Sudan University of Science and Technology

Collage of Graduate Studies

A HYBRID SEMI AUTOMATIC APPROACH FOR THE DESIGN OF DATAWAREHOUSE CONCEPTUAL MODEL

طريقة هجين شبه ألية لتصميم نموذج مفاهيمي لمستودعات البيانات

A thesis Submitted to the College of Graduate Studies,
Faculty of Computer Science and Information Technology,
Sudan University of Science and Technology
In Partial Fulfillment of the
Requirements for the degree of

DOCTOR OF PHILOSOPHY in Computer Science

By

Elhaj Elamin Babekir Gafar

Supervisor

Prof. Dr. Jamel O. FEKI

August 2018

بسم الله الرحمن الرحيم

الأيـــة

﴿ بِسْمِ اللّهِ الرَّحْمَنِ الرَّحِيمِ ﴾

قال تعالي:

لِّلَّهِ مَا فِي السَّمَاوَاتِ وَمَا فِي الْأَرْضِ وَإِن تُبْدُوا مَا فِي أَنفُسِكُمْ أَوْ تُخْفُوهُ يُحَاسِبْكُم بِهِ اللَّهُ فَيَغْفِرُ لِمَن يَشَاءُ وَيُعَذِّبُ مَن يَشَاءُ وَاللَّهُ عَلَىٰ كُلِّ شَيْءٍ قَدِيرٌ ﴿٢٨٤﴾

قالَ اجعَلني عَلى خَزائِنِ الأَرضِ إِنّي حَفيظٌ عَليمٌ ﴿٥٥﴾وَكَذلِكَ مَكَّنّا لِيوسُفَ فِي الأَرضِ يَتَبَوَّأُ مِنها حَيثُ يَشاءُ نُصيبُ بِرَحمَتِنا مَن نَشاءُ وَلا نُضيعُ أَجرَ المُحسِنينَ ﴿٥٦﴾

البقرة:٢٨٤

يوسف:٥٥-٥٦

Dedication

This thesis is dedicated to my parents, for their prayers, invocations, support and encouragement.

ACKNOWLEDGEMENT

Foremost, I would like to express my sincere gratitude to my supervisor Doctor Jamel Feki, Professor at the University of Sfax-Tunisia and working actually in the University of Jeddah, Jeddah, Saudi Arabia for the continuous support since we have started this research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of writing research papers and the Ph.D. thesis.

My sincere thanks also go to Professor Ezzeddin Mohamed Osman for his continued support. He was always there when I needed any advice.

I would also like to thank all members of the College of Computer Science and Information technology of the Sudan University of Science and Technology (SUST) for their comments and helps.

I thank Abdel hammed Almahi's family for their endless support and hospitality during my study and research.

Last but not the least, I would like to thank my family: my wife Um Ebtihal, for her support and encouragement; my children: Ebtihal, Mohammed, Mazin, Moubtagha and Malaz for their patience when I was not with them.

ABSTRACT

Organizations strive to build data warehouses (DW) to support their decision making process. The studies show that the most important task for organization's manager is to take the right decision in the right time. Hence, the decision-making process is considered among the main important goals to be achieved by competitor enterprises.

The design process of a DW raises many problems and is considered as a complex and tedious task. DW designers follow different levels concerning the design process. Among these levels, the conceptual level received a significant coverage in the literature. There are many directions one can follow to design a DW. Indeed, some designers prefer to start from user requirements (User requirements driven or *top-down approaches*), another camp of DW designers prefers to start the design process from the data source (DS) data-model (data-source driven or *bottom-up approaches*). Recently, DW designers follow a hybrid approach that benefits from the features of the above two approaches. Putting these two sources together raises a structural problem known as heterogeneity; using ontology while designing a DW helps organizations to solve the problems of semantic heterogeneity between data sources and users' requirements.

This thesis uses a hybrid semi-automatic approach relying on a semantic resource for the design of a DW conceptual model.

First, a heuristic-rule based approach to generate star schemas from relational data source has been defined. To come up with this task, the tables' structures of the relational DS from the DBMS repository have been extracted; as well, a reverse engineering process to classify these DS tables into tables modeling entities and tables modeling relationships has been conducted. This classification helps to generate a multidimensional model (star schema) from the DS data-model.

Second, a natural language approach to build star schemas from business requirements has been employed. This task encompasses three steps: business requirements elicitation, user requirements normalization and multidimensional star schemas generation.

Third, a matching process between star schemas from the DS and that from business requirements is tackled. The matching process is relying on a semantic resource, WordNet in particular, to overcome heterogeneity; as well, the DW designer has been allowed to intervene to approve the star schemas.

The approach has been tested on various examples from the literature. The generated results show the feasibility of the proposed approach for generating approved star schemas.

ABSTRACT (Arabic)

المستخلص

تسعى المنظمات لبناء مستودع البيانات لدعم عملية صنع القرار الخاصة بها . تُظهر الدراسات أن أهم عمل لمدير المؤسسة هي اتخاذ القرار الصحيح في الوقت المناسب . ومن ثم، تعتبر عملية صنع القرار من بين الأهداف الرئيسية التي يتعين تحقيقها من قبل الشركات المنافسة .

تصميم مستودع البيانات يظهر العديد من المشاكل وهي تعتبر مهمة معقدة ومملة . يتبع مصممو مستودع البيانات طرق متعددة في تصميم المستويات ال مختلفة. من بين هذه المستويات، يكتسب النموذج المفاهيمي اهتماما كبيرا في الدراسات السابقة .هناك العديد من الاتجاهات التي يمكن للمرء أن يتبعها لتصميم مستودعات البيانات . في الواقع، يفضل بعض المصممين البدء من متطلبات المستخدم (في اتجاه متطلبات المستخدم أو من أعلى لأسفل)، يفضل معسكر آخر من مصممي مستودع البيانات بدء عملية التصميم من نموذج بيانات مصدر البيانات (أساليب تعتمد على مصدر البيانات أو من أسفل إلى أعلى). في الآونة الأخيرة، يتبع مصممومستودع البيانات نهجًا هجينًا يستفيد من ميزات الطريقتين السابقتين . يسبب وضع هذين المصدرين معًا مشكلة هيكلية تُعرف باسم عدم التجانس . استخدام علم الوجود أثناء تصميم مستودع البيانات يساعد المؤسسات على حل مشاكل عدم التجانس الدلالي بين مصادر البيانات ومتطلبات المستخدمين .

في هذه البحث، تم إستخدم طريقة هجين شبه تلقائيةٍ تعتمد على مورد دلالي لتصميم نموذج مفاهيمي لمستودع البيانات.

أولاً، تم تحديدٍ أسلوبًا قائمًا على القواعد المجردة لإنشاء هياكل نجمية من مصدر البيانات العلائقية . ولإعداد هذه المهمة، تم استخراج هياكل الجداول الخاصة بـ مصادر البيانات العلائقية من مستودع نظام إدارة قواعد البيانات ؛كذلك، تم إجراء عملية هندسة عكسية لتصنيف جداول مصادر البيانات هذه إلى جداول نماذج الكيانات وعلاقات نمذجة الجداول . يساعد هذا التصنيف على إنشاء نموذج متعدد الأبعاد (هيكل نجوم) من نموذج البيانات.

ثانيًا، تم استخدم منهجًا لغويًا طبيعيًا لبناء هياكل النجوم من متطلبات الأعمال. تشمل هذه المهمة ثلاث خطوات: استنتاج متطلبات الأعمال، تطبيع متطلبات المستخدمين وتوليد هياكل النجوم المتعددة الأبعاد.

ثالثًا، تتم معالجة عملية المطابقة بين هياكل النجوم من مصادر البيانات والتطابق مع متطلبات الأعمال. وتم الاعتماد على مورد دلالي في عملية المطابقة على وجه الخصوص،وذلك للتغلب على عدم التجانس. كذلك، تم السماح لمصمم مستودع البيانات بالتدخل لاعتماد الهياكل النجمية.

تم اختبار الطريقة على أمثلة مختلفة من الأ عمال السابقة. أظهرت النتائج جدوى الطريقة المقترحة لتوليد هياكل النجوم.

Table of Contents

List of Tables

List of Figures

LIST OF SYMBOLS / ABBREVIATIONS

BR	Business Requirements
BR-Stars	Business Requirements Stars schemas
DB	Database
DBMS	Data Base Management System
DM	Data Mart
DS	Data Source
DS-Stars	Data Source Star schemas
DSS	Decision Support System
DW	Data Warehouse
DWsing	Data Warehousing
E/R	Entity Relationship
ETL	Extract Transfer Load
FK	Foreign Key
IS	Information System
IDE	Integrated Development Environment
MD Model	Multidimensional Model
NL	Natural Language
OLAP	On Line Analytical Processing
OLTP	On Line Transaction Processing
PK	Primary Key
QVT	Query View Transformation
SBF	Set of Business Facts
SCF	Set of Common Facts
SDF	Set of Data source Facts
SQL	Structure Query Language
UML	Unified Modeling Language

CHAPTER I

Introduction

1.1 Introduction

The Data warehouse (DW) is an important technology for organizations to stand in the competitive market. The basic role of a DW for organization is that it provides comfortable situations for mangers to make good decisions. Based on these facts, organizations strive to build a DW as modern support for the decision making process.

Designers and researchers in DW field argue that designing a DW is a complex, tedious, and time-consuming task. However, as an important tool for organizations, researchers are encouraged to propose valuable approaches and concepts for the DW design. Through their design journey, DW researchers and designers have suggested techniques and methods about the approaches to build the DW. In the literature, there are three approaches for designing a DW. One approach is called *Top-down*, where designers start the design process from user requirements. Another approach is said *Bottom up*, where designers prefer to start from the data source (DS). A hybrid approach merges two approaches, one *Top-down* and one *Bottom up* in order to benefit from the advantages of each one.

The design of a DW performs at three modeling levels namely *conceptual* model, *logical* model and *physical* model. Among these models, the conceptual model gains a significant importance. Based on this, many works have been proposed for designing a DW conceptual model. However, there is no standard designers can follow to design the conceptual model.
In this dissertation, our aim is to suggest an approach for the design of a DW conceptual model. Recognizing the three hard difficulties of a DW (complex, tedious and time consuming task), our approach is characterize by the following features; first, it is hybrid, so both DS and requirements are considered. Second, it is semi-automatic, so that the DW designer can intervene in many stages of the design process and approve the correctness of

the results. Third, it is based on a semantic resource, so the problems of heterogeneity could be controlled and solved.

1.2 Problem statement

DW support organizations with *integrated, subject oriented and non-volatile data*. The DW serves as a large repository of integrated data issued from several DS for data analysis purpose.

DW design process is considered as a complex and tedious task. On the one hand, this complexity appears as the lack of standard concepts and terminology between various stakeholders involved in the DW design process (i.e., uncommon terminological and semantic referential) and the lack of methodological framework and software tools those assist DW designers during the design process. On the other hand, the complexity is due to heterogeneity which could be divided into three types: syntactic, structural and semantic heterogeneity (Thenmozhi and Vivekanandan, 2013).

Although the DW designing process has been tackled by researchers so far, the clear fact sounded up in the literature say that there is no agreement between stakeholders in this field reveal or put an answer to questions like: how to design a DW? What are the appropriate components that may be used to build the DW so as to obey the decision makers' needs? How these components will be used (architecture)? To fill this gab, we want to answer these questions benefiting from a conclusion that explore two points: i) DW should be designed using a hybrid approach ii) using semantic resource helps to overcome problems that affect largely the design process.

Recently, DW designers concentrate their efforts in tackling hybrid approach and using semantic resource. Indeed, these works differ in their methods/ways for some issues such as, generating the elements of the multidimensional model, applying ontology, matching the generated schema etc. In the literature, if we take applying ontology as one issue, we find that most works applying this feature, build ontology based database OBDB to represent ontology domain. This may solve the heterogeneity issues; however, building such ontology will increase the cost and the complexity of this process. On the other hand, some works proposed to build star schemas to guarantee completeness between DS and user requirements. In our work, we suggest generating star schemas from the DS and star

schemas from the user requirements, matching these schemas will increase the ability to accurately produce a proper multidimensional model. Additionally, we plan to use the general ontology WordNet to solve the heterogeneity problems and avoid the cost and complexity of building domain ontology.

1.3 Research significance

Decision makers need to extract synthetic information from a vast data in their organization; DW was developed to obey this need for good business practices. A good decision depends on the way of DW modeling to produce accurate, understandable and significant results.

Recently, DW researchers strive to propose methods to generate a multidimensional conceptual model. In this sense, two remarks should be remembered about the DW design process(Golfarelli and Rizzi, 1999): the first one is that "*although there is a pushing demand for working solutions coming from enterprises and the wide offer of advanced technologies from producers, few attempts towards devising a specific methodology for DW design have been made*". The second remark is that "*the statistical reports about DW projects failures reveal that a major cause lies in the absence of a global view of the design process*".

Moreover, the use of hybrid approach seems to be helpful to solve the disadvantages of using either top down or bottom up approaches solely.

Additionally, the concept of ontology has emerged during the last decade. Several researchers have introduced this concept in many computer field applications as an efficient means to overcome semantic difficulties. The recorded success has encouraged the use of ontology in solving the problems of heterogeneity namely semantic heterogeneity (Pardillo and Mazon, 2011).

Using a hybrid approach, will enhance constructing accurate and complete model. Indeed, involving designer in this process will give approval means to this process. Additionally, the role of WordNet as semantic resource seems to be great in solving semantic heterogeneity while decreasing the time and cost of building a domain ontology. In this

research, we suggest to build a DW conceptual model using this components/architecture to alleviate some difficulties in this area.

1.4 Research Question

The main question addressed in this research is how to build an accurate conceptual DW model using hybrid approach and a semantic resource. Other related sub questions are:

- How to generate star schemas from a relational data source?
- How to generate star schemas from users' requirements?
- What is the mechanism for matching these schemas and using WordNet?
- What is the role of the designer for the approval process?

1.5 Research objectives

The main objective of this research is to build a validated conceptual DW model after reconciling Business-requirements star schemas with Data-source star schemas. Our accurate model aims to involve decision makers during the DW design process. The reconciliation relies on a matching process. The goal behind this matching is to reveal the missing concepts between the two schemas. Below we can list subsidiary objectives:

- Use of semantic resource WordNet to solve the heterogeneity issues.
- Involve the DW designer in the design process to offer feedbacks for the completion of matching process.
- Decrease the cost and complexity of building ontology by reusing WordNet.

1.6 Research scope

Building accurate conceptual model for DW facilitate the decision-making process.

In this research, we want to build star schemas representing a conceptual model for the DW; to do so; data sources should be loaded from relational sources of the operational information system; as well, users' requirements should be taken gathered. To overcome syntactic and semantic heterogeneities, we use WordNet as a general semantic resource.

1.7 Proposed solution

Considering previous observations, we suggest a hybrid and semi-automatic ontology-based approach for the DW design process. Being hybrid, business requirements should be elicited to build what we call Business-Requirements star schemas (BR-Stars), and then the structured DS in form of a database model (i.e. relational) is recovered to build Data-Source star schemas (DS-Stars). Thus, the proposed approach expects to gain the best features of top-down and bottom-up approaches.

The build of these two types of star schemas relies on a set of specific heuristics we define to identify multidimensional concepts (facts and their measures, dimensions with their attributes) from DS and requirements. Note that these heuristics will be automated.

Furthermore, since it is hybrid the approach aims to build a validated conceptual DW model after reconciling BR-Stars with DS-Stars. This reconciliation relies on a matching process and needs quality metrics to assess schemas resemblances. The goal behind the process of matching is to reveal the missing concepts between the two schemas.

Among its main advantages, the proposed a approach reuses a general semantic resource which is usually free open source as WordNet and, therefore reduces the time and cost of the conceptual design. In addition, it does not require skilled specialists in the ontological domain.

Figure 1.1 depicts our approach where the DW designer can intervene mainly through the: i) Construction of BR-/DS-Stars, ii) Matching of these schemas, and iii) Approval of schemas (i.e., validation). The feedback coming from the designer serves to design robust star schemas.

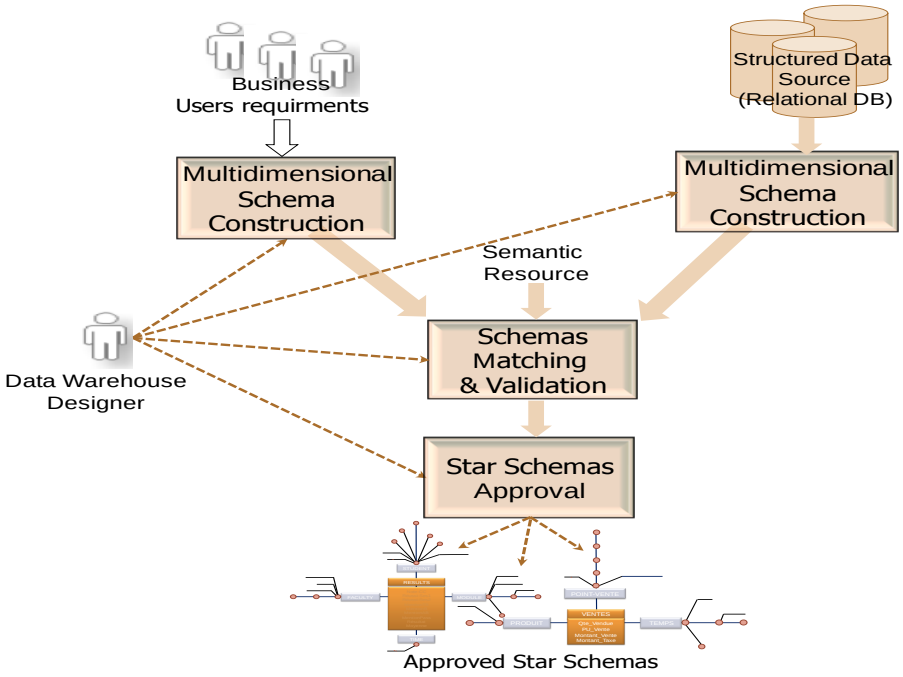

Figure 1.1 Overview of the proposed approach

Our approach agrees with other approaches in that: it is hybrid because it tackles both DS and BR; in addition, it is semi-automatic since the DW designer contributes at different levels of the design process. Furthermore, it relies on a semantic resource in order to overcome the semantic conflicts in names of fields. However, it is important to note that it differs from those of the literature in that it combines: top-down and bottom-up approaches, uses a semantic resource and involves the DW designer to approve the obtained star schemas.

1.8 Research Methodology

The complexity of the DW design process came from the lack of standards designers can follow. Fortunately, the conclusion that the DW should be designed by considering both hybrid approach and by using semantic resource is one result for the researchers' efforts. Still the complexity is present when we talk about how designers can use the hybrid approach and semantic resource. What tools designers can use for each phase to be accurate and clear? Indeed, it is a long complex journey to build the DW. Based on this statement, we follow a planned strategy to build our conceptual design.

Firstly, we comprehensively reviewed the existing works in this area; as well, we make a comparative study between various works. The twofold orientations that this comparison relies on are DW design approaches and the use of semantic resource. Based on the results of this comparative study, we carefully set up our framework components and its architecture.

1.8.1 Tools for designing the conceptual model

Indeed, an activity to fulfill our framework needs support by some tools. Recall that, to build our conceptual model, we have divided our work into three logical phases. First, designing star schemas from data source; second, generate star schemas from business requirements; third, matching the mentioned schemas to generate the approved star schemas which represent our conceptual model. For simplicity purpose, we describe our tools based on their use in the previous phases. It is worth mentioning that, the first phase and second phase has the same order of accuracy (as they are parallel):

1.8.1.1 The reverse engineering process

We used a reverse engineering process before generating star schemas from data source. The objective of this process is to help us knowing which database table initially represents an entity of the real world and which one represents a relationship between entities. In DW field, entities are used for building dimensions, whereas relationships are used to design facts. Indeed, few works use reverse engineering process during the design of the DW. From our viewpoint, it is important process to differentiate accurately between dimension and fact.

1.8.1.2 Database Management System (Oracle)

For generating star schemas from data source, we assume that the database of the operational data source is implemented using the Oracle relational database management system (DBMS). The DBMS stores the metadata of the Meta model that supports the framework. Our justification for using oracle is that it is standard and well-known tool for creating and processing the data in the data source. Additionally, we used java language as

programming environment for developing our software prototypes. As well, the Net Beans 8.0.2 is an IDE for Java that is a simple and flexible tool for designing interfaces.

1.8.1.3 Matrix representation of requirements

This method will be used in our framework during the generation of star schemas from business requirements; accurately, during the Business requirements refinement step. This helps to represent the gathered requirements to facilitate the extraction of the multidimensional elements and then the *BR-Star* schemas derivation.

1.8.1.4 WordNet for matching

The hybrid approach needs a matching process between schemas generated from the data source (*DS-Star)* and the schemas build in behave of business requirements (*BR-Star)*. Additionally, the matching process yields what scientists called heterogeneity problems between elements. Recently, researchers in data warehousing field used ontology as a means to overcome the problem of heterogeneity. However, in our philosophy, building domain ontology increases the time and the complexity of process already complex. Therefore, we used a ready ontology *WordNet* as a general semantic resource. Our aim is to shorten the period of designing our conceptual model. The results we obtained reflect the ability of ontology for solving the heterogeneity shaping the matching elements.

1.8.1.5 RiTa Library

RiTa is an open source free Library designed to be very simple; it has good features. We used RiTa by integrated it with *WordNet* database. RiTa compares two words under testing and checks their resemblance by referencing the WordNet database and returns the distance between the two words semantically; if the distance equals 1, this means that there is no relation between the words; if the distance is 0, this indicates that the two words are synonym (have the same meaning). There is another variation as fraction for the variable distance, this fraction between [0-1]. In our framework, we use a value (less than 1) as threshold; so, whenever the distance is 0 or less than 1, this means that those two words are synonyms. Additionally, RiTa can be implemented in java language.

1.8.1.6 Quality metrics

In order to assess the resemblance of schemas during the reconciliation of *BR- Star schemas* with *DS-Star schemas* we use quality metrics. Our objective behind using these metrics is that it provides a means for measuring the matching between elements in *BR-Star schemas* and elements in *DS-Star schemas.* We have defined four metrics for this purpose.

1.9 Our contributions

Our overall objective is a semi-automatic hybrid approach for designing a conceptual DW model using a semantic resource. As it stated in the literature, designing DW is a complex task and it consumes much time. Having known these complexities, we need to plan carefully to produce our multidimensional model. It is obvious that this process should a stepwise process, as it is complex task. To come up with this process, we divide our contribution into four steps: first we make a comprehensive survey to shed light on the previous proposed works in this area. Second, we use a heuristic based approach for generating star schemas from relational DS. Third, to deal with requirements, we build a model called star schemas from requirements (SSReq). Fourth, in order to generate our approved star schemas' model, we develop a matching process based on the star schemas produced from the data sources and the star schemas generated from business requirements. Hereafter we give a conscious description for each of the mentioned step.

1.9.1 Comprehensive survey in DW design approaches

In order to establish a strong background about the DW design methodologies, and understand its concepts and activities, it is mandatory for us to have a good knowledge in this area. Interestingly, despite the fact that this area is as rich as compared to other areas, unfortunately, until now there are no standards to follow the accomplishment of this task. This situation encourages us to conduct a massive study in the previous works that tackle these concepts. Our study organizes the previous works as conventional approaches and ontology based approaches. Authors in the former works follow a top-down approach or a bottom-up approach or hybrid approach; the shared feature among these approaches is that

they not using ontology. Whereas the later works use a means of semantic resource. The result of the massive study is crowned by detailed comparisons. For the first comparison, we suggest the following criteria:

- Automation: The approach is automatic, semi-automatic or manual.
- Use of semantic resource by the approach.
- Input of the approach: this depends on the type of the approach and the representing method of semantic resource, if used.
- Output of the approach: whether it is a DW conceptual schema or logical schema.

On the second comparison, we aim to make our studding narrower by concentrating our efforts on works using a semantic resource and a hybrid approach. This gives us the opportunity to highlight the shortages in this area. The criteria for the second comparison encompass:

- Technique used for generating the multidimensional model.
- First consideration whether it is DS or user requirements driven.
- The automation, which maybe automatic, semi-automatic or manual.

The final output whether it is Star schemas or others.

1.9.2 Heuristic based approach for automating star schemas construction

As our approach is hybrid, in one-step, we should construct the star schemas from relational DS. To come up with this step, we have applied distinctive methods; indeed, our approach for generating star schemas from a relational DS encompasses three main phases:

- i) *Database model extraction*,
- ii) *Reverse engineering process*, and
- iii) *Multidimensional schema generation.*

The notable issue is that these phases are automatic. Another important feature is that in the reverse engineering process we are the first work that classifies the relational tables of the source database into *three* types: i) Strong entities, ii) Weak entities, and iii)Relationships . Additionally, we have defined a set of heuristic rules for extracting the star schemas'

elements (fact, dimension, etc.). We compare our approach with other works in the literature, the results shows the accuracy of our approach.

1.9.3 Constructing star schemas from business requirements

The second step in our hybrid approach is the generation of star schemas from the model of requirements (SSReq). To do so, first, we have elected a natural language syntax in order to simplify the expressions of the business requirements. Indeed, DW projects going fail as a result of poor representation of users' needs; SSReq model alleviates this problem by using the natural language syntax. Second, requirements elicitation phase is a sound feature in SSReq model; this helps in avoiding the problems shaping the process of collecting requirements from business users as it is a public method. Third, we utilize a matrix of requirements in order to normalize the elicited requirements; as well, to avoid the redundancy in the elicited requirements. Fourth, SSReq automates the generation of star schemas elements (facts, dimensions, etc.); this process relies on eight heuristics we have defined.

1.9.4 Generating approved multidimensional model by matching star schemas

Our contribution in this step is a matching process between star schemas generated from a DS and star schemas constructed from BRs. Our aim behind this matching is to generate approved star schemas that represent our final conceptual DW model. To reach our objective, first we define two Boolean functions; these functions help in determining whether a BR-star schema element (fact, dimension) is identical/synonym to an element of DS-star schema. Second, as the matching process is complex we develop our algorithm *MatchStars.* Third, in order to assist our matching process we define four metrics to measure the similarity between two star schemas. For uncommon elements (fact, dimension) our approach lets the DW designer intervene to match these elements manually. We use WordNet as a semantic resource for solving the heterogeneity between names of elements.

1.10 Organization of the thesis

This thesis organized in seven chapters; hereafter we give a brief description for each one.

Chapter II highlights the main concepts and terms that we need in the context of our thesis. We give definitions for these concepts. As well, we introduce the DW design approaches; we give a concise note about the cons and pros for each approach. Additionally, we coin our motivation for design a DW conceptual model.

Chapter III, entitled Literature Review, reflects the previous studies that tackled the process of designing a DW conceptual model. Our noticeable remark is that this area gains much attention from the researchers; indeed, this reflects the importance of DW as a main tool to generate strategic decisions for organizations. However, the absence of standard is the fact that no one can ignore. For organizational purposes of this chapter, we divide our survey based on two parts: the first one deals with works related to DW design approaches whereas the second part introduces works using ontology as important factor for overcoming the heterogeneity issues shaping the nature of DW activities.

In chapter IV, we reveal the construction of star schemas from a relational DS. This process encompasses three main phases: i) *Database model extraction*, ii) *Reverse engineering process*, and iii) *Multidimensional schema generation.*

The fifth chapter deals with the construction of star schemas from business requirements. We developed a SSReq model to represent the different activities. SSReq model encompasses three steps: i) Business requirements elicitation; ii) requirements normalization; and iii) generation of Multidimensional schemas.

In chapter VI, the star schemas generated from the DS and the star schemas constructed from BR are matched together. The matching process relies on a set of metrics. The aim behind this matching process is to generate the DW model as a set of star schemas compliant to the DS model and users requirements. This process encompasses three steps: i) Matching DS-Stars with BR-Stars; ii) Involvement of the DW designer; and iii) Generation of approved star schemas.

Finally, the results of our thesis are explained in chapter VII. In its all form, the results discussed in this chapter are three parts: first, the results obtained from applying our activities for generating the star schemas from relational data source. Second, the results extracted from applying our methods for constructing star schemas from business requirements. Third, the results generated by applying the matching process between the star schemas issued from data source and the schemas issued from business requirements. We conclude our work and reveal the future work that will be addressed to complete the construction of a DW project.

1.11 Conclusion

In this chapter, we introduced the construction of the DW conceptual model. First, a detailed description about the importance of DW for organization has been tackled. We considered what is the problem statement of the thesis?, the significance and the objectives of the research were tackled. Additionally, the scope and the proposed solution have been determined. Finally, the research methodology and our contributions have been detailed.

As each discipline have its own jargon, it is helpful to give a background for the concepts and terminology in DW research area. The next chapter tackles in details the DW concepts.

Chapter II

Background

2.1 Introduction

In this chapter, we will introduce the background concepts in our research. Accurately, we are interested in the Decision Support System (DSS) field which is an information system dedicated for decisional purposes. Concretely, our aim is to construct a DW conceptual model. This chapter clarifies two subjects: first we introduce the DSS, its components and concepts, and then we tackle the DW and explore its relationship with the DSS.

2.2 Decision Support System

A Decision Support System (DSS) is considered as a special type of Information System (IS). To reveal what is it and its benefits for organizations, the best way is to start from what it relies on; i.e., the operational IS. IS started as a sub-discipline of computer science, the aim was to ease the understanding management of technology within organizations (Inmon, 1992). Through 1960's and 1970's and the beginning of 1980's systems provided decision making emerged DSS. The story began in 1960's; during this period researchers began to study the use of computerized models to help them in decision-making and planning. By reasons of computing power and networks IS became the heart of business. The rapid development in IS yield another concept namely the Strategic IS. These systems have a great impact in nature on organizations, as they can change the organization itself. From this strategic IS concept, many researchers considered the information as a strategic resource.

The initial definition of DSS as stated by(SUDUC *et al.*, 2010) is a: "*computer-based interactive systems that help decision makers to use data and models to solve unstructured problems"*. However, this definition now extended to include any system involved in decision-making process. The importance of DSS comes from support to the analytical

tasks in organization. Of course, there are many important reasons why organizations seek to use these systems.

2.2.1 The benefits of DSS for organizations

Why Organizations uses DSS?, the answer to this question coined in (Turban, Aronson and Liang, 2005)(Druzdzel and Flynn, 2002). Corporations try to develop large scale DSS for the following reasons:

- The environments that Companies' work in are unstable or rapidly changed.
- The competition between the companies is becoming higher than before due to globalization.
- Existing conventional systems do not support decision-making.
- Accurate and new information is needed.
- DSS improved communication.
- DSS can reduce the cost.
- Support choice among well-defined alternatives.

No doubt, any organization try to stand in the competition market, to do so, it should apply some means of systems to support the decision-making. As well, organizations need the accurate and fresh information; this information can be provided by DSS. Additionally, every organization try to reduce the cost of its operational charges, this can be achieved by developinga DSS. Therefore, DSS is becoming more and more mandatory for any modern organization. Furthermore, we should ask the following question: is DSS strong enough to achieve the listed needs for the organization. Let us explain the advantages of DSS for organizations:

- DSS provides powerful models and tools to evaluate alternative decisions.
- Enables managers to answer "*What if?"* questions.
- User friendly and highly interactive.
- Allow the inclusion of new data from external data sources.
- Improves personal efficiency.
- Facilitates interpersonal communication.
- Increases organizational control.

- Generates new evidence in support of a decision.
- Reveals new approaches to thinking about the problem space.

A DSS has been supported by many disciplines, as in computer science. As authors in (Bernus *et al.*, 2008) explain, the ongoing innovations in database management, such as data models, multidimensional data models considered as cornerstone for the progress of DSS. Additionally, some techniques of computer science offered mechanisms for storing and retrieving information, which is a basic component to be taken in decision-making (Bonczek, Holsapple and Whinston, 2014). Note that a modern DSS is built on the concept of Data Warehouse (DW) that we develop in the next subsection. Logically, the DW for the DSS plays a similar role as the database for a conventional IS.

2.3 Data Warehouse: Heart of the Decision Support System

The Data warehouse concept was introduced first by IBM researchers in late of 1980s (Kozielski and Wrembel, 2009)(Golfarelli and Rizzi, 2009b). Through the literature, researchers differ about considering DW as a concept, or as architecture or as a technology. Studies in (Nebot and Berlanga, 2010)(Romero, Simitsis and Abelló, 2011) consider a DW as a decision support tool. Data warehousing term referred to as technology (Selma *et al.*, 2012); other studies called it technique as (Abdalaziz Ahmedl and Mohamed Ahmed, 2014).

Historically, the DW concept can be viewed according to two camps: that of Ralf Kimball and the other of Bill Inmon. According to Ralph Kimball *"*a *Data warehouse is a copy of transaction data, specifically structured for query and analysis*" (Kozielski and Wrembel, 2009). The most popular definition for DW is coined by Bill Inmon *"A data warehouse is a subject oriented, integrated, time variant, non volatile collection of data in support of management's decisions*" (Inmon, 1992). To clarify this definition, we explain its components:

- *Subject oriented*: A DW deals with major subjects in organizations such as customers, suppliers and sales. DW is not a universal structure; rather it is oriented to a certain area in the organization. DW helps decision makers focusing on

modeling and analyzing data and excluding data that are not useful for the decision process. Hence, it provides a simple and concise view around particular subjects.

- *Integrated*: The DW is constructed by integrating multiple heterogeneous data sources, such as relational database, Web data, etc.
- *Time variant*: Data are stored as aim of providing historical information about the organization activities. DW data are explicitly or implicitly time-stamped.
- *Non volatile*: As the DW physically is considered as a separate store of data issued from operational environments, hence it requires two operations: *loading* and *access of data; i.e., once data are loaded into the DW, changes will be forbidden*.

Further definitions for DW can be found through the literature. But, all these definitions derived from the former two definitions.

Organizations strive to build DW to support their decision making process. The studies show that the most important task for organization's managers is to take the right decision at the right time. Hence, decision-making is considered as the goal to be achieved by competitor enterprises. As we have explained before, a DSS supports the collection of data to be analyzed, so managers (i.e., decision-makers) can easily take the opportunity to make a strategic decision. Here, the quality of data is an important factor to make the correct decision. On the other hand, the DW represents the unified source of data that feeds the DSS environment to achieve the decision making process. ***So we can claim that the DW represents the heart of DSS***. Figure 2.1 shows this relation.

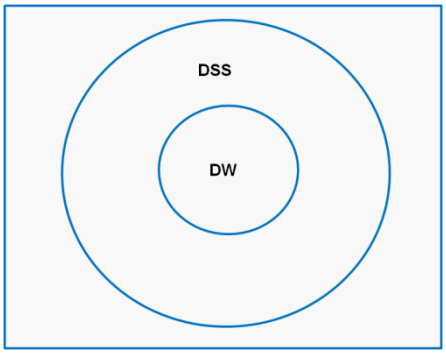

Figure 2.1. The DW is the heart of modern DSSs

As a recent domain different from the operational IS, the DW design and implementation cannot use existing methodologies, data models, technologies… Therefore, practitioners in conjunction with the research DW community have suggested new dedicated solutions for these various levels (DW modeling, querying, design methods…). Hereafter we introduce the concept of multidimensional model.

2.2.1 Multidimensional Model Concepts

Kimball first introduced the multidimensional concepts for DWsing. Although it is not standard model like the E/R model, the *multidimensional model* is helpful for organizing large amount of data, providing a meaningful way of data analysis for managers. Indeed, DW and its dedicated OLAP (On Line Analytical Processing) tools are based on this specific model. Multidimensional model's basic concepts encompass *fact, dimension, measure and hierarchy*. These concepts are useful to design the DW model as a conceptual schema called Star schema. In the next section, we give a concise definition for each of these concepts:

2.2.1.1. Fact concept

The ***Fact*** is the main concept; it represents the subject (i.e., business activity) to be analyzed by decision makers. Fact naturally comes as an event or process, as examples, the *Sale, Buy...* activities *are* facts from the commercial domain. The aim behind analyzing fact is to understand the behavior of the business activity it represents. Facts usually have associated numerical attributes called measures. These measures are the indicators that the user wants to analyze. They are described in the next section.

2.2.1.2. Measure concept

The attributes (i.e., data fields) describing the fact are called ***Measures*** or *indicators* of the analyzed activity. Measures are often numeric to be aggregated through aggregate functions (Sum, Average, Min…).For instance, *Amount, Quantity* are measures for the *Sale* fact.

The set of measures contained within a fact are recorded, and then analyzed, according to a set of criteria grouped into dimensions.

2.2.1.3. Dimension concept

In the multidimensional modeling, the ***Dimension*** concept is composed of attributes and represents an axis of analysis. These attributes play a vital role in the DW. Table 2.1 depicts the characteristics of fact and dimension.

Each fact's measure can be analyzed according to a set of dimensions. As examples, we can cite the *PRODUCT*, *CUSTOMER*, *DATE*… as dimensions for the *Sale* fact. (e.g., the fact *Quantity* measure is recorded for a given sold product, for a given customer, at a given date…).The dimension attributes are organized from the finest to the highest granularity into hierarchies.

Table 2.1 Fact Characteristics vs. Dimension Characteristics

Fact	Dimension
Tends to be measured	Tends to be Analysis-context descriptors
The attributes tend to be numeric	Sometimes the attribute appears as numeric
Can be specified at varying levels of detail	Provides context of measures
Represents elements that are aggregated.	Controls the aggregation of measures

2.2.1.4. Hierarchy concept

Within a ***Hierarchy*** each attribute represents a level for aggregation of the fact measures; such a level attribute is often called a ***parameter***.

For example, the attributes of the PRODUCT dimension could be Prod_ID, Sub-Category and Category. Their semantic organization into a hierarchy is:*Prod_ID* → *Sub-Category* → *Category*.

Among the dimensions, we usually find the *TIME* dimension as a very important dimension because data are recorded and analyzed through time. The following ordered parameters

build a sample hierarchy for the *TIME* dimension:*Date* → *Month* → *Quarter* → *Year* (the highest).

One fact connected to its dimensions builds a *star schema*.

2.2.1.5. Star schema

A ***Star schema*** is a model consisting of one fact and its associated dimensions. The fact size (in number of records) is much larger than a dimension table. The dimension is often usually, generally contains static (time-independent) information about the data recorded in the fact. The name "*Star schema*" perhaps comes from its shape, as the star schema hasa single*fact* surrouded by its *dimensions* with one-to-many relationships towards the fact. Figure 2.2 depicts an example of a star schema.

An Example of fact is: **SALES** (Qty, Amount) with PRODUCTS, CLIENTS, and TIME dimensions

Products (ID-P, Sub-Category, Category)
Clients (ID-C, City, Country)
Time (ID_T, Month-No, Year)

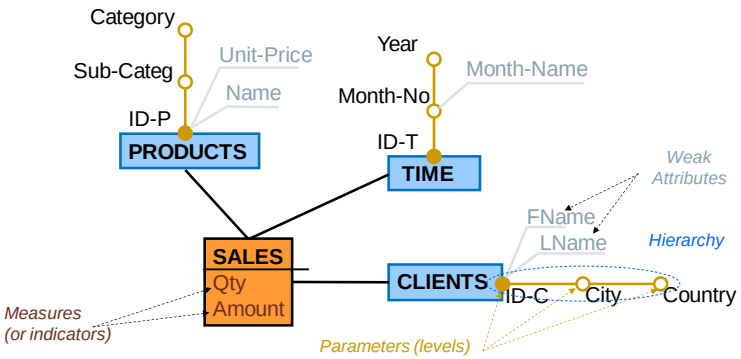

Figure 2.2. Star schema for SALES analyses

The multidimensional model is represented as a star schema; as well there are two other representations for the multidimensional model: *Snowflake* and *Constellation* schemas. Here after a brief description about each one.

2.2.1.6. Snowflake:

Despite the fact that a *Snowflake* schema derives from the *Star* schema, this architecture is considered more complex compared to the star schema. Hence, it used when the star schema unable to represent the complexity of the data structure. The obvious difference between the star schema and the snowflake schemas is that dimensional tables in snowflake are more normalized, so they tend to have relational database design. The objective behind this optimization is to reduce the redundancy. The main drawback of snowflake schema is the complication during join operations, this because of the normalization properties that splits dimensional tables.

2.2.1.7. Constellation schema

The ***Constellation*** schema appears as multiple facts having shared dimensions. It names as well galaxy schema. Constellation schema can be viewed as a collection of stars and hence it is called Fact Constellation.

Table 2.2 depicts the difference between the star schema, snowflake and constellation schemas.

Table 2.2. A comparison between star schema, snowflake and constellation schema

Criteria	**Star schema**	**Snowflake**	**Constellation**
Ease of use	Easy to understand	Complex queries	More complex
Flexibility	High flexibility	Low flexibility	Moderate flexibility
Redundancy	Has redundant data	No redundancy	More normalized
Query performance	Fast	Slow	Slower
Number of joins	Fewer joins	Much joins	Maximum number of joins
Number of fact	One fact	One fact	Multiple facts
Number of dimension tables	Single dimension table for each dimension	May have more than one dimension table for each dimension	-

From the previous comparison in table 2.2, it is clear that despite the simplest shape of the star schema, it has the following advantages (Chandwani and Uppal, 2015)(Jeusfeld and Thoun, 2016):

- Fast query performance.
- Load performance and ease of administration.
- Built in referential integrity.
- Easy understood.

The previous rich advantages of star schema reported it as the most popular architecture for designing multidimensional models. This encourages us to use the star schema to construct the DW conceptual model.

For the design of the star schema, there are three main classes of approaches, namely: *To-down*, *Bottom-Up* and *hybrid* (or mixed). Because the present work is closely related to these approaches, we give an overview for each one in the following section.

2.2.2 Data warehouse design approaches

Through the literature, DW concepts are shared by two camps: that of Ralph Kimball and the one of Bill Inmon. Each of these camps has its impact on various DW activities. On the one hand, Inmon known as the father of the DWsing because he is the first to publish a book entitled“ Building the DW”(Inmon, 2002) in 1990. In this book Inmon illustrated the DW concepts. On the other hand, in 1996 Kimball published an opposite model, for that of Inmon, in his book “The Data Warehouse Toolkit”(Kimball and Ross, 2011). As well, Kimball innovate the notion of the multidimensional model (Breslin, 2004). From that time until now, every work in the DW area either follows Inmon approach or agrees with the Kimball vision. Accurately, in the DW design process, the trend of Kimball is to derive the multidimensional model from users’ requirements. So, the acceptance of the DW in Kimball’s vision measured by how it used. In other words, how the DW is user-friendly. Kimball’s approach agrees with the traditional modeling approach (start designing the conceptual model from requirements); the difference is that Kimball uses different models that used in the traditional modeling. On other side, Inmon follows the online transaction processing OLTP model systems approach, beginning the conceptual model from data sources.

Summing up, a DW can be designed by following two approaches(Carme, Mazonand Rizzi, 2010): Top down approach and Bottom up approach. The former innovate by Kimball, whereas the later justified by Bill Inmon. In recent years, the research trend goes to a hybrid approach that provides solutions to the problems found in the former approaches. Hereafter, we give a short description about each approach.

2.2.2.1 Top-down approaches

These approaches start designing the DW from user requirements (Nebot et al. 2009)(Romero, Simitsis and Abelló, 2011)(Mullin and Motz, 2011)(Nabli, Feki and Gargouri, 2005).One motivation for this camp to use this approach is that the requirement analysis phase is crucial to meet the user needs and expectations(Kimball and Ross, 2011).Another motivation is that researchers following this approach see that overlooking requirement analysis is the main reason for the DW projects going fail. In the DW literature, some researchers named the top down approach as *demand driven* approach or requirements driven approach. However, experts advise to use this approach only when a Bottom-up approach is impossible to be used particularly when the model of the DS is unavailable or complex. Below are the advantages and disadvantages of Top-down approaches (Carme, Mazon and Rizzi, 2010).

Advantages of Top-down approaches

- Achieve good results in respect to user requirements, because it takes in consideration the global picture of the goal to be achieved.
- It ensures a consistent and integrated DW.

Disadvantages of top-down approaches

- The risk to miss some interesting analytical data available in the data source.
- Long-term implementation and high cost.
- It is difficult to forecast the specific needs of every department.
- Users cannot check for this project to be useful, so they lost trust because of long-term implementation.
- This approach requires an expertise in analytical requirement definition in suitable way to produce schemas from the enterprise information system.

An opposite approaches to the former approach was coined by Inmon. In the following lines, we reveal the justification to use the Bottom-up approach as well as its advantages and disadvantages.

2.2.2.2 Bottom-up approaches

This category of approaches is generally more universally accepted than top-down one (Carmè, Mazón and Rizzi, 2010). Indeed, through our literature review, the numbers of articles that follow this approach (Romero and Abelló, 2010b)(Romero & Abelló 2007)(Hachaichi and Feki, 2013), are considerably more than that of the top-down approach. Some features may be needed to successfully apply this approach, such as*; data source schema should show a good level of normalization, as well, data source schema should not be too complex*. Bottom up approach in the literature is named variously; in some works it is called as supply driven approach; in other works, it called data driven approach. Below is the advantages and disadvantages of this category of approaches (Nabli, Feki and Gargouri, 2009)(Carmè, Mazón and Rizzi, 2010).

Advantages of Bottom up approaches:

- The first data mart should be a reference point for the whole DW; so, the upcoming data marts can be easily added.
- It is highly advisable that the selected data mart exploits consistent data already made available.

Disadvantages of Bottom-up approaches

- The main drawback of these approaches is that when the DW schema is derived from the data source it may generate too many results that could be of weak interest to decision makers.
- Another drawback is that these approaches ignore the motivation of end users sufficiently for participating in the development of the DW (Kaldeich & e Sá 2004).

2.2.2.3 Hybrid approaches

In this category of approaches, the requirements and data source analysis is conducted at the same time; therefore, it is called a mixed approach. As this category gain

the benefits of the former approaches, it is considered as the driving force in DW design process (Kaldeich and Sá, 2004). Many research papers follow hybrid approach (Bargui, Ben-Abdallah and Feki, 2010)(Thenmozhi and K.Vivekanandan, 2012)(Thenmozhi and Vivekanandan, 2013)(Nebot and Berlanga, 2010)(Khouri et al. 2012)(Khouri and Bellatreche, 2011)(Giorgini & Garzetti 2008)(Bonifati *et al.*, 2001)(Mazón and Trujillo, 2009)(Mazon, Trujillo and LechtenbÄorger, 2007)(Tria, Lefons and Tangorra, 2012)(Romero and Abelló, 2006)(Romero and Abelló, 2010c)(Calvanese et al. 2006). In fact, hybrid approaches can be classified into pure hybrid approaches and integration driven hybrid approaches. The former approaches consider during the design process each of the DS and business goals separately, whereas the later integrates a bottom-up approach with a top-down one (Romero and Abelló, 2009)(Tria et al. 2012). They may be achieved either sequentially; i.e., the two stages executed in a predefined order; or even in parallel, when the two stages executed independently, the comparison and integration of schemas are performed later (Tria et al. 2012). Figure 2.3 recapitulates the classification of DW design approaches.

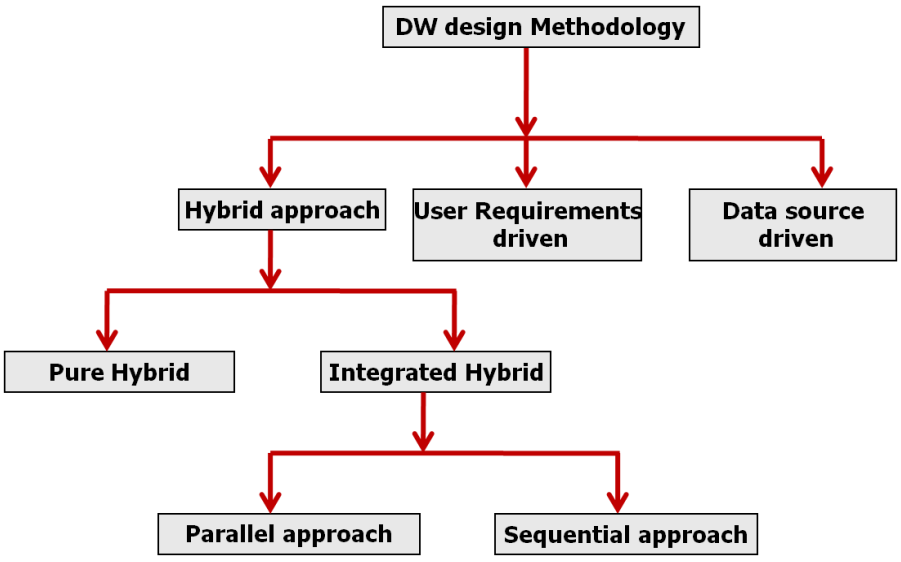

Figure 2.3. Classification of DW methodologies

Whatever the approach we use for designing a DW, different data sources are needed for the ETL process that Extracts, Transforms and Loads the DW repository with data for analysis. However, the integration process yield challenging issues, as each source has its own data structure and meaning. The result for this is *syntactic heterogeneity* and *semantic heterogeneity*. Syntactic heterogeneity refers to ambiguity caused by the use of different models or languages (Cruz & Xiao 2005). Semantic heterogeneity is considered much more complex [80]and can be caused by divergent meanings(Cruz & Xiao 2005). To overcome these heterogeneities, researchers argue that using a semantic resource – as ontology - is among the best solutions for this problem. Next, we introduce the concept of ontology and its benefits for the DW design process.

2.3 Ontology Concept

Ontology is a model for specifying the semantics of concepts used by various heterogeneous sources in a well-defined and unambiguous way (Khouri et al. 2012). Several scientists have introduced this concept in many computer field applications as an efficient means to overcome various difficulties. Ontology term has many definitions, but Gruber coins the widely used definition "Ontology *is a formal, explicit specification of a shared conceptualization".*

Formal means that ontology is machine-readable and executes the use of natural languages, whereas *explicit* refers to that the concepts and the relations between them and the constraints on their use are explicitly defined. *A conceptualization* represent abstract model of phenomenon. *Shared* means that ontologies aim to represent knowledge intended for the use of a group.

The recorded success has encouraged the use of ontology in DSS. Using ontology for designing a DW helps organizations to solve the problems of heterogeneity. Experience shows that, whereas syntactic and structural heterogeneity may be easily solved, it is difficult to treat semantic heterogeneity (Thenmozhi and K.Vivekanandan, 2012)(Pardillo and Mazon, 2011)(Chaudhuri and Dayal, 1997).

Ontologies have been studied so far among the process of DW design, through this way, many tools/models related to ontology have been proposed as well as languages *RDF-RDFS – OWL – PLIP*, etc.

Using ontologies helps organizations in many ways/levels (Thenmozhi and K.Vivekanandan, 2012)(Selma *et al.*, 2012)(Khouri and France, 2010). Benefits of ontology use is the DW design process are:

- Ontology provides the conceptual representation of the domain.
- Ontology helps to represent business requirements in a formal way.
- Ontology used to solve the problem of syntactic and semantic heterogeneities between different data sources.
- Ontology helps to represent Web data into robust information thatcan be incorporated in a DW.
- Ontology is used to analyze the knowledge related to a specific field of modeling and facilitates the distinction of the various domain concepts.
- Ontology provides a way for automating the design of multidimensional schema.

However, there is not a clear vision of situations where ontologies may be applied and there is not still general agreement about how to elaborate ontologies (Pardillo and Mazon, 2011) for a DSS.

2.4 The motivation

The DW is the important technology for any organization to be stand in the competitive market. Nevertheless, despite the estimated efforts of researchers in this area, many studies reported that still there is shortage in designing a DW. The most important phase of designing a DW is the conceptual model. Researchers agree that the DW design should consider both data source and business requirements; additionally, the heterogeneity between the various sources should be taken into account. From our viewpoint, we can note the three following issues. First, few works shed lights on designing DW using the hybrid approach combined with a semantic resource; second, designing a DW conceptual model is a tedious, complicated and time consuming task as it contains many detailed steps; third, yet there is no clear steps one can follow to design a DW conceptual model.

The above drawbacks motivated us to shed light on designing a DW conceptual model. We start designing the DW model according to the hybrid approach and we use a semantic resource. Note that our approach is distinct from the existing ones because it relies on a reverse engineering process that facilitates extracting the multidimensional components from the DS model. As well, we use a NL-based method for representing users' requirements; additionally, we automate the generation of multidimensional elements from both DS and business requirements. Moreover, we plan to use an existing-ready semantic resource (WordNet), our philosophy here is to shorten the DW project development time.

2.5 Conclusion

This chapter provided a good background about the concepts and terminologies in DW research area. Two subjects have been clarified: first, the DSS has been introduced, as well, its components and concepts are covered. Second, the DW is described and its relationship with the DSS is explored. In addition, the ontology concept was explored and its role as a solution for the heterogeneity problems shaping the nature of DW was revealed. Finally, the drawbacks for designing a DW conceptual model were discussed.

Chapter III

Literature Review

3.1 Introduction

This chapter tackles the previous studies in the DW design research area; more accurately, it reviews the literature about the approaches used in designing a conceptual model for DW.

In the DW literature, several works have been proposed, various approaches or/and methods have been used to facilitate the DW design process. In common to these methods, using hybrid approach and ontology have been considered as an important factor for succeed.

For organizing purposes of this chapter, we consider two axes: the first one represents issues related to the DW design. In fact, DW can be designed by using three approaches: user requirements driven approach, data sources driven approach and hybrid approach; these approaches are called conventional approaches. The second axis tackles ontologies in data warehousing. In between, we may merge the two axes as necessary.

3.2 Data warehousing approaches

Functionally, the DWsing process consists of three phases: i) extracting data from operational data sources, ii) organizing and integrating data consistently into the DW, and iii) accessing the integrated data in an efficient and flexible fashion by decision makers (Thenmozhi and Vivekanandan, 2012)(Golfarelli, Maio and Rizzi, 1998)(Cravero and Sepúlveda, 2014)(Golfarelli, Maio and Rizzi, 1998)(El-Sappagh, Hendawi and El Bastawissy, 2011).

In the literature review, DW designers follow different directions concerning the number of the design phases. Although the core phases are four (Rizzi *et al.*, 2006), studies in (Peralta, Illarze and Ruggia, 2003)(Selma *et al.*, 2012) proposed three steps relative to conceptual

level, logical level and physical level, whereas the study in (Luján-Mora and Trujillo, 2006) follows the conventional core four steps. Another comprehensive study conducted by (Golfarelli and Rizzi, 2009a) defines eight steps to design a DW, these steps include: requirements analysis, analysis and reconciliation, conceptual design, workload refinement, logical design, data staging design, physical design and implementation.

Whatever the design approach, the design step concerning the conceptual level gains the significant care in the literature. Indeed, almost researchers and practitioners agree that the conceptual design is considered as a keystone on which depends the success or the failure of the DW project (Golfarelli, Maio and Rizzi, 1998)(He *et al.*, 2011)(Tryfona, Busborg and Borch Christiansen, 1999)(Golfarelli and Rizzi, 2009b)(El-Sappagh, Hendawi and El Bastawissy, 2011)(Tsois, Karayannidis and Sellis, 2001). Furthermore, the failure in DW projects is usually returned to the poorness and inadequacy of conceptual design methods (He *et al.*, 2011)(Phipps and Davis, 2002). However, for our knowledge, there is no standard methodology designers can follow to achieve the conceptual design phase.

3.2.1 Conventional approaches

DW design process passes through many stages. It is difficult to talk about any activity for DW without mention the efforts of two pioneers in the domain: Bill Inmon and Ralph Kimball, in this scene, Kimball innovates the Users' requirements (Top-down) driven approach for designing DW, whereas Inmon justifies the Data source (Bottom-up) driven approach. Recently, researchers in DW area innovate the hybrid approach. Below we tackle papers that follow each of these approaches.

3.2.1.1 User requirements driven approaches

In user requirements driven approach, DW designers start generating the multidimensional model elements (fact, dimension…) by considering business users' requirements. The philosophy of designers using this approach is that a requirement is an important factor for designing the DW. The following works follow this approach.

As example, (Nabli, Feki and Gargouri, 2005) proposed a methodology for generating Data Mart (DM) schemas from OLAP requirements. Their approach consists of two models: The OLAP requirements model specified as a set of two-dimensional fact sheets (the input

model) and DM multidimensional model (the output model). However, the difference between the two models is not so far; this means that both models have the same concepts (fact, measures, dimensions and hierarchies). Additionally, there are no semantic means to formalize the semantic relationships between the concepts in the business users' requirements. Moreover, the authors did not give any description of the elicitation process of requirements. In (Bargui et al. 2008), the authors proposed a method to define a DW Requirements Model (DWRM). The specification of OLAP requirements in this model relies on a natural language template; this model is used to extract semi-automatically the multidimensional concepts and generate data mart (DM) schemas. However, they formalized the query styles by linguistic patterns; the limitation here is that these patterns formalize only the commonly and frequently used styles of query writing. Additionally, their work does not offer a method for organizing the attributes into hierarchies.

In (Mazón *et al.*, 2005)the authors classify business goals into three categories: strategic goals, decision goals, and information goals. They use the i* method to model the business goals and the relationships among them. To extract the multidimensional elements (facts and dimensions) from the i*model, they defined guidelines with UML profile. Nevertheless, their method for defining goals is complex and tedious for the stakeholders to understand. The study in (Prat, Akoka and Comyn-wattiau, 2006) presents a UML based method that encompasses the conceptual, logical and physical phases of DW design. Authors in this study defined a generic multidimensional meta model unifying the concepts of the main multidimensional meta models used at each phase. As well, to ease the mapping from one level to the following one, they provide a set of semi-automated Transformation rules. A conceptual model called *starER* model is proposed by (Tryfona et al. 1999); this model combines star model and entity relationship model; authors firstly addressed the modeling requirements of a DW.

A recent work proposed by (Kozmina, Niedrite and Solodovnikova, 2008). Authors in this work offers a Meta model for formalizing business requirements. They develop an algorithm for generating the DW conceptual model described using Common Warehouse Metamodel (CWM). However; their work is applicable only when there is available knowledge base of other DW with similar requirements.

Although these works innovates various methods in order to design the DW, nevertheless all these works have the shared drawback that represented by considering only user requirements; hence, ignoring important concepts/elements in the data source. This drawback encourages other researchers to start designing the DW from the data source data-model that could be an E/R Diagram, a UML class Diagram or a relational schema.

3.2.1.2 Data sources driven approaches

In the literature, some works follow the Inmon approach where authors construct the DW from the DS data-model. For example, in (Song, Khare and Dai, 2007)the authors proposed the SAMSTAR method; its input is an Entity Relationship Diagram and its output is a set of star schemas. The heuristics used in this method are based on the observation that there is a *many-to-one* relationship between a fact and a dimension. The classification of tables into two categories as potential fact and dimension is based on the following: tables lying on the *many* side of the *many-to-one* relationship are candidate for facts, whereas tables lying on the *one* side are candidate for dimensions. However, although authors of this method have designed an algorithm to define the components of star schemas, we note that the initial observation for this classification is inaccurate, as many tables lying on the *many* side may be candidate for dimension as well; so this classification may generate a wide range of candidate facts. In order to alleviate the complexity of the DW design process, SERM (Structured Entity Relational Model) had been proposed in (Boehnlein et al. 1999) where the authors develop an idea to derive the initial DW model from a conceptual Entity relationship (E/R) model. The SERM describes in three stages how to transform the E/R model into structured form to obey the needs of multidimensional modeling. However, their work is manual and hence needs a high-level expertise person in the application domain of the DW.An automated approach to derive facts had been given in(Carmè, Mazón and Rizzi, 2010), authors in this work proposed heuristics to define facts from relational data sources. The novel idea of this work over its relevant works that extracts facts from relational data sources is that, this work aims to formalize the process of detecting facts, whereas other works uses informal mechanisms as guidelines or glossaries to support designers. They formalize their heuristics by means of (Query/View/Transformation). However, some of these heuristics are weak (a table may

transforms into a fact if it has a large ratio of numerical attributes) and did not reflect the constraints related to the nature of Relational DS schema. This rule may be applied to extract dimension as well; because a table that has a large ratio of numerical attributes may be an Entity as well; so this heuristic did not help to achieve the property of disjoint-classification of tables in the given schema; the disjoint between fact and dimension means that the rule should classify a table accurately as fact or dimension but not both at the same time. In (Hachaichi et al. 2010) the authors build star schemas from both XML and relational data source after combining these two data sources. Few works perform a reverse engineering process to classify the database relational tables into tables describing relationships and tables representing entities. In fact, rules applied in this classification are variable. In our opinion, it is important to improve these rules. This is why we suggest rules those take into consideration some specifics related to the nature of Relational DS schema; particularly, we give attention to the *Weak entity*, of course not all schemas have Weak entity, but if the schema has Weak entity, our rules will help to classify those concepts. Therefore, our rules produce better results.

A conceptual model dedicated for DWs baptized DFM (Dimensional Fact Model) was proposed by Golfarelli (Golfarelli et al. 1998); in this model the representation of the decisional world (i.e., reality) is called dimensional schema. The basic concepts of the DFM are the fact, dimensions and hierarchies.

Another study entails this approach tackled by (Jensen et al. 2004). Authors aim to construct automatically a multidimensional schema from relational OLTP databases, by using a set of algorithms. The discovering process consists of four steps. After obtaining metadata from the DS, the authors annotate this with the analytical information. Next, they discover inter-relationships between tables and construct dimensions from these relationships. Finally, within each dimension, they discover hierarchies.

Constructing the DW only from the data source have the drawback that represented by generating many concepts/elements, as well as the risk to miss some important concepts for users. Fortunately, some researchers innovates the idea of designing a DW from both data source and business requirements, this new approach known as hybrid approach.

3.2.1.3 Hybrid approaches

In the hybrid approach, both user requirements and data source are considered. Hence, the works follow this approach avoid the disadvantages shaped the previous mentioned approaches.

The nearest approach to our approach is given in (Bonifati *et al.*, 2001). On the one hand, authors use Goal/Question/Metric (GQM) technique for capturing users' requirements by means of interviews. The capturing goals are then aggregated and redefined by means of abstraction sheets; from these sheets, the star schema is generated. On the other hand, they use the graph for constructing the star schemas from the DSs. Despite their efforts on using systematic approach, their works suffers these limitations: first, in behave of requirements, they use interview technique in capturing user requirements, but they do not define style for formalizing the goals, this may lead to that the users can give their goals in a different manner that complicates the process of star schema construction. Second, in behave of generating star schemas from DSs, authors dedicated algorithm for exploring the ER model; mapping this model to a connectivity graph without using the reverse engineering process may result in poor defining the star schema's elements. Additionally, their work is manual in most of its steps and there is no consideration for the semantic resource.

The study in (Giorgini, Rizzi and Garzetti, 2008) proposed a goal oriented method for the DW design. Whereas Authors in (Giorgini, Rizzi and Garzetti, 2008) proposed a goal oriented technique for requirements analysis GRAnD which can be employed with user requirements driven framework or with hybrid framework. The authors in (Tria, Lefons and Tangorra, 2012) use hybrid approach to build their model. Their hybrid framework has two steps: requirements analysis and conceptual design. First, from the user requirements they use the *i** model to obtain the multidimensional model, this model formatted as a UML schema. The UML schema after that is reconciled with the DS. By using the attribute tree, they produce fact schemas as the final output. Although their work is systemic and organized, but it lacks the automation, so it is difficult and time-consuming task for the DW designer. In (Mazón and Trujillo, 2009), authors proposed a hybrid approach, they developed a conceptual model called platform independent model (PIM) from an information requirements model; this model then, must be reconciled with the data sources.

Authors in (Mazon, Trujillo and LechtenbÄorger, 2007) proposed a hybrid approach to develop aDW, they make use of two conceptual multidimensional models; on one hand, they first used a conceptual multidimensional model for capturing users' requirements; on the other hand, they used a multidimensional normal forms to define a set of Query/View/Transformation relations. They reconcile the users' requirements model with that of data sources to ensure its correctness. The authors succeed in applying a systematic approach for developing the multidimensional models. However, they ignore the star schema and its elements that are fundamental components in the multidimensional model.

Authors in (Romero and Abelló, 2006) start analyzing the requirements in terms of SQL queries in first step. After that, they analyze the operational relational data sources in parallel with the first step. The aim of the second step is to extract additional knowledge needed to validate users' requirements as multidimensional. They used a multidimensional graph to store multidimensional information about the query. In their approach, the query is accepted if it generates a non-empty set of multidimensional schemas; otherwise, the query is rejected. The drawback of this approach is that there is no writing style for the query. Additionally, expressing users' requirements into SQL queries need an expert in this domain; moreover, the output model is a constellation schema that is more complicated than star schema.

An automated approach based on multidimensional design by examples (MDBE) has been proposed in (Romero and Abelló, 2010). Multidimensional conceptual schemas are derived from relational sources according to end users' requirements. Therefore, this framework composes of two steps: requirements formalization and the MDBE method. A graph oriented approach *GrHyMM* was proposed by (Tria et al. 2011); it follows a hybrid approach to generate a conceptual model. The graph is used for representing data sources, whereas the requirements analysis are tackled by *i** framework.

Summing up, researchers in the DWsing field considered the hybrid approach as important way to design a DW since the hybrid approach solves the problems found in both users' requirements driven approach and data source driven approach. However, the nature of DW projects requires merging data loading from different sources in one repository, the challenge here is that merging elements from different sources may yield heterogeneity problems. Using a semantic resource is a promising solution to solve this problem.

3.2.2 Ontology based approaches

Researchers in the DW design filed try always to find methods/concepts to clarify and facilitate the tasks of this process. Indeed, many researchers argue that using semantic resource will help DW designers as well as managers to solve the heterogeneity data types that intended by means of integrating internal or external data sources. Ontology as semantic resource used so far in various ways; i.e., some works combines it with data sources, whereas others used it to model the domain. Below, we review various works using ontology to gain valuable step towards DW designing process.

3.2.2.1 User requirements and ontology

These works start designing the DW schema from users' requirements, and they use ontology as semantic resource to overcome the heterogeneity issues. The study in (Nebot et al. 2009), follows the approach that combines DW and semantic Web technologies. They proposed a framework to design multidimensional analysis models and use it to build integrated ontology called Multidimensional Integrated Ontology (MIO). Another work following this approach is (Mullin and Motz, 2011), which introduces a methodology called OntoDW consists of four phases. This methodology build DW from domain ontology, hence, authors define rules to derive a conceptual multidimensional model and a logical model from the ontology.

So far, the above works use ontology and start designing DW from users' requirements; so, the heterogeneity problems may be solved. However, designing the DW based on users' requirements only have the main drawback that is some important data in DS may not be considered, hence, the output DW model will be incomplete.

3.2.2.2 Data source and ontology

The following works start designing the DW from the DS. Nevertheless, they use semantic resources to overcome the problem of heterogeneity that appears when integrating various data sources in one repository.

The study in (Nabli, Feki and Gargouri, 2009)tackles the notion of decisional ontology for the specification of analytical requirements. Authors, in this study, proposed ontology based method to standardize the multidimensional terminology. Their method consists of four phases: extraction of multidimensional concepts, comparison, upgrades the ontology with concepts and optimization of the ontology relations.

In (Romero and Abelló, 2010b), authors proposed AMDO framework, it deals with identifying the multidimensional knowledge contained in the sources. Its input is an ontology domain containing both data sources and the domain vocabulary. This framework aims to solve two problems related to the bottom up approach: namely, generating too many results and depending in relational OLTP DS. In one hand, the proposed solution for the first problem is using the concepts of filtering functions and searching patterns, whilst the normalization for the input logical schema is the solution for the second problem. Authors in (Romero and Abelló, 2007a)and (Romero and Abelló, 2007) aims to find the business multidimensional concepts from domain ontology. The domain ontology represents different and heterogeneous data sources. Authors in (Romero and Abelló, 2007a) try to avoid automating the DW design process from relational data sources only. The same authors in(Romero and Abelló, 2007a) proposed in (Romero and Abelló, 2007) approach that entails semantic web research area. These two works share a common feature, they employee ontology to avoid to extract multidimensional concepts from relational sources. In Tables 3.1 and 3.2 we compare these works based on our set of significant criteria.

The study in (Dai, Khare and Song, 2007)describes the SAMSTAR method where the authors develop a star schema by analyzing the semantics and structure in a given E/R Diagram and by applying a quantitative algorithm.

However, although these studies (data source based and requirements based) solved various problems related to heterogeneity by using ontology, their approaches suffer a little in means of well designing DW to support decision-making process. On the one hand, the main drawback of DS driven approach is that the obtained DW multidimensional schema may have some elements those do not meet the decision makers' needs (Romero 2009) (Nabli, Feki and Gargouri, 2009). Although some researchers try to filter these results, but

this will add additional cost and complexity to the process. On the other hand, the risk to miss some interesting concepts available in the data source represents a problem for users' requirements driven approaches.

Table 3.1 depicts the survey of the previous works based on the following criteria: i) Type of approach whether it is Hybrid, Top down or Bottom up; ii) Level of automation; iii) Usage of semantic resource, and iv) the input and the output of the design approach.

Table 3.1. Comparison of DW designing methodologies

Work reference	Type of approach	Automatic	Use of semantic resource	Input	Output
(Nabli, Feki and Gargouri, 2005)	Top Down	Yes	No	OLAP Requirements	DM Schemas
(Bargui, Feki and Ben-Abdallah, 2008)	Top Down	Semi	No	Analytical Requirements	Multidimensional Schemas
(Mazon 2005)	Top Down	No	No	Business goals	Multidimensional Schema
(Prat, Akoka and Comyn-wattiau, 2006)	Top Down	Semi	No	UML class diagram representing BR	Physical MOLAP platform
(Tryfona, Busborg and Christiansen, 1999)	Top Down	No	No	User requirements	StarER model
(Nebot *et al.*, 2009)	Top Down	No	Yes	Domain ontologies + Application ontologies	Semantic DW
(Nebot and Berlanga, 2010)	Top Down	Semi	Yes	Semantic data	facts
(Mullin and Motz, 2011)	Top Down	Semi	Yes	Information Requirements	DW Relational Design
(Inf, Böhnlein and Ende, 1999)	Bottom Up	Semi	No	Operational sources	Multidimensional data structures
(Carmè, Mazón and Rizzi, 2010)	Bottom Up	Manual	No	Relational data sources	Multidimensional schemata
(Hachaichi, Feki and Ben-Abdallah, 2010)	Bottom Up		No	Relational data sources+ XML Documents	DM Schemas
(Golfarelli, Maio and Rizzi, 1998)	Bottom Up	Semi	No	Conceptual or logical schemas (RDB)	Dimensional Fact model
(Jensen, Holmgren and Pedersen, 2004)	Bottom Up	Yes	No	Relational OLTP DB	Snowflake schema
(Nabli, Feki and Gargouri, 2009)	Bottom Up	Manual	Yes	Heterogeneous DS schemas	Decisional ontology
(Romero and Abelló, 2010b)	Bottom Up	Yes	Yes	Ontology domain (DS + domain vocabularies)	Constellation schema
(Romero and Abelló, 2007a)	Bottom Up	Semi	Yes	Ontology DS schema	Constellation schema
(Romero and Abelló, 2007b)	Bottom Up	Semi	Yes	Ontology domain	Dimension hierarchies
(Song, Khare and Dai, 2007)	Bottom Up	Semi	Yes	Entity relationship diagram	Star schemas

(Bonifati *et al.*, 2001)	Hybrid	Manual	No	Star schema from DS + star schema from BR	Star schemas
(Giorgini, Rizzi and Garzetti, 2008)	Hybrid	Semi	No	Organizational model + decisional model + DS schema	Fact schemas
(Tria, Lefons and Tangorra, 2012)	Hybrid	Semi	No	BR + DS	Fact schemas
(Mazón and Trujillo, 2009)	Hybrid	Yes	No	Conceptual MD from BR + DS schema	Implementation code for MD model
(Mazon, Trujillo and LechtenbÄorger, 2007)	Hybrid	Semi	No	MD model from BR + MD normal forms from DS	DW conceptual schema
(Romero and Abelló, 2006)	Hybrid	Yes	No	BR + the integrated logical model of the DS schemas	Constellation schema
(Romero and Abelló, 2010c)	Hybrid	Yes	No	Integrated DS + SQL queries	Constellation schema
(dell'Aquila *et al.*, 2010)	Hybrid	Semi	No	Sources schema + Requirements analysis	Conceptual schema

Summing up, the trend now in DWsing goes to enhance this process by using both the hybrid approach and the semantic resource.

3.2.2.3 Hybrid approaches using ontology

The second axis to organize this survey reveals how ontology was used in various researches that applied the hybrid approach. As ontologies may vary in different aspects, these may have: content, structure, implementation and a conceptual scope (Gali et al. 2004). As an example, the study in (Thenmozhi and K.Vivekanandan, 2012) transforms each DS into ontology format; to do so authors used *RDBtoOnto* tool to convert a relational DS into ontology; XML and text sources are converted by *JXML2OWL* and *OntoLT* tools respectively. The authors used the global ontology as input which has been built from local ontologies by *Protégé* tool; the requirements represented in logical format by logical converter so as to map it with global ontology; after that, they can extract automatically facts and dimensions. Once again, in (Thenmozhi and K.Vivekanandan, 2012) the authors propose an ontology based hybrid approach to derive a DW multidimensional schema. However, their suggested approach is based on transforming the DS model into ontology; from our viewpoint, this will reduce the generosity of their proposal, instead it is better to reuse a general ontology and involve the decision maker during the design process. This will make the DW design process more rapid and less costly.

In (Selma *et al.*, 2012), which is an extension for the work proposed in (Thenmozhi and K.Vivekanandan, 2012), user requirements and data sources are represented using ontology. It is worth mentioning that they used ontology here for analysis task; by merging capabilities, they defined global ontology; from this global ontology, they extracted the multidimensional elements (facts, dimensions, measures) using algorithms. After filtering the results issued from the algorithms, they match it with the information requirements to construct the conceptual model for the DW. Despite their work encompasses conceptual model and logical model for DW as well as algorithms for generating the multidimensional elements; they based their rules not in reverse engineering process. Therefore, the identification of facts and dimensions may be ambiguous.

The authors in (Khouri et al. 2012)defined an ontology based database making users' requirements consistent within the DW structure. They proposed a requirement model following a goal driven approach; the specification of requirements produces graphs which represent goals and their logical relationships in AND-OR graph format; the data sources in this study considered as ontology based database (OBDB) follow the architecture of OntoDB model. The Authors in (Khouri and Bellatreche, 2011) walked in a different direction to facilitate the DW design process; they proposed an ontology-based case tool called DWOBS which basically supposes the existence of a set of OBDBs representing integrated global ontology as well as a set of decisional requirements. The conceptual model of the DW generated by means of projection process encompasses decisional requirements and the global ontology. This work explains the design of the DW in its three levels: conceptual, logical and physical. However, to achieve this goal it uses many technologies and tools that may increase the complexity of this process. Additionally, there is no chance for the DW designer for intervene to confirm the resulting multidimensional model. In (Jiang *et al.*, 2011), authors proposed a hybrid approach based on ontology. Firstly, they develop ontology concepts based on the data dictionary of operational systems, then, from the E/R model, they extract the relationships of concepts and extended top down from the business model.

A hybrid approach is proposed in (Romero and Abelló, 2010b); the distinctive feature of this work is that an automatic analysis of the DS leads to the design of the DW schema.

Additionally, it belongs to the works that build the multidimensional model from ontologies. However, the design process of the DW consumes much effort and time because this approach relies on ontology as driving force for generating the multidimensional model. Moreover, the authors described their approach as a reengineering process; hence, the most choices in this situation as ontology language are UML (Unified Modeling Language) or E/R diagram. The problem here is that generating ontology from these data sources needs a heavy pre-process that increases the period of constructing the DW. Additionally, the full automation for generating the multidimensional model in this work decreases the chance for intervening the DW designer in confirming the generation of multidimensional elements.

A recent work (Di et al. 2015)adopts a hybrid approach using a semantic resource; the authors use their previous hybrid design methodology (GrHyMM). They produce fact schemas by considering at the same time both the DS and the user needs. In behave of requirements analysis, they adopts *i** methodology; by this methodology, the authors use the information requirements to define workload which contains the analytical queries that the decision makers need to achieve. On the side of DS, all the DS schemas should be analyzed and reconciled, the aim behind this, is to obtain global conceptual schema, this one transforms into a relational schema that represents the input to the conceptual design. The integration process depends on ontology approach. Finally, their approach relies on graph based multidimensional model for generating the multidimensional schema in forms of cubes. Although the authors success in applying the basic factors to construct a modern based DW, their work encompasses many limitations: first, there is no elicitation phase regarding the user requirements; second, they represent their multidimensional model as graph rather than star schemas; third, they build domain ontology which increases the time and support the complexity of designing the DW.

Authors in (Romero, Simitsis and Abelló, 2011)proposed a novel framework called *GEM*. The inputs to this framework are a set of user requirements and information about the operational sources. The output is twofold: multidimensional model and the conceptual ETL flow. OWL ontology used here to capture the semantics of the data sources. Although

this work is satisfy the use of semantic resource and the hybrid approach, its drawback is the cost of mapping the concepts in the operational data source into WOL ontology.

From the previous review, it is clear that research in DW design process goes towards using the hybrid approach to deal with both data source and requirements. Additionally, authors try to use ontology as a helpful tool for resolving the heterogeneity issues shaping the activities during the design process. Hence, it is as agreement between the researchers in this field that using hybrid approach and ontology considered as important factors for designing success multidimensional model. However, first, few works in the literature use hybrid approach with semantic resource; second, the few proposed works vary in their use of those important factors. This situation draw again the real fact that there is not yet standard method designer can follow to achieve this complex process. In order to reflect this variation and explore the shortage in this area, we made a comparison between our approach and the works applied these factors. We choose distinctive criteria for our comparison, these encompasses:

- The techniques for reconciling data source and requirements.
- The degree of automation.
- The first consideration whether it is the requirements or data sources.
- The existence of star schemas as multidimensional model.

Table 3.2 summarized the comparison between the proposed works and our work.

Table 3.2. Comparison between proposed hybrid approaches and our work

The work	**Technique**	**First consideration**	**The automation**	**Semantic resource**	**Star schemas**
(Thenmozhi and Vivekanandan, 2012)	Algorithms	User requirements	Yes	Yes	No
(Selma *et al.*, 2012)	graph	Parallel	Yes	Yes	No

(Khouri and Bellatreche, 2011)	graph	Parallel	Yes	Yes	No
(Jiang *et al.*, 2011)	Algorithms	Requirements Model	Yes	Yes	No
(Romero and Abelló, 2010b)	Search patterns	Data sources	Yes	Yes	No
(Di, Lefons and Tangorra, 2015)	graph	User requirements	Yes	Yes	No
(Romero, Simitsis and Abelló, 2011)	Algorithms	User requirements	Semi	Yes	No
Our work	Star schema matching	Parallel	Semi-automatic	Yes	Yes

Indeed, using hybrid approach in designing DW cannot considered as atomic activity. It may compose of many stages researchers should follow to be achieved. One important stage is the matching between the schemas generated from data source and schemas build from business requirements. The importance of this stage encourage us to shed light on the works applied it.

3.3The matching process

As mentioned above, the researchers in DW area agree about designing DW by using hybrid approach and ontology for solving the heterogeneity issues. However, using hybrid approach needs a matching process between the schemas generated from the DS and the schemas in behave of users' requirements. The matching process as stage in DW

inherits the difficulty shaping the nature of designing the DW. Indeed, these difficulties do not came from nonsense, as each step in this construction process consists of sub-steps, which can be carried out using various methods and/or techniques. Additionally, these steps should coined together in a systematic manner to produce a satisfying multidimensional model (i.e., schema). To shed light in this complexity, as an example, we cite the matching process between the DS and users' requirements; for this purpose, in (Romero and Abelló, 2006)(Di, Lefons and Tangorra, 2015)the authors base their work on a graph technique. In (Romero and Abelló, 2010a)the authors use search patterns, whereas (Mazon, Trujillo and LechtenbÄorger, 2007) uses three multidimensional normal forms to define a set of Query/View/Transformation (QVT) relations for accomplishing the agreement between the multidimensional model obtained from user requirementsand the DS.

In not far away for the matching process, the proposed works vary in whether they first consider users' requirements or DS structure (i.e., schema). As an example, in (Di, Lefons and Tangorra, 2015)(BONIFATI et al. 2001)the authors first consider requirements then reconciling them against the DS; other works begin with considering first the DS and then the requirements (Romero and Abelló, 2010a). Additionally, the solutions vary in the way the semantic resource had been applied; in this trend, the authors in (Romero and Abelló, 2010a) developed a domain ontology in order to automate the generation of star schemas; authors in (Thenmozhi and K.Vivekanandan, 2012) use a global ontology. On behave of automating the DW construction process; existing contributions vary between generating the star schema manually or through a semi-automatic approach, while others try to automate the full process beginning from generating multidimensional models from users' requirements and models from the DS as well as the matching process that produces final star schemas.

Summing up, the DW development process needs additional investigation to fill this gap. Table 3.3 summarizes the techniques used in hybrid approaches for the purposes of reconciling the DS with users' requirements, the degree of automation, and the use of a semantic resource.

Table 3.3 Comparison between proposed hybrid approaches.

Work reference	Matching Technique	Automation	Use of semantic resource
(Romero and Abelló, 2006)	Graph	Automatic	No
(Romero and Abelló, 2010a)	Search patterns	Automatic	Yes
(Mazon, Trujillo and LechtenbÄorger, 2007)	Multidimensional normal forms	No	No
(Romero and Abelló, 2010c)	Graph/SQL queries	Automatic	No
(Thenmozhi and Vivekanandan, 2012)	Algorithms	Automatic	Yes
(Di, Lefons and Tangorra, 2015)	Graph	Automatic	Yes
(Bonifati *et al.*, 2001)	Graph and Goal/Question metrics	No	No

3.4 Conclusion

This chapter tackled the previous studies in DW design research area. Two axes were explored in details: in the first one, issues related to the DW design have been described. In the second axis, ontologies in data warehousing were tackled. From these two axes a comprehensive comparison has been conducted.

In the next chapter, the process of constructing Star schemas from data source DS will be considered.

Chapter IV

Construct star schemas from relational data sources

4.1 Introduction

This chapter explains all the tools and concepts we have used to construct the star schemas from relational data sources. Indeed, generating multidimensional model from the data sources is not an optional task. This will not make a conflict with the agreement fact that the DW should be constructed using hybrid approach. By using the hybrid approach, designers should not bypass or ignore the data sources in organization. Recall that, DW design process is a complex task, this hard fact appears whenever we talk about any point along the phases of this process. When we say generating multidimensional model from the data sources, this expression is not atomic. Much questions arise here, i.e. what is the format of the sources that this data in? Is the generation process automatic or semi automatic or manual? Which method/s the designer should follow to generate this model? Is there a means of semantic resource or not?.Additionally, each of these questions can yield many other questions; really it is complex task.

4.2 Phases of our approach

In this chapter, we lay the ground our first model "heuristic based approach for automating star schemas construction from a relational data source". We construct star schemas from a relational data source using a heuristic based approach. From our viewpoint, the DW designer should be involved in this long complex process, this is why our approach is semi-automatic; our philosophy as many other researchers not to use a full automation process, because the confirmation of the DW designer is helpful for correctly generating the multidimensional model.

Let us remember that our objective is to develop a hybrid, semi-automatic approach for DW construction as shown in figure 4.1.

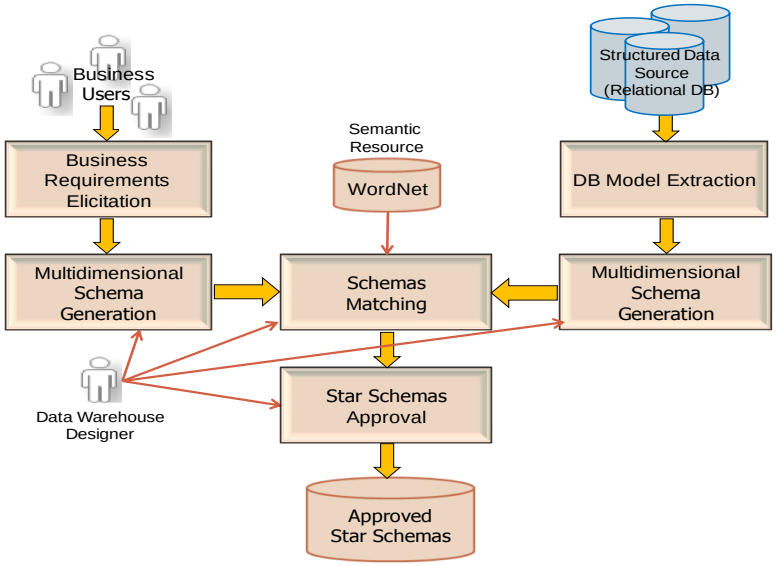

Figure 4.1. Hybrid,Semi Automatic Approach for the Design of Multidimensional Schemas

In this chapter, we generate star schemas from a relational database; to come up with this objective, we propose a novel approach (figure 4.2) with some specifics for each of its three phases, namely: i) *Database model extraction*, ii) *Reverse engineering process*, and iii) *Multidimensional schema generation.*

Hereafter we explain the three phases of the proposed approach.

4.2.1. Database schema extraction

We extract the tables' structures of the relational database (DB) source from the repository of the Relational Database Management System (RDBMS) by querying system views. For each table we get the name, the name and type of each of its columns, and, in addition, the primary key and foreign key constraints as they are vital for the next steps; in fact these constraints first help us to classify the transactional database tables into three classes namely: *Entity-table*, *Relationship table and Weak entity-table***.** Secondly, they will be very useful to trace the links between tables in order to construct dimensional hierarchies.

4.2.2. Reverse engineering on the relational database schema

The aim behind the reverse engineering process is to return the DB table in the relational schema to its initial state (strong entity or weak entity or relationship). The reverse engineering process is a distinctive feature of our work as it enables us to know which DB tables were initially entities of the real world, and which ones were relationships. Indeed, to identify facts and dimensions, we should know which tables in the schema are suitable to be candidate for representing facts and which ones could play the role of dimensions. As in data warehousing entities are typically used to design dimensions whereas relationships are for building facts(Golfarelli, Maio and Rizzi, 1998)(Cabibbo and Torlone, 1998). For this phase, we have defined a set of three heuristics rules: R1 for entities identification, R2 for relationship, and R3 for weak entity.

The involvement of the DW designer in reverse engineering phase is important to approve the correctness of the classification.

4.2.3. Multidimensional schema generation

This task aims to build multidimensional (MD) schemas from the classified tables in the previous phase. Defining a set of appropriate heuristics is required for the automation of this task. These heuristics are to find out automatically the MD schemas elements (facts, dimensions, measures, etc.).

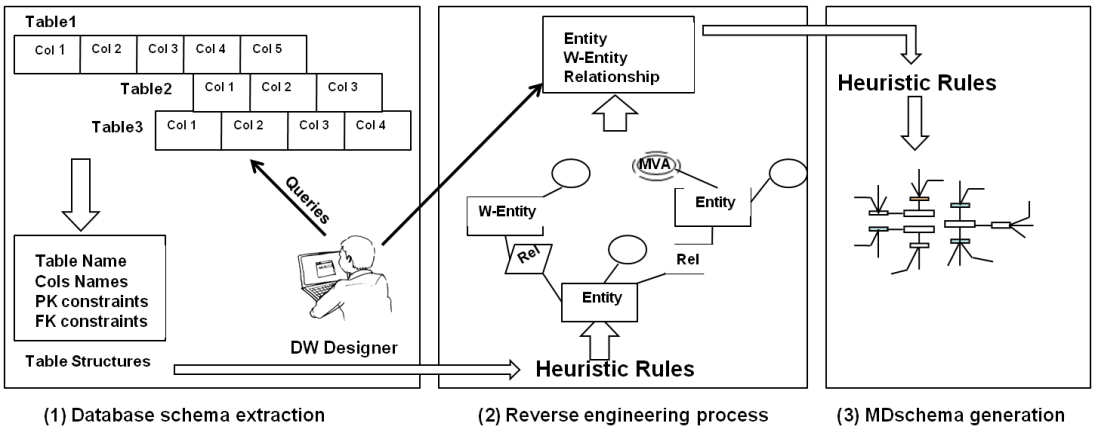

Figure 4.2. Heuristic Based Approach for Automating Star Schemas Construction.

4.3 Heuristics rules for the reverse engineering process

Let us point out that in data warehousing literature, approaches starting from ER diagram build facts from relationships whereas dimensions are mainly built from entities (Golfarelli and Rizzi, 2009b). In our framework, we give this task a great attention, as it is a basic step to identify facts and dimensions. In order to identify entities and relationships within a relational data source, the literature works define rules for the reverse engineering process. Nevertheless, these works classify the schema tables into two categories only.

In order to improve the quality of the result; i.e., more accurate star schemas produced by our approach, we classify the relational DB tables into three categories: i) Strong entity ii) Relationship iii) and Weak entity. Our rules adopted from (Elmasri & Navathe 2010) for identifying these three categories are as follows:

R1. Identification of strong entities

- *Every table in the schema having a single-attribute primary key is a candidate Entity.*

Figure 4.3 shows example of strong entity table.

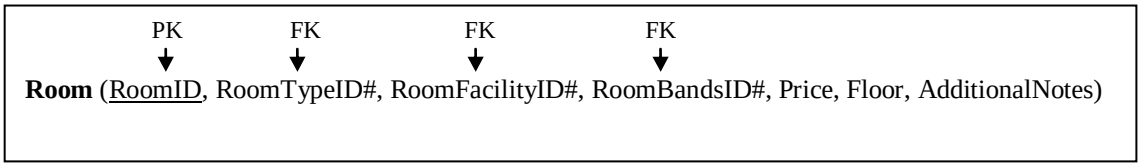

Figure 4.3. Example of Relational Table Modeling Strong Entity

R2) Identification of relationships:

- *Every table in the schema satisfying the two conditions below is candidate for a relationship:*
- *A primarykey composed of foreign-key attributes, and*
- *The number of foreign keys within the primary key >1.*

Figure 4.4 is an example of relational table which its primary key is composed of two foreign keys referring two entities (Customer and Concert). This table satisfies the conditions of Rule R_2; consequently, it is classified as a relationship.

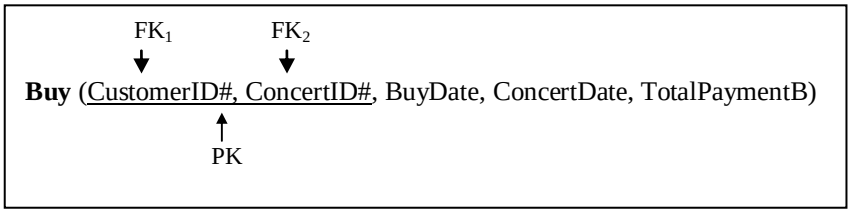

Figure 4.4.Example of Table that Model a Relationship

R3. Identification of weak entities:

Weak Entity is an entity type that does not have key attributes of its own (Elmasri & Navathe, 2010), the weak entity has an attribute that partially identified the Entity, this attribute called the partial key; as an example, the *EntityName* attribute can represent a partial key in a table because it did not distinguish all instances of the table. The rule for identifying the weak entity is as follows:

Every table in the schema satisfying the three conditions below is a candidate for a weak entity table:

- *Its primary key composed of more than one attribute, one of them is a partial key, and*
- *the number of its primary key attributes is greater than the number of foreign key attributes in the PK, and*
- *Has only one foreign key attribute.*

Figure 4.5 explain the weak entity *dependent*; note that, in this table, we can find two dependents with the same name, so the key of this table partially identifies the instances of the table.

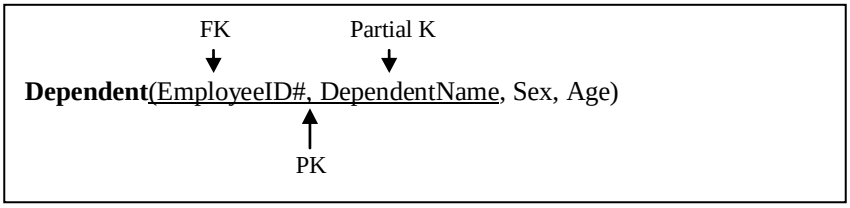

Figure 4.5. Example of Weak Entity.

We have applied these rules for the Booking schema (figure 4.6) issued from(Hachaichi, Feki and Ben-Abdallah, 2010); the initial results show that our rules classified Room as an entity, whereas Room had been classified as entity and fact at the same time in (Hachaichi, Feki and Ben-Abdallah, 2010). The similarity between our works and that in (Hachaichi, Feki and Ben-Abdallah, 2010) represented by classifying the tables into two categories: Entity and relationship. The justification of this similarity is that the booking schema in (Hachaichi, Feki and Ben-Abdallah, 2010) does not have a weak entity.

Room (RoomID, RoomTypeID#, RoomFacilityID#, RoomBandsID#, Price, Floor, AdditionalNotes)
RoomTypes (RoomTypeID, TypeDesc)
RoomFacilities (RoomFacilityID, FacilityDesc)
RoomBands (RoomBandsID, BandDesc)
Payments (PaymentID, CustmerID#, PaymentMethodID#, PaymentAmount, paymentComments)
PaymentMethods (PaymentMethodID, PaymentMethod)
County (CountyID, CountyName)
Bookings (CustomerID#, DateBookingMade, TimeBookingMade, RoomID#, BookedStartDate, BookedEndDate, TotalPaymentDueDate, TotalPaymentDueAmount, BookingComments)
Customer (CustomerID, CustomerForeNames, CustomerSurNames, CustomerDOB, CustomerHomePhone, CustomerWorkPhone, CustomerMobilePhone, CustomerEmail, CityID#)
State (StateID, StateName, CountyID#)
City (CityID, CityName, StateID#)
Singer (SingerID, SingerForeNames, SingerSurNames)
Concert (ConcertID, ConcertName, CityID#, SingerID#)
Buy (CustomerID#, ConcertID#, BuyDate, ConcertDate, TotalPaymentB)

Figure 4.6. The Schema of Hotel's Booking Database (Hachaichi, Feki and Ben-Abdallah, 2010)

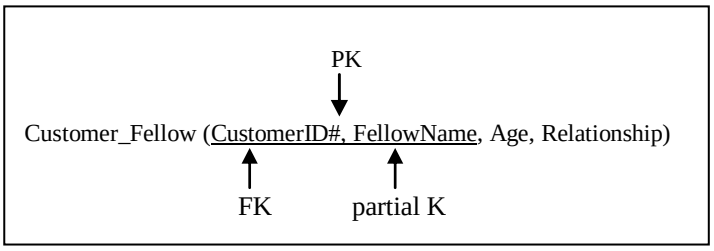

Figure 4.7. Our Proposed Additional Table to the Schema in (hachaichi, feki and ben-abdallah, 2010)

In order to apply all our rules and reflect their accuracy, we add a table to the DB schema given in figure 4.6; this table is *Customer_Fellow* (described in figure 4.7); this table represents a weak entity table. After the involvement of this new table, our rules classify the schema tables into three categories: strong entity, relationship and weak entity.

These results show the importance of the concept Weak-entity. Thanks to this, we were able to identify precisely the relationship table which is a potential fact (R2); this precision was not possible without these concepts in other approaches (as example if we based our rules on *m: 1* relationship, we can classify a weak entity table as a relationship).
Table 4.1 shows the data source tables in figure 4.6 and figure 4.7 classified into Entity tables, relationship tables, weak entity tables as well as the rules applied each time.

Table 4.1 Classified tables of figure 4.6 and figure 4.7.

Relational tables	Class	Classification Rule
Room	Entity	R1
Payments	Entity	R1
RoomBands	Entity	R1
RoomFacilities	Entity	R1
PaymentMethods	Entity	R1
RoomTypes	Entity	R1
Customer	Entity	R1
County	Entity	R1
State	Entity	R1
City	Entity	R1
Singer	Entity	R1
Concert	Entity	R1
Buy	Relationship	R2
Bookings	Relationship	R2
Customer_Fellow	Weak Entity	R3

Identification of facts

Facts are built from relationship tables(Hachaichi, Feki and Ben-Abdallah, 2010)(Golfarelli, Maio and Rizzi, 1998) identified in the previous step. In addition, some Entity-tables may be suitable to create facts. Mainly this occurs when it has non-key numeric attribute not included in the primary key. We define the following two rules for fact extraction:

R4. *Any relationship-Table issued from the reverse engineering process (by rule R2) is a candidate to be a fact.*

R5. *Any Entity-Table respecting the following conditions is a candidate to be a fact:*

- *Has a single attribute primary key, and*
- *Has more than one foreign key ,and*
- *Has non-key numeric attributes (so that the number of all numeric attributes in the table >3), and*
- *Does not have its primary key attribute as foreign key in one of the defined relationship (R2).*

The justification of the constraint "*the number of all numeric attributes in the table >3*" is as follow: one attribute as primary key; at least two attributes as foreign keys; and one attribute as non key attribute.

So far, we can extract the candidate facts using rules R4 and R5. Basically, fact represents the start point for generating the remaining elements of the star schema (measures and dimensions). So, we gave more attention to the definition of rules to extract facts as the other star schema elements depend on the generated facts. The following rules reveal the process of identifying measures and dimensions.

Identification of Measures

Generally, measures are numeric attributes of the tables representing facts. We extract measures using rule R6.

R6. *For a given fact-table T* (i.e., table elected as a fact), *the measures are the set of numeric attributes issued from T Minus the set of attributes representing primary key or foreign key of T.*

Table 4.2 shows measures extracted from the fact tables of our running example.

Table 4.2. Extracted Measures for each Fact

Fact	Measure
Bookings	TotalPaymentDueAmount
Buy	TotalPaymentB
Payments	PaymentAmount

Identification of Dimensions

In data warehousing the *Dimension* concept represents the axis of analysis (Adamson, 2010); rationally, dimension can be any table that its primary key attribute participates as a foreign key in a *fact-table*. We propose the following rules to extract dimensions.

R7. Each table referred by a foreign key in a fact-table F extracted by the rules (R4, R5) will be a candidate dimension for F.

In data warehousing, decision-makers analyze the evolution of their business data through time; consequently, any data recorded in the DW must be related to the Date/Time dimension. For example, a sale is realized at a given date. As the Date/Time exists in the transactional system as an attribute (e.g., sale date), we define rule *R8* for building the Date dimension.

R8. *A Date or Time attribute in a table F transformed into a fact will be a candidate dimension for F.*

Table 4.3 shows each fact of our running example, the set of its extracted dimensions, as well, the rule used to extract each dimension.

Table 4.3. Facts and Their Identified Dimensions

Fact	Dimensions	Rule
Bookings	Customer	R8
	Room	R8
	DateBookingMade	R9
	TimeBookingMade	R9
	BookedStsrtDate	R9
	BookedEndDate	R9
	TotalPaymentDueDate	R9
Buy	Customer	R8
	Concert	R8
	BuyDate	R9
Payments	Customer	R8
	PaymentMethods	R8

Parameters and hierarchies

In multidimensional modeling, the attributes of a dimension are organized into hierarchies; each hierarchy is composed of a number of parameters(Jensen, Pedersen and Thomsen, 2010). Within a dimension, some attributes could not be considered as parameters; they describe level attributes and then called weak attributes. As well, the temporal attributes (Date and Time) in our result are organized from the lowest granularity (day) to the highest granularity (year). Table 4.4 shows the dimensions and their identified parameters.

Each hierarchy is built in levels as follows:

Step one: level one in the hierarchy represent the identifier of the dimension (Hachaichi, Feki and Ben-Abdallah, 2010)(Adamson, 2010).

Step two: for each dimension, level two represent the primary key attribute of a table in the schema referenced in the dimension table as foreign key.

- *Step three*: level three extracted recursively by applying step two onto other tables in the schema.

Figure 4.8 depicts the Customer dimension of our running example.

Table 4.4. Dimensions and The Identified Parameters for each Dimension

Dimension	Parameters
Customer	City
	State
	County
Concert	City
	State
	County
	Concert-Date
Room	Price
PaymentMethod	PM-Name
Date	Day
	Month
	Year

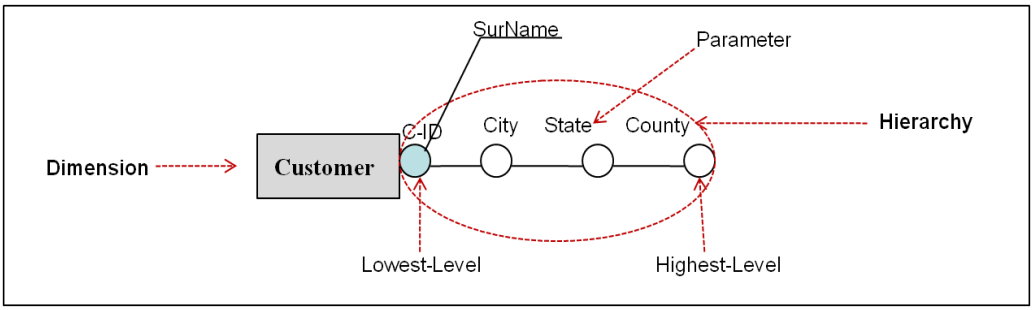

Figure 4.8. Customer Dimension with its Parameters Organized in Hierarchy

4.4 Conclusion

In this chapter, the approach for constructing star schemas from relational data source has been introduced. The phases of this approach are described; as well, the heuristic rules that defined for the reverse engineering process have been explored. Additionally, the approach for defining rules for extracting the elements of the star schema (facts, measures and dimensions) has been illustrated. The next chapter reveals the construction of star schemas from business requirements.

Chapter V

Generating star schemas from business requirements

5.1 Introduction

In this chapter we interest in introducing the other side to apply our hybrid approach. Particularly, our aim is to generate star schemas from business requirements. Researchers argue that many ambiguities related to the design of DW are caused by misunderstanding of business requirements; as yet there is no agreement/approach both in academic and professional organizations that can be followed to fulfill this task. Studies show that 80% of DW projects failed due to misunderstanding of user's requirements to fulfill business goals (Malinowski and Zim´anyi, 2008)(Schiefer, List and Bruckner, 2002).Before we dive in details for designing our model from the business requirements, let's give a conscious introduction about the difference between requirements in conventional systems and requirements in decisional systems.

As there is a main difference between conventional operational systems and DW/decisional systems in both project lifecycles (Rilston *et al.*, 2002), user requirements phase is as well concerned by this difference. One side for this difference is represented by the role of users in each system; in operational systems; users are interested in the transactional daily operations, whereas users in DW include senior executives, Key departmental managers and Business analysts (Ponniah, 2001).

To analyze the requirements for the business, the system analyst needs to concentrate on what information the business needs, not on how to provide information (Ponniah, 2001).

In the requirements engineering concepts, the requirements can be categorized into two types: functional requirements and nonfunctional requirements (Salinesi and Gam, 2007)(Rainardi, 2008). The difference between those two concepts can be observed from the meaning of the two words: *What* and *How*. Namely, the functional requirements define *what* information the DW is expected to provide, in other words, it describes the core

functionality of the computer application. While the non-functional requirements determine *how* this information should be provided to be used correctly. In the literature, the requirements specification phase and the conceptual design phase are separated from each other (Vaisman and Zimányi, 2014). However, some researchers argue that these phases overlap (Ballard *et al.*, 1998).

In fact, the process of dealing with requirements in DW projects seems to be more complex. On one hand, the main reason for this complexity is the changing nature of requirements. On the other hand, other reasons contribute to this complexity are that user requirements may be incomplete or inaccurate; and/or users are often unable to clearly formulate their particular requirements. Additionally, business users have not a shared understanding of the business goals (Golfarelli, 2010).In order to help DW designers, we introduce a method to generate star schemas from business users' requirements. The distinctive criteria of our work are representedby the upcoming features: first, we give the requirements elicitation phase its importance as a critical activity to the requirement analysis phase. Second, in order to optimize the elicited requirements, we propose to use a matrix of requirements, to our knowledge, few works use such a matrix. Third, we automate the process of generating the star schemas from the matrix of requirements.

5.2 Phases of our approach

The process of generating star schemas from business requirements in our proposed method encompasses three steps as shown in figure 5.1: i) Business requirements elicitation; ii) requirements normalization; and iii) generation of multidimensional schemas.

5.2.1 Business requirements elicitation

Requirements elicitation is a critical task in systems design process. It represents the most serious phase of system development and the hardest to repair its effects when conducted with lacks(Rajagopal *et al.*, 2005). Requirements elicitation can be defined as "*a process through which the customers, buyers or users of a software system discover, reveal, articulate and understand their requirements*" (Raghavan, Zelesnik and Ford, 1994). During this step, stakeholders should avoid many difficulties associated with it as it is a social interaction activity. The aim of the elicitation process is to define the business

requirements that lead objectively to understand the business processes. To elicit decision makers' requirements, these requirements can be expressed by natural language-like. In our method, the business users describe their needs/goals by a means of queries since queries are easy to formulate.

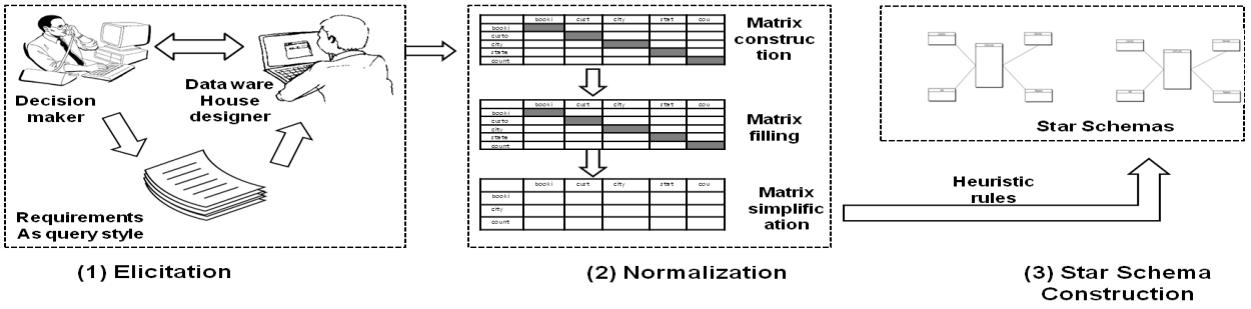

Figure 5.1.Construction of DW Star Schemas from Business Requirements

The requirements used to generate the star schemas should satisfy the following conditions:

- *Completeness*. The requirements should be complete. If the analyst misses necessary design information or misunderstands any process, the resulting DW will not give the decision makers the ability to define their decisions correctly. In our approach, as in figure 5.1, the requirements elicitation is a continuous process between the decision maker and the DW designer. So we can guarantee the involvement of most of the business requirements.
- *Simplicity*. The requirements expression should be simple. To satisfy this demand, many authors formulate their requirements in a natural language like; this gives flexibility for decision makers to be more natural. In our work, we use a natural language template in order to help users and enhance the process simplicity.
- *Correctness.* The requirements should precisely describe the business nature. This reflects the role of semantic resource to distinguish between various concepts/words related to the business.

The requirements elicited from different users levels should represent in a unified format, so the use of query-style to represent the requirements is helpful for system analyst; this because different users can express the same requirements with different words. Figure 5.2 depicts the formal template for a decisional query.

Analyze <process > **By** <parameter>

a) Simple query template

Analyze<process>**By**<parameter1 [and parameter2 and ….] **for** parameterN>

b) Compound query template

Figure 5.2. Two Query Templates

The important words in the templates (figure 5.2.a and figure5.2.b) are process and parameter. Note that the keyword Analyze specifies the business activity to analyze (i.e., the *process* means the fact), and the *By* introduces the analyzing criteria (parameter) which denote dimensions or levels.

In our case study "bookings system" (Hachaichi, Feki and Ben-Abdallah, 2010), figure 4.6, which is an integrated relational schema from e-Ticket schema and hotel rooms booking schema, the decision makers can list these needs/goals as objectives for their business. Figure 5.6 shows our case study "bookings system. Table 5.1 depicts examples of the goals representing the system processes.

Table 5.1**.** Sample Elicitation of Business Processes Goals

Queries of the Booking process	Queries of the *Buy* process
Q1) **Analyze**Booking*s* **By**City for Month a price and year 2010	**Q11)** **Analyze** Buy **By** Customer and City for City YYY
Q2) **Analyze** Bookings **By** Room_type For Room of type AAA	**Q12)** **Analyze** Buy **By** Concert and Telephone for State DDD
Q3) **Analyze** Bookings **By** Customer for State RRR	**Q13)** **Analyze** Buy **By** Month and Hour for City YYY
Q4) **Analyze** Bookings **By** Customer for Year 2010	**Q14)** **Analyze** Buy **By** Concert for County RRR
Q5) **Analyze** Bookings **By** Room facility for Price >100$	**Q15)** **Analyze** Buy **By** City for Singer John
Q6) **Analyze** Bookings **By** State and Year	**Q16)** **Analyze** Buy **By** City and Year
Q7) **Analyze** Bookings **By** Room_bands for City YYY	**Q17)** **Analyze** Buy **By** State and Price

Q8) **Analyze** Bookings **By** Room_type for City YYY	**Queries of the *Payments* process**
Q9) **Analyze** Bookings **By** Hour and Minutes	**Q18)** **Analyze**Payments **By**Customer and City for Year 2010.
Q10) **Analyze** Bookings **By** Month and Day and City	**Q19)** **Analyze** Payments **By** PaymentMethod and Price

The requirements elicitation process involves various levels of business users (e.g., Sponsors, Executive managers, etc). Each of those users may use a different word for the same concept. Table 5.2 and table 5.3 shows examples for this redundancy.

Table 5.2. Goals Elicited from Different Users.

Business user	Requirement/ Goal
Sponsor	*Analyze the amount of bookings by **customer** and room.*
Decision maker	*Analyze the amount of bookings by **subscriber** and room.*
Executive manager	*Analyze the amount of bookingsby **client** and room.*

Table 5.3 Goals Elicited from Different Users (Bookings Synonym).

Business user	Requirement / Goal
Sponsor	*Analyze the amount of **reservations** by customer and price.*
Decision maker	*Analyze the amount of **bookings** by customer and price.*

So, it is clear that, the elicited requirements inherit the redundancy. In table 5.2, the three different words *customer*, *subscriber* and *client* are synonyms and therefore point to the same concept of the business domain (client means a customer or someone who receive services). Additionally, as in Table 5.3, the booking process may be referred to as a reservation process. In order to alleviate such issues, we use a matrix based solution for normalizing the elicited requirements.

5.2.2 Business requirements normalization

The organization's users (DW Users, Customers, Business Managers, etc) and the business team (Executive Management, Sponsors/Funders) explain their needs in a way that it may not be complete and/or correct; so, the DW designer try to normalize these requirements to meet the three criteria we previously defined: completeness, simplicity and correctness. Thus requirements will be detailed, testable and complete(Raghavan, Zelesnik and Ford, 1994).

The main function of the requirements normalization process is to represent the gathered requirements according to a given format that facilitates avoiding the redundancy in the set of the elicited business requirements, the extraction of the multidimensional elements and then the star schema derivation. To do so, we use a *matrix of requirements* which is a square matrix for representing requirements. During the normalization phase, the DW designer may modify/correct requirements structures to minimize data redundancy. The result should be a non-redundant set of well-defined requirements.

The normalization of the requirements process splits into three steps: i) Matrix construction ii) Matrix filling and iii) Matrix simplification.

Firstly, we define the matrix of requirements;

Definition Matrix of requirements:

The matrix of requirements is a square matrix; more precisely, it is an identical diagonally matrix. The rows and the columns of the matrix represented by the processes and the parameters elicited from the user's needs. First, we need to enter the processes and the parameters in the matrix; we name this process "matrix construction".

5.2.2.1. Matrix Construction

The processes names in the requirements are put in first rows and columns as depicted in Figure 5.3; the names of parameters for each process are inserted to columns at the right of the process name and in rows under the process names. Nevertheless, how the DW designer correctly differentiates between which words are processes and which words

are parameters? To help resolve this ambiguity, we define two heuristic rules for identifying processes and parameters.

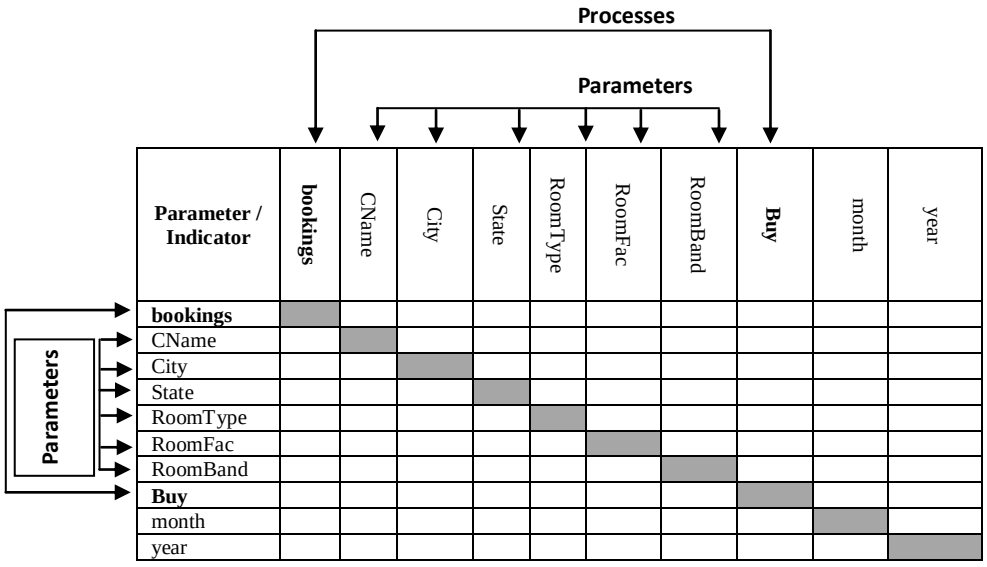

Figure 5.3. Constructed Matrix of Requirements

Heuristics rules for determining processes and parameters:

R1*: Any word (w) in the query (q) located after the word* ***analyze*** *will be considered as candidate to be a* ***process*** *(p).*

R2*: Any word (w) in the query (q) located after the word* ***by*** *or* ***and*** *will be considered as candidate to be a* ***parameter*** *(pm).*

Algorithm1 describes the steps of entering processes and parameters in the matrix of requirements as well as the process of matrix filling. Note that, important question may be asked here: how do we enter processes with synonyms names (table 5.3); recall that, we solve this by elaborating the normalization phase to avoid such redundancy.

We applied algorithm1 for the business process goals in table 5.1. Figure 7.11 depicts the result as constructed matrix.

Algorithm1: Fill Requirements Matrix.
Aims: Matrix filling with names of processes and parameters.
Input:
Rm: Requirements Matrix.
Q: Set of n queries $\{q_1, q_2,\ldots,q_n\}$ written according to templates in Figure 2.
W: Set of m words $\{w_1, w_2,\ldots,w_m\}$in each query.

```
Output: Rm Filled matrix.
Begin
    Str array of int;
    String s,    token = "  ";
    int counter=0, token_counter=0, indexToken=0, c=0; k=0; j=0;
    int RowSize, ColumnSize;
    Begin    // matrix construction phase
        Foreach query q in Q do
            Foreach word q.w do
              if(w satisfies R1) //w is a process
                  Rm[k,j+1] =Rm[j+1,k]) = w
             else   if(w satisfies R2) //w is a parameter
                  Rm[k,j+2] =Rm[j+2,k]) = w
                  Increment j;
                  end if
              endif
           End Foreach
       End Foreach
    End
    Begin      // matrix filling phase
       token ← next token in s          //s contains the query
       indexToken ← 1
       if(indexToken>= 1)
          if(indexToken equals 1)      // the token is process
             c← index of token in Rm[r,0];
          endif
          For(int r = 0; r <= RowSize; r++)
               For(int cl=0;cl<= ColumnSize; cl++)
                  If(token equals  Rm[r,0] )
                       Token_counter ← 2    // token is parameter
                       While(s has more elements)
                       token  ← next token in s
                       Token_counter++;
                  else      if(token equals  Rm[0,cl] )
                       Rm [c, cl] ← "√";
                        Endif
                  Endif
              Endfor
          Endfor
       endif
    End
End
```

Matrix Filling:

Matrix filling is the process of filling the constructed matrix with data according to Algorithm1.

Simplify the matrix:

The simplification process is automatic; i.e., any row or/and column that has not at least one sign (√) will be deleted. To come up with this process we define rules R3 and R4.

Heuristics rules for simplifying the constructed matrix:

The following rules explain the process of simplifying the constructed matrix:

R3: *any cell along any column in the constructed matrix not filled with the sign (√) will be deleted.*

R4: *any cell along any row in the constructed matrix not filled with the sign (√) will be deleted.*

Namely, the DW designer deletes from the rows all the parameters and deletes from the columns all the processes. This simplification process helps the DW designer determine the Multidimensional elements. Figure 5.4 depicts the representation matrix after the simplification phase for our running example. The columns' headers represent parameters whereas the rows' headers represent the processes.

	City	State	County	Customer	RoomType	RoomFacility	RoomBands	Price	Month	Day	Hour	Concert
Bookings	√	√		√	√	√	√	√	√	√	√	√
Buy	√	√	√					√	√	√	√	√
Payment	√			√				√				

Figure 5.4. The Representation Matrix after the Simplification Phase

5.3 Multidimensional schema generation

This phase defines the generation of multidimensional schema elements. From the simplified matrix of figure 5.4, the DW designer can define the facts, dimensions and

hierarchies in three steps, in order to automate these steps; it is helpful to define heuristic rules for extraction of each element:

Heuristics rules for generating multidimensional elements:

Definition of facts:

The facts elements can be generated from the representation matrix after the simplification phase; as researchers argue that the processes in the business requirements represented in the DW as facts (Bargui, Feki and Ben-Abdallah, 2008)(Bargui et al. 2009), our rule for defining facts are as follow:

R5: *each row's header in the simplified representation matrix is a candidate for fact*.

The facts represent the rows' headers; recall that, as we described in the simplification phase, that the DW designer deletes from the rows all the parameters; the first cell in each row (row's header) that remains will represent the processes, so now, the DW designer can define all the facts from these processes in the rows' headers. Table 5.4 shows the generated facts for our running example.

Table 5.4. The Generated Facts

Fact name	Rule used to extract the fact
Bookings	R1
Buy	R1
Payment	R1

Definition of Dimensions:

The columns' headers in the simplified matrix represent the parameters; to define the dimensions, the DW designer will group the related parameters (conceptually) in one dimension, i.e., the parameters (day, month, year) will represent the *date dimension*. Consequently, the DW designer can define along each row (fact) the column' header that

has a sign (√) and the (logically) related parameters in these columns will represent the dimension for that fact; sometimes one parameter can represent dimension, i.e., Price. Our rules for defining parameters and dimensions as follows:

R6: *each column's header in the simplified representation matrix does not belong to a set of logically related parameters is a candidate for dimension.*

R7: *along each row which represents fact, if at least one parameter from the set of logically related parameters has the sign (√), this implies that the set of logically related parameters is a dimension for that fact.*

Table 5.5 shows the result of generating each dimension and its parameters.

Table 5.5. The Generated Dimensions with Their Parameters

The dimension	The parameters	Rule
Date	Month	R7
	Day	R7
	Hour	R7
Customer	County	R7
	State	R7
	City	R7
paymentMethod	-	R6
Price	-	R6
Room	-	R6
Concert	-	R6

.

Definition of hierarchies:

Finally, for determining the hierarchies, the DW designer can see from the simplified matrix whether the check sign (√) for the attributes falls into same level (belong to one fact or two facts or more). Recall from the construction phase that all the parameters related to a certain process will be drawn beside this process. Each related parameters organized logically as block which implicitly forms a dimension.

Heuristic rule for defining hierarchies:

R8: *from the column's header, each set of logically related parameters defined by R7, is a candidate for a hierarchy for dimension.*

Note that, the DW designer manually organized these logically related parameters as hierarchies for a certain dimension.

The parameters which represent the column's headers are organized logically based on the existence of the sign (√). For example if the parameters: City, State and County are checked with the sign (√) along with one process (row header), the DW designer can consider these parameters logically tided; so they represent hierarchies for Customer dimension. As well, each process in each row have these parameters or one of them, this can be used as indicator that these parameters are belong to this process as hierarchies. Table 5.6 depicts each fact and dimensions with hierarchies.

Table 5.6. The Generated Dimensions Organized in Hierarchies

Fact	Dimensions	Hierarchies
Bookings	Customer	City
		State
		County
	Date	Hour
		Day
		Month
	Room	-
	Price	-
Buy	Customer	City
		State
		County
	Date	Hour
		Day
		Month

	Concert	-
	Price	-
Payment	Customer	City
		State
		County
	Date	Hour
		Day
		Month
	PaymentMethod	-
	Price	-

5.4 Conclusion

In this chapter, the approach for generating star schemas from business requirements has been described; the steps of the model SSReq which represents generating star schemas from business requirements have been shown. Additionally, the rules defined for automating the generation of the star schemas elements (facts, measures and dimensions) have been defined. The next chapter reveals the matching between the star schemas elements generated from DS and the star schemas elements extracted from BR.

Chapter VI

Star Schemas Matching

6.1 Introduction

The data warehouse (DW) has been considered as a required technology for modern decision support systems (DSS) for organizations. Indeed, the DW offers efficient capabilities for supportingdecision-makers. Despite the valuable efforts and attempts from researchers devoted to developing DW approaches, several DW projects failed (Bargui, Ben-abdallah and Feki, 2016)(Golfarelli & Rizzi 1999).Recently, researchers agree that any successful attempt to develop DW should consider two features, namely i) the construction of the multidimensional schema by using a hybrid approach relying on user requirements and data sources; and ii) the use of semantic resource to overcome the heterogeneity issues shaping the DW construction process (Thenmozhi and Vivekanandan, 2014)(Romero and Abelló, 2010c)(Thenmozhi& Vivekanandan 2012)(Selma *et al.*, 2012).

Furthermore, it is commonly admitted that the intervention of the decision-maker during the DW construction process greatly improves the quality of the DW schema. Obviously, a full automation of the design process may lead to inaccurate results; for instance, defining rules for extracting facts and dimensions automatically may be misleading, as real-world entities (tables, objects…) and relationships both may have common characteristics and therefore may play the role of fact and dimension. Therefore, it is helpful to let the decision-maker or the DW designer intervene for interpreting and evaluating the correctness of designed schemas.

Furthermore, it is widely admitted that the multidimensional star schema represents the keystone structure for DW modeling. This is due to its simplicity, efficiency and its intuitive shape when compared to other multidimensional schemas (e.g., constellation or snowflake schemas).A star schema is composed of one fact that models a business activity

of interest for decision-makers, and has *n (n>1)* dimensions surrounding the fact (Adamson, 2010).Relying the construction of the DW schema on a hybrid approach requires a matching process between BR-Stars(star schemas designed from Business Requirements) on the one hand and DS-Stars(star designed on the Data Source) from the other hand. Indeed, despite the agreement among DW community about the basic methods for constructing the DW, we reveal three issues in the literature review. First, there is no agreement about which technique to use for performing the matching process, if a technique could be adopted. Secondly which kind of semantic resource should be used, if any? Thirdly, whether the approach considers BR first as in (Mazon, Trujillo and LechtenbÄorger, 2007)(Thenmozhi& Vivekanandan 2012)(Di et al. 2015) or the DS first as in(Romero and Abelló, 2010a) or combines both BR and DS simultaneously (Romero and Abelló, 2010c)(Romero and Abelló, 2006).

These issues motivated us to investigate a new hybrid and semi-automatic approach focusing on the matching process of multidimensional star schemas; this approach aims to help DW designers to design star schemas closely related to BR and DS, and that are subject to low effort of adaptation/validation, therefore enhancing the DW quality and reducing costs. Our proposed approach differs from the literature approaches since:

- It relies on a semantic resource; in fact, we have elected WordNet for the matching, this choice enables us to avoid building specific domain ontology; consequently, this will shorten the design/development time of the DW.
- The matching process accepts as input two *complementary* sets of star schemas; this is to consider both BR and DS simultaneously and, therefore build star schemas closely related to users' requirements and the organization's data source.
- The approach involves the DW designer that improves the provable of the resulting multidimensional star schemas.

6.2 Phases of generated approved star schemas

Recent proposed works for constructing the DW greatly concentrate on two features: the use of a hybrid approach; that means, the design process considers both users

requirements and DS data model in order to produce a multidimensional schema. The second feature is to solve the heterogeneity problems by means of using a semantic resource. An important issue is how to apply these two features; hence, the resulting DW helps the organizations offering a reliable multidimensional model compliant to both features.Notably, the matching process we proposed results from two complementary works: One deal with the generation of star schemas from a relational DS model (Elamin, Altalhi and Feki, 2017); the second work tackles the construction of star schemas from BRs (Elamin, Alshomrani and Feki, 2017). Our approach offers a combination of the following features for generating the DW multidimensional model. First, it is hybrid, so we generate star schemas from DS model and star schemas from BRs. Secondly, it is semi-automatic, thus the DW designer can intervene to approve the correctness of the generated star schemas components. Thirdly, our approach relies on a semantic resource to overcome heterogeneity problems due to DS and BRs differences. In the literature, ontology is employed to solve the problems of heterogeneity; however, building domain ontology increases the time and the cost for developing the DW. In our approach we aim to decrease the time for developing the DW, to do so, we have selected WordNet as a ready open source semantic resource for solving the heterogeneity between the DS-Stars elements and the BR-Stars elements. More precisely, our approach consists of three steps: i) Matching DS-Star schemas with BR-Star schemas; ii) Involvement of the DW designer; and iii) Generation of approved star schemas. Figure 6.1 depicts the framework for this approach.

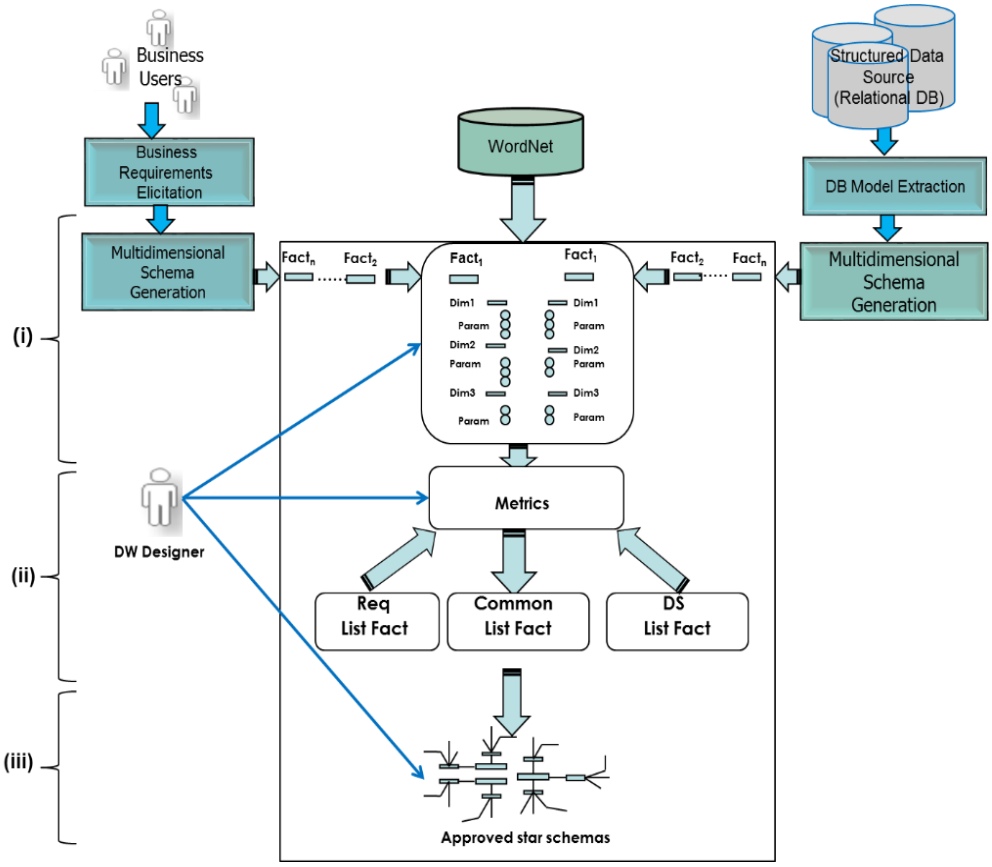

Figure 6.1. Proposed framework for generating Approved star schemas from BRs and DS model

Hereafter, we detail each step for generating approved star schemas.

6.2.1 Matching DS-Star schemas with BR-Star schemas

Schema matching is a complex and time-consuming process (Do, Melnik and Rahm, 2003); it aims to match correctly DS-Star schema elements (fact, dimension, etc...) with BR-Star schema elements. Furthermore, we detect and suggest solutions for unmatched between schemas elements such as diversity in names of elements. Naturally, we optimize the matching step not by computing a Cartesian product of the two schemas' elements. So then, we identify 'similar schemas' (i.e., schemas analyzing the same business activity, having identical or synonym fact names, and common dimensions) and then we match each couple of similar schemas. To do so, we develop an algorithm called *MatchStars*, as well we define two Boolean functions for checking whether two names of fact or dimensions are identical/synonyms or not. Additionally, we define four semantic metrics to measure the

similarity between two star schemas. To assist the DW designer solving the problem of unmatched elements, our approach allows him/her to intervene and matches them manually.

Hereafter we introduce the notation we use.

Notation

- *FName(S): A function that returns the Fact Name from a star schema S.*
- *SBF: A set composed of three lists:*
  - *List of fact names from BR-Stars*
  - *List of Dimension names in each fact from BR-Stars*
  - *List of parameter names in each dimension in each BR-Stars*
- *SDF: A set composed of three lists:*
  - *List of fact names from DS-Stars*
  - *List of Dimension names in each fact from DS -Stars*
  - *List of parameter names in each dimension in each DS -Stars*
- *SCF: A set composed of three lists:*
  - *List of all fact names Common to BR-Stars and DS-Stars.*
  - *List of Dimension names Common to BR-Stars and DS-Stars*
  - *List of parameter names Common to BR-Stars and DS-Stars.*

For the next illustrations, Figure 6.2 depicts an example of three DS-Stars schemas and three BR-Star schemas.

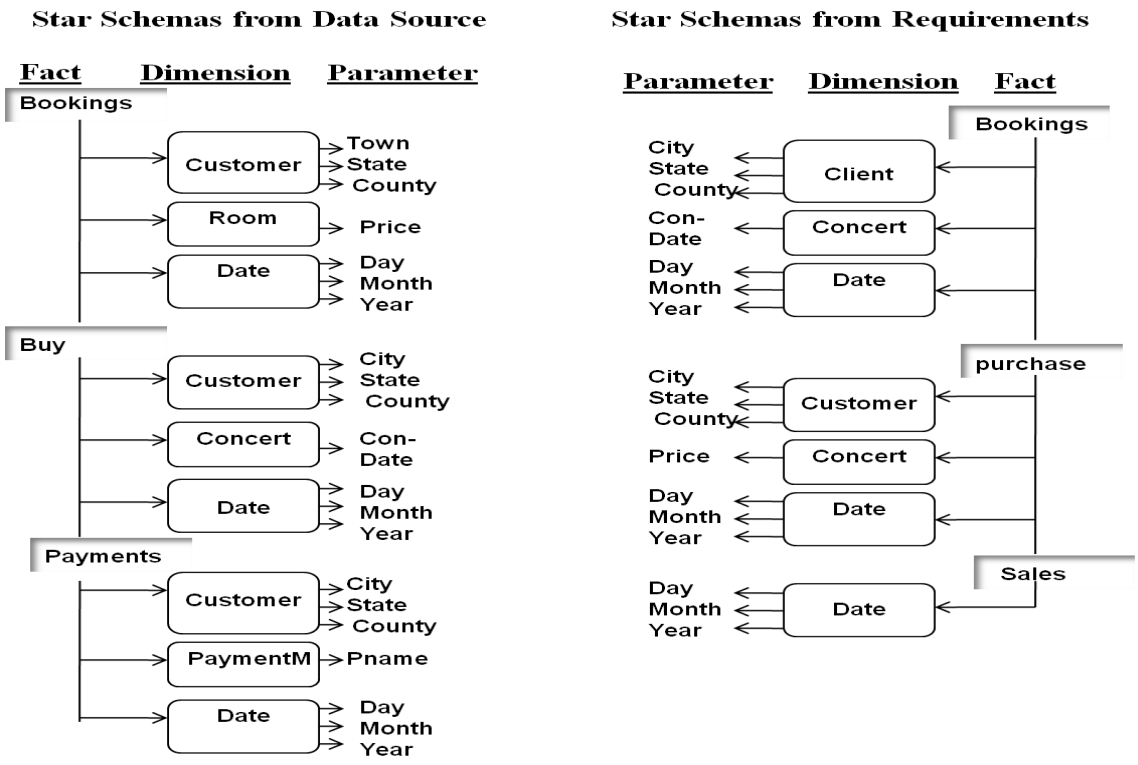

Figure 6.2: Running example with three of *BR-Star Schemas* and three *DS-Star Schemas*

In the upcoming subsection we introduce the boolean functions for matching *BR-Star Schemas* and *DS-Star Schemas.* The objective of these functions is to reveal the construction of (S*CF).*

6.2.1.1Boolean functions for matching DS-Stars and BR-Stars

Hereafter we highlight for each of the two Boolean function Identical and Synonym, its objective, syntax, and then we give the result of its application on our running example of figure 6.2.Note that the results of applying these functions will be append to the *SCF* set of lists.

1) *Function Identical (e1, e2)*: returns *True* if the two-element names *e1* and *e2* are identical and *False* otherwise; *e1, e2* may be fact names or dimension names. When true, this function means that the match of these elements makes sense.

Note that, we apply this function in two steps of the matching process; firstly on fact names, and secondly, on dimension names of identical facts having identical names. We append the common fact name and the common dimension names into *SCFFact* list and *Dimension* list.

For our running example (figure 6.2),the function*Identical (Bookings, Bookings)* is returns True; so then we reuse this function on the dimensions of the two *Bookings* facts in BR-Star and DS-Star. *Identical (Date, Date)* is also True, so we append the *Bookings* fact to the *SCF Fact* list as well as the common dimension *Date* in *Dimension* list to *SCF*.

2) *Function Synonym (e_1, e_2):* returns *True* if the two element names e_1 and e_2 are synonym, and *False* otherwise.

If *Synonym (e_1, e_2)* is *True* considering facts, we reuse it again on their dimensions. If we find synonym dimensions, we append the synonym fact name and the synonym dimension names of this fact to the *Fact* list and *Dimension* list in *SCF* respectively.

For our running example (in figure 6.2), *Synonym (Buy, Purchase)* is *True* for the two facts; therefore we continue to look for the synonym dimensions between the facts *Buy* (in DS-Star) and *Purchase*(in BR-Star). Thus, since *Synonym (Customer, Client)* returns *True* for the two dimensions, we append the *Purchase* fact to the *Fact* list in *SCF* as well as the synonym *Client* dimension to the *Dimension* list in *SCF*.

Table 6.1 depicts the result ofthe Identical and Synonym functions on the example in figure6.2.Note that, so far, only the lists in ***SCF***haveelements(i.e. fact, dimension) while ***SDF*** and ***SBF***remain empty.

Table 6.1. Result of the application of the Boolean functions on the example in figure 6.2

SCF: Set of Common Facts Lists		
Fact	**Dimension**	**Parameters**
Bookings	Date	-
	Client	
Purchase	Customer	-
	Concert	

SDF: Set of DS Facts Lists		
Fact	**Dimensions**	**Parameters**
-	-	-
-		

SBF: Set of BR Facts Lists		
Fact	**Dimensions**	**Parameters**
-	-	-
-		

	Date			

Note that, the DW designer will be asked totreat the uncommon facts and uncommon dimensions later manually in the matching phase. It is worth mentioning that the uncommon dimensions have two cases: in one hand, the dimension name is loadable from the DS, but it is not considered by the business user in BR. On the other hand, the dimension name is not loadable with data from the DS, but it is required by the BR business.

Inorder to support the tractability of our approach, we highlight the main steps of the matching process in the *MatchStars* algorithm. We used the open library Rita that takes two words, it returns whether these two words are synonyms or not based on a threshold we have defined (less than 1). The threshold value 1 causes synonym words to be different (see table 6.2).

Algorithm ***MatchStars***
Aims: Matches the elements of BR-Star Schemas with the elements of DS-Star Schemas.
Input:

- S-BR: Set of star schemas from business requirements.
- S-DS: Set of star schemas from data source.
- fr: a fact in a star schema belonging to S-BR.
- fd: a fact in a star schema belonging to S-DS.
- DIM1: Set of dimensions of fr.
- DIM2: Set of dimensions of fd.

Output: SCF, SBF and SDF
Begin

```
Boolean flag= false;
String str="";
Integer i=1, j=1;
Foreach fact fr in S-BR do
  Foreach fact fd in S-DS do
      If (synonym (fr.name, fd.name) or Rita (fr.name,
         fd.name) <1)
      Foreach Dimension d1 in fr do
        Foreach Dimension d2 in fd do
        If (synonym (d1.name, dim2.name) or Rita
                  (d1.name, dim2.name) <1)
                  SCF[i].Dimension=d1.name
                    Increment i;
        Endif
        Endfor
      Endfor
          Flag = true
```

```
            Str  = fr.name
             break
      Endif
           Else
           Flag = false
      If( rita(fr.name, fd.name)<1)
              SCF[j].Fact=fr.name
      Endfor
      If flag
           SCF[j].Fact=str
         Else
            SBF[j].Fact=fr.name
            SDF[j].Fact=fd.name
             Increment  j;
Endif
Endfor
End
```

Table 6.2 depicts the result of applying the *MatchStars* algorithm on examples in figure 6.2 when different threshold values are taken.

Table 6.2. Result of the Application of MatchStars Algorithm on Example in figure 6.2 with Different Values for Threshold

SCF:Set of Common Facts Lists			**SDF: Set of DS Facts Lists**			**SBF: Set of BR Facts Lists**			**Thre shold**
Fact	**Dimension**	**Parameters**	**Fact**	**Dimension**	**Parameters**	**Fact**	**Dimension**	**Parameter**	
Bookings	Date	-	Payments	Customer	-	Sales	Date	-	**<1**
				Payment					
	Client			Date					
Purchase	Customer	-	-			-			
	Concert								
	Date								
Bookings	Date	-	Payments	Customer	-	Sales	Date	-	1
				Payment					
	Client			Date					
			Purchase	Customer	-	-			
				Concert					
				Date					

Table 6.2 depicts the result of applying the MatchStars algorithm on examples in figure 6.2 when different threshold values are taken.

Note that, in our example in figure 6.2 we have equal number of facts for both BR-Stars and DS-Stars. Inorder to increase the capabilities of our approach, assume now that BR-Stars are limited to the first two facts (*Bookings* and *Purchase*) and the DS-Stars as it is (containing three facts). In this case we append the additional fact (Payments) in the DS-Stars to the SDF. Table 6.3 depicts the result for this case.

Table 6.3 Result generated after applying *MatchStars* algorithm for example 6.2 having BR-Stars limited the first two facts (*Bookings* and *Purchase*).

SCF:Set of Common Facts Lists				**SDF: Set of DS Facts Lists**				**SBF: Set of BR Facts Lists**		
Fact	**Dimension**	**Parameter**		**Fact**	**Dimension**	**Parameter**		**Fact**	**Dimension**	**Parameter**
Bookings	Date			Payments	Customer					
	Client				Payment M					
	Room				Date					
	Concert									
Purchase	Customer									
	Concert									
	Date									

In the upcoming paragraphs we want to measure the matching between the DS-Stars and BR-Stars. Semantic metrics is appropriate tool for this measurement.

6.2.1.2. Metrics for measuring the matching between DS-Stars and BR-Stars

In order to measure the matching between DS-Stars and BR-Stars, we define four metrics namely Common Fact/s (*CF*), Ratio of common fact/s (*RCF*), Common Dimension/s (*CD*) and Ratio of common dimension/s (*RCD*).

The objective of the first metric (CF) is to returns the number of common fact/s between DS-Stars and the BR-Stars. The syntax of this metric is as follows:

$$CF = |(S - BR) \cap (S - DS)| \qquad (1)$$

Where S-BR means the set of star schemas from business requirements; and S-DS means the set of star schemas from data source.

We can calculate the ratio of common fact/s between the DS-Stars and BR-Stars by the second metric (RCF):

$$RCF = \frac{CF}{|(S-BR)\cup(S-DS)|} \qquad (2)$$

$$|(S-BR)\cup(S-DS)| = |(S-BR)| + |(S-DS)| - |(S-BR)\cap(S-DS)|$$

The common dimension/s (*CD*) between (DS-Stars) and (BR-Stars) (*CD*) can be defined by first determining the common dimension/s between each couple of the similar facts between (DS-Stars) and (BR-Stars), then, the union of these common dimensions will give the common dimensions. Note that, we need the number of common dimensions, so we use the cardinality concept. So the syntax of our third metric will be as follows:

$$CD = |\bigcup_{i=1}^{CF}(Dim_1(fr)_i \cap Dim_2(fd)_i)| \qquad (3)$$

Here *CF* represents the number of common facts (metric 1); Dim_1 *(fr)* and Dim_2 *(fd)* represents the set of dimensions of a fact in BR-Stars and the set of dimensions of a fact in DS -Stars respectively.

To measure the ratio of common dimensions we suppose that there are *n* facts in BR-Stars, and *m* facts in DS-Stars; we define (*RCD*) metric as follows:

$$RCD = \frac{CD}{\bigcup_{i=1}^{n}|Dim_1(fr)_i| + \bigcup_{j=1}^{m}|Dim_2(fd)_j| - \bigcup_{i=1,j=a}^{CF}|Dim_1(fr)_i \cap Dim_2(fd)_j|} \qquad (4)$$

Sometimes the construction process of star schemas either from DS or from BR may produce facts empty of measures. This concept of factless facts (i.e., fact having no measures) exists in the DW literature. In our approach, we pay a particular attention to these facts because they may be real factless fact erroneous factless facts. The Designer can decide whether a factless fact should be removed or kept.

6.2.1.3 Rule for empty facts

R1) For each fact f in *BR-Stars* or in DS-*Stars* if the fact contains no measure (M):

The decision return to the Designer.

Finally, we have three sets of lists as in table 6.2; namely, the *set of common facts (SCF),* the *set of business requirements fact (SBF),* and the *set of data source fact (SDF).* Our consideration will devote to the *set of common fact (SCF) lists,* which will be used to be the nuclear of the approved star schemas. The other two set of lists will be matched manually by the DW designer. The result of the manual matching will be added to the *set of common fact lists (SCF).*Note that the final result of the matching process is a set of approved star schemas.

In the previous functions and metrics we use the semantic resource WordNet to overcome the heterogeneity in the matching process. In this subsection we detail the benefits of using the semantic resource.

Usage of a semantic resource

The matching of star schemas requires the usage of a semantic resource, the aim is to identify whether a name of a given concept such as fact or dimension is semantically equivalent to another or not. In our framework, we use the free and open source WordNet as a general semantic resource. The reason behind using WordNet is its simplicity; hence, we can decrease the total time for constructing the multidimensional model. The role of WordNet is obvious in the previous functions: *Identical* (e_1, e_2), *Synonym* (e_1, e_2).

As well, we use Rita, which is free and open source library designed to be simple but have a powerful features(Song, Khare and Dai, 2007). For us, RiTa is suitable since it can be integrated with WordNet database, another reason is that RiTa can implement in java. RiTa works by taking two words under testing and check their resemblance by referencing the WordNet database and return the distance between the two words semantically; if the distance equals 1, this means that there is no relation between those words; if the distance is 0, this is indicate that the two words are synonym (have the same meaning). There is another variation as fraction for the variable distance, this fraction between [0,1]. In our framework we use the value (less than 1) as threshold; so, whenever the distance between the two words is less than 1, this means that those two words are synonyms. The word can be in form of noun or verb. Table 6.4 depicts usefulness of WordNet to overcome the heterogeneity that may arise in the matching process.

Table 6.4. Fact/ Dimensions synonym.

The word	Type (Fact/dimension)	Synonym
Bookings	Fact	Reservation/Engagement
Payment	Fact	defrayment
Buy	Fact	purchase
Customer	Dimension	Client
City	Dimension	Metropolis

Note that, there will be six facts or more, thanks to WordNet, now the number of facts is only three, as the three facts are synonyms. This considerable redundancy may hinder the design of the multidimensional model. Indeed, the full automation of generating the approved star schemas may be inaccurate. The involvement of DW designer is helpful in this process; the second step in our approach is to intervene the DW designer.

6.2.2 Involvement of the DW designer

Note that, our approach is semi-automatic; hence, the DW designer can intervene to reflect the correctness of the generated multidimensional elements. The role of the DW designer in this step is twofold: on one hand, the DW designer checks the *set of common fact to* confirm the generated elements in means of semantic confirmation (see figure 6.1). On the other hand, the DW designer manually matches the elements in the *set of business Requirements fact lists* with those elements in the *set of data source fact lists*. The result of this matching may contain (facts/dimensions/parameters); the DW designer added these elements to the *set of common fact lists (SCF),* finally, the combination is forming the approved star schemas.

6.2.3 Generating approved star schemas

The final output of our framework is a set of approved star schemas; these schemas should satisfy what the decision maker's needs, as well as, loaded correctly with data from the operational data source. Moreover, to enhance schema validity, we defined multidimensional constraints such as avoid empty facts, as well as empty dimensions.

The sources of our approved star schemas are three sets, namely: the *set of common facts (SCF)* which contains the common facts and its consequences as dimensions and parameters in both BR-Star schemas and the DS-Star schemas, the *set of business requirements facts (SBF)* which include the facts and dimensions found only in BR-Star schemas, and *the set of data source facts (SDF)*, that contains all facts with their dimensions that found only in the DS-Star schemas. The approved star schemas encompass the elements in *set of common fact lists* which contains the common facts and the common dimensions as well as the common parameters. Addition to the elements in the *set of common fact lists,* the approved star schemas includes the result of matching the elements in the *set of business requirements facts* and elements in the *set of data source facts* manually by the DW designer. We have checked our system with various examples of star schemas, so as to generate the resulted approved star schemas.

6.3 Conclusion

In this chapter, a matching process between the star schemas generated from DS and the star schemas extracted from BR has been presented. In this process, answers for many questions have been revealed, such as: how the matching done, what are the appropriate functions should be defined to complete this matching?, what are the metrics that can be used to measure the accuracy of the matching?. The algorithm *MatchStars* has been defined to highlight the main steps of the matching process. In addition, we described the role of ontology for solving the problems of heterogeneity between the two schemas has been described. As well, the benefit of evolving the DW designer in the process to confirm the generation of the approved star schemas has been explored. So far, the steps of our hybrid approach for generating approved star schemas in the previous three chapters have been described; in the next chapter, we will shows the future works and our recommendations.

Chapter VII

Results

7.1 Introduction

This chapter is twofold objective: first, we present and explain the results obtained from our framework's parts. Recall that our framework can be seen as three parts, the first one represent designing star schemas from the data source; the second part tackle generating star schemas from business requirements; the third part reveal the process of matching the star schemas generated from data sources with the star schemas obtained from business requirements. Secondly, we discuss the results obtained from the three previous parts and compared it with the results in the literature.

Before start explaining the results obtained from the three parts, let's remember that most of DW researchers agree that the DW should be constructed by considering two important factors: following the hybrid approach and usage of semantic resource. As we said previously, each of these factors is not atomic subject. In other words, the hybrid approach can be accomplished by different methods and techniques; as well as ontology, there are various techniques can be applied to use ontology for DW construction. Interestingly, the noticeable remark in DW area is that despite the rich proposed works, still there are limitations shaping the activities of this process. Thanks to all proposed works that shed light, give concepts, innovations all previous time, their contributions leads us to say the agreement fact "the DW should be constructed by following the hybrid approach and use of ontology". Unfortunately, the limitations exist; the absence of standard exists. The conclusion is to say that DW design process needs more investigation.

7.2 Results

In the upcoming sub sections we reveal our results for each part in our framework. Namely, our contribution in generating star schemas from data source; our contribution regarding the techniques and methods we used to obtain the star schemas from the business requirements; and finally, we explore our opinion in matching these schemas along with ontology to construct our approved multidimensional model.

7.2.1 Results for generating star schemas from data source

Our results in this model are twofold: first, our system automatically classifies the schema tables into three categories: i) strong Entity, ii) relationship and iii) weak Entity. In our knowledge, our work is the first to classify the schema tables into three categories. Hence, it is helpful issue to add this category (tables modeling Weak Entity) to the schema tables. Despite this important remark, most of the previous works in the literature ignore this challenge. Figure 7.1 shows the result of this classification for the schema in figure 4.6. On the other hand, figure 7.2 depicts that the DW designer can intervene and change one table from Entity to relationship.

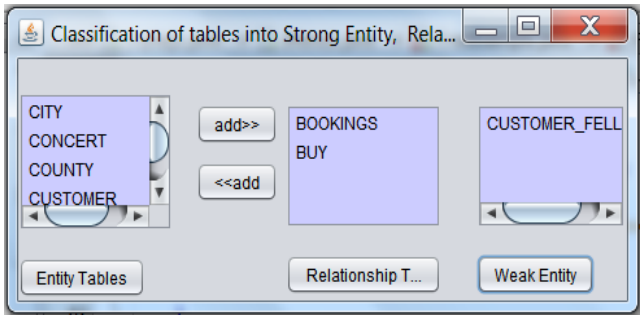

Figure 7.1. The automatic result for classified tables into three categories

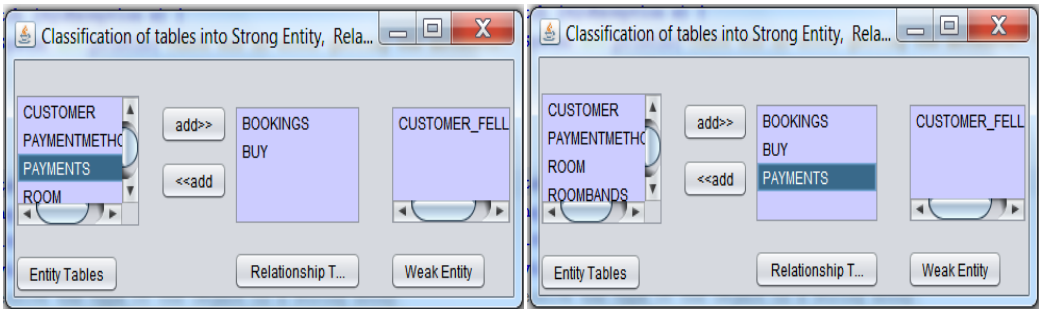

Figure 7.2. Our results show changing one table from entity to relationship

Second, based on our previous rules in chapter 4 section 3 applied on schema in figure 4.6 and the additional two tables in figure 4.7, we obtained the multidimensional model depicted in figure 7.3. These results are more accurate than those described in (Hachaichi, Feki and Ben-Abdallah, 2010). Indeed, this improvement is due to the fact that our rules classified the *Room* table as a dimension; this is a rational result for two reasons, firstly, *Room* is not an event to be classify as fact according to the definition of the fact concept (Golfarelli and Rizzi, 2009b). Secondly, the *Room* table has its primary key as foreign key in another fact table: *Bookings*, so it is likely to be dimension rather than a fact. Although, some authors argue that the table may appear as both fact and dimension (Adamson, 2010), but this will cause an ambiguous issue, we solve this ambiguity by detailed description for the characteristics of the *Room* table as explained above.

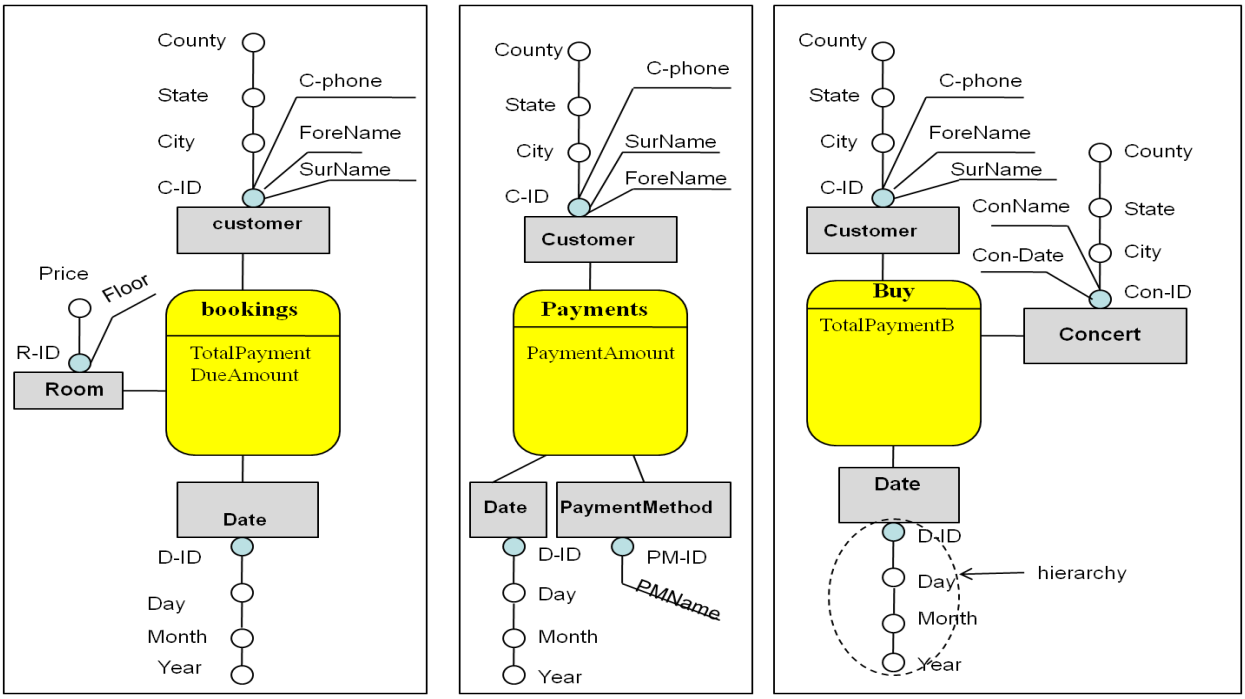

Figure7.3. The Resulting Star Schemas

Figure 7.4 shows the generated star schemas; first, our system automatically produces the facts; second, the DW designer can select one fact and press the measure bottom, the system automatically produces the measure/s for that fact; third, the DW designer press the dimension bottom, the system in turn, produce/s all the dimensions for the generated fact; in this third step as well the identifier for each dimension is generated automatically. Additionally, the right screenshot shows that the DW designer can select a different fact in order to generate its measures, dimensions and the dimensions' identifiers.

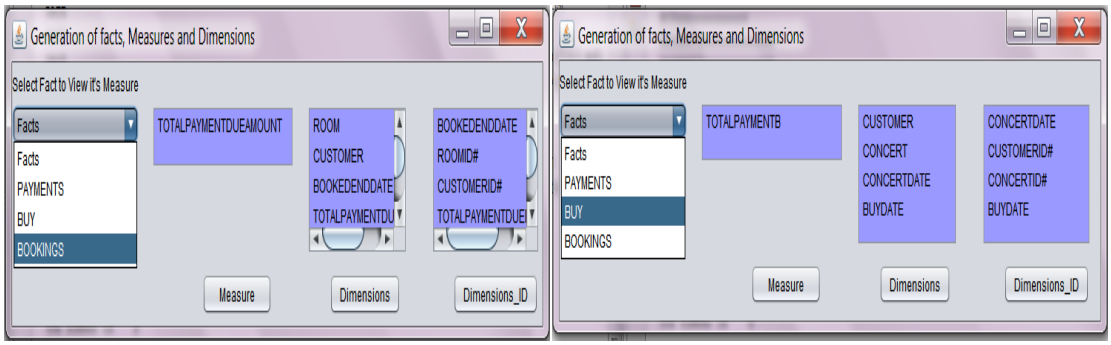

Figure7.4. Generation of facts, measures, dimensions and their identifiers

Additionally, we applied our rules on other schemas (e.g., Company, University) and we obtain rational results (figures 7.5, 7.6, 7.7, and 7.8); as these schemas include weak entity (Elmasri & Navathe 2010).

Employee3 (Fname, Minit, Lname, SSN, BDate, Address, Sex, Salary, SuperSSN#, DNO#)

Department3 (Dname, Dnumber, MGRSSN#, MGRstartDate)

Dept-Location (Dnumber#, Dlocation#)

Project3 (Pname, Pnumber, Plocation, Dnum#)

Works-On (Essn#, Pno#, Hours)

Figure 7.5. A sample extract of database schema (Company)(Elmasri & Navathe, 2010)

Figure 7.6. Classification of company's schema tables

STUDENT (ID_STD, First_Name, Last_Name, City, Birth_Date, Gender, Housing, Tel, Nationality, ID_FAC#)

FACULTY (ID_FAC, Extended_Name, Short_Name, City, ID_Univ#)

COURSE (ID_CRS, Name, Curr_ID#, Semester_No)

CURRICULUM (ID_Curr, Designation, Study_Years)

COURSE-RESULT (ID_STD#, ID_CRS#, Year, Grade-Oral,Grade-Crs-Sess1, Grade-Crs-Sess2, Final-Grade-Crs)

ANNUAL-RESULT (ID_STD#, Univ_Year, Term, Final-Grade-Sess1, Final-Grade-Sess2, Final- Grade, Result)

UNIVERSITY (ID_Univ, Extended_Name, Short_Name)

Figure 7.7. The schema of university database (Choura and Feki, 2011)

Figure 7.8. Classification of university's schema tables

In order to reveal the strength of our approach in generating star schemas from relational data sources, we compare our results with the results of other works in the literature. We used the following properties for the comparison:

- The elements in the reverse engineering process.
- Methods used to defined heuristics rules.
- The source table for defining facts.
- The source table for defining dimensions.
- The disjointness between fact and dimension.
- The automation of constructing the star schemas.

Table 7.1 shows our results compared to other works' results.

Table 7.1 Our Results Compared to Other Approaches' Results

The comparable property	Our results	Other approaches' results
The elements of the reverse engineering process	i. Strong entity. ii. Relationship. iii. Weak entity.	i. Entity. ii. Relationship.
Definition of the heuristic rules	based on the nature of Entity Relationship Diagram (ERD)	based on M:1 relationship
Facts definition	define facts from real relationship table	May define facts from relationship table or weak entity table
Dimensions definition	define dimension from strong entity table.	May define dimension from weak entity table
disjointness between fact tables and dimension tables	Achieve the disjointness	Not achieve the disjointness.

The automation of generated star schemas.	Automatic	Some works are manual.

The design process of a DW is a complex task. The reasons behind this complexity are varying due to the various approaches designers can follow to complete this process. One of these approaches is generating star schemas from the data source. In the following lines we want to reveal our experiment for dealing with such complexity. Our aim is to help the DW designer taking away some complexity colored this research area.

We follow the method of defining heuristic rules for generating the star schemas. Indeed, this will facilitate the automation process for building DW. Of course, we are not alone in this trend, many authors' walks through this way. However, the rules defined so far have some drawbacks (incomplete, weak, inaccurate or do not reflect the data source nature). To fill this gap, we have defined complete, accurate rules that reflect the nature of the data source. Our rules are *twofold*: the first class rules defined for the reverse engineering process. For our knowledge, our work is the first one to classify the schema tables into three classes: *strong entity*, *relationship, and weak entity*. The second class rules defined for extracting the star schema elements (fact, measures and dimensions). We used the relational schema in (Hachaichi, Feki and Ben-Abdallah, 2010); through our work, we faced this problem:

In one hand, **Room** is a fact table:

- Has more than one foreign key.
- Has non key numeric attribute *(price*).

On the other hand, Room is a dimension table:

- Its primary key attribute *(RoomID*) appear as foreign key in a fact table (Bookings).

……..How can we solve this ambiguity?

This question is very hard for the DW designer, but for all stakeholders. It is important to solve this ambiguity; otherwise, the resulting DW will be unclear.

Our defined rules classify **Room** table as dimension, and this agrees with the description of both fact and dimension characteristics (table 2.1). As well, *Room* word is not an event or process to be classified as fact (Adamson, 2010).

In our framework, the activities for constructing the star schemas from data source and the steps for generating star schemas from business requirements is simultaneously achieved. The following lines reveal the results generated by applying our model generating Star Schemas from Requirements (SSReq).

7.2.2 Results for generating star schemas from business requirements

The results obtained in this part reflect the simplicity and flexibility of our approach for generating multidimensional elements from business requirements. Indeed, many authors proposed ideas for doing so, but, for our knowledge, this issue still remains complex and tedious. Our contribution is a simple method that helps the DW designer and the business users to clearly identify their needs in a query form with natural language; the role of DW designer is to input these queries in our system which automatically generates the multidimensional elements. To avoid ambiguity, we define query template styles, the simple style used in case of simple query (i.e. single fact and single parameter), the compound style used for that query that required analyzing single fact by two or more parameters. The generated star schemas depend on the needs of the business users, as an example, the level of details (hierarchy that suitable for their business). Our approach strong by means of using the natural language for expressing the requirements; as well, we used the matrix of requirements which make it possible to normalize the requirements in a consistent form; this form helps the DW designer in generating the multidimensional elements. Figures 7.9, 7.10, 7.11, and 7.12 show the interfaces of our system.

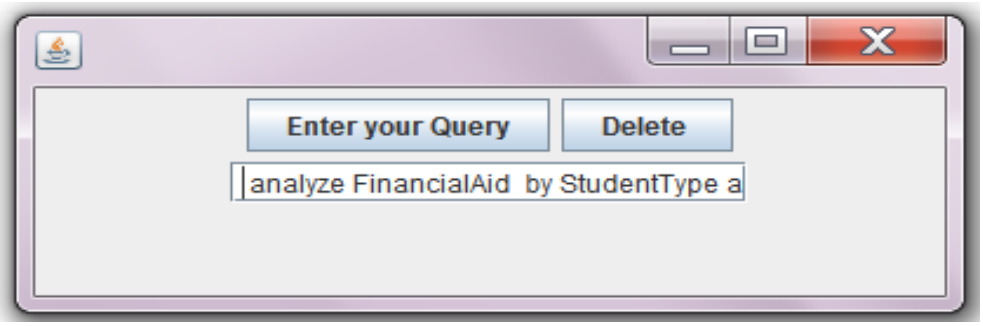

Figure 7.9. An Interface for User to Enter the Query

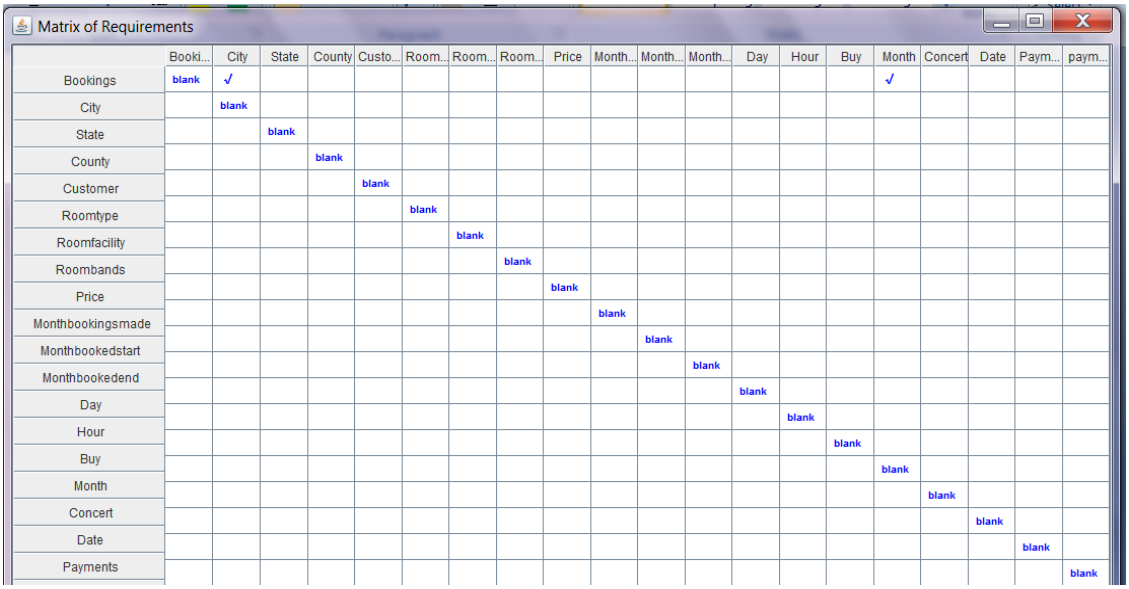

Figure 7.10. Matrix of Requirements After Entering (Q1 in the Booking System)

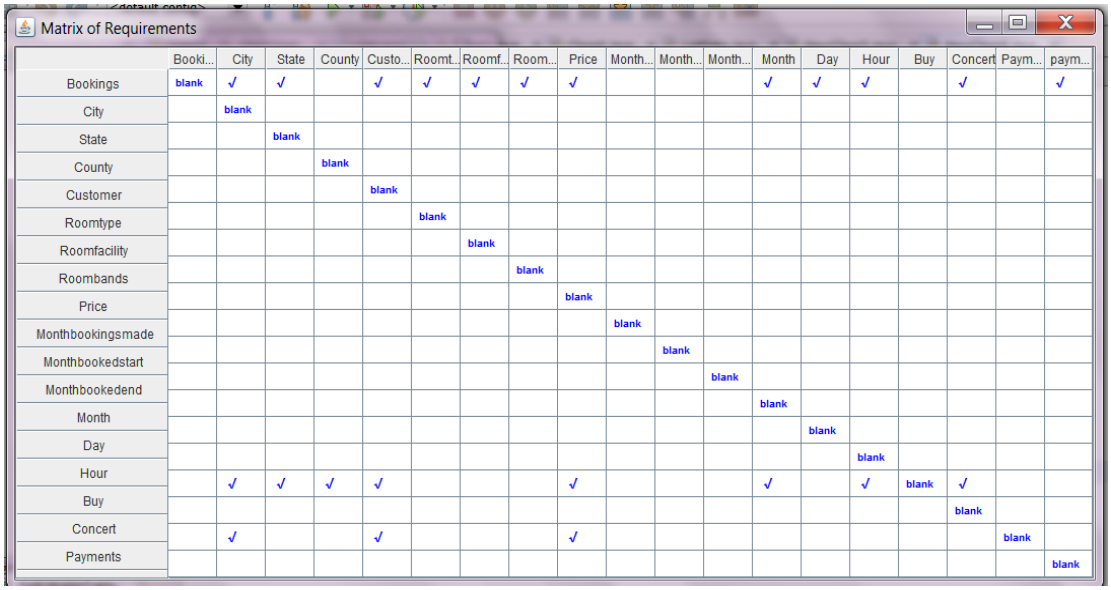

Figure 7.11. Matrix of Requirements after entering all Queries

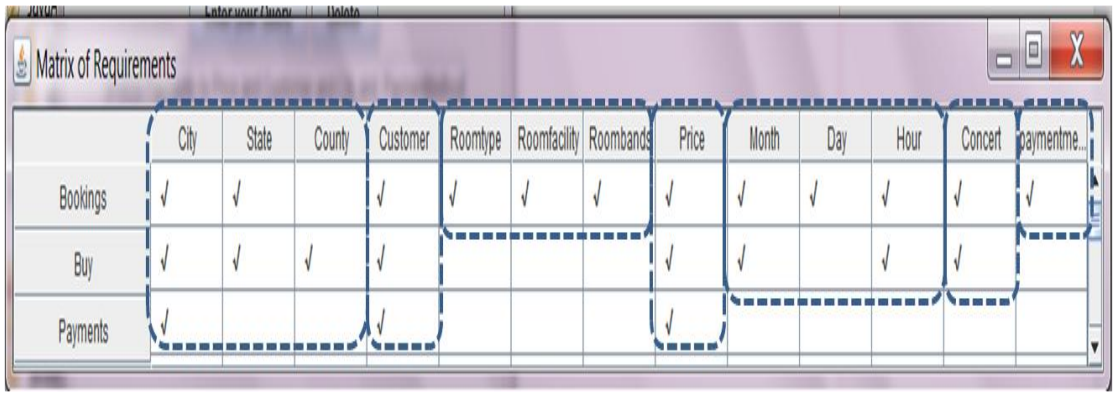

Figure 7.12. The Simplification Process for Generating Multidimensional Elements (bookings case study)

To support the suitability of our approach for generating star schemas from business requirements, it is helpful to apply this approach for a real case study.

Real Case study:

In order to reflect the correctness and the simplicity of our model, it is reasonable to check its ability for generating the multidimensional elements from real business requirements. We chose the college of graduate studies in Sudan University of science and technology (SUST) to be our real case study. The college of high studies in SUST encompasses many

faculties i.e. (Computer sciences, sciences, Medical laboratory sciences and Education); these faculties give certificates in three programs (high diploma, master and PhD). The number of students enrolled in the college about four thousands student. The requirements needed by the business users in the College as they said can be organized into two types:

- Academic requirements: this type of requirements deal with processes for tracking the students' affairs like admission, registration and courses etc.
- Financial requirements: this kind of requirements addresses the processes like FeePayment, Doctors' Merit and Financial Aid.

The noticeable drawbacks of the current system in the college are as follows:

- The difficulties of tracking the students' records among the academic year.
- The absence of the analyzing vision by the higher management users.
- The need for the lowest level employees to explain or to give information about the student record.
- The amount of wasted time taking by employees for tracking the students' affairs.
- The availability of errors in reports generated by the system.

All these drawbacks encourage us planning to design a data mart for the college. Now, in this section we want to use our model (SSReq) for generating star schemas from the business requirements in the college. Table 7.2 shows the processes that the business users in the College need to achieve whereas table 7.3 depicts the star schemas generated by our model.

Table 7.2. Business Requirements for the Graduate College in (SUST)

Queries of the process Fillingform	**Queries of the Enrollment**
Q1) Analyze Fillingform *By* Male and by PhD and by AcademicYear and C_O_Cs	**Q10)** Analyze Enrollment By PhD and by AcademicYear.
Q2) Analyze Fillingform *By* HDiploma and by AcademicYear and by Female and by C_O_Science.	**Q11)** Analyze Enrollment By C_O_Csand by Term and Female.
Q3) Analyze Fillingform *By*C_O_MLaboratory and by Term .	**Q12)** Analyze Enrollment By Male and by HDiploma and by Term and by AcademicYear.
Queries of the process Admission	**Queries of the process FeePayment**
Q4) Analyze Admission By PhD and by AcademicYear and by C_O_Science and by Female.	**Q13)** Analyze FeePayment By Abroad and by AcademicYear and by C_O_MLaboratory.

Q5) Analyze Admission By C_O_MLaboratory and by AcademicYear and by HDiploma.	**Q14)** Analyze FeePayment By PhD and by Male .
Q6) Analyze Admission By Male and by C_O_Cs and by Master .	**Q15)** Analyze FeePayment By C_O_Educationand by AcademicYear and by Local.
Queries of the process Registration	**Queries of the process FinancialAid**
Q7) Analyze Registration By HDiploma and by AcademicYear and by Female and by Local.	**Q16)** Analyze FinancialAid By Female and by AcademicYear and by Master.
Q8) Analyze Registration By C_O_MLaboratory and by Term and by Female and by Abroad.	**Q17)** Analyze FinancialAid By C_O_Cs and by AcademicYear.
Q9) Analyze Registration By Male and by AcademicYear and by . PhD and by C_O_Education	**Q18)** Analyze FinancialAid By Male and by Master and AcademicYear.

Table 7.3. The Star Schemas Results from Business Requirements in College of Graduate Studies case Study

The Fact	The Dimension	Parameters
FillingForm	StudentType	Male
		Female
	ProgramName	PhD
		Master
		HDiploma
	Date	AcademicYear
		Term
	College	C_O_Science
		C_O_Cs
		C_O_Education
Admission	StudentType	Male
		Female
	ProgramName	PhD
		Master
		HDiploma
	Date	AcademicYear
		Term
	College	C_O_Science
		C_O_Cs
		C_O_Education
Registration	StudentType	Male
		Female
	StudentLocation	Abroad
		Local
	ProgramName	PhD
		Master
		HDiploma
	Date	AcademicYear
		Term
	College	C_O_Science
		C_O_Cs
		C_O_Education
	StudentType	Male
		Female
	ProgramName	PhD
		Master

Enrollment		HDiploma
	Date	AcademicYear
		Term
	College	C_O_Science
		C_O_Cs
		C_O_Education
FeePayment	StudentType	Male
		Female
	StudentLocation	Abroad
		Local
	ProgramName	PhD
		Master
		HDiploma
	Date	AcademicYear
		Term
	College	C_O_Science
		C_O_Cs
		C_O_Education
FinancialAid	StudentType	Male
		Female
	ProgramName	PhD
		Master
		HDiploma
	Date	AcademicYear
		Term
	College	C_O_Science
		C_O_Cs
		C_O_Education

Figure 7.13 shows the simplification phase and the generation of multidimensional elements from the business requirements in the College.

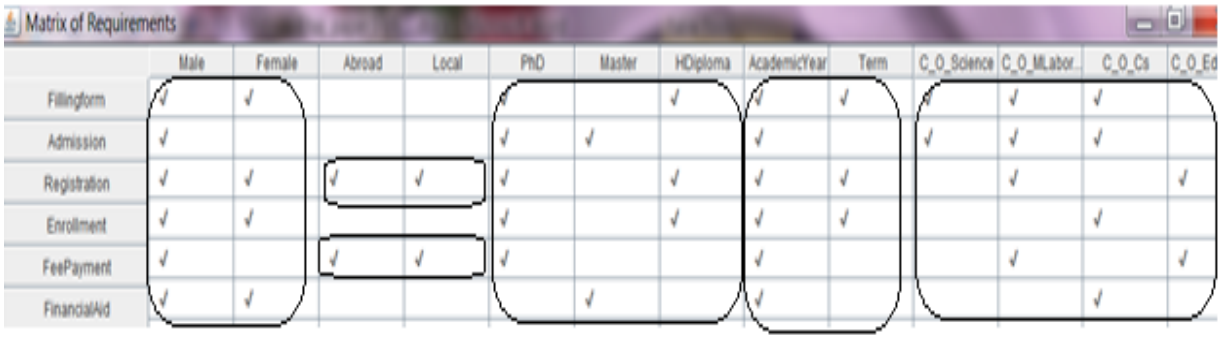

Figure 7.13. The Simplification Process for Generating Multidimensional Elements (SUST) case Study.

7.2.3 Results for matching star schemas from business requirements and star schemas generated from data source

The result of this part is approved star schemas which represent the conceptual model for the DW. To construct this model, we made a matching process between the star schemas generated from the business requirements in our previous work (Elamin, Alshomrani and Feki, 2017) depicted in figure 5.1 and the star schemas issued from the data source in (Elamin, Altalhi and Feki, 2017) shown in figure 4.2. To enhance our results we used WordNet as semantic resource, the aim behind using WordNet is its simplicity as free an open source dictionary and hence we can shorten the period of constructing the multidimensional model semantically. In order to increase the approvability of our model, our approach gives the DW designer the chance to intervene to confirm the acceptance of the concepts (facts, dimensions, etc) in the matching process. The generated results (figures 7.14, 7.15) show the capabilities of our approach for generating the approved star schemas. Our defined functions and metrics covered the various cases that may a raised in the matching process.

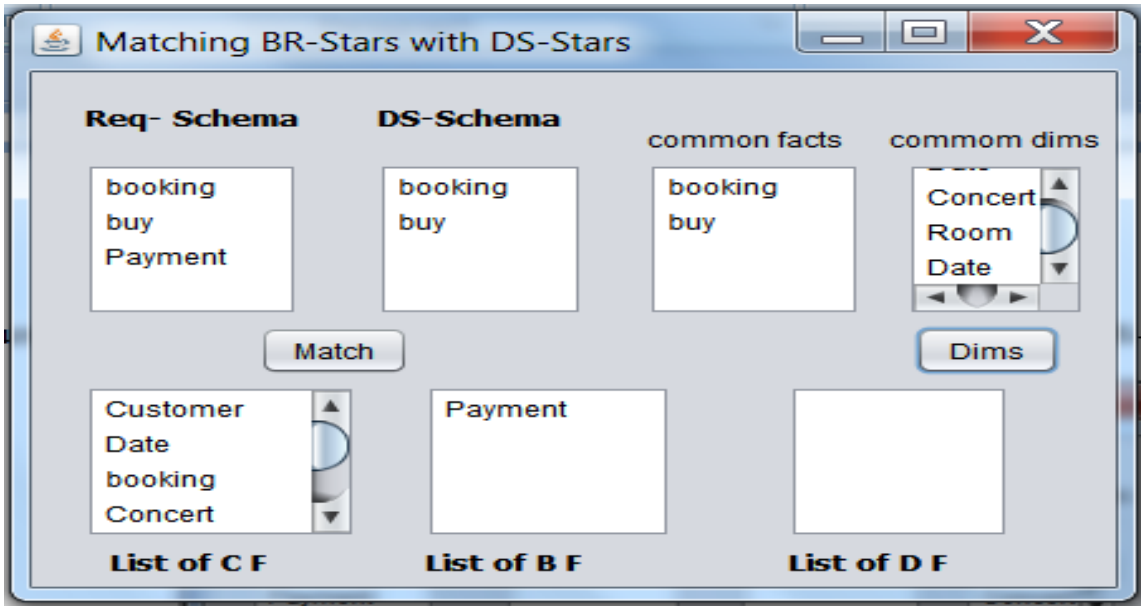

Figure 7.14. Star Schemas approach's Result for Matching BR-Star Schemas with DS-Stars Schemas

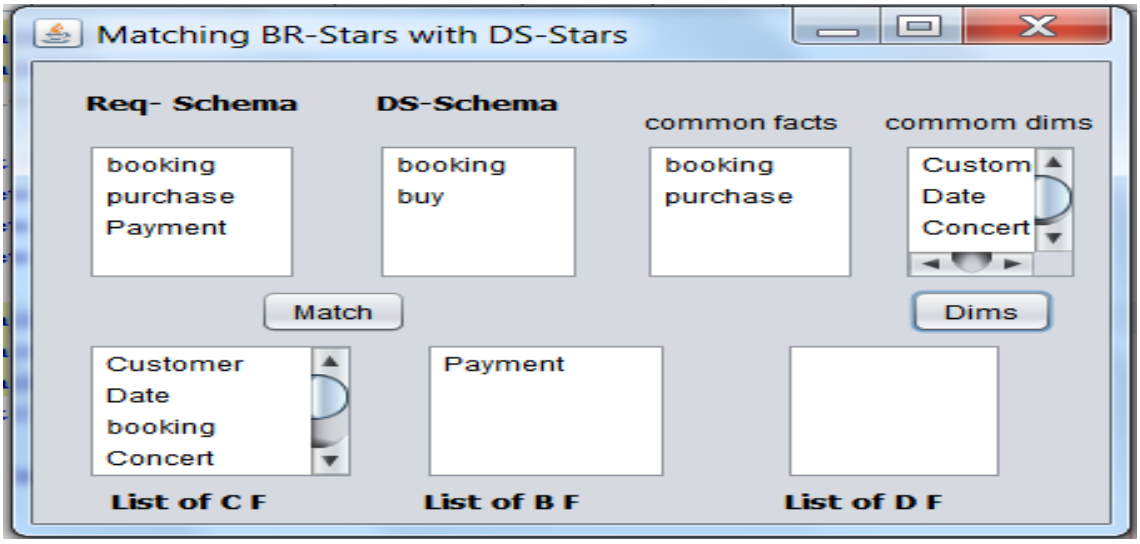

Figure 7.15. Star Schemas approach's Result for matching BR-star schemas with DS-stars schemas having Synonyms

7.3 Chapter Conclusion

The objective of this chapter was to present the results of our thesis; for organizing purpose, we divide our description into two sides: in the first one, the results obtained from each of our framework's parts are presented and explained. In the second side, the results obtained from the three previous parts are discussed and compared with the results in the literature. The obtained results showed the simplicity and suitability of our approach in shorten the period of designing the conceptual model of DW while committing to correct results.

Conclusion and recommendations

In this Thesis, a conceptual model for DW has been designed. To come up with this, a comparative study involving several research works tackling the data warehouse (DW) design process has been conducted. The DW represents the keystone in modern decision support systems (DSS). The DW design process is a complex, tedious and error-prone activity which lacks software tools that assist the DW designer and future users (i.e., decision makers) during the conceptual design steps. These steps, generally performed manually, require skilled persons with competences in several design specialties in classical information systems modeling in order to understand the data source structure when applying a bottom-up approach; in requirements elicitation when the approach design is top-down, the concepts and terminology of the application domain to solve semantic difficulties; and in multidimensional modeling.

Among the main conclusions of this thesis is there is a research trend to the usage of ontology and the hybrid approach during the conceptual design process; nevertheless, on one hand, the efforts investigated so far in using the ontology concept focus on a single/isolated step of the DW design process. Additionally, the previous works try to build domain ontology which increases the complexity and time for constructing the DW. On other hand, the few works that used the hybrid approach did not automate the processes of generating the conceptual model.

The objective of this study was first to highlight this main lack and, secondly to use a semi-automatic hybrid approach covering several steps to build star schemas compliant to the data-source and decision-makers requirements simultaneously. The proposed approach encompasses three phases: first, generate star schemas from business requirements; to come up with this phase a model called SSReq was designed; as well, an algorithm (algorithm1) has been defined. Second, construct star schemas from relational data source; in this phase a heuristic based approach for automating the generated multidimensional model elements has been used. Third, matching the schemas generated from business requirements and the schemas in behave of data source; in this phase, the DW designer can intervene to validate and approved the multidimensional model. Additionally, in this phase the algorithm

MatchStars for the matching process was defined. Moreover, in the matching process a general semantic resource (i.e., WordNet) has been used to overcome the semantic heterogeneity issues. Using such a resource alleviates the design time and cost because efforts for the development of a specific ontology are no longer necessary. Furthermore, being hybrid our approach leads to: i) define two classes of heuristics for extracting multidimensional components starting either from a relational data source and from user requirements, and ii) match star schemas issued from user requirements and from the data source model; for this semi-automatic matching specific metrics have been defined to decide which schemas are similar (i.e., analyzing the same organization business process) and hence should be matched. The usage of metrics is a promising idea to optimize the matching process.

A software prototype that supports the conceptual model has been developed; the results show that our approach facilitates the DW design process; because most of its phases are automated.

Recommendations

The DW design process is a complex task, i.e., it has many strategies designers can follow to come up with it. Based on this fact, each step in DW design can be accomplished by various methods/techniques. In this thesis, the star schemas have been generated from both business requirements and data source. Basically, the data source we have used in this thesis is relational data source. But, as data can be found in other sources (e.g.. XML), we recommend that other works can follow our approach for generating the approved star schemas, but they can use another data source i.e. XML or other sources.

Another recommendation can be said here, is about using another domain/system than the domain used in this thesis (Booking system). Examples for such domains are educational and healthcare domain. Generating approved star schemas from both business requirements and data source from such domains and matches these schemas together will verify the applicability of this approach.

The matching process between the star schemas from data source and that build from requirements depends in this thesis on algorithms and metrics we define for this process. Mainly, these metrics explained the steps for matching facts and dimensions as

elements of multidimensional model. So, we recommend other researchers to define metrics for matching the parameters of each dimension.

Future work

The core phases to design DW are: conceptual model phase, logical model phase and physical model. To complete the picture, our future work will be a logical model and physical model for DW based on our conceptual model.

Another step as future work will be studding other aspects of DW such as security of DW based on our approach. As well, using NOSQL approach in data source side.

Additionally, our study opens up an area for researchers to shorten the period of designing DW by using WordNet as semantic resource for solving the heterogeneity problems.

Reference

Abdalaziz Ahmedl, R. and Mohamed Ahmed, T. (2014) 'Generating Data Warehouse Schema', *International Journal in Foundations of Computer Science & Technology*, 4(1), pp. 1–16. doi: 10.5121/ijfcst.2014.4101.

Adamson, C. (2010) *Star Schema the complete reference*. McGraw Hill Professional.

Ballard, C. *et al.* (1998) *Data Modeling Techniques for Data Warehousing*. First Edit. IBM Corporation, International Technical Support Organization.

Bargui, F., BEN-ABDALLAH, H. and FEKI, J. (2009) 'Multidimensional Concept Extraction and Validation from OLAP Requirements in NL', in. IEEE. doi: 10.1109/NLPKE.2009.5313769.

Bargui, F., Ben-Abdallah, H. and Feki, J. (2010) 'A Hybrid Approach for Data Mart Schema Design from NL-OLAP Requirements', in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 295–296. doi: 10.1007/978-3-642-12550-8_26.

Bargui, F., Ben-abdallah, H. and Feki, J. (2016) 'A natural language-based approach for a semi-automatic data mart design and ETL generation', 125(April). doi: 10.1080/12460125.2016.1158066.

Bargui, F., Feki, J. and Ben-Abdallah, H. (2008) 'A natural language approach for data mart schema design', in *9th International Arab Conference on Information Technology (ACIT)*.

Boehnlein, M. and Ulbrich-vom Ende, A. (1999) 'Deriving initial data warehouse structures from the conceptual data models of the underlying operational information systems', in *Proceedings of the 2nd ACM international workshop on Data warehousing and OLAP - DOLAP '99*. New York, New York, USA: ACM Press, pp. 15–21. doi: 10.1145/319757.319780.

BONCZEK, R. H., HOLSAPPLE, C. W. and WHINSTON, A. B. (1981) *Foundations of Decision Support Systems*. Elsevier Inc.

Bonifati, A. *et al.* (2001) 'Designing data marts for data warehouses', *ACM Transactions on Software Engineering and Methodology*, 10(4), pp. 452–483. doi: 10.1145/384189.384190.

Breslin, M. (2004) 'Data Warehousing Battle of the Giants : Comparing the Basics of the Kimball and Inmon Models', *Business Intelligence Journal*, pp. 6–20.

Burstein, F. and W. Holsapple, C. (2008) *Handbook on Decision Support Systems 1*. Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/978-3-540-48713-5.

Cabibbo, L. and Torlone, R. (1998) 'A logical approach to multidimensional databases', in. Springer, Berlin, Heidelberg, pp. 183–197. doi: 10.1007/BFb0100985.

Calvanese, D. *et al.* (2006) 'Enterprise modeling and Data Warehousing in Telecom Italia', *Information Systems*, 31(1), pp. 1–32. doi: 10.1016/j.is.2004.07.002.

Carmè, A., Mazón, J.-N. and Rizzi, S. (2010) 'A Model-Driven Heuristic Approach for Detecting Multidimensional Facts in Relational Data Sources', in, pp. 13–24. doi: 10.1007/978-3-642-15105-7_2.

Chandwani, G. and Uppal, V. (2015) 'Implementation of Star Schemas from ER Model', *International Journal of Database Theory and Application*, 8(3), pp. 111–130.

Chaudhuri, S. and Dayal, U. (1997) 'An overview of data warehousing and OLAP technology', *ACM SIGMOD Record*, 26(1), pp. 65–74. doi: 10.1145/248603.248616.

Choura, H. and Feki, J. (2011) 'MDA compliant approach for data mart schemas generation', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6918 LNCS, pp. 262–269. doi: 10.1007/978-3-642-24443-8_27.

Cravero, A. and Sepúlveda, S. (2014) 'Multidimensional Design Paradigms for Data Warehouses: A Systematic Mapping Study', *Journal of Software Engineering and Applications*, 7(1), pp. 53–61. doi: 10.4236/jsea.2014.71006.

Cruz, I. F. and Xiao, H. (2005) 'The Role of Ontologies in Data Integration', *Engineering Intelligent Systems*, 13(4), pp. 245–252.
dell'Aquila, C. *et al.* (2010) 'Logic Programming for Data Warehouse Conceptual Schema Validation', in *Lecture Notes in Computer Science*, pp. 1–12. doi: 10.1007/978-3-642-15105-7_1.
Di, T., Lefons, E. and Tangorra, F. (2015) 'Academic data warehouse design using a hybrid methodology', *Computer Science and Information Systems*, 12(1), pp. 135–160. doi: 10.2298/CSIS140325087D.
Do, H., Melnik, S. and Rahm, E. (2003) 'Comparison of Schema Matching Evaluations'. Springer-Verlag Berlin Heidelberg, pp. 221–237. doi: 10.1007/3-540-36560-5_17.
Druzdzel, M. J. and Flynn, R. R. (2002) *Decision Support Systems*. Available at: http://www.sis.pitt.edu/_dsl.
Elamin, E., Alshomrani, S. and Feki, J. (2017) 'SSReq: A method for designing Star Schemas from decisional requirements', in *2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE)*. IEEE, pp. 1–7. doi: 10.1109/ICCCCEE.2017.7867645.
Elamin, E., Altalhi, A. and Feki, J. (2017) 'Heuristic Based Approach for Automating Multidimensional Schemas Construction', *International Journal of Computer and Information Technology*, 6(6), pp. 286–295.
Elmasri & Navathe (2010) *Fundamentals of database systems*. SIXTH EDIT. Addison-Wesley.
El-Sappagh, S. H. A., Hendawi, A. M. A. and El Bastawissy, A. H. (2011) 'A proposed model for data warehouse ETL processes', *Journal of King Saud University - Computer and Information Sciences*. King Saud University, 23(2), pp. 91–104. doi: 10.1016/j.jksuci.2011.05.005.
Gali, A. *et al.* (2004) 'From Ontology to Relational Databases', in *Conceptual Modeling for Advanced Application Domains*, pp. 278–289. doi: 10.1007/978-3-540-30466-1_26.
Giorgini, P., Rizzi, S. and Garzetti, M. (2008) 'GRAnD: A goal-oriented approach to requirement analysis in data warehouses', *Decision Support Systems*, 45(1), pp. 4–21. doi: 10.1016/j.dss.2006.12.001.
Golfarelli, M. (2010) 'From User Requirements to Conceptual Design in Warehouse Design - a Survey', *Data Warehousing Design and Advanced Engineering Applications*. IGI Global, pp. 1–16. doi: 10.4018/978-1-60566-756-0.ch001.
Golfarelli, M., Maio, D. and Rizzi, S. (1998) 'Conceptual design of data warehouses from E/R schemes', in *Proceedings of the Thirty-First Hawaii International Conference on System Sciences*. IEEE Comput. Soc, pp. 334–343. doi: 10.1109/HICSS.1998.649228.
GOLFARELLI, M., MAIO, D. and RIZZI, S. (1998) 'THE DIMENSIONAL FACT MODEL: A CONCEPTUAL MODEL FOR DATA WAREHOUSES', *International Journal of Cooperative Information Systems*, 7(02n03), pp. 215–247. doi: 10.1142/S0218843098000118.
Golfarelli, M. and Rizzi, S. (1998) 'A methodological framework for data warehouse design', in *Proceedings of the 1st ACM international workshop on Data warehousing and OLAP - DOLAP '98*. New York, New York, USA: ACM Press, pp. 3–9. doi: 10.1145/294260.294261.
Golfarelli, M. and Rizzi, S. (1999) 'Designing the Data Warehouse: Key Steps and Crucial Issues', *Journal of Computer Science and Information Management*, 2(3), pp. 1–14. doi: 10111241900.
Golfarelli, M. and Rizzi, S. (2009a) 'A comprehensive approach to data warehouse testing', *Proceeding of the ACM twelfth international workshop on Data warehousing and OLAP - DOLAP '09*, p. 17. doi: 10.1145/1651291.1651295.
Golfarelli, M. and Rizzi, S. (2009b) 'Data warehouse design: Modern principles and methodologies', *Data Warehouse*. New York, New York, USA: McGraw-Hill Osborne Media, pp. 1–42, 420–423.
Hachaichi, Y. and Feki, J. (2013) 'AN AUTOMATIC METHOD FOR THE DESIGN OF MULTIDIMENSIONAL SCHEMAS FROM OBJECT ORIENTED DATABASES', *International Journal of Information Technology &Decision Making*, 12(6), pp. 1223–1259. doi: 10.1142/S0219622013500351.

Hachaichi, Y., Feki, J. and Ben-Abdallah, H. (2010) 'Designing Data Marts from XML and Relational Data Sources', in *Data Warehousing Design and Advanced Engineering Applications*. IGI Global, pp. 55–80. doi: 10.4018/978-1-60566-756-0.ch004.
He, L. *et al.* (2011) 'An Ontology-Based Conceptual Modeling Method for Data Warehouse', *Information Technology, Computer Engineering and Management Sciences (ICM), 2011 International Conference on*, 4, pp. 130–133. doi: 10.1109/ICM.2011.171.
Inmon, W. H. (2002) *Building the Data Warehouse*. Third Edit. Edited by R. Elliott. John Wiley & Sons, Inc.
Jensen, C. S., Pedersen, T. B. and Thomsen, C. (2010) *Multidimensional Databases and Data Warehousing, Synthesis Lectures on Data Management*. doi: 10.2200/S00299ED1V01Y201009DTM009.
Jensen, M. R., Holmgren, T. and Pedersen, T. B. (2004) 'Discovering Multidimensional Structure in Relational Data', in *Data Warehousing and Knowledge Discovery*. Springer-Verlag Berlin Heidelberg, pp. 138–148. doi: 10.1007/978-3-540-30076-2_14.
Jeusfeld, M. A. and Thoun, S. (2016) 'Key Performance Indicators in Data Warehouses', in *Lecture Notes in Business Information Processing, vol 253*, pp. 111–129. doi: 10.1007/978-3-319-39243-1_5.
Jiang, L. *et al.* (2011) 'An automatic method of data warehouses multi-dimension modeling for distributed information systems', in *Proceedings of the 2011 15th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, pp. 9–16. doi: 10.1109/CSCWD.2011.5960048.
Kaldeich, C. and Sá, J. O. e (2004) 'Data Warehouse Methodology: A Process Driven Approach', in *Advanced Information Engineering*, pp. 536–549. doi: 10.1007/978-3-540-25975-6_38.
Khouri, S. and Bellatreche, L. (2011) 'An Automatic Method of Data Warehouses Multidimension Modeling for Distributed Information Systems', pp. 438–441.
Khouri, S. and France, F. (2010) 'A Methodology and Tool for Conceptual Designing a Data Warehouse from Ontology-based Sources', pp. 19–23.
Kimball, R. and Ross, M. (2011) *The data warehouse toolkit: the complete guide to dimensional modeling*.
Kozielski, S. and Wrembel, R. (2009) *Kozmaina*. doi: 10.1007/978-0-387-87431-9.
Kozmina, N., Niedrite, L. and Solodovnikova, D. (2008) 'A Knowledge-based Method for Transforming Requirements to Conceptual Model of a Data Warehouse A Knowledge-based Method for Transforming Requirements to Conceptual Model of a Data Warehouse', in *Proceedings of the 8th International Baltic Conference on Databases and Informational Systems (DBIS 2008)*. Estonia.
Luján-Mora, S. and Trujillo, J. (2006) 'Physical modeling of data warehouses using UML component and deployment diagrams: Design and implementation issues', *Journal of Database Management*, 17(2), pp. 12–42. doi: 10.4018/jdm.2006040102.
M.Thenmozhi and K.Vivekanandan (2014) 'An Ontological Approach to Handle Multidimensional Schema Evolution for Data Warehouse', *International Journal of Database Management Systems*, 6(3), pp. 33–52. doi: 10.5121/ijdms.2014.6303.
Malinowski, E. and Zim´anyi, E. (2008) *Advanced data warehouse design: from conventional to spatial and temporal applications*.
Mazón, J.-N. *et al.*(2005) 'REBNITA 2005, 1ST INTERNATIONAL WORKSHOP ON REQUIREMENTS ENGINEERING FOR BUSINESS NEED AND IT ALIGNMENT', in.
Mazón, J.-N. and Trujillo, J. (2009) 'A hybrid model driven development framework for the multidimensional modeling of data warehouses!', *ACM SIGMOD Record*, 38(2), p. 12. doi: 10.1145/1815918.1815920.
Mazon, J.-N., Trujillo, J. and LechtenbÄorger, J. (2007) 'Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms', *Data & Knowledge Engineering*, 63(3), pp. 725–751. doi: 10.1016/j.datak.2007.04.004.

Mullin, A. and Motz, R. (2011) 'The OntoDW Methodology'.
Nabli, A., Feki, J. and Gargouri, F. (2005) 'Automatic Construction of Multidimensional Schema from OLAP Requirements', (Figure 1).
Nabli, A., Feki, J. and Gargouri, F. (2009) 'An Ontology Based Method for Normalisation of', pp. 235–246.
Nebot, V. *et al.* (2009) 'No Title', pp. 1–35.
Nebot, V. and Berlanga, R. (2010) 'Building Data Warehouses with Semantic Data'.
Pardillo, J. and Mazon, J. N. (2011) 'Using Ontologies for the Design of Data Warehouses', *International Journal of Database Management Systems*, 3(2), pp. 73–87. doi: 10.5121/ijdms.2011.3205.
Peralta, V., Illarze, A. and Ruggia, R. (2003) 'On the Applicability of Rules to Automate Data Warehouse Logical Design', *Proc. Decision Systems Engineering Workshop (DSE'03), in conjunction with the 15th Conf. on Advanced Inform. Syst. Eng. (CAiSE '03),* pp. 329–340.
Phipps, C. and Davis, K. C. (2002) 'Automating data warehouse conceptual schema design and evaluation.', *Dmdw*, pp. 1–10. Available at: http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-58/phipps-davis.pdf?origin=publication{_}detail.
Ponniah, P. (2001) *Data Warehousing Fundamentals: A Comprehensive Guide for IT Professionals*, *Data Warehousing Fundamentals - A Comprehensive Guide for IT Professionals*.
Prat, N., Akoka, J. and Comyn-wattiau, I. (2006) 'A UML-based data warehouse design method', 42, pp. 1449–1473. doi: 10.1016/j.dss.2005.12.001.
Raghavan, S., Zelesnik, G. and Ford, G. (1994) *Lecture Notes on Requirements Elicitation*.
Rainardi, V. (2008) 'Building a data warehouse: with examples in SQL Server', in *9th International Arab Conference on Information Technology (ACIT)*. Berkeley, CA: Apress. doi: 10.1007/978-1-4302-0528-9.
Rajagopal, P. *et al.* (2005) 'A New Approach for Software Requirements Elicitation', 2.
Rilston, F. *et al.* (2002) 'Towards a Methodology for Requirements Analysis of Data Warehouse Systems', in *Proc. of the XVI Simpósio Brasileiro de Engenharia de Software*. Rio Grande do Sul, Brazil, pp. 146–161. doi: 10.1.1.121.6268&.
Rizzi, S. *et al.*(2006) 'Research in data warehouse modeling and design', in *Proceedings of the 9th ACM international workshop on Data warehousing and OLAP - DOLAP '06*. New York, New York, USA: ACM Press, p. 3. doi: 10.1145/1183512.1183515.
Romero, O. and Abelló, A. (2006) 'Multidimensional Design by Examples', in *Lecture Notes in Computer Science, vol 4081*. Berlin, Heidelberg: Springer, pp. 85–94. doi: 10.1007/11823728_9.
Romero, O. and Abelló, A. (2007a) 'Automating multidimensional design from ontologies', in *Proceedings of the ACM tenth international workshop on Data warehousing and OLAP - DOLAP '07*. New York, New York, USA: ACM Press, p. 1. doi: 10.1145/1317331.1317333.
Romero, O. and Abelló, A. (2007b) 'Generating Multidimensional Schemas from the Semantic Web', in *CAiSE Forum*, pp. 69–72.
Romero, O. and Abelló, A. (2009) 'A Survey of Multidimensional Modeling Methodologies', *International Journal of Data Warehousing and Mining*, 5(2), pp. 1–23. doi: 10.4018/jdwm.2009040101.
Romero, O. and Abelló, A. (2010a) 'A framework for multidimensional design of data warehouses from ontologies', *DATAK*. Elsevier B.V., 69(11), pp. 1138–1157. doi: 10.1016/j.datak.2010.07.007.
Romero, O. and Abelló, A. (2010b) 'Academic Data Warehouse Design Using a Hybrid Methodology', *DATAK*. Elsevier B.V., 69(11), pp. 1138–1157. doi: 10.1016/j.datak.2010.07.007.
Romero, O. and Abelló, A. (2010c) 'Automatic validation of requirements to support multidimensional design', *DATAK*. Elsevier B.V., 69(9), pp. 917–942. doi: 10.1016/j.datak.2010.03.006.
Romero, O., Simitsis, A. and Abelló, A. (2011) 'GEM: Requirement-Driven Generation of ETL and Multidimensional Conceptual Designs', in *Lecture Notes in Computer Science, vol 6862*. Berlin, Heidelberg: Springer, pp. 80–95. doi: 10.1007/978-3-642-23544-3_7.

Salinesi, C. and Gam, I. (2007) ‘A Requirement-driven Approach for Designing Data Warehouses’, in *Requirements Engineering: Foundations for Software Quality (REFSQ’06)*.
Schiefer, J., List, B. and Bruckner, R. M. (2002) ‘A Holistic Approach for Managing Requirements of DataWarehouse Systems’, in *Proceedings of the Americas Conference on Information Systems*, pp. 77–87.
Selma, K. *et al.* (2012) ‘Ontology-based structured web data warehouses for sustainable interoperability: requirement modeling, design methodology and tool’, *Computers in Industry*, 63(8), pp. 799–812. doi: 10.1016/j.compind.2012.08.001.
Song, I. Y., Khare, R. and Dai, B. (2007) ‘SAMSTAR: A Semi-Automated Lexical Method for Generating Star Schemas from an Entity-Relationship Diagram’, in *Proceedings of the ACM tenth international workshop on Data warehousing and OLAP - DOLAP ’07*. New York, New York, USA: ACM Press, p. 9. doi: 10.1145/1317331.1317334.
SUDUC, A.-M. *et al.* (2010) ‘Evolution of Decision Support Systems Research Field in Numbers’, *Informatica Economicã*, 14(4), pp. 78–86.
Thenmozhi, M. and K.Vivekanandan (2012) ‘Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms’, 54(8).
Thenmozhi, M. and Vivekanandan, K. (2012) ‘An Ontology based Hybrid Approach to Derive Multidimensional Schema for Data Warehouse’, *International Journal of Computer Applications*, 54(8), pp. 36–42. doi: 10.5120/8590-2343.
Thenmozhi, M. and Vivekanandan, K. (2013) ‘A Tool for Data Warehouse Multidimensional Schema Design using Ontology’, 10(2), pp. 161–168.
Tria, F. Di, Lefons, E. and Tangorra, F. (2012) ‘Hybrid methodology for data warehouse conceptual design by UML schemas’, *Information and Software Technology*. Elsevier B.V., 54(4), pp. 360–379. doi: 10.1016/j.infsof.2011.11.004.
Tryfona, N., Busborg, F. and Borch Christiansen, J. G. (1999) ‘starER: A Conceptual Model for Data Warehouse Design’, in *Proceedings of the 2nd ACM international workshop on Data warehousing and OLAP - DOLAP ’99*. New York, New York, USA: ACM Press, pp. 3–8. doi: 10.1145/319757.319776.
Tsois, A., Karayannidis, N. and Sellis, T. K. (2001) ‘MAC: Conceptual data modeling for OLAP.’, *Proceedings of the 3rd Intl. Workshop on Design and Management*, 39, p. 5. Available at: http://dblp.uni-trier.de/db/conf/dmdw/dmdw2001.html#TsoisKS01.
Turban, E., Aronson, J. E. and Liang, T.-P. (2005) *Dicision support systems and intelligent systems*. 7 edition. Prentice Hall.
Vaisman, A. and Zimányi, E. (2014) *Data Warehouse Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/978-3-642-54655-6.

Appendices

Appendix A

Java Code for Generating Star Schemas from Data source

Extracting Strong Entity

```
public void getColumnsMetadata(ArrayList<String> tables) throws SQLException {
      ResultSet rs;
      int entity = 0;
      int relationship = 0;
      ArrayList<String> entity_table;
      entity_table = new ArrayList<>();
      for (String actualTable : tables) {
         rs = metadata.getPrimaryKeys(null, null, actualTable);
                        System.out.println(actualTable.toUpperCase());
              int pkeycount = 0;
                        while (rs.next()) {
                 System.out.println(rs.getString("COLUMN_NAME"));
                 pkeycount++;
                                          }
               int fkcount = 0;
               rs = metadata.getImportedKeys(null, null, actualTable);
               while (rs.next()) {
               fkcount++;
                      }
         int number = 0;
         String columnName;
         String columnType ;
         rs = metadata.getColumns(null, null, actualTable, null);
         while (rs.next()) {
            columnName = rs.getString("COLUMN_NAME");
            columnType = rs.getString("TYPE_NAME");
            if (columnType.equals("NUMBER")) {
               number++;
               System.out.println("the number is    " + number);

            }
         }

               if (pkeycount <= 1) {

                   entity_table.add(actualTable);
                   entity++;
               }
              System.out.println("this table represent Strong Entity ");

         System.out.println("\n");
            }
         for (String actualTable : entity_table) {
         dlm1.addElement(actualTable);
      }
```

```
      jList1.setModel(dlm1);
   }// end of getcolumnmetadata()
```

Extracting Relationships

```
public  void getColumnsMetadata2(ArrayList<String> tables)throws SQLException {
          ResultSet rs;
           String columnName;
           ArrayList<String>  my_relation_table;

           ArrayList<String>  primaryk;
           String fkcolumnName;
           ArrayList<String> pkcolumn;
           ArrayList<String> fkcolumn;

                    for (String actualTable : tables) {
              rs=metadata.getPrimaryKeys(null, null,actualTable);
                   System.out.println(actualTable.toUpperCase());
              my_relation_table = new ArrayList<>();
              pkcolumn = new ArrayList<>();
              fkcolumn = new ArrayList<>();
              primaryk = new ArrayList<>();
                              while (rs.next()) {
                    columnName = rs.getString("COLUMN_NAME");

                    System.out.println("PrimaryKeys: columnName=" + columnName);
                      pkcolumn.add(columnName);
                  }
                  rs = metadata.getImportedKeys(null, null, actualTable);
                  int fkcount=0;
                  while (rs.next()){
                     fkcount++;
                  fkcolumnName = rs.getString("FKCOLUMN_NAME");
                     fkcolumn.add(fkcolumnName);
                  }
          for (String pk : pkcolumn) {
             for (String fk : fkcolumn) {
                  if ((pk.equals(fk))){
                      if (fkcount > 1) {
                     my_relation_table.add(actualTable);

                     Set<String> hs = new HashSet<>();
                         hs.addAll(my_relation_table);
                         my_relation_table.clear();
                         my_relation_table.addAll(hs);
                        }
                       System.out.println("this table represent relationship");
                     }
             }
          }
                                  System.out.println("\n");
                for (String my : my_relation_table){
                  dlm2.addElement(my);

                }
               jList2.setModel(dlm2);
```

```
    }
    }
```

Extracting Weak Entity

```
public void getColumnsMetadataWeak(ArrayList<String> tables) throws SQLException {
      ResultSet rs;
      String columnName;
      ArrayList<String> entity2_table;
      ArrayList<String> primaryk;
      String fkcolumnName;
      ArrayList<String> pkcolumn;
      ArrayList<String> fkcolumn;
      for (String actualTable : tables) {
         rs = metadata.getPrimaryKeys(null, null, actualTable);
         System.out.println(actualTable.toUpperCase());
         int pkcount = 0;
         entity2_table = new ArrayList<>();
         pkcolumn = new ArrayList<>();
         fkcolumn = new ArrayList<>();
         primaryk = new ArrayList<>();
         while (rs.next()) {
            pkcount++;
            columnName = rs.getString("COLUMN_NAME");
            System.out.println("PrimaryKeys: columnName=" + columnName);
            pkcolumn.add(columnName);
         }
         for (String haj : pkcolumn) {
            System.out.println(haj + "pk column ");
         }
         int fkcount = 0;
         rs = metadata.getImportedKeys(null, null, actualTable);
         while (rs.next()) {
            fkcount++;
            fkcolumnName = rs.getString("FKCOLUMN_NAME");
            System.out.println("foreignKeys: columnName=" + fkcolumnName);
            fkcolumn.add(fkcolumnName);
         }
         System.out.println("pk=     " + pkcount);
         System.out.println("fk      " + fkcount);
         String columnType;
         int number = 0;
         rs = metadata.getColumns(null, null, actualTable, null);
         while (rs.next()) {
            System.out.println("the table name is:     "+actualTable);
            columnName = rs.getString("COLUMN_NAME");
            number++;
            columnType = rs.getString("TYPE_NAME");
             System.out.println(rs.getString("COLUMN_NAME"));
            System.out.println("column name is"+columnName);
            System.out.println("# of columns is "+number);
}
         for (String pk : pkcolumn) {
            for (String fk : fkcolumn) {
               //if(pk.equals(fk))
               if (pkcount > fkcount) {
                  if (fkcount == 1) {
```

```
                    if(number>2)
                    entity2_table.add(actualTable);
                   Set<String> hs = new HashSet<>();
                        hs.addAll(entity2_table);
                        entity2_table.clear();
                        entity2_table.addAll(hs);
                    }
                }
              }
            }
            for (String m : entity2_table) {
               dlm4.addElement(m);

            }
            jList4.setModel(dlm4);
        }
    }
```

Facts Extraction

```
    public void getColumnsMetadataRF(ArrayList<String> tables) throws SQLException {

        ResultSet rs;
        String columnName32;
        //String kk;
        ArrayList<String> facts32;
        ArrayList<String> facts4;
        dlm4 = new DefaultListModel();
        ArrayList<String> primaryk32;
        String fkcolumnName32 ;
        ArrayList<String> pkcolumn32;
        ArrayList<String> fkcolumn32;
        String my_foreign;
        String fk_original;
        String my_primary;
       // dlm7 = new DefaultListModel();
        ArrayList<String> my_dim;
        ArrayList<String> mydim;
        ArrayList<String> mydim2;
        my_dim = new ArrayList<>();
        mydim = new ArrayList<>();
        facts32 = new ArrayList<>();

        ArrayList<String> datecol;
        ArrayList<String> last_dim;
        ArrayList<String> all_col;
        ArrayList<String> sd;
        ArrayList <String> fk_original_list32;
        fk_original_list32 = new ArrayList<>();
        for (String actualTable : tables) {
            rs = metadata.getPrimaryKeys(null, null,actualTable);

            pkcolumn32 = new ArrayList<>();
            fkcolumn32 = new ArrayList<>();
            primaryk32 = new ArrayList<>();

            mydim2 = new ArrayList<>();
```

```
    datecol = new ArrayList<>();
    last_dim = new ArrayList<>();
    all_col = new ArrayList<>();
    sd = new ArrayList<>();

    while (rs.next()) {
       columnName32 = rs.getString("COLUMN_NAME");
       pkcolumn32.add(columnName32);
    }
    rs = metadata.getImportedKeys(null, null, actualTable);
    int fkcount32 = 0;
    while (rs.next()) {
       fkcount32++;
       fkcolumnName32 = rs.getString("FKCOLUMN_NAME");
       fkcolumn32.add(fkcolumnName32);
    }
    String columnType32;
    String columnType332;
    int numaric_attribute32= 0;
    rs = metadata.getColumns(null, null, actualTable, null);
    while (rs.next()) {
       columnType32 = rs.getString("TYPE_NAME");
       System.out.println(rs.getString("TYPE_NAME"));
       if (columnType32.equals("NUMBER")) {
          numaric_attribute32++;
          System.out.println("the number is    " +numaric_attribute32);
       }
    }
    for (String pk : pkcolumn32) {
       for (String fk : fkcolumn32) {
          if ((pk.equals(fk)) || (numaric_attribute32 > 3)) {
             if (fkcount32 > 1) {
                facts32.add(actualTable);
             }
          }
       }
    }

    System.out.println("\n");
    for (String m : facts32) {
       rs = metadata.getImportedKeys(null, null, m);

       while (rs.next()) {

          fk_original = rs.getString("FKCOLUMN_NAME");
          fk_original_list32.add(fk_original);

          my_foreign = rs.getString("PKTABLE_NAME");
          mydim.add(my_foreign);
       }
    }
 }   // end of actual tables
 for(String fd:facts32)
      for(String xx:mydim){
         if(facts32.contains(xx)){
            facts32.remove(xx);
```

```
                Set<String> hs = new HashSet<>();
                hs.addAll(facts32);
                facts32.clear();
                facts32.addAll(hs);
            }
            }

             for (String my : facts32) {
             dlm4.addElement(my);
             }
               jList4.setModel(dlm4);

    }
```

Dimensions identification

```
public void getColumnsMetadata6(ArrayList<String> tables) throws SQLException {
     ResultSet rs;
      String columnName3;
      String fk;
      ArrayList<String> facts3;
      ArrayList<String> facts4;
      dlm6 = new DefaultListModel();
      ArrayList<String> primaryk3;
      String fkcolumnName3 = null;
      ArrayList<String> pkcolumn3;
      ArrayList<String> fkcolumn3;
      String my_foreign;
      String myforeign;
      String fk_original;
      String my_primary;
      dlm7 = new DefaultListModel();
      ArrayList<String> my_dim;
      ArrayList<String> mydim;
      ArrayList<String> mydim2;
     // ArrayList<String> fk_original=null;
      ArrayList<String> mydim3=new ArrayList<>();

      my_dim = new ArrayList<>();
      mydim = new ArrayList<>();
      facts3 = new ArrayList<>();

      ArrayList<String> datecol=new ArrayList<>();;
      ArrayList<String> last_dim;
      ArrayList<String> all_col;
      ArrayList<String> sd;
      ArrayList <String> fk_original_list = null;
      ArrayList <String> fk_no_room=new ArrayList<>();


      for (String actualTable : tables) {
         rs = metadata.getPrimaryKeys(null, null,actualTable);
         pkcolumn3 = new ArrayList<>();
         fkcolumn3 = new ArrayList<>();
         primaryk3 = new ArrayList<>();
         String fk_origin;
         mydim2 = new ArrayList<>();
```

```
    last_dim = new ArrayList<>();
    all_col = new ArrayList<>();
    sd = new ArrayList<>();
    fk_original_list=new ArrayList<>();

    while (rs.next()) {
       columnName3 = rs.getString("COLUMN_NAME");
       pkcolumn3.add(columnName3);
    }
    rs = metadata.getImportedKeys(null, null, actualTable);
    int fkcount3 = 0;
    while (rs.next()) {
       fkcount3++;
       fkcolumnName3 = rs.getString("FKCOLUMN_NAME");
       fkcolumn3.add(fkcolumnName3);
    }
    String columnType3;
    String columnType33;
    int numaric_attribute3= 0;
    rs = metadata.getColumns(null, null, actualTable, null);
    while (rs.next()) {
       //columnName3 = rs.getString("COLUMN_NAME");
       columnType3 = rs.getString("TYPE_NAME");
       System.out.println(rs.getString("TYPE_NAME"));
       if (columnType3.equals("NUMBER")) {
          numaric_attribute3++;
          System.out.println("the number is    " +numaric_attribute3);
       }
    }
         rs = metadata.getColumns(null, null, actualTable, null);
         while (rs.next()){
          columnType33 = rs.getString("TYPE_NAME");
         // System.out.println(columnType33+"88888888888");
           if((columnType33.equals("DATE"))||(columnType33.equals("TIME"))) {
           String datecolumn33 = rs.getString("COLUMN_NAME");
           datecol.add(datecolumn33);
           Set<String> hs = new HashSet<>();
                hs.addAll(datecol);
                datecol.clear();
                datecol.addAll(hs);
          System.out.println("date column name is    " +datecolumn33);
       }
         }

    for (String pk : pkcolumn3) {
       for (String fkk : fkcolumn3) {
          if ((pk.equals(fkk)) || (numaric_attribute3 > 3)) {
             if (fkcount3 > 1) {
                facts3.add(actualTable);
             }
          }
       }
    }
}

    System.out.println("\n");
```

```
        for (String m : facts3) {
           rs = metadata.getImportedKeys(null, null, m);

           System.out.println("first wrong facts      "+m.toUpperCase());
           while (rs.next()) {
              fk_original= rs.getString("FKCOLUMN_NAME");
              fk_original_list.add(fk_original);
              my_foreign = rs.getString("PKTABLE_NAME");
              mydim.add(my_foreign);
           }
        }
         for(String xx:mydim){
            if(facts3.contains(xx)){
               facts3.remove(xx);
                    Set<String> hs = new HashSet<>();
                    hs.addAll(facts3);
                    facts3.clear();
                    facts3.addAll(hs);
         }
         }
          for (String mm : facts3) {
          rs = metadata.getImportedKeys(null, null, mm);

           System.out.println("real facts       "+mm.toUpperCase());
           while (rs.next()) {
              fk = rs.getString("FKCOLUMN_NAME");
              fk_no_room.add(fk);
              myforeign = rs.getString("PKTABLE_NAME");// write it as FKCOLUMN_NAME
              mydim3.add(myforeign);
           }
        }

                for (String ee : datecol) {
                 mydim3.add(ee);
                 fk_no_room.add(ee);
                }
                 Set<String> hs = new HashSet<>();
                    hs.addAll(fk_no_room);
                    fk_no_room.clear();
                    fk_no_room.addAll(hs);

             System.out.println("the dimensions are:     ");
             for (String aa:mydim3){
         }
           for (String we : mydim3) {
             dlm6.addElement(we);
           }
             for (String tt:fk_no_room)
        {
           dlm7.addElement(tt);
        }
           jList6.setModel(dlm6);
           jList7.setModel(dlm7);
    }
```

Measures Identification

```
public void getColumnsMetadataRM(ArrayList<String> tables) throws SQLException {

 ResultSet rs;
       String columnName3;
        String columnName;
       String fk;
      String columnType;

       ArrayList<String> facts;
       ArrayList<String> facts4;
       dlm6 = new DefaultListModel();
       ArrayList<String> primaryk3;
      ArrayList<String> nonkeymr1;

       String fkcolumnName3 = null;
       ArrayList<String> pkcolumn3;
       ArrayList<String> fkcolumn3;
       String my_foreign;
       String myforeign;
       String fk_original;
       String pk_original;

       String my_primary;
       dlm7 = new DefaultListModel();
       ArrayList<String> my_dim;
       ArrayList<String> mydim1;
       ArrayList<String> mydim2;
      // ArrayList<String> fk_original=null;
       ArrayList<String> mydim3=new ArrayList<>();
       nonkeymr1 = new ArrayList<>();
       my_dim = new ArrayList<>();
       mydim1 = new ArrayList<>();
       facts = new ArrayList<>();

       ArrayList<String> datecol=new ArrayList<>();;
       ArrayList<String> last_dim;
       ArrayList<String> all_col;
       ArrayList<String> sd;
       ArrayList <String> fk_original_list = null;
      ArrayList <String> pk_original_list = null;

       ArrayList <String> fk_no_room=new ArrayList<>();


       for (String actualTable : tables) {
          rs = metadata.getPrimaryKeys(null, null,actualTable);
          pkcolumn3 = new ArrayList<>();
          fkcolumn3 = new ArrayList<>();
          primaryk3 = new ArrayList<>();
          String fk_origin;
          mydim2 = new ArrayList<>();
          last_dim = new ArrayList<>();
          all_col = new ArrayList<>();
          sd = new ArrayList<>();
          fk_original_list=new ArrayList<>();
```

```
    pk_original_list=new ArrayList<>();

    while (rs.next()) {
       columnName3 = rs.getString("COLUMN_NAME");
       pkcolumn3.add(columnName3);
    }
    rs = metadata.getImportedKeys(null, null, actualTable);
    int fkcount3 = 0;
    while (rs.next()) {
       fkcount3++;
       fkcolumnName3 = rs.getString("FKCOLUMN_NAME");
       fkcolumn3.add(fkcolumnName3);
    }
    String columnType3;
    String columnType33;
    int numaric_attribute3= 0;
    rs = metadata.getColumns(null, null, actualTable, null);
    while (rs.next()) {
       //columnName3 = rs.getString("COLUMN_NAME");
       columnType3 = rs.getString("TYPE_NAME");
       System.out.println(rs.getString("TYPE_NAME"));
       if (columnType3.equals("NUMBER")) {
          numaric_attribute3++;
          System.out.println("the number is    " +numaric_attribute3);
       }
    }


    for (String pk : pkcolumn3) {
       for (String fkk : fkcolumn3) {
          if ((pk.equals(fkk)) || (numaric_attribute3 > 3)) {
             if (fkcount3 > 1) {
                facts.add(actualTable);
             }
          }
       }
    }
}

    System.out.println("\n");
    for (String m : facts) {
       rs = metadata.getImportedKeys(null, null, m);
       while (rs.next()) {
          fk_original= rs.getString("FKCOLUMN_NAME");
          fk_original_list.add(fk_original);
         // mydim contains all foreign keys of the wrong fact
          my_foreign = rs.getString("PKTABLE_NAME");
          mydim1.add(my_foreign);
       }
    }
    for (String m : facts) {
        rs = metadata.getPrimaryKeys(null, null, m);
       while (rs.next()) {
          pk_original= rs.getString("COLUMN_NAME");
          pk_original_list.add(pk_original);
    }
```

```
    }
       for(String pkeys:pk_original_list){
         fk_original_list.add(pkeys);

    }

        for(String xy:fk_original_list){
      System.out.println(xy+"88  88   88  88");
    }



    for(String xx:mydim1){
         if(facts.contains(xx)){
            facts.remove(xx);
                 Set<String> hs = new HashSet<>();
                 hs.addAll(facts);
                 facts.clear();
                 facts.addAll(hs);
     }
     }
     for (String m : facts) {

    rs = metadata.getColumns(null, null, m, null);
    while (rs.next()) {
       columnName = rs.getString("COLUMN_NAME");
       columnType = rs.getString("TYPE_NAME");
       if (columnType.equals("NUMBER")) {
          nonkeymr1.add(columnName);
}
}
     }

 for(String as:nonkeymr1){
 }

       for (String c : mydim1) {
         for (String a : fk_original_list) {
           if ((nonkeymr1.contains(c)) || (nonkeymr1.contains(a))) {
           nonkeymr1.remove(c);
           nonkeymr1.remove(a);
       }
    }

    Set<String> hs = new HashSet<>();
    hs.addAll(nonkeymr1);
    nonkeymr1.clear();
    nonkeymr1.addAll(hs);
}
// end of facts3
      for(String dd:nonkeymr1){
}
for (String cc : nonkeymr1) {
    dlm6.addElement(cc);
}
```

```
        jList5.setModel(dlm6);

    }
```

Appendix B

Java Code for Generating Star Schemas from Business Requirements

Initialization phase

```
public static void init() {
      for (int i = 0; i < str.length; i++) {
         for (int j = 0; j < str[i].length; j++) {
            str[i][j] = 0;
         }
      }

   }
```

Viewing phase

```
public void view() {

      for (int i = 0; i < str.length; i++) {
         for (int j = 0; j < str[i].length; j++) {
            //str[i][j] = 0;
            if (str[i][j] == 1) {
               table.setValueAt("   √    ", i, j);
            }

         }
      }

   }
```

Filling Matrix of Requirements

```
ArrayList<String> tokan = new ArrayList<String>();
      StringTokenizer s = new StringTokenizer(tok);
      toka = new String[8];
      ArrayList<String> tota;
      tota = new ArrayList<>();

      String token = "";
      int counter = 0;
      int token_counter = 0;
      //int index = columnNames.indexOf(token);
      int idexToken = 0, c = 0;

      // while (s.hasMoreElements()) {
      counter = 0;
     // token = s.nextToken();
      token = s.nextToken();
      idexToken = 1;
      if (idexToken >= 1) {
         if (idexToken == 1) {
            c = lm.getIndex(token);// the position of the process in the row data
```

```
        }
        ///  read rows
        for (int z = 0; z < lm.getSize(); z++) {
           if (token.equals(lm.getElementAt(z))) { ///  loop   token
              token_counter = 2;
              //idexToken = 2;
              while (s.hasMoreElements()) {
                 token = s.nextToken(); ///  read coulm
                 counter = 0;
                 for (String cl : columnNames) {
                    if (token.equals(table.getColumnName(counter))) {
                       store(c, counter);
                       //  JOptionPane.showMessageDialog(null, lm.getIndex(token));
                       System.out.println("the counter is: " + counter);
                       System.out.println("the token counter is: " + token_counter);
                    }
                    counter++;
                 }// for
                 token_counter++;
                // idexToken++;
              }
           }
        }
     }

     view();
for (int j = 0; j <= 10; j++) //   System.out.println("hhh"+newa[j]);
     {
        for (int x = 0; x <= 18; x++) {  // previous is 18
for (int y = 0; y <= 18; y++) {
              if (x == y) {
table.setValueAt("  blank", x, y);
                 table.setForeground(Color.blue);

              }
           }
        }
     }
```

Appendix C

Java Code for matching Data source schemas with Requirements schemas

Matching Facts

```
public  void factlist() throws IOException {
Schema ds = new Schema("data source");
       ds.addFact(df1);
       ds.addFact(df2);
         for(int i=0; i < ds.fa.size(); i++) {
          String s = ds.fa.get(i).name;
            System.out.println(" data fact name is: "+s);

       }
         for(int i=0; i < sr.fa.size(); i++) {
          String s = sr.fa.get(i).name;
       }

        for (Dimension dim : df1.dim) {
           String ss = dim.name;
           }

        RitaProj ritaobj ;
         boolean flag= false;
         String str="";
         for (Fact f : ds.fa.toArray(new Fact[0])) {
            ds_schema.add(f);
         }
         for (Fact f : sr.fa.toArray(new Fact[0])) {
             Req_schema.add(f);
         }
         for (Fact fr : sr.fa.toArray(new Fact[0])) {
           for ( Fact fd : ds.fa.toArray(new Fact[0])) {
              ritaobj = new RitaProj(fr.name, fd.name);
              System.out.println("n = "+ ritaobj.n );
              System.out.println("v = "+ ritaobj.v );
              System.out.println("yahia   "+ sr.fa.size() );
              if(sr.fa.size()>= ds.fa.size()){
              if((fr.name.equals(fd.name))||(new RitaProj(fr.name, fd.name).n <1 )){
                 for (Dimension dim1 : fr.dim) {
                    for (Dimension dim2 : fd.dim) {
                       if(dim1.name.equals(dim2.name)){
                        LCD.add(dim1.name);
                        LCF.add(dim1.name);
                       }
                     }
                     }
                 flag= true;
                 str=fr.name;
                 System.out.println(" the common is      "+str);
                 break;
              }
```

```
            else
            flag=false;

          }
       if(new RitaProj(fr.name, fd.name).n < 0.3)
          {
            LCF.add(fr.name);
          }
       }
        if (flag == true){
           LCF.add(str);
           CF.add(str);
        }
        else
          LRF.add(fr.name);

       }
          for (Fact r_schema: Req_schema){
          dlm1.addElement(r_schema.name);
          }
             jList1.setModel(dlm1);
           for (Fact r_schema: ds_schema){
          dlm2.addElement(r_schema.name);
          }
             jList2.setModel(dlm2);
          for (String r_schema: LCF){
          dlm3.addElement(r_schema);
          jList3.setModel(dlm3);
          }
         for (String r_schema: CF){

          dlm7.addElement(r_schema);
          }
           jList7.setModel(dlm7);
          for (String w : LRF) {
            dlm4.addElement(w);
            jList4.setModel(dlm4);
          }
         for (String w : LDF) {
              Set<String> hs = new HashSet<>();
               hs.addAll(LDF);
               LDF.clear();
               LDF.addAll(hs);

            dlm5.addElement(w);
            jList5.setModel(dlm5);
            }
    }// end of factlist()//
```

Matching Dimensions

```
public  void dimlist() throws IOException {
        ArrayList<String> LCD= new ArrayList<>();
        ArrayList<String> CF = new ArrayList<>();
        ArrayList<String> LCF= new ArrayList<>();
String ss= "";
```

```
for (Fact fr : sr.fa.toArray(new Fact[0])) {
  for ( Fact fd : ds.fa.toArray(new Fact[0])) {
     ritaobj = new RitaProj(fr.name, fd.name);
     System.out.println("n = "+ ritaobj.n );
     System.out.println("v = "+ ritaobj.v );
     System.out.println("yahia   "+ sr.fa.size() );
     if(sr.fa.size()>=ds.fa.size()){
     if((fr.name.equals(fd.name))||(new RitaProj(fr.name, fd.name).n < 0.3)){
        for (Dimension dim1 : fr.dim) {
          for (Dimension dim2 : fd.dim) {
             if(dim1.name.equals(dim2.name)){
              CF.add(fr.name);
              LCD.add(dim1.name);
             }
            }
            }
            }
            }
            }

           }
     for (String w : LCD) {
        dlm6.addElement(w);
        jList6.setModel(dlm6);
        }

     for (String w : LCF) {
        dlm8.addElement(w);
        //jList3.setModel(dlm8);
     }

}// end of dimlist()//
```

List of Publications

[1] E. Elhaj, F. Jamel. Toward an Ontology Based Approach for Data Warehousing: State of the Art and Proposal. The International Arab Conference on Information Technology (ACIT2014), University of Nizwa, Oman,(2014), pp.170-179.

[2] Elamin E, Alshomrani S, Feki J. SSReq: A method for designing Star Schemas from decisional requirements. In Communication, Control, Computing and Electronics Engineering (ICCCCEE), 2017 International Conference on 2017 Jan 16 (pp. 1-7). IEEE.

[3] E .Elamin, A . Altalhi, and J. Feki. "Heuristic Based Approach for Automating Multidimensional Schemas Construction". International Journal of Computer and Information Technology (ISSN: 2279–0764) Volume. 2017 November.

[4] Accepted paper: A Semantic Resource Based Approach for Star Schemas Matching. IJDMS.