hasProprity=true;
                        break;
                    }
                }
            }
        }

        if (hasProprity==false)

```

```

        fileVector.addElement(line);
    }while(moreLine);

    lineNumberReader.close();
}
catch (NullPointerException npe)
{
    moreLine = false;
}
catch (IOException e)
{
    System.out.println("IOException found by LiuYi at 5");
}

update(fileVector, fileName);
}
}

```

(10) ruleGenerator.java

```

import javax.swing.UIManager;
import java.util.*;

public class ruleGenerator
{
    boolean packFrame = false;

    //Construct the application
    public void Mapping(Vector ODSname,
                       Vector ODSattribute,
                       Vector ODStype,
                       Vector DB,
                       Vector DBname,
                       Vector DBattribute,
                       Vector DBtype,
                       String UID,
                       String PWD)
    {
        Frame2 frame = new Frame2(ODSname,ODSattribute, ODStype, DB, DBname, DBattribute, DBtype, UID,
        PWD);

        //Validate frames that have preset sizes
        //Pack frames that have useful preferred size info, e.g. from their layout

        if (packFrame)
        {
            frame.pack();
        }
        else
        {
            frame.validate();
        }

        frame.setVisible(true);
    }

    //Construct the application
    public ruleGenerator(Vector ODSname, Vector ODSattribute, Vector ODStype, Vector DB, Vector
    DBname, Vector DBattribute, Vector DBtype, String UID, String PWD )
    {
        try
        {
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        }
        catch(Exception e)
        {

```

```

        e.printStackTrace();
    }

    Mapping(ODSname,ODSattribute, ODStype, DB, DBname, DBattribute, DBtype, UID, PWD);
}
}

```

(11) Frame2.java

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import java.io.*;

public class Frame2 extends JFrame
{
    /**
    Define the variable for the GUI components
    */

    JPanel contentPane;
    JLabel jLabel1 = new JLabel();
    JComboBox jComboBox1 = new JComboBox();
    JComboBox jComboBox2 = new JComboBox();
    Box box1;
    JRadioButton jRadioButton1 = new JRadioButton();
    JRadioButton jRadioButton2 = new JRadioButton();
    JRadioButton jRadioButton3 = new JRadioButton();
    JRadioButton jRadioButton4 = new JRadioButton();
    JRadioButton jRadioButton5 = new JRadioButton();
    JRadioButton jRadioButton6 = new JRadioButton();
    JRadioButton jRadioButton7 = new JRadioButton();
    JRadioButton jRadioButton8 = new JRadioButton();
    JRadioButton jRadioButton9 = new JRadioButton();
    JTextField jTextField1 = new JTextField();
    JTextField jTextField2 = new JTextField();
    JTextField jTextField3 = new JTextField();
    JTextField jTextField4 = new JTextField();
    JTextField jTextField5 = new JTextField();
    JTextField jTextField6 = new JTextField();
    JTextField jTextField7 = new JTextField();
    JTextField jTextField8 = new JTextField();
    JTextField jTextField9 = new JTextField();
    JTextField jTextField10 = new JTextField();
    JTextField jTextField11 = new JTextField();
    JCheckBox jCheckBox1 = new JCheckBox();
    JCheckBox jCheckBox2 = new JCheckBox();
    JCheckBox jCheckBox3 = new JCheckBox();
    JCheckBox jCheckBox4 = new JCheckBox();
    JCheckBox jCheckBox5 = new JCheckBox();
    JCheckBox jCheckBox6 = new JCheckBox();
    JCheckBox jCheckBox7 = new JCheckBox();
    JCheckBox jCheckBox8 = new JCheckBox();
    JCheckBox jCheckBox9 = new JCheckBox();
    JLabel jLabel2 = new JLabel();

    JButton jButton1 = new JButton();
    JButton jButton2 = new JButton();
    JTextArea jTextArea1 = new JTextArea();
    JLabel jLabel3 = new JLabel();
    JLabel jLabel4 = new JLabel();
    JTextArea jTextArea2 = new JTextArea();
    String file="";

    //input group
    Vector ODStableName= new Vector();
    Vector DBtableName = new Vector();

```

```

Vector ODSattribute= new Vector();
Vector DBattribute = new Vector();
Vector ODSdataType = new Vector();
Vector DBdataType = new Vector();
Vector databaseName= new Vector();

//current group
String ODSname= new String();
String DBname = new String();
String database = new String();
Vector ODSattr= new Vector();
Vector DBattr = new Vector();
Vector ODStype= new Vector();
Vector DBtype = new Vector();

//output group
Vector ODSn = new Vector();
Vector ODSa = new Vector();
Vector ODSt = new Vector();
Vector DBn = new Vector();
Vector DBa = new Vector();
Vector DBt = new Vector();
Vector DB = new Vector();
Vector constraint =new Vector();

//some global variables for DB
String attribute=new String();
String type =new String();

//some global variables for ODS
String attr =new String();
String dataType =new String();

//some global variables for UID/PWD
String UID = new String();
String PWD = new String();

/*****
Construct the frame
*****/

public Frame2(Vector ODStableName,
              Vector ODSattribute,
              Vector ODSdataType,
              Vector databaseName,
              Vector DBtableName,
              Vector DBattribute,
              Vector DBdataType,
              String UID,
              String PWD)
{
    //set the window
    enableEvents(AWTEvent.WINDOW_EVENT_MASK);

    //set init values of the vectors
    this.ODStableName=ODStableName;
    this.DBtableName =DBtableName;
    this.ODSattribute=ODSattribute;
    this.DBattribute =DBattribute;
    this.ODSdataType =ODSdataType;
    this.DBdataType =DBdataType;
    this.databaseName=databaseName;
    this.UID=UID;
    this.PWD=PWD;

    //start to construct the frame
    try
    {
        jbInit();
    }
}

```

```

catch(Exception e)
{
    e.printStackTrace();
}
}

/*****
Component initialization
*****/

private void jbInit() throws Exception
{
    //copy an uchange vector for futrue using
    ODSname=(String) ODStableName.elementAt(0);
    DBname =(String) DBtableName.elementAt(0);
    database =(String) databaseName.elementAt(0);
    ODSattr=searchAttribute(ODSname, ODStableName, ODSattribute);
    DBattr =searchAttribute(DBname, DBtableName, DBattribute );
    ODStype=searchType(ODSname, ODStableName, ODSdataType);
    DBtype =searchType(DBname, DBtableName, DBdataType );

    //set the windows component abd their prperty
    box1 = Box.createVerticalBox();
    jLabell.setFont(new java.awt.Font("Dialog", 1, 25));
    jLabell.setForeground(Color.yellow);
    jLabell.setText("DATA INTEGRATION RULE GENERATOR");
    jLabell.setBounds(new Rectangle(32, 24, 627, 38));
    contentPane = (JPanel) this.getContentPane();
    contentPane.setLayout(null);
    this.getContentPane().setBackground(SystemColor.desktop);
    this.setSize(new Dimension(700, 700));
    this.setTitle("PROPOSAL DEMONSTRATION");
    contentPane.setBackground(SystemColor.desktop);

    jComboBox1.setBounds(new Rectangle(31, 97, 141, 31));

    String tempVariable=null;

    //add ODS names into linked list
    for (int i=0; i<ODStableName.size(); i++)
        if ( !((String) ODStableName.elementAt(i)).equals(tempVariable) )
            {
                jComboBox1.addItem((String) ODStableName.elementAt(i));
                tempVariable=(String) ODStableName.elementAt(i);
            }

    jComboBox1.addItemListener
    (
        new java.awt.event.ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                jComboBox1_itemStateChanged(e);
            }
        }
    );

    jComboBox2.setBounds(new Rectangle(246, 97, 144, 31));

    tempVariable=null;

    //add DB names into linked list
    for (int i=0; i<DBtableName.size(); i++)
        if ( !((String) DBtableName.elementAt(i)).equals(tempVariable) )
            {
                jComboBox2.addItem((String) databaseName.elementAt(i)+"."+((String) DBtableName.elementAt(i)));
                tempVariable=(String) DBtableName.elementAt(i);
            }
}

```

```

jComboBox2.addItemListener
(
  new java.awt.event.ItemListener()
  {
    public void itemStateChanged(ItemEvent e)
    {
      jComboBox2_itemStateChanged(e);
    }
  }
);

box1.setBounds(new Rectangle(33, 150, 110, 241));

/*****
set dynamic display fot warehouse tables
*****/

if(ODSattr.size()>=1)
{
  jRadioButton1.setBackground(SystemColor.desktop);
  jRadioButton1.setVisible(true);
  jRadioButton1.setText( (String)ODSattr.elementAt(0) );
  jRadioButton1.setForeground(Color.yellow);
  jRadioButton1.addItemListener
  (
    new java.awt.event.ItemListener()
    {

      public void itemStateChanged(ItemEvent e)
      {
        radioStateChanged(e);
      }
    }
  );
}
else
{
  jRadioButton1.setBackground(SystemColor.desktop);
  jRadioButton1.setVisible(false);
}

if(ODSattr.size()>=2)
{
  jRadioButton2.setBackground(SystemColor.desktop);
  jRadioButton2.setVisible(true);
  jRadioButton2.setText( (String) ODSattr.elementAt(1) );
  jRadioButton2.setForeground(Color.yellow);

  jRadioButton2.addItemListener
  (
    new java.awt.event.ItemListener()
    {

      public void itemStateChanged(ItemEvent e)
      {
        radioStateChanged(e);
      }
    }
  );
}
else
{
  jRadioButton2.setBackground(SystemColor.desktop);
  jRadioButton2.setVisible(false);
}

if(ODSattr.size()>=3)
{
  jRadioButton3.setBackground(SystemColor.desktop);
  jRadioButton3.setVisible(true);

```

```

jRadioButton3.setText( (String) ODSattr.elementAt(2) );
jRadioButton3.setForeground(Color.yellow);

jRadioButton3.addItemListener
(
    new java.awt.event.ItemListener()
    {
        public void itemStateChanged(ItemEvent e)
        {
            radioStateChanged(e);
        }
    }
);
}
else
{
    jRadioButton3.setBackground(SystemColor.desktop);
    jRadioButton3.setVisible(false);
}

if(ODSattr.size()>=4)
{
    jRadioButton4.setBackground(SystemColor.desktop);
    jRadioButton4.setVisible(true);
    jRadioButton4.setText( (String) ODSattr.elementAt(3) );
    jRadioButton4.setForeground(Color.yellow);

    jRadioButton4.addItemListener
    (
        new java.awt.event.ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                radioStateChanged(e);
            }
        }
    );
}
else
{
    jRadioButton4.setBackground(SystemColor.desktop);
    jRadioButton4.setVisible(false);
}

if(ODSattr.size()>=5)
{
    jRadioButton5.setBackground(SystemColor.desktop);
    jRadioButton5.setVisible(true);
    jRadioButton5.setText( (String) ODSattr.elementAt(4) );
    jRadioButton5.setForeground(Color.yellow);

    jRadioButton5.addItemListener
    (
        new java.awt.event.ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                radioStateChanged(e);
            }
        }
    );
}
else
{
    jRadioButton5.setBackground(SystemColor.desktop);
    jRadioButton5.setVisible(false);
}
}

```



```

if(ODSattr.size()>=6)
{
jRadioButton6.setBackground(SystemColor.desktop);
jRadioButton6.setVisible(true);
jRadioButton6.setText( (String) ODSattr.elementAt(5) );
jRadioButton6.setForeground(Color.yellow);

jRadioButton6.addItemListener
(
new java.awt.event.ItemListener()
{

public void itemStateChanged(ItemEvent e)
{
radioStateChanged(e);
}
}
);
}
else
{
jRadioButton6.setBackground(SystemColor.desktop);
jRadioButton6.setVisible(false);
}

if(ODSattr.size()>=7)
{
jRadioButton7.setBackground(SystemColor.desktop);
jRadioButton7.setVisible(true);
jRadioButton7.setText( (String) ODSattr.elementAt(6) );
jRadioButton7.setForeground(Color.yellow);

jRadioButton7.addItemListener
(
new java.awt.event.ItemListener()
{

public void itemStateChanged(ItemEvent e)
{
radioStateChanged(e);
}
}
);
}
else
{
jRadioButton7.setBackground(SystemColor.desktop);
jRadioButton7.setVisible(false);
}

if(ODSattr.size()>=8)
{
jRadioButton8.setBackground(SystemColor.desktop);
jRadioButton8.setVisible(true);
jRadioButton8.setText( (String) ODSattr.elementAt(7) );
jRadioButton8.setForeground(Color.yellow);

jRadioButton8.addItemListener
(
new java.awt.event.ItemListener()
{

public void itemStateChanged(ItemEvent e)
{
radioStateChanged(e);
}
}
);
}
else
{

```

```

jRadioButton8.setBackground(SystemColor.desktop);
jRadioButton8.setVisible(false);
}

if(ODSattr.size()>=9)
{
jRadioButton9.setBackground(SystemColor.desktop);
jRadioButton9.setVisible(true);
jRadioButton9.setText( (String) ODSattr.elementAt(8) );
jRadioButton9.setForeground(Color.yellow);

jRadioButton9.addItemListener
(
new java.awt.event.ItemListener()
{
public void itemStateChanged(ItemEvent e)
{
radioStateChanged(e);
}
}
);
}
else
{
jRadioButton9.setBackground(SystemColor.desktop);
jRadioButton9.setVisible(false);
}

/*****
set dynamic display for text fields
*****/

jTextField1.setBackground(SystemColor.white);
jTextField1.setBounds(new Rectangle(164, 150, 115, 23));
if (DBattr.size()>=1)
jTextField1.setVisible(true);
else
jTextField1.setVisible(false);

jTextField2.setBackground(SystemColor.white);
jTextField2.setBounds(new Rectangle(164, 175, 115, 23));
if (DBattr.size()>=2)
jTextField2.setVisible(true);
else
jTextField2.setVisible(false);

jTextField3.setBackground(SystemColor.white);
jTextField3.setBounds(new Rectangle(164, 201, 115, 23));
if (DBattr.size()>=3)
jTextField3.setVisible(true);
else
jTextField3.setVisible(false);

jTextField4.setBackground(SystemColor.white);
jTextField4.setBounds(new Rectangle(164, 227, 115, 23));
if (DBattr.size()>=4)
jTextField4.setVisible(true);
else
jTextField4.setVisible(false);

jTextField5.setBackground(SystemColor.white);
jTextField5.setBounds(new Rectangle(164, 252, 115, 23));
if (DBattr.size()>=5)
jTextField5.setVisible(true);
else
jTextField5.setVisible(false);

jTextField6.setBackground(SystemColor.white);
jTextField6.setBounds(new Rectangle(164, 277, 115, 23));
if (DBattr.size()>=6)

```

```

    jTextField6.setVisible(true);
else
    jTextField6.setVisible(false);

jTextField7.setBackground(SystemColor.white);
jTextField7.setBounds(new Rectangle(164, 300, 115, 23));
if (DBAttr.size()>=7)
    jTextField7.setVisible(true);
else
    jTextField7.setVisible(false);

jTextField8.setBackground(SystemColor.white);
jTextField8.setBounds(new Rectangle(164, 325, 115, 23));
if (DBAttr.size()>=8)
    jTextField8.setVisible(true);
else
    jTextField8.setVisible(false);

jTextField9.setBackground(SystemColor.white);
jTextField9.setBounds(new Rectangle(164, 350, 115, 23));
if (DBAttr.size()>=9)
    jTextField9.setVisible(true);
else
    jTextField9.setVisible(false);

/*****
set source checking box and their prity
*****/

jCheckBox1.setBackground(SystemColor.desktop);
jCheckBox1.setForeground(Color.yellow);
jCheckBox1.setBounds(new Rectangle(292, 150, 95, 25));
if(DBAttr.size()>=1)
{
    jCheckBox1.setText( (String) DBAttr.elementAt(0) );
    jCheckBox1.setVisible(true);
    jCheckBox1.addItemListener
    (
        new java.awt.event.ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                jCheckBoxStateChanged(e);
            }
        }
    );
}
else
    jCheckBox1.setVisible(false);

jCheckBox2.setBounds(new Rectangle(292, 174, 95, 25));
jCheckBox2.setBackground(SystemColor.desktop);
jCheckBox2.setForeground(Color.yellow);
if(DBAttr.size()>=2)
{
    jCheckBox2.setText( (String) DBAttr.elementAt(1) );
    jCheckBox2.setVisible(true);
    jCheckBox2.addItemListener
    (
        new java.awt.event.ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                jCheckBoxStateChanged(e);
            }
        }
    );
}
else
    jCheckBox2.setVisible(false);

```

```

jCheckBox3.setBounds(new Rectangle(292, 198, 95, 25));
jCheckBox3.setBackground(SystemColor.desktop);
jCheckBox3.setForeground(Color.yellow);
if(DBAttr.size()>=3)
{
    jCheckBox3.setText( (String) DBAttr.elementAt(2) );
    jCheckBox3.setVisible(true);
    jCheckBox3.addItemListener
    (
        new java.awt.event.ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                jCheckBoxStateChanged(e);
            }
        }
    );
}
else
    jCheckBox3.setVisible(false);

jCheckBox4.setBounds(new Rectangle(292, 223, 95, 25));
jCheckBox4.setBackground(SystemColor.desktop);
jCheckBox4.setForeground(Color.yellow);
if(DBAttr.size()>=4)
{
    jCheckBox4.setText( (String) DBAttr.elementAt(3) );
    jCheckBox4.setVisible(true);
    jCheckBox4.addItemListener
    (
        new java.awt.event.ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                jCheckBoxStateChanged(e);
            }
        }
    );
}
else
    jCheckBox4.setVisible(false);

jCheckBox5.setBounds(new Rectangle(292, 250, 95, 25));
jCheckBox5.setBackground(SystemColor.desktop);
jCheckBox5.setForeground(Color.yellow);
if(DBAttr.size()>=5)
{
    jCheckBox5.setText( (String) DBAttr.elementAt(4) );
    jCheckBox5.setVisible(true);
    jCheckBox5.addItemListener
    (
        new java.awt.event.ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                jCheckBoxStateChanged(e);
            }
        }
    );
}
else
    jCheckBox5.setVisible(false);

jCheckBox6.setBounds(new Rectangle(292, 275, 95, 25));
jCheckBox6.setBackground(SystemColor.desktop);
jCheckBox6.setForeground(Color.yellow);
if(DBAttr.size()>=6)
{
    jCheckBox6.setText( (String) DBAttr.elementAt(5) );
    jCheckBox6.setVisible(true);
    jCheckBox6.addItemListener

```

```

(
    new java.awt.event.ItemListener()
    {
        public void itemStateChanged(ItemEvent e)
        {
            jCheckBoxStateChanged(e);
        }
    }
);
}
else
    jCheckBox6.setVisible(false);

jCheckBox7.setBounds(new Rectangle(292, 299, 95, 25));
jCheckBox7.setBackground(SystemColor.desktop);
jCheckBox7.setForeground(Color.yellow);
if(DBAttr.size()>=7)
{
    jCheckBox7.setText( (String) DBAttr.elementAt(6) );
    jCheckBox7.setVisible(true);
    jCheckBox7.addItemListener
    (
        new java.awt.event.ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                jCheckBoxStateChanged(e);
            }
        }
    );
}
else
    jCheckBox7.setVisible(false);

jCheckBox8.setBounds(new Rectangle(292, 323, 95, 25));
jCheckBox8.setBackground(SystemColor.desktop);
jCheckBox8.setForeground(Color.yellow);
if(DBAttr.size()>=8)
{
    jCheckBox8.setText( (String) DBAttr.elementAt(7) );
    jCheckBox8.setVisible(true);
    jCheckBox8.addItemListener
    (
        new java.awt.event.ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                jCheckBoxStateChanged(e);
            }
        }
    );
}
else
    jCheckBox8.setVisible(false);

jCheckBox9.setBounds(new Rectangle(292, 349, 95, 25));
jCheckBox9.setBackground(SystemColor.desktop);
jCheckBox9.setForeground(Color.yellow);
if(DBAttr.size()>=9)
{
    jCheckBox9.setText( (String) DBAttr.elementAt(8) );
    jCheckBox9.setVisible(true);
    jCheckBox9.addItemListener
    (
        new java.awt.event.ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                jCheckBoxStateChanged(e);
            }
        }
    )
}

```

```

    );
}
else
    jCheckBox9.setVisible(false);

/*****
set other
*****/

jLabel2.setFont(new java.awt.Font("Dialog", 1, 16));
jLabel2.setForeground(Color.yellow);
jLabel2.setText(" CONSTRAINTS");
jLabel2.setBounds(new Rectangle(33, 388, 355, 33));
jTextField10.setBackground(SystemColor.white);
jTextField10.setBounds(new Rectangle(33, 425, 160, 33));
jTextField11.setBackground(SystemColor.white);
jTextField11.setBounds(new Rectangle(220, 425, 160, 33));
jButton1.setText("GENERATE RULE");
jButton1.setBounds(new Rectangle(33, 478, 160, 50));
jButton1.addActionListener
(
    new java.awt.event.ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            button1_actionPerformed(e);
        }
    }
);
jButton2.setText("NEXT >>");
jButton2.setBounds(new Rectangle(219, 479, 165, 50));
jButton2.addActionListener
(
    new java.awt.event.ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            button2_actionPerformed(e);
        }
    }
);

jTextArea1.setBackground(SystemColor.white);
jTextArea1.setBounds(new Rectangle(406, 153, 259, 221));

jLabel3.setFont(new java.awt.Font("Dialog", 1, 16));
jLabel3.setForeground(Color.yellow);
jLabel3.setText("INTEGRATION RULE");
jLabel3.setBounds(new Rectangle(408, 97, 241, 33));
jLabel4.setFont(new java.awt.Font("Dialog", 1, 16));
jLabel4.setForeground(Color.yellow);
jLabel4.setText("WARNING");
jLabel4.setBounds(new Rectangle(405, 390, 169, 29));

jTextArea2.setBackground(SystemColor.white);
jTextArea2.setBounds(new Rectangle(408, 426, 256, 165));

contentPane.add(jLabel1, null);
contentPane.add(jComboBox1, null);
contentPane.add(box1, null);
box1.add(jRadioButton1, null);
box1.add(jRadioButton2, null);
box1.add(jRadioButton3, null);
box1.add(jRadioButton4, null);
box1.add(jRadioButton5, null);
box1.add(jRadioButton6, null);
box1.add(jRadioButton7, null);
box1.add(jRadioButton8, null);
box1.add(jRadioButton9, null);
contentPane.add(jTextField1, null);
contentPane.add(jTextField9, null);

```

```

contentPane.add(jTextField8, null);
contentPane.add(jTextField7, null);
contentPane.add(jTextField6, null);
contentPane.add(jTextField3, null);
contentPane.add(jTextField2, null);
contentPane.add(jTextField4, null);
contentPane.add(jTextField5, null);
contentPane.add(jCheckBox1, null);
contentPane.add(jCheckBox2, null);
contentPane.add(jCheckBox3, null);
contentPane.add(jCheckBox4, null);
contentPane.add(jCheckBox5, null);
contentPane.add(jCheckBox6, null);
contentPane.add(jCheckBox7, null);
contentPane.add(jCheckBox9, null);
contentPane.add(jCheckBox8, null);
contentPane.add(jLabel2, null);
contentPane.add(jTextField10, null);
contentPane.add(jTextField11, null);
contentPane.add(jButton1, null);
contentPane.add(jButton2, null);
contentPane.add(jComboBox2, null);
contentPane.add(jTextArea1, null);
contentPane.add(jLabel3, null);
contentPane.add(jLabel4, null);
contentPane.add(jTextArea2, null);
}

/*****
set warehouse linked list
*****/

void jComboBox1_itemStateChanged(ItemEvent e)
{
//copy an uchange vector for futrue using
ODSname=(String) e.getItem();
ODSattr=searchAttribute(ODSname, ODStableName, ODSattribute);
ODSdtype=searchType(ODSname, ODStableName, ODSdataType);

/*****
set dynamic display fot warehouse tables
*****/

if(ODSattr.size()>=1)
{
jRadioButton1.setBackground(SystemColor.desktop);
jRadioButton1.setVisible(true);
jRadioButton1.setText( (String)ODSattr.elementAt(0) );
jRadioButton1.setForeground(Color.yellow);
jRadioButton1.addItemListener
(
new java.awt.event.ItemListener()
{
public void itemStateChanged(ItemEvent e)
{
radioStateChanged(e);
}
}
);
}
else
{
jRadioButton1.setBackground(SystemColor.desktop);
jRadioButton1.setVisible(false);
}

if(ODSattr.size()>=2)
{
jRadioButton2.setBackground(SystemColor.desktop);

```

```

jRadioButton2.setVisible(true);
jRadioButton2.setText( (String) ODSattr.elementAt(1) );
jRadioButton2.setForeground(Color.yellow);

jRadioButton2.addItemListener
(
    new java.awt.event.ItemListener()
    {

        public void itemStateChanged(ItemEvent e)
        {
            radioStateChanged(e);
        }
    }
);
}
else
{
    jRadioButton2.setBackground(SystemColor.desktop);
    jRadioButton2.setVisible(false);
}

if(ODSattr.size()>=3)
{
    jRadioButton3.setBackground(SystemColor.desktop);
    jRadioButton3.setVisible(true);
    jRadioButton3.setText( (String) ODSattr.elementAt(2) );
    jRadioButton3.setForeground(Color.yellow);

    jRadioButton3.addItemListener
    (
        new java.awt.event.ItemListener()
        {

            public void itemStateChanged(ItemEvent e)
            {
                radioStateChanged(e);
            }
        }
    );
}
else
{
    jRadioButton3.setBackground(SystemColor.desktop);
    jRadioButton3.setVisible(false);
}

if(ODSattr.size()>=4)
{
    jRadioButton4.setBackground(SystemColor.desktop);
    jRadioButton4.setVisible(true);
    jRadioButton4.setText( (String) ODSattr.elementAt(3) );
    jRadioButton4.setForeground(Color.yellow);

    jRadioButton4.addItemListener
    (
        new java.awt.event.ItemListener()
        {

            public void itemStateChanged(ItemEvent e)
            {
                radioStateChanged(e);
            }
        }
    );
}
else
{
    jRadioButton4.setBackground(SystemColor.desktop);
    jRadioButton4.setVisible(false);
}
}

```



```

if(ODSattr.size()>=5)
{
jRadioButton5.setBackground(SystemColor.desktop);
jRadioButton5.setVisible(true);
jRadioButton5.setText( (String) ODSattr.elementAt(4) );
jRadioButton5.setForeground(Color.yellow);

jRadioButton5.addItemListener
(
new java.awt.event.ItemListener()
{

public void itemStateChanged(ItemEvent e)
{
radioStateChanged(e);
}
}
);
}
else
{
jRadioButton5.setBackground(SystemColor.desktop);
jRadioButton5.setVisible(false);
}

if(ODSattr.size()>=6)
{
jRadioButton6.setBackground(SystemColor.desktop);
jRadioButton6.setVisible(true);
jRadioButton6.setText( (String) ODSattr.elementAt(5) );
jRadioButton6.setForeground(Color.yellow);

jRadioButton6.addItemListener
(
new java.awt.event.ItemListener()
{

public void itemStateChanged(ItemEvent e)
{
radioStateChanged(e);
}
}
);
}
else
{
jRadioButton6.setBackground(SystemColor.desktop);
jRadioButton6.setVisible(false);
}

if(ODSattr.size()>=7)
{
jRadioButton7.setBackground(SystemColor.desktop);
jRadioButton7.setVisible(true);
jRadioButton7.setText( (String) ODSattr.elementAt(6) );
jRadioButton7.setForeground(Color.yellow);

jRadioButton7.addItemListener
(
new java.awt.event.ItemListener()
{

public void itemStateChanged(ItemEvent e)
{
radioStateChanged(e);
}
}
);
}
else

```

```

{
  jButton7.setBackground(SystemColor.desktop);
  jButton7.setVisible(false);
}

if(ODSattr.size()>=8)
{
  jButton8.setBackground(SystemColor.desktop);
  jButton8.setVisible(true);
  jButton8.setText( (String) ODSattr.elementAt(7) );
  jButton8.setForeground(Color.yellow);

  jButton8.addItemListener
  (
    new java.awt.event.ItemListener()
    {
      public void itemStateChanged(ItemEvent e)
      {
        radioStateChanged(e);
      }
    }
  );
}
else
{
  jButton8.setBackground(SystemColor.desktop);
  jButton8.setVisible(false);
}

if(ODSattr.size()>=9)
{
  jButton9.setBackground(SystemColor.desktop);
  jButton9.setVisible(true);
  jButton9.setText( (String) ODSattr.elementAt(8) );
  jButton9.setForeground(Color.yellow);

  jButton9.addItemListener
  (
    new java.awt.event.ItemListener()
    {
      public void itemStateChanged(ItemEvent e)
      {
        radioStateChanged(e);
      }
    }
  );
}
else
{
  jButton9.setBackground(SystemColor.desktop);
  jButton9.setVisible(false);
}

contentPane.add(box1, null);
box1.add(jButton1, null);
box1.add(jButton2, null);
box1.add(jButton3, null);
box1.add(jButton4, null);
box1.add(jButton5, null);
box1.add(jButton6, null);
box1.add(jButton7, null);
box1.add(jButton8, null);
box1.add(jButton9, null);
}

/*****
Display the current attribute list of current table in source
*****/

```

```

void jComboBox2_itemStateChanged(ItemEvent e)
{
    DBname =(String) e.getItem();
    database = DBname.substring(0, DBname.indexOf("."));
    DBname= DBname.substring(DBname.indexOf(".")+1, DBname.length());
    DBattr =searchAttribute(DBname, DBtableName, DBattribute );
    DBtype =searchType(DBname, DBtableName, DBdataType );

    /*****
    set dynamic display for text fileds
    *****/

    jTextField1.setBackground(SystemColor.white);
    jTextField1.setBounds(new Rectangle(164, 150, 115, 23));
    if (DBattr.size()>=1)
        jTextField1.setVisible(true);
    else
        jTextField1.setVisible(false);

    jTextField2.setBackground(SystemColor.white);
    jTextField2.setBounds(new Rectangle(164, 175, 115, 23));
    if (DBattr.size()>=2)
        jTextField2.setVisible(true);
    else
        jTextField2.setVisible(false);

    jTextField3.setBackground(SystemColor.white);
    jTextField3.setBounds(new Rectangle(164, 201, 115, 23));
    if (DBattr.size()>=3)
        jTextField3.setVisible(true);
    else
        jTextField3.setVisible(false);

    jTextField4.setBackground(SystemColor.white);
    jTextField4.setBounds(new Rectangle(164, 227, 115, 23));
    if (DBattr.size()>=4)
        jTextField4.setVisible(true);
    else
        jTextField4.setVisible(false);

    jTextField5.setBackground(SystemColor.white);
    jTextField5.setBounds(new Rectangle(164, 252, 115, 23));
    if (DBattr.size()>=5)
        jTextField5.setVisible(true);
    else
        jTextField5.setVisible(false);

    jTextField6.setBackground(SystemColor.white);
    jTextField6.setBounds(new Rectangle(164, 277, 115, 23));
    if (DBattr.size()>=6)
        jTextField6.setVisible(true);
    else
        jTextField6.setVisible(false);

    jTextField7.setBackground(SystemColor.white);
    jTextField7.setBounds(new Rectangle(164, 300, 115, 23));
    if (DBattr.size()>=7)
        jTextField7.setVisible(true);
    else
        jTextField7.setVisible(false);

    jTextField8.setBackground(SystemColor.white);
    jTextField8.setBounds(new Rectangle(164, 325, 115, 23));
    if (DBattr.size()>=8)
        jTextField8.setVisible(true);
    else
        jTextField8.setVisible(false);
}

```

```

jTextField9.setBackground(SystemColor.white);
jTextField9.setBounds(new Rectangle(164, 350, 115, 23));
if (DBattr.size()>=90)
    jTextField9.setVisible(true);
else
    jTextField9.setVisible(false);

/*****
set source checking box and their prity
*****/

jCheckBox1.setBackground(SystemColor.desktop);
jCheckBox1.setForeground(Color.yellow);
jCheckBox1.setBounds(new Rectangle(292, 150, 95, 25));
if(DBattr.size()>=1)
{
    jCheckBox1.setText( (String) DBattr.elementAt(0) );
    jCheckBox1.setVisible(true);
    jCheckBox1.addItemListener
    (
        new java.awt.event.ItemListener()
        {
            {
                public void itemStateChanged(ItemEvent e)
                {
                    jCheckBoxStateChanged(e);
                }
            }
        }
    );
}
else
    jCheckBox1.setVisible(false);

jCheckBox2.setBounds(new Rectangle(292, 174, 95, 25));
jCheckBox2.setBackground(SystemColor.desktop);
jCheckBox2.setForeground(Color.yellow);
if(DBattr.size()>=2)
{
    jCheckBox2.setText( (String) DBattr.elementAt(1) );
    jCheckBox2.setVisible(true);
    jCheckBox2.addItemListener
    (
        new java.awt.event.ItemListener()
        {
            {
                public void itemStateChanged(ItemEvent e)
                {
                    jCheckBoxStateChanged(e);
                }
            }
        }
    );
}
else
    jCheckBox2.setVisible(false);

jCheckBox3.setBounds(new Rectangle(292, 198, 95, 25));
jCheckBox3.setBackground(SystemColor.desktop);
jCheckBox3.setForeground(Color.yellow);
if(DBattr.size()>=3)
{
    jCheckBox3.setText( (String) DBattr.elementAt(2) );
    jCheckBox3.setVisible(true);
    jCheckBox3.addItemListener
    (
        new java.awt.event.ItemListener()
        {
            {
                public void itemStateChanged(ItemEvent e)
                {
                    jCheckBoxStateChanged(e);
                }
            }
        }
    );
}
}

```

```

else
    jCheckBox3.setVisible(false);

jCheckBox4.setBounds(new Rectangle(292, 223, 95, 25));
jCheckBox4.setBackground(SystemColor.desktop);
jCheckBox4.setForeground(Color.yellow);
if(DBAttr.size()>=4)
{
    jCheckBox4.setText( (String) DBAttr.elementAt(3) );
    jCheckBox4.setVisible(true);
    jCheckBox4.addItemListener
    (
        new java.awt.event.ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                jCheckBoxStateChanged(e);
            }
        }
    );
}
else
    jCheckBox4.setVisible(false);

jCheckBox5.setBounds(new Rectangle(292, 250, 95, 25));
jCheckBox5.setBackground(SystemColor.desktop);
jCheckBox5.setForeground(Color.yellow);
if(DBAttr.size()>=5)
{
    jCheckBox5.setText( (String) DBAttr.elementAt(4) );
    jCheckBox5.setVisible(true);
    jCheckBox5.addItemListener
    (
        new java.awt.event.ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                jCheckBoxStateChanged(e);
            }
        }
    );
}
else
    jCheckBox5.setVisible(false);

jCheckBox6.setBounds(new Rectangle(292, 275, 95, 25));
jCheckBox6.setBackground(SystemColor.desktop);
jCheckBox6.setForeground(Color.yellow);
if(DBAttr.size()>=6)
{
    jCheckBox6.setText( (String) DBAttr.elementAt(5) );
    jCheckBox6.setVisible(true);
    jCheckBox6.addItemListener
    (
        new java.awt.event.ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                jCheckBoxStateChanged(e);
            }
        }
    );
}
else
    jCheckBox6.setVisible(false);

jCheckBox7.setBounds(new Rectangle(292, 299, 95, 25));
jCheckBox7.setBackground(SystemColor.desktop);
jCheckBox7.setForeground(Color.yellow);
if(DBAttr.size()>=7)
{

```

```

jCheckBox7.setText( (String) DBattr.elementAt(6) );
jCheckBox7.setVisible(true);
jCheckBox7.addItemListener
(
    new java.awt.event.ItemListener()
    {
        public void itemStateChanged(ItemEvent e)
        {
            jCheckBoxStateChanged(e);
        }
    }
);
}
else
    jCheckBox7.setVisible(false);

jCheckBox8.setBounds(new Rectangle(292, 323, 95, 25));
jCheckBox8.setBackground(SystemColor.desktop);
jCheckBox8.setForeground(Color.yellow);
if(DBattr.size()>=8)
{
    jCheckBox8.setText( (String) DBattr.elementAt(7) );
    jCheckBox8.setVisible(true);
    jCheckBox8.addItemListener
    (
        new java.awt.event.ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                jCheckBoxStateChanged(e);
            }
        }
    );
}
else
    jCheckBox8.setVisible(false);

jCheckBox9.setBounds(new Rectangle(292, 349, 95, 25));
jCheckBox9.setBackground(SystemColor.desktop);
jCheckBox9.setForeground(Color.yellow);
if(DBattr.size()>=9)
{
    jCheckBox9.setText( (String) DBattr.elementAt(8) );
    jCheckBox9.setVisible(true);
    jCheckBox9.addItemListener
    (
        new java.awt.event.ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                jCheckBoxStateChanged(e);
            }
        }
    );
}
else
    jCheckBox9.setVisible(false);

//add elements
contentPane.add(jTextField1, null);
contentPane.add(jTextField9, null);
contentPane.add(jTextField8, null);
contentPane.add(jTextField7, null);
contentPane.add(jTextField6, null);
contentPane.add(jTextField3, null);
contentPane.add(jTextField2, null);
contentPane.add(jTextField4, null);
contentPane.add(jTextField5, null);
contentPane.add(jCheckBox1, null);
contentPane.add(jCheckBox2, null);

```

```

contentPane.add(jCheckBox3, null);
contentPane.add(jCheckBox4, null);
contentPane.add(jCheckBox5, null);
contentPane.add(jCheckBox6, null);
contentPane.add(jCheckBox7, null);
contentPane.add(jCheckBox9, null);
contentPane.add(jCheckBox8, null);
}

/*****
set radio button's value
*****/

void radioStateChanged(ItemEvent e)
{
    if(e.getSource()==jRadioButton1)
    {
        attr    =(String)ODSattr.elementAt(0);
        dataType=(String)ODStype.elementAt(0);
    }

    else if(e.getSource()==jRadioButton2)
    {
        attr    =(String)ODSattr.elementAt(1);
        dataType=(String)ODStype.elementAt(1);
    }

    else if(e.getSource()==jRadioButton3)
    {
        attr    =(String)ODSattr.elementAt(2);
        dataType=(String)ODStype.elementAt(2);
    }

    else if(e.getSource()==jRadioButton4)
    {
        attr    =(String)ODSattr.elementAt(3);
        dataType=(String)ODStype.elementAt(3);
    }

    else if(e.getSource()==jRadioButton5)
    {
        attr    =(String)ODSattr.elementAt(4);
        dataType=(String)ODStype.elementAt(4);
    }

    else if(e.getSource()==jRadioButton6)
    {
        attr    =(String)ODSattr.elementAt(5);
        dataType=(String)ODStype.elementAt(5);
    }

    else if(e.getSource()==jRadioButton7)
    {
        attr    =(String)ODSattr.elementAt(6);
        dataType=(String)ODStype.elementAt(6);
    }

    else if(e.getSource()==jRadioButton8)
    {
        attr    =(String)ODSattr.elementAt(7);
        dataType=(String)ODStype.elementAt(7);
    }

    else if(e.getSource()==jRadioButton9)
    {
        attr    =(String)ODSattr.elementAt(8);
        dataType=(String)ODStype.elementAt(8);
    }
}

```

```

/*****
when check box choose, set the temp variable
*****/

void jCheckBoxStateChanged(ItemEvent e)
{
    if (jCheckBox1.isSelected())
    {
        attribute=(String)DBAttr.elementAt(0);
        type=(String)DBtype.elementAt(0);
    }

    if (jCheckBox2.isSelected())
    {
        attribute=(String)DBAttr.elementAt(1);
        type=(String)DBtype.elementAt(1);
    }

    if (jCheckBox3.isSelected())
    {
        attribute=(String)DBAttr.elementAt(2);
        type=(String)DBtype.elementAt(2);
    }

    if (jCheckBox4.isSelected())
    {
        attribute=(String)DBAttr.elementAt(3);
        type=(String)DBtype.elementAt(3);
    }

    if (jCheckBox5.isSelected())
    {
        attribute=(String)DBAttr.elementAt(4);
        type=(String)DBtype.elementAt(4);
    }

    if (jCheckBox6.isSelected())
    {
        attribute=(String)DBAttr.elementAt(5);
        type=(String)DBtype.elementAt(5);
    }

    if (jCheckBox7.isSelected())
    {
        attribute=(String)DBAttr.elementAt(6);
        type=(String)DBtype.elementAt(6);
    }

    if (jCheckBox8.isSelected())
    {
        attribute=(String)DBAttr.elementAt(7);
        type=(String)DBtype.elementAt(7);
    }

    if (jCheckBox9.isSelected())
    {
        attribute=(String)DBAttr.elementAt(8);
        type=(String)DBtype.elementAt(8);
    }
}

/*****
push the submit button, set variable into output vector
and display the integration rule at text area window
*****/

void button1_actionPerformed(ActionEvent e)

```



```

{
//Checking from file to read rule or by manual
String s = jTextField11.getText();
jTextField11.setText("");

if (!s.equals(""))
{

jTextArea2.append("Start reading file "+s+"\n");

//get input from a file
//prompt for a file name and return it
String fileName = s;

//save the lines in the file into a vector
inputFromFile(fileName);

//display the file
for (int i=0; i<ODSa.size(); i++)
{

if ( ((String) constraint.elementAt(i)).equals("null") )
{
jTextArea1.append( (String) ODSn.elementAt(i)+
". "+
(String) ODSa.elementAt(i)+
" <== "+
(String) DBn.elementAt(i)+
". "+
(String) DBa.elementAt(i)+
" without constraint "+
"\n");

constraint.setElementAt("",i);
}
else
jTextArea1.append( (String) ODSn.elementAt(i)+
". "+
(String) ODSa.elementAt(i)+
" <== "+
(String) DBn.elementAt(i)+
". "+
(String) DBa.elementAt(i)+
" with constraint "+
(String) constraint.elementAt(i)+
"\n");

}

}
else if (!file.equals(""))
{

jTextArea2.append("Start reading file from "+file+" again\n");

//get input from a file
//prompt for a file name and return it
String fileName = file;

file="";

//save the lines in the file into a vector
inputFromFile(fileName);

//display the file
for (int i=0; i<ODSa.size(); i++)
{

if ( ((String) constraint.elementAt(i)).equals("null"))
{
jTextArea1.append( (String) ODSn.elementAt(i)+

```

```

                "."+
                (String) ODSa.elementAt(i)+
                " <== "+
                (String) DBn.elementAt(i)+
                "."+
                (String) DBa.elementAt(i)+"\n");

        constraint.setElementAt("",i);
    }
    else
        jTextArea1.append( (String) ODSn.elementAt(i)+
                "."+
                (String) ODSa.elementAt(i)+
                " <== "+
                (String) DBn.elementAt(i)+
                "."+
                (String) DBa.elementAt(i)+
                " with constraint "+
                (String) constraint.elementAt(i)+
                "\n");
    }
}
else
{
    //new insert parts for virtual integration
    if (!jCheckBox1.isSelected() &&
        !jCheckBox2.isSelected() &&
        !jCheckBox3.isSelected() &&
        !jCheckBox4.isSelected() &&
        !jCheckBox5.isSelected() &&
        !jCheckBox6.isSelected() &&
        !jCheckBox7.isSelected() &&
        !jCheckBox8.isSelected() &&
        !jCheckBox9.isSelected() &&
        !jRadioButton1.isSelected() &&
        !jRadioButton2.isSelected() &&
        !jRadioButton3.isSelected() &&
        !jRadioButton4.isSelected() &&
        !jRadioButton5.isSelected() &&
        !jRadioButton6.isSelected() &&
        !jRadioButton7.isSelected() &&
        !jRadioButton8.isSelected() &&
        !jRadioButton9.isSelected() )
    {
        jTextArea2.append( "Please choose one relation");
        /*
        for( int i=0; i<DBattr.size(); i++)
        {
            //Do nothing here. must choose some icon
            ODSn.addElement(ODSname);
            ODSa.addElement((String) ODSattr.elementAt(i) );
            ODSt.addElement((String) ODStype.elementAt(i) );
            DB.addElement( database );
            DBn.addElement( DBname );
            DBa.addElement((String) DBattr.elementAt(i) );
            DBt.addElement((String) DBtype.elementAt(i) );
            constraint.addElement("null");

            jTextArea1.append( ODSname+"."+
                (String) ODSattr.elementAt(i)+
                " <== "+database+"."+DBname+"."+
                (String) DBattr.elementAt(i)+"\n");
        }
        */
    }
    else if ((!jCheckBox1.isSelected() &&
        !jCheckBox2.isSelected() &&
        !jCheckBox3.isSelected() &&
        !jCheckBox4.isSelected() &&
        !jCheckBox5.isSelected() &&
        !jCheckBox6.isSelected() &&
        !jCheckBox7.isSelected() &&

```

```

        !jCheckBox8.isSelected() &&
        !jCheckBox9.isSelected() ) &&
        (jRadioButton1.isSelected() ||
         jRadioButton2.isSelected() ||
         jRadioButton3.isSelected() ||
         jRadioButton4.isSelected() ||
         jRadioButton5.isSelected() ||
         jRadioButton6.isSelected() ||
         jRadioButton7.isSelected() ||
         jRadioButton8.isSelected() ||
         jRadioButton9.isSelected())
    }

    if (jRadioButton1.isSelected() && !((String) ODSattr.elementAt(0)).equals("time") )
    {
        ODSn.addElement( ODSname );
        ODSa.addElement( (String) ODSattr.elementAt(0) );
        ODSt.addElement( (String) ODStype.elementAt(0) );
        DB.addElement(database);
        DBn.addElement(" ");
        DBa.addElement(" ");
        DBt.addElement(" ");
        constraint.addElement("\'+database+'");
        jTextAreal.append( ODSname+"."+((String) ODSattr.elementAt(0))+" <== '"+database+"'\n");
    }
    else if (jRadioButton2.isSelected() && !((String) ODSattr.elementAt(1)).equals("time") )
    {
        ODSn.addElement( ODSname );
        ODSa.addElement( (String) ODSattr.elementAt(1) );
        ODSt.addElement( (String) ODStype.elementAt(1) );
        DB.addElement(database);
        DBn.addElement(" ");
        DBa.addElement(" ");
        DBt.addElement(" ");
        constraint.addElement("\'+database+'");
        jTextAreal.append( ODSname+"."+((String) ODSattr.elementAt(1))+" <== '"+database+"'\n");
    }
    else if (jRadioButton3.isSelected() && !((String) ODSattr.elementAt(2)).equals("time") )
    {
        ODSn.addElement( ODSname );
        ODSa.addElement( (String) ODSattr.elementAt(2) );
        ODSt.addElement( (String) ODStype.elementAt(2) );
        DB.addElement(database);
        DBn.addElement(" ");
        DBa.addElement(" ");
        DBt.addElement(" ");
        constraint.addElement("\'+database+'");
        jTextAreal.append( ODSname+"."+((String) ODSattr.elementAt(2))+" <== '"+database+"'\n");
    }
    else if (jRadioButton4.isSelected() && !((String) ODSattr.elementAt(3)).equals("time") )
    {
        ODSn.addElement( ODSname );
        ODSa.addElement( (String) ODSattr.elementAt(3) );
        ODSt.addElement( (String) ODStype.elementAt(3) );
        DB.addElement(database);
        DBn.addElement(" ");
        DBa.addElement(" ");
        DBt.addElement(" ");
        constraint.addElement("\'+database+'");
        jTextAreal.append( ODSname+"."+((String) ODSattr.elementAt(3))+" <== '"+database+"'\n");
    }
    else if (jRadioButton5.isSelected() && !((String) ODSattr.elementAt(4)).equals("time") )
    {
        ODSn.addElement( ODSname );
        ODSa.addElement( (String) ODSattr.elementAt(4) );
        ODSt.addElement( (String) ODStype.elementAt(4) );
        DB.addElement(database);
        DBn.addElement(" ");
        DBa.addElement(" ");
        DBt.addElement(" ");
        constraint.addElement("\'+database+'");
    }

```

```

jTextAreal.append( ODSname+"."+((String) ODSattr.elementAt(4))+" <== '"+database+"\n");
}
else if (jRadioButton6.isSelected() && !((String) ODSattr.elementAt(5)).equals("time") )
{
ODSn.addElement( ODSname );
ODSa.addElement( (String) ODSattr.elementAt(5) );
ODSt.addElement( (String) ODStype.elementAt(5) );
DB.addElement(database);
DBn.addElement(" ");
DBa.addElement(" ");
DBt.addElement(" ");
constraint.addElement("\'+database+'");
jTextAreal.append( ODSname+"."+((String) ODSattr.elementAt(5))+" <== '"+database+"\n");
}
else if (jRadioButton7.isSelected() && !((String) ODSattr.elementAt(6)).equals("time") )
{
ODSn.addElement( ODSname );
ODSa.addElement( (String) ODSattr.elementAt(6) );
ODSt.addElement( (String) ODStype.elementAt(6) );
DB.addElement(database);
DBn.addElement(" ");
DBa.addElement(" ");
DBt.addElement(" ");
constraint.addElement("\'+database+'");
jTextAreal.append( ODSname+"."+((String) ODSattr.elementAt(6))+" <== '"+database+"\n");
}
else if (jRadioButton8.isSelected() && !((String) ODSattr.elementAt(7)).equals("time") )
{
ODSn.addElement( ODSname );
ODSa.addElement( (String) ODSattr.elementAt(7) );
ODSt.addElement( (String) ODStype.elementAt(7) );
DB.addElement(database);
DBn.addElement(" ");
DBa.addElement(" ");
DBt.addElement(" ");
constraint.addElement("\'+database+'");
jTextAreal.append( ODSname+"."+((String) ODSattr.elementAt(7))+" <== '"+database+"\n");
}
else if (jRadioButton9.isSelected() && !((String) ODSattr.elementAt(8)).equals("time") )
{
ODSn.addElement( ODSname );
ODSa.addElement( (String) ODSattr.elementAt(8) );
ODSt.addElement( (String) ODStype.elementAt(8) );
DB.addElement(database);
DBn.addElement(" ");
DBa.addElement(" ");
DBt.addElement(" ");
constraint.addElement("\'+database+'");
jTextAreal.append( ODSname+"."+((String) ODSattr.elementAt(8))+" <== '"+database+"\n");
}
else
{
ODSn.addElement( ODSname );
//////////may be wrong here
ODSa.addElement( "time" );
ODSt.addElement( " " );
DB.addElement(database);
DBn.addElement(" ");
DBa.addElement(" ");
DBt.addElement(" ");
constraint.addElement("to_char(sysdate())");
jTextAreal.append( ODSname+"."+((String) ODSattr.elementAt(0))+" <== system time");
}
}
else
{
//only for dispaly, so it is local variable
String constraints =new String();

//add ODS integrated information into vector
ODSn.addElement(ODSname);

```

```

ODSa.addElement(attr);
ODSt.addElement(dataType);

//add DB integrated information into vector
DB.addElement(database);
DBn.addElement(DBname);
DBa.addElement(attribute);
DBt.addElement(type);

if (jCheckBox1.isSelected())
if ((jTextField10.getText()).equals(""))
{
constraint.addElement(jTextField1.getText());
constraints=jTextField1.getText();
}
else
{
constraint.addElement(jTextField10.getText());
constraints=jTextField10.getText();
}

if (jCheckBox2.isSelected())
if ((jTextField10.getText()).equals(""))
{
constraint.addElement(jTextField2.getText());
constraints=jTextField2.getText();
}
else
{
constraint.addElement(jTextField10.getText());
constraints=jTextField10.getText();
}

if (jCheckBox3.isSelected())
if ((jTextField10.getText()).equals(""))
{
constraint.addElement(jTextField3.getText());
constraints=jTextField3.getText();
}
else
{
constraint.addElement(jTextField10.getText());
constraints=jTextField10.getText();
}

if (jCheckBox4.isSelected())
if ((jTextField10.getText()).equals(""))
{
constraint.addElement(jTextField4.getText());
constraints=jTextField4.getText();
}
else
{
constraint.addElement(jTextField10.getText());
constraints=jTextField10.getText();
}

if (jCheckBox5.isSelected())
if ((jTextField10.getText()).equals(""))
{
constraint.addElement(jTextField5.getText());
constraints=jTextField5.getText();
}
else
{
constraint.addElement(jTextField10.getText());
constraints=jTextField10.getText();
}

if (jCheckBox6.isSelected())

```

```

if ((jTextField10.getText()).equals(""))
{
    constraint.addElement(jTextField6.getText());
    constraints=jTextField6.getText();
}
else
{
    constraint.addElement(jTextField10.getText());
    constraints=jTextField10.getText();
}

if (jCheckBox7.isSelected())
if ((jTextField10.getText()).equals(""))
{
    constraint.addElement(jTextField7.getText());
    constraints=jTextField7.getText();
}
else
{
    constraint.addElement(jTextField10.getText());
    constraints=jTextField10.getText();
}

if (jCheckBox8.isSelected())
if ((jTextField10.getText()).equals(""))
{
    constraint.addElement(jTextField8.getText());
    constraints=jTextField8.getText();
}
else
{
    constraint.addElement(jTextField10.getText());
    constraints=jTextField10.getText();
}

if (jCheckBox9.isSelected())
if ((jTextField10.getText()).equals(""))
{
    constraint.addElement(jTextField9.getText());
    constraints=jTextField9.getText();
}
else
{
    constraint.addElement(jTextField10.getText());
    constraints=jTextField10.getText();
}

jTextAreal.append(ODSname+"."+attr+" <== "+DBname+"."+attribute+
" with constraints "+constraints+"\n");
}

//recover any component false
jCheckBox1.setSelected(false);
jCheckBox2.setSelected(false);
jCheckBox3.setSelected(false);
jCheckBox4.setSelected(false);
jCheckBox5.setSelected(false);
jCheckBox6.setSelected(false);
jCheckBox7.setSelected(false);
jCheckBox8.setSelected(false);
jCheckBox9.setSelected(false);
jTextField1.setText("");
jTextField2.setText("");
jTextField3.setText("");
jTextField4.setText("");
jTextField5.setText("");
jTextField6.setText("");
jTextField7.setText("");
jTextField8.setText("");
jTextField9.setText("");
jTextField10.setText("");

```

```

jTextField11.setText("");
jRadioButton1.setSelected(false);
jRadioButton2.setSelected(false);
jRadioButton3.setSelected(false);
jRadioButton4.setSelected(false);
jRadioButton5.setSelected(false);
jRadioButton6.setSelected(false);
jRadioButton7.setSelected(false);
jRadioButton8.setSelected(false);
jRadioButton9.setSelected(false);
}
}

/*****
set button 4 to call the data integration
*****/

void button2_actionPerformed(ActionEvent e)
{
    setVisible(false);

    dataIntegration di=new dataIntegration( ODSn,
                                           ODSa,
                                           ODSr,
                                           DB,
                                           DBn,
                                           DBa,
                                           DBt,
                                           constraint,
                                           UID,
                                           PWD );
}

/*****
Overridden so we can exit when window is closed
*****/

protected void processWindowEvent(WindowEvent e)
{
    super.processWindowEvent(e);
    if (e.getID() == WindowEvent.WINDOW_CLOSING)
    {
        System.exit(0);
    }
}

/*****
Search attribute of one table and return the group of attributes's type
*****/

private Vector searchType(String name, Vector allName, Vector allType)
{
    Vector output=new Vector();

    for(int i=0; i<allType.size(); i++)
        if ( ((String) allName.elementAt(i)).equals(name) )
            output.addElement((String) allType.elementAt(i));

    return output;
}

/*****
Search attribute of one table and return the group of attributes
*****/

```

```

private Vector searchAttribute(String name, Vector tableName, Vector attribute)
{
    Vector output=new Vector();

    for(int i=0; i<tableName.size(); i++)
        if ( ((String) tableName.elementAt(i)).equals(name) )
            output.addElement((String)attribute.elementAt(i));

    return output;
}

/*****
inputFromFile reads command from a file, insert each command
line into a vector, and returns data in this vector
*****/
private void inputFromFile(String fileName)
{
    LineNumberReader lineNumberReader;

    Vector inputVector = new Vector();
    String a_line = new String("");

    boolean more_line = true;

    try
    {
        FileReader fileReader = new FileReader(fileName);
        lineNumberReader = new LineNumberReader(fileReader);

        do
        {
            //reads 1 line at a time
            a_line = lineNumberReader.readLine().trim();

            if (a_line.length() == 0)
                continue;
            else
                //save this line as a element in the vector
                inputVector.addElement(a_line);
        }

        while (more_line);
        lineNumberReader.close();
    }
    catch (NullPointerException npe)
    {
        //if reach the EndOfFile flag
        if (inputVector.size() ==0)
            //the input file is empty
            //return ""
            inputVector.addElement("");

        more_line = false;
    }
    catch (IOException e)
    {
        //for other file I/O errors
        file = JOptionPane.showInputDialog("No such file\nPlease retype the file name\nFor future
information, Please Contact Liu Yi\nat 519-253-3000-3003");

        inputVector.removeAllElements();
    }

    //Read them into Vector
    String inString=new String();
    for (int i=0; i<inputVector.size(); i++)
    {
        inString= (String) inputVector.elementAt(i);
        StringTokenizer st = new StringTokenizer(inString, ",");
    }
}

```



```

    if (st.countTokens() != 8)
    {
        JOptionPane.showMessageDialog(null, "Error variable number\n\nSystem will be
terminated.\nPlease contact Liu Yi \nat 519-253-3000-3003\nfor future helping");
        System.exit(1);
    }

    ODSn.addElement( st.nextToken() );
    ODSa.addElement( st.nextToken() );
    ODSr.addElement( st.nextToken() );
    DB.addElement( st.nextToken() );
    DBn.addElement( st.nextToken() );
    DBa.addElement( st.nextToken() );
    DBt.addElement( st.nextToken() );
    constraint.addElement( st.nextToken() );

    if(st.hasMoreTokens())
    {
        ODStableName.removeAllElements();
        databaseName.removeAllElements();
        DBStableName.removeAllElements();
        ODSattribute.removeAllElements();
        DBattribute.removeAllElements();
        ODSdataType.removeAllElements();
        DBdataType.removeAllElements();
        constraint.removeAllElements();

        JOptionPane.showMessageDialog(null, "System will be terminated.\n\nPlease contact LiuYi \nat
519-253-3000-3003\nfor future helping");
        System.exit(1);
    }
}
}
}
}
}

```

(12) dataIntegration.java

```

/*****
          797 proposal Project

          Data integration

          Supervisor      Dr. Ezeife
          Student Name:   LiuYi
          Student Numnber: 999030031

Class data integration is for integrating source information
creating the query statement for the integration based on
integration rule (1-5). it offers the information for code
generator.

=====
-----
Data Integration
-----
+ dataIntegration
+ void  extractingIntegration();
+ void  cleaningIntegration();
+ void  loadingIntegration();
- vector tokenize(String);
*****/

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

```

```

import java.util.*;
import java.io.*;

public class dataIntegration
{
    /*****
    Set private source and ODS table information in vector
    and also OUT (that is after integration vector). ODS have
    redundancy rows in order to correspondant DB rows
    input shorter and output longest
    *****/

    //construction the object of the code generator
    codeGenerator cg=new ccdeGenerator();

    //input data
    private Vector inputODSname      = new Vector();
    private Vector inputODSattribute= new Vector();
    private Vector inputODStype      = new Vector();
    private Vector inputSDB          = new Vector();
    private Vector inputSDBname      = new Vector();
    private Vector inputSDBattribute= new Vector();
    private Vector inputSDBtype      = new Vector();
    private Vector inputConstraint   = new Vector();

    //current data
    String currentODSname      = new String();
    Vector currentODSattribute= new Vector();
    Vector currentODStype      = new Vector();
    String currentSDB          = new String();
    String currentSDBname      = new String();
    Vector currentSDBattribute= new Vector();
    Vector currentSDBtype      = new Vector();
    Vector currentConstraint   = new Vector();

    //output data
    String outputODSname      = new String();
    Vector outputODSattribute= new Vector();
    Vector outputODStype      = new Vector();
    String outputSDB          = new String();
    String outputSDBname      = new String();
    Vector outputSDBattribute= new Vector();
    Vector outputSDBtype      = new Vector();
    Vector outputConstraint   = new Vector();

    //some global variables for UID/PWD
    String UID = new String();
    String PWD = new String();

    /*****
    Accepting 7 vector information from data integration rule
    generator, seperating them table by table and transfer them
    to the integration to integrate the data
    *****/
    public dataIntegration( Vector ODSname,
                           Vector ODSattribute,
                           Vector ODStype,
                           Vector SDB,
                           Vector SDBname,
                           Vector SDBattribute,
                           Vector SDBtype,
                           Vector constraint,
                           String UID,
                           String PWD)
    {
        //set local rules from parameter
        inputODSname      = ODSname;
        inputODSattribute= ODSattribute;
        inputODStype      = ODStype;
        inputSDB          = SDB;
    }
}

```

```

inputSDBname      = SDBname;
inputSDBattribute= SDBattribute;
inputSDBtype     = SDBtype;
inputConstraint  = constraint;
this.UID=UID;
this.PWD=PWD;

//only for keep vector for non garbage collection
int length=ODSname.size();
String inputODSnameArray[] =new String[length];
String inputODSattributeArray[]=new String[length];
String inputODStypeArray[] =new String[length];
String inputSDBArray[]     =new String[length];
String inputSDBnameArray[] =new String[length];
String inputSDBattributeArray[]=new String[length];
String inputSDBtypeArray[] =new String[length];
String inputConstraintArray[] =new String[length];

for (int i=0; i<length; i++)
{
    inputODSnameArray[i]      =(String)inputODSname.elementAt(i);
    inputODSattributeArray[i]=(String)inputODSattribute.elementAt(i);
    inputODStypeArray[i]     =(String)inputODStype.elementAt(i);
    inputSDBArray[i]        =(String)inputSDB.elementAt(i);
    inputSDBnameArray[i]    =(String)inputSDBname.elementAt(i);
    inputSDBattributeArray[i]=(String)inputSDBattribute.elementAt(i);
    inputSDBtypeArray[i]    =(String)inputSDBtype.elementAt(i);
    inputConstraintArray[i] =(String)inputConstraint.elementAt(i);
}

mainGenerator mg=new mainGenerator(inputSDB,
                                   inputSDBname,
                                   inputODSname,
                                   inputSDBattribute,
                                   inputSDBtype,
                                   inputODSattribute,
                                   inputODStype,
                                   inputConstraint,
                                   UID,
                                   PWD);

for (int i=0; i<length; i++)
{
    inputSDB.addElement(inputSDBArray[i]);
    inputSDBname.addElement(inputSDBnameArray[i]);
    inputODSname.addElement(inputODSnameArray[i]);
    inputSDBattribute.addElement(inputSDBattributeArray[i]);
    inputSDBtype.addElement(inputSDBtypeArray[i]);
    inputODSattribute.addElement(inputODSattributeArray[i]);
    inputODStype.addElement(inputODStypeArray[i]);
    inputConstraint.addElement(inputConstraintArray[i]);
}

//remove any about the fact tables
while (inputODSname.indexOf("fact")!= -1)
{
    int position=(int) inputODSname.indexOf("fact");
    inputODSname.removeElementAt(position);
    inputODSattribute.removeElementAt(position);
    inputODStype.removeElementAt(position);
    inputSDB.removeElementAt(position);
    inputSDBname.removeElementAt(position);
    inputSDBattribute.removeElementAt(position);
    inputSDBtype.removeElementAt(position);
    inputConstraint.removeElementAt(position);
}

extractingIntegration();

//recover local input rules from parameter
inputODSname      = ODSname;

```

```

inputODSattribute= ODSattribute;
inputODStype      = ODStype;

//cleaningIntegration();

//recover local input rules from parameter
inputODSname      = ODSname;
inputODSattribute= ODSattribute;
inputODStype      = ODStype;

//loadingIntegration();

execution ex=new execution();
}

/*****
extractingIntegration is for creating the integrating attribute
for the proc data extraction
*****/

public void extractingIntegration()
{
//Define a variable
int position=-1;

//seperating by each sdb table, put them into current
//variable set and call the code integration method
while(inputSDBname.size()>0)
{
//set current source name

currentSDBname=(String) inputSDBname.elementAt(0);
currentSDB      =(String) inputSDB.elementAt(0);

//choose the first SDB, put the information
//about this source into current vector
for (int j=0; j<inputSDBname.size(); j++)
if (((String)inputSDBname.elementAt(j)).equals(currentSDBname))
{
//set ODS name for current rule
currentODSname=(String) inputODSname.elementAt(j);

//set vector for current rule
currentODSattribute.addElement((String) inputODSattribute.elementAt(j));
currentODStype.addElement(      (String) inputODStype.elementAt(j) );
currentSDBattribute.addElement((String) inputSDBattribute.elementAt(j));
currentSDBtype.addElement(      (String) inputSDBtype.elementAt(j) );
currentConstraint.addElement( (String) inputConstraint.elementAt(j) );
}

//delete the current information from input vector
//so, keep the input vector is un_integrated
while (inputSDBname.indexOf(currentSDBname)!=-1)
{
position=(int) inputSDBname.indexOf(currentSDBname);
inputODSname.removeElementAt(position);
inputODSattribute.removeElementAt(position);
inputODStype.removeElementAt(position);
inputSDB.removeElementAt(position);
inputSDBname.removeElementAt(position);
inputSDBattribute.removeElementAt(position);
inputSDBtype.removeElementAt(position );
inputConstraint.removeElementAt(position);
}

//call integration method to integration the current information
//and also generating the proc code current set into output set

```

```

integratingExtractionInfo();

//fomilazed the variable type
formulazation();

//after integration the current rule,
//generate the rpc code using output
cg.extractingGenerator(outputSDB,
                        outputSDBname,
                        outputODSname,
                        outputSDBattribute,
                        outputSDBtype,
                        outputODSattribute,
                        outputODStype,
                        outputConstraint);

//remove the current and output set for set another set
//remove current data
currentODSname = null;
currentSDB     = null;
currentSDBname = null;
currentODSattribute.removeAllElements();
currentODStype.removeAllElements();
currentSDBattribute.removeAllElements();
currentSDBtype.removeAllElements();
currentConstraint.removeAllElements();

//remove output data
outputSDB     = null;
outputSDBname = null;
outputODSname = null;
outputSDBattribute.removeAllElements();
outputSDBtype.removeAllElements();
outputODSattribute.removeAllElements();
outputODStype.removeAllElements();
outputConstraint.removeAllElements();
}
}

/*****
integration is main function to control all part of integration
*****/

public void integratingExtractionInfo()
{
    if (currentSDBname.equals("null"))
    {
        // outputODSname=currentODSname;
        // outputSDBname=currentSDBname;
        // outputODSattribute= ;
        // outputODStype= ;
        // outputSDBattribute= ;
        // outputSDBtype= ;
        // outputConstraint= ;   ???
    }
    else
    {
        //set the table name for future code generation
        outputODSname=currentODSname;
        outputSDB     =currentSDB;
        outputSDBname=currentSDBname;

        //intergrate the current set line by line
        for (int i=0; i<currentSDBattribute.size(); i++)
        {
            //add value no matter it is integrated or not
            //only need integrating, add new set else add old
            structure(i);
        }
    }
}

```

```

//data type and measure only replace the current value
dataType(i);

//data type and measure only replace the current value
measurement(i);
}
}
}

/*****
Integration different structure of attribute.
(integration rule 3)

Algorithm:
Step 1. Read DB constraints
Step 2. tokenize the rule
Step 2. check any ++ sing
      yes
      1. search all of the piece of target
      2. delect these vector
Step 3.

*****/

public void structure(int lineNumber)
{
//define local variable
String  constString= new String();
Vector  constVector= new Vector();
boolean hasIntegrated = false, hasSameLine=false;

//check it is structure problem or not
//tokenize the constratints
constString = (String)currentConstraint.elementAt(lineNumber);
constVector = tokenize(constString);

//chech each to see it is ++ or not
for (int i=0; i<constVector.size(); i++)
{
if ( ((String)constVector.elementAt(i)).equals("||" ) )
{
hasIntegrated =true;

//search the line is in ODSa or not, if it is in ODSa
//do nothing, else update it into the line i of ODSa
for (int j=0; j<outputConstraint.size(); j++)
if ( ((String) outputConstraint.elementAt(j)).equals(constString) )
hasSameLine=true;

//if there is same line add space, else add constraints
if (hasSameLine)
{
outputConstraint.addElement(" ");
outputODSattribute.addElement(" ");
}
else
{
outputConstraint.addElement((String) currentConstraint.elementAt(lineNumber) );
outputODSattribute.addElement( (String) currentODSattribute.elementAt(lineNumber) );
}

//it need integratin, so add rest line into output set
outputSDBattribute.addElement( (String) currentSDBattribute.elementAt(lineNumber));
outputSDBtype.addElement(      (String) currentSDBtype.elementAt(lineNumber) );
outputODStype.addElement(      (String) currentODStype.elementAt(lineNumber) );

}
}

} //end if
} //end for

//check it is integrated or not for this line
//if it is integrated, do nothing else set output

```

```

//value for this line. in this case convert value is
//db attribute
if (!hasIntegrated)
{
    outputODSattribute.addElement( (String) currentODSattribute.elementAt(lineNumber) );
    outputSDBattribute.addElement( (String) currentSDBattribute.elementAt(lineNumber) );
    outputODStype.addElement( (String) currentODStype.elementAt(lineNumber) );
    outputSDBtype.addElement( (String) currentSDBtype.elementAt(lineNumber) );
    outputConstraint.addElement( (String) currentSDBattribute.elementAt(lineNumber) );
} //end if
}

/*****
Integration different data type of attribute.
(integration rule 4)

Algorithm:
Step 1. Read ODS type
Step 2. Read DB type
Step 3. Convert DB type to ODS type based on 4 cases
    1. number to char
    2. char to number
    3. date to char
    4. char to date

*****/

public void dataType(int lineNumber)
{
    String ODStypeString = new String();
    String SDBtypeString = new String();

    //get both data type
    ODStypeString = (String) currentODStype.elementAt(lineNumber);
    SDBtypeString = (String) currentSDBtype.elementAt(lineNumber);

    //check both are same or not, if it is same, it don't need integrating
    //else need integrating
    if ( !ODStypeString.equals(SDBtypeString) )
    {
        //for different case to convert
        //convert from char to number
        if ( (ODStypeString.substring(0,6)).equals("number") &&
            (SDBtypeString.substring(0,8)).equals("varchar2") )

            outputConstraint.setElementAt("to_number( : " +
                (String)outputConstraint.elementAt(lineNumber)+
                ")", lineNumber );

        //convert from number to char
        else if ( (ODStypeString.substring(0,8)).equals("varchar2") &&
            (SDBtypeString.substring(0,6)).equals("number") )

            outputConstraint.setElementAt("to_char( : " +
                (String)outputConstraint.elementAt(lineNumber)+
                ")", lineNumber );

        //convert from date to char
        else if ( (ODStypeString.substring(0,8)).equals("varchar2") &&
            (SDBtypeString.substring(0,4)).equals("date") )

            outputConstraint.setElementAt("to_char( : " +
                (String)outputConstraint.elementAt(lineNumber) +
                ", 'Month dd, yyyy')", lineNumber );

        //convert char to date
        else if ( (ODStypeString.substring(0,4)).equals("date") &&

```

```

        (SDBtypeString.substring(0,8)).equals("varchar2" )

outputConstraint.setElementAt("to_date( : " +
        (String)outputConstraint.elementAt(lineNumber)+
        ", 'month dd yyyy')", lineNumber);

//Can not convert by SQL language
else
//warning type conversion error
JOptionPane.showMessageDialog(null, "Illegal data type conversion\nSource data type"+
        SDBtypeString.substring(0,8)+
        "\ncan not convert to target data type"+
        ODStypeString.substring(0,4)+
        "\nPlease contact Liu Yi at University of Windsor\n at 519-253-
3000-3000\nfor future helping");
    }
}

/*****
Integration different measurement of attribute.
(integration rule 2)

Algorithm:
Step 1. Read DB name, type, constraint
Step 2. Scan DB constraints
Step 3. is 'measurement integration'?
        Yes, change DB attribute's format (with this change)
Step 4. repeat 2 and 3
*****/

public void measurement(int lineNumber)
{
//define some variavle for local
Vector lineVector = new Vector();
String line      = new String();
String element   = new String();

//tokenize the line
line = (String)currentConstraint.elementAt(lineNumber);
lineVector = tokenize(line);

//check it to see it need integrating or not
for (int i=0; i<lineVector.size(); i++)
{
    element = (String)lineVector.elementAt(i);

    if( element.equals("+") ||
        element.equals("-") ||
        element.equals("**") ||
        element.equals("/") )
    {
//put the convert condition
outputConstraint.setElementAt((String)outputConstraint.elementAt(lineNumber)+line, lineNumber);
    }
}
}

/*****
Try to form the SQL variable into proc variable format and also
the linke of two string
*****/
public void formulazation()
{
outputODStype.removeAllElements();
String type = new String();

for (int i=0; i<outputSDBtype.size(); i++)

```



```

{
    type = (String) outputSDBtype.elementAt(i);

    if ( (type.substring(0,4)).equals("date") )
    {
        outputODStype.addElement("date");
    }
    else if ( ((type.substring(0,6)).equals("number")) && (type.length()>10) )
    {
        outputODStype.addElement("float");
    }
    else if ( ((type.substring(0,6)).equals("number")) && (type.length() <= 10) )
    {
        outputODStype.addElement("int");
    }
    else if ( (type.substring(0,8)).equals("varchar2") )
    {
        outputODStype.addElement("char");
    }
    else
    {
        System.out.println("Illegal data type conversition by LiuYi from Data type integration 1");
    }
}
}

```

```

/*****
Generation is a function to call the code generator.
no variable name
no variable type
no attribute type
*****/

```

```

public void cleaningIntegration()
{
    int position=0;

    //seperating by each odstable, put them into current
    //variable and call different integration method
    while( inputODSname.size()>0)
    {
        outputODSname=(String)inputODSname.elementAt(0);

        //set current information for this table
        for (int i=0; i<inputODSname.size(); i++)
            if (((String)inputODSname.elementAt(i)).equals(outputODSname))
            {
                //set vector for current rule
                outputODSattribute.addElement((String) inputODSattribute.elementAt(i));
                outputODStype.addElement( (String) inputODStype.elementAt(i));
            }

        //delete the current information from input vector
        //so, keep the input vector is un_integrated
        while(inputODSname.indexOf(outputODSname)!=-1)
        {
            position=inputODSname.indexOf(outputODSname);
            inputODSname.removeElementAt(position);
            inputODStype.removeElementAt(position);
            inputODSattribute.removeElementAt(position);
        }

        conversition();

        //clean the duplication record
        for (int i=0; i<outputODSattribute.size(); i++)
        {
            // outputSDBname used for a temp variable set
            outputSDBname=((String) outputODSattribute.elementAt(i));

```

```

for (int j=i+1 ; j<outputODSattribute.size(); j++)
    if( ( (String)outputODSattribute.elementAt(j) ).equals(outputSDBname) )
    {
        outputODSattribute.removeElementAt(j);
        outputODStype.removeElementAt(j);
        j--;
    }
}

//call code generator
//cg.cleaningGenerator(outputODSname, outputODSattribute, outputODStype);

//remove the current and output set for set another set
//remove current data
outputODSname = null;
outputODSattribute.removeAllElements();
outputODStype.removeAllElements();
}
}

/*****
Generation is a function to call the code generator.
no variable name
no variable type
no attribute type
*****/

public void loadingIntegration()
{
    int position=0;

    //seperating by each odstable, put them into current
    //variable and call different integration method
    while( inputODSname.size(>0)
    {
        outputODSname=(String)inputODSname.elementAt(0);

        //set current information for this table
        for (int i=0; i<inputODSname.size(); i++)
            if ( ((String)inputODSname.elementAt(i) ).equals(outputODSname) )
            {
                //set vector for current rule
                outputODSattribute.addElement((String) inputODSattribute.elementAt(i));
                outputODStype.addElement(      (String) inputODStype.elementAt(i));
            }

        //delete the current information from input vector
        //so, keep the input vector is un_integrated
        while(inputODSname.indexOf(outputODSname)!=-1)
        {
            position=inputODSname.indexOf(outputODSname);
            inputODSname.removeElementAt(position);
            inputODStype.removeElementAt(position);
            inputODSattribute.removeElementAt(position);
        }

        conversition();

        //clean the duplication record
        for (int i=0; i<outputODSattribute.size(); i++)
        {
            // outputSDBname used for a temp variable set
            outputSDBname=((String) outputODSattribute.elementAt(i));

            for (int j=i+1 ; j<outputODSattribute.size(); j++)
                if( ( (String)outputODSattribute.elementAt(j) ).equals(outputSDBname) )
                {
                    outputODSattribute.removeElementAt(j);
                    outputODStype.removeElementAt(j);
                    j--;
                }
            }
        }
    }
}

```

```

    }
}

//call code generator
cg.loadingGenerator(outputODSname, outputODSattribute, outputODStype, "nothing", "nothing");

//remove the current and output set for set another set
//remove current data
outputODSname = "";
outputODSattribute.removeAllElements();
outputODStype.removeAllElements();
}
}

/*****
Try to form the SQL variable into proc variable format and also
the linke of two string
*****/
public void conversion()
{
    String type = new String();

    for (int i=0; i<outputODStype.size(); i++)
    {
        type = (String) outputODStype.elementAt(i);

        if ( (type.substring(0,4)).equals("date") )
        {
            outputODStype.setElementAt("date",i);
        }
        else if ( ((type.substring(0,6)).equals("number")) && (type.length()>10) )
        {
            outputODStype.setElementAt("float",i);
        }
        else if ( ((type.substring(0,6)).equals("number")) && (type.length() <= 10) )
        {
            outputODStype.setElementAt("int",i);
        }
        else if ( (type.substring(0,8)).equals("varchar2") )
        {
            outputODStype.setElementAt("char",i);
        }
        else
        {
            System.out.println("Illegal data type conversion by LiuYi from Data type integration 2");
        }
    }
}

/*****
Take a line string as input and output the vector
which includes the string element in this line
*****/
public Vector tokenize(String inString)
{
    Vector tokenVector = new Vector();
    String tokenString = new String();
    StringTokenizer st = new StringTokenizer(inString);

    while (st.hasMoreTokens())
    {
        tokenString = st.nextToken();
        tokenVector.addElement(tokenString);
    }
    return tokenVector;
}
}

```

(13) codeGenerator.java

```
/*
    797 Project

    codeGenerator

    Supervisor      Dr. Ezeife
    Student Name:   LiuYi
    Student Number: 999030031

Class codeGenerator is due for the creations of the
different pro.C programming based on the templates,
The input information are the information from
integration rule in the format of the Vector.

=====
codeGenerator
-----
+ void creatingGenerator();
+ void extractingGenerator();
+ void cleaningGenerator();
+ void loadingGenerator();
- bool locating(String);
*****/

import java.util.*;
import java.io.*;
import javax.swing.*;

public class codeGenerator
{
    Vector currentSDBname      =new Vector();
    Vector currentSDBtableName=new Vector();
    Vector currentODStableName=new Vector();
    Vector currentSDBattribute=new Vector();
    Vector currentODSattribute=new Vector();
    Vector currentSDBdataType =new Vector();
    Vector currentODSdataType =new Vector();
    Vector currentConstraint  =new Vector();
    String template1="";
    String template="";
    String template3="";
    String template4="";

    public void creatingGenerator(String tableName,
                                  Vector variableName,
                                  Vector variableDataType,
                                  String userid,
                                  String password)
    {
        //Defining the function name for the hasProprity $ff
        String functionName = "creating"+tableName;

        //Defining the output file name. For pro.C,
        //it is file name plus dot pc
        //String template = "creating.pc";
        if (template1.equals(""))
            template1 = JOptionPane.showInputDialog("Please Input Creation Program Name");

        /***Component setup***/

        //Determining if the 'Create' is already in the template
        //if this output file is aviable, add one more creating
        //component into it, else, add main component and creating
        //component into the template
        if (search("#include", template1))

```

```

append("creatingComponent", templatel, templatel);
else
{
append("creatingComponent", "mainComponent", templatel);

//insert metadata creation
append("metadataComponent", templatel, templatel);
doubleInsert("metadata","$ff", templatel );

//insert time deminsion
append("timeComponent", templatel, templatel);
doubleInsert("time","$ff", templatel );

singleInsert(userid, "SUID", templatel);
singleInsert(password, "SPWD", templatel);
}

/**Prority setup**

//Inserting new function into $ff prority and keep one
//more $ff in this template in order to issue the location
//next new function inserted
doubleInsert("ODS"+functionName,"$ff", templatel );
doubleInsert("DW"+functionName, "$ff", templatel );

//Inserting new function into $f prority and no more
//$f left
singleInsert("ODS"+functionName,"$ODSf", templatel);
singleInsert("DW"+functionName,"$DWf", templatel);

//Inserting new table name into template
singleInsert("ODS"+tableName, "$ODSt", templatel);
singleInsert("DW"+tableName, "$DWT", templatel);

//Inserting the different variable and their type into template
for (int i=0; i<variableName.size()-1;i++)
    tribleInsert((String)variableName.elementAt(i), "$vv", (String)variableDataType.elementAt(i),
"$dd", templatel );

//Inserting the last variable and their type into template
singleInsert((String)variableName.lastElement(), "$v", templatel);
singleInsert((String)variableDataType.lastElement(), "$d", templatel);

//Keep the template clean in order to avoid next component
//prority insert into this component's prority
remove("$vv", templatel);
remove("$dd", templatel);

//displayCode(templatel);

//displayCode(templatel);
append("initMetadataComponent", templatel, templatel);
doubleInsert(tableName+"Init", "$ff", templatel);
singleInsert(tableName+"Init", "$f", templatel);
singleInsert("'" +tableName+"'", "$t", templatel);
}

```

```

/*****
Generating only one DW table creation program.

```

Algorithm:

- Step 1. Write functionName into \$ff in template
- Step 2. Insert creatingComponent into template in \$bb
- Step 3. Write functionName into \$f in template
- Step 4. if it is not the last variable
 - {
 - take one variable name from Vector
 - Write variableName at \$vv
 - Write variableDataType at \$aa
 - }

```

        else
        {
            Write variableName at $v
            Write variableDataType at $a
            end of the program
        }
Step 5. Repeat Step 4

*****/

public void extractingGenerator(String SDBname,
                               String SDBtableName,
                               String ODStableName,
                               Vector SDBattribute,
                               Vector SDBtype,
                               Vector ODSattribute,
                               Vector ODStype,
                               Vector convertValue)
{
    if (ODStableName.equals("fact"))
    {
        for (int i=0; i<SDBattribute.size(); i++)
        {
            currentSDBname.addElement(SDBname);
            currentSDBtableName.addElement(SDBtableName);
            currentODStableName.addElement(ODStableName);
            currentSDBattribute.addElement((String)SDBattribute.elementAt(i));
            currentODSattribute.addElement((String)ODSattribute.elementAt(i));
            currentSDBdataType.addElement((String)SDBtype.elementAt(i));
            currentODSdataType.addElement((String)ODStype.elementAt(i));
            currentConstraint.addElement((String)convertValue.elementAt(i));
        }

        return;
    }
    else
    {
        //Defining the function name for the hasPriority $ff
        String functionName = "extracting"+SDBtableName;

        //Defining the output file name. For pro.C,
        //it is file name plus dot pc
        if (template.equals(""))
            template = JOptionPane.showInputDialog("Please Input Creation Program Name again");

        /***Component setup***/

        //Determining if the 'include' is already in the template
        //if this output file is available, add one more creating
        //component into it, else, add main component and creating
        //component into the template
        if (search("#include", template))
            append("extractingComponent", template, template);
        else
            append("extractingComponent", "mainComponent", template);

        /***Priority setup***/

        //Inserting new function into $ff priority and keep one
        //more $ff in this template in order to issue the location
        //next new function inserted
        doubleInsert(functionName,"$ff", template);

        //Inserting new function into $f priority and no more $f left
        singleInsert(functionName,"$f", template);

        //Inserting new cursor name into template
        singleInsert(SDBtableName+"Cursor", "$C", template);
    }
}

```

```

//Inserting new table name into template
singleInsert(SDBtableName, "$t", template);

//insert ODStable name into template
singleInsert("ODS"+ODStableName, "$ODSt", template);

//Inserting the different variable and their type into template
for (int i=0; i<SDBAttribute.size();i++)
if (template.indexOf(".sql")!=0 || template.indexOf(".SQL")!=0)
{
    if ( ((String)ODStype.elementAt(i)).equals("char") )
        tribleInsert( "varchar2(20)", "$dd",
            (String)SDBAttribute.elementAt(i), "$vv",
            template);
    else
        tribleInsert( (String)ODStype.elementAt(i), "$dd",
            (String)SDBAttribute.elementAt(i), "$vv",
            template);
}
else
{
    if ( ((String)ODStype.elementAt(i)).equals("char") )
        tribleInsert( (String)ODStype.elementAt(i), "$dd",
            (String)SDBAttribute.elementAt(i)+"[20]", "$vv",
            template);
    else
        tribleInsert( (String)ODStype.elementAt(i), "$dd",
            (String)SDBAttribute.elementAt(i), "$vv",
            template);
}

//insert for one to one line dollar sign
for (int i=0; i<SDBAttribute.size()-1;i++)
{
    doubleInsert( (String)SDBAttribute.elementAt(i), "$vv", template);
}
//Keep the template clean in order to avoid next component
//propriety insert into this component's propriety
remove("$vv", template);
remove("$dd", template);

//Inserting the last variable and their type into template
singleInsert((String)SDBAttribute.lastElement(), "$v", template);

//insert for one to one line dollar sign
for (int i=0; i<ODSAttribute.size()-1;i++)
if ( !((String)convertValue.elementAt(i)).equals(" ") )
    doubleInsert( (String)ODSAttribute.elementAt(i), "$oo", template);

//Inserting the last variable and their type into template
if ( !((String)convertValue.lastElement()).equals("") )
    singleInsert((String)ODSAttribute.lastElement(), "$o", template);

//insert for one to one line dollar sign
for (int i=0; i<ODSAttribute.size()-1;i++)
if ( !((String)convertValue.elementAt(i)).equals(" ") )
    doubleInsert( (String)convertValue.elementAt(i), "$cc", template);

//Inserting the last variable and their type into template
if ( !((String)convertValue.lastElement()).equals(" ") )
    singleInsert((String)convertValue.lastElement(), "$c", template);

//Keep the template clean in order to avoid next component
//propriety insert into this component's propriety
remove("$oo", template);
remove("$cc", template);

//insert for one to one line dollar sign

```

```

for (int i=0; i<ODSattribute.size()-1;i++)
{
    if (template.indexOf(".sql")!=0 || template.indexOf(".SQL")!=0)
        doubleInsert((String)convertValue.elementAt(i), "$pp", template);

    else
    {
        if ( !((String)convertValue.elementAt(i)).equals(" ") )
            if (((String)convertValue.elementAt(i)).substring(0,2)).equals("to") )
                doubleInsert((String)convertValue.elementAt(i), "$pp", template);
            else
                doubleInsert( ": "+(String)convertValue.elementAt(i), "$pp", template);
        }
    }

//Inserting the last variable and their type into template
if ( !((String)convertValue.lastElement()).equals(" ") )
{
    if (template.indexOf(".sql")!=0 || template.indexOf(".SQL")!=0)
    {
        if (((String)convertValue.lastElement()).substring(0,2)).equals("to") )
            singleInsert((String)convertValue.lastElement(), "$p", template);
        else
            singleInsert((String)convertValue.lastElement(), "$p", template);
    }
    else
    {
        if (((String)convertValue.lastElement()).substring(0,2)).equals("to") )
            singleInsert((String)convertValue.lastElement(), "$p", template);
        else
            singleInsert(": "+(String)convertValue.lastElement(), "$p", template);
    }
}

remove("$pp", template);

//displayCode(template);
}
}

```

/*****
 Cleaning only one DW table creation program.

Algorithm:

- Step 1. Write functionName into \$ff in template
- Step 2. Insert creatingComponent into template in \$bb
- Step 3. Write functionName into \$f in template
- Step 4. if it is not the last variable
 - {
 - take one variable name from Vector
 - Write variableName at \$vv
 - Write variableDataType at \$aa
 - }
 - else
 - {
 - Write variableName at \$v
 - Write variableDataType at \$a
 - end of the program
 - }

Step 5. Repeat Step 4

*****/

```

public void cleaningGenerator(String ODStableName,
                             Vector ODSattribute,
                             Vector ODStype,
                             Vector ODScheck,
                             String userid,
                             String password)
{
    int key=-1, nonKey=-1;

```



```

//Defining the function name for the hasProperty $ff
String functionName = "cleaning"+ODStableName;

//Defining the output file name. For pro.C,
//it is file name plus dot pc
//Defining the output file name. For pro.C,
//it is file name plus dot pc
//String template = "creating.pc";
if (template3.equals(""))
    template3 = JOptionPane.showInputDialog("Please Input Cleaning Program Name");

/**Component setup**

//Determining if the 'include' is already in the template
//if this output file is available, add one more creating
//component into it, else, add main component and creating
//component into the template

if (search("#include", template3))
{
    for (int i=0; i<ODScheck.size(); i++)
    {
        if (((String)ODScheck.elementAt(i)).equals("key"))
            key=i;
        if (((String)ODScheck.elementAt(i)).equals("nonKey"))
            nonKey=i;
    }

singleInsert(userid, "$UID", template3);
singleInsert(password, "$PWD", template3);

//Insert for non key redundancy cleaning
if (nonKey>=0)
{
    append("nonKeyCleaningComponent", template3, template3);

    //insert the file name, maybe it is wrong
    doubleInsert( ODStableName+"NonKeyCleaning", "$ff", template3);

    //insert the MAX variable for the key
    singleInsert( ODStableName+"NonKeyCleaning", "$f", template3);

    //insert the MAX variable for the key
    singleInsert( (String)ODSattribute.elementAt(nonKey), "$s", template3);

    //Inserting new cursor name into template
    singleInsert("MAX"+ODStableName+"Cursor", "$c", template3);

    //Inserting new table name into template
    singleInsert("ODS"+ODStableName, "$ODSt", template3);

    for (int i=0; i<ODSattribute.size()-1; i++)
        if ( !((String)ODScheck.elementAt(i)).equals("key") &&
            !((String)ODScheck.elementAt(i)).equals("nonKey") )
            doubleInsert((String)ODSattribute.elementAt(i), "$vv", template3);

    singleInsert((String)ODSattribute.lastElement(), "$v", template3);

    remove("$vv", template3);
}

//Insert for key redundancy cleaning
if (key>=0)
{
    append("keyCleaningComponent", template3, template3);

    //insert the file name, maybe it is wrong
    doubleInsert( ODStableName+"KeyCleaning", "$ff", template3);
}

```

```

//insert the file name, maybe it is wrong
singleInsert( ODStableName+"KeyCleaning", "$F", template3);

//insert the MAX variable for the key
singleInsert( (String)ODSattribute.elementAt(key), "$s", template3);

//Inserting new cursor name into template
singleInsert("MAXNIUM"+ODStableName+"Cursor", "$c", template3);

//Inserting new cursor name into template
singleInsert(ODStableName+"Cursor", "$cc", template3);

//Inserting new table name into template
singleInsert("ODS"+ODStableName, "$ODSt", template3);

for (int i=0; i<ODSattribute.size()-1; i++)
    if ( !((String)ODScheck.elementAt(i)).equals("key"))
        doubleInsert((String)ODSattribute.elementAt(i), "$vv", template3);

singleInsert((String)ODSattribute.lastElement(), "$v", template3);

remove("$vv", template3);
}

append("cleaningComponent", template3, template3);
}
else
    append("cleaningComponent", "mainComponent", template3);

/**Proprity setup**

//Inserting new function into $ff proprity and keep one
//more $ff in this template in order to issue the location
//next new function inserted
doubleInsert(functionName,"$ff", template3);

//Inserting new function into $f proprity and no more $f left
singleInsert(functionName,"$f", template3);

//Inserting new cursor name into template
singleInsert("ODS"+ODStableName+"Cursor", "$ODSc", template3);

//Inserting new cursor name into template
singleInsert("DW"+ODStableName+"Cursor", "$DWc", template3);

//Inserting new table name into template
singleInsert("ODS"+ODStableName, "$ODSt", template3);

//insert ODStable name into template
singleInsert("DW"+ODStableName, "$DWt", template3);

//Inserting the different variable and their type into template
for (int i=0; i<ODSattribute.size(); i++)
if (template3.indexOf(".sql")!=0 || template3.indexOf(".SQL")!=0)
{
    if ( ((String)ODStype.elementAt(i)).equals("char") )
        tribleInsert( "varchar2(20)", "$ODSdd",
            "ODS"+(String)ODSattribute.elementAt(i), "$ODSvv",
            template3);
    else
        tribleInsert( (String)ODStype.elementAt(i), "$ODSdd",
            "ODS"+(String)ODSattribute.elementAt(i), "$ODSvv",
            template3);
}
else
{
    if ( ((String)ODStype.elementAt(i)).equals("char") )
        tribleInsert( (String)ODStype.elementAt(i), "$ODSdd",
            "ODS"+(String)ODSattribute.elementAt(i)+"[20]", "$ODSvv",

```

```

        template3);
    else
        tripleInsert( (String)ODStype.elementAt(i), "$ODSdd",
            "ODS"+(String)ODSattribute.elementAt(i), "$ODSvv",
            template3);
}

//Inserting the different variable and their type into template
for (int i=0; i<ODSattribute.size(); i++)
if (template3.indexOf(".sql")!=0 || template3.indexOf(".SQL")!=0)
{
    if ( ((String)ODStype.elementAt(i)).equals("char") )
        tripleInsert( "varchar2(20)", "$DWdd",
            "DW"+(String)ODSattribute.elementAt(i), "$DWvv",
            template3);
    else
        tripleInsert( (String)ODStype.elementAt(i), "$DWdd",
            "DW"+(String)ODSattribute.elementAt(i), "$DWvv",
            template3);
}
else
{
    if ( ((String)ODStype.elementAt(i)).equals("char") )
        tripleInsert( (String)ODStype.elementAt(i), "$DWdd",
            "DW"+(String)ODSattribute.elementAt(i)+"[20]", "$DWvv",
            template3);
    else
        tripleInsert( (String)ODStype.elementAt(i), "$DWdd",
            "DW"+(String)ODSattribute.elementAt(i), "$DWvv",
            template3);
}

//insert for one to one line dollar sign
for (int i=0; i<ODSattribute.size()-1; i++)
    doubleInsert( "ODS"+(String)ODSattribute.elementAt(i), "$ODSvv", template3);

//insert for one to one line dollar sign
for (int i=0; i<ODSattribute.size()-1; i++)
    doubleInsert( "DW"+(String)ODSattribute.elementAt(i), "$DWvv", template3);

//Keep the template clean in order to avoid next component
//propriety insert into this component's propriety
remove("$ODSvv", template3);
remove("$DWvv", template3);
remove("$ODSdd", template3);
remove("$DWdd", template3);

//Inserting the last variable and their type into template
singleInsert("ODS" + (String)ODSattribute.lastElement(), "$ODSv", template3);

//Inserting the last variable and their type into template
singleInsert("DW"+(String)ODSattribute.lastElement(), "$DWv", template3);

}

/*****
Loading only one DW table creation program.

Algorithm:
Step 1. Write functionName into $ff in template
Step 2. Insert creatingComponent into template in $bb
Step 3. Write functionName into $f in template
Step 4. if it is not the last variable
    {
        take one variable name from Vector
        Write variableName at $vv
        Write variableDataType at $aa
    }
else

```

```

        {
            Write variableName at $v
            Write variableDataType at $a
            end of the program
        }
Step 5. Repeat Step 4
*****/

public void loadingGenerator(String ODStableName,
                            Vector ODSattribute,
                            Vector ODStype,
                            String userid,
                            String password)
{
    //Defining the function name for the hasProprity $ff
    String functionName = "loading"+ODStableName;

    //Defining the output file name. For pro.C,
    //it is file name plus dot pc
    if (template4.equals(""))
        template4 = JOptionPane.showInputDialog("Please Input Loading Program Name");

    //***Component setup***

    //Determining if the 'include' is already in the template
    //if this output file is available, add one more creating
    //component into it, else, add main component and creating
    //component into the template
    if (search("#include", template4))
        append("loadingComponent", template4, template4);
    else
    {
        append("loadingComponent", "mainComponent", template4);

        singleInsert(userid, "$UID", template4);
        singleInsert(password, "$PWD", template4);
    }

    //***Proprity setup***

    //Inserting new function into $ff proprity and keep one
    //more $ff in this template in order to issue the location
    //next new function inserted
    doubleInsert(functionName, "$ff", template4);

    //Inserting new function into $f proprity and no more $f left
    singleInsert(functionName, "$f", template4);

    //Inserting new cursor name into template
    singleInsert("ODS"+ODStableName+"Cursor", "$ODSc", template4);

    //Inserting new table name into template
    singleInsert("ODS"+ODStableName, "$ODSt", template4);

    //insert ODStable name into template
    singleInsert("DW"+ODStableName, "$DWt", template4);

    //Inserting the different variable and their type into template
    for (int i=0; i<ODSattribute.size(); i++)
    if (template4.indexOf(".sql")!=0 || template4.indexOf(".SQL")!=0)
    {
        if ( ((String)ODStype.elementAt(i)).equals("char") )
            tripleInsert( "varchar2(20)", "$dd",
                (String)ODSattribute.elementAt(i), "$vv",
                template4);
    }
}

```

```

else
    tripleInsert( (String)ODStype.elementAt(i), "$dd",
                 (String)ODSattribute.elementAt(i), "$vv",
                 template4);
}
else
{
    if ( ((String)ODStype.elementAt(i)).equals("char") )
        tripleInsert( (String)ODStype.elementAt(i), "$dd",
                     (String)ODSattribute.elementAt(i)+"[20]", "$vv",
                     template4);
    else
        tripleInsert( (String)ODStype.elementAt(i), "$dd",
                     (String)ODSattribute.elementAt(i), "$vv",
                     template4);
}
//insert for one to one line dollar sign
for (int i=0; i<ODSattribute.size()-1; i++)
    doubleInsert((String)ODSattribute.elementAt(i), "$vv", template4);

//Inserting the last variable and their type into template
singleInsert((String)ODSattribute.lastElement(), "$v", template4);

//Keep the template clean in order to avoid next component
//propriety insert into this component's propriety
remove("$vv", template4);
remove("$dd", template4);

singleInsert("'" + ODStableName + "'", "$t", template4);
}

/*****
singleInsert the real function, single variable ect into right position

Algorithm
Step 1: Read onr line into string 'line'
Step 2: Check any mark in the line
        Yes: insert Content before the mark
Step 3: put this line into vector
Step 3: repeat Step 1
Step 4: Delete the old file based on file name
Step 5: create new file wilth the same name as old one
Step 6: Wirte all line from vector to file

*****/

void singleInsert(String content, String mark, String fileName)
{
    //Creating object line reader to read line by line
    LineNumberReader lineNumberReader;

    //Defining the target vector to fill the generated code
    Vector fileVector = new Vector();

    //Seperating the elements of one line by token
    Vector lineVector = new Vector();

    //Some temp parameters
    String line = new String("");
    String newLine = new String("");
    boolean moreLine = true;
    boolean hasPropriety = true;

    try
    {
        //Open tha target file template
        FileReader fileReader = new FileReader(fileName);

```

```

//Reading the file line by line
lineNumberReader = new LineNumberReader(fileReader);

//Loop to read every line from this file tempalte
do
{
    //reading one line from file to 'line'
    line = lineNumberReader.readLine().trim();

    //Empty line or not
    if (line.length() == 0)
    {
        //Add empty to target vector
        fileVector.addElement(" ");

        //Going back to reading next line
        continue;
    }
    else
    {
        //Symbol is for the search is any proprity in this line
        hasProprity = false;

        //Tokenize the line element
        lineVector = tokenize(line);

        //Checking every line to see the mark is inside or not
        for (int i=0; i<lineVector.size();i++)
        {
            if (((String)lineVector.elementAt(i)).equals(mark))
            {
                //When hasProprity is true, there is a proprity in this line
                hasProprity=true;

                //Insert into target vector
                lineVector.setElementAt(content,i);

                //Forming the whole line after the insert
                newLine="";
                for (int j=0; j<lineVector.size(); j++)
                    newLine=newLine+" "+(String)lineVector.elementAt(j);

                //when have property, add the new line into file but not old line
                fileVector.addElement(newLine);

                break;
            } //End if
        } //End for
    } //End else

    //If no proprity in this line, add the origen line into target vector
    if (hasProprity==false)
        fileVector.addElement(line);

}while(moreLine);

lineNumberReader.close();
}
catch (NullPointerException npe)
{
    moreLine = false;
}
catch (IOException e)
{
}

update(fileVector, fileName);
}

```

```

/*****
doubleInsert the real function, repeatable variable into right position
After insertion, also need keep the $ symbol in the next line. it must
be only one symbol in one line

```

Algorithm

```

Step 1: Read onr line into string 'line'
Step 2: Check any mark in the line
        Yes: insert Content before the mark
Step 3: put this line into vector
Step 3: repeat Step 1
Step 4: Delete the old file based on file name
Step 5: create new file wilth the same name as old one
Step 6: Wirte all line from vector to file

```

```

*****/

void doubleInsert(String content, String mark, String fileName)
{
    //Creating object line reader to read line by line
    LineNumberReader lineNumberReader;

    //Defining the target vector to fill the generated code
    Vector fileVector = new Vector();

    //Seperating the elements of one line by token
    Vector lineVector = new Vector();

    //Some temp parameters
    String line = new String("");
    String newLine = new String("");
    boolean moreLine = true;

    try
    {
        //Open tha target file template
        FileReader fileReader = new FileReader(fileName);

        //Reading the file line by line
        lineNumberReader = new LineNumberReader(fileReader);

        //Loop to read every line from this file tempalte
        do
        {
            //reading one line from file to 'line'
            line = lineNumberReader.readLine().trim();

            //Empty line or not
            if (line.length() == 0)
            {
                //Add empty to target vector
                fileVector.addElement(" ");

                //Going back to reading next line
                continue;
            }
            else
            {
                //Tokenize the line element
                lineVector = tokenize(line);

                //check it is declare or not. if it is declare, not do anything
                if (dollarNumber(lineVector)==1)
                {
                    //Checking every line to see the mark is inside or not
                    for (int i=0; i<lineVector.size(); i++)
                    //check it is the mark we want replace
                    if (((String)lineVector.elementAt(i)).equals(mark))

```

```

    {

        //Insert into target vector
        lineVector.setElementAt(content,i);

        //Forming the whole line after the insert
        newLine="";

        for (int j=0; j<lineVector.size(); j++)
            newLine=newLine+" "+(String)lineVector.elementAt(j);

        //add new line
        fileVector.addElement(newLine);

        break;
    } //End if
} //End if

//keep the repeatable line
fileVector.addElement(line);

} //End else
}while(moreLine); //End do

lineNumberReader.close();
} //End try

catch (NullPointerException npe)
{
    moreLine = false;
}
catch (IOException e)
{
}

update(fileVector, fileName);
}

/*****
tribeInsert the real function, repeatable variable into right position
After insertion, also need keep the $ symbol in the next line. it must
be only two symbol in one line

Algorithm
Step 1: Read onr line into string 'line'
Step 2: Check any mark in the line
        Yes: insert Content before the mark
Step 3: put this line into vector
Step 3: repeat Step 1
Step 4: Delete the old file based on file name
Step 5: create new file wilth the same name as old one
Step 6: Wirte all line from vector to file

*****/

void tribeInsert(String firstContent, String firstMark,
                String secondContent, String secondMark,
                String fileName)
{
    //Creating object line reader to read line by line
    LineNumberReader lineNumberReader;

    //Defining the target vector to fill the generated code
    Vector fileVector = new Vector();

```



```

//Seperating the elements of one line by token
Vector lineVector = new Vector();

//Some temp parameters
String line = new String("");
String newLine = new String("");
boolean moreLine = true;

try
{
//Open the target file template
FileReader fileReader = new FileReader(fileName);

//Reading the file line by line
LineNumberReader lineNumberReader = new LineNumberReader(fileReader);

//Loop to read every line from this file template
do
{
//reading one line from file to 'line'
line = lineNumberReader.readLine().trim();

//Empty line or not
if (line.length() == 0)
{
//Add empty to target vector
fileVector.addElement(" ");

//Going back to reading next line
continue;
}
else
{
//Tokenize the line element
lineVector = tokenize(line);

//check it is declare or not. if it is not declare, not do anything
if (dollarNumber(lineVector)==2)
{
boolean correctTribbleInsert=false;

//Checking every line to see the mark is inside or not
//need to replace twice
for (int i=0; i<lineVector.size(); i++)
{
if (((String)lineVector.elementAt(i)).equals(firstMark))
{
//Insert into target vector
lineVector.setElementAt(firstContent,i);
//Mark for it is tribble but not we need or not

correctTribbleInsert=true;
}
}

if (((String)lineVector.elementAt(i)).equals(secondMark))
{
//Insert into target vector
lineVector.setElementAt(secondContent,i);
//Mark for it is tribble but not we need or not
correctTribbleInsert=true;
}
}

if (correctTribbleInsert)
{
//Forming the whole line after the insert
newLine="";
for (int j=0; j<lineVector.size(); j++)

```

```

        newLine=newLine+" "+(String)lineVector.elementAt(j);

        //add new line
        fileVector.addElement(newLine);
    }
}

//keep the repeatable line
fileVector.addElement(line);

}
}while(moreLine);

lineNumberReader.close();
}
catch (NullPointerException npe)
{
    moreLine = false;
}
catch (IOException e)
{
}

update(fileVector, fileName);
}

```

```

/*****

```

```

Append a file into the end of another file

```

```

Algorithm

```

```

Step 1: open target file

```

```

Step 2: open source file

```

```

Step 3: read one line from source file

```

```

Step 4: add into target file

```

```

Step 5: repeat step 3/4 until the end of source file

```

```

Step 6: close both file

```

```

*****/

```

```

public void append(String sourceFile, String targetFile, String procFile)

```

```

{
    LineNumberReader lineNumberReader, lineReader;
    Vector targetFileVector = new Vector();
    boolean moreLine = true;
    String line = new String("");

    try
    {
        FileReader targetFileReader = new FileReader(targetFile);
        lineNumberReader = new LineNumberReader(targetFileReader);

```

```

        do
        {
            line = lineNumberReader.readLine().trim();
            targetFileVector.addElement(line);
        }while(moreLine);

```

```

    }
    catch (NullPointerException npe)
    {
        moreLine = false;
    }
    catch (IOException e)
    {
        System.out.println("IOException found by LiuYi at 1");
    }
}

```

```

}

moreLine=true;

try
{

    FileReader sourceFileReader = new FileReader(sourceFile);
    lineNumberReader = new LineNumberReader(sourceFileReader);

    do
    {
        line = lineNumberReader.readLine().trim();
        targetFileVector.addElement(line);
    }while(moreLine);

}
catch (NullPointerException npe)
{
    moreLine = false;
}
catch (IOException e)
{
    System.out.println("IOException found by LiuYi at 2");
}

update(targetFileVector, procFile);
}

```

```

/*****
update is for update the file from Vector

```

Algorithm

Step 1: create file with same name as target
Step 2: write line by line from Vector into this file
Step 3: read this file and displayCode for demo

```

*****/
public void update(Vector fileVector, String fileName)
{

    FileOutputStream outputFile;

    try
    {
        outputFile = new FileOutputStream(fileName);
        PrintStream output = new PrintStream(outputFile);

        for (int i=0; i<fileVector.size(); i++)
            output.println(fileVector.elementAt(i));

        output.close();
    }
    catch (NullPointerException npe)
    {
        System.out.println("IOException found by LiuYi at 3");
    }
    catch (IOException e)
    {
        System.out.println("IOException found by LiuYi at 4");
    }
}

```

```

/*****
Remove a line with special Mark from a file

```

Algorithm

- Step 1: open file
- Step 2: read one line from file
- Step 3: tokenize the line
- Step 4: is the token a mark
yes: remove this line
- Step 5: repeat step 3/4 until the end of the file

```
*****/  
  
public void remove(String mark, String fileName)  
{  
    LineNumberReader lineNumberReader;  
    Vector fileVector = new Vector();  
    Vector lineVector = new Vector();  
    String line = new String("");  
    String newLine = new String("");  
    boolean moreLine = true;  
    boolean hasProprity = true;  
  
    try  
    {  
        FileReader fileReader = new FileReader(fileName);  
        lineNumberReader = new LineNumberReader(fileReader);  
  
        do  
        {  
            line = lineNumberReader.readLine().trim();  
  
            if (line.length() == 0)  
            {  
                fileVector.addElement(" ");  
                continue;  
            }  
            else  
            {  
                hasProprity = false;  
  
                lineVector = tokenize(line);  
  
                for (int i=0; i<lineVector.size(); i++)  
                {  
                    if (((String)lineVector.elementAt(i)).equals(mark))  
                    {  
                        hasProprity=true;  
                        break;  
                    }  
                }  
            }  
  
            if (hasProprity==false)  
                fileVector.addElement(line);  
  
        }while(moreLine);  
  
        lineNumberReader.close();  
    }  
    catch (NullPointerException npe)  
    {  
        moreLine = false;  
    }  
    catch (IOException e)  
    {  
        System.out.println("IOException found by LiuYi at 5");  
    }  
  
    update(fileVector, fileName);  
}  
  
/*****
```

searching special mark from a file

Algorithm

Step 1: open file
Step 2: read one line from file
Step 3: tokenize the line
Step 4: is the token a mark
 return true
Step 5: repeat step 3/4 until the end of the file
Step 6: return false

...../

```
public boolean search(String mark, String fileName)
{
    LineNumberReader lineNumberReader;
    Vector fileVector = new Vector();
    Vector lineVector = new Vector();
    String line = new String("");
    boolean moreLine = true, result = false;

    try
    {
        FileReader fileReader = new FileReader(fileName);
        lineNumberReader = new LineNumberReader(fileReader);

        do
        {
            line = lineNumberReader.readLine().trim();

            lineVector = tokenize(line);

            for (int i=0; i<lineVector.size(); i++)
            {
                if (((String)lineVector.elementAt(i)).equals(mark))
                {
                    result=true;
                    break;
                }
            }
        }while(moreLine);

        lineNumberReader.close();

    }
    catch (NullPointerException npe)
    {
        moreLine = false;
    }
    catch (IOException e)
    {
    }

    return result;
}
```

...../displayCode a file contents

Algorithm

Step 1: open file
Step 2: read one line from the file
Step 3: output to output device
Step 4: repeat step 3/4 until the end of source file
Step 5: close both file

```

*****/
public void displayCode(String fileName)
{
    LineNumberReader lineNumberReader;
    boolean moreLine = true;
    String line = new String("");

    try
    {
        FileReader targetFileReader = new FileReader(fileName);
        lineNumberReader = new LineNumberReader(targetFileReader);

        do
        {
            line = lineNumberReader.readLine().trim();

        }while(moreLine);

    }
    catch (NullPointerException npe)
    {
        moreLine = false;
    }
    catch (IOException e)
    {
        System.out.println("IOException when no more line from file found by LiuYi at 7");
    }
}

```

```

/*****
Take a line vector as input and check the vector
to see more propriety in the same line
*****/

```

```

public int dollarNumber(Vector input)
{
    int count=0;

    for(int i=0; i<input.size(); i++)
        if ( ((String)input.elementAt(i)).equals("$aa") ||
            ((String)input.elementAt(i)).equals("$dd") ||
            ((String)input.elementAt(i)).equals("$ff") ||
            ((String)input.elementAt(i)).equals("$vv") ||
            ((String)input.elementAt(i)).equals("$oo") ||
            ((String)input.elementAt(i)).equals("$ODSvv") ||
            ((String)input.elementAt(i)).equals("$ODSdd") ||
            ((String)input.elementAt(i)).equals("$DWvv") ||
            ((String)input.elementAt(i)).equals("$DWdd") ||
            ((String)input.elementAt(i)).equals("$cc") ||
            ((String)input.elementAt(i)).equals("$pp") ||
            ((String)input.elementAt(i)).equals("$ss") )
            count++;

    return count;
}

```

```

/*****
Take a line string as input and output the vector
which includes the string element in this line
*****/

```

```

public Vector tokenize(String inString)
{

```

```

Vector tokenVector = new Vector();
String tokenString = new String();
StringTokenizer st = new StringTokenizer(inString);

while (st.hasMoreTokens())
{
    tokenString = st.nextToken();
    tokenVector.addElement(tokenString);
}
return tokenVector;
}

/*****
Replace a line with special line number from a file

Algorithm
Step 1: open file
Step 2: read one line from file
Step 3: tokenize the line
Step 4: is the token a mark
         yes: remove this line
Step 5: repeat step 3/4 until the end of the file
*****/

public void replace(String fileName, int lineNumber, String lineContent)
{
    LineNumberReader lineNumberReader;
    String line;
    Vector fileVector = new Vector();
    boolean moreLine = true;
    int counter=1;

    try
    {
        FileReader fileReader = new FileReader(fileName);
        lineNumberReader = new LineNumberReader(fileReader);

        do
        {
            line = lineNumberReader.readLine().trim();

            if (lineNumber!=counter)
            {
                fileVector.addElement(line);
            }
            else
            {
                fileVector.addElement(lineContent);
            }
            counter++;
        }while(moreLine);

        lineNumberReader.close();
    }
    catch (NullPointerException npe)
    {
        moreLine = false;
    }
    catch (IOException e)
    {
        System.out.println("IOException found by LiuYi at 5");
    }

    update(fileVector, fileName);
}

/*****

```

removeLine a line with special line number from a file

Algorithm

Step 1: open file
Step 2: read one line from file
Step 3: tokenize the line
Step 4: is the token a mark
 yes: remove this line
Step 5: repeat step 3/4 until the end of the file

```
*****/
public void removeLine(String fileName, int lineNumber)
{
    LineNumberReader lineNumberReader;
    String line;
    Vector fileVector = new Vector();
    boolean moreLine = true;
    int counter=1;

    try
    {
        FileReader fileReader = new FileReader(fileName);
        lineNumberReader = new LineNumberReader(fileReader);

        do
        {
            line = lineNumberReader.readLine().trim();

            if (lineNumber!=counter)
            {
                fileVector.addElement(line);
            }

            counter++;

        }while(moreLine);

        lineNumberReader.close();
    }
    catch (NullPointerException npe)
    {
        moreLine = false;
    }
    catch (IOException e)
    {
        System.out.println("IOException found by LiuYi at 5");
    }

    update(fileVector, fileName);
}
} //End of Class
```

(14) mainGenerator.java

```
import java.util.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class mainGenerator
{
    String currentSDBname = new String();
    Vector currentSDBtableName = new Vector();
```



```

Vector currentODStableName = new Vector();
Vector currentSDBattribute = new Vector();
Vector currentSDBdataType = new Vector();
Vector currentODSattribute = new Vector();
Vector currentODSdataType = new Vector();
Vector currentConstraint = new Vector();
Vector currentTemp = new Vector();
int position=0;
String template="";

//some global variables for UID/PWD
String UID = new String();
String PWD = new String();

public mainGenerator(Vector inputSDBname,
                    Vector inputSDBtableName,
                    Vector inputODStableName,
                    Vector inputSDBattribute,
                    Vector inputSDBdataType,
                    Vector inputODSattribute,
                    Vector inputODSdataType,
                    Vector inputConstraint,
                    String UID,
                    String PWD)
{
    this.UID=UID;
    this.PWD=PWD;

    //delete the current information from input vector
    //so, keep the input vector is un_integrated
    int k=0, currentSize=inputODStableName.size();
    while(currentSize>0 && k<currentSize )
    {
        if ( !((String)inputODStableName.elementAt(k)).equals("fact") )
        {
            inputSDBname.removeElementAt(k);
            inputSDBtableName.removeElementAt(k);
            inputODStableName.removeElementAt(k);
            inputSDBattribute.removeElementAt(k);
            inputODSattribute.removeElementAt(k);
            inputSDBdataType.removeElementAt(k);
            inputODSdataType.removeElementAt(k);
            inputConstraint.removeElementAt(k);
        }
        else
        {
            k++;
        }
        currentSize=inputODStableName.size();
    }

    for (int i=0; i<inputSDBname.size(); i++)
    {
        //data type and measure only replace the current value
        String ODStypeString = new String();
        String SDBtypeString = new String();

        //get both data type
        ODStypeString = (String) inputODSdataType.elementAt(i);
        SDBtypeString = (String) inputSDBdataType.elementAt(i);

        //check both are same or not, if it is same, it don't need integrating
        //else need integrating
        if ( !ODStypeString.equals(SDBtypeString) && !SDBtypeString.equals(" ") )
        {
            //for different case to convert
            //convert from char to number
            if ( (ODStypeString.substring(0,6)).equals("number") &&
                (SDBtypeString.substring(0,8)).equals("varchar2") )
            {
                if (template.indexOf(".sql")!=0 || template.indexOf(".SQL")!=0)

```

```

        inputConstraint.setElementAt("to_number("+(String)inputSDBattribute.elementAt(i)+")", i);
    else
        inputConstraint.setElementAt("to_number(:"+(String)inputSDBattribute.elementAt(i)+")", i);
    }
    //convert from number to char
    else if ( (ODStypeString.substring(0,8)).equals("varchar2") &&
        (SDBtypeString.substring(0,6)).equals("number") )
    {
        if (template.indexOf(".sql")!=0 || template.indexOf(".SQL")!=0)
            inputConstraint.setElementAt("to_char("+(String)inputSDBattribute.elementAt(i)+")",i);
        else
            inputConstraint.setElementAt("to_char(:"+(String)inputSDBattribute.elementAt(i)+")",i);
    }
    //convert from date to char
    else if ( (ODStypeString.substring(0,8)).equals("varchar2") &&
        (SDBtypeString.substring(0,4)).equals("date") )
    {
        if (template.indexOf(".sql")!=0 || template.indexOf(".SQL")!=0)
            inputConstraint.setElementAt("to_char("+(String)inputSDBattribute.elementAt(i)+", 'Month dd,
YYYY')",i);
        else
            inputConstraint.setElementAt("to_char(:"+(String)inputSDBattribute.elementAt(i)+", 'Month dd,
YYYY')",i);
    }
    //convert char to date
    else if ( (ODStypeString.substring(0,4)).equals("date") &&
        (SDBtypeString.substring(0,8)).equals("varchar2") )
    {
        if (template.indexOf(".sql")!=0 || template.indexOf(".SQL")!=0)
            inputConstraint.setElementAt("to_date("+(String)inputSDBattribute.elementAt(i)+", 'month dd
YYYY')",i);
        else
            inputConstraint.setElementAt("to_date(:"+(String)inputSDBattribute.elementAt(i)+", 'Month dd,
YYYY')",i);
    }
    //Can not convert by SQL language
    else
        //warning type conversion error
        JOptionPane.showMessageDialog(null, "Illegal data type conversion\nSource data type"+
            SDBtypeString.substring(0,8)+
            "\ncan not convert to target data type"+
            ODStypeString.substring(0,4)+
            "\nPlease contact Liu Yi at University of Windsor\n at " +
            "519-253-3000-3003\nfor future helping");
    }
}

String type = new String();

for (int i=0; i<inputSDBdataType.size(); i++)
{
    type = (String)inputSDBdataType.elementAt(i);

    if (type.equals(" "))
    {
        continue;
    }
    else if ( (type.substring(0,4)).equals("date") )
    {
        inputSDBdataType.setElementAt("date",i);
    }
    else if ( ((type.substring(0,6)).equals("number")) && (type.length()>10) )
    {
        inputSDBdataType.setElementAt("float",i);
    }
    else if ( ((type.substring(0,6)).equals("number")) && (type.length() <= 10) )
    {
        inputSDBdataType.setElementAt("int",i);
    }
    else if ( (type.substring(0,8)).equals("varchar2") )
    {

```

```

    inputSDBdataType.setElementAt("char",i);
}
else
{
    System.out.println("Illegal data type conversion by LiuYi from Data type integration 1");
}
}

while(inputSDBname.size()>0)
{
    //set current source name

    currentSDBname = (String) inputSDBname.elementAt(0);

    //choose the first SDB, put the information
    //about this source into current vector
    for (int i=0; i<inputSDBname.size(); i++)
        if (((String)inputSDBname.elementAt(i)).equals(currentSDBname) )
        {
            currentODStableName.addElement((String) inputODStableName.elementAt(i));
            currentODSattribute.addElement((String) inputODSattribute.elementAt(i));
            currentODSdataType.addElement( (String) inputODSdataType.elementAt(i));

            currentSDBtableName.addElement((String) inputSDBtableName.elementAt(i));
            currentSDBattribute.addElement((String) inputSDBattribute.elementAt(i));
            currentSDBdataType.addElement( (String) inputSDBdataType.elementAt(i));

            currentConstraint.addElement((String) inputConstraint.elementAt(i));
        }

    //delete the current information from input vector
    //so, keep the input vector is un_integrated
    while (inputSDBname.indexOf(currentSDBname)!=-1)
    {
        position=(int) inputSDBname.indexOf(currentSDBname);

        inputSDBname.removeElementAt(position);
        inputSDBtableName.removeElementAt(position);
        inputODStableName.removeElementAt(position);
        inputSDBattribute.removeElementAt(position);
        inputODSattribute.removeElementAt(position);
        inputSDBdataType.removeElementAt(position);
        inputODSdataType.removeElementAt(position );
        inputConstraint.removeElementAt(position);
    }

    //Generation the code
    //Defining the fuction name for the hasProprity $ff
    String functionName = "extracting"+currentSDBname+"Fact";

    //Defining the output file name. For pro.C,
    //it is file name plus dot pc
    if (template.equals(""))
        template = JOptionPane.showInputDialog("Please Input Creation Program Name");

    /***Component setup***/

    //Determining if the 'include' is already in the template
    //if this output file is aviable, add one more creating
    //component into it, else, add main component and creating
    //component into the template
    if (search("#include", template))
        append("factComponent", template, template);
    else
        append("factComponent", "mainComponent", template);

    singleInsert(UID, "$UID", template);
    singleInsert(PWD, "$PWD", template);
}

```

```

/**Proprity setup**

//Inserting new function into $ff proprity and keep one
//more $ff in this template in order to issue the location
//next new function inserted
doubleInsert(functionName,"$ff", template);

//Inserting new function into $f proprity and no more $f left
singleInsert(functionName,"$f", template);

//Inserting new cursor name into template
singleInsert(currentSDBname+"FactCursor", "$C", template);

//insert ODStable name into template
singleInsert("ODS"+(String)currentODStableName.elementAt(0), "$ODSt", template);

//Inserting the different variable and their type into template
for (int i=0; i<currentSDBattribute.size();i++)
if (template.indexOf(".sql")!=0 || template.indexOf(".SQL")!=0)
{
if ( !((String)currentSDBattribute.elementAt(i)).equals(" ") )
{
if ( ((String)currentODSdataType.elementAt(i)).equals("varchar2(20)") )
{
tribeInsert( (String)currentODSdataType.elementAt(i), "$dd",
(String)currentSDBattribute.elementAt(i), "$vv",
template);
}
else
{
tribeInsert( (String)currentSDBdataType.elementAt(i), "$dd",
(String)currentSDBattribute.elementAt(i), "$vv",
template);
}
}
}
else
{
if ( !((String)currentSDBattribute.elementAt(i)).equals(" ") )
{
if ( ((String)currentODSdataType.elementAt(i)).equals("varchar2(20)") )
{
tribeInsert( (String)currentSDBdataType.elementAt(i), "$dd",
(String)currentSDBattribute.elementAt(i)+"[20]", "$vv",
template);
}
else
{
tribeInsert( (String)currentSDBdataType.elementAt(i), "$dd",
(String)currentSDBattribute.elementAt(i), "$vv",
template);
}
}
}
}

//insert for one to one line dollar sign
for (int i=0; i<currentSDBattribute.size()-1;i++)
{
if ( !((String)currentSDBattribute.elementAt(i)).equals(" ") )
{
doubleInsert( (String)currentSDBattribute.elementAt(i), "$vv", template);
doubleInsert(
(String)currentSDBtableName.elementAt(i)+"."+
(String)currentSDBattribute.elementAt(i), "$cc",
template);
}
}

//Keep the template clean in order to avoid next component
//proprity insert into this component's proprity
remove("$vv", template);
remove("$dd", template);

```

```

remove("$cc", template);

//Inserting the last variable and their type into template
singleInsert((String)currentSDBAttribute.lastElement(), "$v", template);
singleInsert((String)currentSDBTableName.lastElement()+"."+
(String)currentSDBAttribute.lastElement(), "$c", template);

//insert for one to one line dollar sign
for (int i=0; i<currentODSAttribute.size()-1;i++)
    doubleInsert( (String)currentODSAttribute.elementAt(i), "$oo", template);
remove("$oo", template);

//Inserting the last variable and their type into template
singleInsert((String)currentODSAttribute.lastElement(), "$o", template);

//Keep the template clean in order to avoid next component
//propriety insert into this component's propriety

//***simple integration here

//insert for one to one line dollar sign
for (int i=0; i<currentConstraint.size()-1;i++)
{
    if (template.indexOf(".sql")!=0 || template.indexOf(".SQL")!=0)
    {
        if (!(String)currentConstraint.elementAt(i).equals(""))
            doubleInsert((String)currentConstraint.elementAt(i), "$pp", template);
        else
            doubleInsert((String)currentSDBAttribute.elementAt(i), "$pp", template);
    }
    else
    {
        if (!(String)currentConstraint.elementAt(i).equals(""))
            doubleInsert((String)currentConstraint.elementAt(i), "$pp", template);
        else
            doubleInsert(": "+(String)currentSDBAttribute.elementAt(i), "$pp", template);
    }
}
remove("$pp", template);

if (template.indexOf(".sql")!=0 || template.indexOf(".SQL")!=0)
    singleInsert((String)currentSDBAttribute.lastElement(), "$p", template);
else
    singleInsert(": "+(String)currentSDBAttribute.lastElement(), "$p", template);

while(currentSDBTableName.size(>0)
{
    //set current source name
    String temp= (String) currentSDBTableName.elementAt(0);

    if (!temp.equals(" "))
    {
        currentTemp.addElement( temp );
    }
    //delete the current information from input vector
    //so, keep the input vector is un_integrated
    while (currentSDBTableName.indexOf(temp)!=-1)
    {
        position=(int) currentSDBTableName.indexOf(temp);

        currentSDBTableName.removeElementAt(position);
        currentODStableName.removeElementAt(position);
        currentSDBAttribute.removeElementAt(position);
        currentODSAttribute.removeElementAt(position);
        currentSDBdataType.removeElementAt(position);
        currentODSdataType.removeElementAt(position );
        currentConstraint.removeElementAt(position);
    }
}
}

```

```

for (int i=0; i<currentTemp.size()-1;i++)
{
    doubleInsert( (String)currentTemp.elementAt(i), "$tt", template);
}
remove("$tt", template);

//Inserting the last variable and their type into template
singleInsert((String)currentTemp.lastElement(), "$t", template);

//displayCode(template);
//remove output data
currentSDBname = null;
currentTemp.removeAllElements();
currentSDBtableName.removeAllElements();
currentODStableName.removeAllElements();
currentSDBattribute.removeAllElements();
currentSDBdataType.removeAllElements();
currentODSattribute.removeAllElements();
currentODSdataType.removeAllElements();
currentConstraint.removeAllElements();
}
}

/*****
singleInsert the real function, single variable ect into right position

Algorithm
Step 1: Read onr line into string 'line'
Step 2: Check any mark in the line
        Yes: insert Content before the mark
Step 3: put this line into vector
Step 3: repeat Step 1
Step 4: Delete the old file based on file name
Step 5: create new file wilth the same name as old one
Step 6: Wirte all line from vector to file

*****/

public void singleInsert(String content, String mark, String fileName)
{
    //Creating object line reader to read line by line
    LineNumberReader lineNumberReader;

    //Defining the target vector to fill the generated code
    Vector fileVector = new Vector();

    //Seperating the elements of one line by token
    Vector lineVector = new Vector();

    //Some temp parameters
    String line = new String("");
    String newLine = new String("");
    boolean moreLine = true;
    boolean hasProprity = true;

    try
    {
        //Open tha target file template
        FileReader fileReader = new FileReader(fileName);

        //Reading the file line by line
        lineNumberReader = new LineNumberReader(fileReader);

        //Loop to read every line from this file tempalte
        do
        {
            //reading one line from file to 'line'
            line = lineNumberReader.readLine().trim();

            //Empty line or not
            if (line.length() == 0)

```

```

{
//Add empty to target vector
fileVector.addElement(" ");

//Going back to reading next line
continue;
}
else
{
//Symbol is for the search is any property in this line
hasProperty = false;

//Tokenize the line element
lineVector = tokenize(line);

//Checking every line to see the mark is inside or not
for (int i=0; i<lineVector.size();i++)
{
if (((String)lineVector.elementAt(i)).equals(mark))
{
//When hasProperty is true, there is a property in this line
hasProperty=true;

//Insert into target vector
lineVector.setElementAt(content,i);

//Forming the whole line after the insert
newLine="";
for (int j=0; j<lineVector.size(); j++)
newLine=newLine+" "+(String)lineVector.elementAt(j);

//when have property, add the new line into file but not old line
fileVector.addElement(newLine);

break;
} //End if
} //End for
} //End else

//If no property in this line, add the origin line into target vector
if (hasProperty==false)
fileVector.addElement(line);

}while(moreLine);

lineNumberReader.close();
}
catch (NullPointerException npe)
{
moreLine = false;
}
catch (IOException e)
{
}

update(fileVector, fileName);
}

```

```

/*****
doubleInsert the real function, repeatable variable into right position
After insertion, also need keep the $ symbol in the next line. it must
be only one symbol in one line

```

Algorithm

```

Step 1: Read onr line into string 'line'
Step 2: Check any mark in the line
        Yes: insert Content before the mark
Step 3: put this line into vector

```

Step 3: repeat Step 1
 Step 4: Delete the old file based on file name
 Step 5: create new file with the same name as old one
 Step 6: Write all line from vector to file

```

...../

public void doubleInsert(String content, String mark, String fileName)
{
    //Creating object line reader to read line by line
    LineNumberReader lineNumberReader;

    //Defining the target vector to fill the generated code
    Vector fileVector = new Vector();

    //Separating the elements of one line by token
    Vector lineVector = new Vector();

    //Some temp parameters
    String line = new String("");
    String newLine = new String("");
    boolean moreLine = true;

    try
    {
        //Open the target file template
        FileReader fileReader = new FileReader(fileName);

        //Reading the file line by line
        lineNumberReader = new LineNumberReader(fileReader);

        //Loop to read every line from this file template
        do
        {
            //reading one line from file to 'line'
            line = lineNumberReader.readLine().trim();

            //Empty line or not
            if (line.length() == 0)
            {
                //Add empty to target vector
                fileVector.addElement(" ");

                //Going back to reading next line
                continue;
            }
            else
            {
                //Tokenize the line element
                lineVector = tokenize(line);

                //check it is declare or not. if it is declare, not do anything
                if (dollarNumber(lineVector)==1)
                {
                    //Checking every line to see the mark is inside or not
                    for (int i=0; i<lineVector.size(); i++)
                    //check it is the mark we want replace
                    if (((String)lineVector.elementAt(i)).equals(mark))
                    {
                        //Insert into target vector
                        lineVector.setElementAt(content,i);

                        //Forming the whole line after the insert
                        newLine="";

                        for (int j=0; j<lineVector.size(); j++)
                            newLine=newLine+" "+(String)lineVector.elementAt(j);

                        //add new line
                    }
                }
            }
        }
        while (moreLine);
    }
    catch (IOException e)
    {
        e.printStackTrace();
    }
}

```



```

        fileVector.addElement(newLine);

        break;
    } //End if
} //End if

//keep the repeatable line
fileVector.addElement(line);

} //End else
}while(moreLine); //End do

lineNumberReader.close();
} //End try

catch (NullPointerException npe)
{
    moreLine = false;
}
catch (IOException e)
{
}

update(fileVector, fileName);
}

/*****
tribleInsert the real function, repeatable variable into right position
After insertion, also need keep the $ symbol in the next line. it must
be only two symbol in one line

Algorithm
Step 1: Read onr line into string 'line'
Step 2: Check any mark in the line
        Yes: insert Content before the mark
Step 3: put this line into vector
Step 3: repeat Step 1
Step 4: Delete the old file based on file name
Step 5: create new file wilth the same name as old one
Step 6: Wirte all line from vector to file

*****/

void tribleInsert(String firstContent, String firstMark,
                  String secondContent, String secondMark,
                  String fileName)
{
    //Creating object line reader to read line by line
    LineNumberReader lineNumberReader;

    //Defining the target vector to fill the generated code
    Vector fileVector = new Vector();

    //Seperating the elements of one line by token
    Vector lineVector = new Vector();

    //Some temp parameters
    String line = new String("");
    String newLine = new String("");
    boolean moreLine = true;

    try
    {
        //Open tha target file template
        FileReader fileReader = new FileReader(fileName);

```

```

//Reading the file line by line
lineNumberReader = new LineNumberReader(fileReader);

//Loop to read every line from this file tempalte
do
{
//reading one line from file to 'line'
line = lineNumberReader.readLine().trim();

//Empty line or not
if (line.length() == 0)
{
//Add empty to target vector
fileVector.addElement(" ");

//Going back to reading next line
continue;
}
else
{
//Tokenize the line element
lineVector = tokenize(line);

//check it is declare or not. if it is not declare, not do anything
if (dollarNumber(lineVector)==2)
{
boolean correctTribleInsert=false;

//Checking every line to see the mark is inside or not
//need to replace twice
for (int i=0; i<lineVector.size(); i++)
{
if (((String)lineVector.elementAt(i)).equals(firstMark))
{
//Insert into target vector
lineVector.setElementAt(firstContent,i);
//Mark for it is trible but not we need or not

correctTribleInsert=true;
}

if (((String)lineVector.elementAt(i)).equals(secondMark))
{
//Insert into target vector
lineVector.setElementAt(secondContent,i);
//Mark for it is trible but not we need or not
correctTribleInsert=true;
}
}
}

if (correctTribleInsert)
{
//Forming the whole line after the insert
newLine="";
for (int j=0; j<lineVector.size(); j++)
newLine=newLine+" "+(String)lineVector.elementAt(j);

//add new line
fileVector.addElement(newLine);
}
}

//keep the repeatable line
fileVector.addElement(line);
}
}while (moreLine);

```

```

        lineNumberReader.close();
    }
    catch (NullPointerException npe)
    {
        moreLine = false;
    }
    catch (IOException e)
    {
    }

    update(fileVector, fileName);
}

```

```

/*****
Append a file into the end of another file

```

```

Algorithm
Step 1: open target file
Step 2: open source file
Step 3: read one line from source file
Step 4: add into target file
Step 5: repeat step 3/4 until the end of source file
Step 6: close both file

```

```

*****/

```

```

public void append(String sourceFile, String targetFile, String procFile)
{
    LineNumberReader lineNumberReader, lineReader;
    Vector targetFileVector = new Vector();
    boolean moreLine = true;
    String line = new String("");

    try
    {
        FileReader targetFileReader = new FileReader(targetFile);
        lineNumberReader = new LineNumberReader(targetFileReader);

        do
        {
            line = lineNumberReader.readLine().trim();
            targetFileVector.addElement(line);
        }while(moreLine);

    }
    catch (NullPointerException npe)
    {
        moreLine = false;
    }
    catch (IOException e)
    {
        System.out.println("IOException found by LiuYi at 1");
    }

    moreLine=true;

    try
    {

        FileReader sourceFileReader = new FileReader(sourceFile);
        lineReader = new LineNumberReader(sourceFileReader);

        do
        {

```

```

        line = lineReader.readLine().trim();
        targetFileVector.addElement(line);
    }while(moreLine);

}
catch (NullPointerException npe)
{
    moreLine = false;
}
catch (IOException e)
{
    System.out.println("IOException found by LiuYi at 2");
}

update(targetFileVector, procFile);
}

```

```

/*****
update is for update the file from Vector

```

Algorithm

Step 1: create file with same name as target
Step 2: write line by line from Vector into this file
Step 3: read this file and displayCode for demo

```

*****/
public void update(Vector fileVector, String fileName)
{

```

```

    FileOutputStream outputFile;

    try
    {
        outputFile = new FileOutputStream(fileName);
        PrintStream output = new PrintStream(outputFile);

        for (int i=0; i<fileVector.size(); i++)
            output.println(fileVector.elementAt(i));

        output.close();
    }
    catch (NullPointerException npe)
    {
        System.out.println("IOException found by LiuYi at 3");
    }
    catch (IOException e)
    {
        System.out.println("IOException found by LiuYi at 4");
    }
}

```

```

/*****
Remove a line with special Mark from a file

```

Algorithm

Step 1: open file
Step 2: read one line from file
Step 3: tokenize the line
Step 4: is the token a mark
 yes: remove this line
Step 5: repeat step 3/4 until the end of the file

```

*****/
public void remove(String mark, String fileName)
{

```

```

LineNumberReader lineNumberReader;
Vector fileVector = new Vector();
Vector lineVector = new Vector();
String line = new String("");
String newLine = new String("");
boolean moreLine = true;
boolean hasProprity = true;

try
{
    FileReader fileReader = new FileReader(fileName);
    lineNumberReader = new LineNumberReader(fileReader);

    do
    {
        line = lineNumberReader.readLine().trim();

        if (line.length() == 0)
        {
            fileVector.addElement(" ");
            continue;
        }
        else
        {
            hasProprity = false;

            lineVector = tokenize(line);

            for (int i=0; i<lineVector.size(); i++)
            {
                if (((String)lineVector.elementAt(i)).equals(mark))
                {
                    hasProprity=true;
                    break;
                }
            }
        }

        if (hasProprity==false)
            fileVector.addElement(line);

    }while(moreLine);

    lineNumberReader.close();
}
catch (NullPointerException npe)
{
    moreLine = false;
}
catch (IOException e)
{
    System.out.println("IOException found by LiuYi at 5");
}

update(fileVector, fileName);
}

/*****
searching special mark from a file

Algorithm
Step 1: open file
Step 2: read one line from file
Step 3: tokenize the line
Step 4: is the token a mark
         return true
Step 5: repeat step 3/4 until the end of the file
Step 6: return flase

*****/

```

```

public boolean search(String mark, String fileName)
{
    LineNumberReader lineNumberReader;
    Vector fileVector = new Vector();
    Vector lineVector = new Vector();
    String line = new String("");
    boolean moreLine = true, result = false;

    try
    {
        FileReader fileReader = new FileReader(fileName);
        lineNumberReader = new LineNumberReader(fileReader);

        do
        {
            line = lineNumberReader.readLine().trim();

            lineVector = tokenize(line);

            for (int i=0; i<lineVector.size(); i++)
            {
                if ((String)lineVector.elementAt(i).equals(mark))
                {
                    result=true;
                    break;
                }
            }
        }while(moreLine);

        lineNumberReader.close();

    }
    catch (NullPointerException npe)
    {
        moreLine = false;
    }
    catch (IOException e)
    {
    }

    return result;
}

```

```

/*****
displayCode a file contents

```

```

Algorithm
Step 1: open file
Step 2: read one line from the file
Step 3: output to output device
Step 4: repeat step 3/4 until the end of source file
Step 5: close both file

```

```

*****/

```

```

public void displayCode(String fileName)
{
    LineNumberReader lineNumberReader;
    boolean moreLine = true;
    String line = new String("");

    try
    {
        FileReader targetFileReader = new FileReader(fileName);

```

```

lineNumberReader = new LineNumberReader(targetFileReader);

do
{
    line = lineNumberReader.readLine().trim();
}while(!moreLine);

}
catch (NullPointerException npe)
{
    moreLine = false;
}
catch (IOException e)
{
    System.out.println("IOException when no more line from file found by LiuYi at 7");
}
}

```

```

/*****
Take a line vector as input and check the vector
to see more propriety in the same line
*****/

```

```

public int dollarNumber(Vector input)
{
    int count=0;

    for(int i=0; i<input.size(); i++)
        if ( ((String)input.elementAt(i)).equals("$aa") ||
              ((String)input.elementAt(i)).equals("$dd") ||
              ((String)input.elementAt(i)).equals("$ff") ||
              ((String)input.elementAt(i)).equals("$vv") ||
              ((String)input.elementAt(i)).equals("$oo") ||
              ((String)input.elementAt(i)).equals("$ODSvv") ||
              ((String)input.elementAt(i)).equals("$ODSdd") ||
              ((String)input.elementAt(i)).equals("$DWvv") ||
              ((String)input.elementAt(i)).equals("$DWdd") ||
              ((String)input.elementAt(i)).equals("$cc") ||
              ((String)input.elementAt(i)).equals("$pp") ||
              ((String)input.elementAt(i)).equals("$tt") ||
              ((String)input.elementAt(i)).equals("$ss") )
            count++;

    return count;
}

```

```

/*****
Take a line string as input and output the vector
which includes the string element in this line
*****/

```

```

public Vector tokenize(String inString)
{
    Vector tokenVector = new Vector();
    String tokenString = new String();
    StringTokenizer st = new StringTokenizer(inString);

    while (st.hasMoreTokens())
    {
        tokenString = st.nextToken();
        tokenVector.addElement(tokenString);
    }
    return tokenVector;
}

```

```

/*****
Replace a line with special line number from a file

Algorithm
Step 1: open file
Step 2: read one line from file
Step 3: tokenize the line
Step 4: is the token a mark
         yes: remove this line
Step 5: repeat step 3/4 until the end of the file
*****/

public void replace(String fileName, int lineNumber, String lineContent)
{
    LineNumberReader lineNumberReader;
    String          line;
    Vector          fileVector = new Vector();
    boolean        moreLine = true;
    int            counter=i;

    try
    {
        FileReader fileReader = new FileReader(fileName);
        lineNumberReader = new LineNumberReader(fileReader);

        do
        {
            line = lineNumberReader.readLine().trim();

            if (lineNumber!=counter)
            {
                fileVector.addElement(line);
            }
            else
            {
                fileVector.addElement(lineContent);
            }
            counter++;

        }while(moreLine);

        lineNumberReader.close();
    }
    catch (NullPointerException npe)
    {
        moreLine = false;
    }
    catch (IOException e)
    {
        System.out.println("IOException found by LiuYi at 5");
    }

    update(fileVector, fileName);
}

/*****
removeLine a line with special line number from a file

Algorithm
Step 1: open file
Step 2: read one line from file
Step 3: tokenize the line
Step 4: is the token a mark
         yes: remove this line
Step 5: repeat step 3/4 until the end of the file
*****/

```



```

public void removeLine(String fileName, int lineNumber)
{
    LineNumberReader lineNumberReader;
    String          line;
    Vector          fileVector = new Vector();
    boolean         moreLine = true;
    int             counter=1;

    try
    {
        FileReader fileReader = new FileReader(fileName);
        lineNumberReader = new LineNumberReader(fileReader);

        do
        {
            line = lineNumberReader.readLine().trim();

            if (lineNumber!=counter)
            {
                fileVector.addElement(line);
            }

            counter++;

        }while(moreLine);

        lineNumberReader.close();
    }
    catch (NullPointerException npe)
    {
        moreLine = false;
    }
    catch (IOException e)
    {
        System.out.println("IOException found by LiuYi at 5");
    }

    update(fileVector, fileName);
}

} //End of Class

```

(15) execution.java

```

import javax.swing.UIManager;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;

import java.io.*;

public class execution
{
    boolean packFrame = false;

    //Construct the application
    public execution()
    {
        Frame5 frame = new Frame5();
        //Validate frames that have preset sizes
        //Pack frames that have useful preferred size info, e.g. from their layout
        if (packFrame)
        {
            frame.pack();
        }
        else
    }
}

```

```

    {
        frame.validate();
    }
    //Center the window
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = frame.getSize();
    if (frameSize.height > screenSize.height) {
        frameSize.height = screenSize.height;
    }
    if (frameSize.width > screenSize.width) {
        frameSize.width = screenSize.width;
    }
    frame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
    frame.setVisible(true);
}

//Main method
public static void main(String[] args)
{
    try
    {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    new execution();
}
}

```

(16) Frame5.java

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Frame5 extends JFrame {
    JPanel contentPane;
    JLabel jLabel1 = new JLabel();
    JButton jButton1 = new JButton();
    JButton jButton2 = new JButton();
    JButton jButton3 = new JButton();
    JButton jButton4 = new JButton();
    JButton jButton5 = new JButton();
    JButton jButton6 = new JButton();
    JButton jButton7 = new JButton();
    JButton jButton8 = new JButton();
    JButton jButton9 = new JButton();
    JButton jButton10 = new JButton();
    JTextArea jTextArea1 = new JTextArea();
    JButton jButton11 = new JButton();

    //Construct the frame
    public Frame5() {
        enableEvents(AWTEvent.WINDOW_EVENT_MASK);
        try {
            jbInit();
        }
        catch(Exception e) {
            e.printStackTrace();
        }
    }

    //Component initialization
    private void jbInit() throws Exception
    {

```

```

codeGenerator cg=new codeGenerator();

cg.remove("$ff", "creating.pc");
cg.remove("$ff", "extracting.pc");
cg.remove("$ff", "cleaning.pc");
cg.remove("$ff", "loading.pc");

jLabel1.setFont(new java.awt.Font("Dialog", 1, 18));
jLabel1.setForeground(Color.yellow);
jLabel1.setText("WIG PROGRAMS COMPILER AND RUNNING");
jLabel1.setBounds(new Rectangle(95, 20, 423, 46));
contentPane = (JPanel) this.getContentPane();
contentPane.setLayout(null);
contentPane.setBackground(SystemColor.desktop);
this.setSize(new Dimension(610, 376));
this.setTitle("Frame Title");
jButton1.setText("VIEW GENERATED CODES");
jButton1.setBounds(new Rectangle(48, 70, 493, 42));
jButton1.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        jButton1_actionPerformed(e);
    }
});
jButton2.setFont(new java.awt.Font("Dialog", 1, 10));
jButton2.setText("COMPILE CREATING");
jButton2.setBounds(new Rectangle(48, 131, 150, 38));
jButton2.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        jButton2_actionPerformed(e);
    }
});
jButton3.setFont(new java.awt.Font("Dialog", 1, 10));
jButton3.setText("RUN CREATING");
jButton3.setBounds(new Rectangle(213, 131, 132, 41));
jButton3.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        jButton3_actionPerformed(e);
    }
});
jButton4.setFont(new java.awt.Font("Dialog", 1, 10));
jButton4.setText("COMPILE EXTRACTING");
jButton4.setBounds(new Rectangle(49, 183, 149, 38));
jButton4.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        jButton4_actionPerformed(e);
    }
});
jButton5.setFont(new java.awt.Font("Dialog", 1, 10));
jButton5.setText("RUN EXTRACTING");
jButton5.setBounds(new Rectangle(213, 183, 131, 40));
jButton5.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        jButton5_actionPerformed(e);
    }
});
jButton6.setFont(new java.awt.Font("Dialog", 1, 10));
jButton6.setText("COMPILE CLEANING");
jButton6.setBounds(new Rectangle(52, 237, 146, 39));
jButton6.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        jButton6_actionPerformed(e);
    }
});
jButton7.setFont(new java.awt.Font("Dialog", 1, 10));

```

```

jButton7.setText("RUN CLEANING");
jButton7.setBounds(new Rectangle(215, 239, 129, 39));
jButton7.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        jButton7_actionPerformed(e);
    }
});
jButton8.setFont(new java.awt.Font("Dialog", 1, 10));
jButton8.setText("COMPILE LOADING");
jButton8.setBounds(new Rectangle(53, 297, 145, 37));
jButton8.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        jButton8_actionPerformed(e);
    }
});
jButton9.setFont(new java.awt.Font("Dialog", 1, 10));
jButton9.setText("RUN LOADING");
jButton9.setBounds(new Rectangle(215, 297, 129, 39));
jButton9.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        jButton9_actionPerformed(e);
    }
});
jButton10.setText("AUTO");
jButton10.setBounds(new Rectangle(358, 276, 82, 59));
jButton10.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        jButton10_actionPerformed(e);
    }
});
jTextArea1.setBounds(new Rectangle(357, 132, 178, 133));
jButton11.setText("NEXT >>");
jButton11.setBounds(new Rectangle(457, 277, 77, 60));
jButton11.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        jButton11_actionPerformed(e);
    }
});
contentPane.add(jLabel1, null);
contentPane.add(jButton1, null);
contentPane.add(jButton2, null);
contentPane.add(jButton4, null);
contentPane.add(jButton6, null);
contentPane.add(jButton8, null);
contentPane.add(jButton10, null);
contentPane.add(jButton3, null);
contentPane.add(jButton5, null);
contentPane.add(jButton7, null);
contentPane.add(jButton9, null);
contentPane.add(jTextArea1, null);
contentPane.add(jButton11, null);

setSize(700,700);
show();
}

//Overridden so we can exit when window is closed
protected void processWindowEvent(WindowEvent e) {
    super.processWindowEvent(e);
    if (e.getID() == WindowEvent.WINDOW_CLOSING) {
        System.exit(0);
    }
}
}

```

```

void jButton1_actionPerformed(ActionEvent e) {
    display dp=new display();
}

void jButton3_actionPerformed(ActionEvent e) {

    setVisible(false);
    System.exit(0);
}

void jButton2_actionPerformed(ActionEvent e) {

    setVisible(false);
    System.exit(0);
}

void jButton4_actionPerformed(ActionEvent e) {

    setVisible(false);
    System.exit(0);
}

void jButton5_actionPerformed(ActionEvent e) {

    setVisible(false);
    System.exit(0);
}

void jButton6_actionPerformed(ActionEvent e) {

    setVisible(false);
    System.exit(0);
}

void jButton7_actionPerformed(ActionEvent e) {

    setVisible(false);
    System.exit(0);
}

void jButton8_actionPerformed(ActionEvent e) {

    setVisible(false);
    System.exit(0);
}

void jButton9_actionPerformed(ActionEvent e) {

    setVisible(false);
    System.exit(0);
}

void jButton10_actionPerformed(ActionEvent e) {

    setVisible(false);
    System.exit(0);
}

void jButton11_actionPerformed(ActionEvent e)
{
    setVisible(false);
    System.exit(0);
}
}

```

(17) display.java

```

import javax.swing.UIManager;
import java.awt.*;

public class display
{
    boolean packFrame = false;

    //Construct the application
    public display()
    {
        Frame3 frame = new Frame3();

        //Validate frames that have preset sizes
        //Pack frames that have useful preferred size info, e.g. from their layout
        if (packFrame)
        {
            frame.pack();
        }
        else
        {
            frame.validate();
        }

        //Center the window
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        Dimension frameSize = frame.getSize();

        if (frameSize.height > screenSize.height)
        {
            frameSize.height = screenSize.height;
        }
        if (frameSize.width > screenSize.width)
        {
            frameSize.width = screenSize.width;
        }
        frame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
        frame.setVisible(true);
    }

    //Main method
    public void displayCode()
    {
        try
        {
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }

        new display();
    }
}

```

(18) Frame3.java

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import java.io.*;

public class Frame3 extends JFrame
{
    JPanel contentPane;

```

```

Label label1 = new Label();
Button button2 = new Button();
TextArea textArea1 = new TextArea();
Choice choice1 = new Choice();
Choice choice2 = new Choice();

//Construct the frame
public Frame3()
{
    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
    try
    {
        jbInit();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}

//Component initialization
private void jbInit() throws Exception
{
    label1.setBounds(new Rectangle(56, 18, 476, 22));
    label1.setFont(new java.awt.Font("Dialog", 1, 26));
    label1.setForeground(Color.yellow);
    label1.setText("TARGET PROGRAM DISPLAY");
    contentPane = (JPanel) this.getContentPane();
    contentPane.setLayout(null);
    this.setForeground(Color.red);
    this.setSize(new Dimension(700, 700));
    this.setTitle("PROPOSAL DEMONSTRATION");
    button2.setBackground(Color.lightGray);
    button2.setBounds(new Rectangle(200, 650, 163, 42));
    button2.setFont(new java.awt.Font("Dialog", 1, 16));
    button2.setForeground(Color.yellow);
    button2.setLabel("EXIT");
    button2.addActionListener
    (
        new java.awt.event.ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                button2_actionPerformed(e);
            }
        }
    );
    textArea1.setBackground(SystemColor.text);
    textArea1.setBounds(new Rectangle(30, 156, 498, 480));

    choice1.setBackground(Color.lightGray);
    choice1.setBounds(new Rectangle(29, 106, 243, 48));
    choice1.setFont(new java.awt.Font("Dialog", 1, 18));
    choice1.setForeground(Color.yellow);
    choice2.setFont(new java.awt.Font("Dialog", 1, 18));
    choice2.setForeground(Color.yellow);
    choice1.add("OUTPUT PROGRAM");
    choice1.add("TABLE CREATING");
    choice1.add("DATA EXTRACTING");
    choice1.add("DATA CLEANING");
    choice1.add("DATA LOADING");
    choice1.addItemListener
    (
        new java.awt.event.ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                choice1_itemStateChanged(e);
            }
        }
    )
}

```

```

    }
    );
    choice2.setBackground(Color.lightGray);
    choice2.setBounds(new Rectangle(282, 106, 244, 48));
    choice2.add("OUTPUT LANGUAGE");
    choice2.add("PRO.C LANGUAGE");
    choice2.add("MORE .....");
    choice2.addItemListener
    (
        new java.awt.event.ItemListener()
        {

            public void itemStateChanged(ItemEvent e)
            {
                choice2_itemStateChanged(e);
            }
        }
    );
    contentPane.setBackground(SystemColor.desktop);
    contentPane.add(button2, null);
    contentPane.add(label1, null);
    contentPane.add(textArea1, null);
    contentPane.add(choice1, null);
    contentPane.add(choice2, null);
}

//Overridden so we can exit when window is closed
protected void processWindowEvent(WindowEvent e)
{
    super.processWindowEvent(e);
    if (e.getID() == WindowEvent.WINDOW_CLOSING)
    {
        System.exit(0);
    }
}

void button2_actionPerformed(ActionEvent e)
{
    setVisible(false);

    // endProposal ep = new endProposal();
    // ep.displayEnd();
}

void choice1_itemStateChanged(ItemEvent e)
{
    if (e.getItem()=="TABLE CREATING")
    {
        displayCode("creating.pc");
    }
    if (e.getItem()=="DATA EXTRACTING")
    {
        displayCode("extracting.pc");
    }
    if (e.getItem()=="DATA CLEANING")
    {
        displayCode("cleaning.pc");
    }
    if (e.getItem()=="DATA LOADING")
    {
        displayCode("loading.pc");
    }
}

void choice2_itemStateChanged(ItemEvent e)
{
    if (e.getItem()=="PRO.C LANGUAGE")
    {
        textArea1.append("PRO.C LANGUAGE CODE IS AVIABLE FOR LIU YI'S PROPOSAL .\n");
    }
}

```



```

    if (e.getItem()=="MORE    ....")
    {
        textAreal.append("OTHER LANGUAGE IS NOE AVIABLE IN LIU YI'S PROPOSAL.\n");
    }
}

void displayCode(String fileName)
{
    LineNumberReader lineNumberReader;
    boolean moreLine = true;
    String line = new String("");

    try
    {
        FileReader targetFileReader = new FileReader(fileName);
        lineNumberReader = new LineNumberReader(targetFileReader);

        do
        {
            line = lineNumberReader.readLine().trim();
            textAreal.append(line+"\n");
        }while(moreLine);

    }
    catch (NullPointerException npe)
    {
        moreLine = false;
    }
    catch (IOException e)
    {
        textAreal.append("NOTE: Target Programing Code "+fileName+" Is Not Aviable Yet.\n");
    }
}
}

```

(19) endWIG.java

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;

public class endWIG extends JFrame
{
    public JPanel title =new JPanel();
    public JPanel control=new JPanel();
    public JButton button =new JButton();
    public JLabel label =new JLabel();
    public JLabel label0 =new JLabel();
    public GridBagLayout gb=new GridBagLayout();;
    public GridBagConstraints gbc=new GridBagConstraints();;

    public endWIG()
    {
        super ("Wharehouse Integration Generation");

        Container c=getContentPane();
        c.setLayout(new BorderLayout());

        Icon bug0=new ImageIcon("angry.gif");
        label0.setText(" ");
        label0.setIcon(bug0);
        label0.setHorizontalTextPosition(SwingConstants.CENTER);
        label0.setVerticalTextPosition(SwingConstants.BOTTOM);
        label0.setToolTipText("label0");
        label0.setBackground(SystemColor.desktop);
    }
}

```

```

label0.setFont(new Font("TimesRoman",Font.BOLD,26));
title.setBackground(SystemColor.desktop);
title.add(label0);
c.add(title, BorderLayout.NORTH);

Icon bug=new ImageIcon("plant.gif");
label.setText("Thank you enjoy Warehouse Integration Generator");
label.setForeground(Color.yellow);
label.setIcon(bug);
label.setHorizontalTextPosition(SwingConstants.CENTER);
label.setVerticalTextPosition(SwingConstants.TOP);
label.setToolTipText("label");
label.setBackground(SystemColor.desktop);
label.setFont(new Font("TimesRoman",Font.BOLD,36));
title.setBackground(SystemColor.desktop);
title.add(label);
c.add(title, BorderLayout.CENTER);

control.setLayout(gb);
gbc.weightx=1;
gbc.weighty=1;
gbc.gridwidth=GridBagConstraints.REMAINDER;

Icon bug1=new ImageIcon("boy.gif");
button=new JButton("Finish",bug1);
button.setRolloverIcon(bug1);
button.setBackground(SystemColor.desktop);
button.setFont(new Font("TimesRoman",Font.BOLD,18));
gb.setConstraints(button,gbc);
control.setBackground(SystemColor.desktop);
control.add(button);

ButtonHandler buttonHandler=new ButtonHandler();
button.addActionListener(buttonHandler);

c.setBackground(SystemColor.desktop);
c.add(control, BorderLayout.SOUTH);

setSize(700,700);
show();
}

public class ButtonHandler implements ActionListener
{
    public void actionPerformed (ActionEvent e)
    {
        setVisible(false);
        System.exit(0);
    }
}

public void displayEnd()
{
    endWIG app=new endWIG();

    app.addWindowListener
    (
        new WindowAdapter()
        {
            public void windowClosing(WindowEvent e)
            {
                System.exit(0);
            }
        }
    );
}

public static void main(String[] args)
{

```

```
endWIG app=new endWIG();  
app.addWindowListener  
{  
  new WindowAdapter()  
  {  
    public void windowClosing(WindowEvent e)  
    {  
      System.exit(0);  
    }  
  }  
};  
}
```

B.2 Pro*C Components

(1) mainComponent

```

/*****
      Thesis Project

      Main Component

      Supervisor      Dr. Ezeife
      Student Name:   LiuYi
      Student Number: 999030031

Main component is the pro.C program that used for connect
Oracle database and management of the different function
in order to execute the whole test.

*****/

#include <stdio.h>
#include <string.h>

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
char *username=" $UID \t\n";
char *password=" $PWD \t\n";
EXEC SQL END DECLARE SECTION;

void Sff ();

void main()
{

EXEC SQL CONNECT :username identified by :password;

if (sqlca.sqlcode < 0)
{
printf("\n%s",sqlca.sqlerrm.sqlerrmc);
EXEC SQL ROLLBACK WORK RELEASE;
exit(1);
}
else
printf("Login Oracle successfully\n");

Sff ();

EXEC SQL COMMIT WORK RELEASE;
exit(0);
}

```

(2) metadataComponent

```

/*****
      Thesis Project

      (Repeatable) Metadata Component

      Supervisor      Dr. Ezeife

```

Student Name: LiuYi
Student Numnber: 999030031

Metadata component is the pro.C program that Generated from
Metadata template for creating the data warehouse metadata
tables

*****/

```
void metadata ()
{
EXEC SQL CREATE TABLE metadata
(
lastUpdate date,
tableName varchar2(20),
recordNumber number(10)
);
}
```

(3) timeComponent

```
void time ()
{
EXEC SQL CREATE TABLE time
(
day varchar2(20),
month varchar2(20),
year varchar2(20)
);
}
```

(4) initMetadataComponent

/*****

Thesis Project

(Repeatable) Init Metadata Component

Supervisor Dr. Ezeife
Student Name: LiuYi
Student Numnber: 999030031

Init Metadata component is the pro.C program that Initialize
matadata that is insert the table name and init date and the
record number

*****/

```
void $f ()
{
EXEC SQL INSERT INTO metadata
VALUES (SYSDATE, $t , 0);
}
```

(5) linkingComponent

```
#include <stdio.h>
#include <string.h>
```

```
void main()
{
EXEC SQL INCLUDE SQLCA;
```

```

EXEC SQL BEGIN DECLARE SECTION;
  char *username=" $targetUserID \n";
  char *password=" $targetPassword \n";
EXEC SQL END DECLARE SECTION;

EXEC SQL CONNECT :username identified by :password;

if (sqlca.sqlcode < 0)
{
  printf("\n%s",sqlca.sqlerrm.sqlerrmc);
  EXEC SQL ROLLBACK WORK RELEASE;
  exit(1);
}
else
  printf("Login Oracle Target Database successfully\n");

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
  char *username=" $sourceUserID \n";
  char *password=" $sourcePassword \n";
EXEC SQL END DECLARE SECTION;

EXEC SQL CONNECT :username identified by :password;

if (sqlca.sqlcode < 0)
{
  printf("\n%s",sqlca.sqlerrm.sqlerrmc);
  EXEC SQL ROLLBACK WORK RELEASE;
  exit(1);
}
else
  printf("Login Oracle Source Database successfully\n");

Sdd

EXEC SQL COMMIT WORK RELEASE;
exit(0);
}

```

(6) creatingComponent

```

/*****
                Thesis Project

                (Repeatable) Creating Component

                Supervisor      Dr. Ezeife
                Student Name:   LiuYi
                Student Numnber: 999030031

Creating component is the pro.C program that Generated from
Creating template
only for creating the ODS table and data
warehouse table based on the user interesting query
*****/

void $ODSF ()
{
EXEC SQL CREATE TABLE $ODSt
(
$vv $dd ,
$V $d
);
}

void $DWf ()
{
EXEC SQL CREATE TABLE $DWt

```

```

(
Svv $dd ,
Sv $d
);
}

```

(7) extractingComponent

```

/*****
                Thesis Project

        (Repeatable) Extracting Component

                Supervisor      Dr. Ezeife
                Student Name:   LiuYi
                Student Number: 999030031

Extracting component is the pro.C program that Generated from
Extracting template
only for extracting the different format
data from the different database source to ODS. This is the
main part for the data integration

*****/

```

```

void Sf ()
{
    EXEC SQL BEGIN DECLARE SECTION;
    Sdd Sv ;
    EXEC SQL END DECLARE SECTION;

    EXEC SQL DECLARE %C CURSOR FOR SELECT
    Sv ,
    Sv
    FROM St ;

    EXEC SQL OPEN %C ;
    EXEC SQL WHENEVER NOT FOUND DO break;

    while(sqlca.sqlcode==0)
    {
        EXEC SQL FETCH %C INTO
        : Sv ,
        : Sv ;

        EXEC SQL INSERT INTO $ODSt
        (
        $oo ,
        $o
        )
        VALUES
        (
        Spp ,
        Sp
        );
    }
    EXEC SQL CLOSE %C ;
}

```

(8) factComponent

```

/*****
                Thesis Project

        (Repeatable) Extracting Component

```

Supervisor Dr. Ezeife
Student Name: LiuYi
Student Number: 999030031

Extracting component is the pro.C program that Generated from
Extracting template only for extracting the different format
data from the different database source to ODS. This is the
main part for the data integration

...../

```
void Sf ()  
{  
  EXEC SQL BEGIN DECLARE SECTION;  
    $dd $vv ;  
  EXEC SQL END DECLARE SECTION;  
  
  EXEC SQL DECLARE $C CURSOR FOR SELECT  
    $cc ,  
    $C  
  FROM  
    $tt ,  
    $T  
  WHERE  
    $rr ;  
  
  EXEC SQL OPEN $C ;  
  EXEC SQL WHENEVER NOT FOUND DO break;  
  
  while(sqlca.sqlcode==0)  
  {  
    EXEC SQL FETCH $C INTO  
      : $vv ,  
      : $v ;  
  
    EXEC SQL INSERT INTO $ODSt  
    {  
      $oo ,  
      $O  
    }  
    VALUES  
    {  
      $pp ,  
      $P  
    };  
  }  
  EXEC SQL CLOSE $C ;  
}
```

(9) cleaningComponent

...../

Thesis Project

(Repeatable) Cleaning Component

Supervisor Dr. Ezeife
Student Name: LiuYi
Student Number: 999030031

Cleaning component is the pro.C program that Generated from
cleaning template
only remove the full same row in ODS and
full same row between ODS and DW. For similar duplication,
keep the MAX value for the richest information
and remove
less information record


```

*****/
void $f ()
{
EXEC SQL BEGIN DECLARE SECTION;
  $ODSdd $ODSvv ;
  $DWdd  $DWvv  ;
EXEC SQL END DECLARE SECTION;

/* delete redundancy for same row */
EXEC SQL CREATE TABLE TEMP AS SELECT DISTINCT * FROM $ODSt ;
EXEC SQL DROP TABLE $ODSt ;
EXEC SQL CREATE TABLE $ODSt AS SELECT DISTINCT * FROM TEMP;
EXEC SQL DROP TABLE TEMP;

/* remove the same rows between DS and DW */
EXEC SQL DECLARE $ODSc CURSOR FOR SELECT * FROM $ODSt ;

EXEC SQL DECLARE $DWc CURSOR FOR SELECT * FROM $DWt ;

EXEC SQL OPEN $ODSc ;
EXEC SQL WHENEVER NOT FOUND DO break;

while(sqlca.sqlcode==0)
{
EXEC SQL FETCH $ODSc INTO
  : $ODSvv ,
  : $ODSv ;

EXEC SQL OPEN $DWc ;
EXEC SQL WHENEVER NOT FOUND DO break;

while(sqlca.sqlcode==0)
{
EXEC SQL FETCH $DWc INTO
  : $DWvv ,
  : $DWv ;

EXEC SQL DELETE FROM $ODSt
  WHERE : $ODSv = : $DWv ;
}

EXEC SQL CLOSE $DWc ;
}

EXEC SQL CLOSE $ODSc ;
}

```

(10) keyCleaningComponent

```

/*****
                Thesis Project

                Key Cleaning Component

                Supervisor      Dr. Ezeife
                Student Name:   LiuYi
                Student Number: 999030031

```

Key Cleaning component is the pro.C program that used for remove the key redundancy in the same table.and update the fact non unique key table such as ID = C030031 and 999030031 in fact and deminsion tables

```

*****/

```

```

void $f ()
{
  EXEC SQL BEGIN DECLARE SECTION;
  char max[20];
  char current[20];
  EXEC SQL END DECLARE SECTION;

  EXEC SQL DECLARE $c CURSOR FOR
  SELECT MAX( $s )
  FROM $ODSt
  GROUP BY
  $vv ,
  $v ;

  EXEC SQL OPEN $c ;
  EXEC SQL WHENEVER NOT FOUND DO break;

  while(sqlca.sqlcode==0)
  {
    EXEC SQL FETCH $c INTO :max;

    EXEC SQL DECLARE $cc CURSOR FOR
    select $s
    FROM $ODSt
    where
    (
      $vv ,
      $v
    )
    in
    (
      select
      $vv ,
      $v
      from $ODSt
      where $s = :max );

    EXEC SQL OPEN $cc ;
    EXEC SQL WHENEVER NOT FOUND DO break;

    while(sqlca.sqlcode==0)
    {
      EXEC SQL FETCH $cc INTO :current;

      EXEC SQL UPDATE ODSfact SET $s = :max
      WHERE $s = :current;

    }

    EXEC SQL CLOSE $cc ;
  }

  EXEC SQL CLOSE $c ;

  /* delete redundancy for similar row */
  EXEC SQL CREATE TABLE TEMP AS SELECT DISTINCT * FROM $ODSt
  WHERE $s IN
  (
    SELECT MAX( $s )
    FROM $ODSt
    GROUP BY
    $vv ,
    $v
  );
  EXEC SQL DROP TABLE $ODSt ;
  EXEC SQL CREATE TABLE $ODSt AS SELECT DISTINCT * FROM TEMP;
  EXEC SQL DROP TABLE TEMP;
}

```

(11) nonKeyCleaningComponent

```
/*.....  
                Thesis Project  
  
                Non Key Cleaning Component  
  
                Supervisor      Dr. Ezeife  
                Student Name:   LiuYi  
                Student Numnber: 999030031
```

Non Key Cleaning component is the pro.C program that used for remove the non key redundancy in the same table. E.g., J. Smith and John Smith.

```
...../  
  
void $f ()  
{  
    EXEC SQL BEGIN DECLARE SECTION;  
        char maxValue[20];  
    EXEC SQL END DECLARE SECTION;  
  
    EXEC SQL DECLARE $c CURSOR FOR  
    SELECT MAX( $s )  
    FROM $ODSt  
    GROUP BY  
        $vv ,  
        $v ;  
  
    EXEC SQL OPEN $c ;  
    EXEC SQL WHENEVER NOT FOUND DO break;  
  
    while(sqlca.sqlcode==0)  
    {  
        EXEC SQL FETCH $c INTO :maxValue;  
  
        EXEC SQL UPDATE $ODSt SET $s = :maxValue  
        WHERE  
        {  
            $vv ,  
            $v  
        }  
        in  
        {  
            SELECT  
            $vv ,  
            $v  
            FROM $ODSt  
            WHERE $s = :maxValue  
        }  
    }  
  
    EXEC SQL CLOSE $c ;  
}
```

(12) loadingComponent

```
/*.....  
                Thesis Project  
  
                (Repeatable) Loading Component  
  
                Supervisor      Dr. Ezeife  
                Student Name:   LiuYi  
                Student Numnber: 999030031
```

Loading component is the pro.C program that loading the
datas from ODS to DW

```
*****/
void $f ()
{

EXEC SQL BEGIN DECLARE SECTION;
  int ODSrowNumber;
  $dd $vv ;
EXEC SQL END DECLARE SECTION;

/* only work for fact
EXEC SQL SELECT COUNT(*) INTO      :ODSrowNumber FROM $ODSt ;

EXEC SQL UPDATE metadata
SET recordNumber = recordNumber + :ODSrowNumber,
  lastUpdate = SYSDATE
WHERE tableName = $t ;
*/

EXEC SQL DECLARE $ODSc CURSOR FOR SELECT * FROM
  $ODSt ;

EXEC SQL OPEN $ODSc ;
EXEC SQL WHENEVER NOT FOUND DO break;

while(sqlca.sqlcode==0)
{
  EXEC SQL FETCH $ODSc INTO
    : $vv ,
    : $v ;

  EXEC SQL INSERT INTO $Dwt VALUES
  (
    : $vv ,
    : $v
  );
}
EXEC SQL CLOSE $ODSc ;

/*****
EXEC SQL INSERT INTO time VALUES
(
  substr(to_char(SYSDATE), 0, 2),
  substr(to_char(SYSDATE), 4, 3),
  '20'||substr(to_char(SYSDATE), 8, 2)
)
*****/
}
```

B.3 PL/SQL Components

(1) mainComponent

```
-- In PL/SQL, there is not Header and even if the  
-- #include is not a header statement in PL/SQL codes.  
-- So, only keeps mainComponent empty
```

(2) metadataComponent

```
--Start create metadata table
```

```
CREATE TABLE metadata  
(  
  lastUpdate  date,  
  tableName   varchar2(20),  
  recordNumber number(10)  
);
```

```
--End of metadata creating
```

(3) timeComponent

```
CREATE TABLE time  
(  
  day    varchar2(20),  
  month  varchar2(20),  
  year   varchar2(20)  
);
```

(4) initMetadataComponent

```
-- Start Init
```

```
INSERT INTO metadata VALUES (SYSDATE, $t , 0);
```

```
-- End of Init
```

(5) creatingComponent

```
--Start creating
```

```
CREATE TABLE $ODSt  
(  
  customerid varchar2(20);  
  $vv $dd ,  
  $v $d  
);
```

```

CREATE TABLE SDWt
(
Svv $dd ,
Sv $d
);

--End creating

```

(6) extractingComponent

```

--Start demision integration

DECLARE
  Svv $dd ;

CURSOR $C IS SELECT
  Svv ,
  Sv
FROM $t ;

BEGIN
  OPEN $C ;

  LOOP

    FETCH $C INTO
      Svv ,
      Sv ;
    EXIT WHEN $C %NOTFOUND;

    INSERT INTO $DSt
    (
      $oo ,
      $o
    )
    VALUES
    (
      $pp ,
      $p
    );

  END LOOP;

  CLOSE $C ;

END;

/

--End of deminsion integration

```

(7) factComponent

```

-- start fact integration

DECLARE
  Svv $dd ;

CURSOR $C IS SELECT
  $cc ,
  $c
FROM
  $tt ,
  $t

```

```

WHERE
  $r ;

BEGIN
  OPEN $C ;
  LOOP
    FETCH $C INTO
      $vv ,
      $v ;
    EXIT WHEN $C %NOTFOUND;

    INSERT INTO $ODSt
      (
        $oo ,
        $o
      )
    VALUES
      (
        $pp ,
        $p
      );
  END LOOP;

  CLOSE $C ;
END;

```

```

/

--End of fact

```

(8) cleaningComponent

```

--Start Claning

CREATE TABLE TEMP AS SELECT DISTINCT * FROM $ODSt ;
DROP TABLE $ODSt ;
CREATE TABLE $ODSt AS SELECT DISTINCT * FROM TEMP;
DROP TABLE TEMP;
/

DECLARE

  $ODSvv $ODSdd ;
  $DWvv $DWdd ;

  CURSOR $ODSc IS SELECT * FROM $ODSt ;
  CURSOR $DWc IS SELECT * FROM $DWt ;

BEGIN

  OPEN $ODSc ;
  LOOP
    FETCH $ODSc INTO
      $ODSvv ,
      $ODSv ;
    EXIT WHEN $ODSc %NOTFOUND;

    OPEN $DWc ;
    LOOP
      FETCH $DWc INTO
        $DWvv ,
        $DWv ;
      EXIT WHEN $DWc %NOTFOUND;

      DELETE FROM $ODSt
        WHERE $ODSv = $DWv ;
    END LOOP;

  CLOSE $DWc ;

```

```

END LOOP;

CLOSE $ODSc ;
END;
/

-- End of cleaning

```

(9) keyCleaningComponent

```

--Start Key cleaning

DECLARE
maxValue VARCHAR2(20);
currentValue VARCHAR2(20);

CURSOR $c is
SELECT MAX( $s )
FROM $ODSt
GROUP BY
  $vv ,
  $v ;

CURSOR $cc IS
select $s
FROM $ODSt
where
(
  $vv ,
  $v
)
in
(
  select
  $vv ,
  $v
  from $ODSt
  where $s = maxValue
);

BEGIN
OPEN $c ;
LOOP
  FETCH $c INTO maxValue;
  EXIT WHEN $c %NOTFOUND;

  OPEN $cc ;
  LOOP
    FETCH $cc INTO currentValue;
    EXIT WHEN $cc %NOTFOUND;

    UPDATE ODSfact SET $s = maxValue
    WHERE $s = currentValue;
  END LOOP;
  CLOSE $cc ;

END LOOP;
CLOSE $c ;

END;
/

CREATE TABLE TEMP AS SELECT DISTINCT * FROM $ODSt
WHERE $s IN
(
  SELECT MAX( $s )

```



```

    FROM $ODSt
    GROUP BY
    $vv ,
    $v
);
DROP TABLE $ODSt ;
CREATE TABLE $ODSt AS SELECT DISTINCT * FROM TEMP;
DROP TABLE TEMP;
/

```

--End of key cleaning

(10) nonKeycleaningComponent

--Start non key cleaning

```

DECLARE
  maxValue VARCHAR2(20);

CURSOR $c IS
  SELECT MAX( $s )
  FROM $ODSt
  GROUP BY
    $vv ,
    $v ;

BEGIN
  OPEN $c ;
  LOOP
    FETCH $c INTO maxValue;
    EXIT WHEN $c %NOTFOUND;

    UPDATE $ODSt SET $s = maxValue
    WHERE
    (
      $vv ,
      $v
    )
    in
    (
      SELECT
      $vv ,
      $v
      FROM $ODSt
      WHERE $s = maxValue
    );
  END LOOP;  CLOSE $c ;

END;
/

```

--End of non key cleaning

(11) loadingComponent

--Start loading

```

DECLARE
  ODSrowNumber int;
  $vv $dd ;
  -- $v $d ;

CURSOR $ODSc IS SELECT * FROM
$ODSt ;

BEGIN

```

```

SELECT COUNT(*) INTO ODSrowNumber FROM $ODSt ;

UPDATE metadata
SET recordNumber = recordNumber + ODSrowNumber,
    lastUpdate = SYSDATE
WHERE tableName = St ;

OPEN $ODSc ;
LOOP
  FETCH $ODSc INTO
  $vv ,
  $v ;
  EXIT WHEN $ODSc %NOTFOUND;

  INSERT INTO $DWt VALUES
  (
    $vv ,
    $v
  );
END LOOP;
CLOSE $ODSc ;

INSERT INTO time VALUES
(
  substr(to_char(SYSDATE), 0, 2),
  substr(to_char(SYSDATE), 4, 3),
  '20'||substr(to_char(SYSDATE), 8, 2)
);

END;
/

--End of loading

```

B.3 Banking DW/SDB definitions

(1) data warehouse definition

```

fact, customerid, varchar2(20), null
fact, accounttype, varchar2(20), null
fact, transtype, varchar2(20), null
fact, branchid, varchar2(20), null
fact, amount, number(10), null
fact, balance, number(10), null
fact, time, varchar2(20), null
customer, customerid, varchar2(20), key
customer, name, varchar2(20), nonKey
customer, address, varchar2(20), null
customer, city, varchar2(20), null
customer, phone, varchar2(20), null
branch, branchid, varchar2(20), key
branch, address, varchar2(20), null
branch, city, varchar2(20), null

```

(2) source database definition

```

saving, saving, cid, number(10)
saving, saving, branchid, varchar2(20)
saving, saving, transtype, varchar2(20)
saving, saving, amt, number(10)
saving, saving, status, varchar2(10)
saving, branch, branchid, varchar2(20)
saving, branch, address, varchar2(20)
saving, branch, city, varchar2(10)
saving, customer, cid, number(10)
saving, customer, cname, varchar2(20)
saving, customer, caddress, varchar2(20)

```

```

saving, customer, ccity, varchar2(20)
saving, customer, cphone, varchar2(10)
saving, balance, cid, number(10)
saving, balance, opendate, date
saving, balance, balance, number(10)
checking, cchecking, accid, varchar2(20)
checking, cchecking, bid, varchar2(20)
checking, cchecking, type, varchar2(20)
checking, cchecking, amt, number(10)
checking, checkingcustomer, acctid, varchar2(20)
checking, checkingcustomer, lastname, varchar2(20)
checking, checkingcustomer, firstname, varchar2(20)
checking, checkingcustomer, addr, varchar2(20)
checking, checkingcustomer, city, varchar2(20)
checking, checkingcustomer, phone, varchar2(20)
checking, checkingcustomer, open_date, number(10)
checking, checkingcustomer, bal, number(10)
checking, checkingbranch, bid, varchar2(20)
checking, checkingbranch, baddress, varchar2(20)
checking, checkingbranch, bcity, varchar2(20)

```

(3) Integration rule definition

```

fact, customerid, varchar2(20), saving, saving, cid, number(10), null
fact, accounttype, varchar2(20), saving, , , , 'savings'
fact, transtype, varchar2(20), saving, saving, transtype, varchar2(20), null
fact, branchid, varchar2(20), saving, branch, branchid, varchar2(20), null
fact, amount, number(10), saving, saving, amt, number(10), null
fact, time, varchar2(20), saving, , , , to_char(SYSDATE)
fact, balance, number(10), saving, balance, balance, number(10), null
fact, customerid, varchar2(20), checking, checking, accid, varchar2(20), null
fact, accounttype, varchar2(20), checking, , , , 'checking'
fact, transtype, varchar2(20), checking, checking, type, varchar2(20), null
fact, branchid, varchar2(20), checking, checking, bid, varchar2(20), null
fact, amount, number(10), checking, checking, amt, number(10), null
fact, time, varchar2(20), checking, , , , to_char(SYSDATE)
fact, balance, number(10), checking, checkingcustomer, bal, number(10), null
customer, customerid, varchar2(20), saving, customer, cid, number(10), null
customer, name, varchar2(20), saving, customer, cname, varchar2(20), null
customer, address, varchar2(20), saving, customer, caddress, varchar2(20), null
customer, city, varchar2(20), saving, customer, ccity, varchar2(20), null
customer, phone, varchar2(20), saving, customer, cphone, varchar2(20), null
customer, customerid, varchar2(20), checking, checkingcustomer, accid, varchar2(20), null
customer, name, varchar2(20), checking, checkingcustomer, lastname, varchar2(20), lastname || firstname
customer, name, varchar2(20), checking, checkingcustomer, firstname, varchar2(20), lastname || firstname
customer, address, varchar2(20), checking, checkingcustomer, addr, varchar2(20), null
customer, city, varchar2(20), checking, checkingcustomer, city, varchar2(20), null
customer, phone, varchar2(20), checking, checkingcustomer, phone, varchar2(20), null
branch, branchid, varchar2(20), saving, branch, branchid, varchar2(20), null
branch, address, varchar2(20), saving, branch, address, varchar2(20), null
branch, city, varchar2(20), saving, branch, city, varchar2(20), null
branch, branchid, varchar2(20), checking, checkingbranch, bid, varchar2(20), null
branch, address, varchar2(20), checking, checkingbranch, baddress, varchar2(20), null
branch, city, varchar2(20), checking, checkingbranch, bcity, varchar2(20), null

```

VITA AUCTORIS

NAME Liu Yi

PLACE OF BIRTH Beijing, P. R. China

YEAR OF BIRTH May 10, 1960

EDUCATION

M. Sc., Computer Science
University of Windsor,
Windsor, Ontario, Canada
1999 – 2000

M. Sc., Computer Information System
Beijing Polytechnic University
Beijing, P. R. China
1985 – 1988

B. Sc., Mathematics and Statistics
Beijing Economical Institute
Beijing, P. R. China
1980 - 1984