Comparing Data Integration Algorithms

Initial Background Report

Name: Sebastian Tsierkezos

Supervisor: Dr Sandra Sampaio

School of Computer Science

Abstract

The problem of data integration has existed for many decades and is still a topic of much research and development. Organisations and businesses have realised the value of generating knowledge through data integration and this driving force has led to a wide range of integration approaches and solutions. The purpose of the project in question is to analyse and evaluate data integration algorithms and this report describes the background information necessary in order to successfully implement and complete the task in question. The report presents some of the most important aspects of data integration, describing the problem and the motivation behind solutions to this. A variety of background information is given, such as the most common approaches to conceptual schema generation and data integration methods and application designs. The general problems of data integration are subsequently illustrated with a schema integration example. Furthermore, the report describes research methods, such as the software development methodology and evaluation techniques, that will be necessary to bring the project to completion.

Contents

Chapter 1

Introduction

Ever since the early 1960s the use of databases has become ever more prevalent in our modern societies. The storage of data has become a big part of our ability to use and manipulate information in order to generate knowledge. The increased use of databases and the development of collaboration between organisations have naturally led to the need for these organisations, whether they are different departments of a government or private companies, to make the data they own available to each other. In this way they are able to generate knowledge and in general make this information more useful.

1.1 The Problem of Data Integration

Data integration is defined as the issue of merging data that exists in various sources with the intention of providing the end user with a single, combined view of the data [1]. Furthermore, given the Local Conceptual Schema of existing component databases, data integration is the process of unifying these to form a Global Conceptual Schema. The issue of data integration generally arises when the component systems already exist, that is to say, integration is generally a bottom-up process. The problem of data integration is not new and has existed for quite some time. As previously mentioned, the rapid and continued adoption of database systems since the 1960s, has meant that the problem of data integration has existed for equally as long.

The issue of data integration has become ever more important in the modern day business environment as large corporations realise the importance of the data which is in their possession and on their information systems. Information that can be drawn from this data can give businesses a competitive advantage and help them survive highly competitive environments [2]. Online retailers have found that data integration coupled with data mining techniques can yield ever more interesting and useful buying trends, which can be used to further focus their marketing campaigns. Furthermore, data integration has gained importance within the scientific community, where various data sets that have been produced by different teams of researchers are being merged in order to extract more useful data and conclusions [3]. Another application of data integration is the combining of data sources that exist throughout the World Wide Web. The fact that web data sources are autonomous and heterogeneous makes the problem of web integration particularly tricky. Web integration can lead to the production of much more reliable search engines.

The main purpose of integrating multiple database sources is to provide a single, consolidated view of the distinct sources, so that the user is given the impression that they are interacting with a single system. There are several reasons why one would wish to combine data sources, two of the most important however are; firstly, through this new, logical, single access point the user is able to reuse the data at his disposal more easily and furthermore, it aids information access. Secondly, the integration of related data sources means that users are querying

and manipulating a larger data set and for this reason they are able to retrieve more absolute and complete results. Whereas the underlying, physical data sources are heterogeneous, users are actually interacting with a system that appears homogeneous even if this is in fact only achieved logically [2].

To achieve this holistic view we must first resolve any inconsistencies between the schemas of the physical data sources. The structure and semantics of the individual schemas must be analysed in order to identify naming inconsistencies, implicit attributes, absent attributes and other differences so that these may be rectified and a new data model developed. In general, however, there is no quick fix solution to integrating several data sources and to a large extent the approach taken, in each case, is governed by the individual characteristics of the information systems being integrated.

1.2Project Description

The main purpose of this project is to investigate the problem of data integration within relational databases. A vast amount of literature exists on the topic and a variety of data integration algorithms, taking different approaches to the problem, have been described. Two of the most popular and widely used data integration algorithms that are present in current literature will be selected, based on which are thought to be the most efficient and most promising. The two algorithms will be designed and implemented and an analysis of these will be undertaken.

An important aspect of the project is to design and implement experiments in order to test and evaluate the algorithms. This will be necessary in order to draw relevant conclusions about the different features of the algorithms and where one algorithm is more efficient and successful than the other.

1.3Aims and Objectives

The aim of this project is to investigate the problem of data integration and to give insight into the issue, using data integration algorithms as a vehicle to achieve this. The project includes both a set of primary objectives and a set of secondary objectives. Primary objectives are those that are deemed mission critical, meaning their completion is of the utmost importance to the project’s success. Secondary objectives are those that are not considered to affect the project’s overall outcome, but would be welcome extra features and achievements, adding to the project's core functionality.

The primary objectives of the project are:

- Take relational tables as input and analyse their schemas in order to identify their differences and similarities.

- Be able to automatically identify and provide solutions for data and schema inconsistencies, such as:

1. Naming inconsistencies
2. Implicit attributes
3. Absent attributes
4. Tables that are included in other tables and more.

- The overall process however will be semi-automatic, that is, the program will require a certain amount of user input in order to carry out full schema integration. This user input is related to the semantics of the integration process, that is, the user will give semantics to attribute and table names.
- Provide a user interface that clearly displays and conveys the integration information to the user, prompting the user for any further integration information that might be required.
- Fully integrate pairs of relational tables into one coherent schema.
- Plan and carry out thorough testing of the integration algorithms, analysing their performance and the results produced.

The secondary objectives of the project are:

- Provide the ability for the user to query over the conceptual schema, that is, be able to carry out simple, read only queries. Updates shall not be dealt with within the context of this project.
- Be able to fragment the query into sub-queries that will be executed at each of the component databases, returning the set of relevant results.
- Re-build the results returned from each sub-query, so as to give the impression of a single result to the user.

1.4Report Structure

The following is a short summary of the remaining chapters of the report and the concepts that will be covered.

Chapter 2 - Background: This chapter covers a variety of topics that are relevant to the project in question. Background information that has been acquired from a review of literature in the field of data integration will be presented. Different approaches that can be taken to deal with the problem of data integration are presented. Example systems are also briefly discussed and compared.

Chapter 3 - Research Methods: This chapter deals with the finer details of the project. It covers the methods that will be used to implement the project. Furthermore, an explanation of the software methodology that will be utilised is

given, as well as reasons behind the choice of programming and implementation tools. Finally, a short introduction to the evaluation techniques that will be executed are touched upon and a Gantt chart of the project work plan is provided.

Chapter 2

Background

This chapter presents background knowledge that has been gained through a review of the literature in the field of data integration. Firstly, approaches that can be undertaken in order to generate a common conceptual schema are discussed. Furthermore, the differences in querying ability of these approaches are covered. The differences between structural and semantic integration are defined, followed by data integration application designs and example systems. Finally, a summary of data integration algorithms covered in the literature is given.

2.1 Approaches for Conceptual Schema Generation

The component data sources contain the information we wish to convey to the user, through the development of a global model or a global schema. This global schema in essence displays the information in a unified manner as a virtual database and is directly related to the data sources.In order to generate this global schema we may approach the problem in two main ways, Global-As-View or Local-As-View. This section describes and compares the characteristics of these two approaches.

2.1.1 Global-As-View

In the global-as-view approach the global conceptual schema is expressed in direct relation to the data sources [3]. The global schema is expressed as views over the component data sources. Therefore, this mapping directly defines how to access the data when checking the elements of the mediated schema. It must be noted, that for this reason, the global as view approach is inefficient if the data sources are constantly changing. However, this approach is effective when it comes to querying, because of the direct manner in which the data sources can be used to access data. Adding new sources to the data integration system is not so simple; the views associated with the global schema may require updating.

2.1.2 Local-As-View

The *local-as-view* method, does not describe the global schema in a direct manner as is present in the previous method but in a way which is disassociated from the data sources. The mapping between the global schema and the data sources is created by creating views of each source over the global schema [3]. In other words, the content of each of the data sources is mapped in terms of a view over the mediated schema. The local-as-view approach is particularly useful when the global schema of the data integration system is established within an organisation or company. Furthermore, it is a simple process to add new sources to the system as all that is required is to create a new mapping for that source, without having to change the mappings for the other data sources [1].

2.1.3 Comparison between GAV and LAV

In both the GAV and LAV approach it can be observed that the basic characteristic of being able to answer queries in terms of the global schema exists. Furthermore, in both cases queries over this mediated schema have to be translated in terms of a set of queries over the component data sources [1]. However, query processing using the local-as-view approach is difficult due to the fact that information about the data in the global schema is gained through views that correspond to the component databases, thus we have fragmented information about the data. Querying in the GAV approach is significantly simplified, this is because we are easily able to understand which source queries map to the mediated schema.

As previously mentioned, adding new data sources in the LAV approach is easy. This is due to the fact that describing new sources is not dependent on knowing anything about other sources and the associations that exist between these sources [3]. In GAV, adding another component database is difficult because views need to be updated.

With LAV, creating more definitive descriptions of the data sources is possible. It is simpler to define constraints on the data within the sources and describe the sources which may have different structures due to the increased expressiveness of the view definition language. The simplicity of defining constrains has the advantage of being able to reduce the number of sources that are applicable to a particular query

2.2 Data Integration Methods

2.2.1Structural Integration

The purpose of structural integration is to try and resolve a variety of conflicts with regards to the structure of the schema. The schema of two data sources may suffer from structural heterogeneity for a number of reasons. Problems that may need resolution include that of type conflicts; for example an address attribute in one schema may have been represented as a string, whereas, in the second schema it has been defined as a struct. Further structural issues that may arise are naming inconsistencies, where attributes that define the same real world object have different names, or different real world objects have the same schematic name. Implicit attributes and absent attributes are another issue that has to be contended with, if the task of structural integration is to be successful. Dependency conflicts arise when for example a salary attribute is represented in terms of net salary and tax in another schema. It is the aim of this project to be able to automatically identify and provide solutions for these types of data and schema inconsistencies.

2.2.2Semantic Integration

Semantics is defined as “the meaning or the interpretation of a word, sentence, or other language form [5].” Semantic integration deals with the issues that are raised when data sources have heterogeneous semantics, that is, the meaning of certain data constructs can be ambiguous or hold a different meaning. The

designers of the system have, in essence, conceptualised the database in different terms. It may not be possible, based on the information held about the source, to decide whether or not two relations which have the same name represent the same thing. Therefore, if we try to integrate these two objects we may end up integrating relations with heterogeneous semantics and thus we will get incorrect results that make no sense whatsoever [6]. It is therefore necessary to ensure that data that is to be integrated is semantically correct.

Semantic integration is defined as "the task of grouping, combining or completing data from different sources by taking into account explicit and precise data semantics in order to avoid that semantically incompatible data is structurally merged [6]." It follows that only data that is deemed to be related to the same real world object must be combined. The problem with this is that there are no set semantic rules that apply for every human user. Semantic heterogeneity can be overcome with the help of ontologies, which are defined "as formal, explicit specifications of a shared conceptualisation [7]." Semantic integration is a field in which much research is being carried out and many open problems still have to be overcome. For this reason, semantic integration will not be covered within the scope of this project.

2.3 Data IntegrationApplication Design

This section describes the general approaches that exist in tackling the issue of data integration. These different data integration designs vary from a simple user centred approach, such as manual integration, to complex application architectures such as middleware integration or the common data storage approach.

2.3.1 Manual Integration

Manual integration is the process by which users interact with the data sources and software systems, integrating data directly. All the process of integration is left to the user and as such the user must view the underlying architecture and the schemas of the data sources. They must also have knowledge of the querying languages that are implemented at each source [6]. Obviously, there are certain disadvantages to such an approach. Firstly, the process of manual integration is slower than automated techniques that exist. Secondly, the user must be highly technically skilled and have knowledge of the technologies in use. Many software systems do, however make some use of manual integration when the automated integration undertaken by the software is erroneous or not adequate.

2.3.2 Common User Interface

Data integration is carried out through the use of a common user interface; this is an interface that has a homogeneous look and feel to it. Less work is necessary in order to carry out integration, as relevant information is drawn from data sources and is presented to the user as a view on the interface. The data, however, is still not integrated automatically and the integration has to be done

by the user. In essence, the common user interface approach is a system, perhaps a web application that is superimposed over several information systems and databases, integrating the data from these, so as to aid in information retrieval. An example of a common user interface is iGoogle or My Yahoo!, where data is drawn from the web and displayed to the user but integration and homogenization has not been completed. As illustrated in the figure below, the user is able to select the information that will be displayed on their homepage, drawn from several information sources across the web.

Figure 1: A portion of the iGoogle homepage [20]

2.3.3 Middleware

Database middleware systems draw data from multiple information systems or databases in order to provide a single integrated schema of these. The middleware is able to bring together information from a variety of sources, that are implemented using different technologies and is also able to query data within these sources. One of the features of a database middleware system is that it must be able to provide transparent connection to the sources and be able to execute complex queries. Furthermore a middleware system must be able to make allowances for data sources that have limited capabilities [8].

A variety of middleware architectures exist in the literature, one of these being the Garlic architecture. The Garlic architecture has many features which are common throughout middleware systems. Figure 1 displays the main components of this architecture. The Garlic middleware is able to execute complex SQL queries over varying data sources through its communication with wrappers. The task of a wrapper class is to both be able to describe the data that exists within the data source and provide methods for the retrieval of information from this source. The source will most likely contain data that is of a different format and modelled differently and as such the wrapper must map the data to conform to Garlic's data model. A global schema is created from the individual schemas of the data sources through a wrapper registration mechanism. The advantage of using wrappers is that new data sources can be added without disruption to existing sources as this process is independent of previous implementations. Furthermore, wrappers can be updated and changed in order to achieve new and evolving tasks. One disadvantage is that, depending on the data repository, it may be quite a complex task to write a wrapper.

A conceptual entity may be part of several objects that exist in different repositories but a user's view of such an entity must be of a single object. The

Garlic architecture makes use of object views that enable more restructuring of the data [8]. One reason the use of these objects is important is that they allow for new methods and new functionality to be defined for these.

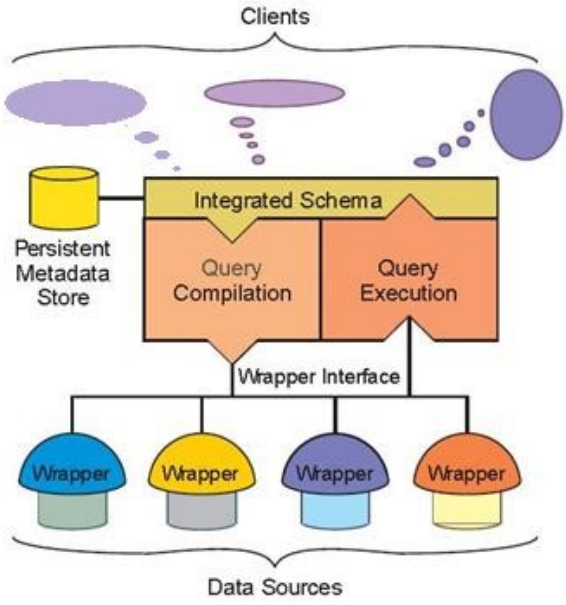

Figure 2: Garlic Architecture [8]

2.3.4 Uniform Data Access

In this approach to data integration, applications have a global view of the physical data which is to be integrated. The distributed source data is logically integrated at the data access level [6]. An example of a type of system that adopts this approach to integrate data is mediated query systems. The purpose of these systems is to make sensible use of large amounts of heterogeneous data, on which they are built on top, by providing a single point for querying the data. Mediated query systems are implemented using mediators which are capable of querying the data sources. A query can be expressed in a unified manner using the system and this query is then broken into sub-queries which are sent to the heterogeneous data sources that are of interest [10]. If data exists within this sourcethat is relevant to the query, then the results are sent back to the mediator and integrated before being presented to the user. Mediated query systems and more generally, the uniform data access approach can be inefficient since data access, homogenisation and integration of the data all occur at execution time.

An advantage of mediated query systems, however, is that they offer support for every data type possible. Furthermore, these systems are able to handle both specific and un-specific queries, whilst also dealing with a dynamic set of information sources [10]. The ability to update data is not a feature that is present in these systems.

A further integration solution that implements the uniform data access approach is portals. Portals present information from diverse sources in a coherent way. They are basically web pages that present information to users in a unified manner, where the data displayed can be personalised in order to suit the individual users. Web mining techniques may be implemented to discover a user's profile by using click- stream analysis. This process collects and analyses data about which pages users visit and in what order. Using these tools the portal can decide upon which pages a user may find interesting and aggregate them and present them to him in a unified fashion.

2.3.5 Common Data Storage

This approach is implemented in data warehouses and operational data stores. The method, which can be described as the eager or in advance approach to integration, retrieves relevant data from the sources, merges them in a logical way and then stores them in a central database [9], as is illustrated in the figure below. This information is said to be materialised as data is drawn from the local sources and integrated to form a single new database. This new database can be used for data mining, analysis or reporting. When a query is submitted it is executed at this central database and not at the original data repositories. The local sources can thus either be withdrawn, if they are no longer required, or remain functional.

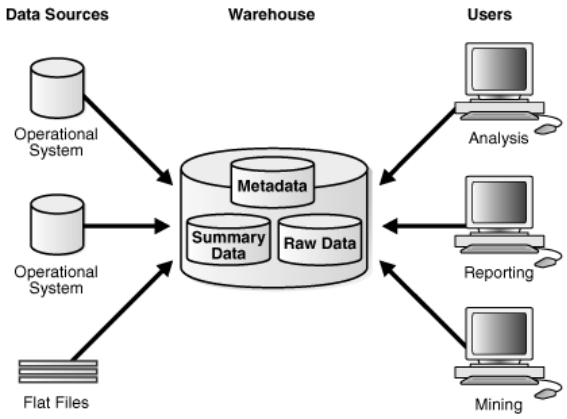

Figure 3: The Architecture of a Data Warehouse [19]

The task of data integration through common data storage is undertaken primarily when the data to be stored is going to be analysed in depth and efficient query processing is of the utmost importance. Faster data access can be achieved through the usage of a central database since data is not constantly transferred from one point to another over possibly unreliable communication lines. Furthermore, queries are executed more efficiently since they do not need to be translated, filtered and then merged as in de-centralised approaches.

Data warehouses and operational data stores are often used when very few or no updates are expected and complex analytical queries are required to be executed. Through these analytical queries organisations aim to gain insight into Customer Relationship Management and Enterprise Resource Planning. The

difference between data warehouses and operational data stores is that in the latter, changes to the local data sources are immediately propagated to the centralised database.

The disadvantage of using data warehousing is that it is not suitable for data repositories that change on a regular basis since this will lead to the phenomenon of stale data in the centralised database [6]. Generally, if local sources are updated and changed regularly and these changes are of importance to the central data store then the operational data store approach is taken so as to avoid the above phenomenon.

2.4 Example Systems

2.4.1 Federated Database Systems

A federated database system is defined as “a collection of cooperating database systems that are autonomous and possibly heterogeneous [11]. “ The component databases are integrated, each having its own database management system (DBMS), but also, a management system that controls the coordination of the databases, called a federated database management system. The databases themselves and the individual DBMS’s play a pivotal role in deciding the overall architecture of the federated system, since these can vary greatly. The variations in the type of component database systems that might make up a federated database system are illustrated in figure 2. In the figure, component DBMS’s cooperate but do not have a high level of autonomy nor do they impose a high level of heterogeneity on each other. A global conceptual schema is created from the local schemas of each of the component DBMS’s. As is illustrated in the diagram, a federated database system may include systems which have a variety of architectures, even another federated system. By integrating data from underlying databases this approach to integration is an example of *uniform data access*. The new federated system is said to be a fully-fledged database management system as it has its own data model, is capable of executing global queries, and supports global transactions as well as global access control [6]. The advantages of such a system are that the user has uniform view of the data and is unaware that there are multiple database systems involved. Federated database systems offer flexibility to an organisation, with little need to change current information systems. A disadvantage to a federated system is that some queries may be costly and difficult to optimise due to the involvement of several systems. Furthermore, federated systems are not suitable for systems where there are many dynamic changes to the schemas as it is difficult to automate the generation of the global conceptual schema.

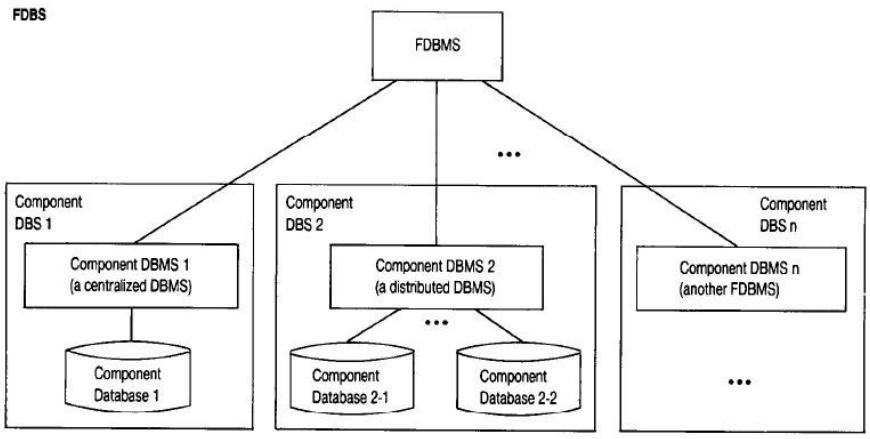

Figure 4: An FDBS and its components [11].

2.4.2 Peer-to-Peer

This approach to data integration is described as decentralised and distributed, where nodes across a network act independently, sharing and integrating data. This can be regarded as a uniform data access approach if the integration then occurs manually or a data access interface approach if the integration is done automatically through the use of some application [6]. A peer to peer system is made up of local mappings as well as peer to peer mappings and each query is executed over a single peer [21].

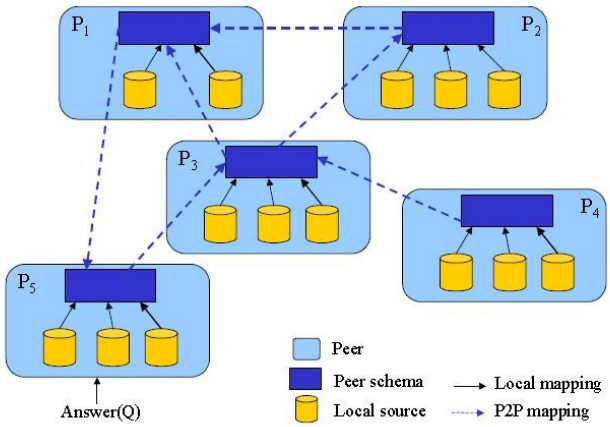

Figure 5: A Peer-to-Peer system illustrating the mapping mechanism [21].

This peer-to-peer integration system is made up of a set of autonomous nodes that exports its data contents as a schema and stores the physical data locally. A single node is related to other nodes within the network through a set of peer-to-peer mappings where each mapping is a schema level assertion which relates data in a node X to data in a node Y [21]. Due to the fact that we are dealing with a peer-to-peer network there is the distinct advantage of having a very robust flexible topology. However, problems arise, such as thespeed and efficiency of search and how reliable data retrieval is since nodes are constantly joining, leaving and possibly failing.

2.5An Example Methodology for Schema Integration

This section describes one of several methodologies available for schema integration. The figure below gives two different, basic conceptual schemas. It must be noted that Topics in the first schema holds the same meaning as KeyWords in the second [22]. Publication in the second schema contains more information that Books in the first, as well as being a more abstract concept.

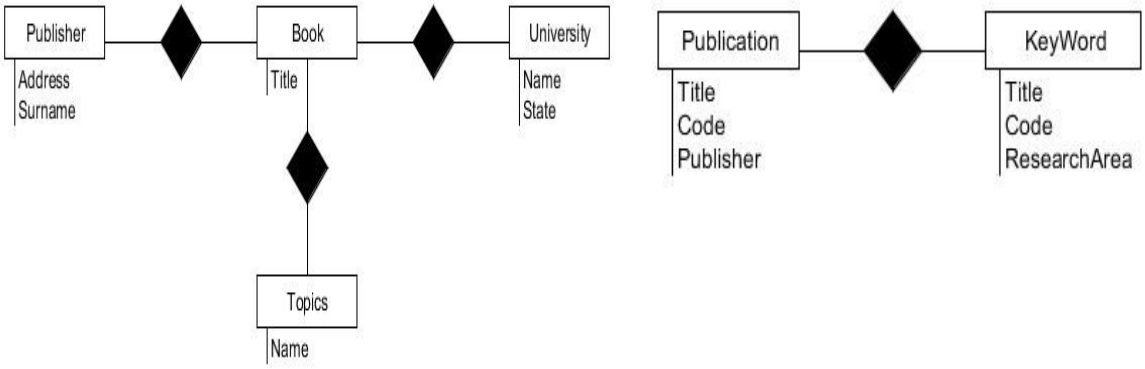

Figure 6a: Original Schemas[22]

The following figures display the possible cycle of events undertaken in order to integrate the two schemas. Firstly, we begin by renaming KeyWords, since both this entity and Topics both relate to the same real world object. This change can be observed infigure 6b, where KeyWords has been renamed to Topics. We can also see that Publisher is an entity in the first schema but an attribute in the second. Therefore we create a new Publisher entity in order to replace the attribute in the second schema, shown in figure 6c.The next step in the integration process is to superimpose the two schemas as illustrated infigure 6d. Within the superimposed schema we notice that there is a subset relationship between Book and Publication illustrated by figure 6e [22]. Finally, we can remove the properties in Book that are common to those of publication. The final merged schema is shown in figure 6f.

Figure 6b: Renaming KeyWords to Topics

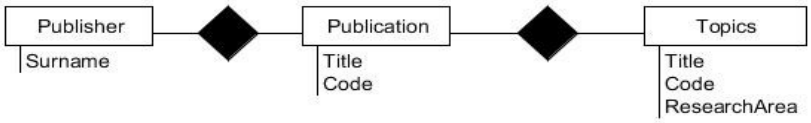

Figure 6c: Make publisher into an entity from an attribute.

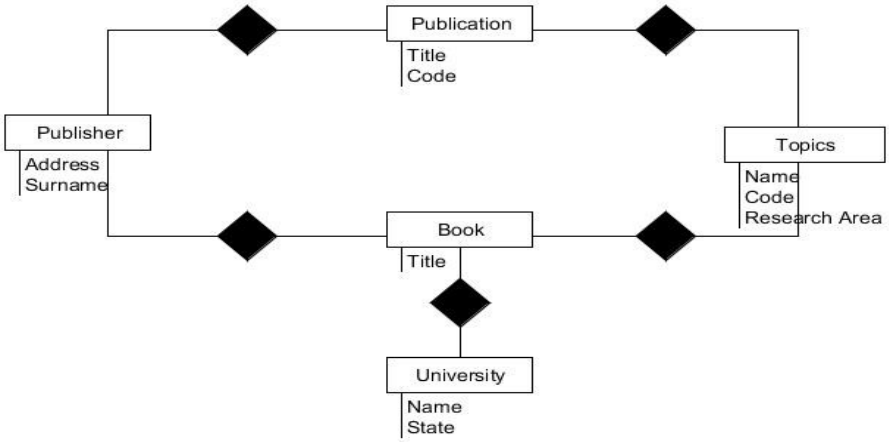

Figure 6d: Superimpose the two schemas

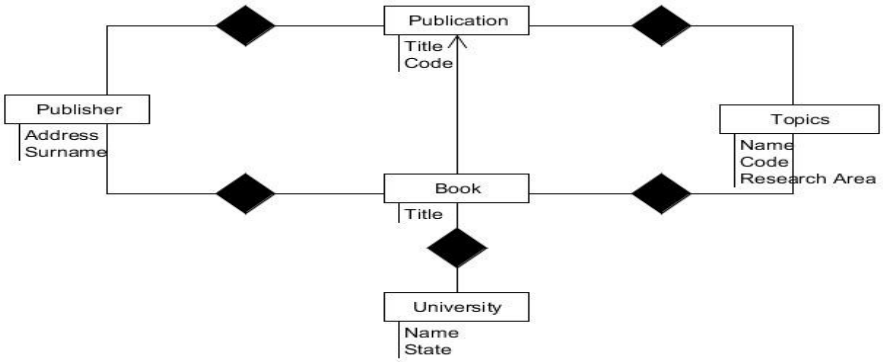

Figure 6e: Make Book a subset of Publication

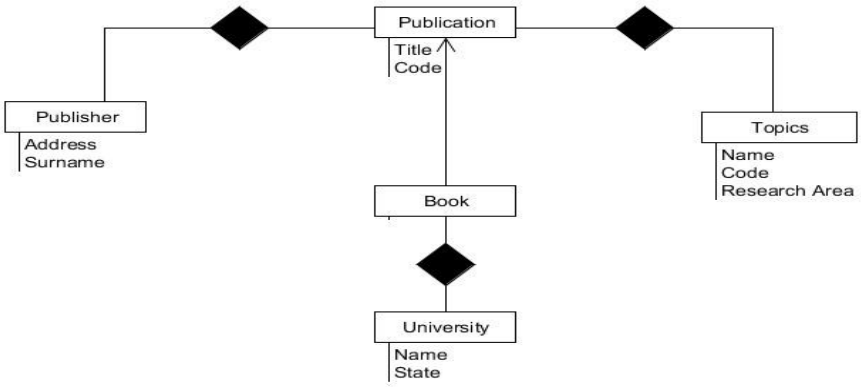

Figure 6f: Remove Books common properties

The above schema integration example is simple, but illustrated some of the problems that a user and data integration algorithm will have to contend with.

2.6Summary of Chapter and Conclusion

The chapter has presented the most popular approaches to conceptual schema generation as well as described the structural and semantic methods to integration. Furthermore, a description of data integration application designs and the advantages and disadvantages to each approach given. Finally, an example methodology to schematic integration has been used as a vehicle to illustrate the task of integration and the kind of problems that will be tackled with respect to the implementation of the data integration algorithms.

My research and analysis of the literature available on the topic of data integration has led me to certain conclusions about the project and general ideas about how it should be implemented. Firstly, despite starting with the intention of not carrying out updates on the data that is to be integrated, the Common data storage approach will not be used. This is because it is not expected that very complex queries will be executed and as such a data warehouse will be of little advantage. It follows that a virtual approach, whereby only a logical integration is undertaken will be followed; the physical data will remain at the local sources. As such, if the secondary objective of enabling querying is achieved, these will be executed over the global conceptual schema and will have to be fragmented, translated and then sent to the data sources to be executed so that they may return the results to be integrated and displayed to the user. Furthermore, a global-as-view approach will be applied in order to generate the global schema. This is due to the fact that it is simpler to query the local databases as global-as-view expresses the mediated schema in relation to the data sources, whereas with local-as-view we would be querying over views, which is sa substantially more complex process. A variation of the middleware approach will perhaps be implemented as this approach lends itself to querying. As previously mentioned, the system will be a mixture of two approaches, where the system will attempt to automatically integrate the data but some user input will be required. The integration will, therefore, be semi-automated.

Chapter 3

Research Methods

3.1 Software Methodology

The selection of a suitable software methodology before starting software development is an important part of achieving success in any project. Failure to choose a successful and well suited methodology can result in massive project failure. The Chartered Institute for IT estimates that in 2002, 70 per cent of projects adopted the waterfall software methodology and of these almost a quarter never reached completion due to project failures. Furthermore, a large proportion of the remaining three quarters suffered project cost or schedule over runs [12]. The reasons behind these failures range from insufficient risk management to insufficient domain knowledge.

Having surveyed several software methodologies and undertaking analyses of the different approaches taken in each, a conclusion has been reached about which methodology would be most suited to implementing and analysing data integration algorithms. A short description of the evolutionary prototyping software methodology is provided and its advantages and disadvantages are discussed.

3.1.1Prototyping

It is expected that it will be very difficult to express the projects requirements and fully understand how the system will evolve with respect to the data integration algorithms and the user interface. Therefore, a careful undertaking of requirements analysis as well as regular reviews of these will aid in reducing the number of unknowns. This will helps further define how the user interface should be developed. With a software development methodology that incorporates prototyping, requirements analysis and development is aided by the fact that we have a prototype of the system to work with, early on in the software lifecycle. This prototype can be used to test design solutions and to demonstrate concepts and ideas and thus help get a better understanding of how one should proceed.

Having a software prototype up and running helps in both requirements elicitation and requirements validation.

Requirements elicitation: We will have the ability to experiment with the system in general, gaining a greater understanding of the data integration algorithms and the user interface. This will assist in getting new ideas and finding strengths and weaknesses within the design of the application. Through this greater understanding we will possibly be able to identify new requirements.

Requirements Validation: The use of prototyping helps to pinpoint weaknesses and errors in previously defined requirements. The system can therefore be altered and the specification changed as a result of our improved appreciation of how the system functions.

In general, prototyping can be used as a tool and technique for analysing risk and in turn reducing this.

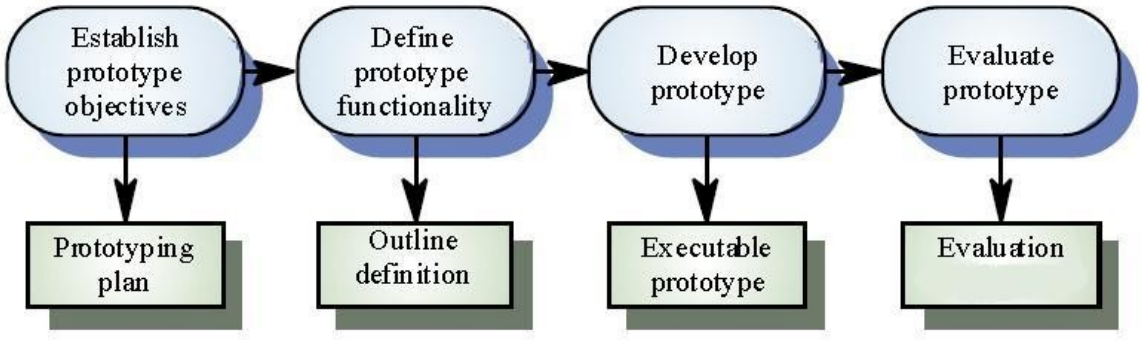

Figure 7: process model for prototype development [13]

The above figure depicts each stage of the prototype development process and the activities associated.

3.1.2 Evolutionary Prototyping Methodology

Evolutionary prototyping is a software methodology that makes use of prototyping. It is the intention of this methodology to have a working version of the system up and running very early on in the lifecycle. Initially, focus is given to the high priority, or primary objectives of a system or project. With these implemented and running, secondary objectives are tackled later on in the development.

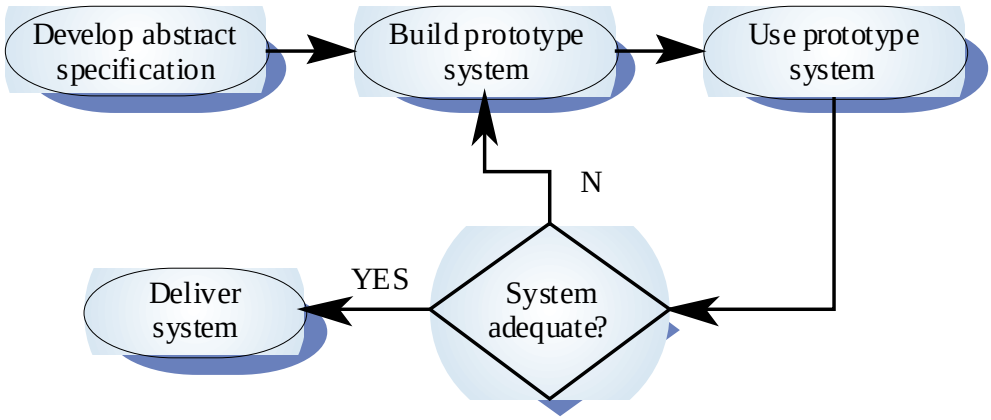

Figure 8: Evolutionary prototyping [13]

After exposing ourselves to this initial, robust prototype, it can be refined over many prototype iterations until we are satisfied with the results. The implementation of this incremental or iterative process is expected to yield an application and user interface that is better suited to the purpose and of higher quality. Evolutionary prototyping has become popular amongst the software

development community as it has proven successful. Furthermore, evolutionary prototyping shares much in common with rapid application development techniques.

Evolutionary Prototyping vs Waterfall Model

The incremental nature of the evolutionary approach is in stark contrast to that of the waterfall software development methodology. This approach is largely a linear process, as is illustrated in figure 5, where each step must be completed before moving on to the next. It has been documented from past failures that this software methodology is not suitable for every project and is not tailored to the dynamic nature of software development [12]. The freezing of the software specification early on in development means that many problems may be left unsolved and very often the system does not fulfil its intended uses and results in poorly structured system. Errors and design problems are identified at the end of the software development lifecycle and it can be very difficult and expensive to make the necessary changes to correct these [13]. The Waterfall model should only be used when the requirements of a project are extremely well understood. For the above reasons, the waterfall model has been deemed unsuitable for the data integration project, especially as we are dealing with a user interface that will most likely have to be changed several times.

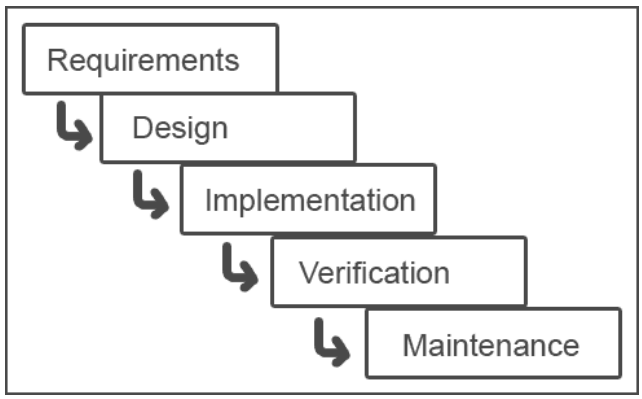

Figure 9: The Waterfall Model.

Characteristics of the evolutionary approach:

- Specification, design and implementation all occur simultaneously in the development lifecycle.
- Incremental system development, iterative process of prototype analysis and re-development.
- Suitable for small to medium sized systems

Advantages of evolutionary prototyping include:

- We are able to see steady progress and have a working version of the software up and running early on in the software lifecycle.

- Useful when the requirements might change and when the application area is not fully understood.
- The prototype can be used to better understand the system requirements and therefore better define specifications resulting in better quality software.
- Better usability of the system
- Reduced development effort.
- The system more closely matches the user's requirements [12].

Disadvantages

- It is not possible to know, from the beginning of the lifecycle, exactly how long the project will take.
- We do not know exactly how many iterations will be required.
- System performance may be compromised if inefficient prototyping code is implemented.
- Maintenance problems: due to the fact that prototype coding may not be structured optimally due to incremental changes, the maintenance of the program will likely only be possible by the original programmer.

3.2 Programming Tools

In order to implement the data integration algorithms and the project in general three technologies have been identified as important and necessary. These have been selected because they are widely used and proven technologies as well as for the fact that they incorporate all the necessary characteristics required for a successful implementation. A description of the technologies and their features follows.

3.2.1 MySQL

MySQL will be used for the storage of the databases that are to be logically integrated and queried over. It is the most popular, open source, SQL relational database management system. The first reason for using MySQL is that it uses SQL, structured query language, the most common language used to access and manipulate data stores. Furthermore, it is able to support relational data stores and the functions that will be necessary in order to successfully implement the project. The MySQL database server has proven itself extremely fast, reliable and able to deal with databases more rapidly than current competition [14]. This point is illustrated by the fact that high profile organisations such as Google and Facebook make use of it.

One of the main reasons, however, for using MySQL is that there is native support for a wide range of application programming interfaces. [14] An added advantage is that it works on many platforms and supports a variety of data

types, adding to its flexibility. Most SQL statements, functions and operators are implemented in MySQL and the MySQL server can easily be accessed using various protocols. A large community of users and developers means MySQL Is very well supported, therefore making solutions to problems easier to find. Finally, most programming languages have API's which include libraries that can be used to connect to MySQL.

3.2.2 JDBC

Also known as Java database connectivity, JDBC offers a standard library for accessing relational databases, regardless of the database product used. It is a uniform interface to databases and therefore if one wishes to change database vendor all that is required is a change of database driver. It enables users to connect to a database and gives them the ability to initiate queries. With JDBC a user is able to send an SQL statement and then process the results returned. A MySQL specific driver exists which is well supported, making MySQL and JDBC a strong pairing.

JDBC consists of two main parts:

JDBC API:

- An industry standard API for database independent connectivity between the Java programming language and a database.
- Provides call level API for SQL based database access [15].

JDBC Driver Manager:

- Establishes communication between a vendor specific driver that in turn communicates with the physical database.
- Checks drivers available and creates connections between the database and corresponding driver.

The figure below displays how JDBC is used to establish a connection with the database and the java application.

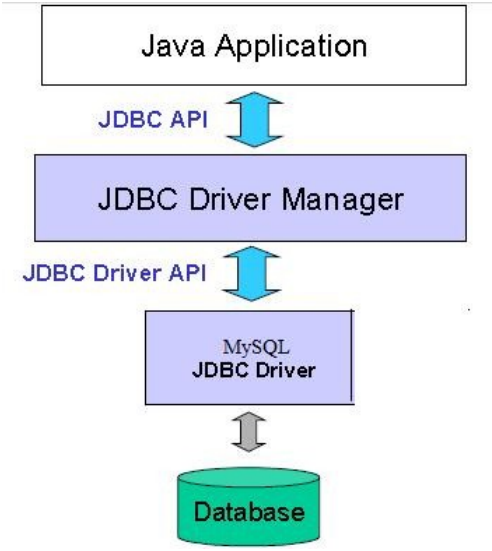

Figure 10: The JDBC Driver Manager

JDBC Functionality: Important Classes

Driver Manager: Loads the JDBC drivers and is responsible for establishing connections to the data source.
Connection: Represents the connection to the database and handles Statement, PreparedStatement and CallableStatement objects.
Statement: Used to represent static, plain SQL statements. Quick and easy to use but tend to be more error prone.
PreparedStatement: Represent precompiled SQl statements. Generally, these are considered to give performance benefits over plain statements.
CallableStatement: Represent stored procedures that can used to run stored preceduresin the relational database management system.
ResultSet: Represents data that is returned after querying the database [16].

3.2.3Java

Java is an object oriented language with similar syntax to that of the C programming language. Applications built using Java can be run on any machine that has had the Java interpreter ported to it. This programming language has not only been selected because it complements the other tools being used but also because it is a powerful, versatile language that offers all the necessary features for completion of this project.

3.3 Evaluation

3.3.1 Algorithms

The purpose of this project is to compare data integration algorithms to give insight into how they function, highlighting their strengths and weaknesses. For this reason, a comprehensive and methodical evaluation of the algorithms must be carried out.

The first step in the evaluation of the algorithms occurs at the research level, articles have seen found describing different data integration algorithms and these have been analyses in order to evaluate their merits. The valuation process described in each of the papers is of particular interest and the results published can be used as a good measure of the success of the individual approaches. Using this critical evaluation method a few promising algorithms have been identified for further study and perhaps implementation.

The next step will be to implement the algorithms that have been selected, experiment with them, testing them based on certain criteria and then evaluating the results received. As described in the previous section on the software methodology that will be used, the process will be incremental and iterative. This means that after evaluation, we will re-implement the algorithms, making any changes deemed necessary or desirable. The improved algorithms will then be experimented with, checking that any changes have proven fruitful before re-evaluating them for further enhancements.

It is expected that different data integration algorithms will yield very different results for the same inputs and that one particular algorithm will be better suited to certain inputs than another. Therefore, in order to carry out correct, scientific experiments only one factor will be changed at a time. That is, before making any changes to the algorithm a variety of different inputs will be tested, one at a time, incrementally and the results assessed. The evaluation process is directly related to the success of the project and so designing good data will be at of the utmost importance. At each increment, we will have to decide if the experiments and data tested are really useful and give insight into the efficiency of the algorithms. If this is not the case, we will have to redesign the experiments and the data. Furthermore, the design of the data should be able to show the different features of the algorithms. If one algorithm is able to identify inconsistencies more easily and reliably than another then this will be regarded as a positive characteristic of the algorithm. The identification of deliberate schema inconsistencies will give insight into where the algorithms are strongest and where improvements can be made. Time permitting, a third algorithm which will include the strongest features of both will be produced.

Two approaches to testing the application for errors will be followed:

White-Box Testing:This testing technique can be used to verify that the application's code is functioning correctly. It is concerned with the inner workings of the system, that is, tests are designed with the source code's logic and structure in mind [18]. Using the source code of the application we can create test cases that test each path of execution at least once and test all a path's true and false cases [17]. Furthermore, we are able to create test cases that check all loop structures and test data structures to make sure that they are valid. White-box testing is generally exercised at an early stage in the design process, in order to identify any possible glitches with the systems code.

Black-Box Testing: This testing technique is explained as knowing the functions an application is supposed to execute and carrying out tests that show each of

these functions is working optimally and at the same time identifying any errors that may arise [17]. Black-box testing is designed to identify a different class of errors to those of white-box testing. It is, therefore, necessary to undertake both types of testing. It does not take into account the internal structure of the system but rather uses external descriptions of the system to create test cases. Black-box testing aims to identify such errors as incorrect or missing functions, errors in the data structures and external database access, performance or behaviour issues and initialisation and termination errors [17]. Lastly, it should be noted that black-box testing is carried out towards the end of the development lifecycle.

3.3.2 User Interface

A good user interface is fundamental to being able to correctly analyse and evaluate the data integration algorithms. In order to improve the user interface a thorough analysis of several schemas will be undertaken, in order to better understand where the user will be giving input. Through this process we will gain better insight into the role the user interface will play in the user-application interaction. Users, in general, will help by providing semantics to attribute and table names, therefore making correlations between the two different schemas. The user interface will be evaluated and improved on in a similar incremental process as described above. It is expected that the evaluation of the user interface will take place before that of the algorithms as the algorithm's evaluation is dependent on the former.

3.4 Work Plan

The following Gantt chart depicts the work plan for the project. Each task related to the project is time-boxed, so as to keep to a strict schedule and ensure that the project is successfully completed. Note that testing has not been included in the Gantt chart as this will occur within every prototype iteration. Each iteration of the lifecycle will be followed by a period of writing up the project dissertation. This is so that results and observations made will be fresh and this will aid in writing a more accurate and complete dissertation.

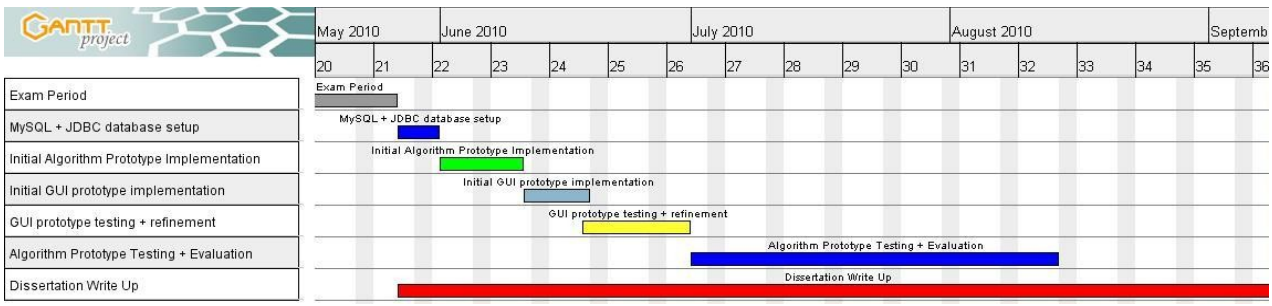

Figure 11: Project Work Plan

4 References

[1] Maurizio Lenzerini (2002),“Data Integration: A Theoretical Perspective.”Dipartimento di Informutica e Sistemistica, University of Rome, Rome, Italy.

[2] Patrick Ziegler and Klaus R. Dittrich, “Three Decades of Data Integration-All Problems Solved?”*Database Technology Research Group, Department of Informatics, University of Zurich* Page 1.

[3] Alon Halevy, AnandRajaraman, Joann Ordille (2006), “Data Integration: the teenage years”

[4]Dittrich, Klaus R. and Jonscher, Dirk (1999). “All Together Now — Towards Integrating the World’s Information Systems.”

[5]World’s Information Systems. semantics." *The American Heritage® Dictionary of the English Language, Fourth Edition*. Houghton Mifflin Company, 2004. 15th Apr. 2010. <Dictionary.com http://dictionary.reference.com/browse/semantics>.

[6] Patrick Ziegler and Klaus R. Dittrich, “Three Decades of Data Integration-All Problems Solved?”*Database Technology Research Group, Department of Informatics, University of Zurich* Page 7.

[7] *T. Gruber (1993). "A translation approach to portable ontology specifications". In: Knowledge Acquisition.* 5: 199-199.

[8] L. M Haas,R. J. Miller, B. Niswonger, M. Tork Roth, P. M. Schwarz, E. L. Wimmers (1997), “Transforming Heterogeneous Data with Database Middleware: Beyond Integration.”

[9] Jennifer Widom (1995), “Research Problems in Data Warehousing.” Department of Computer Science, Stanford University.

[10]RuxandraDomenig, Klaus R. Dittrich (1999), “An overview and classification of Mediated Query Systems.” Department of Information Technology, University of Zurich.

[11] Amit P. Sheth, James A. Larson (1990), “Federated Database Systems for Managing Distributed,Heterogeneous, and Autonomous Databases.”

[12] A study in project failures**, Dr John McManus and Dr Trevor Wood-Harper**http://www.bcs.org/server.php?show=ConWebDoc.19584Accessed: April 17th 2010

[13] Ian Sommerville, Software Engineering (2001), Sixth Edition, Chapter 8. Publisher: Addison-Wesley.

[14] MySQL 5.1 Reference Manual :http://dev.mysql.com/doc/refman/5.1/en/Accessed: May 1st 2010

[15] JDBC :http://java.sun.com/products/jdbc/overview.htmlAccessed: May 1st 2010

[16]*Database Programming with JDBC and Java*. By: *George Reese*; Publisher: O'Reilly Media; Released: June 1997.

[17] Roger S. Pressman (2005), Software Engineering, A practitioners Approach. (Sixth edition), McGraw-Hill, Chapter 14.

[18] Laurie Williams (2006), White-Box Testing. Available: http://agile.csc.ncsu.edu/SEMaterials/WhiteBox.pdf. Accessed: May 4th 2010

[19] Oracle Database Data Warehousing Guide, Available: http://download.oracle.com/docs/cd/B28359_01/server.111/b28313/concept.htm. Accessed: May 6th 2010

[20] iGoogle Homepage, Available: http://www.google.com/ig. Accessed: May 4th 2010

[21] Principles of Peer-to-Peer integration, Maurizio Lenzerini, Available: http://www.doc.ic.ac.uk/~pjm/diweb2004/lenzerini_presentation.pdf. Accessed: May 6th 2010

[22] C. Batini and M. Lezerini (1986), “A Comparative Analysis of Methodologies for Database Schema”.Dipartimento di Informutica e Sistemistica, University of Rome, Rome, Italy.