بسم الله الرحمن الرحيم

Sudan University of Science & Technology

College of Graduate Studies

Data Integration using a Multi-agent System

تكامل البيانات باستخدام نظام متعدد الوكلاء

A Thesis Submitted in Partial Fulfillment of the Requirements of the Master's Degree in Computer Science

By:

Amir Abdallah Ahmed Eltyeb

Supervisor:

Dr. Amir Abdelfattah Ahmed Eisa

March 2016

الآيـــــــة

(وَقُلِ اعْمَلُوا فَسَيَرَى اللَّهُ عَمَلَكُمْ وَرَسُولُهُ وَالْمُؤْمِنُونَ وَسَتُرَدُّونَ إِلَىٰ عَالِمِ

الْغَيْبِ وَالشَّهَادَةِ فَيُنَبِّئُكُم بِمَا كُنتُمْ تَعْمَلُونَ)

(التوبة: 105)

صدق الله العظيم

DEDICATION

This thesis is dedicated

To my late father, the reason what I become today

To my dear mother, thanks for your great support and continuous care

To my sisters, my brothers, my late little brother

I am really grateful to all of you

You have been my inspiration, and my soul mates

ACKNOWLEDMENT

Firstly, I would like to express my sincere gratitude to my advisor Dr. Amir Abdelfattah A. Eisa for the continuous support of my Ms.c study: Data Integration Using Multi-agent System, and for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ms.c study.

ABSTRACT

The expansion of the communications networks, together with the increasing availability of information from different sources and the growth of data sources led to a situation in which advanced information system must be able to support transparent access to diverse data residing in different and heterogeneous sources.

A common problem that faces many organizations is that of multiple, distributed information sources and repositories including databases. Decision making process needs information from different, heterogeneous data sources. To access these data, it needs to carry out their integration into a unified format. Data integration of multiple sources requires creating some form of integrated view to allow for distributed querying.

This research, proposes a new approach for data integration that uses an approach based on multi-agent systems (MAS) technology. It considers the different tasks of the data integration process as services offered by actors called agents. To validate the approach, multiagent system for autonomous data integration between two different, heterogeneous database systems is designed and implemented. The proposed approach has proved that multiagent systems are capable of integrating data between different systems in a flexible and efficient way.

المستخلص

إن التوسع في شبكات الاتصالات، جنبا إلى جنب مع تزايد توافر المعلومات من مصادر مختلفة ونمو مصادر البيانات يتطلب ان يكون نظام المعلومات قادراً على دعم وصول فعال ومرن وشفاف لبيانات متنوعة موجودة في مصادر للبيانات متنوعة وغير متجانسة.

وهناك مشكلة مشتركة تواجه العديد من الموسسات وهي تواجد مصادر البيانات في صور متعددة وبطريقة وموزعة بما في ذلك قواعد البيانات كما هو الحال في هذا البحث. وحيث ان عمليةاتخاذ القرار تحتاج للمعلومات من مصادر للبيانات تكون متعددة، متباينة، وموزعة. ومن أجل الوصول إلى هذه البيانات، فنحن بحاجة لاجراء تكامل هذه البيانات في شكل موحد. هذا التكامل للبيانات من مصادر متعددة يتطلب ايجاد شكل يسمح باجراء استعلام علي المصادر الموزعة . حيث توجد العديد من الطرق لتكامل البيانات.

في هذا البحث، يقترح نهجا جديدا لتكامل البيانات يستخدم النهج القائم على تكنولوجيا نظام متعدد الوكلاء (MAS). وتم اعتبار المهام المختلفة لعملية تكامل البيانات علي انها عبارة عن خدمات يقدمها نظام متعدد الوكلاء. وللتحقق من صحة هذه المنهج ، تم تنفيذ النظام حيث يقوم هذا النظام باجراء عملية تكامل البيانات بطريقة ذاتية بين نظامين مختلفين، وغير متجانسين لقواعد البيانات. وتم اختبار النظام حيث وجد أن النظام متعدد الوكلاء يمكن أن يعمل علي تكامل البيانات بين الأنظمة المختلفلة بطريقة مرنة وفعالة.

Table of Contents

Dedication iii
Acknowledgment iv
Abstract v
Table of contents vii
Table of figures x
Chapter One: Introduction 1
1.2 State of the problem 1
1.3 Objective of the study 2
1.4 Research question.....2
1.5 Significance of the research 2
1.6 Research scope 3
1.7 Methodology 3
1.8 Overview 4
Chapter Two: Background and Related Work.....5
2.1 Background 5
2.2 A Brief history of data integration 5
2.3 Related work 8
2.3.1 TSIMMIS 9
2.3.2 CIIMPLEX.....10
2.3.2.1 CIIMPLEX agent system architecture 10
2.3.3 The MOMIS data integration system.....13
2.3.3.1 data integration process and MOMIS architecture 13
Chapter Three: Data Integration and Multi-agent System 20
3.1 Data integration.....20
3.2 Data integration approaches.....20
3.2.1 Manual integration 20
3.2.2 Common user interface 21
3.2.3 Common data storage method (data warehousing) 21

3.2.4 Applications 21
3.2.5 Mediator based approach 22
3.3. Agent and multi-agent system 23
3.3.1 What is an agent 23
3.3.1.1 The Wooldridge, Jennings agent 23
3.3.1.2 Intelligent agents 25
3.3.1.3 Autonomous agents 25
3.3.2 Agent software 26
3.3.3 Multiagent system (MAS) 26
3.3.4 Multiagent system and agent collaboration 27
Chapter Four: Current Systems Description 29
4.1. Student registration and result system 29
4.1.1 Student registration process 29
4.1.2 Extraction results 29
4.1.3 Classified student process 30
4.1.4 Repeat student process 30
4.2 The legacy system and new system 31
4.3 Data integration between the systems 31
4.4 Result processing block diagram 32
Chapter Five: System analysis and design 34
5.1 Analysis and design using Gaia Methodology 34
5.2 A Gaia model 36
5.3 Requirement specification 37
5.4 The analysis phase 37
5.4.1 Role model 37
5.4.2 Interaction model 42
5.5 The design phase 46
5.5.1 Agent model 46
5.6 Multi-agent system architecture 47
5.7 Implementation 48

Chapter Six: Results, Discussions, Conclusion and Recommendations 50
6.1 Results and discussion.....50
6.2 Comparison with related work.....51
6.3 Conclusion.....53
6.4 Recommendations.....54
References.....55
Appendices.....58

Table of Figures

Figure 2.1 TSIMMIS architecture 9
Figure 2.2 CIIMPLEX integration architecture 11
Figure 2.3 The data integration process 14
Figure 2.4 The MOMIS data integration process 16
Figure 4.1 Result processing block diagram 33
Figure 5.1 Models in the Gaia methodology 35
Figure 5.2 Schema for a role Connection 38
Figure 5.3 Schema for a role DataSelection 39
Figure 5.4 Schema for a role AgentCommunication 40
Figure 5.4 Schema for a role DataIntegration 41
Figure 5.5 Agent model 47
Figure 5.6 Mediator agent architecture 48

CHAPTER ONE

INTRODUCTION

The integration of heterogeneous databases and information systems is an area of high practical importance. The very success of information systems and data management technology in a short period of time has caused the virtual omnipresence of stand-alone systems that manage data – “islands of information” – that by now have grown too valuable not to be shared. However, this sharing, and with it the resolution of heterogeneity between systems, entails interesting and nontrivial problems, which have received much research interest in recent years. Ongoing research activity, however, is evidence of the fact that many questions remain unanswered [1].

1.2 Statement of the problem

A common problem faces many organizations is that of multiple, distributed information sources and repositories including databases, object stores, knowledge bases, files systems, digital libraries, information retrieval systems, …etc. Decision making process needs information from multiple, heterogeneous, distributed data sources. Therefore, information integration from heterogeneous data sources becomes increasingly important. Data integration of multiple sources requires creating some form of integrated view to allow for distributed querying. The issues for multiple data source integration are summarized as:

- The main problem is the heterogeneity among the data sources.
- The heterogeneous of data sources makes view integration and conflict resolution big challenges.
- Constructing an integrated view of different data sources is difficult because they will store different types of data, in varying formats, with different meanings, and will be referenced using different names.
- Systems storing the data can be different and communication between them not available.

1.3 Objective of the study

The main goal of the proposed research is to provide data integration between different heterogeneous data sources through multi-agent system technology.

1.4 Research question

How multi-agent system can be implemented in such away to provide data integration between different heterogeneous data sources?

1.5 Significance of the research

This research will be significant endeavor to integrate data between different information systems and different heterogeneous data sources residing in different platforms and residing many enterprises.

1.6 Research Scope

The scope of this research focuses on data integration between different heterogeneous data sources through multi-agent system technology. The multi-agent system technology makes view integration of heterogeneous data sources and constructing an integrated view of different data sources which store different types of data, in varying formats, with different meanings, and will be referenced using different names. Moreover multi-agent system allows the communication between different data sources and different systems through agent communication language (ACL).

1.7 Methodology

For this research the following approach is adopted. Firstly, brief history of data integration was reviewed in order to understand how data integration was evolved. Secondly, relevant literature, publications and studies were reviewed in order to get in-depth information on data integration using multi-agent systems. Thirdly, data integration approaches were reviewed in order to get in-depth information on data integration evolution through various approaches. Fourthly, multi-agent systems were reviewed in order to understand what multi-agent system is, and how it works. Fifthly, our current systems were described to understand how they are works. Sixthly, the proposed multi-agent system is modeled, designed and then implements. Lastly, the multi-agent system has been testing and experimented, and the results were analyzed and discussed and compared with similar work.

1.8 Overview

The remainder of this research is structured as following. The next chapter discusses a background and related work. Chapter three includes overview of data integration and multi-agent system. Chapter four describes the current systems. Chapter five illustrates the analysis and design of our agent system, the methodology of the research, and implementation. Last chapter discusses the results and includes the conclusion and recommendations.

CHAPTER TWO

BACKGROUND AND RELATED WORK

2.1 Background

Issues with combining heterogeneous data sources under a single query interface have existed for some time. The rapid adoption of databases after the 1960s naturally led to the need to share or to merge existing repositories. This merging can take place at several levels in the database architecture [2].

During the past decades the different research communities have investigated the data integration problem that lead to numerous different approaches in a way in which different data sources can be integrated [3].

2.2 A Brief History of Data Integration

Data integration involves combining data resides in different sources and providing users with a unified view of these data [4]. This process becomes significant in a variety of situations, which include both commercial (when two similar companies need to merge their databases) and scientific (combining research result from different bioinformatics repositories, for example) domains. Data integration appears with increasing frequency as the volume and the need to share existing data explodes [5]. It has become the focus of extensive theoretical work, and numerous open problems remain unsolved. In

management circles, people frequently refer to data integration as “Enterprise Information Integration” (EII) [6].

One popular solution is implemented based on data warehousing . The warehouse system extracts, transforms, and loads data from heterogeneous sources into a single view schema so data becomes compatible with each other. This approach offers a tightly coupled architecture because the data are already physically reconciled in a single queryable repository, so it usually takes little time to resolve queries. However, problems lie in the data freshness, that is, information in warehouse is not always up-to-date. Thus updating an original data source may outdate the warehouse, accordingly the extract, transform, and load (ETL) process needs re-execution for synchronization. Difficulties also arise in constructing data warehouses when one has only a query interface to summary data sources and no access to the full data. This problem frequently emerges when integrating several commercial query services like travel or classified advertisement web applications [7].

The trend in data integration has favored loosening the coupling between and providing a unified query-interface to access real time data over a mediated schema, which allows information to be retrieved directly from original databases. This approach relies on mappings between the mediated schema and the schema of original sources, and transforms a query into specialized queries to match the schema of the original databases. Such mappings can be specified in 2 ways : as a mapping from entities in the mediated

schema to entities in the original sources (the "Global As View" (GAV) approach), or as a mapping from entities in the original sources to the mediated schema (the "Local As View" (LAV) approach). The latter approach requires more sophisticated inferences to resolve a query on the mediated schema, but makes it easier to add new data sources to a (stable) mediated schema [8].

Some of the work in data integration research concerns the semantic integration problem. This problem addresses not the structuring of the architecture of the integration, but how to resolve semantic conflicts between heterogeneous data sources. For example if two companies merge their databases, certain concepts and definitions in their respective schemas like "earnings" inevitably have different meanings. In one database it may mean profits in dollars (a floating-point number), while in the other it might represent the number of sales (an integer) [8]. A common strategy for the resolution of such problems involves the use of ontologies which explicitly define schema terms and thus help to resolve semantic conflicts. This approach represents ontology-based data integration [9]. On the other hand, the problem of combining research results from different bioinformatics repositories requires bench-marking of the similarities, computed from different data sources, on a single criterion such as positive predictive value. This enables the data sources to be directly comparable and can be integrated even when the natures of experiments are distinct [10].

It was determined that current data modeling methods were importing data isolation into every data architecture in the form of islands of disparate data and information silos each of which represents a disparate system. This data isolation is an unintended artifact of the data modeling methodology that results in the development of disparate data models. Disparate data models, when instantiated as databases, form disparate databases. Enhanced data model methodologies have been developed to eliminate the data isolation artifact and to promote the development of integrated data models [11]. One enhanced data modeling method recasts data models by augmenting them with structural metadata in the form of standardized data entities. As a result of recasting multiple data models, the set of recast data models will now share one or more commonality relationships that relate the structural metadata now common to these data models. Commonality relationships are a peer-to-peer type of entity relationships that relate the standardized data entities of multiple data models. Multiple data models that contain the same standard data entity may participate in the same commonality relationship. When integrated data models are instantiated as databases and are properly populated from a common set of master data, then these databases are integrated [12].

2.3 Related work

Derived from the data engineering community several solutions have been proposed that based on a mediator architecture where logical database schemas are used

as shared mediated views over the queried schemas. A number of systems have been proposed e.g.

2.3.1 The Stanford-IBM Manager of Multiple Information Sources (TSIMMIS)

The Stanford-IBM Manager of Multiple Information Sources is project that developed some tools for facilitating the rapid integration of heterogeneous information sources that may include both structured and semi-structured data. TSIMMIS is based on mediator-wrapper architecture.

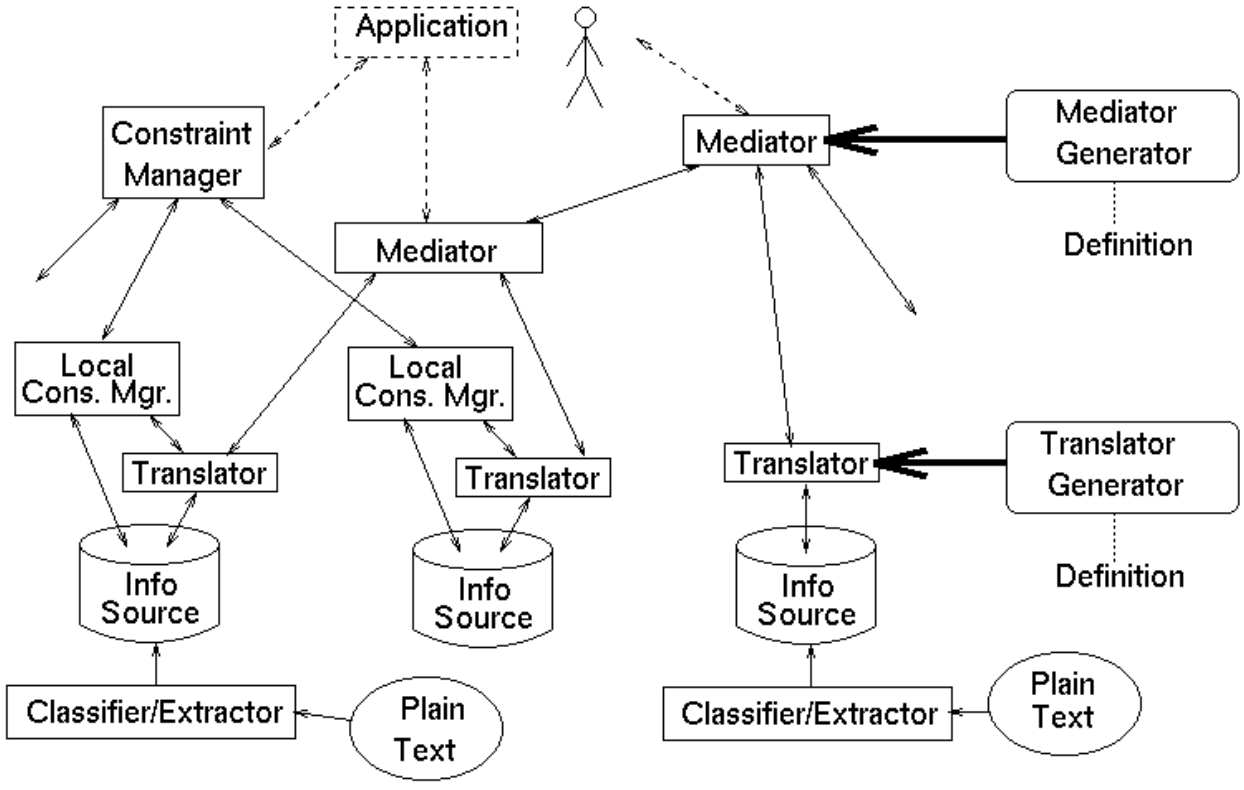


Figure 2.1 TSIMMIS Architecture [13]

Figure 2.1 shows a collection of (disk-shaped) heterogeneous information sources. Above each source is a translator (or wrapper) that logically converts the underlying data objects to a common information model. To do this logical translation, the translator converts queries over information in the common model into requests that the source can execute, and it converts the data returned by the source into the common model.

For the TSIMMIS project they have adopted a simple self-describing (or tagged) object model in which objects have labels, types, values, and an (optional) identifier. Similar models have been in use for years; they call the version the Object Ex-change Model, or OEM. OEM allows simple nesting of objects. All objects and their sub-objects have labels that describe their meaning. [13]

2.3.2 Integrating Manufacturing Software for Intelligent Planning-Execution: A CIIMPLEX Perspective

CIIMPLEX adopts as one of its key technologies the approach of intelligent software agents, a rapidly emerging technology of software systems in networked distributed environment.

2.3.2.1 CIIPLEX Agent System Architecture

In the CIIMPLEX project, the agent system architecture supports inter-agent cooperation with the emphasis on the agent communication infrastructure. Figure 2.2 below gives the

Architecture of the CIIMPLEX enterprise integration with multi-agent system as an integral part.

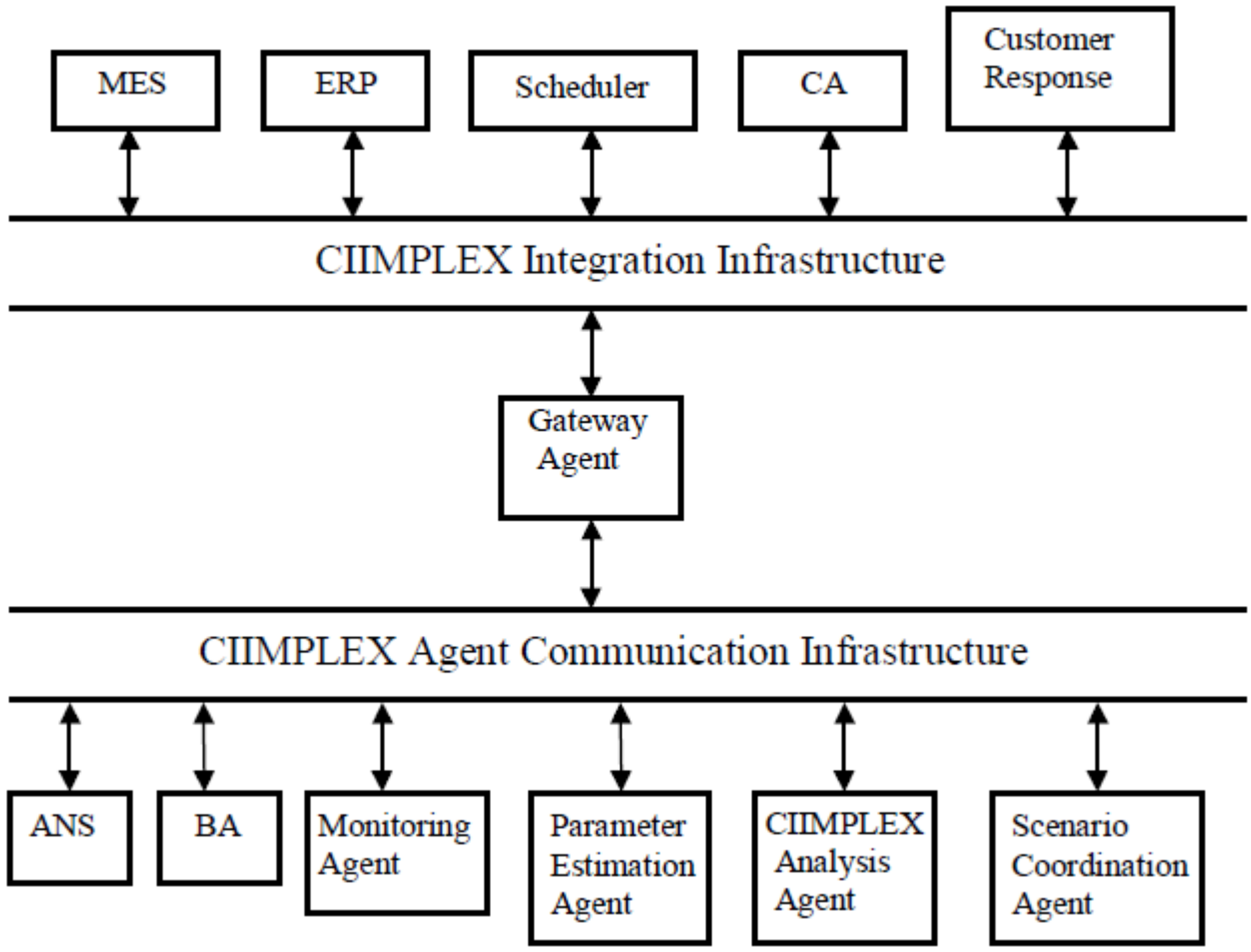


Figure 2.2 CIIMPLEX Integration Architecture [14]

The CIIMPLEX Integration Infrastructure supports communication between legacy application systems. The CIIMPLEX Agent Communication Infrastructure supports

communication between agents. The main reason to separate the two infrastructures is to reduce the necessary modifications to the applications to the minimum.

Besides the server name agent (ANS) and the broker agent (BA), several other types of agents are useful for enterprise integration. For example, data-mining/parameter-estimation agents are needed to collect, aggregate, interpolate and extrapolate the raw transaction data of the low level (shop floor) activities, and to make this aggregated information available for higher level analyses by other agents. Event monitoring agents monitor, detect, and notify about abnormal events that need to be attended. The CIIMPLEX Analysis Agents (CAA) evaluates disturbances to the current planned schedule and recommends appropriate actions to address each disturbance. And the Scenario Coordination Agents (SCA) assist human decision making for specific business scenarios by providing the relevant context, including filtered information, actions, as well as workflow charts. All these agents speak KQML, and use SKIF, a subset of KIF that supports Horn clause deductive inference, as the content language. TCP/IP is chosen as the low-level transport mechanism for agent to agent communication. The shared ontology is an agreement document established by the application vendors and users and other partners in the consortium. The agreement adopts the format of the Business Object Document (BOD) defined by the Open Application Group (OAG). BOD is also used as the message format for communication between applications such as MES and ERP, and between agents and applications. Different transport mechanisms (e.g., MQ Series of

IBM and VisualFlow of Envisionit) are under experimentation for communication to and from applications. A special service agent, called the Gateway Agent (GA), is created to provide interface between the two infrastructures. GA's functions, among other things, include making connections between the two transport mechanisms (TCP/IP and MQ Series) and transforming messages between the two different formats (KQML and BOD).

The agent system architecture outlined above is supported by the agent communication infrastructure called Jackal. As indicated by the name, JACKAL is written in Java to support Agent Communication using the KQML Agent communication Language. The decision to select Java as the implementation language was based mainly on its inter-platform portability, its networking facilities, and its support for multi-thread programming. Except for the ability to understand and process KQML messages, Jackal does not impose any restrictions on the internal architecture and representation of individual agents [14].

2.3.3 Mediator envirOnment for Multiple Information Sources (The MOMIS Data Integration System)

The MOMIS Data Integration System is a framework able to integrate data coming from heterogeneous and distributed data sources (structured and semi-structured) in a semi-automatic way, to bring out new information from apparently unrelated existing data. An object-oriented language, with an underlying Description Logic, called ODLI3,

derived from the standard ODMG is introduced for information extraction. A software component, named wrapper (Fig. 2.3) extracts the schema of a source, translating it in the ODLI3 language. The discovery of relationships between the schemas of the information sources exploits the semantics of the data sources, clustering techniques and description logics inferences. The integration process gives rise to a mediated schema, also called Global Schema (GS) (Fig. 2.3) that is a reconciled, integrated, and virtual view of the underlying sources. MOMIS performs data integration following a virtual approach; no centralized copy of data is made and follows a global-as-view (GAV) approach, the obtained global schema is expressed in terms of the local source schemas [15].

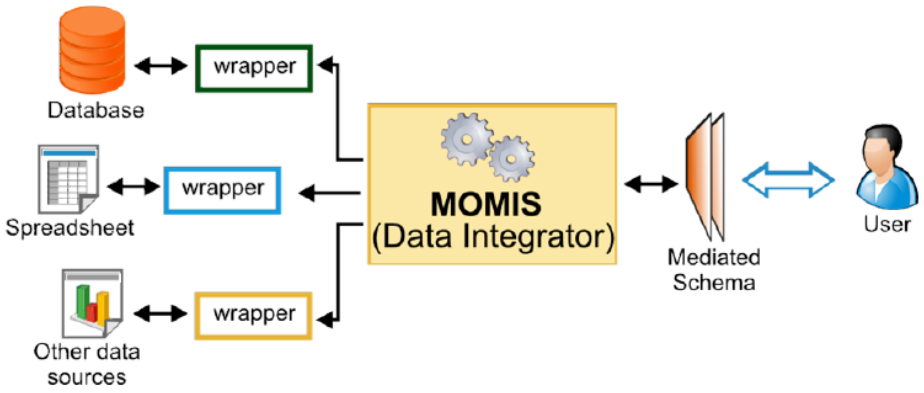


Figure 2.3: The data integration process [15]

The goal of the MOMIS system is the minimization of the integration process costs. In traditional Data Integration Systems, designers have to manually build the integrated schema, defining all the mappings between each global class/attribute and the

corresponding local classes/attributes on the local data sources, thus the integration process requires several days or weeks depending on the size of the integration project. Another drawback is due to the fact that designers can see the global result of the integration only at the end of the overall integration process, and it is only at that time that they can refine mappings in order to improve the integrated schema. To overcome these problems, a first result of integration is semi-automatically derived by MOMIS and proposed to the designer in few minutes; she/he can then improve this integration result, through an iterative refinement process and a set of features. Some of these features are listed below:

- a GUI that facilitates the integration process
- a set of explore and preview tools that allow the designer to preview the integration result during each phase
- the possibility to create different unified views to explore the global result of the data integration process
- a suite of tools to semantically annotate data sources w.r.t. a common lexical reference; these tools allow the designer to import/export the local source annotations, and permit to extend the lexical reference itself with domain glossaries

- a preview of the query plan that allows the designer to visualize, for each executed global query, the set of queries that compose the query plan [15].

2.3.3.1 Data Integration Process and MOMIS Architecture

MOMIS builds a unified schema, called Global Schema (GS), of several (heterogeneous) data sources (also called local sources), and allows users to formulate queries on it. As reported in the previous section it follows a Global- As-View (GAV) approach for the definition of mappings between the GS and local schemas: the GS is expressed in terms of the local schemas. The GS generation process is composed by four main phases:

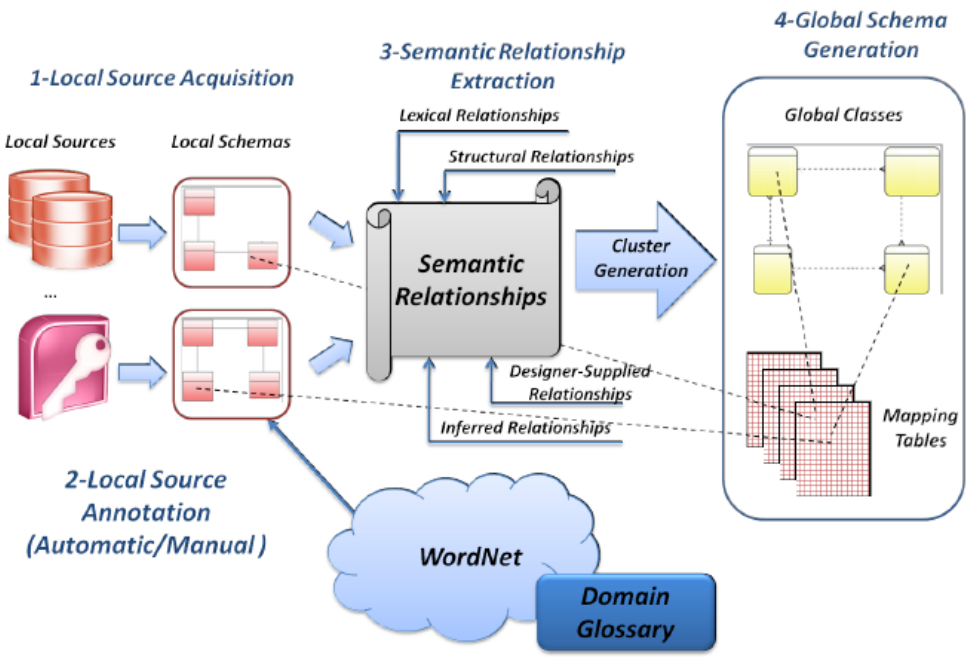


Figure 2.4: The MOMIS data integration process [15]

1. Local Schema Acquisition: (Fig. 2.4-1) the extraction of the Local Source Schema is performed by wrappers that logically extract the schema of each local source and convert it into the common language ODLI3.

2. Local Source Annotation: (Fig. 2.4-2) the designer is asked to *annotate* the local sources, i.e. to associate to each class and attribute name one or more meanings w.r.t. a common lexical reference, that in this case is the lexical database WordNet. WordNet is a thesaurus for the English language, that groups terms (called lemmas in the WordNet terminology) into sets of synonyms called synsets, provides short definitions (called gloss), and connects the synsets through a wide network of semantic relationships.

3. Semantic Relationships Extraction: (Fig. 2.4-3) starting from the annotated local schemas, MOMIS derives a set of intra and interschema semantic relationships in the form of: synonyms (SYN), broader terms/narrower terms (BT/NT) and related terms (RT) relationships. The set of semantic relationships is incrementally built by adding: structural relationships (deriving from the structure of each schema), lexical relationships (deriving from the element annotations, by exploiting the WordNet semantic network), designer-supplied relationships (representing specific domain knowledge) and inferred relationships (deriving from Description Logics equivalence and subsumption computation). The Semantic Relationship Extraction phase is performed by the Global Schema Designer tool.

4. GS generation: starting from the discovered semantic relationships and the local sources schemas, MOMIS generates a GS consisting of a set of global classes, plus mapping Tables which contain the mappings to connect the global attributes of each global class with the local source attributes. The GS generation is a process where classes describing the same or semantically related concepts in different sources are identified and clusterized into the same global class (Fig. 2.4-4). The designer may interactively refine and complete the proposed integration result through the GUI provided by the Global Schema Designer tool. In particular, she/he can: modify the proposed global classes and mappings; select the appropriate Join Function for each global class; define Transformation Functions in order to transform the local attribute values into the corresponding global attribute values; and solve possible data conflicts through the definition of resolution Functions (applied to each global attribute to obtain, starting from the values computed by the Transformation Functions the corresponding value of the global attribute).

Finally, once obtained the desired integration result, a user can pose queries on the GS by using the Query Manager tool. As MOMIS follows a GAV approach, the query processing is performed by means of query unfolding. The query unfolding process generates for each global query (i.e. a query on the GS) a Query Plan composed by a set of queries:

- a set of local queries that have to be executed on the local sources simultaneously by means of wrappers,
- a mapping query for merging the partial results (defined by means of the join function),
- A final query to apply the resolution functions and residual clauses.

Moreover, MOMIS provides the Query Manager Web Service which allows to integrate MOMIS with other applications (e.g. Business Intelligence solutions), and a user-friendly Web Application to guide an enduser, without experience on data integration solutions, to easily compose and execute query on the integrated schema [15].

CHAPTER THREE

DATA INTEGRATION AND MULTI-AGENT SYSTEM

3.1 Data integration

Data integration involves combining data residing in different sources and providing users with a unified view of these data. This process becomes significant in a variety of situations. The complexity and the proliferation of these data raise the problem of their integration. Several approaches have been proposed to data integration [16].

3.2 Data Integration Approaches

The goal of data integration is to gather data from different sources, combine it and present it in such a way that it appears to be a unified whole. Several approaches have been proposed to data integration.

3.2.1 Manual Integration

The manual integration approach would leave all the work to the user. The user has to know where to look for the data. He would need to know the physical location for both the traffic report and the map for your town. You would need to retrieve the traffic report

and the map data directly from their respective databases, and then compare the two sets of data against each other to figure out what's the best route out of town.

3.2.2 Common User Interface

If we used a **common user interface** approach, we would have to do a little less work. We would use an interface such as to make a query. The query results would appear as a view on the interface. We would still have to compare the traffic report to the map to determine the best route, but at least the interface would take care of locating and retrieving the data.

3.2.3 Common Data Storage Method (Data Warehousing)

There's the common data storage method, also known as data warehousing. Using this method, all the data from the various databases we intend to integrate are **extracted**, **transformed** and **loaded**. That means that the data warehouse first pulls all the data from the various data sources. Then, the data warehouse converts all the data into a common format so that one set of data is compatible with another. Then it loads this new data into its own database. When we submit the query, the data warehouse locates the data, retrieves it and presents it in an integrated view.

3.2.4 Applications

Some integration approaches based on an application to do all the work. The applications, which are specialized computer programs, would locate, retrieve and integrate the

information. During the integration process, the applications must manipulate the data so that the information from one source is compatible with the information from the other source. In our example, that would mean we submit a query to an application and it would present a view that combined a map of the town with data from traffic reports. The problem with this approach is that applications become complex and difficult to program as the number of data sources and formats increase.

3.2.5 Mediators Based Approach

In the mediators based approach, the proposed solution maintains the data in their source and builds abstract views from which a mediator tries to satisfy the user's queries. The current architecture of this approach is based on a mediators-wrappers system. It allows the query of distributed and heterogeneous data sources. It operates like a centralized and homogeneous system. The mediator performs the data integration by providing the user a homogeneous and global view of the system. Its task is to reformulate the user's queries according to the various contents of the accessible data sources. Several wrappers correspond to this mediator, one for each data source. Their role is to extract the data selected by the query according to the type of the corresponding data source. The major interest of this system is the query reformulation performed by the mediator, provides the user some flexibility for query writing and formulation. Furthermore, query approximation is integrated into the system to help the mediator to refine the user's query as well as the user always obtains an answer [17].

3.3 Agents and Multi-agent system

An agent is a computer system that is capable of independent (autonomous) action on behalf of its user or owner (figuring out what needs to be done to satisfy design objectives, rather than constantly being told) [18].

The main point about agents is they are autonomous: capable independent action. Thus: An agent is a computer system capable of autonomous action in some environment, in order to achieve its delegated goals.

3.3.1 What is an agent?

Workers involved in agent research have offered a variety of definitions, each hoping to explicate his or her use of the word "agent." These definitions range from the simple to the lengthy and demanding. Each of them grew directly out of the set of examples of agents that the definer had in mind. (This is certainly the case for the version that is proposed below.) The following paragraph exams and compare some of these definitions.

3.3.1.1 The Wooldridge, Jennings Agent

An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives.

A hardware or (more usually) software-based computer system that enjoys the following properties:

- autonomy: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;
- social ability: agents interact with other agents (and possibly humans) via some kind of agent-communication language;
- reactivity: agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the INTERNET, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it;
- pro-activeness: agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by taking the initiative."

The Wooldridge and Jennings definition, in addition to spelling out autonomy, sensing and acting, allows for a broad, but finite, range of environments. They further add a communications requirement. What would be the status of a payroll program with a graphical interface and a decidedly primitive communication language? [19][20].

3.3.1.2 Intelligent agents

The IBM Agent "Intelligent agents are software entities that carry out some set of operations on behalf of a user or another program with some degree of independence or

autonomy, and in so doing, employ some knowledge or representation of the user's goals or desires" [21].

This definition, from IBM's Intelligent Agent Strategy white paper, views an intelligent agent as acting for another, with authority granted by the other. A typical example might be an information gathering agent, though the white paper talks of eight possible applications.

Such agent is a system situated in, and part of, a technical or natural environment, which senses any or some status of that environment, and acts on it, over time, in pursuit of its own agenda. Such agenda evolves from drives (or programmed goals). The agent acts to change part of the environment or of its status and influences what it sensed.

Non-biological examples include intelligent agents, autonomous robots, and various software agents, including artificial life agents, and many computer viruses. Biological examples are not yet defined [22].

3.3.1.3 Autonomous agents

The Maes Agent "Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed."

An autonomous agent is an intelligent agent operating on an owner's behalf but without any interference of that ownership entity [23].

3.3.2 Agent Software

Agent software is a classical program that is qualified as "intelligent”. Intelligent agents are used in many fields such as the networks, on-board technologies (LEAP: Lightweight Extensible Agent Platform) or human learning. An intelligent agent is supposed to have the following intrinsic characteristics: intuitive - it must be ready to take initiatives and to achieve the actions that are assigned to him; reactive - it must listen to the actions of its environment and acts in consequence; sociable - it must be able to communicate with other agents and/or users.

Moreover, agents may be mobile and can independently move through an acceptor network in order to perform various tasks [24].

3.3.3 Multi-agent system (M.A.S)

Multi-agent system is a computerized system composed of multiple interacting intelligent agents within an environment.

A multi-agent system is one that consists of a number of agents, which interact with one-another. In the most general case, agents will be acting on behalf of users with different

goals and motivations. To successfully interact, they will require the ability to cooperate, coordinate, and negotiate with each other, much as people do.

A Multi-Agent System designates a collection of agents, which communicate with each other. The purpose of this collection is to perform a specific task. Each agent is then able to offer specific services in an autonomous way and communicates the results to a receiving agent (human or software) and has a well defined goal. The MAS must respect the programming standards defined by the FIPA (Foundation for Intelligent Physical Agents) [19].

3.3.4 Multi-Agent System and Agent Collaboration

The computing paradigm of multi-agent systems (MAS) has its origin in both distributed artificial intelligence (DAI) and object-oriented distributed systems. There is no consensus on the definition of software agents or of agency, and some people go so far as to suggest that any piece of software or object that can perform a specific given task is an agent. However, the prevailing opinion is that an agent may exhibit three important general characteristics: autonomy, adaptation, and cooperation. By “autonomy” we mean that agents have their own agenda of goals and exhibit goal-directed behavior. They are not simply reactive, but can be pro-active and take initiatives as they deem appropriate. In this sense, agent systems can be viewed as a generalization of the client-server model in that each agent can be both a client and a server and can provide and request services to

and from others. Adaptation implies that agents are capable of adapting to the environment, which includes other agents and human users, and can learn from the experience in order to improve themselves in a changing environment. Cooperation and coordination between agents is probably the most important feature of multi-agent systems. Unlike those stand-alone agents, agents in a multi-agent system collaborate with each other to achieve common goals. In other words, these agents share information, knowledge, and tasks among themselves. The intelligence of MAS is not only reflected by the expertise of individual agents but also exhibited by the emerged collective behavior beyond individual agents. From software engineering point of view, the approach of MAS is also proven to be an effective way to develop large distributed systems. Since agents are relatively independent pieces of software interacting with each other only through message-based interagent communication, system development, integration, and maintenance become easier and less cost. For instance, it is easy to add new agents into the agent system when needed. Also, the modification of legacy applications can be kept minimum when they are to be brought into the system. Aside from adding communication capabilities to a legacy application, nothing else is required to change [19].

CHAPTER FOUR

CURRENT SYSTEMS DESCRIPTION

4.1 Student registration and result system

Our current systems were two database systems, the legacy system is using oracle database and oracle developer interfaces, and the new system is using mysql database and web application interfaces.

System operations include student registration process, and the process of extracting the students' results.

4.1.1 Student registration process

Student registration process starts by entering the basic data for students like University ID, full name, date of birth etc ..., and the registration data including specialization, semester, courses, and the last result obtained by the student ... etc.

4.1.2 Extraction Results

Extraction results process start by entering students' grades in every course for the semester on each individual student, which the system calculates the score for each student individually and in accordance with the academic regulation, and through the procedures and formulas that have been programmed in the system.

4.1.3 Classifieds student according to results

Classifieds student process is done by the system, in accordance with current point average (CPA) scored by student, and the degree scored in any individual course. According to the academic regulation every student must score 50 degree or more in any course to pass the course, further more the student must score 2.50 (CPA) or more to pass the current semester. Every student scored less than 2.00 (CPA) in current semester must repeat the semester with all courses. Every student score less than 50 degree in any course must seat to sub examination in the course. After the sub examination results the student who scored less than 50 degree in any course in the sub exam and scored 2.50 (CPA) or more can pass to next semester but still carry the courses which he failed in.

The student who scored less than 2.50 (CPA) and failed in any course after the sub exam must repeat the semester with the course he failed in.

4.1.4 Repeat student process

The student who resulted in repeat the semester must attend with previous batch of students. This process can be done by the system automatically after setting the result every semester. The system calculates the grade point average (GPA) for the student in any semester and added to the previous (GAP) in Cumulative manner.

4.2 The legacy system and new system

The processes mentioned previously are applied in the both systems, the legacy system and the new system individually. And whereas the college is need to upgrade the legacy system to the new system must be integrate the data for repeat students in the last batch of students in the legacy system to the first batch of students in the new system.

4.3 Data integration between the systems

The process of integration data between the two systems depends on a certain condition. This condition when we must migrate the repeat students in the last batch in the legacy system to the first batch in the new system after preparing the results of the students after the exams in the end of every semester, until we migrates all that certain students to the new system.

The process of this integration can be illustrated as following:

First it must select the basic data for the repeat students from the history data which can be found in the table of the personal data of the students in the legacy system, then it must select the accumulative records of the results for the students from the table of registration, which can be the aggregates of the credit hours in the courses and aggregates of the points. Second it must create new records for the students in the new system and insert the previous data in the specific records. This process can be done manually by the user, which it attempt to automate this process by our multi-agent system it implement.

4.4 Result Processing Block Diagram

Figure 4.1 shows the process of the result in the current system. Firstly the student must register for the semester, if the student does not register for the semester he will dismiss. Then the student must attend all the courses, if the student does not attend 25% of the course he will be repeat the course. Then the student must pass the course work, and if he does not pass the course work also he will be repeat the course again. At the end of the semester the student must attend all the final exams and must scores at least 50 degrees or more in any course, and if the student does not attend one of the exams or he scores less than 50 degree in any course he must attend substitute exam.

After finishing all exams it must calculate the CPA for every student, and if the student passes all the exams he passes to the next semester, and if he failed in nay course and scores more than 2.00 he must attend substitute exam, and if he scores less than 2.00 and more than 1.50 he must repeat the semester, and if he scores less than 1.50 he will be dismiss.

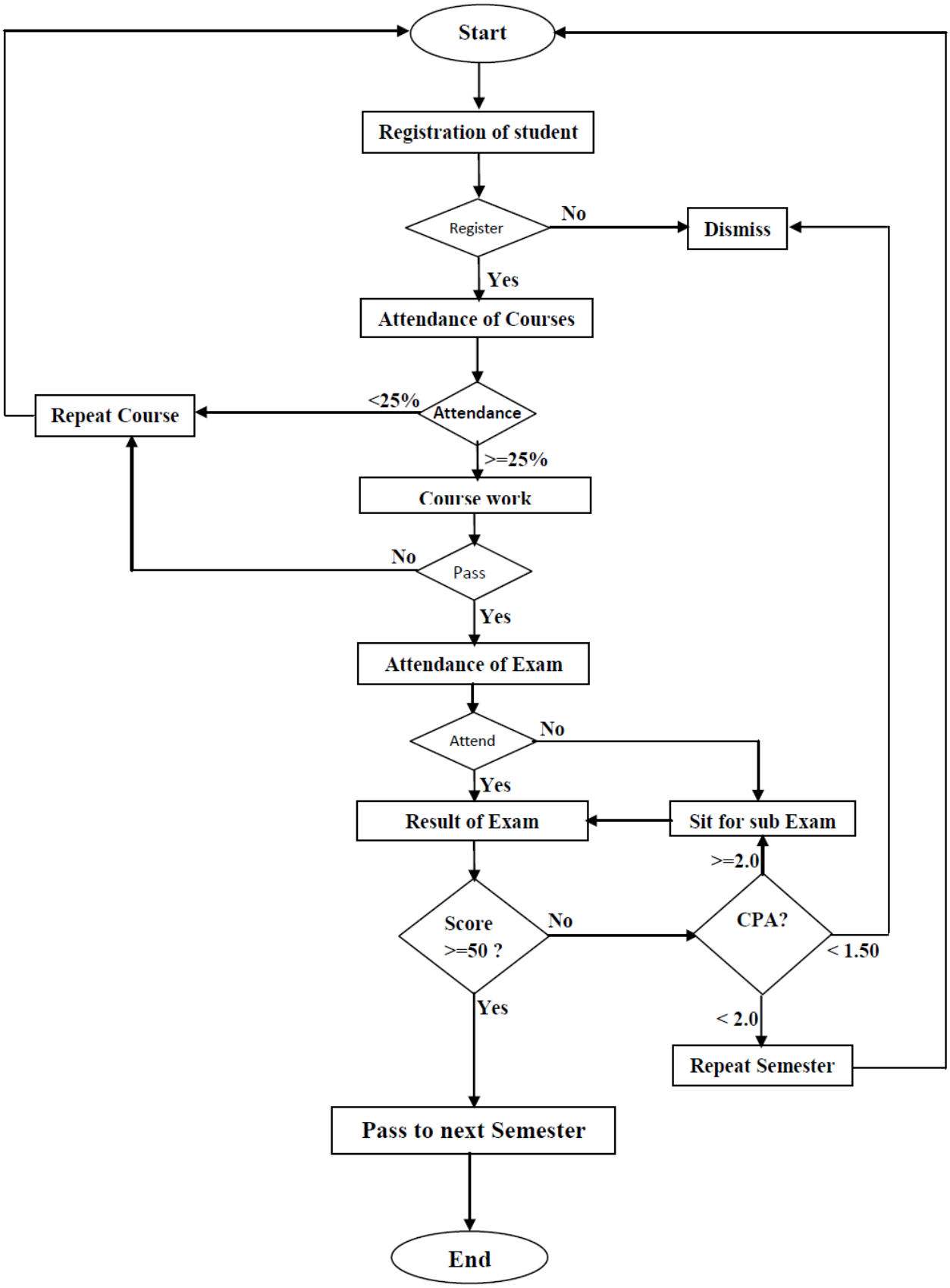


Figure 4.1 Result Processing Block Diagram

CHAPTER FIVE

SYSTEM ANALYSIS AND DESIGN

5.1 Analysis and Design using Gaia Methodology

Gaia is an attempt to define a complete and general methodology that it is specifically tailored to the analysis and design of multi-agent systems. Gaia is a general methodology that supports both the levels of the individual agent structure and the agent society in the MAS development process. MASs, according to Gaia, are viewed as being composed of a number of autonomous interactive agents that live in an organized society in which each agent plays one or more specific roles. Gaia defines the structure of MAS in terms of a role model. The model identifies the roles that agents have to play within the MAS and the interaction protocols between the different roles.

In general, the Gaia process consists in orderly constructing a series of models, as shown in Figure 5.1, aimed at describing both the macro (societal) aspects and the micro (intra-agent) aspects of a multi-agent system, generally conceived as an organized society of individuals (i.e., a computational organization of autonomous entities). In the analysis phase, the role model and the interaction model are constructed, to depict the system as a set of interacting abstract roles. These two models are then used as input to the design stage, in which an agent model, a services model, and an acquaintance model are dened

to form a complete design specification of the multi-agent system to be used for the subsequent implementation phase.

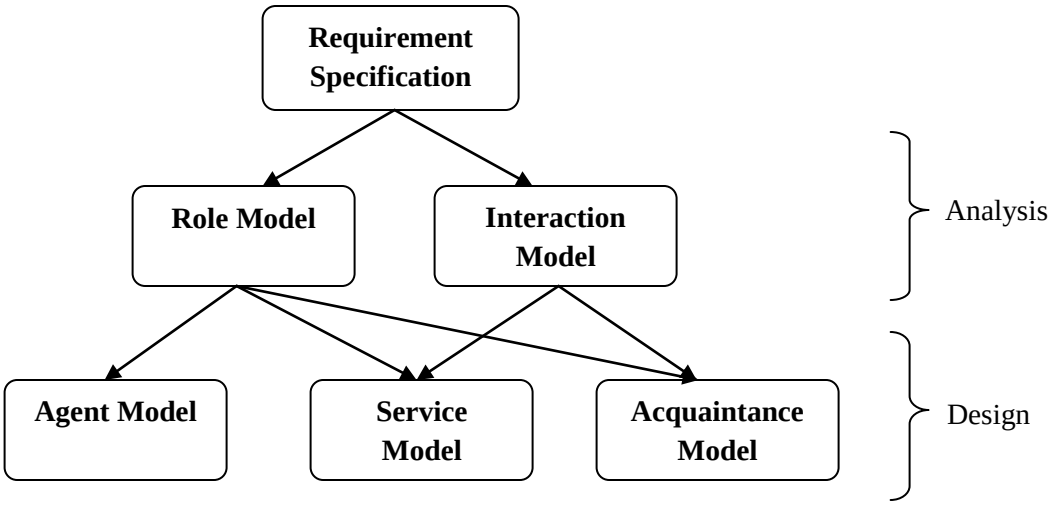


Figure 5.1 Models in the Gaia methodology [25]

The objective of the Gaia analysis process is the identification of the roles and the modeling of interactions between the roles found. Roles consist of four attributes: responsibilities, permissions, activities and protocols. Responsibilities are the key attribute related to a role since they determine the functionality. Responsibilities are of two types: liveness properties – the role has to add something good to the system, and safety properties – the role must prevent and disallow that something bad happens to the system. Liveness describes the tasks that an agent must fulfill given certain environmental conditions and safety ensures that an acceptable state of affairs is maintained during the execution cycle. In order to realize responsibilities, a role has a set of permissions. Permissions represent what the role is allowed to do and in particular,

which information resources it is allowed to access. The activities are tasks that an agent performs without interacting with other agents. Finally, protocols are the specific patterns of interaction. Gaia has formal operators and templates for representing roles and their attributes and also it has schemas that can be used for the representation of interactions between the various roles in a system.

In the Gaia design process the first step is to map roles into agent types and to create the right number of agent instances of each type. An agent type can be an aggregation of one or more agent roles. The second step is to determine the services model needed to fulfill a role in one or several agents. A service can be viewed as a function of the agent and can be derived from the list of protocols, activities, responsibilities and the liveness properties of a role. Finally, the last step is to create the acquaintance model for the representation of communication between the different agents. The acquaintance model does not define the actual messages that are exchanged between the agents it is rather a simple graph that represents the communication pathways between the different agent types [19] [25].

5.2 A Gaia Model

In order to better understand how Gaia conceive and implement a multi-agent system (MAS) we will present how our system can be analyzed, designed and implemented.

5.3 Requirement Specification

For this system we have defined the following requirements:

- The integration process needs multi-agent system consist of three agents.
- The first agent establishing connection and communicate to the legacy system and to retrieve the data from the legacy data source according to certain query.
- The second agent establishing connection and communicate to the new system to integrate the data to the new data source according to certain query.
- The third agent can be mediator agent between the two agents, and communicate to each agent through agent communication language (ACL), this mediator agent to integrate the data between the two agents.

5.4 The Analysis Phase

The objective of this phase is to dissect the system into its component pieces for purposes of studying how those component pieces interact and work.

5.4.1 Role Model

According to Gaia, a role can be viewed as an abstract description of an entity's expected function and is defined by four attributes, namely, responsibilities, permissions, activities, and protocols. A role model identifies the key roles in the system.

The analysis phase has led to the identification of four roles: ***Connection***, ***DataSelection*****, AgentCommunication*****, Dataintegration***

Role: *Connection*
Description: It connect to legacy system, after the extraction of the result,
Protocols and Activities: CheckResultTable, EstablishingConnection, EndConnection
Permissions: Check, Establish, End
Responsibilities: **Liveness: Connection =** (EstablishingConnection.CheckResultTbale) **Saftey:**

Figure 5.2 Schema for a role Connection

Role: *DataSelection*
Description: It selects the data of repeat students from the legacy system, after the extraction of the result,
Protocols and Activities: CheckConnection, SelectData
Permissions: Check, Select
Responsibilities: **Liveness: DataSelection =** (CheckConnection. SelectData) **Saftey:**

Figure 5.3 Schema for a role DataSelection

Role: *AgentCommunication*
Description: It allows communication between the agents in the multi-agent system.
Protocols and Activities: MonitoringSystem, AllowCommuncation
Permissions: Monitor,Allow
Responsibilities: **Liveness: DataSelection =** (MonitorSystem.AllowCommuncation) **Saftey:**

Figure 5.4 Schema for a role AgentCommunication

Role: *Dataintegration*
Description: It integrates the data of repeat students from the mediator agent to the new system after the extraction of the result,
Protocols and Activities: CheckData, IntegratesData
Permissions: Check, Integrate
Responsibilities: **Liveness: DataSelection =** (CheckData. IntegratesData) **Saftey:**

Figure 5.5 Schema for a role DataIntegration

5.4.2 Interaction Model

Interaction models are used to represent links between roles and consist of a set of protocol definitions. The protocol definitions used in this system are detailed below.

Protocol Name: CheckResultTable		
Initiator: First Agent	**Partner:** Connection	**Input:** Result Data
Description: When multi-agent system runs on our system, the first agent will check the table of the result to know if there is update done.		**Output:** No, there is no update or Yes, there is update

Protocol Name: EstablishingConnection		
Initiator: First Agent	**Partner:** Connection	**Input:** Update Result Table
Description: When the first agent detects updating done on the result table, it will establishing Connection between the first Agent and data source in the legacy system.		**Output:** No, Don't establishing connection Yes, Establishing Connection

Protocol Name: CheckData		
Initiator: First Agent	**Partner:** Dataintegration	**Input:** Query
Description: When the first agent detects updating done on the result table, it send query to data source in the legacy system to check the data, if there is repeating student or not.		**Output:** No, there is no repeat student Yes, there is repeat student

Protocol Name: SelectData		
Initiator: First Agent	**Partner:** Dataintegration	**Input:** Selection
Description: When the first agent detects repeating student in the data source of the legacy system, it select the personal data and academic record data for the repeating student.		**Output:** No, don’t select data Yes, select data

Protocol Name: IntegratesData		
Initiator: First Agent	**Partner:** Dataintegration	**Input:** Integration
Description: When the first agent selects the data of repeating student in the data source of the legacy system, it will integrate this data to the data model in the mediator agent.		**Output:** No, don't integrate data Yes, integrate data

Protocol Name: MonitoringSystem		
Initiator: Mediator Agent	**Partner:** AgentCommunication	**Input:** Monitoring
Description: When the multi-agent system run the mediator agent begin to monitor the agents in the system and it activities.		**Output:** No, don't integrate data Yes, integrate data

Protocol Name: AllowCommunication		
Initiator: Mediator Agent	**Partner:** AgentCommuncation	**Input:** Communication
Description: When the mediator-agent monitoring agents activities it allow communication between the agents and between the agents and other systems		**Output:** No, don't integrate data Yes, integrate data

Protocol Name: IntegratesData		
Initiator: Second Agent	**Partner:** Dataintegration	**Input:** Integration
Description: When the mediator agent store the data of repeating student in the data model the second agent integrates this data to the new data source in the new system.		**Output:** No, don't integrate data Yes, integrate data

5.5 The Design Phase

During this phase, the basic architecture of how the system will work and what/how each element will satisfy the users' needs will be laid down. Once again, some of the modeling techniques for the design phase have been borrowed from Gaia, specifically, the ***Agent Model.*** A brief description of the functionalities and responsibilities of our multi-agent system is as given below.

The First Agent: The responsibility of this agent includes connection to the legacy system, check for repeat student, select the data for repeat student, and integrate the data from the data source in the legacy system to the mediator agent.

The Mediator Agent: The responsibility of the mediator agent includes communication between the two agents and integrates the data from one agent to another.

The Second Agent: The responsibility of the second agent includes connection to the new system, and integrates the data from the mediator agent to the data source in the new system.

5.5.1 Agent Model

The purpose of the Gaia Agent Model is to document the agent types that will be used in the system under development, and the agent instances that will realize these agent types at run time. As stated earlier, since the functionalities and responsibilities of the roles

identified in the analysis phase are essentially simple, the mapping between roles and agent types is essentially one-to-one. The agent model depicted below indicates one instance for certain agent type as may be possible in distributed environments running multiple databases, and possibly, one agent providing similar functionalities.

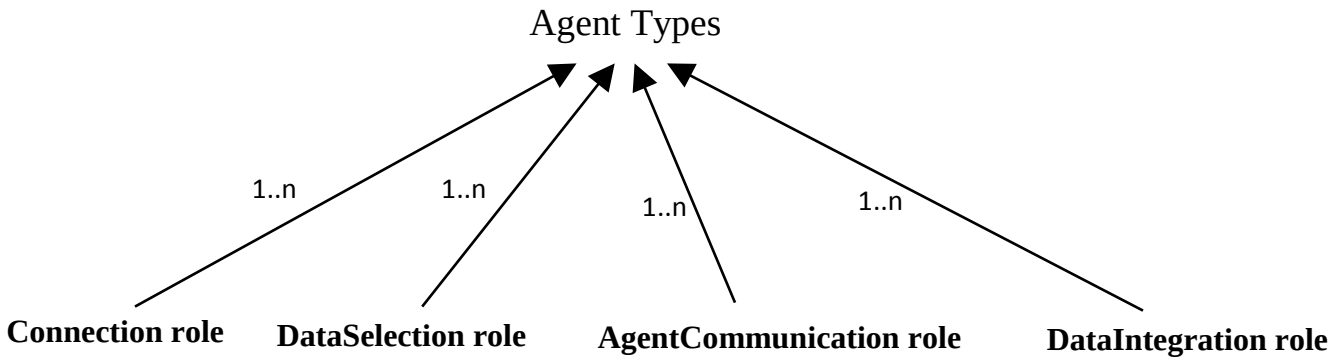


Figure 5.5 Agent Model

5.6 Multi-agent system architecture

The multi-agent system architecture is consisting of the component show below in the *figure 5.6* as follows:

The legacy data source is containing the data for the last batch of students in the legacy system, and to upgrade the legacy system to the new system absolutely we must integrate the data for the repeat students from the last batch in the legacy data source to the first batch of students in the new data source in the new system after every ending of the semester. The multi-agent system consist of three agents, the first agent connect to the legacy system and select the data of repeat students from different tables in the legacy

data source according to a certain conditions and according to the academic regulation. The second agent is manipulating the data by inserting this data in the table of the repeat students in the new data source.

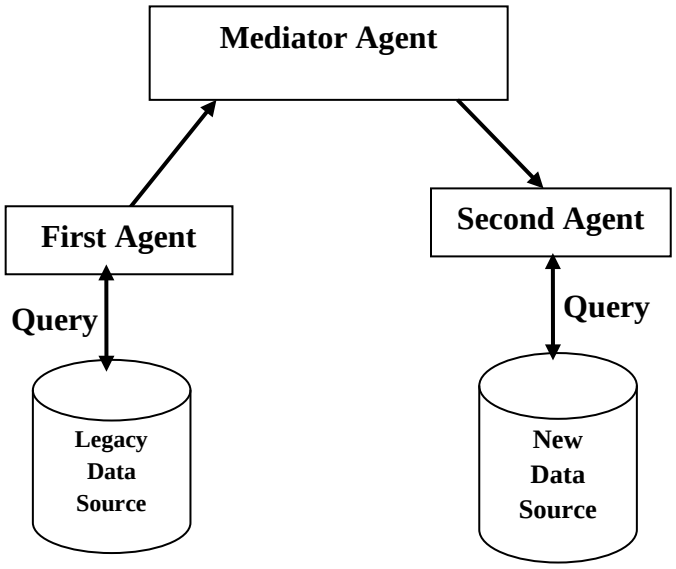


Figure 5.6 Multi-gent system Architecture

5.7 Implementation

In this phase we will illustrate how we implement our agent. We build our multi-agent system in C# language for maximal portability, and it use oracle connector tool called oracle providers to establishing connection between the agent and the legacy data source, also it use mysql connector tool to establishing connection between the agent and the new data source in the other hand.

Our multi-agent system contains number of classes and functions, each of them can execute a specific task in our system.

- First step we implement a class to establishing a connection between the agent and the legacy system data source by using oracle connector tool, this class can establishing the connection when the agent need to select data from the legacy system data source and after finishing the selection of the data, the connection has been closed.
- In this step we also implements a class to establishing a connection between the agent and the new system data source by using mysql connector tool, this class can establishing the connection when the agent need to insert the data which we selected previously, into the new system data source and after finishing the insertion of the data, the connection has been closed.
- Second step we create a query to select the data from the tables in the legacy data source, this query can store the selected data in a model which can store the data as temporary data.
- In this step we also create another query to insert the selected data which has been stored in the model, into the tables in the new data source.
- At last our system is installed as service agent in our operating system, which can be executed autonomously when we start our system.

CHAPTER SIX

RESULTS AND DISCUSSIONS

6.1 Results and discussions

- To test the system it was implemented the multi-agent system as it mentioned previously and execute the system in Microsoft windows operating system.
- Firstly the multi-agent system establishing a connection to the legacy system data source through the first agent from one side, and establishing a connection to the new system data source through the second agent from other side.
- Secondly the mediator agent can be able to communicate to each agent and monitoring the manipulation of the data in the legacy data source, and according to this manipulation our agents can be execute its specific tasks.
- Second step it prepare the results of the students in the last batch in the legacy system after entering the marks of all the courses.
- Next step and according to this manipulation our agent selects the data of the repeat students from the table of results in the legacy data source by the query and in certain conditions.
- Next step the mediator agent stores the selected data in a temporary model.

- And the next step the second agent communicates to the mediator agent and integrates the selected data from the temporary model to the table of repeat students in the new system data source, by inserting this data according to a query.
- In these steps it experiments all of possible cases, which it experiments the cases in all semesters from first semester to sixth semesters in the last batch of students in the legacy system data source.
- Also the multi-agent system generates a log file to records all the data that has been transferred in these steps.

6.2 Comparison with related work

A number of major research initiatives have resulted several large-scale in multi-agent system technology for data integration between different heterogeneous data sources. They include: The TSIMMIS project [13], CIIMPLEX [14], and the MOMIS project [15].

The TSIMMIS project is exploring technology for integrating heterogeneous information sources. Current efforts are focusing on translator and mediator generators, which should significantly reduce the effort required to access new sources and integrate information in different ways. The OEM model described is provides the right flexibility for handling unexpected heterogeneity.

The CIIMPLEX presented a multi-agent system that is capable of supporting intelligent integration of manufacturing planning and execution. With this approach, a set of software agents with specialized expertise can be quickly assembled to help gathering relevant information and knowledge and to cooperate with each other, and with other management systems and human managers and analysts to arrive at timely decisions in dealing with various enterprise scenarios.

The MOMIS project has been conceived to provide an integrated access to heterogeneous information stored in traditional databases (e.g., relational, object oriented) or file systems, as well as in semistructued sources. MOMIS follows a "semantic approach" to information integration based on the conceptual schema, or metadata, of the information sources.

In Our approach the multi-agent system architecture shares with most of them in offering a vision of distributed agents. Most of these architectures employ agents to perform information filtering, monitoring, and brokering. In several respect CIIMPLEX has similar objectives as our multi-agent system: integration between ERP and MES, better manage customer commitment dates, fast and flexible response to disturbances. However, CIIMPLEX develops a pure agent based approach. Most the functions are provided by agents, i.e., new applications. Both TSIMMIS and MOMIS address the need for computer assistance to human problem solving in a complex, information-rich environments.

6.3 Conclusion

Multi-agent system for autonomous data integration between two different systems has been proposed in this research. Some aspects of the data integration has been highlighted, the primary characteristics of agents and multi-agent systems have been described, and has been performed in the form of the design of the agent system proposed. The analysis and design of the proposed system has been presented.

The implementation of this research has been executed, and it was build the multi-agent system using C# language, and also it use oracle connector tool, and mysql connector tool to establishing the connections between the multi-agent system and the data sources. The proposed system has been testing, and it was got the required results that improve the research question.

6.3 Recommendations

- This work can be developed to be applied in wide area for enterprise, to integrate data between different systems.
- In our example, it recommend developing this work and apply to make integration between the system of registration of students and the system of accounting, so as to make sure that the student has paid the tuition fees to the accounting system.
- Also this approach can be developing and apply in the field of electronic governments to integrate the data between different, heterogeneous systems in many government organizations in our country.

References

[1] Koch, C., 2001. Data integration against multiple evolving autonomous schemata (Doctoral dissertation, Vienna U.).

[2] Smith, J.M., Bernstein, P.A., Dayal, U., Goodman, N., Landers, T., Lin, K.W. and Wong, E., 1981, May. Multibase: integrating heterogeneous distributed database systems. In Proceedings of the May 4-7, 1981, national computer conference (pp. 487-499). ACM.

[3] Nagy, M., 2006. Probabilistic Methods for Data Integration in a Multi-Agent Query Answering System. The Open University, Knowledge Media Institute.

[4] Lenzerini, M., 2002, June. Data integration: A theoretical perspective. InProceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (pp. 233-246). ACM

[5] Lane, F., 2006. World Created 161 Billion Gigs of Data in 2006. International Data Corporation (IDC). Retrieved December, 20, p.2011.

[6] Chung, P.T. and Chung, S.H., 2013, May. On data integration and data mining for developing business intelligence. In Systems, Applications and Technology Conference (LISAT), 2013 IEEE Long Island (pp. 1-6). IEEE.

[7] Ahmad, K., WOOK, T.S.F.T. and Samad, R., 2013. KEY BASED APPROACH FOR INTEGRATION OF HETEROGENEOUS DATA SOURCES. Journal of Theoretical & Applied Information Technology, 48(2).

[8] Hai-ping, S., Wei, F., Peng, T. and Yong-sheng, C., 2010, October. Efficient implementation of data integration and sharing of crop germplasm resources investigation. In Computer Application and System Modeling (ICCASM), 2010 International Conference on (Vol. 3, pp. V3-106). IEEE.

[9] Grebla, H.A. and Cenan, C.O., 2010. A Proposed Data Driven Architecture for Cardiology Network Application. BRAIN. Broad Research in Artificial Intelligence and Neuroscience, 1(2), pp.9-20.

[10] Bellos, C., Bibas, A., Kikidis, D., Elliott, S., Stenfelt, S., Sahay, R., Nikita, K., Koutsouris, D. and Fotiadis, D.I., 2013, November. SIFEM Project: Semantic Infostructure interlinking an open source Finite Element tool and libraries with a model repository for the multi-scale Modelling of the inner-ear. In Bioinformatics

and Bioengineering (BIBE), 2013 IEEE 13th International Conference on (pp. 1-4). IEEE.

[11] Mireku Kwakye, M., 2011. A Practical Approach to Merging Multidimensional Data Models (Doctoral dissertation, Université d'Ottawa/University of Ottawa).

[12] IRI the-cosort-company Brochure., 2011. Rapid Architectural Consolidation Engine, The enterprise solution for disparate data models.

[13] Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J. and Widom, J., 1994. The TSIMMIS project: Integration of heterogenous information sources.

[14] Labrou, Y.P.T.F.Y., Tolone, B.C.J.L.W. and Boughannam, A., 1998, December. A multi agent system for enterprise integration. International Journal of Agile Manufacturing.

[15] Beneventano, D., Bergamaschi, S., Castano, S., Corni, A., Guidetti, R., Malvezzi, G., Melchiori, M. and Vincini, M., 2000, September. Information integration: the MOMIS project demonstration. In VLDB (pp. 611-614).

[16] Niang, C., Markhoff, B.B., Sam, Y. and Lo, M., 2013. A semi-automatic approach for global-schema construction in data integration systems. International Journal of Adaptive, Resilient and Autonomic Systems (IJARAS), 4(2), pp.35-53.

[17] Lenzerini, M., 2002, June. Data integration: A theoretical perspective. In Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (pp. 233-246). ACM.

[18] Weiss, G., 1999. Multiagent systems: a modern approach to distributed artificial intelligence. MIT press.

[19] Wooldridge, M., 2009. An introduction to multiagent systems. John Wiley & Sons.

[20] Wooldridge, M. and Jennings, N.R., 1995. Intelligent agents: Theory and practice. Knowledge engineering review, 10(2), pp.115-152.

[21] IBM - http://activist.gpl.ibm.com/WhitePaper/ptc2.htm.

[22] Franklin, S. and Graesser, A., 1996. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In Intelligent agents III agent theories, architectures, and languages (pp. 21-35). Springer Berlin Heidelberg.

[23] Maes, P., 1995. Artificial life meets entertainment: lifelike autonomous agents. Communications of the ACM, 38(11), pp.108-114.

[24] Boussaïd, O., Bentayeb, F. and Darmont, J., 2008. An mas-based etl approach for complex data. arXiv preprint arXiv:0809.2686.

[25] Wooldridge, M., Jennings, N.R. and Kinny, D., 2000. The Gaia methodology for agent-oriented analysis and design. Autonomous Agents and multi-agent systems, 3(3), pp.285-312.

Appendices

A Class to establishing connection with oracle database

```
public DBClassOracle()
{
            _serverName = "amir";
            _dbName = "ORCL";
            _userid = "m";
            _password = "m";

            _cnString = string.Format("Data Source=(DESCRIPTION="

                                      +
"(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST={0})(PORT=1521)))"
                                      +
"(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME={1})));"
                                      + "User Id={2};Password={3};",
_serverName, _dbName, _userid, _password);

            cn = new OracleConnection(_cnString);
        }
```

A Class to establishing connection with mysql database

```
public DBClassMySQL()
        {
            _serverName = "localhost";
            _dbName = "ktc_database";
            _userid = "root";
            _password = "minasmysql";

            _cnString = "SERVER=" + _serverName + ";" + "DATABASE=" +
_dbName + ";" + "UID=" + _userid + ";" + "PASSWORD=" + _password + ";";

            cn = new MySqlConnection(_cnString);
        }
```

A function to open the connection

```
public void OpenConnection()
        {
            try
            {
                if ((cn.State != ConnectionState.Open))
                {
                    cn.Open();
                }
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }
```

A function to close the connection

```
public void CloseConnection()
        {
            try
            {
                if ((cn.State == ConnectionState.Open))
                {
                    cn.Close();
                }
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }
```

A view to select the data for the repeat students from the table of personal_data, and the table of registration.

```
create view RepeatStudent as
select
PERSONAL_DATA.STU_NO,
PERSONAL_DATA.STU_NAME,
PERSONAL_DATA.GRO_NO,
PERSONAL_DATA.PRO_NO,
REGISTRATION.EDU_LEVEL,
REGISTRATION.G_POINTS,
REGISTRATION.G_HOURS,
REGISTRATION.GPA,
REGISTRATION.C_POINTS,
REGISTRATION.C_HOURS,
REGISTRATION.CPA
from PERSONAL_DATA,REGISTRATION
where PERSONAL_DATA.STU_NO=REGISTRATION.STU_NO
and PERSONAL_DATA.STU_NO=registration.STU_NO
and PERSONAL_DATA.GRO_NO=registration.ORI_GRO_NO
and PERSONAL_DATA.PRO_NO=registration.PRO_NO
and PERSONAL_DATA.GRO_NO=6
and REGISTRATION.GPA<2.5;
```

A function to select the data of repeat according to the view

```
{
                        List<RepeatStudentModel> list = new
List<RepeatStudentModel>();
                        for (int i = 1; i <= 6; i++)
                        {
                            _edu_level = i;
                            using (OracleCommand cmd = new
OracleCommand("SELECT * FROM RepeatStudent where EDU_LEVEL =" + _edu_level,
oracleConnection))
                            using (OracleDataReader reader =
cmd.ExecuteReader())
                            {
                                while (reader.Read())
                                {
```

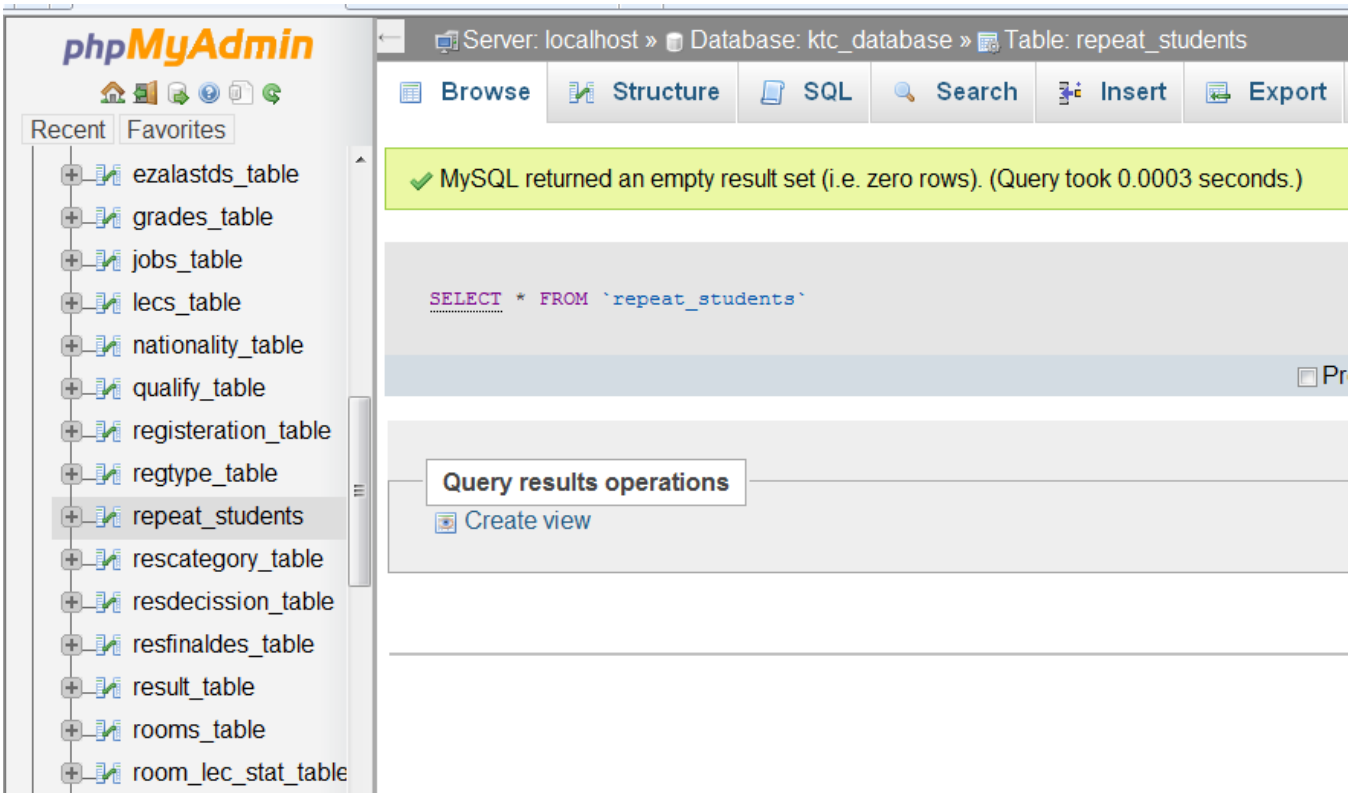
Data model to store the selected data as temporary data

```
RepeatStudentModel model = new RepeatStudentModel();

   model.STU_NO = reader.GetInt32(0);
   model.STU_L_NAME = reader.GetString(1);
   model.GRO_NO = reader.GetInt32(2);
   model.PRO_NO = reader.GetInt32(3);
   model.EDU_LEVEL = reader.GetInt32(4);
   model.G_POINTS = reader.GetDecimal(5);
   model.G_HOURS = reader.GetDecimal(6);
   model.GPA = reader.GetDecimal(7);
   model.C_POINTS = reader.GetDecimal(8);
   model.C_HOURS = reader.GetDecimal(9);
   model.CPA = reader.GetDecimal(10);


                                    list.Add(model);
```

A function to insert the selected data in the table of repeat_student in mysql database according to the data stored in the data model

```
cmd.CommandText = "INSERT INTO
REPEAT_STUDENTS(STU_NO,STU_L_NAME,GRO_NO,PRO_NO,EDU_LEVEL," +
                        "G_POINTS,G_HOURS, GPA ,C_POINTS,C_HOURS, CPA) VALUES(" +
item.STU_NO + ",'" + item.STU_L_NAME + "'," + item.GRO_NO + "," +
                            item.PRO_NO + "," + item.EDU_LEVEL + "," +
item.G_POINTS + "," + item.G_HOURS + "," +
                            item.GPA + "," + item.C_POINTS + "," + item.C_HOURS +
"," + item.CPA + ");";
                        cmd.CommandType = CommandType.Text;
```

Snapshot for the database of the new system **before** running our agent and integrating the data



Snapshot for the database of the new system **after** running our agent integrating the data