Generic Metadata Repository

Undertaken by

Nighat Zehra [323-FAS/MSCS/F06]

Supervised by

Dr. Nayyer Masood

Co. Supervisor

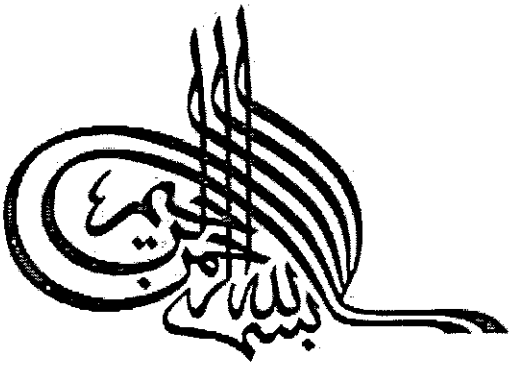
Ms. Tehmina Amjad

Department of Computer Science

Faculty of Basic and Applied Sciences

International Islamic University, H-10, Islamabad

2009



In the name of Almighty Allah,

The most Beneficent, the most Merciful.

Department of Computer Science,

International Islamic University, Islamabad.

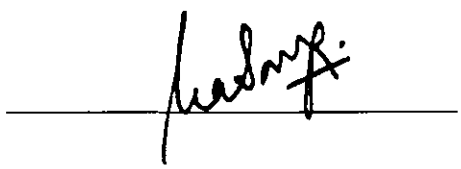
Dated: 04-04-09

Final Approval

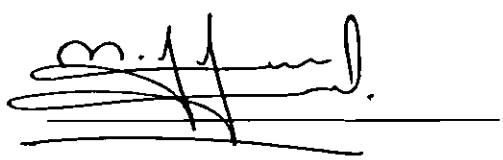
It is certified that we have read the thesis, titled **"Generic Metadata Repository"** submitted by **Miss Nighat Zehra** Reg. No. **323-FAS/MSCS/F06.** It is our judgment that this thesis is of sufficient standard to warrant its acceptance by the International Islamic University, Islamabad, for the Degree of **Master of Science in Computer Science, MS(CS).**

Committee

External Examiner
Dr. Nazir Ahmed Sangi
Associate Professor,
Depratment of Computer Science,
Allama Iqbal Open University, Islamabad.

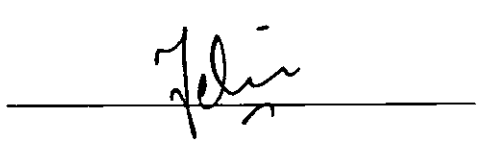


Internal Examiner
Mr. M. Imran Saeed
Assistant Professor,
Department of Computer Science,
International Islamic University, Islamabad.



Supervisor
Dr. Nayyer Masood
Associate Professor,
Department of Computer Science,
Muhammad Ali Jinnah University, Islamabad.

Co. Supervisor
Ms. Tehmina Amjad
Assistant Professor,
Department of Computer Science,
International Islamic University, Islamabad.



A Dissertation Submitted To

Department of Computer Science,

Faculty of Basic and Applied Sciences,

International Islamic University, Islamabad

As a Partial Fulfillment of the Requirement for the Award of the

Degree of *Master of Science in Computer Sciences*

DEDICATION

Dedicated to my beloved parents, sisters, brother and my husband, their prayers always pave the way to success for me.

DECLARATION

I hereby declare that this thesis, "***Generic Metadata Repository***" neither in part nor in full, has been copied from any source, except where cited; hence, acknowledged. It is further declared that I have done this research with the accompanied report entirely on the basis of our personal efforts, under the proficient guidance of our teachers especially our supervisors Dr. Nayyer and Ms. Tehmina. No portion of the work being presented herein, has been submitted to any other university, institute, or seat of learning, in support to any piece of writing for bestowment of any other degree of qualification.

Nighat Zehra
323-FAS/MSCS/F06

ACKNOWLEDGEMENTS

Praise to Allah, who is His infinite wisdom and benevolence, enabled me to undertake and complete this task despite my weakness and human limitations. Words cannot express the gratitude towards my family especially my parents, whose prayers have made me able to achieve this goal.

I cannot express my profound gratitude to my project supervisors, **Dr. Nayyer Masood** and **Ms. Tehmina Amjad** who went through every inch of manuscript with painstaking attention to detail and made infinite number of helpful suggestions. There ever available guidance, support, help and blessing has been source of encouragement for me. Moreover, they suggested many improvements and also attempted to correct my dyslexia spellings.

Acknowledgement is also due to my teachers specially, Mr. Shahid Rauf and Mr. Imran Saeed for dedicatedly instilling and imparting enlightenment to me during the course of studies and afterwards for my project.

Nighat Zehra
323-FAS/MSCS/F06

PROJECT IN BRIEF

Project Title	Generic Metadata Repository
Tools Used	• Visual Studio 2008 • Visual Paradigm • SQL Server 2005
Languages Used	• C # • XML • XMI
System Used	Intel Pentium IV
Operating System Used	Windows XP
Starting Date:	21st November, 2007
Completion Date:	31st Dec, 2008
Undertaken By	Nighat Zehra 323-FAS/MSCS/F06
Supervised By	Dr. Nayyer Masood and Ms. Tehmina Amjad

ABSTRACT

A data warehouse (DW) is a database that integrates information from various operational sources to satisfy decision making requests. For a well performance data warehouse many components must work together. Metadata, usually called data about data, is an important component of DW. In this research work, we concentrate on metadata repository. We have addressed the major problems and issues of metadata repository, these issues are:

(i) *"The repository or metadata schema is truly application specific i-e dedicated to data warehousing"[4]*

(ii) Existing repositories that we have discussed in the literature review are tool specific. They are dependent upon platform that means repository is dependent upon the database management system.

(iii) The problem of designing a repository schema from multiple operational sources.

(iv) The issue of schema integration from multiple operational sources w.r.t DW that is name matching.

Based on the problems and issues of metadata repository we have proposed the architecture of generic metadata repository (GMR). GMR resolves the above mention problems. GMR consists of following operations: (a) extract schema from XML metadata interchange (XMI), (b) integrate schemas, (c) transform relational schema into star schema, and (d) store the metadata about both schema into repository using Extensible Markup Language (XML) documents.

Keywords

Data Warehouse, Metadata Repository, Schema Integration, Schema Transformation.

TABLE OF CONTENTS

LIST of FIGURES

LIST of TABLES

Chapter 1

INTRODUCTION

1 Introduction

The data warehouse is a database that is designed and used for decision making in an organization or enterprise. It contains huge amount of data that is historical and static in nature. Different analytical tools and techniques are applied in the large repository of data to extract non-obvious/hidden information which is used in the decision making.

The data warehouse is defined in [1] as *"a subject-oriented, integrated, time varying, non-volatile collection of data that is used primarily in organizational decision making"*. Typically, the data warehouse is maintained separately from an organization's operational databases, the databases that contain data for day to day operations and reporting. However, these operational databases serve as data source for the data warehouse. It means that the data that has come obsolete from the operational point of view and that has been set aside on a backup medium that data is moved in data warehouse to perform analytical operations. So, the data in the data warehouse is used for, what is called, the on-line analytical processing (OLAP). The functional and performance requirements for the OLAP are quite different from those of the on-line transaction processing (OLTP), the applications traditionally supported by the operational databases [11].

The data warehouse is designed based on what kind of information is important in company's decision making process e.g. sales, marketing, inventory and accounting etc, and it adopts a specialized schema design for maximum efficiency and performance.

The structure of this chapter is as follows: Section 1.1 describes the typical architecture of data warehousing and the process of designing and operating a data warehouse. Section 1.2 gives the components of data warehouse. Section 1.3 and 1.4 provides a scope and objective of the research project. Section 1.5 identifies the problem statement. Section 1.6 discusses the approach that I have followed in my research and finally section 1.7 gives the overall architecture of the thesis.

ability to do complex analysis using the information in the data warehouse. The power user wants to be able to navigate throughout the data warehouse, pick up interesting data, format his own queries, and create custom reports and ad hoc queries. In order to provide information to the wide community of data warehouse users, the information delivery component includes different methods of information delivery. Ad hoc reports are predefined reports primarily meant for novice and casual users.

1.2.5) Metadata:

Metadata in a data warehouse is similar to the data dictionary or the data catalog in a database management system. In the data dictionary, we keep the information about the logical data structures, the information about the files and addresses, the information about the indexes, and so on. The data dictionary contains data about the data in the database. Similarly, the metadata component is the data about the data in the data warehouse. The focus of our research is the design of metadata and the metadata repository so here we are going to discuss the metadata in detail.

Metadata is data about data that describes the data warehouse. It is used for building, maintaining, managing and using the data warehouse. Metadata can be classified into:

(i) Technical metadata, which contains information about warehouse data for use by warehouse designers and administrators when carrying out warehouse development and management tasks.

(ii) Business metadata, which contains information that gives users an easy-to-understand perspective of the information stored in the data warehouse.

Metadata management is provided via metadata repository. Metadata repository management software can be used: to map the source data to the target database; generate code for data transformations; integrate and transform the data; and control moving data to the warehouse.

The repository offers a way to understand what information is available, where it comes from, where it is stored, the transformations performed on the data, its currency and other important facts about the data. Metadata has however taken on a more visible role among

day-to-day knowledge workers. Today it serves as the main catalog, or map to a data warehouse.

The central metadata repository is an essential part of a data warehouse. Metadata can be generated and maintained by an ETL tool as part of the specification of the extraction, transformation and load processes. The repository can also capture the operational statistics on the operation of the ETL process. Ideally, access to data definitions and business rules in the metadata repository should be end user accessible.

The research work in this thesis is about a Generic Metadata Repository (GMR) for data warehouse systems, which aids such systems to reuse the repository. The following paragraphs will describe the summary of my thesis work.

1.3 Scope

The target here is to create a generic metadata repository, which is not restricted to only one data warehouse. The generic integration and transformation methods are proposed. Some major tasks of this research are:

- Architecture for GMR is investigated and developed
- Prototype of GMR is developed

1.4 Objectives

This research project is developed for a reusable component so that any other organization can reuse this component. Keeping this thing in mind our objectives is to store the metadata of operational source and data warehouse into the metadata repository based on XML documents (having schema information) so that these documents can be used by any other organization.

1.5 Problem Statement

The metadata repository is application/organization specific so it can not be reused by any other application/organization. We have developed a Generic Metadata Repository (GMR). The work of the GMR will be demonstrated with the help of "student information system". We have also discussed the problem of designing a schema from multiple operational sources. We have identified an issue of schema integration that is name matching and it is solved by our proposed semi automatic approach CSI.

1.6 Approach

The first step is to investigate the best suitable programming language for the planned GMR which will be **C#** (pronounced C Sharp) and **XML**. The reasons for preferring over other programming languages will be elaborated in chapter 5. The first step is followed by an architecture design and generic operations are defined. And then the development of the sample GMR can begin.

The architectural design of the system is as following. The GMR will convert the E-R schema into XMI schema using visual paradigm. After conversion, system will extract the schema from XMI file and apply the integration process to integrate schemas. After integration, integrated schema is then transformed into data warehouse schema that is the 'star schema'. The metadata of both schemas are stored into the metadata repository using XML documents. This repository can be reused by any other organization for creating their data warehouse.

1.7 The Thesis

The rest of the document has been divided into following chapters:

- Chapter 2 is "*Literature Survey*". We include different research papers to highlight (i) the importance of metadata repository in data warehouse, (ii) the problem of designing integrated schema from multiple operational sources, (iii) the approaches of schema integration, and (iv) the concept of schema transformation. After

summarizing these research papers we have identified the problems in the areas of metadata repository.

- Chapter 3 is *"Research Methodology"* which describes the solution for the problem statement i.e. generic metadata repository; it gives the research methodology and also describes the approach that we have adopted in our thesis.

- Chapter 4 is "System Architecture" that presents the design and architecture of the proposed solution.

- Chapter 5 is *"Implementation"*, in which the description of the languages and tools are given and the details of the prototype that is developed.

- Chapter 6 is *"Experimental Results"*, in which different schemas are tested on implemented techniques.

- Chapter 7 is "*Conclusion"* that briefly concludes the thesis and *References* are given at the end.

Chapter 2

LITERATURE SURVEY

2 Literature Survey

A data warehouse combines different data sources into a single data source for end user access. End user can perform querying and reporting of warehouse information. "The goal of data warehouse is to create a data repository that makes operational data available in a form that is acceptable for decision support and other applications"[16].

Metadata is one important concept that plays a fundamental role in the data warehousing environment. Metadata (meta data, or sometimes meta information) is "data about data"[17] and has to be referred for any kind of access or operation in the data warehouse. That is, whenever a user poses a query the DBMS or a DW tool has to refer to metadata for verifying the correctness of query, for the existence of required data, to find the location of data in the repository, to set an efficient mechanism to access data, to know the format of data to read and present it in a proper format. Similar types of references to the metadata are also required when something is to be written into the DW. This explains the role and importance of metadata in the DW operations. Authors in [1] mention the same fact as: Metadata is used in data warehouse in a various ways:

- It is used to help the decision support system analyst lo locate the contents of data warehouse.
- It is a guide to the mapping of the data from operational system to data warehouse.

The nature of data stored in the metadata includes data about attributes or elements, (name, data type, size, etc) and data about data structures or records (columns, length, fields, etc) and data about data (location of data, how it is associated, etc) [17]. Data warehouse metadata are the pieces of information that is stored in one or more special purpose metadata repositories that includes (a) information on the contents of data warehouse, (b) information on the processes that take place in data warehouse, and (c) sources of data warehouse and so on [23].

CM is transferred into logical modeling. In multidimensional modeling where target databases are relational or multidimensional. In relational implementations are star, snowflake etc. in multidimensional implementations are cubes, dwarfs.

There is an issue of semantic gap between conceptual data models and relational or multidimensional implementation of data cubes. No solutions can cope with generalization/specialization relationships in OLAP hierarchies, how to represent dimension constraints, or less expressive content dependencies. "Future research is required to bridge the semantic gap i-e to preserve all information captured by CM in logical implementations"[7].

"Research on DW modeling and design is far from being dead, because more sophisticated techniques are needed for solving known problems because of new problems raised during the adoption of DW to requirements of today's business"[7].

2.1.2 Data cleaning: problems and current approaches

To improve the data quality by finding and removing the errors is done by data cleaning. Data quality problems are present in single sources and multiple sources. The need of data cleaning increases in multiple sources [21]. This paper presents the ETL processes. Data cleaning is a part of ETL. All data cleaning is performed in data staging area. "Federated database system and web-bases information systems face data transformation steps similar to those of data warehouses"[21]. There is a wrapper for extraction and mediator for integration. To minimize the manual inspection data cleaning tools should be supported. Paper presents the data quality problems. Single source problems at schema level are uniqueness violation; referential integrity violation, illegal values and problems at instance level are missing values, misspellings and duplicate records. The main problem w.r.t schema design is naming conflict [21][24]. Naming conflict occurs when the same name is used for different objects (homonyms) or different names are used for the same object (synonyms). Multiple source problems are naming conflict, structure conflicts or to identify overlapping data. To solve these problems schema integration and data cleaning is required.

Data cleaning consists of different phases: data analysis, definition of transformation workflow and mapping rules, verification, transformation, backflow of cleaned data.

Transformation process requires large number of metadata [21]. For schema translation following steps includes extracting values free form attributes, validation and correction, and standardization. Different tools are available for data transformation and cleaning. Some tools are domain specific e.g. cleaning name and address data.

2.1.3 On metadata interoperability in data warehouse

The use of data warehouse depends on effective management of metadata. Metadata describes all warehouse data that is obtained from multiple sources. It is maintained in central warehouse and accessible in various ways e.g. for querying, OLAP, navigation. Metadata management is required for high quality of warehouse data and provides the flexibility to extend the scope of warehouse. Data warehouse integrates multiple heterogeneous data sources. Data integration is based on integration of metadata. "Metadata integration also has to deal with heterogeneity as source systems substantially differ in the degree and form they provide describing metadata." [3]

Metadata of operational sources describes the structure, semantic, meaningful names and describing comments. Perquisite to metadata integration is metadata interoperability (the ability to exchange metadata between components of data warehouse). This paper presents the environment of data warehouse as shown in figure 2-1.

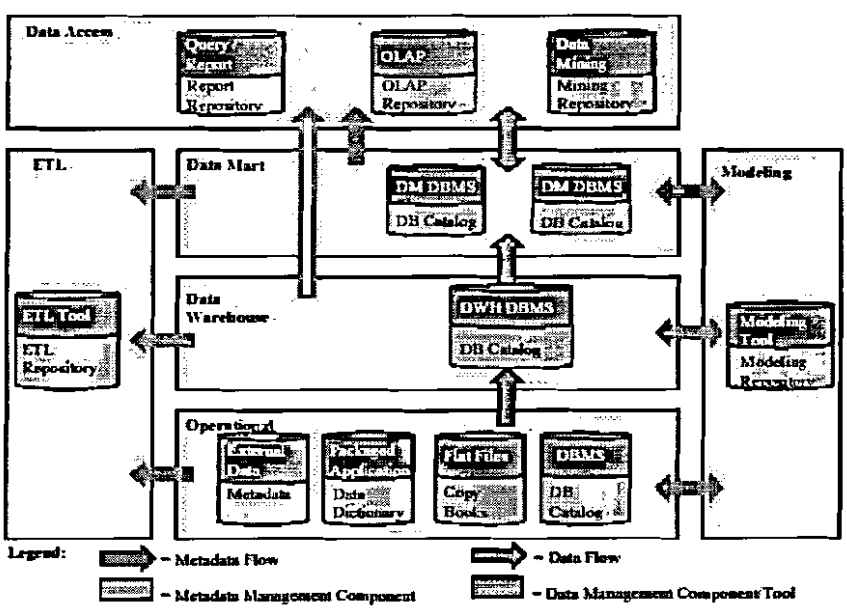


Figure 2-1: Distribution of data and metadata in data warehouse [3]

It consists of file system, DBMS, Data warehouse, Data mart, tools for data modeling, ETL tasks, OLAP, querying. All of these components maintain metadata e.g. database catalogs, dictionaries, or tool specific repositories.

Metadata is accessed by different components so shared metadata needs to flow between components as it is results in metadata replication. The use of different metadata models makes complex metadata interoperability between components. "Consistently supporting shared metadata is thus crucial for data warehouses. Repository support is needed permitting tools and other data warehouse components to access, create, and extend shared metadata."[3]

This paper discusses the interoperability issues for metadata management that is based on three dimensional classifications of major types of metadata. Author proposes a classification scheme for data warehouse metadata differentiating along the dimension data, process and users.

In the paper author, discusses the architectural alternatives for management of shared metadata. There are different approaches centralized, decentralized, shared or federated and mixed approach. Some interoperability mechanism like file exchange, repository API, metadata wrapper and comparison of commercial repository products are also discussed.

There are several commercial repository solutions for data warehouse metadata management is discussed. Major features of these products with respect to the metadata model, metadata interoperability, and other functions are shown in table 2-1[3]. In the table 2-1, we have seen that the Microsoft repository is based on the Microsoft technology.

Aspects			Ardent MetaStage	IBM DataGuide	Microsoft Repository	Sybase WCC	Viasoft Rochade
Metadata Model			proprietary	proprietary, extensible	OIM, extensible	proprietary	proprietary, extensible
Underlying DBMS			UniVerse	DB2	SQL Server	Adaptive IQ, SQL Server	proprietary
Exchange Mechanisms (in 2 directions)	File Format	Import		IBM Tag Language, MDIS (ETI Extract)	XML/XIF	MDIS (DataStage), WAM (Warehouse Architect), ERX (ERWin)	XML/XIF
		Export					
	API	Import	MetaBroker Toolkit	proprietary (C)	proprietary (COM)		proprietary (C++)
		Export					
	Wrapper	Import	Cognos Impromptu, Business Objects, ERwin, ERStudio	Cognos Impromptu, Business Objects, CASE Tools	OLAP Services, English Query		CASE tools, DBMS, Programming Languages
		Export		Cognos Impromptu, Business Objects		Cognos Impromptu, Business Objects, English Wizard	CASE tools, DBMS
Metadata Synchronization			publish-subscribe			publish-subscribe	
	Versioning, Configuration				(COM) object level		attribute level
User Interface	Browsing, Navigating, Search		Web	proprietary GUI + Web	proprietary GUI	proprietary GUI + Web	proprietary GUI + Web
	Impact Analysis		query builder		view of ETL metadata	view of ETL metadata	query builder
Further Information			[Ar99] www.ardentsoftware.com	[IBM98a, IBM98b] www-4.ibm.com/software/data/vw	[BBC+99, BB99] msdn.microsoft.com/repository	[Ov98, Sy99] www.sybase.com/bid	[Vias97] www.viasoft.com

Table 2-1: Commercial Repository Approaches [3]

2.1.4 The role of metadata for data warehousing

Metadata is used to store the meaning or properties of data. The purpose of metadata is to better understand, manage and use that data. Metadata facilities managing, querying, consistent use and understanding of data. This paper gives an overview about the role of metadata plays for data warehousing. "Data warehousing is a collection of concepts and tools which aims at providing and managing a set of integrated data (the data warehouse) for business decision support within an organization."[4]

It captures all kinds of information necessary to extract, transform and load data from source systems into data warehouse. Paper presents the architecture of data warehouse systems as shown in figure 2-2. Main components of this architecture are data sources, ETL, Data Warehouse, Data Mart, Metadata repository. Metadata repository plays a main role in every phase on data warehousing. Metadata helps the user to understand the content of warehouse.

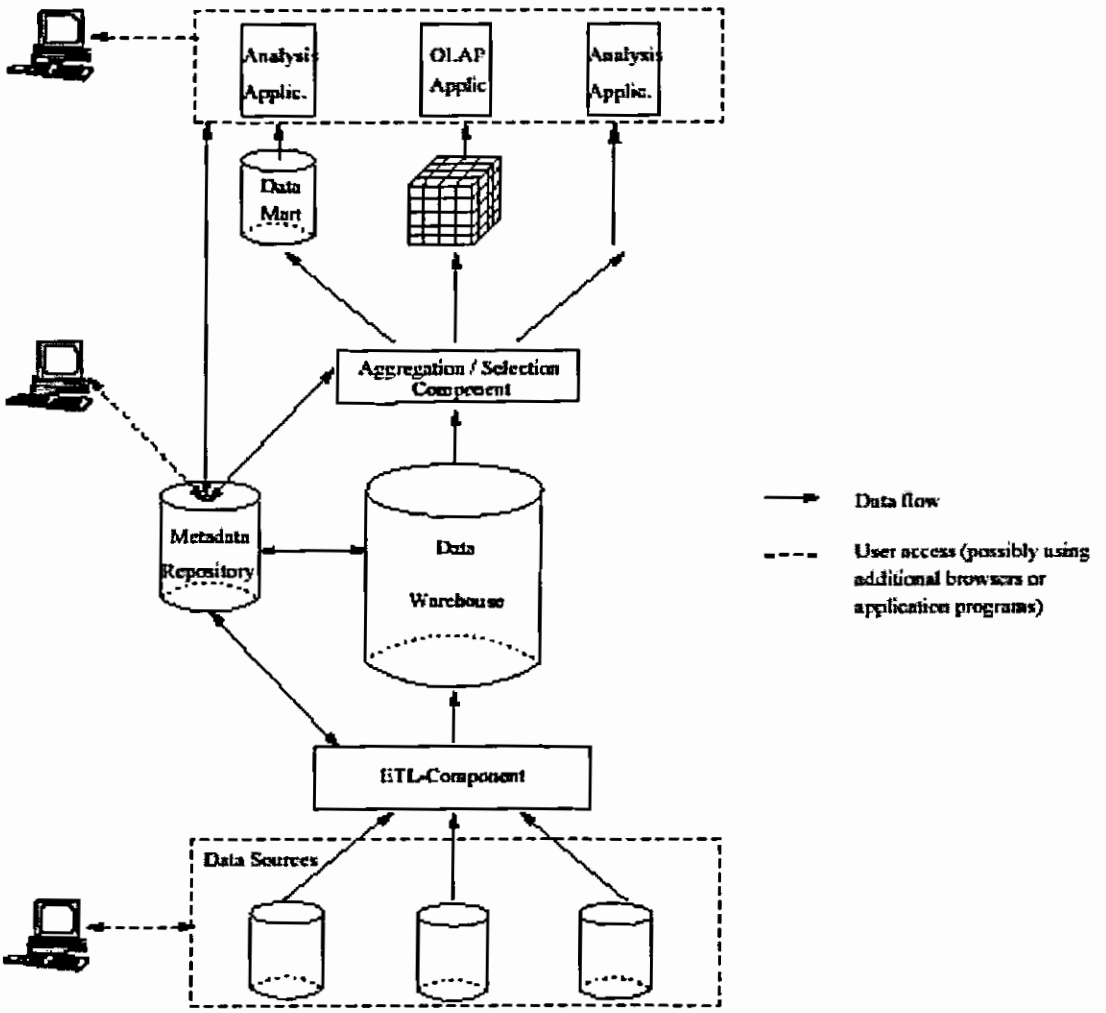


Figure 2-2: A Data Warehouse System [4]

Metadata may be use in three different ways [4]:

Passively: by providing a consistent documentation about structure development process and use of data warehouse.

Actively: by storing semantic aspects e.g. transformation rules that are represented and executed at runtime

Semi-actively: by storing static information. In it metadata is only read but not executed at runtime.

The two purposes for generation and management of metadata are:

1. To minimize the efforts of development and administration
2. To improve the extraction of information from it

Metadata stored and maintained in repository. "Metadata needs for a specific application domain (like data warehousing) actually impose the repository structure (e.g. the metadata schema) and the semantics of metadata to be stored. " [4]

The interaction of any component with the repository requires some mechanisms. Change management deals with handling of changes inside or outside the repository. Repository supports version and configuration management. "The repository meta model or metadata schema is truly application specific i-e dedicated to data warehousing"[4].

The centralized approach to data warehouse metadata management can be realized by general-purpose repository. There are some general purpose repositories available.

- UREP is an object oriented extensible repository. It provides Version control, transactions, user and session management, metadata service, defining object modeling constructs, extending UREP information model, reusing object types and operations and extending repository functionality. It has been licensed by different companies.
- Microsoft Repository is an object oriented extensible repository. It is targeted towards software vendors and users wishing to support the management of metadata in a variety of scenarios including software development and data warehousing.
- Platinum Repository is based on the E/R approach. It provides full bi-directional interfaces to many CASE tools including Erwin, Bachman, Sterling/Cayenne and Oracle Designer.

2.1.5 The process of metadata modeling in industrial data warehouse environments

There are different types of application using data warehouse (DWH) to fulfill the needs of different users. Metadata management system (MDMS) is the best way to provide the data flows and structures. Current metadata models not provide all the needs of possible metadata applications. "Standardized metadata models defined by groups of companies exist but they can not contain all possible kinds of metadata another company may require"[8]. A metadata model is a set of elements which are needed to implement the metadata schema. This paper shows the process of creating metadata model as small and simple. "The goal of this paper is not to show a new metadata model this is applicable to any DWH environment"[8]. In DWH there are two main metadata standards: Common warehouse model (CWM) and Resource description framework (RDF). In data warehousing there are three main areas of interest if we consider from data flow to reporting tools that DWH users work. The first area is ETL i-e a part of DWH. The second interest is reporting system or end user access. This is also defined as data mart. "A data mart is a component of a DWH which can be used for querying a defined subset of all DWH data"[8]. The third area is metadata management. Metadata answer different questions, this paper limited to answer three questions that are.

"Where does data come from? Where does data belong to? (Q1). How is data transformed, calculated, aggregated, etc.? (Q2). What meaning do data have? (Q3)"[8].

These questions lead to three possible dimensions. The first dimension is level of detail of metadata answers to Q1. The second dimension contains not only physical position but also the level of change of data that answers Q2. The third dimension contains level of abstraction shows which metadata belong together by content.

"Adding unstructured metadata to MDMS metadata model is not currently possible"[8]. These three dimension shows there are many possible solutions in metadata modeling.

RDF provides the opportunity to create a metadata model that covers three dimensions. RDF is basic metadata model. It is based on triple statements {subject, predicate, object}. A subject is a resource like a column in a data model. The object is an attribute like a column definition. The kind of relationship that connects an object to a subject is defined by predicate. The most common relationship is p/c which could model as a tree.

Representation and navigation through data model is addressed by first dimension of metadata. If more than one tree is based on more than one data models then RDF is used to link them. With this feature data flow can be model. To add object oriented feature to a RDF statements RDFS is needed. RDFS offer predicates to define resources as classes and as class instances.

There are two types of data schemas stored in MDMS (data schemas from productive databases and enterprise information model). The navigation through metadata inside both schemas is solution of 1st dimension. The links between all schemas span 2nd dimension. And connection of entities from both schemas realizes third dimension.

Metadata integration and mapping is simple and limited because of importing database schemas from one single tool. Paper describes the metadata schema of MDMS built using its metadata model and gives a short overview of its implementation.

A metadata models are getting more complex over time and amount of relations between them grow so need of model operators is getting urgent. These operators have been defined in generic model management that is established in relation databases but not yet in RDF repositories.

"The future development of MDMS is combination of application metadata with database metadata. This will help to answer about which data were accessed or changed by which process"[8].

2.1.6 Repository support for data warehouse evolution

Data warehouse consist of many components which store data for decision making. It can not designed in one step usually change over many years. To design it first starts with creating local data mart. Data marts are easier to implement then the enterprise-wide data warehouse. Data warehouse can be virtual/distributes. Data warehouse is different from OLTP systems. Data quality is important in data warehouse. Process model includes the processes like data loading or update propagation but the process model is specialized to deal with evolution processes. "The advantage of our proposed approach is that all relevant metadata of a data warehouse (architecture, quality, process and evolution information) are stored in a central repository"[6]. The information in the repository is used to find the deficiencies in data warehouse. Data stores can make changes due to some reasons like changes to physical

properties of the source, business rules, user requirements agents of all types can change new algorithms, rules, and so on. If any quality problem occurs then metadata repository is used to check which quality problem is there. There is a future work that the proposed models will be refined and extended to cove r new aspects of data warehouse processes.

2.1.7 Generic schema matching with cupid

Schema matching is an important step in schema integration. Data warehouse loading match is an operation that takes two schemas as input and returns corresponding elements. Today schema matching is doing manually and sometimes using a graphical tool but a minor name variation leads them to astray. Author says that match operation should be independent component and it must be generic means it can apply to many difference application domains. First of all schema matching problem discusses that focuses on a semantics of data may not completely capture in schema so user validation must be there. The goal is to match two schemas and find the mapping and give it to the user for validation.

Taxonomy of matching techniques

"Schema matchers can be characterized by following criteria (a longer survey based on this appears in [9]".

1- schema vs instance level

Schema based matcher consider only schema information and instance based consider only data.

2- element vs structural

An element based matcher find a mapping between individual schema elements and structure level matcher match combinations of elements.

3- linguistic based matcher

It matches the names of schema elements and other description. It compares the equality of names, comparing synonyms and WorldNet[10].

This paper proposes a new schema matching component i-e cupid. To be generic it has some characteristics that are discussed in [9]. The cupid approach is schema based not an instance based. In the paper algorithm of cupid is summarizes with an example. There is a future work of integrating cupid with an off-the-shelf thesaurus and enhance the cupid to make it a

truly general purpose schema matching component so that can be used for schema integration[9].

2.1.8 From enterprise models to dimensional models: A methodology for data warehouse and data mart design

Data warehousing is an important application of database technology. The most important issue of data warehouse is how to design database to support end user queries. In this paper, the design of data warehouse is based on an enterprise data model represented in entity relationship form.

"Kimball proposed a new technique for data modeling specially for designing data warehouses, which he called dimensional modeling"[13]. The objective of the dimensional modeling is to provide a database design that is easy for end user to understand and write queries. Another objective is efficiency of queries is maximized. Kimball began with the data mart as a dimensional model for departmental data and viewed the data warehouse as the enterprise wide collection of data marts. This is bottom up approach. Dimensional modeling begins with the tables rather than entities or attributes such as ERDs. The basic building block that is used in dimensional modeling is star schema. A star schema consists of one central table that is called fact table and a number of tables called dimension tables that are linked with fact table as shown in figure 2-3.

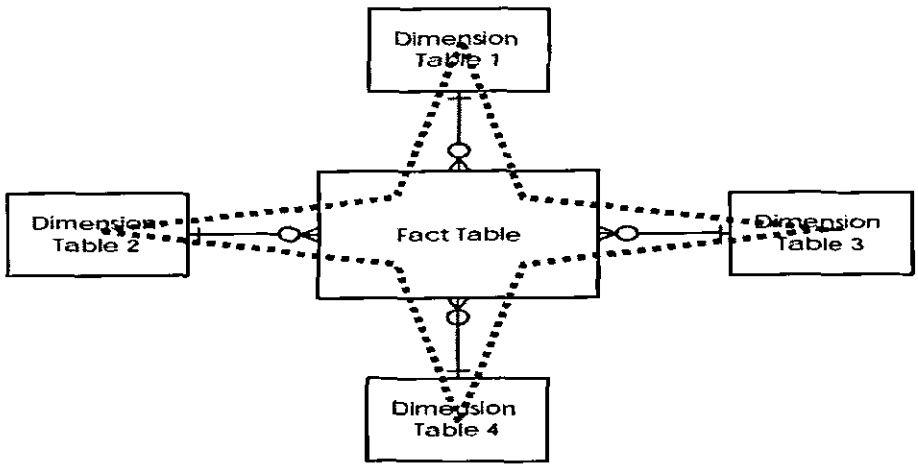


Figure 2-3: Star schema

The center of the star is the fact table, while the points of the star are the dimension tables. The primary key of the fact table is the combination of primary keys of all dimension tables.

A fact table contains the measurements. The fact table is linked to all dimension tables with one to many relationships. The dimension tables contain the textual attributes. An example of star schema is shown in figure 2-4.

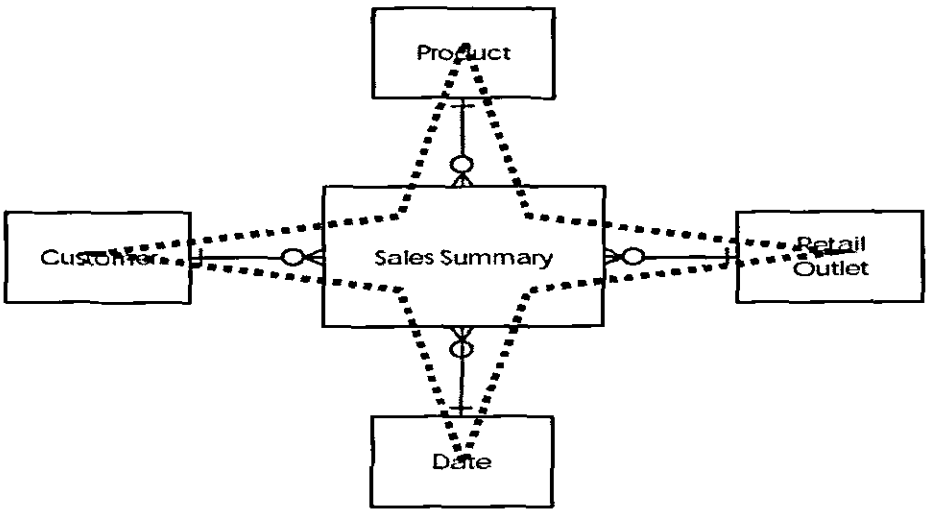


Figure 2-4: Star schema example

The advantage of using star schema is that it reduces the number of tables in database and the number of relationships between them and the number of joins.

Kimball's[13] design approach is the fist approach it contains following steps. It begins by identifying the facts that need to be aggregated and then dimensional attributes to aggregate by and forming star schema based on these. There are different data warehouse design approaches that are discussed in [12] e.g. star schema, snow flake schema and so on.

2.1.9 Conceptual and context based combination of schema matchers

Schema matching is an important operation of schema integration which is a basic problem in many database applications, like data warehousing, E-commerce [18]. Schema matching takes two schemas as input and produces correspondence between schema elements. Different schema matching approaches SemInt, SKAT and COMA are discussed in [19]. In this paper, author proposed a semi automatic schema matching approach. This approach is conceptual and context based combination of schema matchers (C3SM). It has following features concept based matching and correspondence between elements is established. It uses the hybrid matchers that are the combination of different matchers. In future, author wants to

In the following subsection, I discuss the above mentioned phase of SI that is, merging and restructuring.

2.1.10.1 Merging and restructuring

Once the component schemas are conformed, they are merged by means of superimposition of common concepts giving rise to some intermediate integrated schema(s). The intermediate results are analysed, and if necessary restructured in order to achieve several desirable qualities. An integrated schema may be tested against the following qualitative criteria:

— *Completeness and Correctness:* The integrated schema must contain all concepts present in any component schema correctly. The integrated schema must be a representation of the union of the application domains associated with the schemas.
— *Minimality:* If the same concept is represented in more than one component schema, it must be represented once in the integrated schema.
— *Understandable:* The integrated schema should be easy to understand for the integrator and for the end (global) users. This implies that among the several representations of results of integration allowed by a data model, the one that is (qualitatively) the most understandable should be chosen.

Conclusion of Literature Survey

As we have seen the "*Literature Survey*", we have discussed different areas to highlight the problems. We have discussed the two main issues that are the metadata repository is important for a data warehouse and it is a complex task to establish a metadata repository especially from multiple operational sources.

As we have seen the current survey, the schema design from multiple operational sources w.r.t data warehouse is not much focused. However, due to specific needs of the data warehouse domain, the problem of fetching metadata from multiple operational sources must be studied in the perspective of data warehousing specifically. We have also identified the problem of schema integration that is naming conflict. Based on this problem, we have given the detail of schema integration, which consists of schema matching and schema merging. In

the literature survey, we have also discussed the schema matching technique and the concept of schema transformation.

2.2 Problem Statement

Staudt, M [4] discovered a problem "that metadata repository is truly application specific i-e dedicated to data warehousing". Research on metadata that we have discussed in the literature survey had considered that the metadata repository is application/organization specific so it can not be reused by any other application/organization. This approach has following problems:

- The application based repository is used for only a specific application of functional purpose and can not be reused by any other data warehouse. So we have to build it for every organization. It required more time and work to build a repository for every organization.
- The existing repositories that we have discussed in the literature review are the tool specific. They are dependent upon platform that means repository is dependent upon the database management system. Only those can be used which are supported by database management system. Rochade and Platinum [4] repository documentation is not publicly available which makes problem in adoption of their proposals.
- Based on the literature survey we have seen that the problem of multiple operational sources that is naming conflict. This problem occurs when we are doing schema matching i- a part of schema integration. Here, we are going to focus this problem. Name matching works under the assumption that if it has two schema elements have same names they model the same. This may result some very ambiguous results e.g. name could be of an institute or of a student.
- Based on the survey, we have seen that there is a need to integrate the schema matching technique C3SM [19] with schema merging so that it could be used in any schema integration application like data warehousing.

I believe that there is a clear need for metadata repository within data warehouse, so that to better facilitate data warehouse developer, and to make the repository generalize so that reuse of repository can be possible. The remainder of this thesis presents a new methodology for metadata repository.

Chapter 3

RESEARCH METHODOLOGY

The structure of this chapter is as follows. Section 3.1 gives the scope of GMR. Section 3.2 provides the methodology of research that we have adopted.

3.1 Scope of the proposed solution

The scope of the research work is as follows:

- Propose an architecture of GMR
- Propose a semi automatic approach of schema integration based on C3SM [19]
- Develop generic operations for schema integration and transformation
- Implement a prototype

3.2 Methodology

This section describes the methodology of the proposed solution and presents the different phases into which the methodology has been classified. Following is a brief description of this proposed methodology.

The tasks of the proposed generic metadata repository methodology are: first to convert the E-R schema into XMI schema. After conversion into XMI schema, system will extract the schemas from XMI files and analyses the schema for schema matching and merging. The merging is applied on schemas to establish an integrated schema. After integrating the schema, integrated schema is transformed into the data warehouse schema. Both integrated schema and the transformed schema information is stored into the XML documents.

3.2.1 Phases of the proposed methodology

The methodology proposed in this thesis (GMR) decomposes into four phases. These are:

3.2.1.1 *Conversion and extraction,* in which Entity Relationship schemas are converted into the XMI and then schema information is extracted;

3.2.1.2 *Schema integration,* in which schema elements are compared and merge into a single schema;

3.2.1.3 *Schema transformation,* in which the integrated schema is converted into the data warehouse schema i-e star schema; and

3.2.1.4 *Storing into XML documents,* in which the metadata of integrated schema and the data warehouse schema is stored into the generic metadata repository.

The major activities involved in the proposed methodology along with the phases are given below:

3.2.1.1 Conversion and extraction

The aim of conversion activity is to convert an Entity relationship schema of any data source into the XML metadata interchange (XMI). After converting into XMI schema, the process of extraction is perform. This extraction process extracts the schema information using an algorithm so that this schema information is used in the next phase.

3.2.1.2 Schema integration

The aim of this activity is to integrate two schemas. Using the extraction procedure we can get schemas and then we will integrate them into one schema. Schema integration consists of two sub-processes:

(i) Schema matching

(ii) Schema merging

Schema matching and Schema merging:

Schema matching means comparing two schema's to find the semantic relationship between two schema's. Schema merging is the process of integrating several schemas based on their matching into a single schema.

"Currently, schema matching is typically performed manually and perhaps supported by a graphical user interface"[9]. Obviously, manually specifying schema matches is a tedious, time consuming, error-prone, and therefore expensive process. In [9], author says that match operation should be independent component and it must be generic means it can apply to many difference application domains. This requires automated support for schema matching. To provide this automated support, we would like to see a generic, customizable implementation of Match that is usable across application areas. But full automatic match can not be performed because of heterogeneity. Fortunately, there is a lot of previous work on schema matching developed e.g. CUPID, MOMIS and DIKE discussed in [9]. We identify

the problem of name matching in problem statement section. This problem is recovered by applying multiple matchers jointly. We also identify that there is a need of integrating schema matching approach with schema merging. So we have proposed a semi automatic schema integration approach CSI that handles the above problems of name matching by using the concept of C3SM [19] and handles the integration problem.

C3SM based Schema Integration (CSI):

The CSI is a novel schema integration approach with following features:

- It uses multiple matchers, and
- It provides the integrated schema

The match operation takes two schemas as an input and produces a matching result that which elements of the input schemas are logically related to each other. The matching results specify the matching elements with a value between 0 and 1. A 0 indicates total dissimilarities and 1 indicates strong similarities. Based on 0 and 1 matching values merging operation is applied. At the merging stage there is an option that is given to the user that select one schema whose naming will be applied to the integrated schema.

To perform schema matching and merging we are using Word Net 1.6 [10] dictionary but there are few words that are specific to domain and the dictionaries available to us does not contain these words. For example, Student_Id, Emp_Id. Here student id is roll no or registration number and Emp_Id is related to Employee Id or Social Security Number. But here Emp_Id is acting as abbreviation for Employee Id. We could not find this information in Word Net dictionaries and to perform matching and merging with the use of such kind of words we have to create our own dictionary which stores these words. This dictionary is custom (user-defined) dictionary which contains three kinds of words.

- Word
- Synonym
- Abbreviation

Matchers of CSI:

In this section, we are discussing the matchers of CSI. In CSI we combine Exact, Synonym, Abbreviation and Dice Matcher and Merger.

(i) ***Exact Matcher and Merger***

In exact matcher, there is character to character matching involved. For example:

- Student = Student
- Teacher = Teacher

If matcher produces a value of 1 it means match between elements is found and exact merger is applied to merge the elements, otherwise if matcher produces a value of 0 it means match does not found and exact merger is not applied.

(ii) ***Synonym Matcher and Merger***

In synonym matcher, synonyms are used from WorldNet Dictionary and also from our own user defined dictionary. For example:

- University = Institute
- Writer = Author
- Instructor = Teacher

If matcher produces a value of 1 it means match between elements is found and synonym merger is applied to merge the elements, otherwise if matcher produces a value of 0 it means match does not found and synonym merger is not applied.

(iii) ***Abbreviation Matcher and Merger***

In abbreviation matcher, custom dictionary is used to match the schema elements. For example:

- Registration Number = RegNo
- First Name = FName
- Student = std

If matcher produces a value of 1 it means match between elements is found and abbreviation merger is applied to merge the elements, otherwise if matcher produces a value of 0 it means match does not found and abbreviation merger is not applied.

(iv) ***Dice Matcher and Merger***

In dice matcher, we have defined a formula named as dice formula to match the element 1 and element 2.

The dice formula is:

Dice = (number of same character of element 1 in element 2 +
number of same character of element 2 in element 1)
/
(total number of characters of element1 +
total number of characters of element 2)

Example of Dice Macther:

Consider an example to calculate matching using dice formula. Here element 1 and element 2 are:

Element1 = Registration
Element2 = Regoistration

Now applying a dice formula,

Dice = (12+5)/12+13 => 17/25 =0.68

According to the dice formula, the matching value is 0.68.

Here we have seen that there is only a difference of single character in both elements but the matching value is very small. So here I have defined another formula named as reverse dice, in which I have reversed all the characters of both elements and then the dice formula is applied. Therefore, when we applied the dice formula to match the characters, the matching must be in order e.g. if one element is checked once the it should not repeat it again.

Example of Reverse Dice:

Consider an example of reverse dice. Here the element 1 and element 2 have the same values that are defined in the above example. Now reverse the characters of element 1 and element 2, and then apply the reverse dice match formula.

Element1: noitartsiger

Element2: noitartsioger

Dice = (12+12)/12+13 =>24/25 =>0.96

According to the reverse dice concept, the matching value is 0.96.

Here we can get two values of dice matching, so we have to take one maximum value from the both values. So the formula is:

max(Dice (e1,e2) , Dice (e1.reverse,e2.reverse))

In dice match we will take the maximum resulting value according to the value that is defined. If the value is 0.8 then we should take the matching elements whose resuting value is greater than 0.8. When we find the matching value greater than the value that is defined then the dice merger is applied to merge the elements..

Sequence of matchers:

The matchers are applied in the following way:

First of all exact matcher is applied, it takes the schema element names as such. The elements for which matcher is not found through exact matcher then synonym matcher is applied. It compares the schema elements with their synonyms. The elements for which match is not found through synonym matcher then abbreviation matcher is applied. It compares the schema elements with their abbreviation. The elements for which match is not found through abbreviation matcher then dice matcher is applied.

Process of integration

In our methodology, first of all schema is extracted from XMI and store in the data structure name Xmidatabae. So we will create two schema objects of XMIdatabase, named as db1 and db2. In XMIdatabase we have list of tables and relationships. In the table list we have list of columns. We will create two list from db1 and db2, where all elements from db1 store in list1 and all elements from db2 store in list2. We will save the elements in the following form.

Table Name: Column name

Example:

Suppose there are two tables, student and course.

Student(regno,name,semisterno,gpa), Course(code, title, credithour)

The list stores the table name and column name in the following form:

List is:

Student

Student:regno

Student:name

Course

Course:code

Course: title

Now we will compare the elements from list1 with the elements from list2. We will take one element from list1 and apply matching technique with all elements of list2. First of all we will apply the exact matching, if no match found from exact matching then we will apply the synonym matching, if no match found from synonym matching then we will apply the abbreviation matching, and if no match found from all above then we will apply the dice matching. If no matching value found from dice matcher it means there is no matching found so ignore the element.

Cases of matching:

There are four matching cases, which are as follows:

(i) Table with Table matching

(ii) Table with Column matching

(iii)Column with Table matching

(iv)Column with Column matching

These matching cases will use integration approach for matching and merging that is CSI.

(i) Table with Table Matching

If both the elements that we are matching are tables, and there is matching found then we will simply merge them according to the programming technique. E.g.

Schema 1: Course(name,code, cradithour)

Schema 2: Subject(code,title,cradithour)

In this case Course = Subject, so there is table with table matching.

(ii) Table with Column Matching

If the first element is table and the second element is column then we will first find the table of second element and then check that first element has relationship with the table of second element. If it has relationship then we will merge both elements, acording to the programming technique. E.g.

Schema1: Student(regno,name,gpa)

Address(regno,houseno,sreetno,ciy, country)

Schema 2: Student(regno,name,gpa,address)

In this case Address = Student:address, so there is table-column match. In this example, first we will check Address has relationship with student in first schema. If relationship exists then elements will merge according to the programming technique, if no relationship found then we will ignore this matching. In this example relationship is found.

(iii) Column with Table Matching

If the first element is column and the second element is table then we will first find the table of first element and then we will check that the second element has a relationship with table of first element. If it has a relationship then we will merge elements, acording to the programming technique. E.g.

Schema1: Student(regno,name,gpa,address).

Schema 2:Student(regno,name,gpa)

Address(regno,houseno,sreetno,ciy, country)

In this case, Student:address= Address, so there is column-table match. In this example, first we will check that address has a relationship with student in the second schema. if relationship exists than the elements will merge according to the programming technique, if

no relationship found then we will ignore this matching. In this example relationship is found.

(iv) Column with Column Matching

If both the elements that we are going to match are the columns then we will find the table of both elements, and then we will check whether both tables are same. If both tables are same then we will merge the columns according to th programming technique.E.g.

Schema1: Course(code,name)

Schema2: Subject(code,cradithoure,title)

In this case, course:code=subject:code, so there is column-column match. In this example, first we will find the tables of both elements (code ,code). Here course and subject are the table of element1 and element2 respectively. Both tables are same so merge the columns.

Consider another example:

Schema1 Student(name,regno)

Schema2 Course(name,code)

In this case, Student:name=couse:name, there is no column match because both tables are different. So we will discard the matching.

3.2.1.3 Schema transformation

The aim of this activity is to transform schema into data warehouse schema. Using the integration procedure we can get integrated schema and then we will transform the integrated schema into data warehouse schema. "Kimball proposed a new technique for designing data warehouses, which he called dimensional modeling"[13]. Our schema transformation is based on the Kimball's technique [13]. The basic building block that is used in dimensional modeling is star schema. The integrated schema is used to transform into the data warehouse schema that is star schema.

Star schema

Star schema has one large central table (fact table) and a set of smaller tables (dimensions) linked with the fact table.

Fact table:

A fact contains the measurements. A fact table consists of multiple foreign keys, each linked with a primary key in a dimension table. Using the integrated schema, system will find the fact table and its attributes and then this schema information is stored into the metadata repository.

Dimension table

The dimension table contains the textual attributes. There could be any number of dimensions in star schema. Using the integrated schema, system will find the dimensions and their columns and these schema information is then stored into the metadata repository.

3.2.1.4 Storing into XML documents

The aim of this activity is storing the metadata of both operational source schema and the data warehouse schema into the metadata repository using XML documents in the form of XML schemas. "XML schemas contain elements having sub elements which further contains other sub elements or attributes [9]".

Chapter 4

SYSTEM ARCHITECTURE

4 System Architecture

System architecture is the design or set of relations between the parts of a system. It is the most important, pervasive, top level, decisions and then associated rationale about the overall structure. As stated by Bass, Clements and Kazman, the software architecture is the structure or structures of the system, which comprises software components, the externally visible properties of those components and the relationship among them [22].

The architecture of generic metadata repository has been discussed in this chapter. This architecture is supports the methodology described in the previous chapter. The architecture of GMR, which consists of following operations: (a) extract schema from XMI, (b) integrate schemas, (c) transform relational schema into star schema, and (d) store the metadata about both schema into repository using xml documents.

The structure of this chapter is as follows. Section 4.1 gives the design of the system using the viewpoints. Section 4.2 presents the architecture of the system. Section 4.3 provides the behavioral description.

4.1 System Analysis and Design

System analysis and design is the specification or construction of a technical, computer based solution for business requirements identified in the system analysis. It is the evaluation of alternative solutions and the specification of a detailed computer based solution. It is basically the design of the information processing system covering the activities of determining detailed requirements, design of data and information flow. To analyze and design a system we are using viewpoints oriented approach.

4.1.1 Viewpoints

Viewpoints can be used as a way of classifying stakeholders. Stakeholders range from system end users through managers. Each viewpoint specifies a complete functional unit. This means

2.1

Reference: Admin sign in

Rationale: Admin will sign in to get the functionality of the system.

Specification: Admin will enter login and password to sign in. If the login and password is correct then admin can do any functionality of the system otherwise admin can not.

VPs: Admin

Non Functional Requirement: Password must be of more more than 7characters.

2.2

Reference: Manage dictionary

Rationale: Admin will manage the dictionary for schema matching.

Specification: Admin will enter login and password to manage the dictionary. Admin can manage the words, synonyms and abbreviation for the dictionary.

1: Admin selects the Add new word option

- Admin enter the new word.
- System save the word and display the successfully saved message.

1a: If the word already added in to the dictionary. System display error message and reject entries.

2: Admin selects the update word option

- System displays all word information.
- Admin select word, and update the word information according to the requirement.
- System save these updates and display successful message.

2a: if word is not selected. System displays error and asks to select the word.

2b: if update word is already exiting the dictionary. System displays error message and reject entry.

2.2

Specification:

3: Admin selects the add new synonyms option

- Admin select the word.
- System displays all word that is not synonym of selected word.
- Admin select the synonyms for selected word.
- System saves the synonyms and display successfully message.

4: Admin selects the remove synonyms option

- Admin select the word
- System displays all the synonyms.
- Admin select the synonyms.
- System removes the selected synonyms, and display successfully message.

5: Admin selects the new Abbreviation option

- Admin select the word.
- Admin enter abbreviation.
- System save the abbreviation and display successfully message.

  4a: If the abbreviation already added in to the dictionary. System display error message and reject entries.

6: Admin selects the remove Abbreviation option

- Admin select the word.
- System displays all abbreviation of selected word.
- Admin selects an abbreviation.
- System removes the selected abbreviation and display successfully message.

  5a: If the abbreviation not selected. System display error message and ask the admin to select an abbreviation

VPs: Admin

Non Functional Requirement: Null

2.3

Reference: Extract schema

Rationale: Admin will sign in to extract the schema.

Specification: Admin will sign in to extract schema. Admin selects the extract schema option. System displays all schemas on the screen and Admin will select the two schemas. System displays the schema information and save the information into the database.

VPs: Admin

Non Functional

Requirement: Admin should have to select the schemas.

2.4

Reference: Integrate schema

Rationale: Admin will sign in to integrate the schema.

Specification: Admin will sign in to integrate the schema. Admin selects the schema integration option. System displays all schemas on the screen and Admin will select the two schemas and submit the schemas for integration. System will display the schema integration result and save the report of the integration.

VPs: Admin

Non Functional

Requirement: Admin should have to select the schemas.

2.5
Reference: Transform schema
Rationale: Admin will sign in to transform the schema.
Specification: Admin will sign in to transform the schema. Admin selects the schema transformation option. System displays all schemas on the screen and Admin will select the two schemas and submit the schemas for integration. After schema integration admin will apply the schema transformation. System will display the schema transformation result and save the report of the transformation.
VPs: Admin
Non Functional Requirement: Admin should have to select the schemas.

4.2 System Architecture

We propose architecture for a Generic Metadata Repository (GMR). This architecture discusses the problem of integrating heterogeneous operational sources and storing metadata into the XML documents so that any organization can reuse it.

4.2.1 GMR Architecture Context Diagram

The GMR architecture context diagram consists of four parts input, processing, output and user interface as shown in figure 4-2. The repository takes input schema object and then processes the schema objects and generates the output in form of XML files. The user interface module acts as a client acquiring the services of the component and provides the user interface in which the use can specify the schema object.

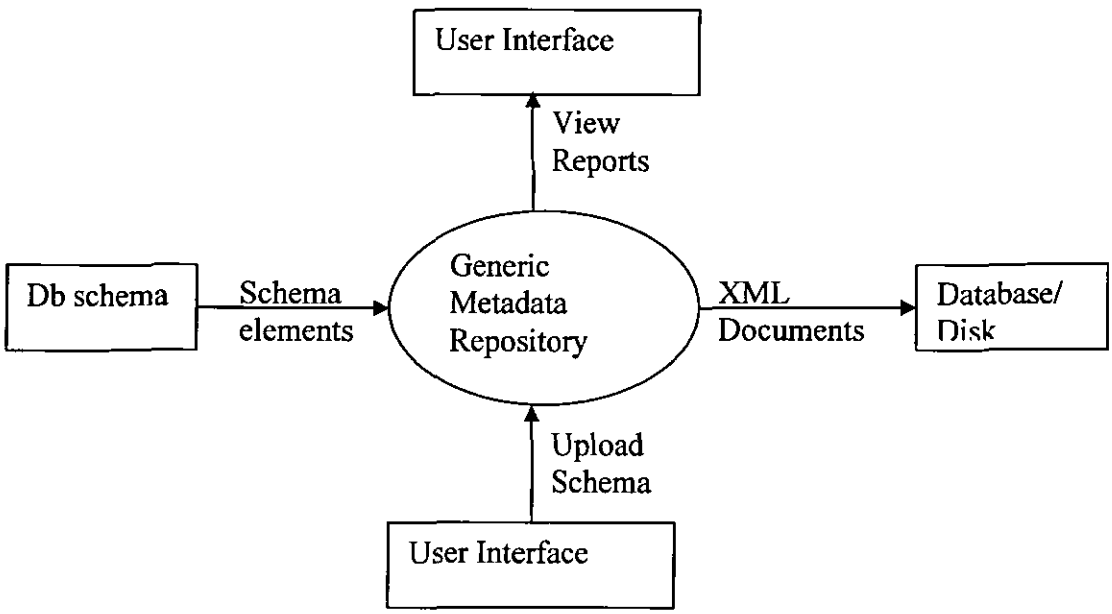


Figure 4-2: GMR Architecture Context Diagram

4.2.2 Three tier Architecture flow diagram

The system will comprise of 3-tier layered named as data layer, application layer and client layer respectively. Data layer will provide the data to the system. Application layer will extract the information from data layer and will manipulate it and resultant outcome will be sent to the client layer. The client layer will be based on the GUI and will display the results to the end user. Figure 4-3 depicts the 3-tier layer.

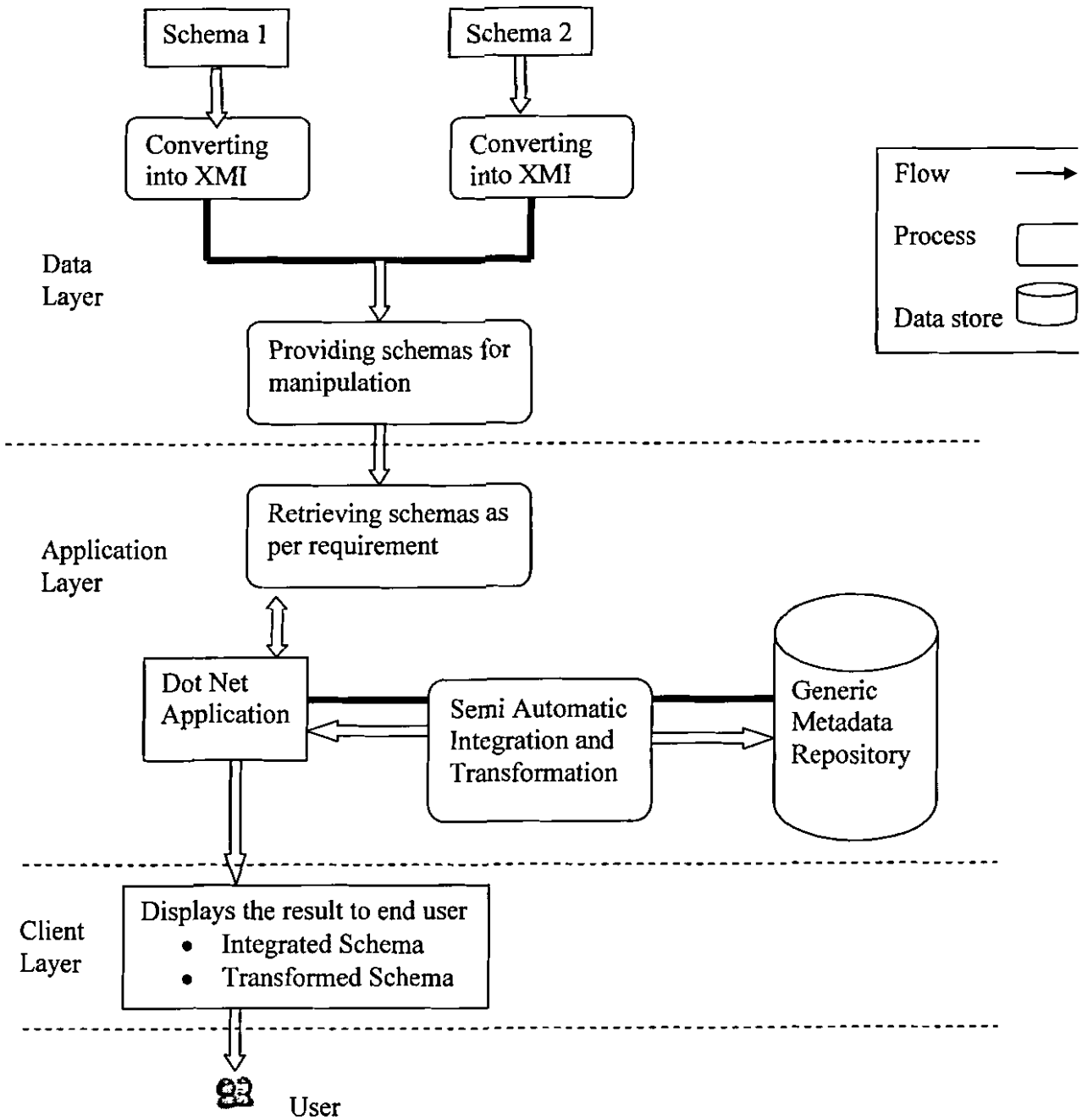


Figure 4-3: The 3-tier architecture

4.2.3 GMR Architecture flow diagram

The GMR consists of following components as shown in figure 4-4. The main components are extracting the schema objects, integrate schema, transform the schema and store the schema into the generic metadata repository.

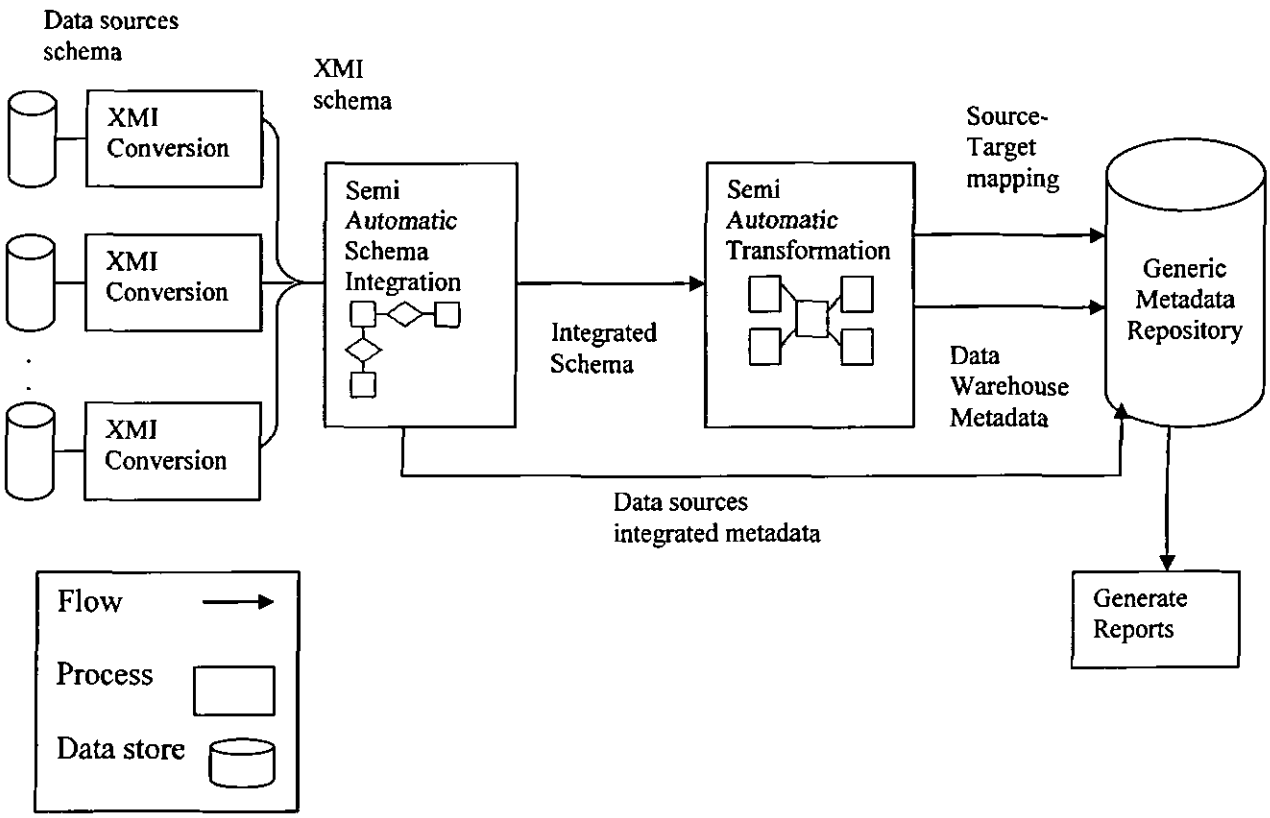


Figure 4-4: GMR Architecture flow diagram

4.2.4 Components of architecture

The architecture of generic metadata repository consists of several components. Each component performs functions and services. The following components are:

(i) Data Sources Schemas:

There are different data source schemas and these schemas are available in the form of entity relationship diagram.

(ii) Conversion into XMI:

The entity relationship diagrams of operational sources are then converted into the XML metadata interchange (XMI). It solves the problem of integrating heterogeneous operational sources.

(iii) Load Schema:

Once the XMI file is generated then it is required to load a particular file into a system. By using an application form user can load the XMI file into the system and it saves into the particular location. The file contains all the information about the data source schema.

(iv) Schema Extraction:

The XMI files that are loaded into the system are then used to extract the schema using generic extract operation. The extraction process extracts all the schema information like table names, column names, data type, length and constraints.

(v) Schema Integration:

Using the extraction procedure we can get schemas then we can apply the integration procedure on the specified extracted schemas. Integration procedure consists of two operations one is schema matching and another is schema merging. First of all we will apply the schema matching using our proposed technique CSI. When we get the matching result then the schema merging is applied. So at the end we will get the integrated schema. The integrated schema information is then stored into the generic metadata repository.

(vi) Schema Transformation:

Once we get the integrated schema by applying CSI approach, we will apply the transformation procedure. The transformation procedure transformed the integrated schema into the data warehouse schema that is star schema. The star schema consists of dimension tables and the fact table. The information about the dimension tables and the fact table is stored in the generic metadata repository.

(vii) Generic Metadata Repository:

Data sources schema and the data warehouse schema information and the mapping between them is stored into the generic metadata repository. This information is then suitable for information delivery. The storage of metadata is based on Extensible Markup Language (XML) documents. These documents are generic and can be reuse.

(viii) Reporting:

Each data source schema is constructed and represented in the form of tree. User can easily understand the schema information. User can see the result of the integration and transformation using reports.

(ix) Interfaces:

The system provides user friendly interfaces in which user can view (explore) the database schema. System can save and load the schema on to the disk for the reuse purpose.

4.3 Behavioral description

A computer program/software always exists in some state. An externally observable mode of behavior (e.g. waiting, computing, printing etc) that is changed when some event occurs (e.g. mouse click etc).

The behavior domain describes a representation of states of software and events that cause the software to change its state. The software behavior is described in two terms:

- States
- Events

The following section discusses the behavior of the GMR in terms of states and events.

States

The GMR software can have the following states.

Ready State

- File reading state
- DB schema retrieving state
- Schema integrating state
- Schema transformation state
- XML files writing state

Events

The following are the events that cause the GMR software to change its state described above.

- Files selected invoke file reading
- File read invoke schema retrieving
- Schema retrieving invoke integration
- Integration complete invoke file writer
- Integration invoke transformation
- Transformation complete invoke file writer
- File writing complete invoke ready state

State transition diagram

The state transition diagram of the system is shown in figure 4-5.

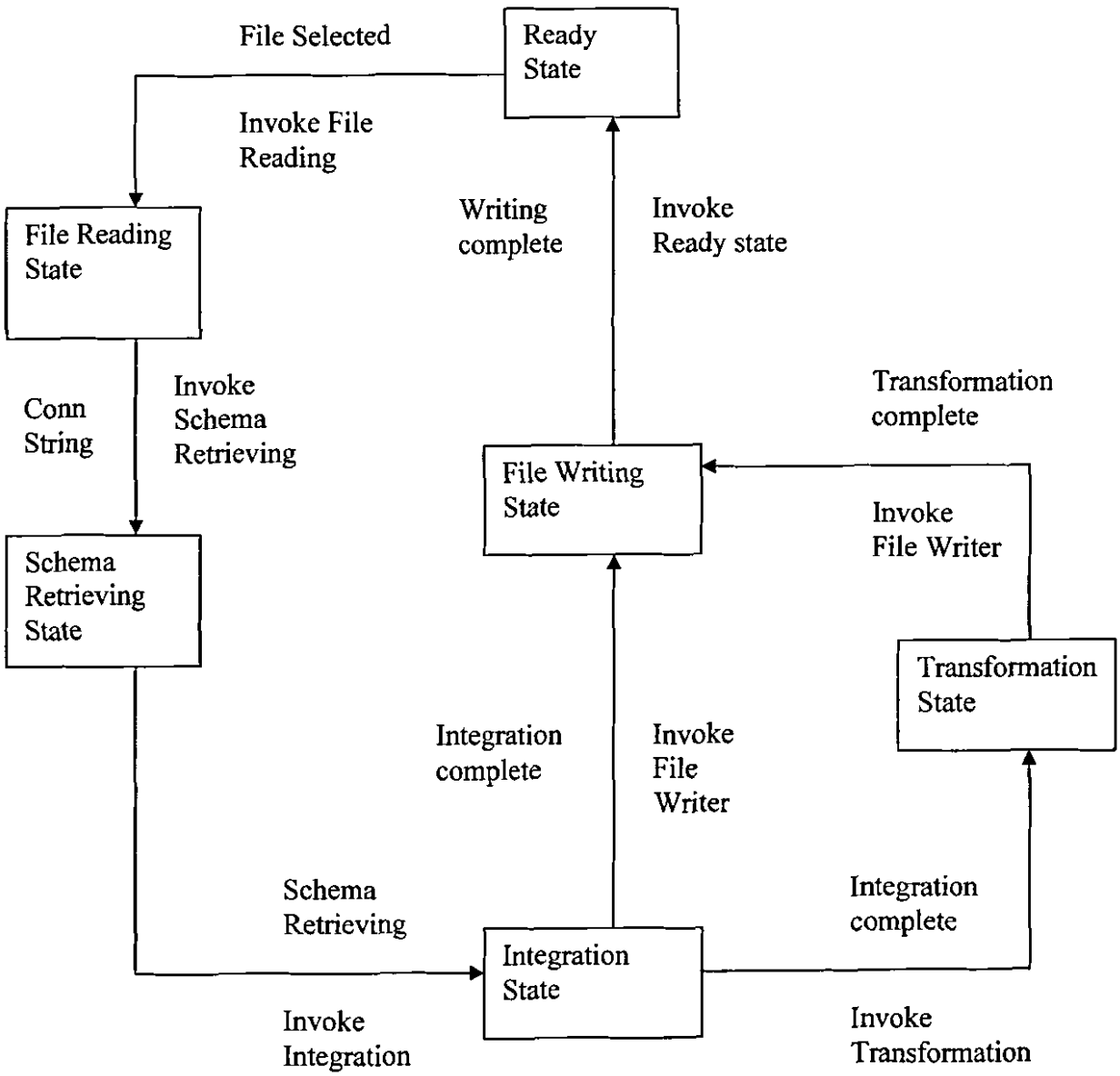


Figure 4-5: State transition diagram

The above diagram shows the control specification of the software that is described here. Following is the scenario in which the software could be used.

The system is in ready state; user selects the database schema and invokes the file reading; then it invokes the schema retrieving then integration state is invoked , after that XML writer is invoked, then transformation is invoked, after that XML writer is invoked, now storing is complete and system is again in ready state.

Chapter 5

IMPLEMENTATION

5.1.2 XML

The XML stands for Extensible Markup language, it's a component language used for describing information as an electronic document and it stores the information intelligibly that's why it is called a Meta markup language [15].

The XML is a general-purpose *specification* for creating custom markup languages. It is classified as an extensible language because it allows its users to define their own elements. Its primary purpose is to help information systems share structured data, particularly via the Internet, and it is used both to encode documents and to serialize data. XML is recommended by the World Wide Web Consortium (W3C). It is a fee-free open standard. The recommendation specifies both the lexical grammar and the requirements for parsing.

Area in which XML will be useful in the near-term include: Exchange of information between organizations. In XML the documents are created by concentrating on what actually information is and how it is structured. The logical structure of the XML document (schema of XML document is defined either in the DTD or MSXML schema, either internally or in some external file.

Advantages of XML

There are many advantages to using XML for information exchange, and they offer many benefits to the user. The Extensive Markup Language uses human language, which is conversable, and not the language used by computers which is binary and ASCII coded. XML is readable by even people who have had no formal introduction to XML or have been coached on it. It is as easy as HTML. XML is fully compatible with applications like JAVA, and it can be combined with any application which is capable of processing XML irrespective of the platform it is being used on. XML is an extremely portable language to the extent that it can be used on large networks with multiple platforms like the internet, and it can be used on handhelds or palmtops or PDAs. XML is an extendable language, meaning that you can create your own tags, or use the tags which have already been created. There are other advantages of using XML. It is a platform and independent language. It can be deployed on any network if it is amicable for usage with the application in use. If the application can work along with XML, then XML can work on any platform and has no

boundaries. It is also vendor independent and system independent. While data is being exchanged using XML, there will be no loss of data even between systems that use totally different formats. XML can also be stored in databases in XML format and human readable format. The advantages of XML include that it can be used as an instrument to share data and application models in wide networks like internet. It supports Unicode, allowing almost any information in any written human language to be communicated. It can represent common computer science data structures: records, lists and trees. Its self-documenting format describes structure and field names as well as specific values. The strict syntax and parsing requirements make the necessary parsing algorithms extremely simple, efficient, and consistent. XML is heavily used as a format for document storage and processing, both online and offline. It is based on international standards.

5.1.3 XMI

The **XML Metadata Interchange (XMI)** is an Object Management Group (OMG) standard for exchanging metadata information via Extensible Markup Language (XML). One purpose of XML Metadata Interchange (XMI) is to enable easy interchange of metadata between UML-based modeling tools and MOF-based metadata repositories in distributed heterogeneous environments. XMI is also commonly used as the medium by which models are passed from modeling tools to software generation tools as part of model-driven engineering.

5.2 Algorithms

The algorithms of system operations are given below:

5.2.1 *Algorithm of integration*

This is algorithm of matching two schemas and make Master schema.

Algorithm

The pseudo code for this algorithm is given below. Following variables are use globally on all functions

```
XMIDataBase database1  : Schema of database 1
XMIDataBase database2  : Schema of database 2
```

```
Int matchValue = 0.90
List list1 = makeList(database1)
List list2 = makeList(database2)
bool[] matches = new bool[size of list1]
string line1,line2,element1, element2
int i = 0,rs


Function makeMasterDB()

      While line1 in list1 then
          element1 = extractElement(line1)
          First applying exact matching
          While line2 in list2 then
                element2 = extractElement(line2)
                If element1 = element2 then
                   rs = 1
                Else
                  rs = 0
                End If
                If rs = 1 then
                      createMasterTable()
                End If
            End Loop
            If matches[i] == false then

                  Dictionary Matching
                  While string line2 in list2
                        element2 = extractElement(line2)
                        rs = dictionary_match(element1, element2)
                        If rs = 1 then
                              createMasterTable()
                        End If
                  End Loop
             End If
             If matches[i] = false then

                  Dice Matching Here
                  While string line2 in list2 then
                        element2 = extractElement(line2)
                        rs = diesMatch(element1, element2)
                        If rs >= matchValue then
                              createMasterTable()
                        End If
                  End Loop
             End If
             i++
      End Loop
      saveRelaionShips()
      saveForignKeys()

End Function
```

This is the function to compare two elements with exactly, using dictionary and dice.

```
Function double compareElement(string element1, string element2)

      double rs = 0
      If element1 = element2 then
            rs = 1
      Else If (rs = dictionary_match(element1, element2)) = 0 Then
            double rs1 = dies(element1, element2)
            double rs2 = dies(revert(element1), revert(element2))
            rs = (rs1 > rs2) ? rs1 : rs2
      End If
      return rs

End Function
```

This is the function return maximum result of two dies.

```
Function double diesMatch(string element1, string element2)

      double rs1 = dies(element1, element2)
      double rs2 = dies(revert(element1), revert(element2))
      return (rs1 > rs2) ? rs1 : rs2

End Function
```

This function return Boolean value, whither element is column.

```
Function boolean isColumn(string element)

      string[] sp = Split the element from ':'
      if sp.Length == 2 then
            return true
      End If
      return false

End Function
```

This function Return table name from given element

```
Function string ExtractTable(string column)

      string[] sp = Split the column from ':'
      return sp[0]

End Function
```

This function makes the list of element from given XMIDataBase.

```
Function List makeList(XMIDataBase db)
      List list
      While XMITable table in db.tables then
            list.Add(table.name)
      End Loop
      While XMITable tab in db.tables then
            While Column col in tab.columns then
                  list.Add(tab.name+":"+col.name)
```

```
            End Loop
      End Loop
      return list
End Function

Function string extractElement(string element)

      string[] sp = split the element from ':'
      If sp.Length = 2 then
            return sp[1].ToLower()
      Else
            return element.ToLower()
      End If

End Function

This function match two element by using dictionary.

Function float dictionary_match(string element1, string element2)

      List list = List of Synonyms of element1 from custom dictionary
      List wornetList = List of Synonyms of element1 From WordNet
      dictionary
      While string syn in list then
            If wornetList not contain syn then
                  wornetList.Add(syn)
            End If
      End Loop
      If wornetList contain element2 then
              return 1
      End If
      list = List of all Abbreviation of element1 from custom dictionary
      If list contain element2 then
            return 1
      End If
      return 0

End Function


This function take the input a string and return it's revert string

Function string revert(string element)

      string r = ""
      for (int i = element.Length - 1 i >= 0 i--)
            r += element[i]
      End Loop
      return r

End Function

This function find dies value between two string

Function double dies(string element1, string element2)

      int i = matchElement(element1,element2)
```

```
      int j = matchElement(element2, element1)
      double val =i + j
      double cnt = element1.Length + element2.Length
      return val / cnt

End Function
```

This function return match value of one element with other element

```
Function int matchElement(string element1, string element2)

      char ch1,ch2
      int count=0,i=0,j=0,ind=0
      bool match = false
      for ( i = 0 i < element1.Lengthi++ )
            match = false
            ch1 = element1[i]
            while match = false AND j < element2.Length then
                  ch2 = element2[j]
                  If ch1 = ch2 then
                        match = true
                        count++
                  End If
                  j++
            End Loop
            If match = false then
                  j = ind
            Else
                  ind=j
            End If
      End Loop
      return count

End Function
```

The architecture of master database is as following

- **Master Table**
  - Master Table Id
  - Matching Table1 XMIID
  - Matching Table2 XMIID
  - Matching table1 name
  - Matching table2 name
  - Master table name
- **Master Column**
  - Master Column id
  - Matching Column1 name
  - Matching Column2 name
  - Master column name
  - Data type
  - Length
  - Is Null able
  - is Primary Key
  - is unique
  - XMI id
  - Master table Id
- **Relation ship**
  - Relation ship id

- From Table id1
- To Tableid2
- From Multiplicity
- To Multiplicity
- From table name
- To table name

- **Foreign Key**
  - *Foreign Key Id*
  - Master Column Id
  - Master Column Name
  - Reference column Id
  - Reference Column name

```
This Function match schema and make master database
Function void createMasterTable()
      If isColumn(line1) then

            If isColumn(line2) then
                  Both column case
                  string t1 = ExtractTable(line1)
                  string t2 = ExtractTable(line2)
                  double rc = compareElement(t1, t2)
                  If rc >= matchValue then

                        Creating new Master column

                        XMITable tab = table from database1 with name t1
                        Column col1 = Column from tab with name element1
                        Column col2 = Column from table of name t2 from
            database2 with
                        name element2

                        int mtid=master table id with name=tab.name And
            xmiid=tab.xmiid

                        These are required fields of master column.

                        Matching Column1 = col1.name
                        Matching Column2 = col2.name
                        Master Column = col1.name
                        Data type = col1.datatype
                        Length = col1.length
                        Null able = col1.nullable
                        Primary Key = col1.primaryKey
                        Unique = col1.unique
                        XMIid = col1.xmiid
                        Master table id = mtid

                        Now save the above fields

                        matches[i] = true

                  End If
            Else
                  column table case
                  string tab = ExtractTable(line1)
                  List list = database2.getRelatioship(line2)
                  double rc
```

```
            While XMITable tb in list then
                  string t1 = tb.name
                  rc = compareElement(tab, t1)
                  If rc >= matchValue then

                        Creating new Master column
                        XMITable tab1 = Table from database1 with
            name tab
                        Column col1 = column from tab1 with name
                        element1
                        XMITable tab2 = table from database2 with
      name line2
                        int mtid = masterTable id with name = tab1.name
                        and xmiid = tab1.xmiid



                        These are required field of Master Column.

                        Matching Column1 = col1.name
                        Matching Column2 = tab2.name
                        Master Column = col1.name
                        Data type = col1.datatype
                        Length = col1.length
                        Null able = col1.nullable
                        Primary Key = col1.primaryKey
                        Unique = col1.unique
                        XMIid = col1.xmiid
                        Master table id = mtid

                        Now save the above fields

                        matches[i] = true
                  End If
            End Loop
      END If
Else
      If isColumn(line2) then

            table column case
            string tab = ExtractTable(line2)
            List list = relation ships from database1 with table
            name line1
            double rc
            While XMITable tb in list then
                  string t1 = tb.name
                  rc = compareElement(t1, tab)
                  If rc >= matchValue then
                        Creating new Master table
                        XMITable tab1 = table from database1 with
            name t1
                        Column col1 = Column from table with name
      tab and column                name element2 from database2
                        XMITable tab2 = table from database1 with
      name line1
                        int mtid = master table id with name =
      tab1.name AND
```

```
                              XMIid = tab1.xmiid

                              These are required field of Master Column.

                              Matching Column1 = tab2.name
                              Matching Column2 = col1.name
                              Master Column = tab2.name
                              Data type = col1.datatype
                              Length = col1.length
                              Null able = col1.nullable
                              Primary Key = col1.primaryKey
                              Unique = col1.unique
                              XMIid = col1.xmiid
                              Master table id = mtid

                              Now save the above fields
                              matches[i] = true
                        End If
                  End Loop
            End If

            Else
                  both table case

                  Create a new master table

                  XMITable t1 = table from database1 with name line1
                  XMITable t2 = Table from database2 with line2

                  These are field of master tables

                  Matching table1 xmmid = t1.xmiid
                  Matching table2 xmmid = t2.xmiid
                  Matching table1 name = t1.name
                  Matching table2 name = t2.name
                  Mater table name = t1.name

                  Save the above field in master table

                  matches[i] = true

            End Else
      End If
End Function
```

This function Save the master tale relationships

```
Function saveRelaionShips()
    While Relationship relation in database1.relationship then
          int mid1 = master table id with relation.from.name,
      relation.from.xmiid
          int mid2 = master table id with relation.to.name,
      relation.to.xmiid
          If mid1 not equal -1 AND mid2 not equal -1 then

              Makes relationship of master tables
```

```
                These are required field of relationship

                 From Table Id = mid1
                 To Table Id = mid2
                 From Multiplicity = relation.fromMultiplicity
                 To Multiplicity = relation.toMultiplicity
                 From table name = relation.from.name
                 To Table name = relation.to.name


                Save this relationship
          End If
    End Loop
End Function


This function save the forign key

Function saveForignKeys()

      While XMITable tables in database1.tables then
            While Column col in tables.columns then
                  int cid1 = Master Column Id with col.name and col.xmiid
                  int cid2 = -1
                  If col.fkey not equal null then
                        cid2 = Master Cloumn id with master column name =
                        col.fkey.column.name and  xmiid =
                        col.fkey.column.xmiid
                        If cid1 not equal -1 AND cid2 not equal -1 then
                              Make Foreign Key
                              These are required field of foreign key
                              Master Column Id = cid1
                              Master Column name = col.name
                              Reference Column Id = cid2
                              Reference Column name = col.fkey.column.name
                        End If
                  End If
            End Loop
      End Loop

End Function
```

5.2.3 Algorithm of transformation

The pseudo code for this algorithm is as follows:

```
Function  insertDWDFColumn(int did,int mdid)
         Make a list of all columns of integrated schema
            ArrayList list = this.selectColumn(mdid);
            While ob in list then

                if (ob[2].Equals("varchar") || ob[2].Equals("char")) then
```

```
                    insert into the data warehouse dimension table
                    columns

            End if
            if (ob[2].Equals("integer") || ob[2].Equals("date")) then

                    insert into the data warehouse fact table columns

            End else

        End Loop

End Function
```

Chapter 6

EXPERIMENTAL RESULTS

6 Experimental Results

In this section, we have built a case study in order to show the working of our proposed framework. We construct the case study from the domain of "Student Information System". A domain expert can make the entity relationship diagrams of two systems since purpose here to show the working of our integration and transformation approach that we have used in our thesis but not the construction of domain knowledge.

The structure of this chapter is as follows. Section 6.1 gives the results of matching results. Section 6.2 gives the results of merging results and the section 6.3 gives the transformation results.

Example Schemas

The two schemas that we have taken as an example in our case study are given below. Based on these schemas the matching, merging and transformation results are displayed. The entity relationship diagram of example schema 1 is shown in figure 6-1 (a).

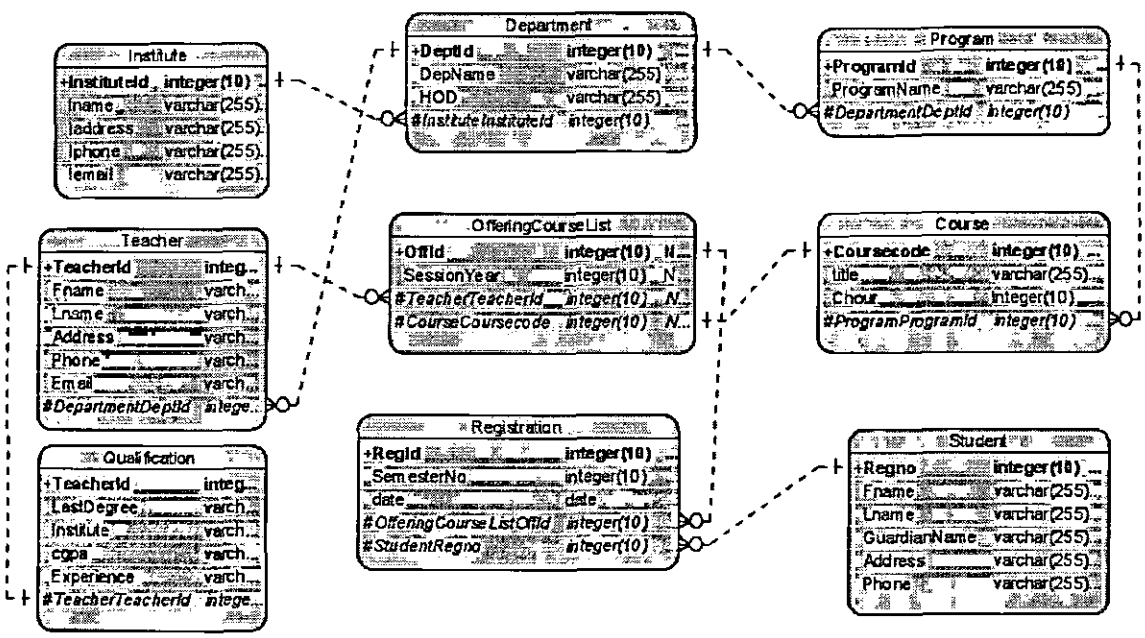


Figure 6-1 (a): Example Schema 1

The entity relationship diagram of example schema 2 is shown in figure 6-1 (b).

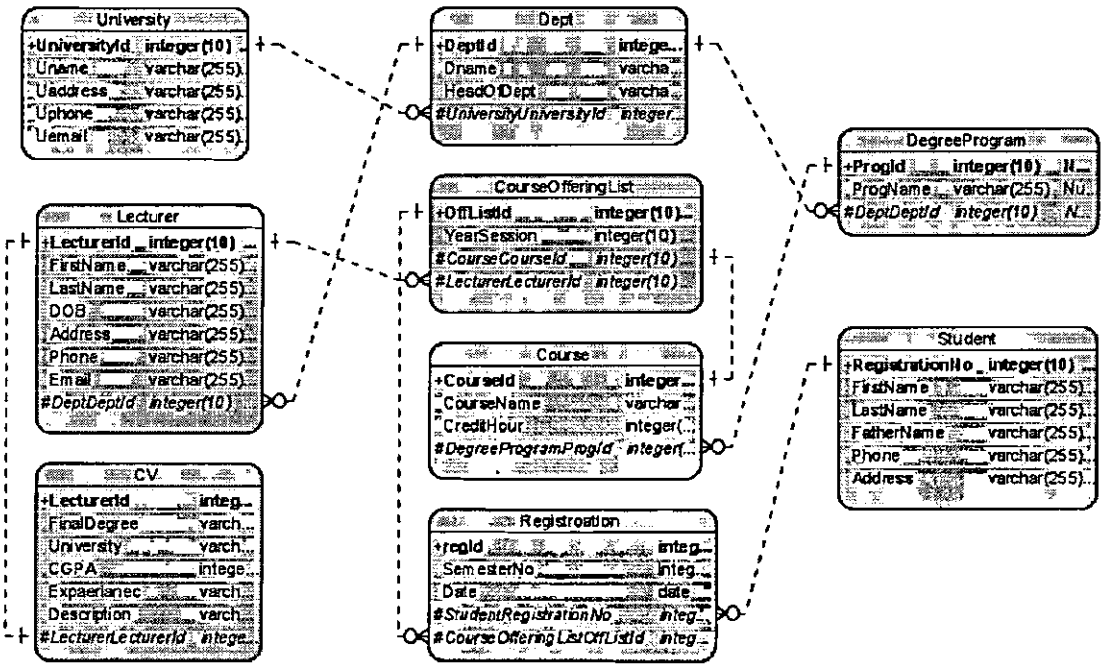


Figure 6-1 (b): Example Schema 2

6.1 Matching Result

Based on the example schemas when we apply our match operation then the matching result is produced. The matching result is shown in table 6-1 (a) and table 6-1 (b).

In table 6-1 (a), first two columns show the schema element table names taken from the two example schemas. Next four columns show the matching results obtained by applying different matchers.

Element 1	Element 2	Exact Matcher	Synonym Matcher	Abbreviation Matcher	Dice Matcher
Student	Student	1	0	0	0
Institute	University	0	1	0	0
Course	Course	1	0	0	0
Registration	Registroation	0	0	0	0.9599
Qualification	CV	0	1	0	0
Teacher	Lecturer	0	1	0	0
Department	Dept	0	0	1	0
Program	DegreeProgram	0	0	1	0

Table 6-1(a): Matching result

In table 6-1 (b), first two columns show the schema element table names and column names taken from the two example schemas. Next four columns show the matching results obtained by applying different matchers.

Element 1	Element 2	Exact Matcher	Synonym Matcher	Abbreviation Matcher	Dice Matcher
Student:Fname	Student:FirstName	0	0	1	0
Student:Lname	Student:LastName	0	0	1	0
Student:Regno	Student:RegistrationNo	0	0	1	0
Student:Phone	Student:Phone	1	0	0	0
Student:Address	Student:Address	1	0	0	0
Student:GuardiannName	Student:FatherName	0	1	0	0
Institute:Iaddress	University:Uaddress	0	1	0	0
Institute:Iname	University:Uname	0	1	0	0
Institute:Iphone	University:Uphone	0	1	0	0
Course:title	Course:CourseName	0	1	0	0
Course:Coursecode	Course:CourseId	0	0	1	0
Course:Chour	Course:CreditHour	0	0	1	0
Registration:SemesterNo	Registroation:SemesterNo	1	0	0	0
Registration:RegId	Registroation:regId	1	0	0	0
Registration:date	Registroation:Date	1	0	0	0
Qualification:Institute	CV:University	0	1	0	0
Qualification:cgpa	CV:CGPA	1	0	0	0
Qualification:LastDegree	CV:FinalDegree	0	1	0	0
Teacher:Fname	Lecturer:FirstName	0	1	0	0
Teacher:Phone	Lecturer:Phone	1	0	0	0
Teacher:Lname	Lecturer:LastName	0	1	0	0
Teacher:Address	Lecturer:Address	1	0	0	0
Teacher:Email	Lecturer:Email	1	0	0	0

Department:DeptId	Dept:DeptId	1	0	0	0
Department:DepName	Dept:Dname	0	0	1	0
Department:HOD	Dept:HeadOfDept	0	0	1	0
Program:ProgramId	DegreeProgram:ProgId	0	0	1	0
Program:ProgramName	DegreeProgram:ProgName	0	0	1	0

Table 6-1(b): Matching result

6.2 Merging Result

When the matching result is produced then the merging operation is applied. The merging result is shown in table 6-2 (a) and table 6-2 (b).

In table 6-2 (a), first two columns show the schema element table names taken from the two example schemas. Third column is the name of integrated element.

Element 1	**Element 2**	**Integrated Element**
Student	Student	Student
Institute	University	Institute
Course	Course	Course
Registration	Registroation	Registration
Qualification	CV	Qualification
Teacher	Lecturer	Teacher
Department	Dept	Department
Program	DegreeProgram	Program
Student	Student	Student

Table 6-2 (a): Merging result

In table 6-2 (b), first two columns show some schema element column names taken from the two example schemas. Third column is the name of integrated element.

Element 1	Element 2	Integrated Element
Fname	FirstName	Fname
Lname	LastName	Lname
GuardianName	FatherName	GuardianName
Address	Address	Address
Iname	Uname	Iname
LastDegree	FinalDegree	LastDegree
cgpa	CGPA	cgpa
DepName	Dname	DepName
HOD	HeadOfDept	HOD
ProgramId	ProgId	ProgramId
Coursecode	CourseId	Coursecode

Table 6-2 (b): Merging result

6.3 Transformation Result

When the integrated schema is produced then the transformation operation is applied. The transformed result is shown in table 6-3 and 6-4.

In table 6-3, dimension tables and dimension columns are displayed.

Dimension Tables	Dimension Columns
Student	Fname
Student	Lname
Student	GuardianName
Student	Address
Student	Phone
Institute	Iname
Institute	Iaddress
Institute	Iphone
Teacher	Fname
Teacher	Lname
Teacher	Address

Teacher	Phone
Teacher	Email
Qualification	LastDegree
Qualification	Institute
Qualification	cgpa
Department	DepName
Department	HOD
Program	ProgramName
Course	title

Table 6-3: Transformation result (dimension columns)

In table 6-4, fact table columns are displayed.

Fact Table Columns	**Data Type**
Regno	integer
DeptId	integer
ProgramId	integer
Coursecode	integer
Chour	integer
RegId	integer
SemesterNo	integer
date	date

Table 6-4: Transformation result (Fact columns)

Chapter 7

CONCLUSION

7 Conclusion

Based upon the experimental results the following conclusion is drawn and few directions for further study are also given. The structure of this chapter is as follows: Section 7.1 gives the conclusion of the thesis and section 7.2 gives the future work of the research.

7.1 Conclusion

The research presented in this thesis concerns different areas to highlight the problems. We have discussed the two main issues that are the metadata repository is important for a data warehouse and it is a tedious and complex task to establish a metadata repository especially from multiple operational sources as we have seen the problem of multiple operation sources i-e naming conflict.

However, due to specific needs of the data warehouse domain, the problem of fetching metadata from heterogeneous and autonomous resources must be studied in the perspective of data warehousing specifically. Based on this problem, we have given the detail of schema integration, which consists of schema matching and schema merging. In the literature survey, we have also discussed the schema matching technique and the concept of schema transformation.

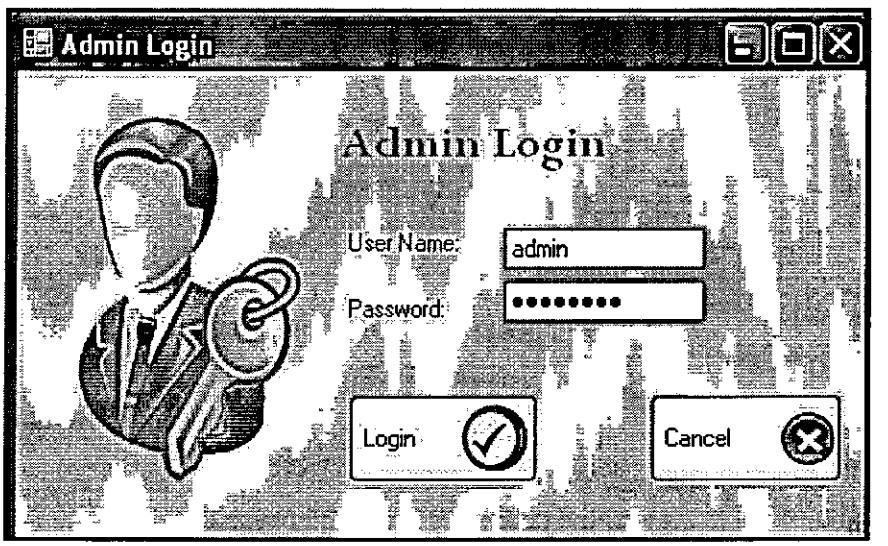
Based on the literature survey, I have chosen to develop a metadata repository within the context of data warehouse that is Generic Metadata Repository (GMR). I have proposed architecture of generic metadata repository, which consists of following operations: (a) extract schema from XMI, (b) integrate schemas, (c) transform relational schema into star schema, and (d) store the metadata about both schema into repository using xml documents.

APPENDICES

Input/Output Screens

Login Window

This is the home page to enter into the Generic Metadata Repository. There is a login window for admin to sign in with existing account. He will enter login and password to enter into the system.

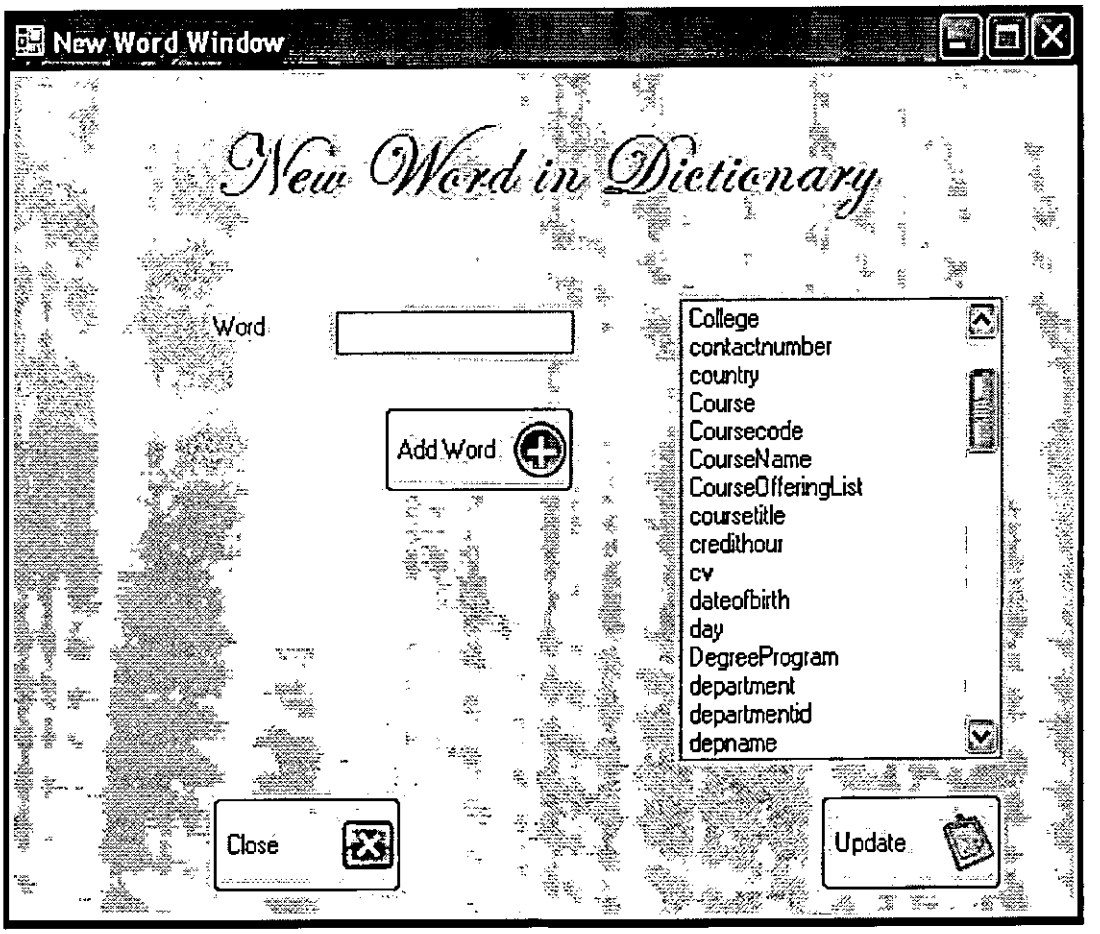


Manage GMR

After entering the login and password admin will be able to manage the generic metadata repository using the menu items. Menu items contains manage dictionary, schema integration and transformation, reports etc.

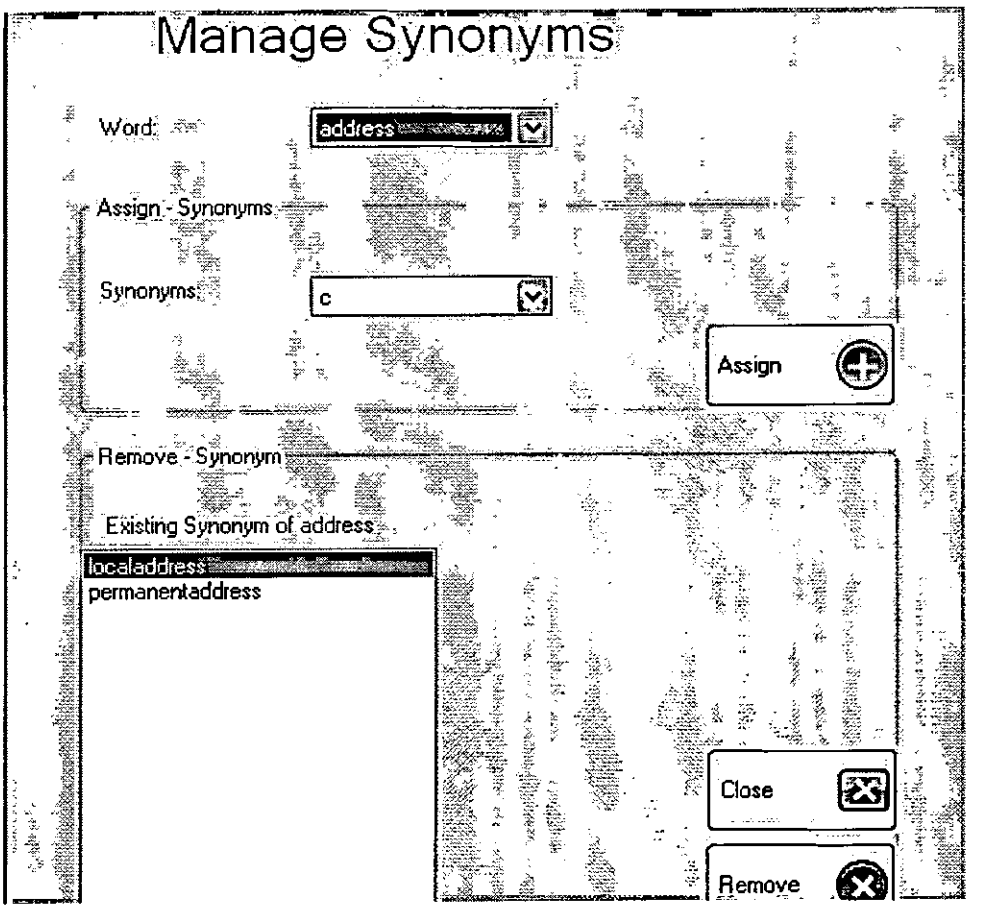
Manage Words

Admin can manage the words in the dictionary. He can add new words and also update them.



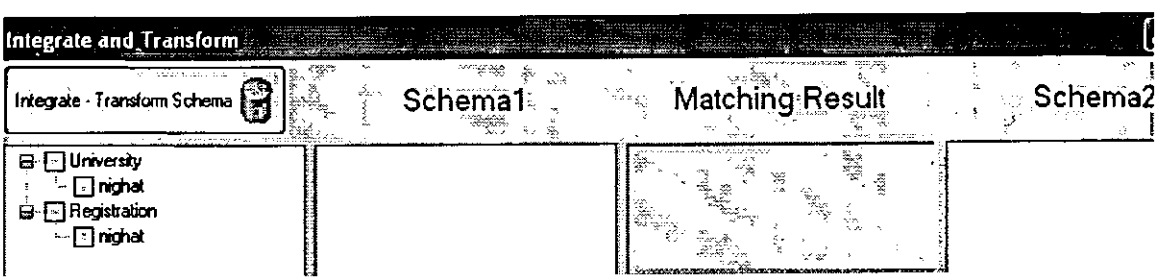
Manage Synonyms

Admin can manage the synonyms in the dictionary. He can assign synonyms to existing words and also remove them.



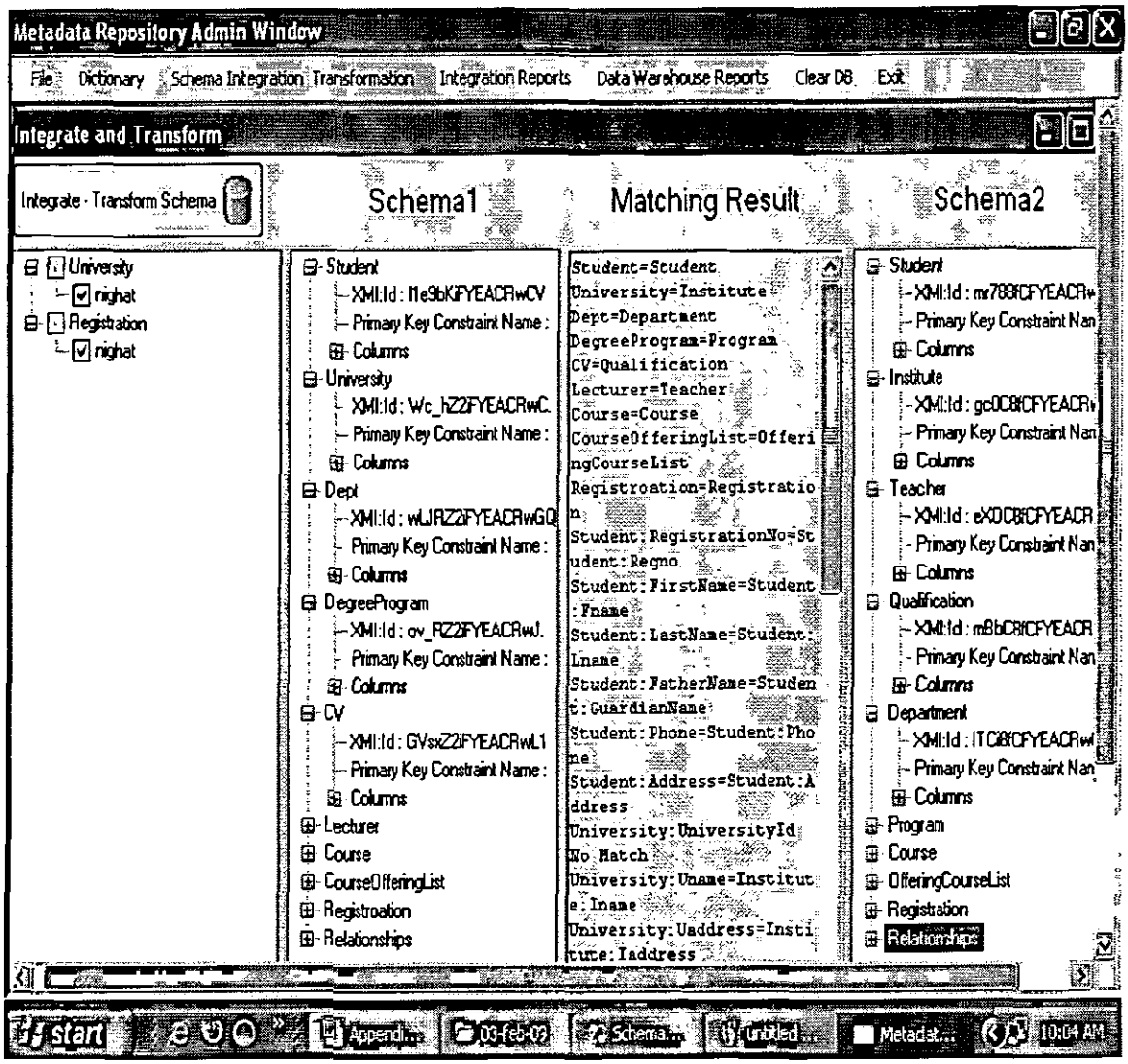
Integrate and Transform Schema

In this window, all existing schemas are displayed in left side. Admin will select any two schemas and then press the Integrate and Transform schema button. Here a message box open and asks an option to select the master schema. After selecting the schemas, integration and transformation process starts. At the end of transformation the XML documents are generated.



Matching Result

After submitting integrate and transform schema button this window is displayed. In this window both schemas are displayed in separate columns including their attributes. The result of matching is displayed in the center.



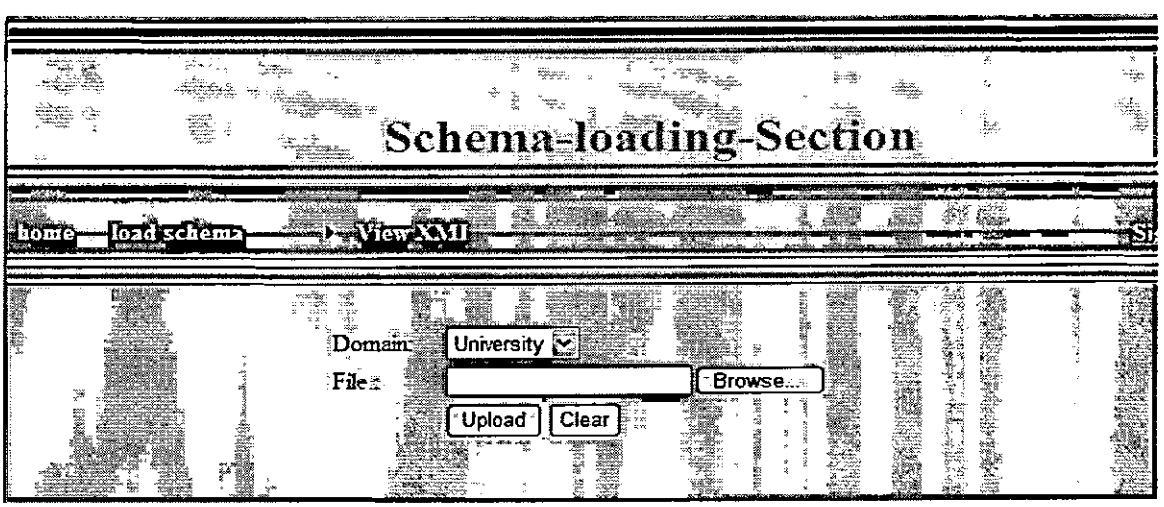
XML Documents

When the integration and transformation complete then the XML documents are generated. An example of integrated table document is given below.

```
<?xml version="1.0" standalone="yes" ?>
- <MergeTable>
  - <MergeTable>
      <Mtid>525</Mtid>
      <xid1>mr788fCFYEACRwOa</xid1>
      <xid2>l1e9bKiFYEACRwCV</xid2>
      <t1>Student</t1>
      <t2>Student</t2>
      <Iname>Student</Iname>
      <mdbid>20</mdbid>
    </MergeTable>
  - <MergeTable>
      <Mtid>526</Mtid>
      <xid1>gc0C8fCFYEACRwVo</xid1>
      <xid2>Wc_hZ2iFYEACRwC.</xid2>
      <t1>Institute</t1>
      <t2>University</t2>
      <Iname>Institute</Iname>
      <mdbid>20</mdbid>
    </MergeTable>
  - <MergeTable>
      <Mtid>527</Mtid>
      <xid1>eXOC8fCFYEACRwbN</xid1>
      <xid2>TwPxZ2iFYEACRwOm</xid2>
      <t1>Teacher</t1>
```

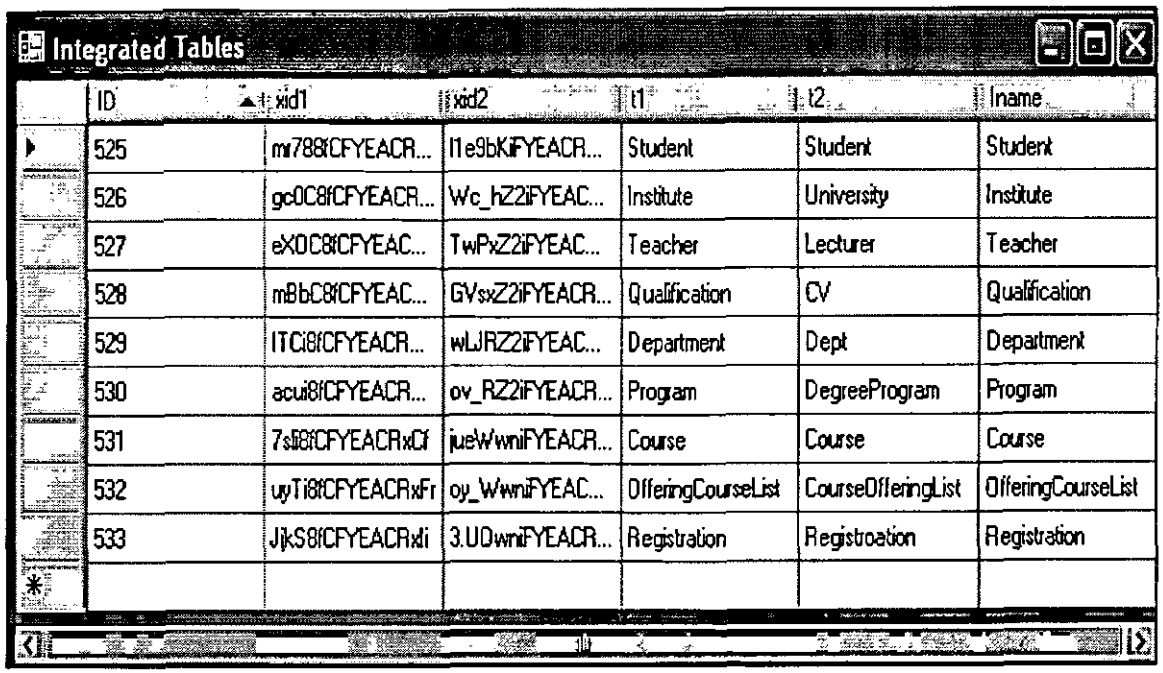
Loading Section

This page is used to load an XMI schema into the system. This schema is then used for integration and transformation.



Integrated tables

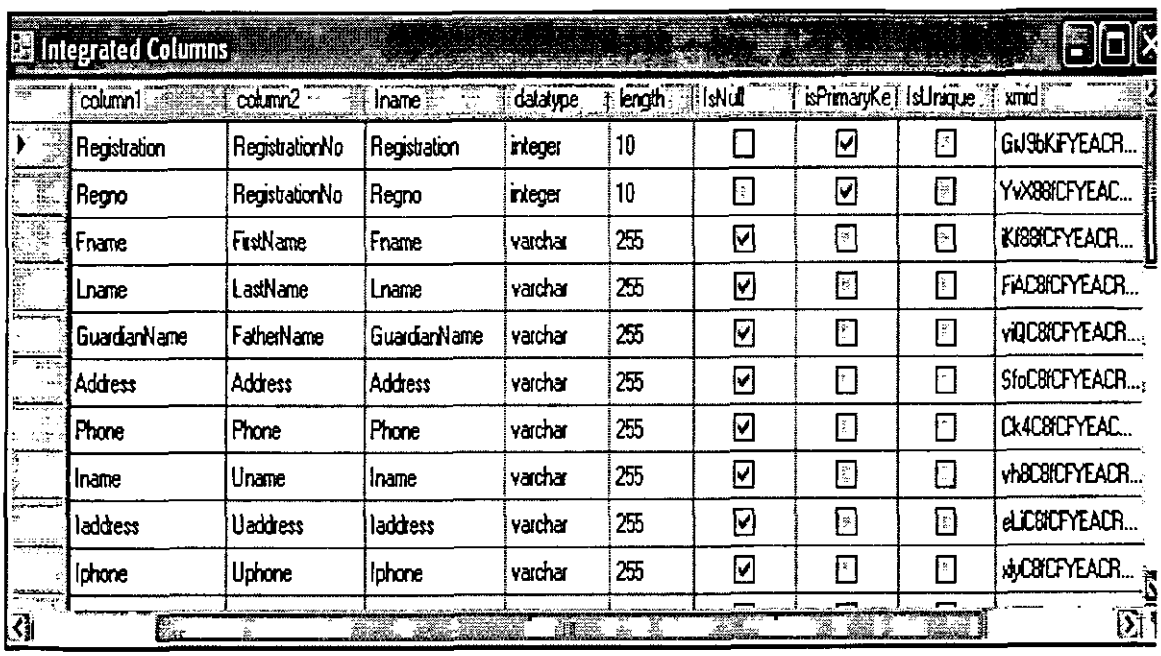
The report of integrated tables is displayed using this window.



ID	xid1	xid2	t1	t2	lname
525	mr788fCFYEACR...	l1e9bKiFYEACR...	Student	Student	Student
526	gc0C8fCFYEACR...	Wc_hZ2iFYEAC...	Institute	University	Institute
527	eX0C8fCFYEAC...	TwPxZ2iFYEAC...	Teacher	Lecturer	Teacher
528	mBbC8fCFYEAC...	GVsxZ2iFYEACR...	Qualification	CV	Qualification
529	ITCi8fCFYEACR...	wLJRZ2iFYEAC...	Department	Dept	Department
530	acui8fCFYEACR...	ov_RZ2iFYEACR...	Program	DegreeProgram	Program
531	7sl88fCFYEACRxCf	jueWwniFYEACR...	Course	Course	Course
532	uyTi8fCFYEACRxFr	oy_WwniFYEAC...	OfferingCourseList	CourseOfferingList	OfferingCourseList
533	JjkS8fCFYEACRxli	3.UDwniFYEACR...	Registration	Registroation	Registration

Integrated Columns

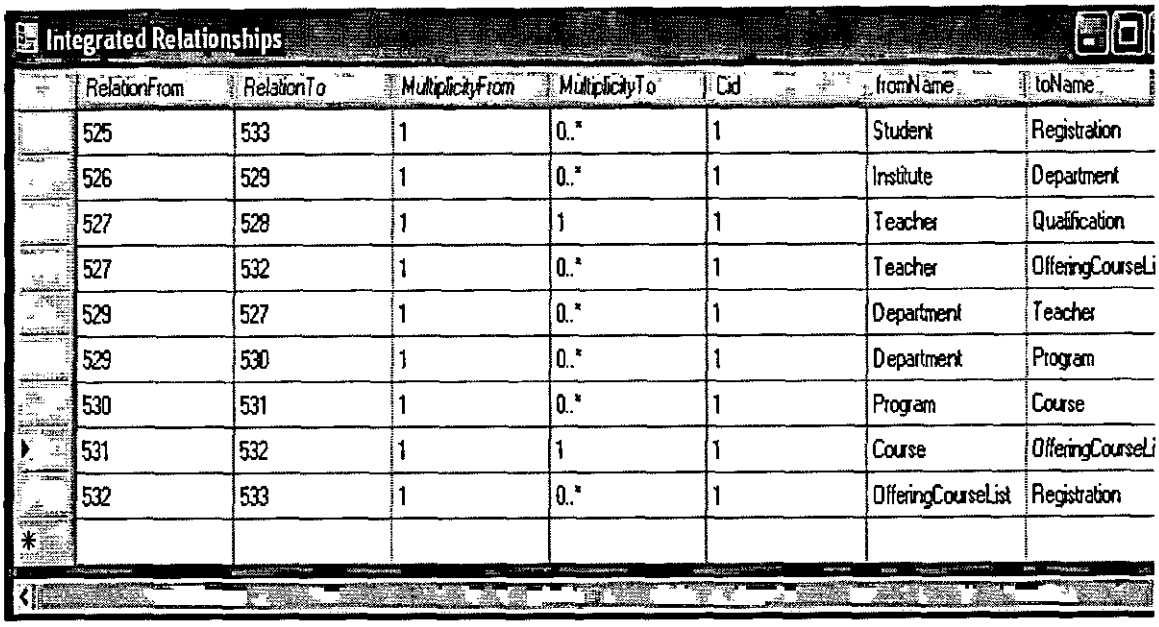
The report of integrated columns is displayed using this window.



column1	column2	lname	datatype	length	IsNull	isPrimaryKe	IsUnique	xmid
Registration	RegistrationNo	Registration	integer	10	☐	☑	☐	GiW9bKiFYEACR...
Regno	RegistrationNo	Regno	integer	10	☐	☑	☐	YvX88fCFYEAC...
Fname	FirstName	Fname	varchar	255	☑	☐	☐	iKl88fCFYEACR...
Lname	LastName	Lname	varchar	255	☑	☐	☐	FiAC8fCFYEACR...
GuardianName	FatherName	GuardianName	varchar	255	☑	☐	☐	viQC8fCFYEACR...
Address	Address	Address	varchar	255	☑	☐	☐	SfoC8fCFYEACR...
Phone	Phone	Phone	varchar	255	☑	☐	☐	Ck4C8fCFYEAC...
Iname	Uname	Iname	varchar	255	☑	☐	☐	vh8C8fCFYEACR...
Iaddress	Uaddress	Iaddress	varchar	255	☑	☐	☐	eLiC8fCFYEACR...
Iphone	Uphone	Iphone	varchar	255	☑	☐	☐	xlyC8fCFYEACR...

Integrated Relationships

The report of integrated relationships is displayed using this window.

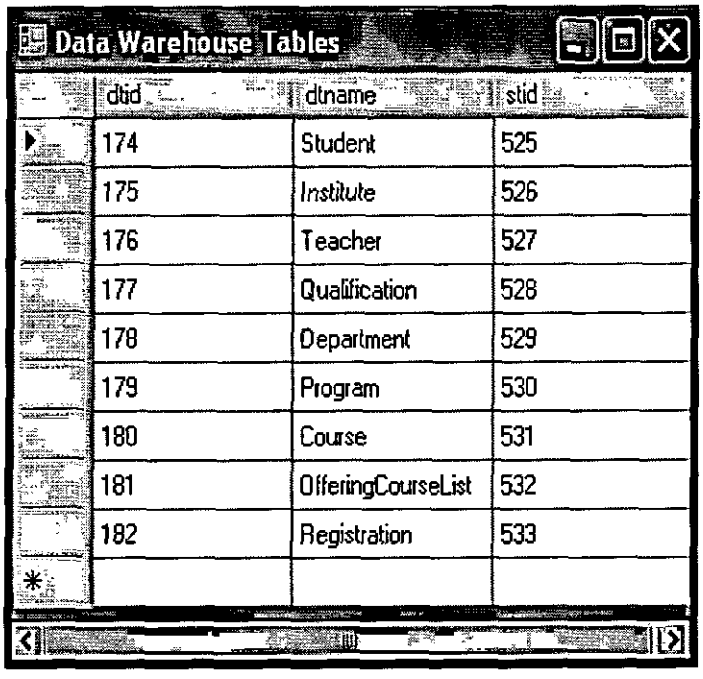


Integrated Relationships

RelationFrom	RelationTo	MultiplicityFrom	MultiplicityTo	Cid	fromName	toName
525	533	1	0..*	1	Student	Registration
526	529	1	0..*	1	Institute	Department
527	528	1	1	1	Teacher	Qualification
527	532	1	0..*	1	Teacher	OfferingCourseLi
529	527	1	0..*	1	Department	Teacher
529	530	1	0..*	1	Department	Program
530	531	1	0..*	1	Program	Course
531	532	1	1	1	Course	OfferingCourseLi
532	533	1	0..*	1	OfferingCourseList	Registration

Data Warehouse Tables

The report of data warehouse tables is displayed using this window.

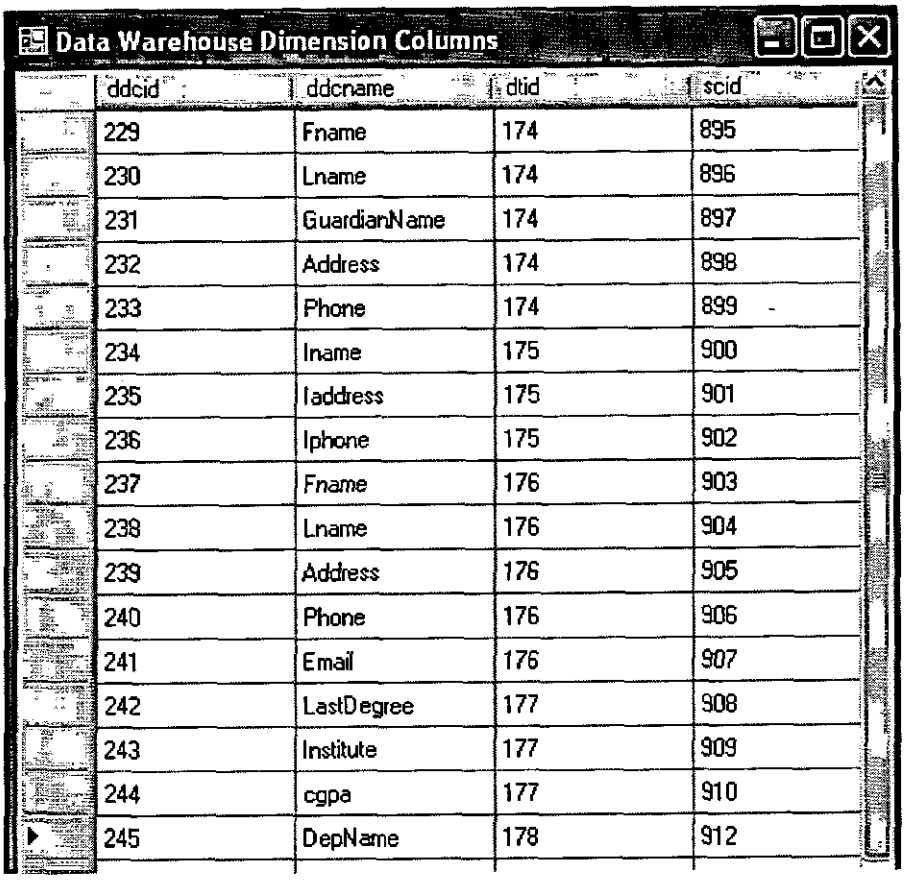


Data Warehouse Tables

dtid	dtname	stid
174	Student	525
175	Institute	526
176	Teacher	527
177	Qualification	528
178	Department	529
179	Program	530
180	Course	531
181	OfferingCourseList	532
182	Registration	533

Data Warehouse Dimension Columns

The report of data warehouse dimension columns is displayed using this window.

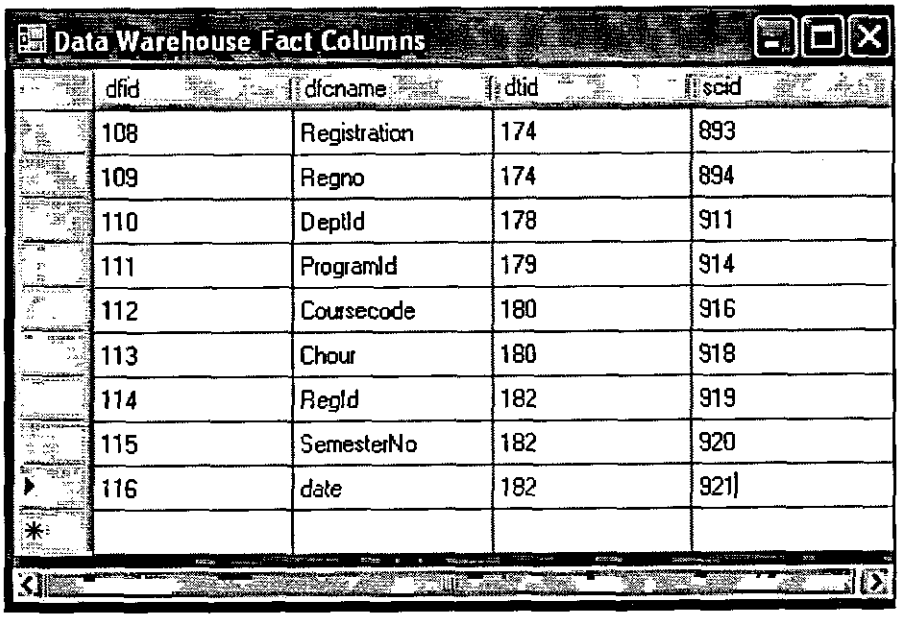


Data Warehouse Dimension Columns

ddcid	ddcname	dtid	scid
229	Fname	174	895
230	Lname	174	896
231	GuardianName	174	897
232	Address	174	898
233	Phone	174	899
234	lname	175	900
235	laddress	175	901
236	lphone	175	902
237	Fname	176	903
238	Lname	176	904
239	Address	176	905
240	Phone	176	906
241	Email	176	907
242	LastDegree	177	908
243	Institute	177	909
244	cgpa	177	910
245	DepName	178	912

Data Warehouse Fact Columns

The report of data warehouse fact columns is displayed using this window.



Data Warehouse Fact Columns

dfid	dfcname	dtid	scid
108	Registration	174	893
109	Regno	174	894
110	DeptId	178	911
111	ProgramId	179	914
112	Coursecode	180	916
113	Chour	180	918
114	RegId	182	919
115	SemesterNo	182	920
116	date	182	921

APPENDIX B

References

[1] Inmon, W.H. "*Building the Data Warehouse*". John Wiley, 1992.

[2] Paulraj Ponniah. "*Data Warehousing Fundamentals*", A comprehensive guide for IT professional. Wiley Publishers, 2001. ISBN: 0471-412546, 1997.

[3] Hong Hai Do, Erhard Rahm. "*On Metadata Interoperability in Data Warehouse*". Techn.Report 1-2000, Dept. of Information Technology, Univ. of Leipzig, March 2000.

[4] Staudt, M.; Vaduva, A.; Vetterli, T. "*The Role of Metadata for Data Warehousing*". Techn.Report 99.06., University of Zurich, Dept. of Information Technology, September 1999.

[5] David Marco, 2001. Available at: http://www.tdan.com/view-articles/4968

[6] Christoph Quix Informatik V, RWTH Aachen. "*Repository Support for Data Warehouse Evolution*". Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'99) Heidelberg, Germany, 14. - 15.6. 1999.

[7] Stefano Rizzi, Alberto Abelló, Jens Lechtenbörger, Juan Trujillo. "*Research in data warehouse modeling and design: dead or alive?*". Proceedings of the 9th ACM international workshop on Data warehousing and OLAP. DOLAP 06, ACM Press, Nov.2006

[8] Claudio Jossen, Klaus R. Dittrich. "*The Process of Metadata Modeling in Industrial Data Warehouse Environments*". BTW Workshops 2007: 16-27

[9] Jayant Madhavan, Philip A. Bernstein, Erhard Rahm. "*Generic Schema Matching with Cupid*". Proceedings of the 27th International Conference on Very Large Data Bases. ACM Press, 2001

[10] WordNet-a lexical database for English: http://wordnet.princeton.edu/.

[11] Surajit Chaudhuri, Umeshwar Dayal. "*An Overview of Data Warehousing and OLAP Technology*". ACM SIGMOD Record, March 1997, pp. 65-74.

[12] D. L. Moody, M. A. R. Kortink. "*From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design*". Proc. Of international workshop on design and management of data warehouse (DMDW'2000)

[13] KIMBALL, R. "*The Data Warehouse Toolkit*", New York: J. Wiley & Sons, 1996.

[14] P. Bernstein and T. Bergstraesser. "*Meta-data support for data transformations using Microsoft Repository*". IEEE Data Engineering Bulletin, 22(1):9-14, March 1999.

[15] Sean McGrath. "*XML by Example*", Prentice Hall PTR; May 28 1998.

[16] Efraim Turban, Jay E. Aronson, Narasimha Bolloju (2001 six edition) *Decision Support Systems and Intelligent Systems.*

[17] "Wikipedia, The Free Encyclopedia" available at http://en.wikipedia.org/wiki/Metadata#Data_warehouse_metadata

[18] Rahm, E., Bernstein P.A. "A *survey of approaches to automatic schema matching*". VLBD Journal 10: 4, 2001.

[19] Nayyer Masood, Omer Iqbal. "*Conceptual and Context based Combination of Schema Matchers*". 4th IEEE ICET (IEEE International Conference on Emerging Technologies), Oct, 2008.

[20] Batini, C., Lenzerini, M., Navathe, S. B. "*A Comparative Analysis of Methodologies for Database Schema Integration*", ACM Computing Surveys, 18(4), p(323-364), Dec., 1986

[21] E. Rahm, H.H Do. "*Data Cleaning: Problems and Current Approaches*". IEEE Tech. Bulletin on Engineering, Dec 2000.

[22] L. Bass, P. Clemens and R. Kazman. "*Software Architecture in Practice*". Addison Wesley, 2nd ed., 2003.

[23] Panos Vassiliadis. "*Data warehousing metadata*", Encyclopedia of database system, Editors-in-chief: Liu, Ling; Ozsu, M.Tamer, Springer, 2009.