UNIVERSITY OF CINCINNATI

Date: 19th May 2009

I, Monali Deshpande,

hereby submit this original work as part of the requirements for the degree of:

Master in Science

in Department of Computer Science of the College of Engineering

It is entitled:

Automating Multiple Schema Generation using Dimensional Design Patterns

Student Signature: Monali Deshpande

This work and its defense approved by:

Committee Chair: Dr. Karen C. Davis
Dr. Raj Bhatnagar
Dr. Chia-Yung Han

Approval of the electronic document:

I have reviewed the Thesis/Dissertation in its final electronic format and certify that it is an accurate copy of the document reviewed and approved by the committee.

Committee Chair signature: Dr. Karen C. Davis
Automating Multiple Schema Generation Using Dimensional Design Patterns

A thesis submitted to the
Division of Graduate Studies and Research
of the University of Cincinnati

in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in the Department of Computer Science
of the College of Engineering
May 2009

by

Monali Deshpande
Bachelor in Computer Engineering, Mumbai University 2001

Thesis Advisor and Committee Chair: Dr. Karen C. Davis
Abstract

A data warehouse is a repository where data is collected from various sources, integrated, and represented in a dimensional model. The structure and description of the data stored in the repository is called the schema. The two basic approaches for designing a data warehouse schema are data-driven and requirement-driven. Data-driven approaches use operational sources as the guide to creating schemas, while requirement-driven approaches are guided by end-user query and analysis needs. Most approaches in the literature are data-driven; however, some researchers have initiated research into requirement-driven methodologies. Jones and Song [JS05, JS07] propose a requirement-driven approach inspired by design patterns from software engineering. They define dimensional design patterns (DDPs) that capture features common to many dimensional schemas. The use of DDPs assists a designer in creating a dimensional schema. We automate the process of creating one or more star schemas using DDPs and perform case studies to illustrate use of the software tool in different domains. In addition, we automate a process to examine the generated schemas to identify shared dimensions, called conformed dimensions that can be further used by the designer to refine and merge the schemas.
Acknowledgements

I want to express my sincerest gratitude to my advisor Dr. Karen Davis, without her guidance, this thesis would not have been possible. She has been a constant source of knowledge, encouragement, and support. I am thankful to Dr. Raj Bhatnagar, and Dr. Han for serving on my thesis committee. I am grateful to the Computer Science department at University of Cincinnati for providing me with the financial support to carry out my education. My sincere thanks to Julie Muenchen for her constant support. I thank my family, and friends for being there through the thick and thin, and for the encouragement and blessings.
# Table of Contents

Chapter 1: Introduction ...................................................................................................... 1  

1.1 General Research Objective .................................................................................... 1  

1.2 Specific Research Objective .................................................................................... 2  

1.3 Research Plan/Approach ......................................................................................... 2  

1.4 Expected Contributions of the Research .................................................................. 2  

1.5 Overview .................................................................................................................. 3  

Chapter 2: Background and Related Work ........................................................................ 4  

2.1 Overview of Overview of Dimensional Modeling .................................................... 4  

2.2 Approaches for Designing a Data Warehouse ......................................................... 5  

2.3 Requirement Gathering ............................................................................................ 6  

2.4 Feature Summary of Data Warehouse Schema Design Techniques ....................... 8  

2.5 Conclusion .............................................................................................................. 11  

Chapter 3: Design Methodology ...................................................................................... 12  

3.1 Design Patterns ...................................................................................................... 12  

3.2 Dimensional Design Patterns .................................................................................. 12  

3.3 A Star Schema with More than One Fact .................................................................... 18  

3.4 Conformed Dimensions ......................................................................................... 19  

3.5 Conclusion .............................................................................................................. 20  

Chapter 4: DDP to Star Schema Algorithm ....................................................................... 21  

4.1 Generate Dimension Tables Using DDPs ............................................................... 21  

4.1.1 Temporal Dimension Table ............................................................................. 22  

4.1.2 Stakeholder Dimension Table ........................................................................ 25  

4.1.3 Location Dimension Table ............................................................................. 25  

4.1.4 Action Dimension Table ................................................................................ 28  

4.1.5 Object Dimension Table ................................................................................ 28  

4.1.6 Qualifier Dimension Table ............................................................................. 31  

4.2 Algorithm for Designing a Star Schema .................................................................. 31  

4.3. Reporting Conformed Dimensions ..................................................................... 34  

4.4 Conclusion .............................................................................................................. 35  

Chapter 5: Experiments, Results and Discussion ............................................................. 36  

5.1 Development Tools ................................................................................................ 36  

5.2 Software Implementation ....................................................................................... 36  

5.2.1 Generating a First Star Schema ..................................................................... 37  

5.2.2 Generating Second Star Schema .................................................................... 45  

5.2.3 Generating a Third Star Schema .................................................................... 52  

5.3 Conclusion .............................................................................................................. 57  

Chapter 6: Contributions and Future Work .................................................................... 58  

6.1 Overview ............................................................................................................... 58  

6.2 Evaluation .............................................................................................................. 58  

6.3 Future Work .......................................................................................................... 59  

References ...................................................................................................................... 60  

Appendix A: Algorithms ................................................................................................ 63  

Appendix B: Case Studies ............................................................................................. 67  

Appendix C: Program Code ........................................................................................... 88
List of Figures

Figure 2.1. Example Data Warehouse Architecture [CD97] ................................................................. 4
Figure 2.2. Example Grocery Store Star Schema [K03] ........................................................................ 5
Figure 3.1. Temporal DDP .................................................................................................................... 13
Figure 3.2. Stakeholder DDP .............................................................................................................. 14
Figure 3.3. Location DDP ..................................................................................................................... 15
Figure 3.4. Action DDP ........................................................................................................................ 16
Figure 3.5. Object DDP ........................................................................................................................ 17
Figure 3.6. Qualifier DDP ..................................................................................................................... 18
Figure 3.7. Star Schema of Order and Shipments [AV98] .................................................................... 19
Figure 4.1. Flowchart for Determining the DDPs Applicable to a Domain ........................................... 21
Figure 4.2. Generation of Temporal Dimension Table ........................................................................ 23
Figure 4.3. Generation of Stakeholder Dimension Table ..................................................................... 24
Figure 4.4. Generation of Location Dimension Table ......................................................................... 26
Figure 4.5. Generation of Action Dimension Table ............................................................................ 27
Figure 4.6. Generation of Object Dimension Table ............................................................................ 29
Figure 4.7. Generation of Qualifier Dimension Table ......................................................................... 30
Figure 4.8. Pseudo-code to Create Star Schema .................................................................................. 32
Figure 4.9. Generate Fact Table for Star Schema ................................................................................ 32
Figure 4.10. Create Temporal Dimension Algorithm ......................................................................... 34
Figure 4.11. Illustration of Generated Star Schema [AV98] ................................................................. 34
Figure 4.12. Query to Report Repeating Dimensions ......................................................................... 34
Figure 4.13. Metadata Table Definitions ............................................................................................ 35
Figure 5.1. Start Page .......................................................................................................................... 37
Figure 5.2. Capturing the Domain Dimensions .................................................................................... 37
Figure 5.3. Temporal DDP ................................................................................................................... 38
Figure 5.4. Mapping of the Temporal DDP to the Month Dimension Table ........................................ 39
Figure 5.5. Month Dimension Table Created from Temporal DDP .................................................... 39
Figure 5.6. Action DDP ....................................................................................................................... 40
Figure 5.7. Mapping of Action DDP to activity Dimension Table .......................................................... 40
Figure 5.8. Activity Dimension Table created from Action DDP .......................................................... 41
Figure 5.9. Object DDP ....................................................................................................................... 41
Figure 5.10. Mapping of Object DDP to production_line Table ........................................................... 42
Figure 5.11. Production_line Dimension Table Created From Object DDP ........................................ 42
Figure 5.12. Populated Dimension Tables [AV98] ............................................................................. 43
Figure 5.13. Fact Table Questionnaire ............................................................................................... 43
Figure 5.14. Fact table activity_fact Definition .................................................................................... 44
Figure 5.15. The Fact Table activity_fact ............................................................................................ 44
Figure 5.16. Activity measuring activity_fact Created by Software Tool ........................................... 44
Figure 5.17. Multiple Star Schema Generation .................................................................................... 45
Figure 5.18. Define All the Dimensions Required for Star Schema ..................................................... 45
List of Tables

Table 2.1 Comparison of Different Approaches to Design Data Warehouse Schema..................10
A data warehouse is a repository where the data is collected from various sources, integrated, and represented in a dimensional model. This data is used for performing analysis such as market trends, time series, and risk analysis, to name few. The basic function of a data warehouse is to provide a structure with high performance access to support user querying.

Designing a data warehouse is a challenging process as it is not easy to structure vast amounts of data drawn from heterogeneous sources that are often of inconsistent granularity. It is essential to detect the data in a data warehouse that is present in different formats, causing redundancy of information and an unstable granularity. Designers maintain a standard format by conforming data across the data marts. The projects involving data warehouses are complex, large and difficult to design; it is challenging for data warehouse designers to design dimensional schemas that express the structure of the repository.

Two basic approaches in designing a data warehouse are data-driven and requirement-driven. In a data-driven approach, designers use relevant data extracted from heterogeneous sources and design a warehouse that ensures support for user querying. In a requirement-driven approach, requirements from end users are mapped to available data sources.

Requirement analysis plays a key role within any software project to reduce risk of failure [RALT06]. However, the requirement-driven approach is not given much attention in the research literature as it is difficult to list all the requirements at the beginning of the project. It is one of the challenging tasks in the research community to create a design methodology that can initially gather requirements to design a data warehouse.

Jones and Song [JS05] propose a requirement-driven methodology to design a data warehouse. They suggest the use of dimensional design patterns (DDPs) to assist a designer in creating a dimensional schema. In their research, Jones and Song [JS05] examine concepts and entities that occur frequently in dimensional applications. They assert that if these commonly occurring entities and concepts could be abstracted into a structure, it would yield useful guidelines for data warehouse schema designers. Their meta-model is called dimensional design patterns (DDPs). A DDP is a set of design patterns. The DDP approach to design a data warehouse uses both general and domain specific questions to determine concepts that should be modeled in a schema.

The work of Jones and Song demonstrated that the use of DDPs could reduce the time to design a data warehouse schema. Their work is confined to designing a single data warehouse schema manually. Our work is motivated by Jones and Song’s meta-model of DDP.

1.1 General Research Objective

Our focus in designing a data warehouse is how to collect data to support organizational requirements and represent them in schemas. This thesis uses a requirement-driven approach [JS05] to design a data warehouse schema. We chose the DDP meta-model proposed by Jones and Song as they demonstrated their technique to be more effective than manually executing the design process. We automate their approach as a tool that need not be restricted to technical savvy users. We also extend their work by creating one or more star schemas and perform case studies to implement it in various domains. Our goal is to provide end users with a software design tool to create a data warehouse schema using dimensional design patterns (DDPs).
1.2 Specific Research Objective

To develop a software tool to automate the process of generating a data warehouse schema requires several specific research objectives to be addressed:

1. Investigate logical models that can be used to develop a data warehouse.
2. Investigate approaches and identify a method for developing a schema.
3. Develop an algorithm for designing a data warehouse schema.
4. Extend the basic approach to include additional functionality.
5. Determine the platform for implementing the algorithm into a software tool.
6. Investigate the functionalities of the software tool to demonstrate the efficacy of our implementation.

1.3 Research Plan/Approach

In order to achieve our research objective, we perform the following tasks:

1. Review the functionality and objectives of the star schema model and what methods are involved in designing a star schema.
2. Survey the literature and select an appropriate method to design a data warehouse schema; the dimensional design patterns approach [JS05] for creating a dimensional schema is selected. Develop an algorithm that uses six different types of DDPs to create dimension tables and fact tables of a star schema.
3. The first step in our algorithm is to identify the applicable domain DDPs. Once the dimensions are identified, we identify the attributes of the corresponding DDP. We ask the user a series of questions to gather the attributes using a common vocabulary to simplify the process of design.
4. The algorithm is designed so that the user can create one or more star schemas. We extend the algorithm functionality by identifying conforming dimensions.
5. The tool is implemented using Microsoft Visual Studio 2008 as the development platform and Microsoft SQL Server 2005. The coding is done in ASP.NET 3.5 and VB.NET.
6. We perform case studies to demonstrate how the algorithm can be applied to different domains.

1.4 Expected Contributions of the Research

Our research is expected to make the following contributions:

1. Discuss the effectiveness of using dimensional modeling and the underlying star schema model for designing a data warehouse.
2. Compare strategies in data-driven and requirement-driven approaches for designing a data warehouse schema. Choose an efficient requirement-driven approach [JS05] for developing our algorithm.

3. Demonstrate how our software tool is iterated to create one or more star schemas.

4. Demonstrate how the software tool reports to the user about the conformed dimensions existing in the multiple star schemas.

5. Demonstrate how the web application is an interactive and an effective tool for non-technically savvy users to design a star schema.

6. Illustrate how DDPs can be used to create multidimensional schemas over different domains.

1.5 Overview

Chapter 2 gives an overview of methods of designing a data warehouse schema and compares methodologies proposed by different authors. In Chapter 3, we discuss how the star schema is chosen as the underlying conceptual model of our work and how DDPs (dimensional design patterns) help create a user friendly questionnaire to gather requirements from the user. The requirements are then used to create fact tables and dimension tables of a star schema. Chapter 4 discusses six types of DDP structures and our algorithm in detail. Chapter 5 illustrates how the software is developed and describes case studies over different domains. We also demonstrate how the software reports to the user about the conforming dimensions. Chapter 6 gives an overview, evaluation, and ideas for future work.
Chapter 2  Background and Related Work

This chapter discusses different methods for designing a data warehouse. Section 2.1 presents an overview of dimensional modeling, a typical method used in industry for implementing data warehouses. A data warehouse can be designed using a requirement-driven approach or a data-driven approach. Section 2.2 shows how data warehouses are typically designed. Our work involves designing a data warehouse using a requirement-driven approach. Section 2.3 discusses research literature about the requirement-driven approach. Section 2.4 presents comparison of approaches to design a data warehouse design. Conclusions are offered in Section 2.5.

2.1  Overview of Dimensional Modeling

A data warehouse is an integrated collection of data from varied sources for performing analysis to enable decision making. It is therefore essential to design a data warehouse such that it can give an efficient querying performance.

![Example Data Warehouse Architecture](image)

**Figure 2.1 Example Data Warehouse Architecture [CD97]**

Figure 2.1 illustrates the architecture of a typical data warehouse. The data stored in a data warehouse is obtained from operational databases or external data sources. The data warehouse extracts and transforms data for analysis, reporting, or data mining.

Dimensional modeling (DM) is a widely used technique in the industry for implementing data warehouses [K97]. The reason behind its success is because it reflects the way in which the business is viewed. Kimball states “DM is a logical design technique that seeks to present data in a standard intuitive framework that allows for high performance access. It is inherently dimensional, and it adheres to a discipline that uses the relational model with some restrictions” [K97]. It serves as an analytical tool in planning the data warehouse, and as a physical design for its implementation in a relational database.

The underlying logical model of dimensional modeling is a star schema, which consists of a central fact table surrounded by dimension tables. Figure 2.2 illustrates a star schema of a grocery retail store. The
central table, *Grocery Store Retail Fact Table*, is the fact table. The fact table is surrounded by various dimension tables such as *Customer_key*, *Time_key*, and *Store_key*.

![Diagram of Grocery Store Star Schema](image)

**Figure 2.2 Example Grocery Store Star Schema [K03]**

Dimension tables consist of the textual information of the business. The dimension table acts as a source for querying constraints, report labels, and groupings. In Figure 2.2, there are 3 dimension tables on the left side of the fact table and 4 dimension tables on the right side of the fact table. The dimension tables consist of primary keys and attributes that describe each dimension in a highly denormalized form.

Kimball defines the fact table as the primary table in a dimensional model where the numerical performance measurements of the business are stored [KR02]. The fact tables consist of foreign keys and metric attributes (also called measures). In Figure 2.2, the *Grocery Store Retail Fact Table* consists of foreign keys, denoted by (FK), such as *Date Key* or *Customer Key*. The measures form the core of the dimensional model and are data elements that can be summed, averaged, or mathematically manipulated. In Figure 2.2, the measures or the metric attributes are *Dollar Sales*, *Unit Sales*, *Cost Dollar Amount* and *Gross Profit Dollar Amount*. The simplicity of structure offered by the star schema enables efficient query performance.

In Section 2.2 we discuss various methods of designing a data warehouse proposed by different authors in the research community. We discuss both the data-driven approach and requirement-driven approach. We compare requirement-driven approaches based on different criteria in order to relate our approach to others in the literature. The discussion in Section 2.3 gives reasons why we adopt the requirement-driven approach of Jones and Song [JS05].

### 2.2 Approaches for Designing a Data Warehouse

Implementation of a data warehouse involves the following activities: data sourcing, data staging (ETL process), and data presentation or end user querying applications. The data sourcing stage identifies and gathers the transactions of the business from various sources. The data staging phase involves cleaning and standardizing data along with conforming dimensions. “The data presentation stage area is where the data is organized, stored and made available for direct querying by users, report writers and other
analytical applications” [KR02]. Designing and developing a data warehouse is primarily done using two
different approaches: data-driven and requirement-driven.

A data-driven approach develops the data warehouse by analyzing the data sources available. Inmon [I96]
introduced an approach of a very different cycle called CLDS, reverse of the classical SDLC (Systems
Development Life Cycle). SDLC is driven by gathering requirements first, followed by design and
development. Unlike SDLC, CLDS starts with data. “Once the data is in hand, it is tested and integrated,
to check if there is a bias to data” [I96]. Bias in data can be caused by the misuse of data mining
techniques leading to false results. Only after this data is queried and tested are the requirements of the
system understood. Adjustments are made to the design of the system after gathering the requirements
and this process is continued with different set of data. CDLS is referred as spiral development
methodology due to constant resetting of the development life cycle for different types of data. Inmon’s
approach delays requirement analysis based on the principle that data requirements cannot be anticipated
until the data warehouse is partially populated and used by a decision support systems analyst.

If this approach is adopted, the basic corporate data is captured and the needs of the data warehouse users
are taken into account after the data warehouse is built. Giorgini et al. [GRG05] suggest that Inmon’s
approach is feasible when the detailed structure of data sources is known a priori and if the source data is
present in a clean and standardized form. Other researchers recommend that requirements gathering be
done before the design process begins [LST00].

Two other data-driven approaches seek to automate the design process. Song et al. [SKD07] provide the
SAMSTAR method that semi-automatically creates star schemas from an entity relationship diagram by
analyzing its structure. SAMSTAR generates facts from entities having numerical measures and the
textual and non-numerical entities are listed as the dimensions. [PD02] derive an automated technique
that generates candidate data warehouse schema by analyzing schemas of operational data sources.
Romero et al. [RA07] generate multidimensional schemas from an ontology representing different data
sources of a business domain.

A second approach for developing a data warehouse is the requirement-driven approach. Requirement-
driven approaches start by determining the requirements from end users and then investigate how to map
these requirements to develop a data warehouse schema [RALT06]. Requirement analysis determines
what data must be made available to the user, the way it is organized, and how frequently it needs to be
refreshed. Some research has discussed requirement engineering [FS03]. Domain specific aspects such as
representation of facts with their properties, integrating data sources with data, distinguishing dimensions
from facts, assurance of aggregating the results, and documenting the process are addressed. Different
ways of requirement gathering and analysis are reflected in various published works. Section 2.3 details
various approaches proposed by the research community.

2.3 Requirement Gathering

In this section we review the research literature about how requirements can be gathered to develop a data
warehouse.

Winter et al. [WS03] propose a four stage technique for designing a data warehouse schema. The first
stage is identifying users, followed by a requirement analysis aimed at analyzing and describing the actual
information to be represented in a data warehouse schema. In the third stage all the business questions for the regarded management process and any uncovered information is gathered from the target users. Finally a data warehouse schema is developed using the information gathered in the previous three stages.

Use Cases are considered as standard notation for object-oriented requirement modeling, and two different research efforts utilize their technology for designing data warehouses [LST00, BL01]. List et al. [LST00] define different dimension objects such as organization, customer, deliverable and time objects. The fact objects contain performance measures such as turnover, profit, and ratio rates. A communication association is established between the fact and dimension objects to define the typical flow of business processes.

Bruckner et al. [BL01] define the process of breaking down high level business requirement deliverables into requirement attributes. There are no dimension objects and no association with different measures as done by List et al. [LST00]. Instead, their approach is to iterate the process of gathering business requirements, followed by collecting user requirements that are the tasks to be accomplished by the users. The next step is to gather detailed system requirements where fine-grained specifications of requirements which align with the business and user requirements criteria are achieved. The final step gathers requirement attributes that consist of properties, performance requirements, design and implementation instances, and quality attributes.

Prakash et al. [PG01] and Giorgini et al. [GRG05] propose a goal-oriented methodology for developing a data warehouse in which the goal is known a priori. In the Goal Decision Information (GDI) model [PG01], a goal is defined as an objective to be achieved using a decision. This association between goal and decision is called is satisfied by. To arrive at a decision some prior information is needed. The association between decision and information is called as is required for. These two associations form the GDI model that is used to develop data warehouse schema. Giorgini et al. state “though this approach shares some similarities with ours, it mainly focuses on requirement analysis and does not show how to move from requirements to design” [GRG05].

Giorgini et al. [GRG05] follow the Tropos methodology of requirement analysis that can be applied in different application domains. Tropos identifies stakeholders as actors who are dependent on one another to achieve goals. The dependencies are found by answering where, why, what and how questions of system functionality. After these dependencies are identified then fact analysis followed by attribute analysis is performed. An entity relationship or n-ary association can be used for mapping the requirements to design.

Paim et al. [PB03] propose a methodology called DWARF (Data WArehouse Requirements deFinition). In this method the rules and guidelines to gather requirements are defined in the initial stage. The requirement process operates in a cyclic approach of acquisition, representation, and evaluation of requirements from the target users to yield a system specification [PB03]. An advantage of the DWARF method is that it involves the users or clients in the construction of a data warehouse. The specifications are tested for quality standards and integrity constraints. The requirements are documented after they are elicited. They introduce a phase of requirement conformance that is unique to data warehouse specification. This phase ensures that all common system requirements are conformed, i.e., the requirements are identical throughout the data marts. The concept of requirement conformance is discussed in Chapter 3 as this phase is implemented in our software.
Frendi and Salinesi’s [FS03] method suggests that the DW requirements be drawn in a top down manner from the goals to improve business process, requirements of business processes, and strategic decision process of an As-Is data model. An As-Is data model depicts the current situation without incorporating any improvements and a To-Be model results from incorporating improvements in the As-Is model. The existing operational data models can help to create DW models using a bottom up method by eliciting new requirements. This method integrates both top down and bottom up approaches to develop a data warehouse schema.

Tsois et al. [TKS01] describe requirements using real world OLAP scenarios that are derived from an OLAP or data warehousing project. Their work describes a method of using queries to develop a MAC (Multidimensional Aggregate Cube) involving dimensions, levels, hierarchies, measures, and cubes.

Guo et al. [GTTY06] propose a hybrid method that integrates the data-driven and requirement-driven approaches. It is implemented in four stages. The goal driven stage captures the long term goals of the enterprise. The KPIs (Key Performance Indicators) are defined to determine the attributes of the data model. The second step, the data driven stage, identifies the data systems whose data would be fed into the data warehouse. The user driven stage collects the requirements of the target users which represent the measures and dimensions. Finally the outcomes of all the three stages are integrated to develop a schema.

The work of Jones and Song [JS05] is based on the definition of a set of design patterns. They examine concepts and entities that occur frequently in dimensional applications. They assert that if these commonly occurring entities and concepts could be abstracted into a structure, it would yield useful guidelines for data warehouse schema designers. Their meta-model is called Dimensional Design Patterns (DDP). Their design process uses both general and domain specific questions to determine concepts that should be modeled in a schema.

Design patterns are very similar to dimension objects [LST00, BL01]. However, Jones and Song do not confine their patterns to only four types but extend the patterns to six dimension designs. The DDPs are based on the answers of what, when, who, how, where and why similar to the Tropos methodology adopted by Giorgini et al. The DDPs are utilized in gathering requirements from users, who are involved in the design process from the beginning.

Jones and Song assessed efficacy of using DDPs for designing a data warehouse schema. They conducted experiments where the designers worked with and utilized their method. Designers were able to create a data warehouse schema in less time with the use of DDPs.

Jones and Song [JS07] extend their work to add the fact DDP and implementation DDPs in addition to DDPs that generate dimensions. The fact DDP helps the data warehouse designer determine any additional specific facts necessary to fulfill the analysis requirements. It assists the data warehouse designer in classifying the measures in the fact table depending on their additive properties. The data warehouse designer can finally decide if the fact table is a transaction, accumulating snapshot, or periodic snapshot fact table.

2.4 Feature Summary of Data Warehouse Schema Design Techniques

In this section we compare our research work with other approaches to design a data warehouse schema. We include both data-driven and requirement-driven approaches. The methodologies we examine are
published by Inmon [I96], Song et al. [SKD07], Romero et al. [RA07], Winter et al. [WS03], Bruckner et al. [BL01], List et al. [LST00], Prakash et al. [PG01], Giorgini et al. [GRG05], Paim et al. [PB03], Frendi and Salinesi [FS03], Tsois et al. [TKS01], Guo et al. [GTTY06], and Jones and Song [JS05, JS07]. We evaluate our work based on the following features:

1. The data-driven or requirement-driven methodology adopted.

2. Top down or bottom up approach of the technique.

3. Underlying model used for either for requirement analysis or requirement engineering.

4. Whether the method proposed by the author is implemented manually or semi-automatically or automatically.

Table 2.1 illustrates the comparison of different approaches of requirement gathering. In Table 2.1, we use the following notations; ○: partial compliance, ●: full compliance, and -: non compliance.
<table>
<thead>
<tr>
<th>Driven by</th>
<th>Approach</th>
<th>Model used for requirement analysis or requirement engineering</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Requirement</td>
<td>Top down</td>
<td>Bottom up</td>
</tr>
<tr>
<td>Inmon [I96]</td>
<td>●</td>
<td>●</td>
<td>-</td>
</tr>
<tr>
<td>SAMSTAR [SKD07]</td>
<td>●</td>
<td>●</td>
<td>-</td>
</tr>
<tr>
<td>[PD02]</td>
<td>●</td>
<td>●</td>
<td>-</td>
</tr>
<tr>
<td>Ontology [RA07]</td>
<td>●</td>
<td>●</td>
<td>-</td>
</tr>
<tr>
<td>Demand-Driven WS03</td>
<td>●</td>
<td>●</td>
<td>-</td>
</tr>
<tr>
<td>Use Case [BL01]</td>
<td>●</td>
<td>●</td>
<td>-</td>
</tr>
<tr>
<td>Use Case [LST00]</td>
<td>●</td>
<td>●</td>
<td>-</td>
</tr>
<tr>
<td>GDI model [PG01]</td>
<td>●</td>
<td>●</td>
<td>-</td>
</tr>
<tr>
<td>Tropos [GRG05] Giorgini et al.</td>
<td>●</td>
<td>●</td>
<td>-</td>
</tr>
<tr>
<td>DWARF [PB03]</td>
<td>●</td>
<td>●</td>
<td>-</td>
</tr>
<tr>
<td>[FS03]</td>
<td>●</td>
<td>●</td>
<td>-</td>
</tr>
<tr>
<td>Multidimensional Aggregation Cube [TKS01]</td>
<td>●</td>
<td>●</td>
<td>-</td>
</tr>
<tr>
<td>DDP [JS05,JS07]</td>
<td>●</td>
<td>●</td>
<td>-</td>
</tr>
<tr>
<td>Triple driven method [GTTY06]</td>
<td>●</td>
<td>●</td>
<td>-</td>
</tr>
<tr>
<td>Our approach</td>
<td>●</td>
<td>●</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2.1 Comparison of Different Approaches to Design Data Warehouse Schema
Table 2.1 illustrates the similarities and differences between approaches to design a data warehouse schema. The table shows that most of the methods are either manual or semi-automated. Bruckner et al. [BL01] and List et al. [LST00] adopt a UML-based approach using Use Cases. Jones and Song [JS05, JS07] generate a star schema using dimensional design patterns (DDPs). Prakash et al. [PG01], Frendi and Salinesi [FS03], and Giorgini et al. [GRG05] present requirement engineering methodologies that do not emphasize an underlying data warehouse model. Guo et al. [GTTY06] adopt both data-driven and requirement-driven approaches and integrate them to form a hybrid method.

In summary, most of the approaches are either manual or focus only on how to gather requirements. Our approach differs from the other approaches in that we have developed a completely automated software tool to generate a star schema or several.

2.5 Conclusion

Our work is motivated by Jones and Song’s meta-model of DDPs [JS05], an efficient technique for designing a data warehouse schema. Our goal is to provide end users with a software design tool to create a data warehouse schema using design dimensional patterns (DDPs). In this thesis, we create an automated software tool that need not be restricted to use by a technically savvy user. Our software tool creates one or more than one star schema and it is employed in various domains to illustrate execution with published case studies.

In the next chapter, we discuss design patterns in software development, and how this concept is applied to dimensional modeling to create dimensional design patterns (DDPs).
Chapter 3  Design Methodology

In this thesis, we develop a software tool that generates one or more star schemas after gathering the requirements from the user. We use a meta-model proposed by Jones and Song [JS05] called Dimensional Design Patterns. In Section 3.1 we discuss the importance of design patterns in software development and how this concept influenced dimensional design patterns. The six different types of DDPs that influence the work of Jones and Song are discussed in Section 3.2. Section 3.3 illustrates a star schema with more than one fact table. The software reports the conformed dimensions existing in multiple star schemas to the user. The concept of conformed dimensions is discussed in Section 3.4. Conclusions are offered in Section 3.5.

3.1  Design Patterns

Pressman states that “a best designer in any field possesses an ability to see patterns that characterizes a problem and corresponding patterns that can be combined to create a solution” [P97]. The purpose of a pattern is to form a structure that captures and resolves commonly occurring problems encountered in software development. Object-oriented design patterns show relationships and interactions between objects or classes. “The designers find recurring patterns of classes and communicating objects in many object-oriented systems. These patterns solve specific design problems and make object-oriented design more flexible and reusable” [GAM95].

Jones and Song utilize the concept of design patterns in their research. They apply the concept of design patterns in dimensional modeling as a mechanism for communicating ideas between designers, programmers and non-programmers. Dimensional Design Patterns form a cohesive language to capture commonly occurring entities in dimensional modeling that can further assist a designer in designing a schema.

3.2  Dimensional Design Patterns

In their research Jones and Song [JS05, JS07] examine concepts and entities that occur frequently in dimensional models. They propose a meta-model called Dimensional Design Patterns (DDP). DDPs assist designers by identifying commonly used dimensions. They are based on the answers of what, when, who, how, where and why. The questions integrate various aspects, found in dimensional modeling, in a structured way.

The six different types of DDPs introduced by Jones and Song are Temporal, Stakeholder, Action, Object, Location, and Qualifier. We describe each DDP below with the help of figures.

The Temporal DDP models different time periods present in an organization. Along with the calendar periods of months, weeks, quarters and years, there may be fiscal periods or audit periods and holiday period of the company that need to be documented in the data warehouse. Figure 3.1 illustrates the Temporal DDP.
The Stakeholder DDP identifies a stakeholder present in an organization or a role assigned to handle specific tasks and responsibilities. "A stakeholder can be associated with and described by an organization or a role. An organization is considered a group of people bound by common work, goals, or interests while a role describes a specific tasks and a set of responsibilities" [JS05]. This DDP is associated with other DDPs to identify any relative information between two entities. For example there can be an action with respect to a stakeholder and the designer may wish to capture it in the dimension table or there can be a location with respect to this DDP. An example would be capturing information of a manager (a role) present in one of the branches of a company. The information of this role (Stakeholder DDP) would be for the particular branch (Location DDP). Figure 3.2 illustrates Stakeholder DDP.
The Location DDP identifies various branches of an organization in the dimensional model. All other dimensions are associated with a Location DDP. An example would be the headquarters of a company might have a manufacturing unit set up next to it. In this case if we wish to capture all activities that are carried out at headquarters, then an Action DDP with respect to a location DDP is presented to a user to gather information. Figure 3.3 illustrates the Location DDP.
The Action DDP models behaviors or the accomplishments of a conceptual or physical entity such as a person or an organization. It depicts the behavior or work accomplished. For example, a credit card payment can be considered as an action dimension that involves steps such as initiate the transaction, authorize the credit card holder and accept payment. This DDP is also associated with other DDP to gather the time and the place where the action was carried out or the stakeholder who performed the action. Figure 3.4 illustrates Action DDP.
The Object (Physical/Conceptual) DDP is useful in defining an abstract concept or a physical object. The conceptual object dimension answers the “how” question asked to the user while the physical object answers the “what” question. Examples of conceptual objects are different types of bank accounts such as savings, checking, revenue accounts, or insurance policies [JS05]. Examples of physical objects would be products or items or a manufacturing unit or office space. Figure 3.5 illustrates the Object DDP.
If the user wishes to gather the profile, cause, particular state or measurement aspect of a dimension then the Qualifier DDP can be utilized. If the user wishes to describe the size of the office space or number of floors of a building, he or she can use the Qualifier DDP. The user can enhance the profile properties of a dimension using the profile information of this DDP. “The state describes the circumstances characterizing a particular condition at a point in time” [JS05]. If an event has occurred, the user can record the cause of the event in this DDP. Figure 3.6 illustrates the Qualifier DDP.

The DDPs described in this section can be used to create a star schema with a single fact table. The next section describes a situation where multiple fact tables can arise, leading to the need for conforming dimensions.
3.3 A Star Schema with More Than One Fact Table

Our algorithm is iterated to create more than one star schema. Adamson and Venerable state there can be instances during data warehouse schema design when all the facts and dimensions in a single star schema are not captured [AV98]. It occurs frequently in practice that multiple fact tables and multiple star schemas are needed. The reason for having multiple facts may be due to the definition of a business process that is discrete and separates the measures into disparate fact tables.

When processes are measured at various grains, some of the attributes in the dimension tables might overlap or involve additional dimension values. Due to the additional dimension values, the business process is viewed in a different way along with the difference in the method of assembling the measures. Adamson and Venerable illustrate this concept by using the example of shipment and order fact tables. In Figure 3.7, the Shipment_facts table is different from the Order_facts table as it has measures shipper_key, sales_dollars and shipment_date_key that focus on where, how, and when the order is shipped rather than when, how, and by whom the order was placed.

Adamson and Venerable assert that if the business processes are measured at different times, multiple fact tables can emerge [AV98]. Referring to the Figure 3.7, quantity_ordered and quantity_shipped are two different measures. The quantity_shipped measure may differ from quantity_ordered as the order may be shipped in several batches rather than a single batch. The order might be shipped in batches with the order received on one date and shipped on multiple dates. Although we refer to the same order, the order date
and quantity have to be captured and noted in different fact tables than the shipment date. The next section describes how overlapping dimensions can be conformed.

Figure 3.7 Star Schema of Order and Shipments [AV98]

3.4 Conformed Dimensions

In Section 3.3, we discussed an instance of a star schema with more than one fact table. When multiple star schemas exist, one dimension may be present in more than one star schema. In Figure 3.7, dimension tables *Date, Product, Customer,* and *Salesperson* are referenced by both the *Order* and *Shipments* fact tables.

“A dimension is said to be conformed when it means the same to every attached fact table” [PB03]. Sometimes the same dimensions are expressed in different formats, causing redundancy of information,
and an inconsistent granularity over the data warehouse. In order to correctly include multiple fact tables in the same schema conforming the dimensions is necessary [KR02].

The data presented in one form may be finely grained data while the same data in a different form may not be expressed at the same granularity. For example, the data attribute can be expressed as “26 January 2009 10:39:00” (DD/MM/YYYY Hr:Min:Sec) in one dimension table and “January 2009” (MM/YYYY) in another dimension table. The first dimension table has more fine grained data (DD/MM/YYYY Hr:Min:Sec) than the second dimension table (MM/YYYY). In order to combine the fact tables into one multi-fact schema a standard format of attributes across all data marts is needed to avoid inconsistent granularity.

Kimball explains conformed dimensions as the dimensions with exactly the same dimension attributes or at least a proper subset of one another. Conformed dimensions have consistent dimension keys, consistent attribute column names, consistent attribute definitions, and consistent attribute values [KR02]. If the attribute values of dimension table that repeats in two different schemas do not match, then the dimension tables are not conformed.

Conforming dimensions can lead to a successful data warehouse environment and also help to build distributed data warehouses. If the dimensions are conformed, it allows new data sources to be added to the existing warehouse, allowing different application domains to be compatible with each other and function in conjunction.

3.5 Conclusion

We use Jones and Song’s DDP meta-model to gather requirements from the user, and iteratively gather additional requirements to create more than one star schema. After the star schemas are created, the software tool reports the conformed dimensions to the user. In the next chapter, with the help of our algorithm, diagrams and an example, we explain the implementation of this concept.
Chapter 4  
DDP to Star Schema Algorithm

Chapter 3 investigates approaches of different authors that have inspired the design of our software. The software uses Dimensional Design Patterns (DDPs) to gather requirements to develop one or more dimensional schemas and derive conformed dimensions.

This chapter discusses in detail how the software makes use of DDPs to collect requirements. Section 4.1 shows the use of six different DDPs to generate subsequent dimension tables and how they are transformed into a question format in the software. Section 4.2 discusses the algorithm in a modular format. Section 4.3 shows an example schema created by selecting the DDPs and its attributes.

4.1 Generating Dimension Tables Using DDPs

A DDP is a structure that helps identify the requirements from the user to design a star schema. The schema can be further refined. If in the initial period of the data warehouse design, if any particular measure is not present, it still can be added after a first draft of the design is completed, instead of redesigning a new data warehouse. An example would be the addition of an audit period that was not initially present in the temporal dimension table; it can be added during subsequent reviews and revisions.

The flowcharts illustrated in this section explain the six DDPs that are used in the algorithm and how they help develop dimension tables. We explain how the DDPs from Chapter 3 are embedded in algorithm to generate dimension tables. The flowcharts of DDPs explain how and what data is gathered from the user to design a star schema. The input/output of the flowchart is represented by a parallelogram. The arrows show the flow of the algorithm and the rectangles represent the processing steps. The diamond shaped decision box indicates a question or branch in the flow chart. Typically a decision box is used when there are two options such as Yes or No. At the end of every flowchart, we have a connector that connects to the next DDP implemented in the algorithm. In this section we explain how each DDP develop its corresponding dimension tables.

Figure 4.1 explains the first step of implementing DDPs to gather requirements in our algorithm. The first step of the algorithm is to identify the DDPs present in the user’s application domain; these DDPs are used to generate the dimension tables of the star schema. Once the DDPs are identified, the next step is to address each DDP and then pose the questions to the user to gather attributes of the dimension tables. After gathering the attributes, we create a dimension table and assign a primary key to the table. An example schema generated by the software with some dimensions present is given in Section 4.2.
4.1.1 Temporal Dimension Table

The Temporal DDP is used to find different time periods utilized by an organization. Fiscal periods can be differentiated from calendar periods. Special events, including weekend indicators or audit periods and holidays, can be captured by this DDP.

Figure 4.2 illustrates the structure of the Temporal DDP and how this DDP helps identify temporal data to develop a dimension table. Once the user confirms the presence of a Temporal DDP, he or she is asked if the Calendar period, Fiscal period, Timestamps or Special period is present. If a user confirms the presence of a Calendar period, then the user is asked the questions with respect to the Calendar period; detailed requirements are gathered at this point. The process is repeated for Fiscal period, Timestamps and Special periods. After the user approves the submission of dimensions and their attributes, a primary key is determined and a temporal dimension is created.

The user is prompted if he or she wishes to create any more temporal dimension tables. If the user approves another temporal dimension table to be created then the above steps are repeated. In Figure 4.2, an arrow shows the iteration for creating more temporal dimension tables.
Figure 4.2 Generation of Temporal Dimension Table
Figure 4.3 Generation of Stakeholder Dimension Table
4.1.2 Stakeholder Dimension Table

Figure 4.3 illustrates the structure of the Stakeholder DDP and how this DDP helps identify stakeholder data to develop a dimension table. Figure 4.3 also shows how the DDP helps identify the information of an organization, the location where the organization is present, people participating in this organization (role) and duties performed by them, and the work carried out by the organization. At times the organization or the manager (role) is involved in some events that occur at some location. This detailed information is captured in this DDP. After the user approves the submission of dimensions and their attributes, a primary key is determined and a stakeholder dimension is created.

The user is prompted if he or she wishes to create any more stakeholder dimension tables. If the user approves another stakeholder dimension table to be created then the above steps are repeated. In Figure 4.3, an arrow shows the iteration for creating more stakeholder dimension tables.

4.1.3 Location Dimension Table

Figure 4.4 illustrates how the Location DDP is used to group together the attributes related to particular site and develop a location dimension table. The DDP identifies if a specific locale is present and gathers the street, city, zip code, province, region, country, county, and state of the locale. If a primary work is accomplished at the locale then the name or title and the definition of the work is gathered from the user. Further, the time information, event information for this locale, and people working at that site are identified in this DDP. The questionnaire implementation prompts the user about whether any office is present at the location and also examines the configuration details such as number of floors, any identification given for the office (for example, headquarters) and the square feet of the office. The contact information of this location such as URL, phone number, and fax number is identified in this DDP. After the user approves the submission of dimensions and their attributes, a primary key is determined and a location dimension is created.

The user is prompted for whether he or she wishes to create any more location dimension tables. If the user approves another location dimension table to be created then the above steps are repeated. In Figure 4.4, an arrow shows the iteration for creating more location dimension tables.
Figure 4.4 Generation of Location Dimension Table
Figure 4.5 Generation of Action Dimension Table
4.1.4 Action Dimension Table

Figure 4.5 illustrates how the Action DDP is used to develop an action dimension table. The Action DDP identifies the behavior, the person involved in the action, if any steps were involved in the action, and the outcome of the act.

The temporal data associated with the action and the location where the action has occurred is gathered from the user. The organization, or role or a person responsible for the occurrence of the event is identified by associating the stakeholder DDP to the action DDP. Finally the questionnaire implementation of qualifier DDP prompts the user to collect the causal events that occurred due to a particular action (example profit can be causal event of action sales); any profile information of the event and configuration would give detailed measurements of the action. After the user approves the submission of dimensions and their attributes, a primary key is determined and an action dimension is created.

The user is prompted if he or she wishes to create any more action dimension tables. If the user approves another action dimension table to be created then the above steps are repeated.

4.1.5 Object Dimension Table

Figure 4.6 illustrates the structure of the Object DDP and how it helps develop an object dimension table. The Object DDP is used gather attributes of a conceptual or physical object. Once the user confirms the presence of an object dimension table, the name of the object, the owner of the object and the responsibility or task performed by the object are gathered from the user. The temporal data related to the object, the locale attributes, and action attributes associated with respect to the object dimension are identified. For example, for a conceptual object such as a savings account, the information of the money withdrawn (action) from a branch account (locale) on a particular day (temporal) is identified by an Object DDP. After the user approves the submission of dimensions and their attributes, a primary key is determined and an object dimension table is created.

The user is prompted if he or she wishes to create any more object dimension tables. If the user approves another object dimension table to be created then the above steps are repeated.
Figure 4.6 Generation of Object Dimension Table

Any object related data present?

Any owner of object?
- Identify name
- Identify role
- Identify responsibility

Any time period associated?
- Identify time and date
- Identify specific time period
- Identify fiscal period
- Identify life span of object
- Identify calendar period
- Identify timestamp

Any action associated?
- Identify name of event
- Identify date of event
- Identify time of event
- Identify purpose of event

Any profile information?
- Identify condition
- Identify status
- Identify causal info
- Identify configuration

Any locale associated?
- Identify specific locale
- Identify work accomplished
- Identify date/time
- Identify contact info.

Gather the Object DDP attributes to create Object dimension table

Identify the primary key
Create a Object dimension table

Any additional Object dimension tables present?
- Yes
- No
Figure 4.7 Generation of Qualifier Dimension Table
4.1.6 Qualifier Dimension Table

The Qualifier DDP is classified into four different groups: profile information, specific state, cause, and unit indicators. The profile information gives the identifying characteristics, the state gives particular circumstances that need to be captured, the cause gives the reason for the occurrence of the event, and the unit indicators define measurement data and their associated units of measure. After the user approves the submission of dimensions and their attributes, a primary key is determined and a qualifier dimension table is created. The user is prompted to create any more qualifier dimension tables. If the user approves another qualifier dimension table to be created, then the above steps are repeated.

In Section 4.2 we discuss the algorithm in pseudo-code. We discuss in detail how star schemas are generated using the DDPs and how the report of conformed dimensions is generated.

4.2 Algorithm for Designing a Star Schema

The purpose of our algorithm is to implement the DDP questionnaire (represented as flowcharts in the previous section) and map user answers to a star schema. One of the advantages of using our software is that the user can be a novice in data warehouse design. The user need not be a designer but he or she should be aware of the domain for which the star schema is created. Whenever the software is used, the user should be knowledgeable of the application domain for which the dimensional schema is developed. The purpose of this software is to create the initial schema output by the software that can be refined after reviewing it.

The input is the list of questions asked to the user and yes or no responses taken from the user for the questions asked. The output consists of multiple star schemas and a report of the repeating dimensions in multiple schemas. The dimension table created would have columns generated at the run time with pre-defined data types for each column.

This algorithm is designed to create a star schema using a four step approach [KR02] proposed by Ralph Kimball. The first two steps of dimensional modeling design are (1) selecting the application domain knowledge, and (2) selecting the grain size. These aspects are captured in our algorithm. To design the star schema we make use of the dimensional design patterns described in Section 4.1. The main objective of this algorithm is to execute Steps 3 and 4 proposed by Kimball of designing dimension tables and fact tables to create a star schema. We extend the functionality of this algorithm to create more than one star schema and then perform joins on all the available star schemas to find conformed dimensions.

Algorithm 4.2.1 Create Star Schemas

1: Input: responses // answers to a question
   column header names // text string to be used in generating fields in the dimension tables
   table name // text string to be used to name the dimension table
2: Output: one or more star schemas
3: repeat
4:   repeat
5:     if Temporal DDP selected then Create Temporal Dimension()
6:     if Location DDP selected then Create Location Dimension()
7:     if Stakeholder DDP selected then Create Stakeholder Dimension()
8:     if Action DDP selected then Create Action Dimension()
9:     if Object DDP selected then Create Object Dimension()
10:    if Qualifier DDP selected then Create Qualifier Dimension()
11:   until user indicates there are no additional dimension tables
12:   Create fact table using table name and the measures gathered from the user
13: end repeat
14: end repeat
14: until user indicates that there are no additional star schemas
15: Prompt user: report conformed dimensions or exit
16: If user selects to report conformed dimensions then call stored procedure Conform () to report the
   conforming dimensions
18: Exit

Figure 4.8 Pseudo-code to Create Star Schemas

Figure 4.8 illustrates the pseudo-code of an algorithm to generate star schemas by gathering requirements from a user. For every star schema that is developed, the algorithm creates dimension tables using the DDPs. Figure 4.9 demonstrates the process of developing a star schema after the dimension tables are created. After the dimension tables are created, the algorithm generates a fact table that consists of the entire primary keys along with any more measures that the user may wish to summarize. A star schema with a fact table along with the dimension tables is generated. After one star schema is created, the algorithm has the ability to iterate so that the user can create more than one star schema, as shown in Figure 4.9. In the last step of the algorithm, the user may opt to generate a report of the conformed dimensions from the multiple star schemas or exit the algorithm.

Figure 4.9 Generate Fact Table for Star Schema
Figure 4.10 illustrates the use of a DDP to develop a dimension table. The user is asked a series of questions to gather the attributes of the dimension table. The “identify” keyword used in the algorithm refers to the series of questions asked to the user and the corresponding attributes gathered from the user. After the attributes are gathered, the user submits the name of the dimension table and a dimension table is generated. The attributes form the column headers or the fields in the dimension table. The mapping of the DDP to the dimension table or the fact table is illustrated in detail in Chapter 5. Appendix A contains the algorithms for the remaining DDPs.

**Algorithm 4.2.2 Create Temporal Dimension**

1: Input: responses // answers to questions
    column header names // text string to be used in generating fields in the dimension tables
    table name // text string to be used to name the dimension table
2: Output: one or more star schemas
3: repeat
4:     if user selects Temporal Dimension then
5:         if user selects calendar period then prompt for the following attributes
6:         identify century
7:         identify decade
8:         identify year
9:         identify quarter
10:        identify month
11:        identify month number
12:        identify week
13:        identify week number
14:        identify day
15:        identify day number
16:        identify date/time
17:     end if
18:     if user selects fiscal period then prompt for the following attributes
19:         identify fiscal year
20:        identify fiscal quarter
21:        identify fiscal quarter number
22:        identify fiscal month
23:        identify fiscal month number
24:        identify fiscal week
25:        identify fiscal week number
26:        identify fiscal date/time
27:     end if
28:     if user selects timestamps then prompt for the following attributes
29:         identify hour
30:        identify minute
31:        identify second
32:        identify timestamp
33:     end if
34:     if user selects special period then prompt for the following attributes
35:         identify special event (description)
36:        identify weekend
37:        identify holiday description
38:        identify audit period
39:     end if
40: end repeat
40:      end if
41:      input from the user the temporal dimension table name table name
42:      generate a primary key using the table name
43:      create dimension table using table name
44:      until user indicates that there are no additional dimensions

Figure 4.10 Create Temporal Dimension Algorithm

An automatically generated star schema based on user selection, is shown in Figure 4.11.

Figure 4.11 Illustration of Generated Star Schema [AV98]

4.3 Reporting Conformed Dimensions

The stored procedure Conform() is invoked when the user opts to list the conform dimensions. This list helps the user to analyze those dimensions, and to maintain a standard format of attributes across all star schemas to avoid inconsistent granularity. Figure 4.11 illustrates the code that is used to output the conformed dimensions from multiple star schemas. We compare every dimension of one star schema with other star schemas using the metadata table.

Figure 4.12 Query to Report Repeating Dimensions

The stored procedure Conform() uses the metadata table that contains the name of the generated table along with the primary key, the star schema to which the table belongs, and a Boolean variable called isfact. The variable isfact is true for a fact table and false for a dimension table. The column starnumber helps identify the star schema to which the dimension table and fact table belongs. For example a
dimension table customer repeating in the first and third star schema would have starnumber 1 and 3 associated to it. Figure 4.12 illustrates the metadata table definition.

![Image of metadata table definitions]

**Figure 4.13 metadata table definitions**

The output of the stored procedure Conform() consists of two columns of dimensions: dimension and repeating dimension. The dimension column is drawn from the first schema, while the repeating dimension column lists identical dimensions in other schemas. An example is given in the next chapter.

### 4.4 Conclusion

A detailed illustration of how questions illustrated here implement a requirement gathering process to create dimension and fact tables of a star schema is given in Chapter 5. Our work contributes the capability to create more than one star schema and identify and report conformed dimensions. We test our software with case studies over different domains such as production and manufacturing.
Chapter 5  Experiments, Results and Discussion

In order to provide better insight into the functionality of our software, we have implemented case studies over various domains. The objective of these case studies is to investigate the effectiveness of the proposed algorithm, study the mapping of DDPs to create star schemas, and report the conformed dimensions. In this chapter we illustrate only one case study. Appendix B contains the illustrations of two other case studies that utilize additional features of the DDPs not shown here. Section 5.1 outlines the features of the development platform. Section 5.2 explains the functionality of the GUI, and the implementation of our algorithm in software. Section 5.3 gives the conclusion of this chapter.

5.1  Development Tools

The software is implemented with Microsoft’s Visual Studio 2008 IDE (Integrated Development Environment) using the ASP.Net 3.5 framework and VB.NET as programming language. SQL Server 2005 is used as our backend database where we create and store star schemas in the form of dimension and fact tables.

VS 2008 (Visual Studio 2008) is an Integrated Development Environment used to create a web-based implementation of our algorithm. ASP.NET is a web application framework developed and marketed by Microsoft to allow programmers to build dynamic web sites, web applications, and web services. One of the features of the ASP.NET framework is a Toolbox that consists of checkboxes, textboxes, dropdown lists, and radio buttons that help a developer to create an interactive front end for a web application.

Our application is written in Visual Basic .NET (VB.NET). It is an object-oriented computer language that supports key concepts such as polymorphism, inheritance, abstraction and encapsulation. One of the advantages of VB.NET other than object-oriented programming is the way the database is accessed. VB .NET uses ADO.NET (Active X Data Object) as its data access and manipulation protocol. SQLConnectionName.Open() is a protocol for opening the connection with the database, performing required operations and then disconnecting the database connection using SQLConnectionName.Close(). Program code appears in Appendix C.

5.2  Software Implementation

We illustrate how our software creates more than one star schema and then report the conformed dimensions. We proceed with an example that does not employ all of the six types of DDPs. In Appendix B we demonstrate how the DDPs that are not a part of the following example can gather attributes of the dimension tables in other schemas. Appendix C shows how our software can be applied to other domains such as Order/Shipment [AV98] and Supplier Performance Schema [AV98].

We show how the DDPs are presented in the form of questionnaire and how they help gather the attributes of the dimension table in Chapter 4. The following explanation illustrates how a star schema is generated by mapping the questions of the DDPs to dimension and fact tables using the software and how the conform dimensions are reported to the user.

We demonstrate an example of a data warehouse schema of a production system that supports processes such as manufacturing.
5.2.1 Generating a First Schema

When the application is run, we have a start page (Figure 5.1) that gives an introduction, the objective of the application, and the author’s name.

After the user clicks the button *Get started with your design*, the start page directs to the page where the user can select the dimensions related to his or her application domain, shown in Figure 5.2.

A dimension is a characteristic of data that allows it to be summarized (such as sales data summarized by a month dimension). Please indicate which of the following dimensions are used to summarize your data.

- **Temporal**: This dimension can be used to capture data over fiscal periods, special audit periods, and calendar periods.
- **Location**: This dimension is used to gather information about different locations, such as office locations or territory/region of a company.
- **Stakeholder**: This dimension summarizes data for an organization, different roles involved, and designations of people, for example.
- **Action**: This dimension captures different activities, functions, and processes involved in an event. A payment is an example of an action dimension item.
- **Object**: This dimension helps define a conceptual (e.g., revenue accounts) or physical (e.g., a product) object used as a feature to summarize data.
- **Qualifier**: This dimension contains descriptive information about another dimension, such as demographics about a customer or details about an office such as space allocated.

With the click of the *Submit* button, the user is directed to the first dimension that is selected. In this case, it is the temporal dimension.

Figure 5.3 illustrates how the questionnaire format of the Temporal DDP is used to generate a temporal dimension table called *Month*.
To create the Temporal dimension called *month*, the user first fills in the textbox with the name to be used for the dimension table in the final data warehouse schema, which is “month” in this case. Then all the dimension attributes are captured by checking the appropriate checkboxes. The user first confirms the existence of the calendar period and checks the following attributes: *month_name*, *month_number*, *quarter* and *year*.

Figure 5.3 Temporal DDP
After the dimension attributes are gathered and the create dimension table button is clicked, a create table command is executed that creates the dimension table month with month_key as the primary key and the checked attributes as the columns. Figure 5.4 gives the month table definition that is automatically created by our software. The output of this iteration is the temporal dimension called month of the star schema as shown in Figure 5.5.

![Figure 5.4 Mapping of the Temporal DDP to the Month Dimension Table](image)

After the create dimension table button is clicked, the page is redirected to the same page. If there is more than one temporal dimension present in the star schema it can be described by the user at this point. Once the author confirms that all temporal dimensions are gathered, with the click on proceed to next dimension button he or she proceeds with specifying the next dimension. The next dimension table is Activity that is an Action dimension.

![Figure 5.5 Month Dimension Table Created From Temporal DDP](image)

After the create dimension table button is clicked, the page is redirected to the same page. If there is more than one temporal dimension present in the star schema it can be described by the user at this point. Once the author confirms that all temporal dimensions are gathered, with the click on proceed to next dimension button he or she proceeds with specifying the next dimension. The next dimension table is Activity that is an Action dimension.

Figure 5.6 illustrates the Action DDP used to generate the action dimension table. The user enters the name of the action as activity. The type attribute is the profile information of the action activity. The checkbox for associating the qualifier dimension with respect to the action dimension is checked. The profile information type is used in accordance with the activity.
The primary key `activity_key` is automatically created with the click of `create dimension table` button. The `activity` dimension table is created with `activity` and `type` as its attributes. Figure 5.7 gives the `activity` table definition that is automatically created by our software.

The output of this iteration is the action dimension called `activity` of the star schema as shown in Figure 5.8.
The next dimension is *production_line* that is object dimension. Figure 5.9 illustrates the Object DDP used to generate the object dimension table. It is common for a factory to maintain more than one production line dimension [AV98]. This dimension captures the *name* of the production line, the *role* of the production, for example it can be used for desktop or notebook production. The *type* of the production is the profile information of the production line hence it captured by the qualifier dimension in association with the object dimension. A *type* can be either a flexible or standard method for production. The location of the production is captured by the country and the facility attribute is used for the location in the country. For example *facility* can be Madrid that is located in *country* Spain.

Figure 5.9 Object DDP

Figure 5.10 gives the *production_line* table definition with *production_line_key* as the primary key that is automatically created by our software.
The output of this iteration is the action dimension called *production_line* of the star schema as shown in Figure 5.11.

In Figure 5.12 we demonstrate how the table may be populated with data after the dimension table is created. The data is given by Adamson [AV98]. The attributes that are captured from the DDP form the column headers of the table.
After the entire dimensions of the first star schema are collected, the next step is to create a fact table from the primary keys of the dimension tables and the measures added by the user. Figure 5.13 illustrates the questionnaire format to develop a fact table.

An additional measure *hours* is included in the fact table *activity_facts*. We maintain a list, *newArr[]*, that stores all the primary keys through the run of the web application. The measure inserted by the user is added to the arraylist *newArr[].* This array list is used in the CREATE table command to create the fact table. Once the user clicks on the *create fact table* button the fact table is created as shown in Figure 5.15. Figure 5.14 gives the *activity_fact* table definition that is automatically created by our software.
Figure 5.14 Fact Table activity_fact definition

The output of this iteration is the fact table called activity_facts of the star schema as shown in Figure 5.15.

Figure 5.15 The Fact Table activity_fact

Figure 5.16 Activity measuring star schema created by software tool

Figure 5.16 shows the star schema generated as the output of all the questions of the DDP answered by the user. After the fact table of first star schema is created, the user is directed to the following page (Figure 5.17), where he or she is informed of the star schema being created. The user is given two options to choose. He or she can either proceed to form another star schema by clicking the button create another star schema. Else if he or she has already created more than one star schema he or she can click the button report the conformed dimensions where he or she will see those dimensions that are repeating in more than one star schema.
If the user clicks on create another star schema button, he or she is directed to the first page where he or she declares the dimensions present in the second star schema. The user proceeds to create a second star schema.

### 5.2.2 Generating a Second Schema

Figure 5.18 illustrates the first page of the software that shows the dimensions required in the generation of star schema. Three dimensions: production_line as the Object dimension, product as the Physical Object dimension, and time as the temporal dimension are selected.

The application directs to the temporal dimension. The time dimension consists of calendar period present with date, month_name, month_number, quarter, year, day_of_month and day_of_week as its attributes. The day_of_week is different than day_of_week. The day_of_week can be 1 for Sunday, 2 for Monday, while day_of_month can be 21 for 21st day of the month.

It can be noted that the temporal dimension “time” has special period present. This DDP helps gather weekend and holiday as the temporal dimension attributes. With this special period attributes, the user can keep record of the presence or absence of the holidays or weekends. Figure 5.19 shows how the questions help gather the attributes of Temporal dimension table.
Input for the *Temporal* dimension

What would you like to call your *Temporal* dimension? This name will be a table in the final data warehouse design. **time**

Indicate the ways that you would like to summarize your data by checking boxes below.

Is Calendar period present?  
- century
- decade
- year
- quarter
- month name
- month number
- week
- week number
- day of week
- day of month
- date time

Is Fiscal period present?  
- fiscal year
- fiscal quarter
- fiscal month
- fiscal week
- fiscal date time

Is a Timestamp present?  
- hour
- minute
- second
- timestamp

Is a Special Period present?  
- event name
- weekend
- audit period
- holiday

Please fill in the boxes below with what you would like to call your special periods in the final data warehouse design. For example, you may have an event that is called tax week.

**Figure 5.19 Temporal DDP for Second Star Schema**
Figure 5.20 Mapping of Temporal DDP to Time Dimension table

Figure 5.20 demonstrates how the attributes that are mapped from the Temporal DDP become the column headers of the table time. The output of this iteration is the dimension table called time of the star schema as shown in Figure 5.21.

Figure 5.21 Time Dimension Table Created from Temporal DDP

In the example, the product is identified by its hard disk capacity in GB (hd_size), product identifier (family), clock cycle (cpu), and which production line is responsible for its production (line). The family,
line, cpu and hd_size attributes become the profile information and hence are classified as qualifier data of the object. The model_number is the name of the object as the product is identified from model_number. Figure 5.22 shows how the questions help gather the attributes of object dimension table.

Figure 5.22 Object DDP

When the button create dimension table is clicked, a CREATE table command is fired that creates dimension table called “product” with product_key as the primary key. Figure 5.23 demonstrates how the attributes that are mapped from the Object DDP become the column headers of the table product.

Figure 5.23 Mapping Object DDP to Product Dimension Table
The output of this iteration is the dimension table called *product* of the star schema as shown in Figure 5.24.

![Figure 5.24 Dimension Table Product Created from Object DDP](image)

The application has the ability to loop through the same page again. So when the user realizes that there is another object dimension present, he or she proceeds to map the object DDP to create *production_line* dimension table. Figure 5.25 demonstrates how the questions help gather the attributes of object dimension table.

![Figure 5.25 Object DDP](image)

Figure 5.26 shows the dimension table *production_line* that is created for second star schema. The attributes of the dimension table is same as that of the *production_line* for the star schema with *activity_facts* fact table.
Figure 5.26 Mapping Object DDP to *production_line* Dimension Table

The output of this iteration is the dimension table called *production_line* of the star schema as shown in Figure 5.27

Figure 5.27 Dimension Table *production_line* Created from Object DDP

Once the user gathers entire object dimensions for that star schema, he or she can proceed to the next type of DDP by clicking on *proceed to next dimension* button. The next step is to create a fact table *production_facts*.

The name of the fact table is given as *production_facts* and *units_produced_qty* is added as the measure of the fact table, as shown in Figure 5.28.
After the create fact table button is clicked, a fact table that includes the primary keys of all the dimension tables determined for this star schema and the measures is created. Figure 5.29 gives the production_fact table definition that is automatically created by our software.

The output of this iteration is the fact table called production_fact of the star schema, shown in Figure 5.30.
Figure 5.31 gives second star schema that is created at the end of the second iteration of software execution.

![Figure 5.31 Star Schema of Production output design](image)

**Figure 5.31 Star Schema of Production output design**

5.2.3 Generating a Third Schema

The user is directed to a page where he or she is given two options of conforming dimensions or creating a new star schema, as shown in Figure 5.32. After create another star schema button is clicked, the application loops in a similar way as in the previous execution of the software.

![Figure 5.32 Multiple Star Schema Generation](image)

**Figure 5.32 Multiple Star Schema Generation**

The user determines the appropriate dimensions necessary for star schema. The two physical object dimensions are (product) and (component), conceptual object dimension (production_line) and temporal dimension (time) are selected as shown in Figure 5.33. The product and components dimensions are classified as physical object dimension. Production_line is not a physical object hence it is classified as conceptual object dimension. Accordingly the appropriate dimensions are selected.
Figure 5.33 Define All the Dimensions Required for Star Schema

The time, production_line, product dimensions are created in similar way as for the previous two star schemas. The object dimension, component, has component_part_number, unit_measure and component_description as the profile information of the object component. The category determines the role of the component. Figure 5.34 illustrates how the Object DDP helps gather the attributes for developing object dimension table component.

Figure 5.34 Object DDP

Figure 5.35 illustrates the component table definition with component_key as the primary key that is automatically created by our software.
The output of this iteration is the dimension table called \textit{component} of the star schema, shown in Figure 5.36.

Once all the dimensions are created the user proceeds to create the fact table. The fact table of third star schema has \textit{usage\_quantity} as measure that is added in addition to foreign keys of the fact table. Figure 5.37 shows the mapping of the measures of fact table \textit{component\_use\_facts}.
What is the area or topic of the data you wish to analyze? This name will be used for the central fact table of the generated schema. Example: order_fact, shipping_fact

component_use_facts

What are the general data items that you wish to investigate? Examples: sales_amount or quantity_on_hand

usage_quantity

Figure 5.37 Mapping of Fact Table from Questionnaire

Figure 5.38 gives the component_use_facts table definition that is automatically created by our software.

component_use_facts

Figure 5.38 production_fact Fact Table Definition

The output of this iteration is the dimension table called component of the star schema, shown in Figure 5.39.

component_use_fact

Figure 5.39 component_use_facts Fact Table
The star schema that the user generates is given in Figure 5.40.

At this stage when the user is directed to the page, where he or she can choose conform dimensions if all star schemas are created, shown in Figure 5.41.

When the Report Conformed Dimensions button is clicked, stored procedure conform() is executed and all the repeating dimensions are given as the output. The store procedure conform() compares and matches all the dimensions present in the database. It examines every dimension of one star schema and compares with the dimensions of all other existing star schemas. It outputs only those dimensions that completely match with one other. Figure 5.42 gives the output of candidates of conformed dimensions.
5.3 Conclusion

In this chapter we discussed the features of SQL Server 2005, ASP.NET 3.5, VB.NET and Visual Studio 2008 used in a software development effort and illustrated the execution of our software. In the next chapter we analyze the results of our software, summarize contributions of our work, and discuss ideas for which our software provides a foundation for future investigation.
Chapter 6 Contributions and Future Work

In this chapter we summarize the objectives met by this thesis. Section 6.1 gives an overview of the work accomplished. Section 6.2 gives the evaluation of the thesis. Finally 6.3 give ideas for future work.

6.1 Overview

The research contributions are as follows.

1. This thesis surveys different methods of designing a data warehouse. The two most important methods are a requirement-driven approach and a data-driven approach. We select the requirement-driven approach for designing a data warehouse.

2. We choose the approach proposed by Jones and Song [JS05, JS07]. Their approach is based on using DDPs (Dimensional Design Patterns) to design a star schema.

3. We implement an algorithm based on DDPs to capture the requirements to develop a data warehouse schema.

4. With the use of our software to design a star schema, there is no need for a designer to be present with the user. The software developed for this thesis consists of questions posed to the user. The questions do not contain any technical jargon so that a non-technical person may use it. The user should, however, be knowledgeable of the domain to be modeled.

5. We demonstrate a step by step implementation of our software. We show how the mapping of DDPs into a questionnaire is performed, and how this questionnaire can be automated to gather attributes to form a star schema. The star schema is created in Microsoft’s SQL Server 2005. The GUI is designed using ASP.NET 3.5 and VB.Net in Visual Studio 2008.

6. We further show how our software can be used to create one or more star schemas.

7. We extend the work of Jones and Song by reporting conformed dimensions among multiple schemas.

6.2 Evaluation

Jones and Song conducted a study with six different types of DDPs. They formed two groups of data warehouse designers. One group used DDPs to create a star schema and other group gathered requirements without using DDPs. Their experiment was to explore the effectiveness and accuracy of the use of DDPs. Although our thesis involves the use of DDPs for generating star schemas, the objectives of our work differ from those of Jones and Song. In this section we discuss the evaluation of our thesis.

Jones and Song conducted their work manually while we automate our algorithm in a software tool. Our software was used in implementing star schemas of various domains [AV98] such as production design, balance sheet summary design, supplier performance, and retailer’s income statement design. DDPs were used to automatically map to dimension and fact tables of a star schema.

The algorithm implements a questionnaire in the form of a web application that is useful for non-technical users along with the technical users. The questions asked do not contain technical jargon, and at times are explained with an example. This helps the user who is a non-technical person and a novice in data warehouse design to create a star schema. The user can independently run this software without being a data warehouse designer.
This software can be used as a first step or a platform towards designing a data warehouse. The user can review and revise the table content, data types or the table names generated by the software.

Another extension of the work of Jones and Song is conforming dimensions across multiple star schemas. We identify and report the conformed dimensions to the user. We do not merge the multiple star schemas to create a star schema with multiple fact tables and sharing dimensions. This report would be useful for the user as a basis to merge multiple star schemas.

6.3 Future Work

The data warehouse schema that is generated by our software has a star schema as the underlying logical model. A data warehouse can be represented in other conceptual models such as the StarER model [TBC99], DFM [GMR98] (Dimensional Fact Model), and Snowflake [CD97], to name few. This thesis can be extended for future work to implement the software using DDPs to create schemas represented in other conceptual models.

Another direction for future work would be to address schema merging in the case of conformed dimensions and multiple fact table star schemas. The dimensions that occur over multiple star schemas can be used to merge several star schemas. This helps in drilling information across several data marts [K05] and can be used for analytical purposes.

Jones and Song [JS07] extend their work by developing a methodology to design the fact table using a Fact DDP. The Fact DDP consists of measures and degenerate keys. Kimball defines a degenerate key as a unique identifying attribute that is associated with the transaction system that originally generated the data [KR02]. Our work can be extended in future work to use the Fact DDP when creating data warehouse schemas.
References:


Appendix A  Algorithm for Designing the Dimension Tables of Star Schema

Algorithm A.1 Create Location Dimension

1: **Input**: responses // answers to questions
column header names // text string to be used in generating fields in the dimension tables
table name // text string to be used to name the dimension table
2: **Output**: one or more one star schemas
3:  repeat
4:       if user selects Location Dimension then
5:         if user selects specific location then prompt for the following attributes
6:           identify street address
7:           identify city
8:           identify state
9:           identify zip code
10:          identify country
11:          identify County
12:          identify township
13:          identify province
14:          identify region
15:         end if
16:         if user selects specific work accomplished/purpose then prompt for the following attributes
17:           identify name/title
18:           identify definition
19:           identify primary work accomplished
20:         end if
21:       if user selects date/time with respect to Location dimension then prompt for time dimension attributes with respect to Location dimension
22:       if user selects organization or role with respect to Location dimension then prompt for Action dimension attributes with respect to Location dimension
23:       if user selects conceptual or physical object with respect to Location dimension then prompt for Object dimension attributes with respect to Location dimension
24:       if user selects qualifier information with respect to Location dimension then prompt for Qualifier dimension attributes with respect to Location dimension
25:       end if
26:          Input from the user the Location dimension table name
27:          Generate a primary key using the table name
28:          until user indicates there are no additional dimensions

Algorithm A.2 Create Stakeholder Dimension

1: **Input**: responses // answers to questions
column header names // text string to be used in generating fields in the dimension tables
table name // text string to be used to name the dimension table
2: **Output**: one or more one star schemas
3:  repeat
if user selects Stakeholder Dimension then
  if user selects specific organization then prompt for the following attributes
    identify name
    identify head of organization
    identify size
    identify role
    identify allocated budget
    identify responsibility
  end if
  if user selects role then prompt for the following attributes
    identify title of role
    identify role description
    identify level of education
    identify salary
    identify years of experience
  end if
  if user selects role then prompt for the following attributes
    identify name/title
    identify definition
    identify primary work accomplished
  end if
  if user selects date/time with respect to Stakeholder dimension then gather time dimension with Stakeholder dimension attributes from user
  if user selects specific events or actions with respect to Stakeholder dimension then gather Action dimension and Stakeholder dimension attributes from user
  if user selects locale with respect to Stakeholder dimension then gather Location dimension and Stakeholder dimension attributes from user
  if user selects object (conceptual/physical) with respect to Stakeholder dimension then gather Object dimension and Stakeholder dimension attributes from user
  if user selects qualifier information with respect to Stakeholder dimension then prompt for Qualifier dimension and Stakeholder dimension attributes from user
end if
Input from the user the Stakeholder dimension table name
Generate a primary key using the table name
until user indicates there are no additional dimensions

**Algorithm A.3 Create Action Dimension**

1: **Input**: responses // answers to questions
column header names // text string to be used in generating fields in the dimension tables
table name // text string to be used to name the dimension table
2: **Output**: one or more one star schemas
3: repeat
4: if user selects Action Dimension then
5:  if user selects specific action/event then prompt for the following attributes
6:  identify action
7:  identify any person in charge of the action
8:  identify no. of steps involved in action
9:  identify results or output of action
10: end if
11: if user selects date/time with respect to Action dimension
then gather time dimension with Action dimension attributes from user
12: if user selects specific locale with respect to Action dimension
then gather Location dimension with Action dimension attributes from user
13: if user selects organization or role with respect to Action dimension
then gather Stakeholder dimension with respect to Action Dimension attributes from user
14: if user selects object with respect to Action dimension
then gather Object dimension and Action dimension attributes from user
15: if user selects qualifier information with respect to Action dimension
then gather Qualifier dimension and Action Dimension attributes from user
end if
16: Input from the user the Action dimension table name
17: Generate a primary key using the table name
18: until user indicates there are no additional dimensions

Algorithm A.4 Create Object Dimension

1: Input: responses // answers to questions
column header names // text string to be used in generating fields in the dimension tables
table name // text string to be used to name the dimension table
2: Output: one or more one star schemas
3: repeat
4: if user selects Object Dimension then prompt for the following attributes
5: identify the classification of the object whether it is physical or conceptual
6: identify the lifespan of the object
7: identify ownership of the Object
8: if user selects date/time with respect to Object dimension present
then gather time dimension with Object dimension attributes from user
9: if user selects specific locale with respect to Object dimension
then gather Location dimension and Object dimension attributes from user
10: if user selects organization or role with respect to Object dimension
then gather Stakeholder dimension and Object Dimension attributes from user
11: if user selects qualifier information with respect to Object dimension
then gather Qualifier dimension and Object Dimension attributes from user
12: if user selects qualifier information with respect to Object dimension
then gather Qualifier dimension and Object Dimension attributes from user
end if
13: Input from the user the Object dimension table name
14: Generate a primary key using the table name
15: Until user indicates there are no additional dimensions
Algorithm A.5 *Create Qualifier Dimension*

1: **Input:** responses // answers to questions
column header names // text string to be used in generating fields in the dimension tables
table name // text string to be used to name the dimension table

2: **Output:** one or more one star schemas

3: repeat

4: If user selects *Qualifier* Dimension present then prompt for the following attributes

5: identify if profile information present

6: identify if State (particular condition) present

7: identify if cause (reason) present

8: identify if Unit (measurement) information present

9: identify if contact information present

10: end if

11: Input from the user the *Qualifier* dimension table name

12: Generate a primary key using the table name

13: Until user indicates there are no additional dimensions
Appendix B  Case Studies

We perform case studies to demonstrate how the DDPs and our software tool are applied to various domains.

Section B.1  Generate Star Schema of Order and Shipments [AV98]

Figure B.1 date Table Definition
Input for Stakeholder dimension

What would you like to call your Stakeholder dimension? This name will be a table in the final data warehouse design.

If stakeholder domain present? Yes [ ]

Which are the ways that you would like to summarize your data?

Check if specific organization present [ ]

- any organization involved [ ]
- presence of any head of organization [ ]
- size [ ]
- allocated budget [ ]
- responsibility [ ]

Check if any role present? [ ]

- title of role [ ]
- description of the role [ ]
- level of education [ ]
- salary [ ]
- years of experience [ ]

Check if any temporal data with respect to stakeholder present? [ ]

Check if any location data with respect to stakeholder present? Yes [ ]

- billing_zip [ ]
- billing_city [ ]
- billing_state [ ]
- shipping_address [ ]

Check if any qualifier data with respect to stakeholder present? Yes [ ]

- name [ ]

Check if any physical/conceptual object data stakeholder present? [ ]

- create dimension table [ ]
- proceed to next dimension [ ]

---

**Figure B.2 customer Dimension Table**
Figure B.3 customer Table Definition
What would you like to call your *Stakeholder* dimension? This name will be a table in the final data warehouse design stakeholder

If stakeholder domain present?  Yes ☐

Which are the ways that you would like to summarize your data?

Check if specific organization present  Yes ☐

- any organization involved ☐
- presence of any head of organization ☐
- size ☐
- allocated budget ☐
- responsibility ☐

Check if any role present?  Yes ☐

- title of role ☐
- description of the role ☐
- level of education ☐
- salary ☐
- years of experience ☐

Check if any temporal data with respect to stakeholder present?  Yes ☐

Check if any action data with respect to stakeholder present?  Yes ☐

Check if any location data with respect to stakeholder present?  Yes ☐

- territory_name ☐
- region_name ☐

Check if any qualifier data with respect to stakeholder present?  Yes ☐

- salesperson_name ☐
- salesperson_code ☐

Check if any physical/conceptual object data stakeholder present?  Yes ☐

- create dimension table ☐
- proceed to next dimension ☐

Figure B.4 *salesperson* Dimension Table
Figure B.5 *salesperson* Table Definition

Figure B.6 *product* Dimension Table
Figure B.7 *product* Table Definition

Figure B.8 *order_facts* Fact Table
Figure B.9 order_facts Table Definition

Figure B.10 Star Schema Generated for Order Process
Input for Stakeholder dimension

What would you like to call your Stakeholder dimension? This name will be a table in the final data warehouse design shipper

If stakeholder domain present? Yes

Which are the ways that you would like to summarize your data?

Check if specific organization present Yes

- any organization involved
- presence of any head of organization
- size
- allocated budget
- responsibility

Check if any role present? Yes

- title of role
- description of the role
- level of education
- salary
- years of experience

Check if any temporal data with respect to stakeholder present? Yes

Check if any action data with respect to stakeholder present? Yes

Check if any location data with respect to stakeholder present? Yes

- shipper_address
- shipper_city
- shipper_state
- shipper_zip

Check if any qualifier data with respect to stakeholder present? Yes

- shipper_name
- shipper_type

Check if any physical/conceptual object data stakeholder present? Yes

create dimension table   proceed to next dimension

Figure B.11 shipper Dimension Table
Figure B.12 *shipper* Table Definition

Figure B.13 Star Schema Generated for Shipment Process
Figure B.14 Report of Conformed Dimensions of Order and Shipments

Figure B.15 Star Schema of Order and Shipments [AV98]
Section B.2  Generate Star Schema of Supplier Performance Schema [AV98]

Input for the *Temporal* dimension

What would you like to call your *Temporal* dimension? This name will be a table in the final data warehouse design. Time_InspectionDate

Indicate the ways that you would like to summarize your data by checking boxes below.

Is Calendar period present?  
- century
- decade
- year
- quarter
- month name
- month number
- week
- week number
- day of week
- day of month
- date-time

Is Fiscal period present?  
- fiscal year
- fiscal quarter
- fiscal month
- fiscal week
- fiscal date/time

Is a Timestamp present?  
- hour
- minute
- second
- timestamp

Is a Special Period present?  

Please fill in the boxes below with what you would like to call your special periods in the final data warehouse design. For example, you may have an event that is called rain week.

<table>
<thead>
<tr>
<th>event name</th>
<th>weekend</th>
<th>audit period</th>
<th>holiday</th>
</tr>
</thead>
</table>

[create dimension table] [proceed to next dimension]

Figure B.17 *time* Dimension Table
### Table Definition

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Allow Nulls</th>
</tr>
</thead>
<tbody>
<tr>
<td>time_key</td>
<td>datetime</td>
<td></td>
</tr>
<tr>
<td>Month</td>
<td>datetime</td>
<td></td>
</tr>
<tr>
<td>Week</td>
<td>datetime</td>
<td></td>
</tr>
<tr>
<td>day_of_week</td>
<td>datetime</td>
<td></td>
</tr>
<tr>
<td>day_of_month</td>
<td>datetime</td>
<td></td>
</tr>
</tbody>
</table>

**Figure B.18** *time* Table Definition
Input for *Stakeholder* dimension

What would you like to call your *Stakeholder* dimension? This name will be a table in the final data warehouse design. 

supplier

If stakeholder domain present? Yes ☑

Which are the ways that you would like to summarize your data?

Check if specific organization present Yes ☑

- any organization involved
- presence of any head of organization
- size
- allocated budget
- responsibility

Check if any role present? Yes ☑

- title of role
- description of the role
- level of education
- salary
- years of experience

Check if any temporal data with respect to stakeholder present? Yes ☑

Check if any action data with respect to stakeholder present? Yes ☑

Check if any location data with respect to stakeholder present? Yes ☑

- city
- state/region
- country

Check if any qualifier data with respect to stakeholder present? Yes ☑

- name

Check if any physical/conceptual object data stakeholder present? Yes ☑

create dimension table  proceed to next dimension

Figure B.19 *supplier* Dimension Table
Figure B.20 supplier Table Definition

Figure B.21 shipment Dimension Table
Figure B.22 shipment dimension table
Input for the Qualifier dimension

What would you like to call your Qualifier dimension? This name would be used in generated schema any profile information present? ✓

name wood_type

characteristics grade

check if specific state? Yes ✓

identify condition form

name of circumstance

identify time

check if any specific cause? Yes ☑

identify name of cause

reason for cause?

any occurrence for event

any units used? Yes ✓

units used nomenclature

any measurements grain_rating

any configuration count_type

create dimension table proceed to creating Fact table

Figure B.23 material Dimension Table
**Figure B.24** material Table Definition

**Figure B.25** defect Dimension Table
Figure B.26 *defect* Table Definition

Figure B.27 *material_defect_facts* Fact Table
Figure B.28 `material_defect_facts` Table Definition

Figure B.29 Star Schema generated for Inspection process
What is the area or topic of the data you wish to analyze? This name will be used for the central fact table of the generated schema. Example: order_fact, shipping_fact

delivery_facts

What are the general data items that you wish to investigate? Examples: sales_amount or quantity_on_hand

delivery_quantity

create_fact_table

Figure B.30 Fact Table delivery_facts

Figure B.31 delivery_facts Fact Table Definition

Figure B.32 Star Schema Generated for Delivery Process
Figure B.31 Report of Conformed Dimensions

Figure B.32 Generated Star schema of Supplier Performance Schema [AV98]
Appendix C Program Code

```html
<body>
<form id="form1" runat="server">

  <div>
    Input for the <i>Temporal</i> dimension:<br />
    What would you like to call your <i>Temporal</i> dimension? This name will be a table in the final data warehouse design.<br />
    <asp:TextBox ID="DimDateTimeBox" runat="server"
                EnableViewstate="False"></asp:TextBox>
  </div>

  Indicate the ways that you would like to summarize your data by checking boxes below:<br />
  Is Calendar period present?<br />
  <asp:CheckBox ID="CheckBox1" runat="server" EnableViewstate="False" />
  <asp:CheckBox ID="CheckBox2" runat="server" EnableViewstate="False" />
  <asp:CheckBox ID="CheckBox3" runat="server" EnableViewstate="False" />

</form>
</body>
```

Figure C.1   Temporal DDP HTML code
Imports System
Imports System.Data
Imports System.Data.SqlClient
Imports System.Collections

Partial Class Input
Inherits System.Web.UI.Page

Protected Sub submit_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles submit.Click
    Dim CreateCmdString As String
    Dim CommandCreate As SqlCommand
    Dim SQLConnectionName As String
    Dim InsertString As String
    Dim CommandInsert As SqlCommand

    SQLConnectionName = New SqlConnection("Data Source=MOLAL-1FTP\SQLExpress;Integrated Security=True")
    SQLConnectionName.Open()

    Dim starnumber As Integer
    starnumber = Session("starnumber")

    Dim stringqueue As New Queue()
    Dim tablename As String
    tablename = DsBitmapBox.Text

    Dim pk As String
    pk = tablename & ".KEY"

    If Session("key") IsNot Nothing Then
        Dim arr As ArrayList = DirectCast(Session("key"), ArrayList)
        Session("key") = arr.Add(pk)
    End If

    If CheckBox1.Checked = True Then
        If CheckBox2.Checked = True Then
            stringqueue.Enqueue("Century")
        End If
        If CheckBox3.Checked = True Then
            stringqueue.Enqueue("Decade")
        End If
        If CheckBox4.Checked = True Then
            stringqueue.Enqueue("Year")
        End If
    End If

    CreateCmdString = "CREATE TABLE " + tablename + ")
    CreateCmdString += "pk + "datetime PRIMARY KEY"
    Try
        For i = 1 To 100
            CreateCmdString += "," + stringqueue.Dequeue() + " "
            CreateCmdString += "datetime"
        Next
    Catch
        CreateCmdString = CreateCmdString + ")"
    End Try

    CommandCreate = New SqlCommand(CreateCmdString, SQLConnectionName)
    CommandCreate.ExecuteNonQuery()

    Dim insertString As String
    Dim CommandInsert As SqlCommand
    insertString = "INSERT INTO metadata (tablename, pk, starnumber, isfact) VALUES (@tablename, @pk, @starnumber, 0)"
    CommandInsert = New SqlCommand(insertString, SQLConnectionName)
    CommandInsert.Parameters.AddWithValue("@tablename", tablename)
    CommandInsert.Parameters.AddWithValue("@pk", pk)
    CommandInsert.Parameters.AddWithValue("@starnumber", starnumber)
    CommandInsert.ExecuteNonQuery()

    SQLConnectionName.Close()
End Sub

Figure C.2   Vb.net Code to Create Temporal Dimension Table
Figure C.3 Fact Table Design Code

Figure C.4 Vb.net code to Create Fact Table
Figure C.5  Create More Star Schemas or Report Conformed Dimensions